Computational and Numerical Methods for Bioelectric Field Problems

Christopher R. Johnson
Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112
Email: crj@sci.utah.edu
Web: www.sci.utah.edu
Reprinted from *Critical Reviews in Biomedical Engineering*,
volume 25, number 1, pp. 1-81, 1997.

Abstract

Fundamental problems in electrophysiology can be studied by computationally modeling and simulating the associated microscopic and macroscopic bioelectric fields. To study such fields computationally, researchers have developed a number of numerical and computational techniques. Advances in computer architectures have allowed researchers to model increasingly complex biophysical system. Modeling such systems requires a researcher to apply a wide variety of computational and numerical methods to describe the underlying physics and physiology of the associated three-dimensional geometries. Issues naturally arise as to the accuracy and efficiency of such methods. In this paper we review computational and numerical methods for solving bioelectric field problems. The motivating applications represent a class of bioelectric field problems that arise in electrocardiography and electroencephalography.

Keywords

Bioelectric fields, computation, modeling, numerical techniques, simulation.

CONTENTS

LIST OF FIGURES

I. Introduction

Computer modeling and simulation continue to grow more important to the field of Bioengineering. The reasons for this growing importance are manyfold. First, mathematical modeling has been shown to be a substantial tool for the investigation of complex biophysical phenomena. Second, since the level of complexity one can model parallels existing hardware configurations, advances in computer architecture have made it feasible to apply the computational paradigm to complex biophysical systems. Hence, while biological complexity continues to outstrip the capabilities of even the largest computational systems, the computational methodology has taken hold in bioengineering and has been used successfully to suggest physiologically and clinically important scenarios and results.

Computational *electrophysiology* can be loosely defined as the simulation and modeling of macroscopic and microscopic bioelectric fields. The electrical activity in the human body is induced by the flow of charged ions across membranes of individual cells. The collective interaction of all the currents from groups of cells gives rise to electric potentials that can be measured from, for example, the scalp (EEG) or the torso surface (ECG). The registration of normal and abnormal bioelectrical signals has been used as a diagnostic tool for many years, and extensive catalogs linking signals to pathologies have evolved. More recently, investigators have begun to develop mathematical descriptions of the origins of the electric signals they measure. The hope exists that these mathematical descriptions will provide researchers in Bioengineering and Medicine with new diagnostic tools that are based on physiological function.

In the field of electrocardiography, computational models of bioelectric phenomenon from sources in the heart have existed for over 25 years. The size and scope of these models have been limited by contemporary computational resources and by the numerical algorithms utilized to approximate the continuous field equations. It has been shown that the electric signals in the body produced by the macroscopically viewed heart can be described as a solution to a quasistatic Poisson's equation [1]. While the analytic solutions of such elliptic partial differential equations are not difficult to achieve for simple geometries such as spheres and cylinders, difficulties arise when considering the complex geometries associated with physiological structures. Realistic geometries pose significant challenges to researchers trying to accurately approximate the bioelectric fields within them. Such challenges include the construction of the geometrical model; the specification of the material properties, some of which are anisotropic; the numerical approximation of the biophysical field equations; and the large-scale nature of the computations.

This paper provides an overview of computational and numerical techniques that can be applied to a class of bioelectric field problems. Bioelectric field problems are found in a wide variety of biomedical applications that range from single cells [2], to organs [3], up to models that incorporate partial to full human structures [4], [5], [6]. We describe some general modeling techniques that will be applicable, in part, to all the aforementioned applications. We focus the review on numerical techniques applicable to bioelectric volume conductor problems that arise in electrocardiography and electroencephalography.

The paper is broken up into the following sections that correspond to the modeling and simulation process. We start by giving a brief statement of the mathematical formulation for a bioelectric volume conductor, continue by describing the model construction process, and follow with sections on numerical solutions and computational considerations, including a section on error analysis coupled with a brief introduction to adaptive methods. We conclude with a section on available on-line software that provide resources useful to biomedical modelers.

II. Problem Formulation

As noted in the classic works of Barr and Plonsey [1], [7], [8], [9], and of Geselowitz [10], [11] most bioelectric field problems can be formulated in terms of either the Poisson or the Laplace equation for electrical conduction. Since Laplace's equation is the homogeneous counterpart of the Poisson equation, we will develop the treatment for a general three-dimensional Poisson problem and discuss simplifications and special cases when necessary. For our discussion, we are interested in the properties of a volume conductor as it pertains to the evaluation of electric fields therein. A bioelectric field is a manifestation of a current density I_V. Bioelectric sources can arise from a variety of endogenous (propagating action potentials, etc.) and applied (defibrillation electrodes, etc.) conditions. Our treatment assumes that the current density is known (excellent treatments of the mathematical and numerical solution of models based on membrane currents can be found in two earlier Critical Review in Biomedical Engineering articles [12], [13]) and we focus our attention on numerically approximating the electric and potential fields within a three-dimensional volume conductor. A volume conductor can be defined as a continuous passive conducting medium that surrounds the region

occupied by the source(s) I_V [9], [10]. The temporal behavior of bioelectric sources is found within the low-frequency range (usually under 1 kHz). As noted by [14], [9], examination of bioelectric fields in regions with typical physiologic conductivities and frequencies under 1 kHz show that *quasi-static* conditions apply. Therefore, at a given time, we can neglect the displacement current such that Ohm's law,

$$\mathbf{I} = \sigma \mathbf{E} + I_V, \tag{1}$$

describes the source-field relationship within the volume conductor. Here, $\mathbf{I}$ represents the total current in the system, a sum of the conduction current σE, and the source currents I_V. Noting that the scalar potential is related to the electric field by virtue of

$$\mathbf{E} = -\nabla \Phi, \tag{2}$$

and noting that, due to the quasi-static nature, $\mathbf{I}$ must be solenoidal (i.e. the divergence of the total current is zero), we arrive at a mathematical description of a *typical* bioelectric volume conductor problem:

$$\nabla \cdot \sigma \nabla \Phi = -I_V \quad \text{in } \Omega. \tag{3}$$

Here Φ is the electrostatic potential, σ is the electrical conductivity tensor, and I_V is the current per unit volume defined within the solution domain, Ω. The associated boundary conditions depend on what type of problem one wishes to solve.

There are generally considered to be two different types of volume conductor problems. One type of problem deals with the interplay between the description of the bioelectric volume source currents I_V with the resulting volume currents as well as volume and surface voltages. Here, the problem statement would be to solve (3) for Φ with a known description of I_V and the Neumann boundary condition:

$$\sigma \nabla \Phi \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_T, \tag{4}$$

which says that the normal component of the electric field is zero on the surface interfacing with air (here denoted by Γ_T). This problem can be used to solve two well known problems in medicine, the *direct* (or *forward*) EEG (electroencephalography) and ECG (electrocardiography) volume conductor problems. In the direct EEG problem, one usually discretizes the brain and surrounding tissue and skull. One then assumes a description of the bioelectric current source within the brain (this usually takes the form of dipoles or multipoles) and calculates the fields within the brain and on the surface of the scalp. Similarly, in one version of the direct ECG problem, one utilizes descriptions of the current sources in the heart (either dipoles, dipole layers, or membrane current source models such as the FitzHugh Nagumo and Beeler Reuter, among others) and calculates the currents and voltages within the volume conductor of the chest and voltages on the surface of the torso.

The *inverse* problems associated with these direct problems involve estimating the current sources I_V within the volume conductor from measurements of voltages on the surface of either the head or body [15], [16]. Thus, one would solve (3) with the boundary conditions:

$$\Phi = \Phi_0 \quad \text{on } \Sigma \subseteq \Gamma_T \tag{5}$$

and

$$\sigma \nabla \Phi \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_T. \tag{6}$$

The first boundary condition is the Dirichlet condition, which says that one has a set of discrete measurements of the voltage on a subset of the outer surface ($\Sigma \subseteq \Gamma_T$). The second is the natural Neumann condition. While they don't look much different than the formulation of the direct problem, these inverse formulations are ill-posed. The bioelectric inverse problem in terms of primary current sources does not have a unique solution, and the solution doesn't depend continuously on the data. Thus, to obtain *useful* solutions, one must restrict the solution domain (i.e. number of physiologically plausible solutions) [17] for the former case, and apply so-called *regularization* techniques to attempt to restore the continuity of the solution on the data in the latter case.

The second general type of bioelectric direct/inverse formulation poses both problems in terms of scalar values at the surface boundaries. For this version of EEG problem, one would take the surface of the brain (cortex) as one bounded surface and the surface of the scalp as the other surface. The direct problem would involve making measurements of

voltage of the surface of the cortex at discrete locations and then calculating the voltages on the surface of the scalp. Similarly, for the ECG problem, voltages could be measured on the surface of the heart and used to calculate the voltages at the surface of the torso, as well as within the volume conductor of the thorax. To formulate the inverse problems, one uses measurements on the surface of the scalp (torso) to calculate the voltages on the surface of the cortex (heart). In this formulation, because we are interested in the distributions of voltages on a surface instead of current sources within a volume, we solve Laplace's equation rather than Poisson's equation. This leads to the following boundary value problem:

$$\nabla \cdot \sigma \nabla \Phi = 0 \qquad \text{in } \Omega \tag{7}$$

subject to the boundary conditions

$$\Phi = \Phi_0 \qquad \text{on } \Sigma \subseteq \Gamma_T \tag{8}$$

and

$$\sigma \nabla \Phi \cdot \mathbf{n} = 0 \qquad \text{on } \Gamma_T. \tag{9}$$

The approximation of defibrillation fields is another direct problem that can be characterized by Laplace's equation [18], [19], [20], [21], [22], [23], [24]. The defibrillation problem involves application of exogenous current/voltage sources either externally on the body surface [25] or internally within the thorax cavity [26]. Clinically, these sources (electrodes) are used to deliver sufficient electric energy to stop the irregular heart rhythms that signify a fibrillating heart.

In the past, the placement of the electrodes for either external or internal defibrillators was chosen based on clinical trial and error. Only recently has the sophistication of thorax and heart models permitted more realistic simulation of cardiac defibrillation. The aim of such models is to assist in determining the optimum electrode placement and strength of shock to terminate the fibrillation [27], [28], [21], [29], [30], [31], [32]. The simulations involved in this case present similar computational challenges, since, once again, the geometry of the torso is complex, the material properties (electrical conductivities) are imperfectly known, and the resulting computations are extensive. Defibrillation gives rise to very large potential gradients near the edge of the electrodes. As the effectiveness of the simulations depends on accurately calculating the amount of current that passes from the electrodes to the heart, these gradients need to be computed accurately. Since the strength of the gradients falls off as $\frac{1}{r^2}$ from the source, special numerical techniques (discussed in the Adaptive Methods section) are necessary for accurately computing the fields on and near the electrodes while still controlling the size of the resulting computation.

Mathematically, the defibrillation problem can be posed as solving Laplace's equation with boundary conditions similar to those in (7), replacing *a priori* knowledge of voltage on the heart boundary by that on the region of the torso under the defibrillation electrodes, that is, $\Phi = \Phi_0$ on $\Sigma \subset \Gamma_T$. Here, the sources are due to defibrillation pulses and the goal is to determine the resulting current density throughout the heart volume. The current density $\mathbf{J}$ can be calculated according to:

$$\mathbf{J} = -\sigma \nabla \Phi. \tag{10}$$

Continuity conditions of the normal component of current density and potential hold at each of the conductivity interfaces.

For the formulation of bioelectric fields in terms of Laplace's equation, the solution to the inverse problem is unique [33]; however, there still exists the problem of ensuring continuity of the solution onto the data. The linear algebra counterpart to the elliptic boundary value problem is often useful in discussing this problem of noncontinuity. The numerical solution to all elliptic boundary value problems (such as the Poisson and Laplace problems) can be formulated in terms of a set of linear equations, $\mathbf{A\Phi} = \mathbf{b}$, where Φ is the vector of voltages, $\mathbf{b}$ is the vector of current sources and/or contributions from the Dirichlet boundary condition, and $\mathbf{A}$ is the matrix that contains all the information pertaining to the model geometry and material properties. For the solution of Laplace's equation, the system can be reformulated as [34]:

$$A\Phi_{in} = \Phi_{out}, \tag{11}$$

where Φ_{in} is the vector of data on the inner surface bounding the solution domain (the electrostatic potentials on the scalp or heart, for example), Φ_{out} is the vector of data that bounds the outer surface (the subset of voltage values on the surface of the cortex or torso, for example), and A is the *transfer matrix* between Φ_{out} and Φ_{in}. This transfer matrix contains the geometry and physical properties (conductivities, dielectric constants, etc.) of the volume conductor. The

direct problem is then simply (well) posed as solving (11) for Φ_{out} given Φ_{in}. Similarly, the inverse problem is to determine Φ_{in} given Φ_{out}.

A characteristic of A for discrete ill-posed problems[1] is that it has a very large condition number. Briefly, the condition number is defined as $\kappa(A) = \|A\| \cdot \|A^{-1}\|$, or the ratio of maximum to minimum singular values as measured in the L_2 norm [36]. If the singular values of A gradually decay to zero and the ratio between the largest and smallest nonzero singular value is large, then the problem is characterized as ill-posed. The condition of A is a measure of the sensitivity of the relative error in the solution to changes in the right-hand side of (11). The ideal problem conditioning occurs for orthogonal matrices that have $\kappa(A) \approx 1$, while an ill-conditioned matrix will have $\kappa(A) \gg 1$. When one inverts a matrix that has a very large condition number (i.e. solves the linear system), the solution process is unstable and is highly susceptible to errors. For ill-conditioned problems, small relative errors in the input data can cause large relative errors in the results. The condition of a matrix also depends on the precision level of computations and is a function of the size of the problem. For example, if the condition number exceeds a linear growth rate with respect to the size of the problem, the problem will become increasingly ill-conditioned. See [36], [37] for more about the condition number of matrices.

A number of *regularization* techniques have arisen to deal with the ill-posed nature of many inverse problems. Formulations involving integral equations have been used extensively for continuous versions of ill-posed problems [38], while linear algebraic formulations have been used for the discrete counterpart. For bioelectric field inverse problems, most researchers have tended to use discrete formulations; however, some have tried to tackle the problem of ill-posedness at the onset by reformulating the original problem [17].

Although many types of regularization approaches are possible, the dominating approach is to require that the 2-norm (or semi-norm) of the solution is small. An additional side constraint consisting of an initial estimate Φ_{in}^* of the solution may also be included in the regularization. One of the most commonly used regularization methods is known as Tikhonov regularization. Here, the idea is to define the regularized solution Φ_α of (11) as minimizing a combination of the residual norm and the side constraint

$$\Phi_\alpha = argmin\|A\Phi_{in} - \Phi_{out}\|_2^2 + \alpha^2\|L(\Phi_{in} - \Phi_{in}^*)\|_2^2. \tag{12}$$

The matrix L is typically either the identity matrix or a discrete approximation of the derivative operator in the original boundary value problem (often either the gradient or laplacian operator for bioelectric field problems). The *regularization parameter* α controls the weight given to minimization of the residual norm relative to the side constraint. The regularization parameter controls the sensitivity of the regularized solution Φ_α to perturbations in A and Φ_{out}; thus, the regularization parameter is an important quantity and should be chosen with care [35].

Besides Tikhonov regularization, discrete regularization techniques include truncated singular value decomposition (TSVD), generalized singular value decomposition (GSVD), maximum entropy, and a number of generalized least squares schemes, including Twomey and variants of Tikhonov methods [39], [40], [41], [42], [35]. All of the previously mentioned methods share a common thread, in that these methods try to reduce the effects of solving an ill-conditioned system by restoring continuity of the solution onto the data. Most of these methods employ operations that try to balance the amount of smoothing with the fidelity of the data by adjustment of the regularization parameter [35]. Recent work in inverse electrocardiography by Brooks et. al. has focused on methods to improve the *a priori* estimation of the regularization parameter [43], [44], [**?**].

For a more in-depth treatment of ill-posed inverse problems, the reader is referred to [39], [40], [42], [35]. For specific information on electrocardiographic inverse problems, see the review articles [45], [46]. A particularly useful reference for discrete ill-posed problems is the Matlab [**?**] package developed by Per Christian Hansen, which is freely available via netlib [47].

A. Bidomain Model

Before moving on to the section on model construction and mesh generation, we briefly discuss the case when volume conductor models are not sufficient to describe the underlying biophysical properties. The reduced mathematical

[1]Strictly speaking, ill-posed problems must be infinite-dimensional. However, certain finite-dimensional discrete problems exhibit similar characteristics such as being sensitive to high-frequency perturbations. The term *discrete ill-posed problem* is routinely given to such problems [35].

formulation of the biophysical fields characterized by (3) is not sufficient to model current flow in the immediate region of the cardiac sources. Here, the microscopic properties of the tissue must be considered [12], [13], [2], [48]. The complex microstructure of cardiac muscle, comprised of coupled cells within an intersitium made up connective tissue, fluid, and vessels, requires that propagation of membrane potentials be accounted for within a model. One approach that has gained in favor in recent years views the cardiac tissue as two coupled, continuous domains: one for intercellular space and the other for the interstitial space. The intracellular potential Φ_i and extracellular potential Φ_e [12], [13], [2] obey the equations

$$\nabla \cdot (\mathbf{g}_i \cdot \nabla \Phi_i) = I_V + f_{is} \tag{13}$$

and

$$\nabla \cdot (\mathbf{g}_e \cdot \nabla \Phi_e) = -I_V + f_{es}, \tag{14}$$

where f_{is} and f_{es} are externally applied currents in the intracellular and the interstitial domains. While potentials and currents are averaged in both domains, the structure is partially preserved by assigning a conductivity tensor at each point in space, given by $\mathbf{g}_i$ and $\mathbf{g}_e$, the intracellular and interstitial effective macroscopic conductivity tensors. The treatment of cardiac tissue as two coupled continua is referred to as the *bidomain model* [12]. As mentioned previously, we will restrict our treatment to bioelectric phenomena related to volume conductor problems rather than propagation models. The reader is referred to the thorough mathematical treatment of the bidomain model recently presented by Henriquez [12].

B. Bioelectric Field Application Areas

We end this section by summarizing the various application areas in which the numerical solutions of bioelectric field problems play a role. We will not attempt a detailed discussion of any of the specific applications, but instead point the interested reader to relevant review articles in each field. We note that the numerical techniques discussed henceforth are pertinent to all the applications summarized below.

A. Electrocardiology There is rich history of the use of computer simulation to solve the forward and inverse problems in electrocardiography [49], [50], [16], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76]. An example of a solution of the forward problem within a large-scale thorax model is shown in Figure 1 from the author [77]. An example of the activation sequence of the heart estimated by using the uniform double layer inverse model is shown in Figure 2 by Huiskamp et al. [49], [78]. An example of a large-scale heart model is shown in Figure 3 by Nielsen and Hunter et al. [79], [80]. One of the largest collections of review articles regarding electrocardiology, including numerical methods, can be found in the three-volume "Comprehensive Electrocardiology," edited by Macfarlane and Lawrie [81]. Also see the following articles that appeared in the *Critical Reviews in Biomedical Engineering* [45], [12], [13], [46], [2].

B. Cardiac Defibrillation The use of external and implantable cardioverter defibrillators has proven to be effective in reducing the mortality rate of those individuals afflicted with heart disease. The improvement in lead technology so far has progressed via two distinctly different paths. Manually constructing lead systems and testing the configurations in animals was, and still is, the development path that the majority of ICD designers use. Now, with the sophistication of computer simulations and the increased understanding of the critical variables necessary for successful defibrillation, ICD designers are beginning to model and test new electrode configurations on computers before verifying the results in animal studies [24], [27], [31], [32]. Many studies have been performed demonstrating the power and sophistication of computer simulations for defibrillation studies. References for computational cardiac defibrillation studies include [82], [83], [84], [85], [86], [87], [88], [89], [90], [29], [30], [91], [92].

C. Electrical Impedance Tomography Electrical impedance tomography (EIT) is a procedure for mapping electrical conductivity properties of the internal tissues by applying electrical current through electrodes attached to the surface of the body and measuring the resulting voltages. Besides providing a map of the conductivity inside the body, which is important for many bioelectricity modeling studies, EIT could be used to assist in monitoring pulmonary problems such as edema, to noninvasively monitor cardiac output, or even to measure local internal temperature increases associated with hyperthermia treatments. References to modeling studies in EIT include [93], [94], [95], [96], [97].

D. Therapeutic and Functional Electrical Stimulation Electrical stimulation of the nervous system by external and/or implantable electrodes has been used to treat symptoms of epilepsy, psychiatric disorders, and spinal cord injury, among many other disorders. Most bioelectric models for therapeutic and functional electrical stimulation are concerned

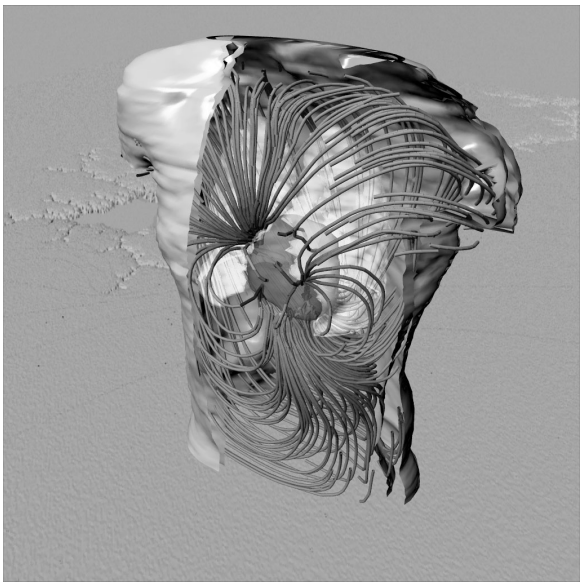

Fig. 1. A visualization of the geometry and electrical current flow in a model of the human thorax for a single time step in the cardiac cycle. From Johnson.

with delivering a therapeutic dose of electricity to the desired region(s) while minimizing the stimulation effects to surrounding regions. References pertaining to electrical stimulation include [98], [99], [100], [2], [101], [102].

E. Electromyography Electromyography was defined by Buchthal [103] as "...the registration of muscle action potentials." According to Buchthal, these potentials "give information as to the state of the muscle and indicate the activity of the motor neurons in reflex and voluntary contraction. Deviation from the normal electromyogram can contribute to the differential diagnosis of disorders of the motor system. Of diagnostic importance are the spontaneous electrical activity in an apparently relaxed muscle as well as the electrical activity which accompanies voluntary or reflex contraction and the action potentials evoked by electrical stimulation of muscle and nerve." Until recently, EMGs were recorded primarily for diagnostic purposes. From biotechnology advancements during the past decade, electromyograms have also become a tool in monitoring and controlling movement in artificial limbs and are used in conjunction with functional electrical stimulation [104]. Modeling work of EMG electrodes can be found in [105]. Additional references pertaining to electromyography include [106], [103], [104].

F. Electroencephalography One of the fundamental problems in computational electroencephalography is the inverse EEG problem. Examples of large-scale head models are shown in Figures 4 from Weinstein [107] and 5 by Oostendorp. If accurate solutions to the inverse EEG problem could be obtained, neurologists would be able to noninvasively view and interpret patient-specific cortical activity, gaining data important to surgeons and researchers. For example, accurate focal source localization is of paramount importance for surgically treating epileptic patients. An accurate inverse EEG solution would allow surgeons to dramatically reduce the amount of brain tissue removed during neurosurgery, as the anomalous regions responsible for the electric discharges triggering the seizures could be pin-pointed rather than merely estimated. A rapid, accurate inverse solution could also be used to guide time-critical functional imaging techniques such as fMRI. Such a tool would be useful for observing patients over extended periods, as is often necessary for patients with sleep disorders and multiple sclerosis. Finally, theories and correlation data suggest that left temporal lobe volume reduction in schizophrenic patients is responsible for their measured P300 topographic asymmetries. This asymmetry is the most robust and consistently replicated electrophysiological abnormality observed in schizophrenic patients, and has to do with the ERP (event-related potential) measured on the patient's scalp approximately 300 milliseconds after an auditory stimulus. These are only a few of the potential benefits that could be provided by solutions to the inverse EEG problem. References on computational electroencephalography include [108], [109], [110], [111], [107], [112], [113], [114], [115], [116], [117], [118].

Beyond the challenges associated with complicated geometries and ill-conditioned inverse problems, bioelectric field problems often present substantial computational difficulties because of the large number of variables needed to adequately describe the underlying physics and physiology. We will address these challenges in section IV. Once we

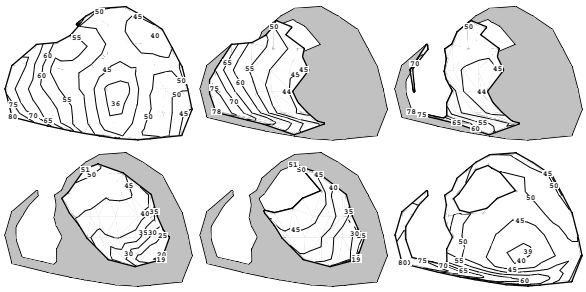

Fig. 2. The activation sequence of the heart estimed by using the uniform double layer model. Isochrones are drawn at 5 ms intervals. The upper panels show from left to right the anterior epicardium, the right ventricular endocardium, and the right ventricular septum (all in frontal view). The lowers panels show the left ventricular septum, the left ventricular endocardium and the posterior epicardium. From Huiskamp et al.

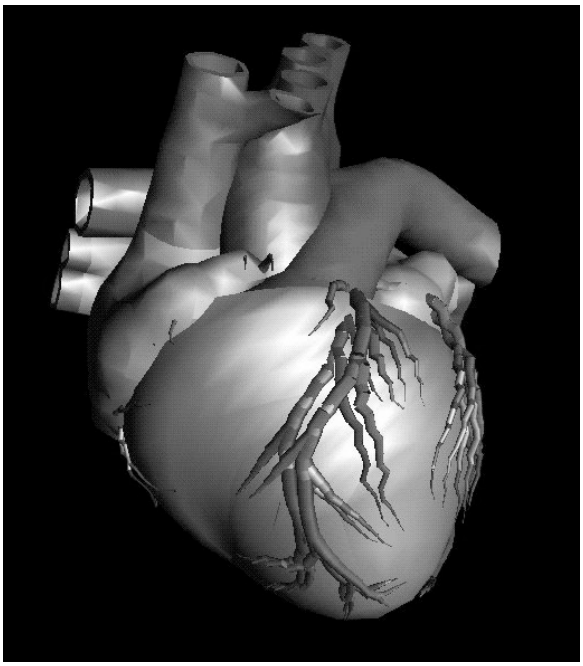

Fig. 3. A large-scale heart model. Coronary arteries and veins shown on the epicardial surface of complete heart model. From Nielsen and Hunter et al.

have stated or derived the mathematical equations that define the physics and physiology of the system, we need to create a discrete version of the geometry describing the volume conductor. This leads us to the next section on model construction and mesh generation.

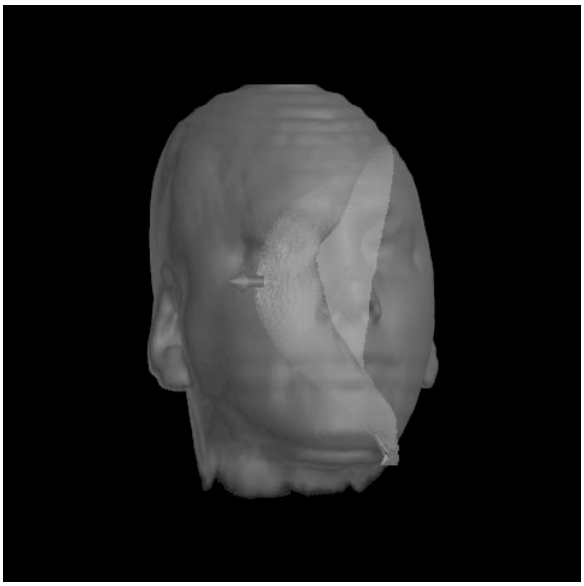

Fig. 4. A visualization of electrical current densities on the scalp, skull, and brain from a finite element simulation of temporal lobe epilepsy. From Weinstein.

III. Model Construction and Mesh Generation

In this section, we describe many aspects of the geometrical modeling process. Most realistic bioelectric volume conductor models are based upon MRI and/or CT images of patient anatomy. As such, procedures such as segmenting the images and automatically generating a discrete polygonal mesh are essential components in modeling and simulation of bioelectric fields. Figure 6 depicts a cut-away of the Utah Torso, a large-scale thorax model highlighting the different anatomical regions. The model was defined by over 20,000 surface points and 1.5 million tetrahedral elements [4], [77].

A. Segmentation

Image segmentation can be defined as the process of defining boundary domains in 2D and 3D images. Before surface reconstruction, mesh generation, and other modeling operations begin, the researcher must define the boundaries of interesting (relevant) regions within the images. In spite of extensive research in the field, there is still no algorithm that can automatically find region boundaries unfailingly from clinically obtained medical images. There are two reasons for this. One is that most of the image segmentation algorithms are still noise sensitive. The second reason is that most segmentation tasks require certain specialized background knowledge about the region(s) of interest, as modelers do not want to include (or cannot include because of size restrictions) the details that are yielded from many image segmentation algorithms.

Generally image segmentation involves classifying an image into individual objects based on object/image properties. A digital image is a discrete sampled representation of a continuous function (intensity and/or chromaticity). Medical images are obtained from MRI, CT, or other medical scanners with discrete sensors that permit each image to be represented as set of pixels indicating the value of the intensity function at every location within the image, usually as a two-dimensional array of intensity values. Digital image segmentation is the process of applying one or more operations to this two-dimensional array to classify the image into regions of similar properties like color, texture or intensity.

An example of the result of segmenting a 2D MRI scan of the thorax is shown in Figure 7.

To ensure the accuracy of the segmentation, many researchers still manually segment images by inserting control points on the images by hand and then invoke data fitting algorithms to fit curves through the control points. However, because a "typical" large-scale model has complex geometry, model construction requires the processing of a large number of images. Furthermore, as suggested by the recent work of Kikinis et al. [119], physicians are interested in near real-time segmentation for use in time-critical (such as surgery) healthcare applications.

These last two points underscore the significant challenge image segmentation and related model construction pose for biomedical modelers. For these reasons, creating new algorithms to permit a computer to recognize objects and take appropriate actions is one of the major research areas in computer vision [120], [121], [122], [123].

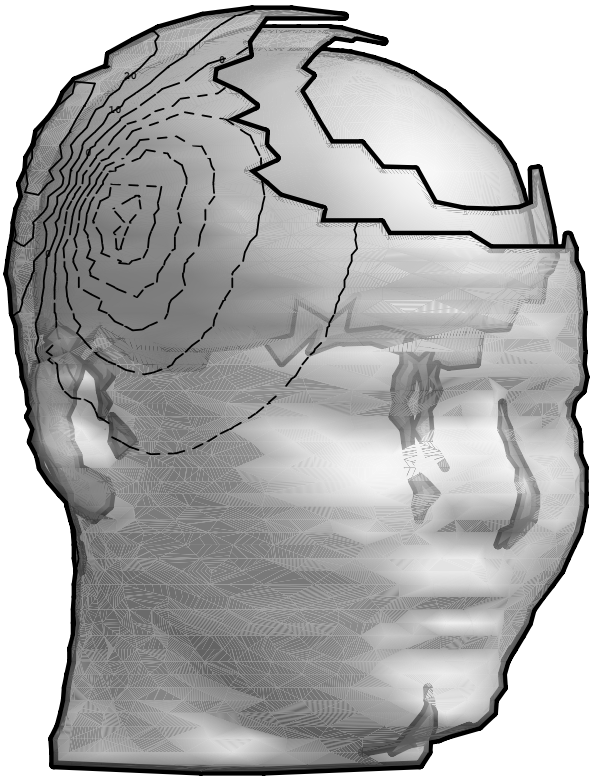

Fig. 5. In the study of evoked potentials, the regions in the brain that are active in repsonse to some kind of external stimulus are modeled as current dipoles. In order to find the positions of the dipoles (i.e. determine which regions of the brain are active) from measured EEG signals, one needs a model of the medium trough which the currents are conducted. The above volume conductor model, which includes the head, the skull and the brain, was constructed from MRI images. On the head the electric potential distrution generated by activity in a small region of the cortex is depicted.

B. Image Preprocessing

The segmentation of a digital image into the component objects may be carried out using one or more operations such as thresholding, edge detection, region growing, etc. Medical scanning devices add noise to the image, and thus decrease the discriminative ability of the algorithms based upon the above operations. To minimize the effects of noise, images usually need to be preprocessed using linear/nonlinear filters. A sample of preprocessing methods that may be applied to an image to make it a more suitable input for the segmentation process are discussed below.

B.1 Gray Scale Transformation

A gray-scale transformation operates on the range of pixel values (p_0, p_n) to convert them into a range (q_0, q_n). Most gray-scale images have approximately 256 distinct gray-scale values. Thus, lookup tables are employed to achieve the transformation. The transformation may involve a brightness thresholding, in which case all pixels above a certain threshold are colored white and those below black. Another common type of the transformation uses contrast enhancement, which is usually achieved using histogram equalization. Histogram equalization attempts to obtain an image with equally distributed brightness across the entire brightness level by enhancing the contrast for values close to the maxima and decreasing the contrast for brightness values close to the histogram minima.

A direct application of the gray-scale transformation to multi-spectral images involves using pseudo color palettes. Here, an indexed palette is used to represent different intensities as colors. The colors in the palette may be assigned so that regions of interest are more brightly or contrastingly colored as compared to the regions of little or no interest, thus causing the interesting regions to stand out in an image.

B.2 Averaging & Local Filtering

Averaging and local filtering methods attempt to minimize the error in the image. Averaging consists of obtaining n static images of the same scene and averaging the corresponding pixels. The effect is to reduce the standard deviation

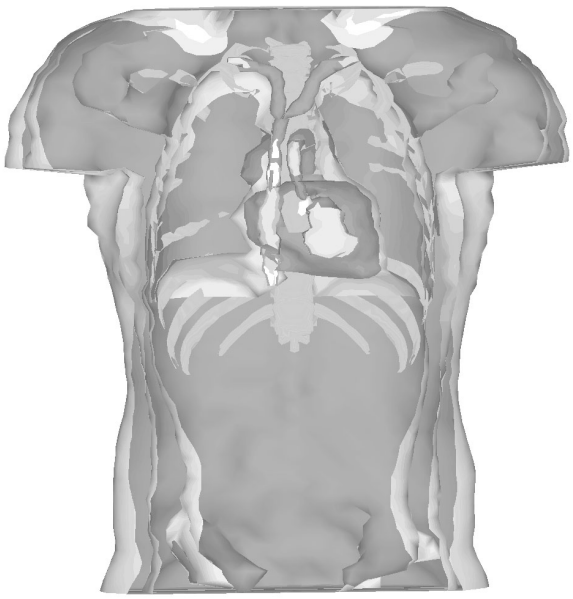

Fig. 6. A cut-away of the Utah Torso Model showing the different anatomical regions included in the model. The Utah Torso Model was derived from MRI scans and segmented to provide the boundary points defining the major electrically relevant anatomical regions, including subcutaneous fat, skeletal muscle, lungs, ribs (clavicle and sternum), epicardium, epicardial fat pads, blood cavities (atria and ventricles), and the major blood vessels (pulmonary artery and vein, aorta, superior and inferior vena cava, and subclavian, innominate and azygous veins).

between pixels by a factor of $\sqrt{n}$. The advantage of image averaging is that smoothing is achieved without blurring the original image. However, it is not always possible to obtain a sequence of static images. If only a single image is available then the averaging is performed in a local neighborhood of the pixel. A 3×3 or 5×5 local neighboring region is first defined and then the value pixels are convolved with a filter kernel centered around the pixel of interest. The size of the filter used depends on the smallest object that needs to be resolved from the image. Examples of different filter kernels include the box, triangle, Gaussian, and sinc filters. Filtering the image usually decreases the noise but at the cost of blurring the image. Another disadvantage of filtering is that a recursive application of the filter leads to severe image degradation within a few passes of the filtering algorithm. However, if the averaging process is limited to pixels that form part of a particular feature, the effects of blurring can often be minimized, thus application of a local filtering algorithm can often enhance the boundary domains and thus increase the success of segmentation algorithms.

B.3 Median Filtering

Median filtering is a smoothing method that reduces the blurring of edges [124] and eliminates impulse noise that can be added during imaging. Impulse noise appears in an image that has a number of pixels that have the obviously incorrect intensities, such as 0 or 255. The idea is to replace the current point in the image by the median of the brightness in its neighborhood, such that is forces points with very distinct intensities to be more like their neighbors. The median is not affected by impulse noise; hence, any impulse noise is eliminated. In addition, median smoothing does not blur the image and can be recursively applied. However, median filtering damages the sharp contours formed from thin lines. Horizontal/Vertical lines may be preserved by defining special neighborhood relationships for the pixels.

C. Segmentation Methods

In this section we discuss the methods that are commonly applied to digital images to segment them into regions. Most of the methods discussed are general and can be applied to images obtained from different medical imaging modalities.

C.1 Edge Detection

Edge detection uses a collection of local image processing methods to locate sharp changes in the intensity function of the image and if often used as a first step to image segmentation after filtering. An edge is a vector variable that is

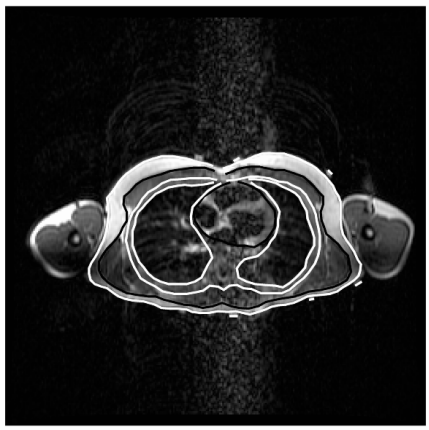

Fig. 7. A 2D MRI scan of the thorax after segmentation. The polylines define the major electrically relevant anatomical regions including the torso surface, subcutaneous fat, skeletal muscle, lungs, ribs, epicardial surface, and electrodes from a body surface potential map.

a property of an individual pixel. The magnitude of the edge is the magnitude of the gradient and the corresponding direction of the edge such that it is rotated -90 degrees with respect to the direction of the gradient. The edge profile is a plot of the intensity function along the edge direction. The edges, then, correspond to pixels where the image intensity changes abruptly. This change in image function can be expressed by a gradient that points in the direction of the largest change. Typical profiles of edges include step, triangle, line, and noisy functions. Many edge detection algorithms are specialized to detect certain kinds of edges. Edge detection operators can be roughly classified into the following types (see [123] and the references therein for further information)

- Operators that determine the maximum value from a series of pixel subtractions (e.g. Homogeneity operator and difference edge detector).
- Operators approximating derivatives of the image function using differences (e.g. Laplacian, Roberts, Prewitt, Sobel, Robinson, and Kirsch).
- Operators based on zero crossings of the second derivative of the image function (e.g. Marr-Hildreth and the Canny edge detector).
- Operators that match an image function to a parametric model of the edges (e.g. Hough transforms).

C.2 Thresholding

Thresholding is the simplest and most intuitive of the segmentation methods. Image thresholding tries to emphasize the strong edges and deemphasize the weak edges. Most regions in an image have a constant reflectivity/transmitivity and hence show up in the image as regions of constant/near constant brightness. This brightness value can be used as a threshold to segment the regions from the image. A simple algorithm to do the thresholding would work as follows:

1. Select an appropriate threshold value.
2. Scan all the pixels in the image.
3. Set all pixels greater than or equal to the threshold to 1.
4. Set all other pixels to 0.

If the regions do not touch each other and have gray-scale values distinct from the background, then thresholding is suitable as a segmentation method. Since thresholding is a simple tool with a reasonable efficacy, many variations have been made to improve upon the original algorithm. For example, most objects within an image may have different gray-scale properties and the gray-scale values may vary depending on the location of the object in the region. A variation on the original algorithm splits the image into multiple sub-images and uses a threshold value that is appropriate within the particular sub-image. Another variation sets the region pixels to 1 if the values are within a certain gray-scale domain

and the background is set to zero.

Selection of an appropriate threshold for segmentation is dependent on the gray-scale of the regions interest. Certain algorithms assist the selection of an appropriate threshold value. For example, P–Tile thresholding assumes a certain prior knowledge about the nature of the image. If the percentage of the pixel occupancy of regions of interest within the image is known, then the threshold is easily calculated by scanning the pixels and finding a gray-scale value at which the percentage of pixels above the value is equal to the percentage of pixel occupancy of interesting regions.

A more general approach than the P–Tile approach is to examine the peaks and valleys in the image histogram. Regions of similar gray-scale values will show up as peaks in the histogram. If there is only one region then, the histogram has two peaks, one for the gray-scale values representing the region and one for the background. Such a histogram is called a bimodal histogram. Most images with multiple gray-scale regions will have many peaks and are therefore called multi-modal. An appropriate threshold value is often selected by taking the midpoint gray-scale value between a set of peaks.

A semi-automatic image segmentation tool specifically for use in model construction for bioelectric field problems has been described by Shen [125]. Shen's software tool combines interactive manual segmentation utilities with an automatic image segmentation algorithm. By combining these two segmentation methods, manual and automatic, a user can obtain accurate boundary descriptions with a minimum of effort. To achieve manual segmentation, the researcher "drops" control points onto the image. Given a 2D image slice, the user places these points on a visible region boundary, then invokes a data fitting algorithm, via a selection from a pull-down menu, to interpolate between these control points to generate a region contour. The program then automatically fits cubic splines to the selected points. Several different data fitting methods can be used. The methods differ based either on the degree of interpolation function used – i.e. linear or quadratic or on the basis function used – B-spline, Bezier spline, etc. In automatic segmentation mode, the researcher selects a particular boundary; then, the program uses a bimodal thresholding algorithm within a local window of the target image, to produce boundary pieces. The individual pieces are then connected to yield the entire region boundary. In the automatic segmentation procedure, users interactively steer the segmentation process. The program begins by prompting the user to select an initial starting point on the region boundary. From that point, local edge detection and contour following programs are used to find the region boundaries. The contour algorithm gives only a small contour segment piece at a time. The program then waits for feedback from the user. The user may inspect and if necessary, correct the identified contour segment. If no corrections are needed, the program continues to find the next contour piece.

Hierarchical thresholding is the method that aims to detect regions in lower resolution images and to improve the precision of the region in higher and full resolution images. A major advantage of this method is that there is less influence of noise on the segmentation, since the lower resolution images are smoothed. Also, because the segmentation proceeds on to higher resolutions, the imprecise borders that result from segmenting smoothed images are refined in the re-segmentation process.

C.3 Edge-based Segmentation

The application to an image of the edge detection operators described earlier results in an edge image. Edge-based segmentation is the process of merging the individual edges into edge chains that represent the borders of existing objects in the image. Edge-based segmentation is more successful if prior knowledge about the shapes of the regions being segmented is known. It is also sensitive to the amount of noise in the image. Noise can affect the segmentation operators by causing the operators to detect edges where none exist, and also by causing the operators to miss existing edges.

The simplest method of segmenting an edge image is to apply a thresholding operator. The edges may be classified based on the fact that the stronger the edge the higher the threshold [126]. By choosing an appropriate threshold, a researcher can ensure that only the significant edges will be chosen to appear in the segmented image. If there is prior knowledge about the nature of the edge, then P–Tile thresholding may be used. However, this method is still limited by the fact that noise is a major disadvantage of both edge detection and thresholding methods.

Edge relaxation is a method of segmenting an edge image that classifies the edges based on their neighbors, on edge confidence, and on appropriately growing the stronger edges to form borders [127], [128], [129], [130]. A crack edge is the edge formed in the gap between two pixels [131], [132]. Each crack edge has 6 neighboring crack edges. The strength of the edge is defined in terms of a $m - n$ tuple where m is the number of strong edges at one end of the edge

and n is the number of strong edges at the other end. A confidence rating of $0-0$ indicates an isolated edge, probably caused by impulse noise in the image, and has a negative influence on the overall edge confidence. A confidence rating $1-1$ indicates an edge that is probably a continuation of a border and that has a strong positive influence on overall edge confidence. Edges with confidence ratings $1-2$ and $1-3$ represent border confluences and have a medium positive effect on overall edge confidence. Edge ratings $0-2$ or $0-3$ represent dead ends, whereas a rating of $0-1$ indicates an uncertainty and has a weak positive or no influence on overall edge confidence. Edges rated $2-2$, $2-3$ or $3-3$ represent bridges between borders [133]. Edge relaxation is an iterative process, with each edge confidence being updated as its neighboring edge strengths are calculated. Each edge is then set to a 1 or 0 based on its confidence value and a threshold for edge selection. Edge relaxation usually works well in the first few iterations and then slowly drifts, giving increasingly worse results.

Other methods for edge segmentation include heuristic graph searching [134], [135], edge following using dynamic programming, and region construction from partial borders.

A method for detecting geometric shapes/borders in images is know as the Hough transform [136]. To detect straight lines in a image, an algorithm transforms all the edge pixels into the parameter space for the line via the slope (m) and the intercept (c). A single point in the parameter space thus represents a line through the image. Each border in the image is transformed into the parameter space and a counter is incremented for each (m, c) pair. The (m, c) pairs with the most occurrences represent the significant lines in the image. Similar inverse transforms can be applied to detect other geometrical/parametric shapes in the image. Fuzzy Hough transforms [137] allow detection of regions whose exact shape is unknown, but in which there is enough *a priori* information to form an approximate representation of the object. After the general Hough transform is applied to the approximate model of the object, the algorithm recovers the true border by locating points with maximum weighted edge strength along a line radial to the detected border. The advantages of the Hough transform are that it allows recovery of partial, obscured or deformed edges and is insensitive to noise. Another advantage is that it can find all occurrences of the same shape in the image in one pass.

C.4 Region Growing

Region growing methods have become popular techniques for extracting surfaces within volumes of medical imaging data. The basic idea of region growing is to divide an image into zones of maximum homogeneity. A region is defined as a set of pixels in which all the pixels forming a path between any pair of pixels belong in the same set. Region growing techniques tend to do well with noisy images where edges are otherwise difficult to detect. The region can be broken into different homogeneous regions according to gray-scale level, texture and/or other criteria [138], [139], [140], [141].

Region growing methods are usually further classified as region merging, region splitting and region splitting and merging. Region merging is a bottom-up approach to image segmentation. At the beginning of the process, each pixel represents one region. A criterion is defined for merging adjacent similar regions. The criterion is then applied to the image, and adjacent regions are merged until there are no two adjacent regions that can satisfy the merging criterion. The most commonly used merging criterion is based on edge strength: regions whose common border consists of weak edges are candidates for merging.

Region splitting is a top down approach and is the opposite of region merging. At the beginning of the process, the entire image is considered as one region. This region obviously will not satisfy the homogeneity criteria. The process will split the image into increasingly more homogeneous regions until all the sub regions satisfy the homogeneity criteria. Although the approaches to region merging and splitting appear opposite, they are not duals of each other, and different segmentations may be created by the two processes.

Splitting and merging uses a pyramid scheme to represent images. The top level images are low resolution filtered versions of the images lower down in the pyramid. If the top level image does not satisfy the homogeneity criteria, it is split into 4 square subregions. Each subregion is derived from the higher resolution image lower in the pyramid. Adjacent regions that satisfy the homogeneity criteria are merged and the merging is propagated back up the pyramid. The process can be represented as a segmentation quad-tree; at the end of the process, each leaf node of the tree represents an individual segmented region.

Lately, segmentation techniques have focused on extending previous segmentation techniques to include an n-dimensional vector for each voxel rather than a scalar value. Such techniques require as input multiple volume data sets, all with identical spatial domains. Ideally, each volume would show different properties of the material and the resultant vectors

could be easily segmented into the correct material types.

One implementation, *VecSeg*, uses MRI scans with different "weightings" [142]. An MRI weighting specifies the frequency of the magnetic pulse, the duration of the pulse, and the relaxation time between pulses. These parameters determine the "material to intensity mapping" that will result from the scan. In a recent use of *VecSeg*, a patient underwent two different weighted scans, which resulted in two volumes, each containing 81 256×256 MRI slices from the same patient. Scans from the two different weightings are shown in Figure 8.

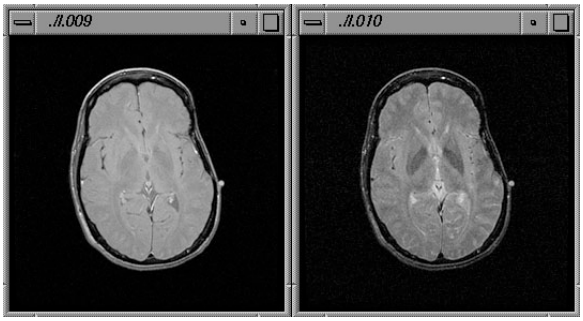

Fig. 8. Two MRI images showing the different data obtained for two different weightings. The left image is a T1 weighting, and the right image is a T2 weighting.

Initially, *VecSeg* reads in the data files and passes the volume data to clipping modules. The clipping modules are linked so that they will always define the same volume. These modules enable the researcher to segment the volume quickly and help to determine the best material ranges with a high degree of interactivity. They enable the user to interact with a small subset of the domain initially, and then, once the desired ranges have been determined, to pass the entire volume in to be segmented. The set of modules used for *VecSeg* are shown in Figure 9

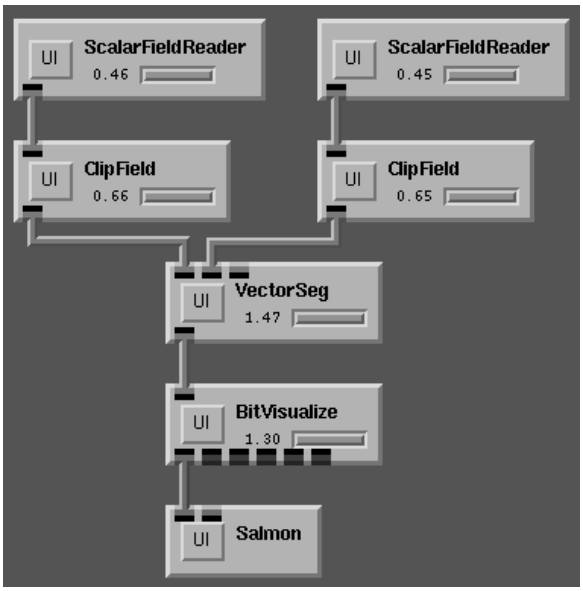

Fig. 9. The network VecSeg uses for vector based automatic segmentation.

The next stage in the *VecSeg* pipeline is the segmenting module, which accepts the clipped volume fields as input. The interface for this module is a 2D array of range sliders for specifying grey-scale values of materials from each input volume. In the example case of *VecSeg*, the module accepts two fields as input and segments for several distinct materials. If a voxel's vector is contained within the space spanned by the ranges of that material, that voxel is set to be "on" for that material.

Finally, *VecSeg*'s vector segmentation module creates a field of n-bit numbers, with one bit corresponding to each material. This field is passed on to a final module that extracts surfaces for each material volume, as well as for a volume of colored points corresponding to each voxel's identified material. The volume of points is a "quick-and-dirty"

method of volume-rendering the segmented data, and is a fast way to look at the segmentation results. The module can also output boundary surfaces for each material. An example of the output from *VecSeg* is shown in Figure 10

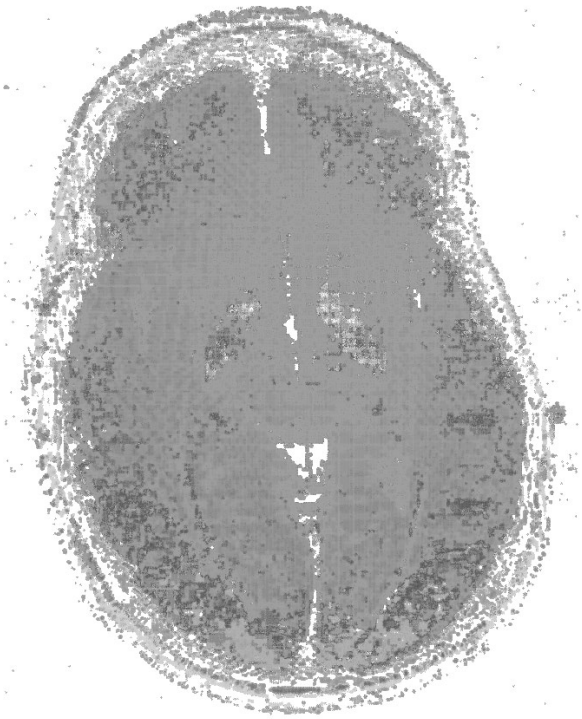

Fig. 10. Automatic segmentation of a portion of a brain using the VecSeg dataflow system.

There are a number of excellent general references in image processing and computer vision that contain extensive sections on filtering and segmentation such as [122], [123], [143], [144], [145]. A particularly useful reference is [120], which includes C code for all of the image processing algorithms discussed throughout the book.

D. Surface Construction

Before applying a particular solution method to realistic models, one needs to create a computational mesh. For the boundary element method, the computational mesh is a three-dimensional surface usually made up triangles. For the finite element method, one needs to construct a volumetric computational mesh. If one uses unstructured elements, then one usually creates a volume tetrahedralization. To utilize certain automatic tetrahedral mesh generators in creating volume representations of these models [146], it is necessary to connect the segmented points/lines into polygonal surface descriptions. Thus surface modeling plays an important role in constructing realistic models for computational bioelectric field problems. Triangular elements are a good choice for irregular surfaces such as those found in the human body, and their use is supported by criteria by which "optimal" triangles can be constructed from given point sets [147], [148], [149].

While the general problem of creating an optimal triangulation from points in three dimensions is quite complex, significantly simpler approaches can be used if the points are constrained to lie in planes, as they almost always do in data collected from imaging systems. If the surfaces to be constructed lie more or less perpendicular to the image planes, then the job is reduced to one of "lacing" the points in adjacent layers into optimal triangles. Algorithms for performing the lacing operations have been described [150], [151], [152] and shown to work, even in situations in which the surface bifurcates, as in, for example, the structure of blood vessels [151], [152]. In [153], MacLeod and I described a triangulation strategy for connecting data from segmented MRI scans into triangulated surfaces. The results are discussed below.

In order to apply triangulation techniques to the construction of a model of the human torso (or head), we combined the points from adjacent layers into three-dimensional "slices", which were the input for our lacing programs. Triangulation of each slice involved the following steps:

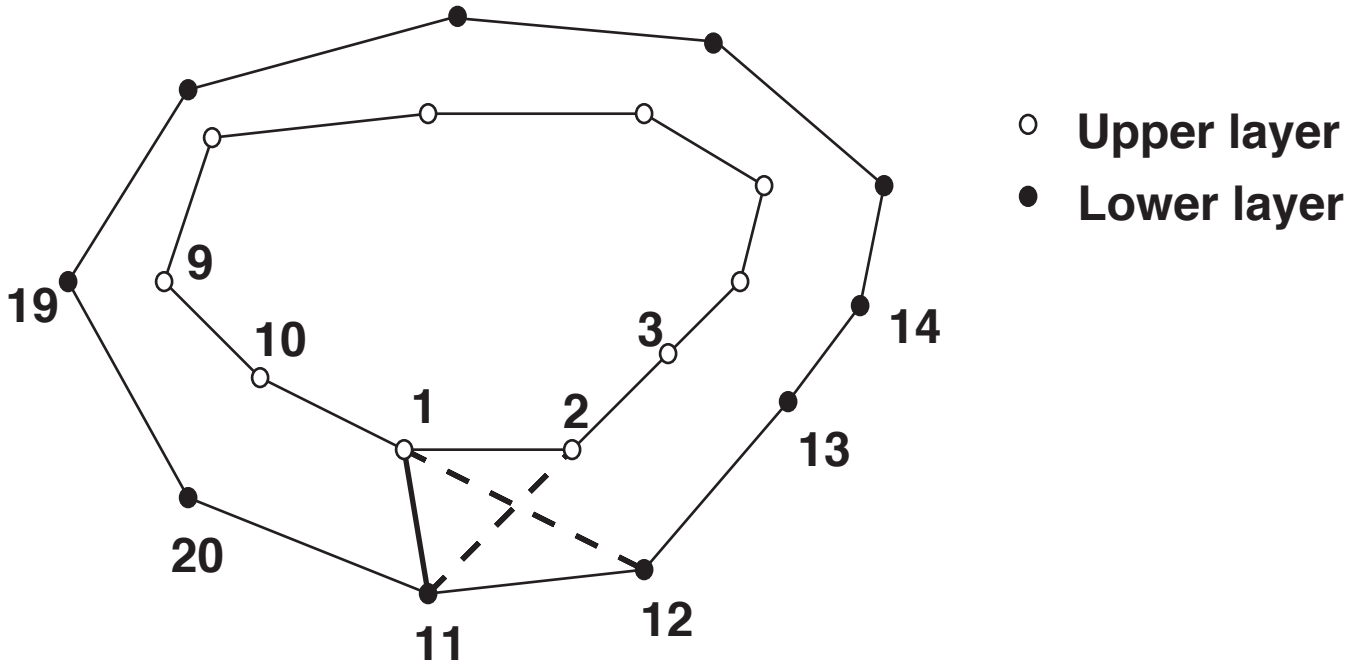

Fig. 11. Example of lacing two layers to triangulate the surface enclosing the resulting slice. Points form the upper layer are shown as empty circles, those from the lower as filled circles.

1. Determine an anatomically based starting point and locate the nearest node from each layer of the slice to that starting point. The anterior midline of the body is a typical starting location.
2. Order the points for each surface in each layer in such a way that they form a continuous sequence, commencing at the starting point and proceeding in a common direction around the surface. This is typically performed during fitting of the sampled points (using a cubic spline, for example).
3. Determine a starting point on one of the layers, and find the nearest point in the second layer; join these to define the first connective segment.
4. Proceeding to the next point in each layer, determine which of the two possible pairs of triangles that can be formed by joining them with the endpoints of the first segment yield the shortest diagonal (and hence the most optimal triangulation in terms of maximizing the size of the smallest angle in each triangle [148]).
5. Once this pair of triangles is defined, and a new connective segment is formed, repeat the previous step until all points have been included, and the surface completely triangulated.

Figure 11 shows this process in a simple case with 10 points in each layer of a slice. The starting point is at point 1, which is connected to point 11. Testing of the two possible diagonals (from points 1 to 12 or points 2 to 11), shows that the latter produces triangles with larger minimum angles and hence is the proper choice. A second triangle, consisting of points 2, 11, and 12 is also immediately constructed and the segment between points 2 and 12 becomes the new starting point for further lacing.

While this basic algorithm will work *most* of the time, there are a number of situations in which additional mechanisms are required. In some cases of extremely irregular surfaces the lacing scheme must be augmented with some checks for triangles which are too large, or must be formed, not between nodes on different layers, but in the plane of one of the layers. Errors frequently occur in regions of significant concavity, in which triangle segments run outside the hull of the body, as, for example, in the crescent-shaped regions of the lower lungs. Simple examples of these situations are given in Figure 12. In panel **A** the upper layer describes a smaller surface than the lower layer, and a single point (point P in the figure) is shared by seven triangles, instead of two or three, as for all the other points. Panel **B** shows a situation in which erroneous triangles (shown in thicker, dashed lines) are constructed across a region which is outside the actual surface. Such triangles must be detected and removed, ideally without user input.

Determining when a triangle lay outside the surface was one of several related problems that arose during the model construction (and which will be described in more detail below). The basic question was whether a point was *inside* or

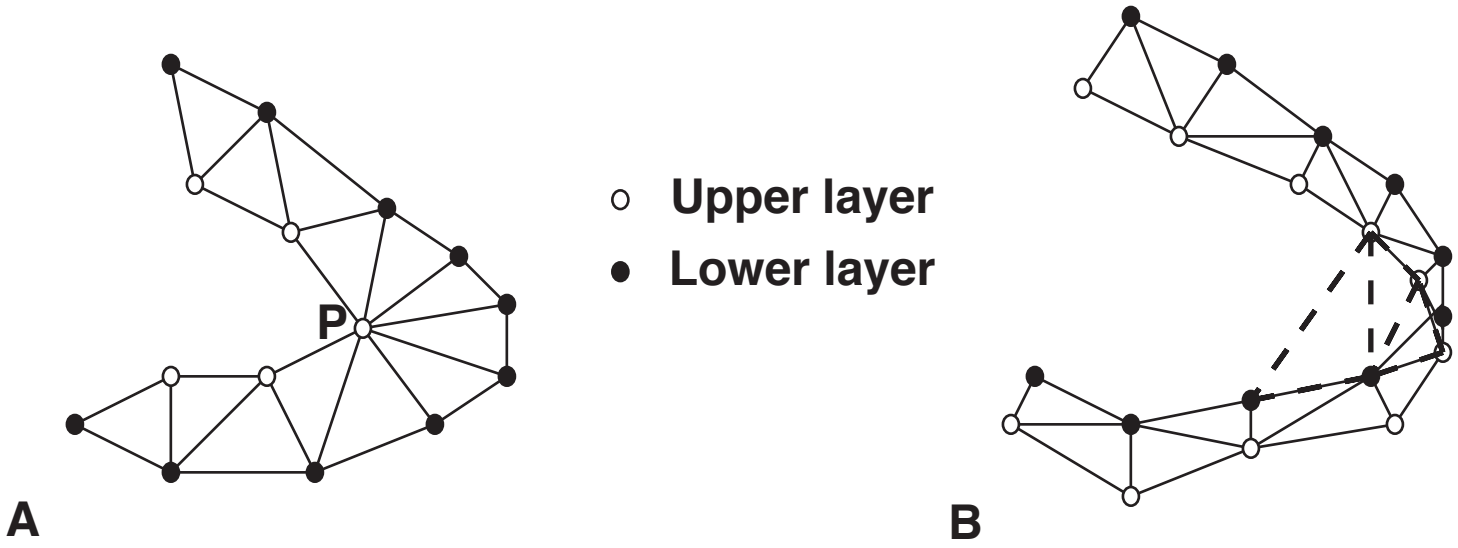

Fig. 12. Two examples of special cases in lacing triangles. In **A**, the two layers join to form a very flat surface since one layer does not lie directly over its neighbor. In panel **B**, a sharply curved surface can result in erroneous triangles which lie outside the actual surface (shown as dashed lines.)

outside a surface. To solve this, we again made use of the fact that all the points in the tomographic data lay in discrete planes. If a point lies inside the curve defining the intersection of a surface and a plane (the "surface curve"), then the algebraic sum of all the angles about the point that are formed between adjacent points on the curve should be equal to 2π. For a point outside the curve (and hence the surface partially defined by the curve), the sum of the angles is zero. Using this fact, and provided that the points have already been ordered in a strict sequence of first-order neighbors, one can construct a set of tests with which the location of a point, relative to a surface, can be determined. This same concept can be extended directly to three dimensions if, instead of planar angles, the sum of the *solid angles* about a point is computed. However, this calculation requires knowledge of the outward normals on an already complete surface tessellation and is therefore unsuited to this application. Once completed, however, a computation of the sum of the solid angles is an excellent means of checking for complete closure and consistency of the outward normals.

In the particular application of triangulation, in which one wishes to detect and remove triangles that lie outside the concave surface, there is, however, potential for error. Since any point that is tested, say, the midpoint of each side of the triangle under examination, actually lies between the two planes of the data points, one must project it onto one or the other of the planes and test the projection of the point relative to the surface curve. Sufficient practical accuracy can be achieved, however, by testing each such point, and perhaps several others in the triangle, projected onto *both* planes. We have found that checking the location of the projected centroid of the triangle against surface curves on both layers (and eliminating the triangle if the centroid lies outside either) is adequate for most purposes.

Once laced into multiple slices, the points and triangle connectivities must be concatenated to make a complete surface description. We usually perform this operation on a surface-by-surface basis so that the end result is a set of points and connectivities for each of the surfaces in the complete model. This allows for easy selection of the surfaces that are to be included in a computation and/or visualization. Before they are ready to use, however, there are two final steps in the model construction process.

The first involves connecting the points which form the 'ends' or 'caps' of the model, that is, the first and last layer of each surface. This can be performed in one of two ways, either by triangulating the points in the layer, or by adding a grid of points to fill in the region defined by the points in the end layers, and then triangulating the resulting augmented layer in two dimensions. We employ both approaches, depending on the size of the opening at each end and the desired density and spatial resolution of the model we are constructing. If a grid is added, its density is, likewise, dependent on the desired resolution of the model. The triangulation of the resulting two-dimensional point set is a straightforward problem and numerous programs exist for this purpose, some even in the public domain (*e.g., voronoi* from the netlib collection). What is usually missing from these algorithms, however, is a provision for concave curves, in which case triangles are constructed which lie outside the curve. Hence we have implemented a check of all the triangles formed by

the two-dimensional triangulation using the inside/outside test described above, applied to the centroid of each triangle.

The final step in constructing a surface model of the geometry involves setting the direction of the outward normal vectors to each of the triangular elements. Finding a normal vector is easily done by computing the cross product of any two sides of the triangle. Ambiguity remains, however, in the final choice of which of two opposite directions is outward and which inward. This can be resolved in most cases by applying, once again, a test to the endpoints of both candidate normal vectors to determine which lies inside and which outside the surface. We add both normal vectors to each of the three vertices of the triangle, then test the endpoints relative to the surface to which the vertex belongs, until we achieve an unambiguous result of one endpoint found inside and the other outside the surface. The coding of this information into the model itself occurs in the order in which the points of each triangle are recorded: the points are always taken in counterclockwise order when viewed from the outside of the surface. As a final, complete check of the integrity of the model, we select a point inside each surface and compute the total solid angle about that point. If the surface description is complete (no holes), and all the outward normals are correct, the result must be 4π.

A recent series or papers by Hoppe et al. [154], [155], [156] presents a method for triangulating 3D surfaces from a set of discrete points. Postprocessing of triangulated surfaces can be accomplished by applying the Laplacian smoothing algorithm of Oostendorp et al. [157].

E. Mesh Generation

Once we have stated or derived the mathematical equations that define the physics of the system, we must figure out how to solve these equations for the particular domain we are interested in. Most numerical methods for solving boundary value problems require that the continuous domain be broken up into discrete elements, the so-called *mesh* or *grid*, which can be used to approximate the governing equation(s) using the particular numerical technique (finite element, boundary element, finite difference, or multigrid) best suited to the problem.

Because of the complex geometries and the enormous number of degrees of freedom (upwards to tens of millions elements) associated with many problems in computational medicine (and specifically the bioelectric field problems considered here), construction of the mesh can become one of the most time consuming aspects of the modeling process. After classifying the relevant regions of interest by segmenting the medical images and deciding upon the particular approximation method to use (and the most appropriate type of element), we need to construct a mesh of the solution domain that conforms to the segmented boundaries and matches the number of degrees of freedom of the fundamental element we've chosen.

There are several different strategies for discretizing geometry into fundamental elements. Bioelectric field simulations require the modeling of complex geometric domains such as those found in the human thorax and head (eg., heart, lungs, skeletal muscle, vasculature, body surface, brain, and skull). These different anatomical structures are extremely irregular and do not permit the efficient use of standard CAD/CAM descriptors commonly used to describe synthetic structures. Instead, the geometric objects that comprise the geometric model are described by segmented sets of 2D or 3D MRI or CT images consisting of contours or surfaces at the boundaries between organs and/or regions of different conductivity. The resulting contour or surface data is then used to construct a polyhedral representation of the solution domain. For bioelectric field problems, two approaches to mesh generation have become standard: the *structured partitioning* (or subsequent subdivision) strategy [158], [159]; and those based upon a *Delaunay triangulation* strategy [158], [160]. Other mesh generation techniques that could be used for bioelectric field problems include mapping methods, paving (advancing front methods) [158], [161], and octree methods [162], [163]. Most researchers have chosen to discretize the solution domain with either tetrahedra for volumes and triangles for surfaces, which are usually used for modeling irregular three-dimensional domains, or hexahedrons (rectangles) used for modeling regular, uniform domains.

In using the structured partitioning strategy, one starts with a set of points that define the bounding surface(s) in three dimensions (contours in two dimensions). The volume (surface) is repeatedly divided into smaller regions until a satisfactory discretization level has been achieved. Usually, the domain is broken up into eight-node hexahedral elements, which can then be subdivided into five (minimally) or six tetrahedral elements if so desired. This methodology has the advantage of being fairly easy to program; furthermore, commercial mesh generators exist for the divide and conquer method. For use in solving bioelectric field problems, its main disadvantage is that it allows elements to overlap interior boundaries. A single element may span two (or more) different conductive regions, for example, when part of an element represents muscle tissue (that could be anisotropic) and the other part of the element falls into a region

representing fat tissue. It then becomes very difficult to assign unique conductivity parameters to each element and at the same time accurately represent the geometry. While this technique works for domains of arbitrary shape, in the case of a complex geometry one must use a large number of elements to adequately define the shape [158].

A second method of mesh generation is the Delaunay triangulation strategy. Given a three-dimensional set of points that define the boundaries and interior regions of the domain to be modeled, one tessellates the point cloud into an optimal mesh of tetrahedra. For bioelectric field problems, the advantages and disadvantages tend to be exactly contrary to those arising from the divide and conquer strategy. The primary advantage is that one can create the mesh to fit any predefined geometry, including subsurfaces, by starting with points which define all the necessary surfaces and subsurfaces and then adding additional interior points to minimize the aspect ratio. For tetrahedra, the aspect ratio can be defined as $\sqrt{\frac{3}{2}}\frac{h_k}{\rho_k}$ [164], [165], where ρ_k denotes the diameter of the sphere circumscribed about the tetrahedron, and h_k is the maximum distance between two vertices. These formulations yield a value of 1 for an equilateral tetrahedron and a value of 0 for a degenerate (flat) element [166]. The closer to the value of 1, the better. The Delaunay criterion is a method for minimizing the occurrence of obtuse angles in the mesh, yielding elements that have aspect ratios as close to 1 as possible given the available point set. While the ideas behind Delaunay triangulation are straightforward, the programming is nontrivial and is the primary drawback to this method. Fortunately, there exist several public domain, two-dimensional programs, including one from Netlib called sweep2.c from the directory Voronoi (see section VII for the URL), as well as several newly available three-dimensional package [167] (see section VII for the URLs).

We now proceed to discuss the Delaunay mesh generation method used by Johnson and Schmidt [146], [82], [168] to create large-scale tetrahedral meshes of the thorax and skull. The method used to create meshes is based upon the Delaunay tessellation algorithm originally proposed by Watson [169] and later extended by Weatherhill [160]. The Delaunay criterion states that the circumsphere of any tetrahedron (triangle[2]) contains no other mesh points. The thrust of the Watson/Weatherhill algorithm is to efficiently insert a point into an existing grid (bounding simplex) in such a way that the Delaunay criterion is met. Certain tesselations are then deleted and new ones are subsequently created using the new point and a subset of the the old points. The general method is applicable to N-dimensions, although engineering applications usually require implementations in two or three dimensions [168], [82]. A more detailed description of the mesh generation algorithm can be found in [82].

The mesh generation procedure consists of five steps as outlined in the following algorithm:

Mesh Generation Algorithm
Inputs: Boundary point representation (in contours)

```
Begin
  Triangulate surfaces
  Construct coarse mesh - Delaunay tessellation
  Determine interior tetrahedra
  Repeat:
    Generate interior point
    Tessellate interior point
    Until (point fails spacing and degree tests)
  Classify regions
End
```

Starting with the boundary points extracted during the segmentation, one then proceeds to adequately represent the surface mesh (line segments in two-dimensions and triangles in three-dimensions). The next step is to construct a coarse mesh of tetrahedra from the boundary points and then to determine tetrahedra within the surface of interest. These interior tetrahedra are used in the fourth step, which iterates between the generation of a new point and subsequent tessellation until certain spacing criteria are satisfied. The spacing criteria are what ultimately determines the size of the mesh.

The final step is to classify the tetrahedra by their material properties, which vary depending on whether the tetrahedra are considered to be in the heart, lungs, etc. Since the material property of interest in bioelectric problems, electric

[2]In describing the algorithm, we use triangle or tetrahedron interchangeably.

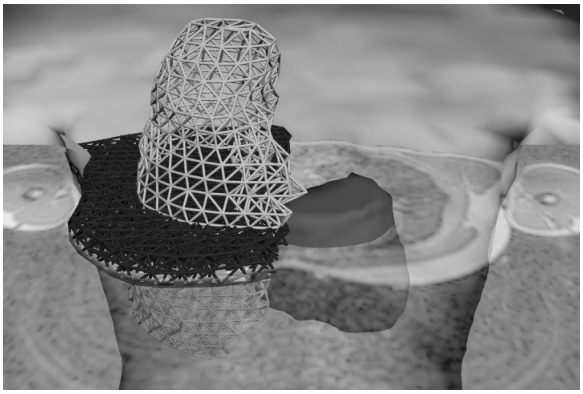

Fig. 13. A view of the model construction process. Starting from MRI scans, the geometric model is constructed through image segmentation, triangulation of the surfaces (such as the lung), and automatic tetrahedral mesh generation.

conductivity, can be anisotropic, each tetrahedron can be assigned a tensor describing local conductivity. To classify a tetrahedron, the position of its centroid relative to the surfaces separating regions of different conductivity is localized. To localize the element, a ray tracing approach [82] is used. This technique projects a ray from the centroid of an element to a point at infinity and counts the instances the ray intersects with a triangulated surface. If the ray crosses through one triangulated surface an odd number of times, the point is considered to be interior to the surface; conversely, if it crosses the surface an even number of times, the tetrahedron is considered to be exterior to the surface. A composite of the segmentation and mesh generation processes are shown in Figure 13.

As previously noted, the construction of large-scale models for solving computational bioelectric field problems can be the most time consuming aspect of the modeling and simulation procedure. As such, recent work by Christensen et al. [170], [171], [172] has focused on developing a technique to automatically map (i.e. warp) a computational mesh generated for an already exiting template geometry into alignment with a new subject-specific case based on differences between CT and/or MR data sets for the two. The warping is accomplished by solving for the deformation field that aligns the existing template with the new anatomy. In this approach, image data are used to drive the deformation of the template. These data are treated as probabilistic measures of inhomogeneous physical properties queried from the underlying material continuum. As the template material deforms, the inhomogeneity moves with it, so the template image deforms with the underlying continua. The model-mapping algorithm produces a transformation that minimizes the difference between the image(s) queried from the deformed template and the individual. This same transformation can subsequently be applied to any other spatially distributed information associated with the template to map the information to the specific case. This and similar research hold promise for reducing the overall time spent in model generation as well as for helping to solve the problem of automatically registering images recorded from different imaging modalities. Figure 14 illustrates the application of the mesh generation algorithm for the Utah Torso Model [82], [165].

For more information on mesh generation and various aspects of biomedical modeling, see [173], [174], [175], [176], [177], [160], [158], [178], [82].

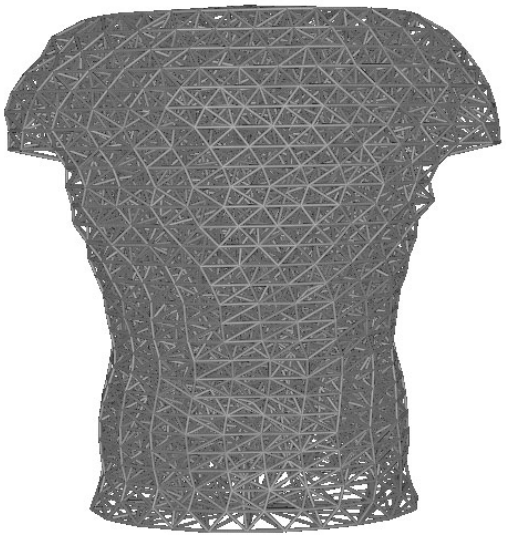

Fig. 14. Visualization of the tetrahedral structure of the Utah Torso Model.

IV. Numerical Methods

Because of the geometrical complexity of, and numerous inhomogeneities inherent in, anatomical structures in physiological models, solutions of bioelectric field problems are usually tractable (except in the most simplified of models) only when one employs a numerical approximation method such as the Finite Difference (FD), the Finite Element (FE), Boundary Element (BE), or the Multigrid (MG) Method to solve the governing field equation(s).

A. Approximation Techniques - The Galerkin Method

The problem posed in (3) can be solved using any of the aforementioned approximation schemes. One technique that addresses three of the previously mentioned techniques (FD, FE, and BE) can be derived by the Galerkin method. The Galerkin method is one of the most widely used methods for discretizing elliptic boundary value problems such as (3) and for treating the spatial portion of time-dependent parabolic problems, which are common in models of cardiac wave propagation. While the Galerkin technique is not essential to the application of any of the techniques, it provides for a unifying bridge between the various numerical methods. To express our problem in a Galerkin form, we begin by rewriting (3) as:

$$A\Phi = -I_v, \tag{15}$$

where A is the differential operator, $A = \nabla\cdot(\sigma\nabla)$. An equivalent statement of (15) is, find Φ such that $(A\Phi + I_v, \overline{\Phi}) = 0$. Here, $\overline{\Phi}$ is an arbitrary *test function*, which can be thought of physically as a virtual potential field, and the notation $(\phi_1, \phi_2) \equiv \int_\Omega \phi_1\phi_2\, d\Omega$ denotes the inner product in $L_2(\Omega)$. Applying Green's theorem, we can equivalently write,

$$(\sigma\nabla\Phi, \nabla\overline{\Phi}) - \langle\frac{\partial\Phi}{\partial n}, \overline{\Phi}\rangle = -(I_v, \overline{\Phi}), \tag{16}$$

where the notation $\langle\phi_1, \phi_2\rangle \equiv \int_S \phi_1\phi_2\, dS$ denotes the inner product on the boundary S. When the Dirichlet, $\Phi = \Phi_0$, and Neumann, $\sigma\nabla\Phi\cdot\mathbf{n} = 0$, boundary conditions are specified on S, we obtain the *weak form* of (3):

$$(\sigma\nabla\Phi, \nabla\overline{\Phi}) = -(I_v, \overline{\Phi}). \tag{17}$$

It is understood that this equation must hold for all test functions, $\overline{\Phi}$, which must vanish at the boundaries where $\Phi = \Phi_0$. The Galerkin approximation ϕ to the weak form solution Φ in (17) can be expressed as:

$$\phi(\mathbf{x}) = \sum_{i=0}^{N} \phi_i\psi_i(\mathbf{x}). \tag{18}$$

The trial functions $\psi_i, i = 0, 1, \ldots, N$ form a basis for an N+1 dimensional space $\mathcal{S}$. We define the *Galerkin approximation* to be that element $\phi \in \mathcal{S}$ which satisfies

$$(\sigma \nabla \phi, \nabla \psi_j) = -(I_v, \psi_j) \quad (\forall \psi_j \in \mathcal{S}). \tag{19}$$

Since our differential operator A is positive definite and self adjoint (i.e., $(A\Phi, \Phi) \geq \alpha(\Phi, \Phi) > 0$ for some non-zero positive constant α and $(A\Phi, \overline{\Phi}) = (\Phi, A\overline{\Phi})$, respectively), then we can define a space E with an inner product defined as $(\Phi, \overline{\Phi})_E = (A\Phi, \overline{\Phi}) \equiv a(\Phi, \overline{\Phi})$ and norm (the so-called energy norm) equal to:

$$\|\Phi\|_E = \{\int_\Omega (\nabla \Phi)^2 d\,\Omega\}^{\frac{1}{2}} = (\Phi, \Phi)_E^{\frac{1}{2}}. \tag{20}$$

The solution Φ of (15) satisfies

$$(A\Phi, \psi_i) = -(I_v, \psi_i) \quad (\forall \psi_i \in \mathcal{S}), \tag{21}$$

and the approximate Galerkin solution obtained by solving (19) satisfies

$$(A\phi, \psi_i) = -(I_v, \psi_i) \quad (\forall \psi_i \in \mathcal{S}). \tag{22}$$

Subtracting (21) from (22) yields

$$(A(\phi - \Phi), \psi_i) = (\phi - \Phi, \psi_i)_E = 0 \quad (\forall \psi_i \in \mathcal{S}). \tag{23}$$

The difference $\phi - \Phi$ denotes the error between the solution in the infinite dimensional space V and the $N+1$ dimensional space $\mathcal{S}$. Equation (23) states that the error is orthogonal to all basis functions spanning the space of possible Galerkin solutions. Consequently, the error is orthogonal to all elements in $\mathcal{S}$ and must therefore be the minimum error. Thus, the Galerkin approximation is an orthogonal projection of the true solution Φ onto the given finite dimensional space of possible approximate solutions. Therefore, the Galerkin approximation is the best approximation in the energy space E. Since the operator is positive definite, the approximate solution is unique. Assume for a moment there are two solutions, ϕ_1 and ϕ_2, satisfying

$$(A\phi_1, \psi_i) = -(I_v, \psi_i) \quad (A\phi_2, \psi_i) = -(I_v, \psi_i) \quad (\forall \psi_i \in \mathcal{S}) \tag{24}$$

respectively. Then, the difference yields

$$(A(\phi_1 - \phi_2), \psi_i) = 0 \quad (\forall \psi_i \in \mathcal{S}). \tag{25}$$

The function arising from subtracting one member from another member in $\mathcal{S}$ also belongs in $\mathcal{S}$; hence, the difference function can be expressed by the set of A orthogonal basis functions spanning $\mathcal{S}$:

$$\sum_{j=0}^{N} \Delta\phi_j (A(\psi_j, \psi_i)) = 0 \quad (\forall \psi_i \in). \tag{26}$$

When $i \neq j$, the terms vanish due to the basis functions being orthogonal with respect to A. Since A is positive definite,

$$(A\Phi_i, \Phi_i) > 0 \quad i = 0, \ldots, N. \tag{27}$$

Thus, $\Delta\phi_i = 0,\ i = 0, \ldots, N$, and by virtue of (26), $\delta\phi = 0$, such that $\phi_1 = \phi_2$. The identity contradicts the assumption of two distinct Galerkin solutions. This proves the solution is unique [179], [180].

B. The Finite Difference Method

Perhaps the most traditional way to solve (3) utilizes the finite difference approach. Finite differences usually discretize the solution domain Ω using a grid of uniform hexahedral elements (we note that there are several special techniques to handle curved boundaries) [181]. The coordinates of a typical grid point are $x = lh, y = mh, z = nh$

(l,m,n=integers), and the value of $\Phi(x,y,z)$ at a grid point is denoted by $\Phi_{l,m,n}$. Taylor's theorem can then be utilized to provide the difference equations. For example:

$$\Phi_{l+1,m,n} = (\Phi + h\frac{\partial \Phi}{\partial x} + \frac{1}{2}h^2\frac{\partial^2 \Phi}{\partial x^2} + \frac{1}{6}h^3\frac{\partial^3 \Phi}{\partial x^3} + \ldots)_{l,m,n}, \tag{28}$$

with similar equations for $\Phi_{l-1,m,n}, \Phi_{l,m+1,n}, \Phi_{l,m-1,n}, \ldots$. The finite difference representation of (3) is

$$\frac{\Phi_{l+1,m,n} - 2\Phi_{l,m,n} + \Phi_{l-1,m,n}}{h^2} + \frac{\Phi_{l,m+1,n} - 2\Phi_{l,m,n} + \Phi_{l,m-1,n}}{h^2} + \frac{\Phi_{l,m,n+1} - 2\Phi_{l,m,n} + \Phi_{l,m,n-1}}{h^2} = -I_{l,m,n}(v), \tag{29}$$

or, equivalently,

$$\Phi_{l+1,m,n} + \Phi_{l-1,m,n} + \Phi_{l,m+1,n} + \Phi_{l,m-1,n} + \Phi_{l,m,n+1} + \Phi_{l,m,n-1} - 6\Phi_{l,m,n} = -h^2 I_{l,m,n}(v). \tag{30}$$

If we define the vector $\mathbf{\Phi}$ to be $[\Phi_{1,1,1} \ldots \Phi_{1,1,N-1}; \ldots \Phi_{1,N-1,1} \ldots \Phi_{N-1,N-1,N-1}]^T$ to designate the $(N-1)^3$ unknown grid values, and pull out all the known information from (30), we can reformulate (3) by its finite difference approximation in the form of the matrix equation $A\mathbf{\Phi} = \mathbf{b}$, where $\mathbf{b}$ is a vector containing the sources and modifications due to the Dirichlet boundary condition.

Unlike the traditional Taylor's series expansion method, the Galerkin approach utilizes basis functions, such as linear piecewise polynomials, to approximate the true solution. For example, the Galerkin approximation to the sample problem (3) would require evaluating (19) for the specific grid formation and specific choice of basis function:

$$\int_\Omega (\sigma_x \frac{\partial \phi}{\partial x}\frac{\partial \psi_i}{\partial x} + \sigma_y \frac{\partial \phi}{\partial y}\frac{\partial \psi_i}{\partial y} + \sigma_z \frac{\partial \phi}{\partial z}\frac{\partial \psi_i}{\partial z}) d\Omega = -\int_\Omega I_v \psi_i d\Omega. \tag{31}$$

Difference quotients are then used to approximate the derivatives in (31). We note that if linear basis functions are utilized in (31), one obtains a formulation which corresponds exactly with the standard finite difference operator. Regardless of the difference scheme or order of basis function, the approximation results in a linear system of equations of the form $A\mathbf{\Phi} = \mathbf{b}$, subject to the appropriate boundary conditions.

Before going on, let's briefly give a simple two-dimensional example of the finite difference formulation for solving the bioelectric field problems governed by equation (3),

$$\nabla^2 \Phi = I(x,y). \tag{32}$$

The discretization of the two-dimensional (assumed regular with spacing $= h$) is given by

$$\nabla^2 \Phi_{i,j} = \frac{1}{h^2}[\Phi_{i+1,j} + \Phi_{i-1,j} + \Phi_{i,j+1} + \Phi_{i,j-1} - 4\Phi_{i,j}] = I(x_i, y_j). \tag{33}$$

Therefore, our solution can be written as,

$$-\Phi_{i+1,j} - \Phi_{i-1,j} - \Phi_{i,j+1} - \Phi_{i,j-1} + 4\Phi_{i,j} = -h^2 I(x_i, y_j), \tag{34}$$

where we have multiplied through by minus one. As an illustration of the kind of system this produces, we order the points in *natural* or *row wise* ordering and look at the system for $N = 2$:

$$\begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} \Phi_{11} \\ \Phi_{21} \\ \Phi_{12} \\ \Phi_{22} \end{pmatrix} = -h^2 \begin{pmatrix} I_{11} \\ I_{21} \\ I_{12} \\ I_{22} \end{pmatrix} + \begin{pmatrix} \Phi_{01} + \Phi_{10} \\ \Phi_{20} + \Phi_{31} \\ \Phi_{02} + \Phi_{13} \\ \Phi_{32} + \Phi_{23} \end{pmatrix} \tag{35}$$

Where, for the case with Dirichlet boundary conditions, the boundary data is given as

$$\begin{aligned} &\Phi_{0,j} = g(0, y_j), \quad \Phi_{N+1,j} = g(1, y_j), \quad j = 0, 1, \ldots N+1 \\ &\Phi_{i,0} = g(x_i, 0), \quad \Phi_{i,N+1} = g(x_i, 1), \quad j = 0, 1, \ldots N+1, \end{aligned} \tag{36}$$

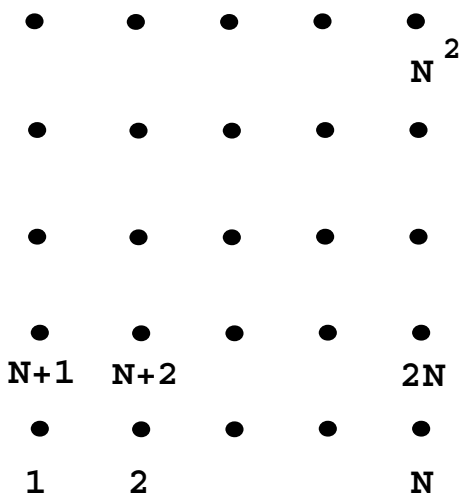

Fig. 15. An illustration of natural ordering for the interior grid points.

we can then generalize the bioelectric field problem with Dirichlet boundary conditions as the $N^2 \times N^2$ linear system,

$$\begin{pmatrix} T_N & -I_N & & \\ -I_N & \ddots & \ddots & \\ & \ddots & & -I_N \\ & & -I_N & T_N \end{pmatrix} \begin{pmatrix} \mathbf{\Phi}_1 \\ \mathbf{\Phi}_2 \\ \vdots \\ \mathbf{\Phi}_N \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 - h^2\mathbf{I}_1 \\ \mathbf{b}_2 - h^2\mathbf{I}_2 \\ \vdots \\ \mathbf{b}_N - h^2\mathbf{I}_N \end{pmatrix}, \tag{37}$$

where

$$\mathbf{\Phi}_i = (\Phi_{1i}, \ldots, \Phi_{Ni})^T, \quad \mathbf{I}_i = (I_{1i}, \ldots, I_{Ni})^T, \quad i = 1, \ldots, N \tag{38}$$

$$\mathbf{b}_1 = (\Phi_{01} + \Phi_{10}, \Phi_{20}, \ldots, \Phi_{N-1,0}, \Phi_{N,0} + \Phi_{N+1,1})^T \tag{39}$$

$$\mathbf{b}_i = (\Phi_{0i}, 0, \ldots, 0, \Phi_{N+1,1})^T, \quad i = 2, \ldots, N-1 \tag{40}$$

$$\mathbf{b}_N = (\Phi_{0,N} + \Phi_{1,N+1}, \Phi_{2,N+1}, \ldots, \Phi_{N-1,N}, \Phi_{N,N+1} + \Phi_{N+1,N})^T. \tag{41}$$

Here, I_N is the $N \times N$ identity matrix and T_N is the $N \times N$ tridiagonal matrix,

$$T_N = \begin{pmatrix} -4 & 1 & & & \\ 1 & -4 & \ddots & & \\ & & \ddots & & \\ & & \ddots & & 1 \\ & & & 1 & -4 \end{pmatrix}. \tag{42}$$

The vector $\mathbf{\Phi}$ consists of the unknown values at the nodes and is of dimension $N^2 \times 1$, while $\mathbf{b}$ is the $N^2 \times 1$ vector consisting of boundary data (with $4N - 4$ non-zero elements).

Thus the solution to our problem can be found by solving for the vector Φ,

$$\mathbf{\Phi} = A^{-1}\mathbf{b}. \tag{43}$$

B.1 Error Estimates for Finite Differences

Thus far in our application of the finite difference method to bioelectric field problems, we have concentrated on implementational details. In an intuitive manner, we know that one might decrease the error in a finite difference approximation by reducing the size of the elements. We also know that the discretization error when using the five-point formula for the finite difference approximation is of $O(h^2)$. What we have yet to do is discuss error estimates in

a more formal way. In general, analyzing errors is somewhat involved. Here, we just touch the surface of error analysis for finite difference methods. For a more in-depth analysis, look at the references, [182], [183], [181], [184].

We analyze the global error for the one-dimensional model problem along the lines of [182], [181], [185]:

$$\frac{d^2u}{dx^2} - p(x)\frac{du}{dx} - q(x)u = f(x), \tag{44}$$

with

$$q(x) \geq K > 0, \quad x \in (a,b), \tag{45}$$

with boundary conditions

$$u(a) = \alpha \quad u(b) = \beta. \tag{46}$$

We will approximate the solution to this equation using second-order accurate difference equations with mesh size h. We will denote the true solution as $u(x)$ and the approximate solution as $\hat{u}$. Thus, we have

$$\frac{1}{h^2}[\hat{u}(x_{i-1}) - 2\hat{u}(x_i) + \hat{u}(x_{i+1})] - \frac{p(x_i)}{2h}[\hat{u}(x_{i+1}) - \hat{u}(x_{i-1})] - q(x_i)\hat{u}(x_i) = f(x_i). \tag{47}$$

We define the **solution error** as $e(x_i) = \hat{u}(x_i) - u(ih)$, where $u(ih)$ denotes the exact solution at node x_i. If we substitute $u(ih)$ in for $\hat{u}$ in the previous equation, we obtain

$$\begin{aligned}\frac{1}{h^2}[u((i-1)h) - 2u(ih)) + u(i+1)h)] - \frac{p(ih)}{2h}[u((i+1)h) - u((i-1)h)] \\ -q(ih)u(ih) = f(ih) + O(h^2).\end{aligned} \tag{48}$$

Subtracting and noting that $x_i = ih$, we have

$$\frac{1}{h^2}[e(x_{i-1}) - 2e(x_i) + e(x_{i+1})] - \frac{p(x_i)}{2h}[e(x_{i+1}) - e(x_{i-1})] - q(x_i)e(x_i) = O(h^2). \tag{49}$$

Our boundary data are the Dirichlet conditions at a and b. We can then represent this last equation in terms of the matrix formula,

$$\begin{pmatrix} a_1 & -c_1 & & & \\ -b_2 & a_2 & -c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & & -b_{N-1} & a_{N-1} \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_{N-1} \end{pmatrix} = -h^2 \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-1} \end{pmatrix}, \tag{50}$$

where

$$a_i = [2 + h^2 q(x_i)] \quad b_i = [1 + \frac{h}{2}p(x_i)], \tag{51}$$

and

$$c_i = [1 - \frac{h}{2}p(x_i)], \quad f_i = O(h^2). \tag{52}$$

We can rewrite this set of equations as

$$\mathbf{Ae} = -\mathbf{h^2 f} \tag{53}$$

and call the vector $\mathbf{e}$ the **discretization** or **global error** of the difference approximation. Or, assuming that A^{-1} exists, our error is

$$\mathbf{e} = -\mathbf{h^2 A^{-1} f}. \tag{54}$$

This is the basic relationship between the global and local discretization errors.

Now what we want to determine is the behavior of A^{-1} as $h \to 0$. One difficulty here is that as $h \to 0$, n tends to ∞. However, Golub [185] has given a simple analysis for a specific case in which $p(x) = 0$ and

$$q(x) \geq \gamma > 0 \quad a \leq x \leq b. \tag{55}$$

We set our global error $e = \max|e_i|$ and our local error $f = \max|f_i| = O(h^2)$. From our error matrix equation and the condition on $q(x)$, we have

$$(2+\gamma h^2)|e_i| \leq 2e + h^2 f \quad i = 1, \ldots, n. \tag{56}$$

Since this equation must hold for all i, we obtain

$$(2+\gamma h^2)|e| \leq 2e + h^2 f \quad i = 1, \ldots, n. \tag{57}$$

Since $f = O(h^2)$, we conclude that

$$e \leq \frac{f}{\gamma} = O(h^2). \tag{58}$$

This says that the global discretization error is $O(h^2)$, provided that the local discretization error is $O(h^2)$. It can be shown that this result holds for the more general case, as well as for the important case when $q(x) = 0$. The analysis, however, gets somewhat involved.

As noted in [185], another way to look at the error is to use the eigenvalue $\lambda_{\min}(\mathbf{A})$ of $\mathbf{A}$ having the smallest magnitude to bound the global discretization error. Using the Euclidean norm, we find that

$$||\mathbf{Ae}||_2 = -h^2||\mathbf{f}||_2. \tag{59}$$

Since $||\mathbf{Ae}||_2 \geq |\lambda_{\min}(\mathbf{A})|||\mathbf{e}||_2$, we can obtain

$$-h^2||\mathbf{f}||_2 \geq |\lambda_{\min}(\mathbf{A})|||\mathbf{e}||_2. \tag{60}$$

Whenever $\mathbf{A}$ is nonsingular, $\lambda_{\min}(\mathbf{A}) \neq 0$, then we have the following bound on the global error [182]:

$$||\mathbf{e}||_2 \leq \frac{-h^2||\mathbf{f}||_2}{|\lambda_{\min}(\mathbf{A})|}. \tag{61}$$

If $\lambda_{\min}(\mathbf{A})$ approaches a nonzero value as $h \to 0$, then the norm of the global error $||\mathbf{e}||_2 \to 0$ at least as fast as the truncation error $||\mathbf{f}||_2$.

For more information on the finite difference method and corresponding error analysis and implementational details see [181], [185], [182], [186], [187], [188], [189]

C. The Finite Element Method

As we have seen above, in the classical numerical treatment for partial differential equations - the finite difference method - the solution domain is approximated by a grid of uniformly spaced nodes. At each node, the governing differential equation is approximated by an algebraic expression that references adjacent grid points. A system of equations is obtained by evaluating the previous algebraic approximations for each node in the domain. Finally, the system is solved for each value of the dependent variable at each node. In the finite element method, the solution domain can be discretized into a number of uniform or non-uniform finite elements that are connected via nodes. The change of the dependent variable with regard to location is approximated within each element by an interpolation function. The interpolation function is defined relative to the values of the variable at the nodes associated with each element. The original boundary value problem is then replaced with an equivalent integral formulation (such as (19)). The interpolation functions are then substituted into the integral equation, integrated, and combined with the results from all other elements in the solution domain. The results of this procedure can be reformulated into a matrix equation of the form $A\mathbf{\Phi} = \mathbf{b}$, which is subsequently solved for the unknown variable [190], [174].

The formulation of the finite element approximation starts with the Galerkin approximation, $(\sigma\nabla\Phi, \nabla\overline{\Phi}) = -(I, \overline{\Phi})$, where $\overline{\Phi}$ is our test function. We now use the finite element method to turn the continuous problems into a discrete formulation. First we discretize the solution domain, $\Omega = \cup_{e=1}^{E}\Omega_e$, and define a finite dimensional subspace, $V_h \subset V = \{\overline{\Phi} : \overline{\Phi}$ is continuous on $\Omega, \nabla\overline{\Phi}$ is piecewise continuous on $\Omega\}$. One usually defines parameters of the function $\overline{\Phi} \in V_h$ at node points $\alpha_i = \overline{\Phi}(x_i), i = 0, 1, \ldots, N$. If we now define the basis functions, $\psi_i \in V_h$, as linear continuous piecewise functions that take the value 1 at node points and zero at other node points, then we can represent the function $\overline{\Phi} \in V_h$ as

$$\overline{\Phi}(x) = \sum_{i=0}^{N} \alpha_i \Psi_i(x), \tag{62}$$

such that each $\overline{\Phi} \in V_h$ can be written in a unique way as a linear combination of the basis functions $\Psi_i \in V_h$. Now the finite element approximation of the original boundary value problem can be stated as,

$$\text{Find } \Phi_h \in V_h \text{ such that } (\sigma \nabla \Phi_h, \nabla \overline{\Phi}) = -(I_v, \overline{\Phi}). \tag{63}$$

Furthermore, if $\Phi_h \in V_h$ satisfies (63), then we have $(\sigma \nabla \Phi_h, \nabla \Psi_i) = -(I_v, \Psi_i)$ [191]. Finally, since Φ_h itself can be expressed as the linear combination

$$\Phi_h = \sum_{i=0}^{N} \xi_i \Psi_i(x) \qquad \xi_i = \Phi_h(x_i), \tag{64}$$

we can then write (63) as

$$\sum_{i=0}^{N} \xi_i (\sigma_{ij} \nabla \Psi_i, \nabla \Psi_j) = -(I_v, \Psi_j) \qquad j = 0, \ldots, N, \tag{65}$$

subject to the Dirichlet boundary condition. Then, the finite element approximation of (3) can equivalently be expressed as a system of N equations with N unknowns $\xi_i, \ldots, \xi_N$ (the electrostatic potentials, for example). In matrix form, the above system can be written as $A\xi = b$, where $A = (a_{ij})$ is called the global stiffness matrix and has elements $(a_{ij}) = (\sigma_{ij} \nabla \Psi_i, \nabla \Psi_j)$, while $b_i = -(I_v, \Psi_i)$ and is usually termed the load vector.

For volume conductor problems, A contains all of the geometry and electrical conductivity information of the model. The matrix A is symmetric and positive definite; thus, it is nonsingular and has a unique solution. Because the basis function differs from zero for only a few intervals, A is sparse - (only a few of its entries are nonzero). For large-scale models, the matrix A is very sparse, such that only a small percentage of each row is non-zero. This fact is exploited when storing and manipulating A as detailed in section V.

C.1 Application of the FE Method for 3-D Domains

We now illustrate the concepts of the finite element method by considering the solution of (3) using linear three-dimensional elements. We start with a 3D domain Ω, which represents the geometry of our volume conductor, and break it up into discrete elements to form a finite dimensional subspace, Ω_h. For 3D domains we have the choice of representing our function as either tetrahedra,

$$\tilde{\Phi} = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z, \tag{66}$$

or hexahedral,

$$\tilde{\Phi} = \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 z + \alpha_5 xy + \alpha_6 yz + \alpha_7 xz + \alpha_8 xyz. \tag{67}$$

For this example, we restrict our development to linear tetrahedra, knowing that it is easy to modify our formulae for hexahedra or to higher order basis functions. We take out a specific tetrahedra from our finite dimensional subspace and apply the previous formulations for the four vertices,

$$\begin{pmatrix} \tilde{\Phi}_1 \\ \tilde{\Phi}_2 \\ \tilde{\Phi}_3 \\ \tilde{\Phi}_4 \end{pmatrix} = \begin{pmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_4 & y_4 & z_4 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{pmatrix}, \tag{68}$$

or

$$\tilde{\mathbf{\Phi}}_\mathbf{i} = \mathbf{C}\alpha, \tag{69}$$

which define the coordinate vertices, and

$$\alpha = \mathbf{C^{-1}} \tilde{\mathbf{\Phi}}_\mathbf{i}, \tag{70}$$

which defines the coefficients. From equations (66) and (70) we can express $\tilde{\Phi}$ at any point within the tetrahedra,

$$\tilde{\Phi} = [1, x, y, z]\alpha = \mathbf{S}\alpha = \mathbf{S C^{-1}} \tilde{\mathbf{\Phi}}_\mathbf{i}, \tag{71}$$

or, most succinctly,

$$\tilde{\Phi} = \sum_i N_i \tilde{\Phi}_i. \tag{72}$$

$\tilde{\Phi}_i$ is the solution value at node i, and $\mathbf{N} = \mathbf{SC^{-1}}$ is the local *shape function* or *basis function*. This can be expressed in a variety of ways in the literature (depending, usually, on whether you are reading engineering or mathematical treatments of finite element analysis):

$$\Phi_j(N_i) = N_i(x, y, z) = f_i(x, y, z) \equiv \frac{a_i + b_i x + c_i y + d_i z}{6V} \tag{73}$$

where

$$6V = \begin{vmatrix} 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \\ 1 & x_3 & y_3 & z_4 \end{vmatrix} \tag{74}$$

defines the volume of the tetrahedra, V.

Now that we have a suitable set of basis functions, we can find the finite element approximation to our 3D problem. Our original problem can be formulated as

$$a(u, v) = (I_v, v) \quad \forall v \in \Omega, \tag{75}$$

where

$$a(u, v) = \int_\Omega \nabla u \cdot \nabla v d\Omega \tag{76}$$

and

$$(I_v, v) = \int_\Omega I_v \cdot v d\Omega. \tag{77}$$

The finite element approximation to the original boundary value problem is

$$a(u_h, v) = (I_v, v) \quad \forall v \in \Omega_h, \tag{78}$$

which has the equivalent form

$$\sum_{i=1}^{N} \xi_i a(\Phi_i, \Phi_j) = (I_v, \Phi_j), \tag{79}$$

where

$$a(\Phi_i, \Phi_j) = a(\Phi_i(N_j), \Phi_j(N_i)), \tag{80}$$

which can be expressed by the matrix and vector elements

$$(a_{ij}) = \int_{\Omega_E} (\frac{\partial N_i}{\partial x}\frac{\partial N_j}{\partial x} + \frac{\partial N_i}{\partial y}\frac{\partial N_j}{\partial y} + \frac{\partial N_i}{\partial z}\frac{\partial N_j}{\partial z}) d\Omega \tag{81}$$

and

$$I_i = \int_{\Omega_E} N_i I_v \, d\Omega. \tag{82}$$

Fortunately, the above quantities are easy to evaluate for linear tetrahedra. As a matter of fact, there are closed form solutions for the matrix elements (a_{ij}):

$$\int_{\Omega_h} N_1^a N_2^b N_3^c N_4^d \, d\Omega = 6V \frac{a!b!c!d!}{(a + b + c + d + 3)!}. \tag{83}$$

Therefore,

$$(a_{ij}) = \int_{\Omega_E} \frac{b_i b_j + c_i c_j + d_i d_j}{6V^2} \, d\Omega = \frac{b_i b_j + c_i c_j + d_i d_j}{6V}, \tag{84}$$

and, for the right hand side, we have, assuming constant sources,

$$I_i = \int_{\Omega_E} \frac{a_i + b_i x + c_i y + d_i z}{6V} I_v \, d\Omega = \frac{V I_v}{4}, \tag{85}$$

which have the compact forms

$$a_{ij}^{(n)} = \frac{1}{6V}(b_i^{(n)} b_j^{(n)} + c_i^{(n)} c_j^{(n)} + d_i^{(n)} d_j^{(n)}) \tag{86}$$

and

$$I_i^{(n)} = \frac{V I_v}{4} \quad \text{for constant sources.} \tag{87}$$

Now we add up all the contributions from each element into a global matrix and global vector:

$$\sum_{n=1}^{Nel} (a_{ij}^{(n)})(\xi_i) = (I_i^{(n)}), \tag{88}$$

where Nel is equal to the total number of elements in the discretized solution domain and i represents the node numbers (vertices). This yields a linear system of equations of the form $\mathbf{A\Phi} = \mathbf{b}$, where $\mathbf{\Phi}$ is our solution vector of voltages, $\mathbf{A}$ represents the geometry and conductivity of our volume conductor, and $\mathbf{b}$ represents the contributions from the current sources and boundary conditions.

For the finite difference method, it turns out that the Dirichlet boundary condition is easy to apply, while the Neumann condition takes a little extra effort. For the finite element method, it is just the opposite. The Neumann boundary condition

$$\nabla\Phi \cdot \mathbf{n} = 0 \tag{89}$$

is satisfied automatically within the Galerkin and variational formulations. This can be seen by using Green's divergence theorem,

$$\int_\Omega \nabla \cdot \mathbf{A} dx = \int_\Gamma \mathbf{A} \cdot \mathbf{n}\, dS, \tag{90}$$

and applying it to the left hand side of the Galerkin finite element formulation:

$$\begin{aligned} \int_\Omega \nabla v \cdot \nabla w \, d\Omega &\equiv \int_\Omega [\frac{\partial v}{\partial x_1}\frac{\partial w}{\partial x_1} + \frac{\partial v}{\partial x_2}\frac{\partial w}{\partial x_2}]\, d\Omega \\ = \int_\Gamma [v\frac{\partial w}{\partial x_1} n_1 + v\frac{\partial w}{\partial x_2} n_2]\, dS &- \int_\Omega v[\frac{\partial^2 w}{\partial x_1^2} + \frac{\partial^2 w}{\partial x_2^2}]\, d\Omega \\ &= \int_\Gamma v\frac{\partial w}{\partial n}\, dS - \int_\Omega v\nabla^2 w \, d\Omega. \end{aligned} \tag{91}$$

If we multiply our original differential equation, $\nabla^2\Phi = -I_v$, by an arbitrary test function and integrate, we obtain

$$(I_v, v) = -\int_\Omega (\nabla^2\Phi) v \, d\Omega = -\int_\Gamma \frac{\partial\Phi}{\partial n} v \, dS + \int_\Omega \nabla\Phi \cdot \nabla v \, d\Omega = a(\Phi, v), \tag{92}$$

where the boundary integral term, $\frac{\partial\Phi}{\partial n}$ vanishes and we obtain the standard Galerkin finite element formulation.

To apply the Dirichlet condition, we have to work a bit harder. To apply the Dirichlet boundary condition directly, one usually modifies the (a_{ij}) matrix and b_i vector such that one can use standard linear system solvers. This is accomplished by implementing the following steps:

Assuming we know the ith value of u_i,

1. Subtract from the ith member of the r.h.s. the product of a_{ij} and the known value of Φ_i (call it $\bar{\Phi}_i$); this yields the new right hand side, $\hat{b}_i = b_i - a_{ij}\bar{\Phi}_j$.
2. Zero the ith row and column of A: $\hat{a}_{ij} = \hat{a}_{ji} = 0$.
3. Assign $\hat{a}_{ii} = 1$.
4. Set the jth member of the r.h.s. equal to $\bar{\Phi}_i$.
5. Continue for each Dirichlet condition.
6. Solve the augmented system, $\hat{A}\Phi = \hat{b_v}$.

As one might imagine, the process of preparing the augmented system is somewhat time consuming if one has a large number of Dirichlet boundary conditions. Another, simpler, way of applying the Dirichlet boundary condition is to use

a *penalty method*. To explain this method, let's assume that node 5 of our finite element mesh is known to be on the boundary, $\Phi_5 = \bar{\Phi}_5$. From the system of linear equations, we have,

$$a_{51}\Phi_1 + a_{52}\Phi_2 + \ldots + a_{55}\Phi_5 + \ldots + a_{5N}\Phi_N = I_5 \tag{93}$$

To enforce the boundary condition at node 5 using the penalty method, we would add the term$\hat{P}\Phi_5$ on the left side and the term $\hat{P}\bar{\Phi}_5$ on the right side,

$$a_{51}\Phi_1 + a_{52}\Phi_2 + \ldots + (a_{55} + \hat{P})\Phi_5 + \ldots + a_{5N}\Phi_N = I_5 + \hat{P}\bar{\Phi}_5. \tag{94}$$

Then, if the number $\hat{P}$ is considerably larger than the terms, $a_{51}, a_{52}, \ldots, a_{5N}$ and I_5, the Dirichlet boundary condition is approximately satisfied. For example, if $\hat{P}$ is chosen to be

$$\hat{P} = N \cdot 10^p \cdot \max(a_{ij}), \tag{95}$$

then the node 5 has the value

$$\Phi_5 = \bar{\Phi}_5 + O(10^{-p}). \tag{96}$$

Thus, $\Phi_5 = \bar{\Phi}_5$ is satisfied to within an error 10^{-p}. A word of caution at this point. While this method usually works well when using direct solvers, it can lead to disastrous results when using some iterative methods, such as the penalty method, which can cause the system to become ill-conditioned.

For more information on the finite element method, consult [192], [193], [191], [194], [195], [196]

D. The Boundary Element Method

For bioelectric field problems with isotropic domains (and few inhomogeneities), another technique, called the boundary element method, may be utilized. This technique utilizes information only upon the boundaries of interest, and thus reduces the dimension of any field problem by one. For differential operators, the response at any given point to sources and boundary conditions depends only on the response at neighboring points. The FD and FE methods approximate differential operators defined on subregions (volume elements) in the domain; hence, direct mutual influence (connectivity) exists only between neighboring elements, and the coefficient matrices generated by these methods have relatively few non-zero coefficients in any given matrix row. As is demonstrated by Maxwell's laws [197], equations in differential forms can often be replaced by equations in integral forms; e.g. the potential distribution in a domain is uniquely defined by the volume sources and the potential and current density on the boundary. The boundary element method utilizes this fact by transforming the differential operator defined in the domain to integral operators defined on the boundary. In the boundary element method [198], [199], [200], only the boundary is discretized; hence, the mesh generation is considerably simpler for this method than for the volume methods. Boundary solutions are obtained directly by solving the set of linear equations; however, potentials and gradients in the domain can be evaluated only after the boundary solutions have been obtained. The boundary element method has a rich history in bioelectric field problems, [201], [202], [45], [203], [204]. We follow the BE formulations given in [179], [180] by Henneberg.

The boundary element formulation of (3) is obtained by choosing the anisotropic Green's function as the test function $\overline{\Phi} = 1/\sigma_e R$, where R is the distance function between the field point $\vec{p}$ and the source point $\vec{q}$:

$$R = \sqrt{\frac{(x_q - x_p)^2}{\sigma_x} + \frac{(y_q - y_p)^2}{\sigma_y} + \frac{(z_q - z_p)^2}{\sigma_z}} \tag{97}$$

and

$$\sigma_e = \sqrt{\sigma_x \sigma_y \sigma_z}. \tag{98}$$

Integration of $A\Phi = -I_v$ by parts twice yields Green's third identity:

$$\frac{1}{2}\Phi + H\Phi - G\frac{\partial \Phi}{\partial n} - \Phi_a = 0, \tag{99}$$

where

$$\Phi_a = (I_v, \overline{\Phi}) \tag{100}$$

and G and H are the single and double layer operators

$$G\frac{\partial\Phi}{\partial n} = \frac{1}{4\pi}\int_S \frac{\partial\Phi}{\partial n}\overline{\Phi}dS \tag{101}$$

and

$$H\Phi = \frac{1}{4\pi}\int_S \Phi\frac{\partial\overline{\Phi}}{\partial n}dS. \tag{102}$$

The Galerkin approximations to the weak form solutions Φ and $\frac{\partial\Phi}{\partial n}$ in (99) are expressed as:

$$\phi(\mathbf{s}) = \sum_{i=0}^{N}\phi_i\psi_i(\mathbf{s})\ ; \qquad \frac{\partial\phi}{\partial n}(\mathbf{s}) = \sum_{i=0}^{N}\left(\frac{\partial\phi}{\partial n}\right)_i\psi_i(\mathbf{s}) \tag{103}$$

where $\mathbf{s}$ denotes a parameterization of the surface. The functions $\phi(\mathbf{s})$ and $\frac{\partial\phi}{\partial n}(\mathbf{s})$ are members of the finite dimensional space V_h, and their coefficients ϕ_i and $\left(\frac{\partial\phi}{\partial n}\right)_i$ are determined by the set of linear equations

$$\sum_{j=0}^{N}\langle\frac{1}{2}\delta_{i,j} + H\psi_j,\psi_i\rangle\phi_j - \sum_{j=0}^{N}\langle G\psi_j,\psi_i\rangle\left(\frac{\partial\phi}{\partial n}\right)_j = \langle\Phi_a,\psi_i\rangle\ ; \quad i=0,\ldots,N, \tag{104}$$

where $\delta_{i,j}$ is the Kronecker delta function. The operator G is symmetric and positive definite; hence, if the potential is known on the boundary, (104) yields a symmetric coefficient matrix. The operator H is non-symmetric; hence, for the Neumann problem and problems with mixed boundary conditions, the coefficient matrix is non-symmetric. In general, the Galerkin formulation presented here does not satisfy (63) and only for the Dirichlet problem does the method classify as an orthogonal projection method [205].

The matrix coefficients in the Galerkin BEM requires the evaluation of double surface integrals, and the method is therefore more demanding on computing resources than the collocation method, which includes only single surface integrals. In the latter method, the basis function ψ_i is replaced by the Dirac delta function $\delta(|\vec{p}-\vec{p}_i|)$, and the inner product $\langle\phi_1,\phi_2\rangle$ is replaced by the bilinear form $(\phi_1,\phi_2)_B = \int_S\phi_1\phi_2 dS$. The latter is required since the Dirac delta function is not square integrable and consequently does not belong in the space V_h [205]. The collocation formulation equivalent to the Galerkin formulation in (104) is obtained:

$$\begin{aligned}\sum_{j=0}^{N}\left(\frac{1}{2}\delta_{i,j} + H\psi_j,\delta(|\vec{p}-\vec{p}_i|)\right)_B\phi_j \quad &- \quad \sum_{j=0}^{N}\left(G\psi_j,\delta(|\vec{p}-\vec{p}_i|)\right)_B\left(\frac{\partial\phi}{\partial n}\right)_j \\ &= \quad \left(\Phi_a,\delta(|\vec{p}-\vec{p}_i|)\right)_B\ ; \quad i=0,\ldots,N;\end{aligned} \tag{105}$$

hence, carrying out the outer integrations yields:

$$\sum_{j=0}^{N}\left(\frac{1}{2}\delta_{i,j} + H\psi_j\right)_i\phi_j - \sum_{j=0}^{N}\left(G\psi_j\right)_i\left(\frac{\partial\phi}{\partial n}\right)_j = (\Phi_a)_i\ ; \quad 0=1,\ldots,N, \tag{106}$$

where $(\ )_i$ denotes the ith field point. The collocation method in (106) is a non-orthogonal projection method [205]; hence, in general, the collocation method is less accurate than the Galerkin method. Since the latter method requires the evaluation of double surface integrals, it is used only if the increased accuracy is essential. We will therefore discuss only the implementation of the collocation method.

Good introductory general treatments of the boundary element technique include [206], [198], [207], [200]. A wealth of information regarding texts on boundary element methods can be obtained from Computational Mechanics Publications Inc. [208].

E. Comparison of Methods

We now give an abbreviated summary of the applicability of each method in solving different types of bioelectric field problems. As outlined above, the FD, FE, and BE methods can all be used to approximate the boundary value problems

that arise in biomedical research problems. The choice depends on the nature of the problem. The FE and FD methods are similar in that the entire solution domain must be discretized, while with the BE method only the bounding surfaces must be discretized. For regular domains, the FD method is generally the easiest method to code and implement, but the FD method usually requires special modifications to define irregular boundaries, abrupt changes in material properties, and complex boundary conditions. While typically more difficult to implement, the BE and FE methods are preferred for problems with irregular, inhomogeneous domains and mixed boundary conditions. The FE method is superior to the BE method for representing nonlinearity and true anisotropy, while the BE method is superior to FE method for problems where only the boundary solution is of interest or where solutions are wanted in a set of highly irregularly spaced points in the domain. Because the computational mesh is simpler for the BE method than for the FE method, the BE program requires less bookkeeping than a FE program. For this reason BE programs are often considered easier to develop than FE programs; however, the difficulties associated with singular integrals in the BE method are often highly underestimated. In general, the FE method is preferred for problems where the domain is highly heterogeneous, whereas the BE method is preferred for highly homogeneous domains.

Recent work by Bradley and Pullan has focused on using combining finite element and boundary element techniques to model the human thorax [209], [210], [211]. In their model, the finite element technique is used for modeling the anisotropic skeletal muscle while the boundary element method is used to model the isotropic tissues. It should also be noted that Hunter and Pullan have developed finite element method based upon using cubic Hermite polynomials as basis functions. This technique allows one to model complicated geometries with many fewer elements [79], [80], [212], [213].

V. Solution Methods and Computational Considerations

A. Renumbering and Storage Schemes

As we've just discussed, the finite element approximation of an elliptic bioelectric field problem governed by Poisson's equation yields a matrix A, which is symmetric, positive definite, and sparse. Now, let's look at the effects of these properties when we try to solve the resulting global linear system, $A\Phi = b$. Before doing any optimization whatsoever, we have an $N \times N$ linear system to solve, where N represents the number of unknowns (the values at the nodes we are trying to determine). Let's say we have a system of 10,000 unknowns (which is fairly moderate for a 3D finite element calculation). To store the entire system, we will need $10,000^2 \times (4 \text{ or } 8 \text{ for single or double precision reals}) = 400 \text{ or } 800Mb$. If we note that the system is symmetric so that we need store only its upper or lower triangular section, we reduce our storage needs significantly. However, our storage requirements for the matrix A are still on the order of N^2, namely, $((10,000 + 1) \times 10,000)/2 \approx 200 \text{ or } 400Mb$. We can still gain *significant* storage savings, though, by exploiting the sparseness of A. In general, there are two basic ways to reduce the necessary storage requirements: *renumbering* techniques and *sparse storage* methods.

Renumbering techniques are aimed primarily at direct methods. They work by minimizing the bandwidth of the matrix so that it can be stored in an $N \times M$ rectangular matrix, where M is the size of the bandwidth. The basic idea of minimizing the bandwidth is a simple one. Defining B as the bandwidth and D_i as the largest distance between the diagonal term of row i and any other term j on that same row, to minimize the storage we would minimize

$$B = \sum_i D_i \quad \text{where } D_i = \max[i - j]. \tag{107}$$

One of the most popular ways of minimizing the bandwidth was put forth in the form of a simple and effective algorithm in a paper by Cuthill and McKee [214]. The algorithm is outlined in four steps:

1. Search for the connectivities between variables (nodes). This requires the creation of a "table" that includes the number of connections each node has to its neighbors and also lists the connections to other variables (nodes).
2. Start with the least connected variable and call it 1. The least connected of those variables connected to what you've called 1 are given the next value, 2. The other variables connected to what you've changed to 1 are called 3 and so on.
3. Continue this process for any new node called 2 and its connections.
4. Continue this process until all the nodes are renumbered.

In the Cuthill-McKee algorithm, one can recognize *level sets* within the renumbering. The first set is the starting node. The second set comprises those nodes connected directly to the starting node. The third set consists of all nodes whose shortest route to the starting node involves just two links, etc. The resulting matrix can be specified as block tridiagonal because the nodes at any level are connected to nodes in no more than three adjacent levels. An optimal renumbering scheme will keep the blocks as small as possible (in graph theory, such a scheme is called a *breadth first search*).

The block structure of the Cuthill-McKee coefficient matrix is fixed by the choice of the starting node, so it is important to select a good one. While no one has discovered a sure way to find the optimal choice, nodes in the highest level of a Cuthill-Mckee ordering generally make good choices for the starting node for a *second* renumbering. This implies that the algorithm should always be run twice, each time restarting with one of the last numbered nodes to achieve the best ordering.

It was recognized by George [215] that if the Cuthill-McKee numbering is reversed, the bandwidth is often reduced beyond the savings achieved by the direct Cuthill-McKee algorithm. Here, reverse is used in the sense that element (i, j) moves to $(n - j + 1, n - i + 1)$. Lui and Sherman [216] proved that the reverse Cuthill-McKee algorithm could not give a less efficient numbering and often gave a more efficient renumbering than the direct version.

To estimate the effects of applying the Cuthill-McKee algorithm, one should calculate the cost of renumbering the nodes and balance this cost against the savings in terms of storage space. It is interesting to note that while there isn't a favorable balance when using Cuthill-McKee with many iterative solvers, it can be proven that renumbering can save a significant amount of CPU time when using certain preconditioned conjugate gradient algorithms. This is especially true when the preconditioning methods are employed using approximate inverses having similar profiles to the original matrix A [193]. For other bandwidth reduction schemes, see [217].

In *sparse storage* schemes, the bandwidth remains the same, but an algorithm is implemented that stores only nonzero elements along with pointers back to the positions of the elements. As we saw before, using the fact that the matrix is

symmetric still leads to a storage requirement that is proportional to N^2. With sparse storage techniques, we can reduce storage requirements until they are proportional to N (at least for first order finite element meshes). One of the first, and still widely used, techniques for efficiently storing global finite element matrices is called the Compressed Sparse Row (CSR) storage, after the method first put forth as the Yale Sparse Storage Scheme [218]. This is one of many row- or column-based storage techniques (others include Modified Compressed Sparse Row format (MSR), the Symmetric Sparse Skyline (SSS) format, Block Sparse Row format (BSR), etc.) [219], [220]. Basically, the idea is never to store a zero and always to efficiently point to the non-zero entries. The CSR sparse storage scheme breaks the global matrix into three vector arrays:

• $a(*)$ = a real valued array of non-zero elements. The array is $1 \times NZ$ in length, where NZ is the number of non-zero elements.

• $ja(*)$ = an integer valued array of column numbers. $ja(*)$ is the column number of $a(*)$. The vector is an integer array of length $1 \times NZ$.

• $ia(*)$ = a mapping vector to denote the starting locations of each of the blocks. $ia(*)$ tells how many non-zero elements there are in each row. The vector is an integer array of length $N + 1$.

As an example of how much disk space such methods can save, let's store the values from our earlier 3D finite element example. We used first order linear tetrahedra in which the unknowns were at the vertices of the tetrahedra. For numerical reasons, we wish these elements to be as equilateral as possible; thus, we could reasonably say there were six elements connected to each node (omitting the boundary points for now). Probabilistically, of these six, three will have smaller numbers of connections than that node and therefore the corresponding coefficients will be three in number. Therefore, including the diagonal, we would probably average about four numbers per row to store. This yields a total of $4N$ numbers that need to be stored, in addition to numbers that correspond to their location. This would yield approximately $4N$ real numbers corresponding to the non-zero elements, $4N$ integers corresponding to their row location, and $4N$ integers corresponding to their column location. Therefore, if we had, say, $N = 10,000$ unknowns, we would need $40,000$ real numbers and $80,000$ integers for a total of $480Kb$. To store the entire $N \times N$ matrix of reals, we would need $800Mb$. Even if we took advantage of the symmetry of the matrix, we would still need $400Mb$. Thus, when applying a CSR type method, we realize an order of magnitude reduction in storage requirements over simply storing the entire matrix [174], [219], [220].

B. Solution Techniques

B.1 Sparse Matrix Methods

Now that we have reduced the bandwidth of our matrix A and stored the data, we still need to solve the linear equations $\mathbf{A\Phi} = \mathbf{b}$ to obtain the final solution. To solve these (often very large) set of equations, we need to consider both the numerical and computational aspects of the solution technique(s) we choose. To get some insight into the problem, let's look at solving a system that was created by a finite difference approximation of (3). As we noted before, the structure of the A matrix is symmetric, positive definite, and sparse. Let's assume we used a five point method, in which there will not be any more than five non-zero elements in any given row (some will have three or four near the top and bottom). To give an idea of how size affects our problem, let's consider that N is of moderate size, $N = 15,000$, so that there are 1.5×10^4 unknowns and the resulting system of equations is $15,000 \times 15,000$. Let's say we decide to use a brute-force Gaussian elimination method to solve the system of equations (and to show why this is not a good idea). Gaussian elimination requires on the order of N^3 operations to solve an $N \times N$ system of equations. Therefore, if we have a $10^4 \times 10^4$ linear system, we must perform on the order of 3.375×10^{12} operations to solve the system. At a rate of 100×10^6 operations per second (100 MFLOPs, the speed of an average workstation), it would take almost nine hours to solve such a problem. Furthermore, if we were to extend this into a larger system of, say, size 10^6, the solution would require 10^{18} operations. Solving this system would take several months on even the fastest existing supercomputer (currently a 140 processor Fujitsu with a peak performance of 170 GFLOPs on the Linpack benchmarks). This is why one wouldn't use a sequential version of Gaussian elimination on a full version of the problem. Recent work by Saad [221], [219], [222] on the incomplete LU method with thresholding (ILUT) has shown promise for efficiently solving large sparse linear systems. A ILUT preconditioning algorithm coupled with a parallel GMRES solver computed a solution to a $N = 15,000$ system of equations in less than ten minutes on a CRAY 2 computer. Being mindful of numerical algorithms and computational considerations is of the utmost importance when

solving large-scale bioelectric field problems.

There are a plethora of available techniques for the solutions of systems of linear equations. The solution techniques can be broadly categorized as *direct* and *iterative* solvers. Direct solvers include Gaussian Elimination and LU Decomposition, while iterative methods include Jacobi, Gauss-Seidel, Successive Overrelaxation (SOR), and Krylov Subspace (which includes Conjugate Gradients) methods, among others. The choice of the particular solution method is highly dependent upon the approximation technique employed to obtain the linear system, upon the size of the resulting system, and upon accessible computational resources. For example, the linear system resulting from the application of the FD or FE method will yield a matrix $\mathbf{A}$ that is symmetric, positive definite, and sparse. The matrix resulting from the FD method will have a specific band-diagonal structure that is dependent on the order of difference equations one used to approximate the governing equation. The matrix resulting from the FE method will be exceedingly sparse, so that only a few of the off diagonal elements will be non-zero. The application of the BE method, on the other hand, will yield a matrix $\mathbf{A}$ that is dense and non-symmetric and thus requires a different choice of solver.

The choice of the optimal solver is further complicated by the size of the system versus access to computational resources. Sequential direct methods are usually confined to single workstations; thus, the size of the system should fit into memory for optimal performance. Sequential iterative methods can be employed when the size of the system exceeds the available memory of the machine; however, one pays a price in terms of performance, as direct methods are usually much faster than iterative methods. In many cases, the size of the system exceeds the computational capability of a single workstation and one must resort to the use of clusters of workstations and/or parallel computers.

When using parallel algorithms, one much be cognizant that the standard algorithms/packages that one may have used to perform numerical linear algebra operations, such as Linpack, may not scale well on parallel systems. For example, if one runs the Linpack Cholesky decomposition routine on a distributed memory parallel computer, one will not see much in the way of an increase in performance as the Linpack Cholesky algorithm uses vector-vector operations, which require $O(n)$ floating point operations (flops) and $O(n)$ memory references. Therefore the ratio of flops to memory references will remain constant no matter now many processors are used. On the other hand, if one were to use the Cholesky decomposition routine in Lapack, one find that the algorithm scales proportional to the number of available processors. This is because Lapack uses matrix-matrix operations in the Cholesky routine. Matrix-matrix operations require $O(n^{3)}$ flops and $O(n^2$ memory references. The ratio of the flops to memory references is thus proportional to the size of the system (and to the amount of available memory). Public domain software that takes advantage of parallel architectures include the PETSc, Aztec, and Lapack packages (URLs can be found in the section on Software). A particularly useful reference that covers both the theoretical and practical aspects of parallel numerical linear algebra is the text by Demmel [223].

While new and improved methods continue to appear in the numerical analysis literature, the author's studies comparing various solution techniques for direct and inverse bioelectric field problems have resulted in the conclusion that the Krylov Subspace methods (specifically, the preconditioned conjugate gradient methods) and Multigrid methods are among the best overall performers for volume conductor problems computed on single workstations. Specifically, the Krylov subspace methods work best when coupled with incomplete factorization preconditioning methods such as the Incomplete LU factorization with thresholding (ILUT) [222] or the Incomplete Cholesky Conjugate Gradient (ICCG). These methods work well for equations that are the result of a FE approximation.[3] When clusters of workstations and/or parallel architectures are considered, the choice is often less clear. For use with some high performance architectures that contain large amounts of memory, parallel direct methods such as LU decomposition become attractive; however, preconditioned conjugate gradient methods still perform well. To date, a small number of researchers solving bioelectric field problems have targeted high-performance architectures [177], [180], [12], [77], [224], [13]. A specific exception, however, are the papers describing large-scale solutions of defibrillation field problems on parallel architectures by the group at New Mexico [225], [29], [91], [226], [92]

B.2 Dense Matrix Methods

As noted previously, approximation of (3) by the boundary element method yields a matrix A that is dense and non-symmetric. Because the BE method generates these often large and dense matrices, the storage and solution techniques

[3]This is specifically for the FE method applied to elliptic problems. Such problems yield a matrix that is symmetric and positive definite. The Cholesky decomposition only exists for symmetric, positive definite matrices.

should be designed before designing the algorithm for computing the matrix coefficients. One reason for this is that for large-scale systems, it is usually not possible to store the coefficient matrices in memory for subsequent assemblage of the system matrix.

By approximating the surface with quadratic quadrilateral Lagrange elements containing nine interpolation nodes [206], [198], the surface integrals in (106) can be approximated as follows:

$$\sum_{j=0}^{N}(G\psi_j)_i\left(\frac{\partial\phi}{\partial n}\right)_j \approx \sum_{j=0}^{N_e}\sum_{l=1}^{9}\left[\int_{-1}^{1}\int_{-1}^{1}\frac{\psi_l(s,t)}{4\pi\sigma_e R_{i,j}(s,t)}|\vec{J}_j(s,t)|dsdt\right]\left(\frac{\partial\phi}{\partial n}\right)_{j,l} \tag{108}$$

$$\approx \sum_{j=0}^{N_e}\sum_{l=1}^{9}\left[\sum_{m=1}^{N_s}\sum_{n=1}^{N_t}\frac{W_m W_n\psi_l(s_m,t_n)}{4\pi\sigma_e R_{i,j}(s_m,t_n)}|\vec{J}_j(s_m,t_n)|\right]\left(\frac{\partial\phi}{\partial n}\right)_{j,l}, \tag{109}$$

and

$$\sum_{j=0}^{N}(H\psi_j)_i\phi_j \approx \sum_{j=0}^{N_e}\sum_{l=1}^{9}\left[\int_{-1}^{1}\int_{-1}^{1}\psi_l(s,t)\frac{-\vec{r}_{i,j}(s,t)\cdot\vec{J}_j(s,t)}{4\pi\sigma_e R_{i,j}^3(s,t)}dsdt\right]\phi_{j,l} \tag{110}$$

$$\approx \sum_{j=0}^{N_e}\sum_{l=1}^{9}\left[\sum_{m=1}^{N_s}\sum_{n=1}^{N_t}\frac{W_m W_n\psi_l(s_m,t_n)}{4\pi\sigma_e R_{i,j}^3(s_m,t_n)}\left(-\vec{r}_{i,j}(s_m,t_n)\cdot\vec{J}_j(s_m,t_n)\right)\right]\phi_{j,l}, \tag{111}$$

where N_e denotes the number of elements and $\vec{J}(s,t)$ denotes the Jacobian associated with the transformation from the Cartesian coordinate system to the curvilinear (s,t) coordinate system. In (109) and (111), further approximation is introduced by employing the Gaussian Quadrature integration scheme [227]. The number of Gauss points in the s and t directions are denoted N_s and N_t, respectively, and the Gauss points are denoted by s_m and t_n, respectively.

The mathematical expressions in (109) and (111) for the matrix coefficients contain five indices (i,j,l,m,n). The j index divides the matrix into slabs each containing 9 columns associated with the same element. Index l enumerates the columns in a single slab and index i enumerates the elements in each column. Hence j and l are partitioned whereas i is sequential from 0 to N. Each matrix element is the result of a double summation (indices m and n) of terms in the Gaussian Quadrature scheme. The Gaussian double summation can be reduced to a single summation by merging the matrices of Gauss point and weights into sequential arrays.

If the summations are performed in the order written in (109) and (111) (do loops nested in the order i,j,l,m,n) each matrix coefficient is finished before the next one is computed. This order of nested summations corresponds to computing the matrix coefficients in a row-wise order. A more efficient nesting of the do loops would be j,m,n,i with the l loop unrolled (written out as 9 separate statements) inside the i loop. The j loop is chosen as the outer loop so that the Jacobian needs to be computed only once for an entire matrix slab. The i loop is chosen as the inner loop in order to obtain a long range of the index of the inner do loop.

Because the BEM employs integral operators, the method results in a full, nonsymmetric matrix A. Although the dimension of a BEM matrix is smaller than that of the FD and FE methods, the number of non-zeroes to be stored is often of equivalent orders of magnitude. For small scale problems, the two coefficient matrices and the assembled system matrix are stored in memory (or on disk). If the boundary conditions change, the program has only to go back and reassemble the system matrix from the two coefficient matrices. For large-scale problems, storage of both coefficient matrices and the assembled system matrix may exceed both memory and disk capacity. Considerable savings on storage is obtained by assembling the system matrix ad hoc without saving the coefficient matrices; unfortunately the matrix coefficients must be recomputed if the boundary conditions are changed.

Solution techniques based on Preconditioned Bi-Conjugate Gradient (BCG) methods are often utilized for the dense systems generated by BE methods.

A discussion of solution methods, storage methods, and parallel computing methods for the solution of biomedical field problems could fill several texts. Thus, the reader is directed to the following references on scientific computing, [185], [228], [229], [230], [231], [225], [232], [233], [234], [235], [236], [237].

VI. Adaptive Methods

Thus far we have discussed how one formulates a bioelectric field problem, discretizes a volume conductor geometry, and finds an approximate solution. We are now faced with answering the difficult question pertaining to the accuracy of our solution. Without reference to experimental data, how can we judge the validity of our solutions? To give yourself an intuitive feel for the problem (and possible solution), consider the approximation of a two-dimensional region discretized into triangular elements. We'll apply the finite element method to solve Laplace's equation in the region.

First, consider the approximation of the potential field $\Phi(x, y)$ by a two dimensional Taylor's series expansion about a point (x, y):

$$\Phi(x+h, y+k) = \Phi(x,y) + [h\frac{\partial\Phi(x,y)}{\partial x} + k\frac{\partial\Phi(x,y)}{\partial y}] + \frac{1}{2!}[h^2\frac{\partial^2\Phi(x,y)}{\partial^2 x} + 2hk\frac{\partial^2\Phi(x,y)}{\partial x\partial y} + k^2\frac{\partial^2\Phi(x,y)}{\partial^2 y}] + \dots \tag{112}$$

where h and k are the maximum x and y distances within an element. Using the first two terms (up to first order terms) in the above Taylor's expansion, we can obtain the standard linear interpolation function for a triangle:

$$\frac{\partial\Phi(x_i,y_i)}{\partial x} = \frac{1}{2A}[\Phi_i(y_j - y_m) + \Phi_m(y_i - y_j) + \Phi_j(y_m - y_i)], \tag{113}$$

where A is the area of the triangle. Likewise, one could calculate the interpolant for the other two nodes and discover that

$$\frac{\partial\Phi(x_i,y_i)}{\partial x} = \frac{\partial\Phi(x_j,y_j)}{\partial x} = \frac{\partial\Phi(x_m,y_m)}{\partial x} \tag{114}$$

is constant over the triangle (and thus so is the gradient in y as well). Thus, we can derive for the triangle the standard linear interpolation formulas that represent the first two terms of the Taylor's series expansion. This means that the error due to discretization (from using linear elements) is proportional to the third term of the Taylor's expansion:

$$\epsilon \approx \frac{1}{2!}[h^2\frac{\partial^2\Phi(x,y)}{\partial^2 x} + 2hk\frac{\partial^2\Phi(x,y)}{\partial x\partial y} + k^2\frac{\partial^2\Phi(x,y)}{\partial^2 y}], \tag{115}$$

where Φ is the exact solution. We can conjecture, then, that the error due to discretization for first order linear elements is proportional to the second derivative. If Φ is a linear function over the element, then the first derivative is a constant and the second derivative is zero and there is no error due to discretization. This implies that the gradient must be constant over each element. If the function is not linear, or the gradient is not constant over an element, the second derivative will not be zero and is proportional to the error incurred due to "improper" discretization. Examining (115), we can easily see that one way to decrease the error is to decrease the size of h and k. As h and k go to zero, the error tends to zero as well. Thus, decreasing the mesh size in places of high errors due to high gradients decreases the error. As an aside, we note that if one divides equation (15) by hk, one can also express the error in terms of the elemental aspect ratio $\frac{h}{k}$, which is a measure of the relative shape of the element. It is easy to see that one must be careful to maintain an aspect ratio as close to unity as possible.

The problem with the preceding heuristic argument is that one has to know the exact solution *a priori* before one can estimate the error. This is certainly a drawback considering we are trying to accurately approximate Φ.

A. Convergence of a Sequence of Approximate Solutions

Let's try to quantify our error a bit further. When we consider the preceding example, it seems to make sense that if we increase the number of degrees of freedom we use to approximate our function, the accuracy must approach the true solution. That is, we would hope that the sequence of approximate solutions will *converge* to the exact solution as the number of degrees of freedom (DOF) increases indefinitely:

$$||\Phi(x) - \tilde{\Phi}_n(x)|| \to 0 \quad \text{as } N \to \infty. \tag{116}$$

This is a statement of *pointwise convergence*. It describes the approximate solution as approaching arbitrarily close to the exact solution at each point in the domain as the number of DOF increases.

Measures of convergence often depend on how the *closeness* of measuring the distance between functions is defined. Another common description of measuring convergence is *uniform convergence*, which requires that the maximum value of $||\Phi(x) - \tilde{\Phi}_n(x)||$ in the domain vanish as $N \to \infty$. This is stronger than pointwise convergence as it requires a uniform rate of convergence at every point in the domain. Two other commonly used measures are *convergence in energy* and *convergence in mean*, which involve measuring an *average* of a function of the pointwise error over the domain [238].

In general, proving pointwise convergence is very difficult except in the simplest cases, while proving the convergence of an averaged value, such as energy, is often easier. Of course, scientists and engineers are often much more interested in assuring that their answers are accurate in a pointwise sense than in an energy sense because they typically want to know values of the solution $\Phi(x)$ and gradients $\nabla\Phi(x)$ at specific places.

One intermediate form of convergence is called the *Cauchy convergence*. Here, we require the sequences of two different approximate solutions to approach arbitrarily close to each other:

$$||\Phi_m(x) - \tilde{\Phi}_n(x)|| \to 0 \quad \text{as } M, N \to \infty. \tag{117}$$

While the pointwise convergence expression would imply the previous equation, it is important to note that the Cauchy convergence does not imply pointwise convergence, as the functions could converge to an answer other than the true solution.

While we cannot be assured of pointwise convergence of these functions for all but the simplest cases, there do exist theorems that ensure that a sequence of approximate solutions must converge to the exact solution (assuming no computational errors) if the basis functions satisfy certain conditions. The theorems can only ensure convergence in an average sense over the entire domain, but it is usually the case that if the solution converges in an average sense (energy, etc.), then it will converge in the pointwise sense as well.

B. Energy Norms

The error in energy, measured by the *energy norm*, is defined in general as [239], [240], [241]

$$||e|| = (\int_\Omega e^T L e \, d\Omega)^{\frac{1}{2}}, \tag{118}$$

where $e = \Phi(x) - \tilde{\Phi}_n(x)$ and L is the differential operator for the governing differential equation (i.e. it contains the derivatives operating on $\Phi(x)$ and any function multiplying $\Phi(x)$). For physical problems, this is often associated with the energy density.

Another common measure of convergence utilizes the L_2 norm. This can be termed the average error and can be associated with errors in any quantity. The L_2 norm is defined as

$$||e||_{L_2} = (\int_\Omega e^T e \, d\Omega)^{\frac{1}{2}}. \tag{119}$$

While the norms given above are defined on the whole domain, one can note that the square of each can be obtained by summing element contributions,

$$||e||^2 = \sum_{i=1}^{M} ||e||_i^2, \tag{120}$$

where i represents an element contribution and m the total element number. Often, for an *optimal* finite element mesh, one tries to make the contributions to this square of the norm equal for all elements.

While the absolute values given by the energy or L_2 norms have little value, one can construct a relative percentage error that can be more readily interpreted:

$$\eta = \frac{||e||}{||\Phi||} \times 100. \tag{121}$$

This quantity, in effect, represents a weighted RMS error. The analysis can be determined for the whole domain or for element subdomains. One can use it in an adaptive algorithm by checking element errors against some predefined tolerance, η_0, and increasing the DOF only of those areas above the predefined tolerance.

To date, few researchers have applied adaptive techniques to bioelectric field problems [34], [82]. We now proceed to describe an adaptive method procedure applied to forward and inverse problems in electrocardiography by Schmidt et al. [82].

Schmidt et al. found (as have researchers in other fields) that, by using *a posteriori* estimates from the finite element approximation of the governing equations, one can locally refine the mesh discretization and reduce the errors in the direct solution. It has been assumed — and their findings support this notion — that improving the accuracy of the direct solution also improves the subsequent inverse solution. The novel aspect of the approach is that it uses local approximations of the error in the numerical solutions to drive an automatic adaptive mesh refinement. The basis of the error estimations comes from recent research on the finite element method [194].

The essential feature of the finite element method is that the approximate forms of the scalar field satisfy the governing equation in each element in some weighted sense. In *h-refinement*, the size of the element is reduced, while in *p-refinement* the order of the basis function is increased. Using either h- or p-refinement methods one can increase the accuracy of the approximation [242]. While either approach can be applied globally, computational limits make it more efficient to apply refinement locally, to regions where it is deemed most beneficial. One way to monitor the overall effect of refinement is to compute the total energy, which must converge monotonically if refinement is progressing effectively.

Improvements via an adaptive h-refinement technique can be implemented by using an estimate of the element energy error, such as the one described here derived from methods suggested by Lewis [243].

The error in the potential, e_Φ, is defined as the difference between the exact potential, Φ, and the calculated potential, $\hat{\Phi}$:

$$e_\Phi = \Phi - \hat{\Phi}. \tag{122}$$

Similarly, the error in the gradient of the potential (electric field), q, is

$$e_q = q - \hat{q}, \tag{123}$$

where $q = \nabla\Phi$ and $\hat{q} = \nabla\hat{\Phi}$. The error norm is defined as:

$$||e_q|| = (\int_\Omega (\nabla\Phi - \nabla\hat{\Phi})^T \sigma (\nabla\Phi - \nabla\hat{\Phi}) d\Omega)^{\frac{1}{2}}. \tag{124}$$

Zienkiewicz [244], [245] has shown that

$$\int_\Omega (\nabla\Phi)^T \sigma (\nabla\hat{\Phi}) d\Omega = \int_\Omega (\nabla\hat{\Phi})^T \sigma (\nabla\Phi) d\Omega = \int_\Omega (\nabla\hat{\Phi})^T \sigma (\nabla\hat{\Phi}) d\Omega. \tag{125}$$

Using this result, (124) becomes

$$||e_q||^2 = \int_\Omega (\nabla\Phi)^T \sigma (\nabla\Phi) d\Omega - \int_\Omega (\nabla\hat{\Phi})^T \sigma (\nabla\hat{\Phi}) d\Omega, \tag{126}$$

or, equivalently,

$$||e_q||^2 = ||q||^2 - ||\hat{q}||^2. \tag{127}$$

Here, $||q||^2$ is a measure of the total energy in the domain. Using this, the percentage error, η, can be defined to be

$$\eta = \frac{||e_q||}{||q||} \cdot 100\%. \tag{128}$$

This error measure gives a ratio of the energy difference and the total energy, which is in effect a percentage error for each element.

Here we have used linear basis functions to describe the variation of the potential in each tetrahedral element. At every point in the mesh, the potential is uniquely specified. However, the gradient is not specified at every point in space

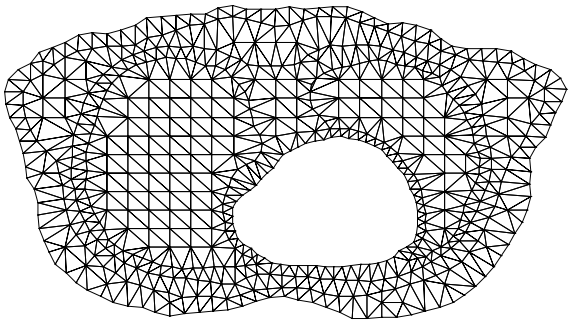

Fig. 16. A triangulated MRI prior to mesh refinement that forms the basis for a finite element approximation for forward and inverse ECG solutions. From Schimdt et al.

but is defined to be constant over each element and discontinuous across element boundaries. Zienkiewicz showed that globally current densities provide more accurate estimates of the energy than constant current densities [244], [245]. We can use the difference between the two estimates to determine which elements need refinement. The smoothed current densities are now defined at the same nodal locations as the potentials and are continuous across element boundaries. The smoothing process uses the Galerkin technique [243] and involves minimizing the difference between the two current densities,

$$\sigma(\nabla\tilde{\Phi} - \nabla\hat{\Phi}), \tag{129}$$

where $\nabla\tilde{\Phi}$ is the smoothed gradient representing the exact gradient and $\nabla\hat{\Phi}$ is the constant gradient from the finite element solution. This method produces a system of equations in the following form:

$$(\int_{\Omega_e} \Psi_i \Psi_j d\Omega_e)(\sigma\nabla\tilde{\Phi}_i) = (\int_{\Omega_e} \sigma\nabla\hat{\Phi}_i \Psi_i d\Omega_e), \tag{130}$$

where Ψ_i is a linear basis function defined at node i, $\nabla\tilde{\Phi}_i$ is the value of the smoothed gradient at that point, σ is the conductivity, and $\nabla\hat{\Phi}_i$ is the constant gradient resulting from the finite element solution. In equation (130), the current density, $\sigma\nabla\tilde{\Phi}_i$ is the unknown that is solved for at each node, i of the grid. Thus, we calculate a new estimate, $\tilde{q}$, which can be used in place of the exact gradient, q. The current density now varies linearly in each tetrahedron and is continuous everywhere.

Once the energy error has been computed, the mesh refinement can begin. The error must be related to some parameter, which guides the mesh refinement. We utilized a spacing function, h_e, which controls the size and number of elements [243]. This spacing function h_e is defined to be the linear interpolation of the spacing values at the four nodes of the tetrahedron. A variable β is defined as the ratio of the element error to the average of the element errors:

$$\beta_e = \frac{\|e_q\|_e}{\|\hat{e}_q\|_e}. \tag{131}$$

If $\beta_e > 1$, then a new spacing function can be defined as

$$h^* = \frac{h}{\beta_e}. \tag{132}$$

Thus, the spacing values at each node of the element are reduced. This new spacing function will then increase the number of points in the regions of large errors. The whole process is repeated until the global error estimate falls below the specified level.

The algorithm does not depend on how the modification was made. An error estimate other than the one just described can be used by simply modifying the spacing values in some fashion. This allows any number of error estimators to be used to guide the adaptive mesh refinement process. Results of applying the above algorithm are show in Figures 16, 17, and 18 from Schmidt et al. [82].

To find out more about adaptive refinement methods, see [238], [191], [34], [246], [239], [82].

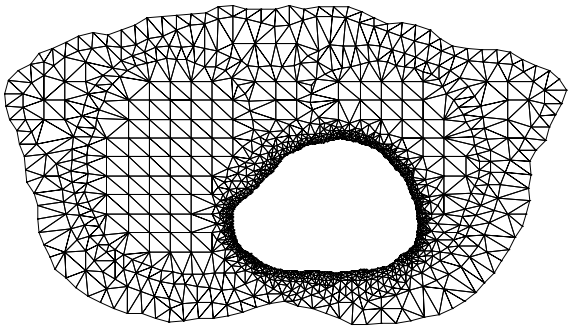

Fig. 17. A triangulated MRI after two iterations of the mesh refinement algorithm. From Schimdt et al.

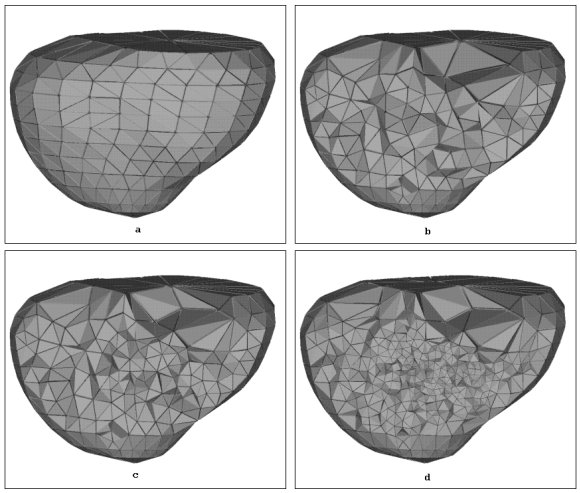

Fig. 18. A model of the isolated dog heart showing the initial surface mesh (a), and a cut-away view just below one of the electrodes (b)–(d) corresponding to the first, third and fifth iteration of the refinement process. From Schmidt et al.

VII. Software

Due to the availability of software such as Netscape, Explorer, and Mosaic, many resources are now freely available via the internet. As several new links are added each day, any list of URLs will be outdated before publication. However, there are a substantial number of apparently stable, frequently updated pages that provide resources useful to biomedical modelers. I have listed several such web pages below.

- Netlib Repository at UTK and ORNL. http://netlib2.cs.utk.edu/. Netlib was one of the first on-line repositories of numerical software and is still one of the best. Here you will find Linpack, Lapack, Voronoi, and a number of other useful numerical software packages and codes.
- Computational Science Education Program (CSEP). http://csep1.phy.ornl.gov/csep.html. CSEP is an excellent on-line resource for computational science and computational engineering. The intended audience consists of students in science and engineering at the advanced undergraduate level and higher. The *e-book* includes a chapter by the author entitled *Direct and Inverse Bioelectric Field Problems* that contains exercises, projects, and C and Fortran computer codes.
- Diffpack. http://www.oslo.sintef.no/diffpack. Diffpack consists of a collection of object-oriented libraries for solving partial differential equations, and several Unix utilities for general software management and numerical programming. In particular, this piece of software is aimed at rapid prototyping of simulators based on PDEs, still offering a high level of efficiency.

One bioelectric field application within Diffpack is HEART.
http://www.oslo.sintef.no/diffpack/projects/dkz/HEART.html. HEART is a collection of C++ classes that solves an equation system consisting of a reaction-diffusion parabolic differential equation and an elliptic equation governing the potential distribution in the cardiac muscle and surrounding tissues.

- The NIST guide to mathematical software on the internet. http://gams.nist.gov/. This is a gateway to the NIST

guide to available mathematical software, a cross-index and virtual repository of mathematical and statistical software components of use in computational science and engineering.

• The Problem Solving Environments Home Page.
http://www.cs.purdue.edu/research/cse/pses/research.html. A problem solving environment (PSE) is a computer system that provides all the computational facilities necessary to solve a target class of problems. These features include advanced solution methods, automatic and semiautomatic selection of solution methods, and ways to easily incorporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware or software. This site lists a number of PSEs as well as links to other computational tools.

• PETSc. http://www.mcs.anl.gov/petsc/petsc.html. PETSc, the Portable, Extensible Toolkit for Scientific Computation is a suite of data structures and routines for the uni- and parallel-processor solution of large-scale scientific application problems modeled by partial differential equations.

• Aztec. http://www.cs.sandia.gov/HPCCIT/aztec.html.
Aztec is a parallel iterative library for solving linear systems, $Ax = b$, which is both easy-to-use and efficient. Simplicity is attained using the notion of a global distributed matrix. The global distributed matrix allows a user to specify pieces (different rows for different processors) of his application matrix exactly as he would in the serial setting (i.e. using a global numbering scheme). Issues such as local numbering, ghost variables, and messages are ignored by the user and are instead computed by an automated transformation function.

• QMG: Mesh Generation Package.
http://www.cs.cornell.edu/Info/People/vavasis/qmg-home.html. The QMG package does finite element mesh generation in two and three dimensions. The package includes geometric modeling software, the mesh generator itself, and a finite element solver. The software is written primarily in C++ and Matlab.

• UG - A Flexible Toolbox for the Adaptive Multigrid Solution of Partial Differential Equations. http://dom.ica3.uni-stuttgart.de/ ∼ug/. UG supports unstructured mesh generation with local refinement and coarsening. Available mesh elements include triangles and quadrilaterals in 2D and tetrahedrons (and hexahedrons soon) in 3D. One can use an arbitrary number of degrees of freedom in nodes, edges, sides, elements with sparse block matrix structure. There is an interactive graphical user interface for Unix workstations using X11 and for Macintosh computers.

• UNM Geometry Center. http://www.geom.umn.edu/software/cglist/. Directory of computational geometry software at the University of Minnesota. This site has a number of excellent geometrical software packages including Voronoi diagram and Delaunay triangulation/tetrahedralization packages, numerical and algebraic computation tools, as well as visualization software.

• David A. Bader's List of Parallel Computing Sites.
http://www.umiacs.umd.edu/ ∼dbader/sites.html. A wealth of information on high-performance computing with hundreds of links to other sites.

• Roadmap of HPCC Applications, Technology, and Markets. http://www.npac.syr.edu/roadmap/. A roadmap of HPCC Applications, Technology, and Markets at the Northeast Parallel Architectures Center (NPAC), Syracuse University.

• BMEnet. http://fairway.ecn.purdue.edu/bme/. The BMEnet is a biomedical engineering resource maintained at Purdue University under a grant from The Whitaker Foundation.

• NAS Scientific Visualization Sites. http://www.nas.nasa.gov/NAS/Visualization/visWeblets.html. This site contains an annotated bibliography of many scientific visualization web sites.

VIII. Summary

I have provided an overview of computational and numerical techniques for bioelectric field problems. Modeling and simulation of bioelectric fields involve applying diverse multidisciplinary skills, from computational geometry and image processing to construct the models, numerical analysis and large-scale computing for approximating the physics and physiology of the models, and scientific visualization to interpret results obtained from simulations.

While great progress has been made in the modeling and simulation of bioelectric fields over the last several years, there is still much to be done before the modeling and simulation approach will be comfortably absorbed into the clinic and device design industry. We are just now approaching the level of geometric, physical, and physiological complexity that may yield solutions to scenerios physiologically realistic enough to provide practical results. Even now, though, each of individual computational modeling and simulation techniques has advanced dramatically, the entire process of

modeling and simulation of bioelectric field problems is anything but streamlined.

The typical process of performing computational modeling and simulation can be described by the following algorithm:

1. Create and/or modify a discretized geometric model.
2. Create and/or modify initial conditions and/or boundary conditions.
3. Compute numerical approximations to the governing equation(s), storing results on disk.
4. Visualize and/or analyze results using a separate visualization package.
5. Make appropriate changes to the model.
6. Go back to step 1, 2 and/or 3.
7. Repeat.

The "art" of obtaining valuable results from a model has up until now required a scientist to execute this lengthy process time and time again. Changes to the model, input parameters, or computational processes are typically made using rudimentary tools, the most common being text editors. While the experienced scientist will instill some degree of automation, usually via scripts, into the process, it is still time consuming and inefficient.

Ideally, scientists and engineers would be provided with a system in which all these heterogeneous but related computational components were linked, so that all aspects of the modeling and simulation process could be controlled graphically within the context of a single application program. While this would be the preferred modus operandi for most computational scientists, it is not the current standard of scientific computing, because the creation of such a program is an enormously difficult task.

The many difficulties in creating such a program arise from the need to integrate a wide range of disparate computing disciplines - such as user interface technology, 3D graphics, parallel computing, programming languages, and numerical analysis - with a wide range of equally disparate application - specific requirements.

A number of researchers are trying to overcome these difficulties by developing Problem Solving Environments (PSEs). As noted by Gallopoulos et al. in [247], "A PSE is a computer system that provides all the computational facilities necessary to solve a target class of problems. These features include advanced solution methods, automatic and semiautomatic selection of solution methods, and ways to easily incoporate novel solution methods. Moreover, PSEs use the language of the target class of problems, so users can run them without specialized knowledge of the underlying computer hardware or software. By exploiting modern technologies such as interactive color graphics, powerful processors, and networks of specialized services, PSEs can track extended problem solving tasks and allow users to review them easily. Overall, they create a framework that is all things to all people: they solve simple or complex problems, support rapid prototyping or detailed analysis, and can be used in introductory education or at the frontiers of science." PSEs that have been developed for solving computational field problems include PDELab, PDEase, PARADIFE, among others (see the PSE home page mentioned in the the Software section for more information).

The first PSE to be developed for use with bioelectric field problems is SCIRun [248], [249], [250]. SCIRun is a scientific programming environment that allows the interactive construction, debugging and steering of large-scale scientific computations. Using this "computational workbench," a scientist can design and modify simulations interactively via a dataflow programming model. SCIRun enables scientists to design and modify models and automatically change parameters and boundary conditions as well as the mesh discretization level needed for an accurate numerical solution. As opposed to the typical "off-line" simulation mode - in which the scientist manually sets input parameters, computes results, visualizes the results via a separate visualization package, then starts again at the beginning - SCIRun "closes the loop" and allows interactive steering of the design and computation phases of the simulation. To make the dataflow programming paradigm applicable to large scientific problems, the researchers have identified ways to avoid the excessive memory use inherent in standard dataflow implementations, and they have implemented fine-grained dataflow in order to further promote computational efficiency.

A specific application of SCIRun is to design internal defibrillator devices and measure their effectiveness in an interactive graphical environment. Using SCIRun, engineers are able to design internal defibrillation devices, place them directly into the computer model, and automatically change parameters (size, shape and number of electrodes) and source terms (position and magnitude of voltage sources) as well as the mesh discretization level needed for an accurate finite element solution. Furthermore, engineers can use the interactive visualization capabilities to visually gauge the effectiveness of their design in terms of distribution of electrical current flow and density maps of current distribution [250]. A SCIRun network that can be used to model cardiac defibrillation is shown in Figure 19.

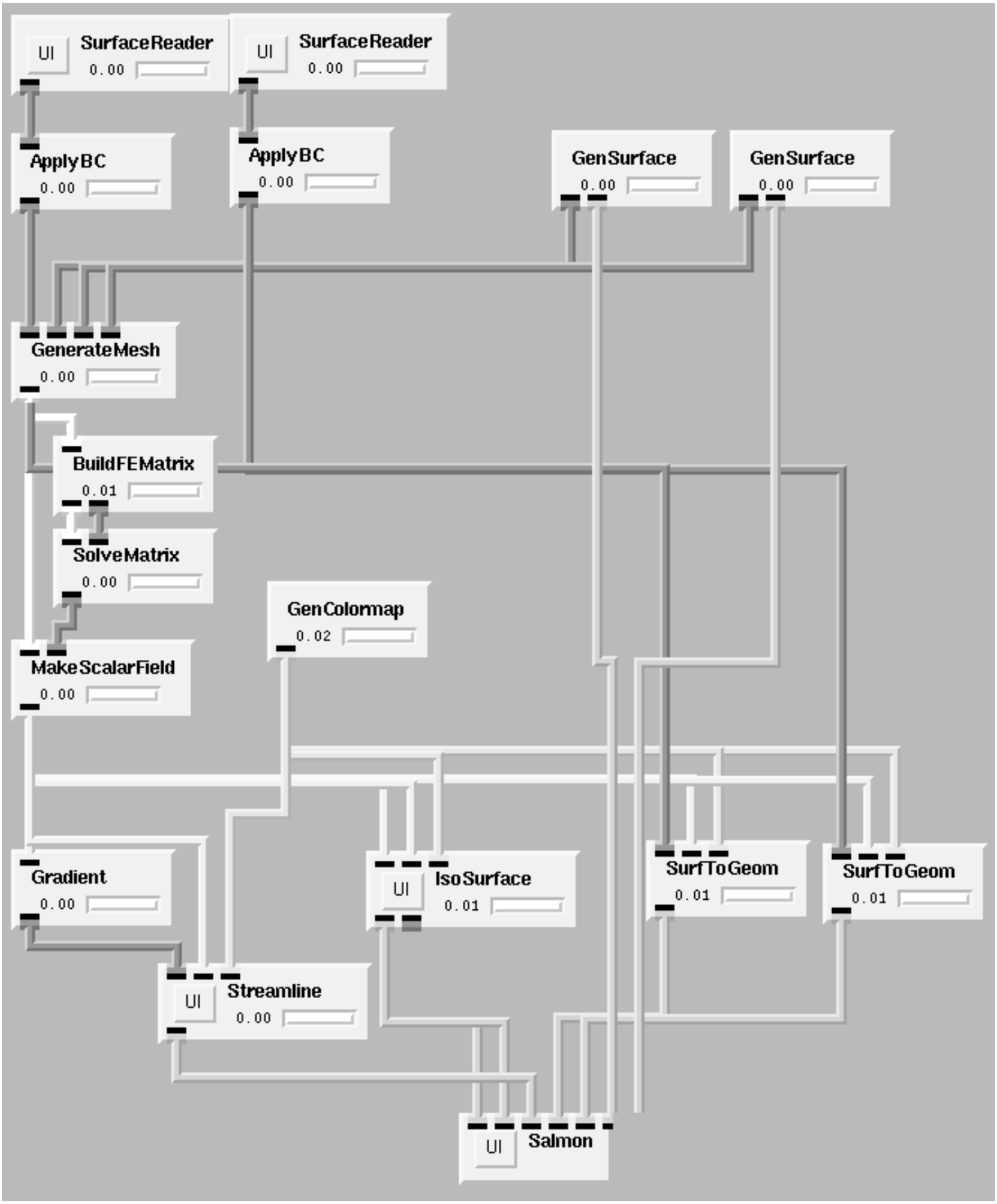

Fig. 19. SCIRun network for the cardiac defibrillation problem

While PSEs will certainly streamline the modeling, simulation, and visualization pipeline, there are still many issues, mostly relating to complexity, for the computational scientist to tackle regarding bioelectric field problems. As the complexity of physiological and geometric models continues to increase, more attention will need to be paid to creating efficient algorithms that take advantage of parallelism. Simultaneously, more emphasis may be placed on adaptive methods (for both temporal and spatial apsects) that will help researchers to produce accurate solutions in a computationally efficient way. In addition, bioelectric field applications will increasingly require powerful parallel computers, such as the SGI Power Challenge, the Intel Paragon, and the Cray T3D/E. This increased computing power provides the ability to perform complex three-dimensional simulations, a significant benefit. However, such simulations present new challenges for computational scientists. How does one effectively analyze and visualize complex 3D data? How does one solve the problems of working with very large datasets often consisting of tens to hundreds of gigabytes? It easy for a researcher to generate hundreds or even thousands of images, datafiles, and results. Organizing that data is a significant problem that will require much work. Scientific database research may have much to offer in this area. How does one provide tools that address these computational problems while serving the needs of scientific users? Clearly, the computational bioengineer will need to couple forces with other computational, mathematical, and computer scientists and engineers to successfully tackle many of these issues.

IX. Acknowledgements

This work was supported in part by awards from the Whitaker Foundation, the NIH, and the NSF. The author would like to thank Prasad Gharpure (segmentation), Robert MacLeod (triangulation), John Schmidt (mesh generation), and Kaj Henneberg (boundary element method) for their significant contributions to the Model Construction and Approximation Methods sections. The author would also like to thank Katharine Coles for her helpful comments and suggestions.

Professor Johnson directs the Scientific Computing and Imaging Institute at the University of Utah where he is a Professor of Computer Science and holds faculty appointments in the Departments of Physics, and Bioengineering. His research interests are in the area of scientific computing. Particular interests include inverse and imaging problems, adaptive methods, problem solving environments, large scale computational problems in medicine, and scientific visualization. Professor Johnson was awarded a Young Investigator's (FIRST) Award from the NIH in 1992, the NSF National Young Investigator (NYI) Award in 1994, and the NSF Presidential Faculty Fellow (PFF) award from President Clinton in 1995. In 1996 he received a DOE Computational Science Award and in 1997 received the Par Excellence Award from the University of Utah Alumni Association and the Presidential Teaching Scholar Award. In 1999, Professor Johnson was Awarded the Governor's Medal for Science and Technology.

REFERENCES

[1] Plonsey, R. and Fleming, D., *Bioelectric Phenomena*, McGraw-Hill Book Company, New York, 1969.

[2] Miller, C.E. and Henriquez, C.S., Finite element analysis of bioelectric phenomena, *Crit. Rev. in Biomed. Eng.*, 1990;18:181–205.

[3] Nenonen, J., Rajala, H.M., and Katilia, T., *Biomagnetic Localization and 3D Modelling*, Helsinki University of Technology, Espoo, Finland, 1992, Report TKK-F-A689.

[4] Johnson, C.R., MacLeod, R.S., and Ershler, P.R., A computer model for the study of electrical current flow in the human thorax, *Computers in Biology and Medicine*, 1992;22:305–323.

[5] Johnson, C.R., MacLeod, R.S., and Matheson, M.A., Computer simulations reveal complexity of electrical activity in the human thorax, *Comp. in Physics*, 1992;6:230–237.

[6] Kim, Y., Fahy, J.B., and Tupper, B.J., Optimal electrode designs for electrosurgery, *IEEE Trans. Biomed. Eng.*, 1986;33:845–853.

[7] Plonsey, R. and Barr, R. C., *Bioelectricity*, Plenum Press, New York, 1988.

[8] Barr, R.C., Basic electrophysiology, in *The Biomedical Engineering Handbook*, Bronzino, J.D., Ed., pp. 101–118. CRC Press, Boca Raton, 1995.

[9] Plonsey, R., Volume conductor theory, in *The Biomedical Engineering Handbook*, pp. 119–125. CRC Press, Boca Raton, 1995.

[10] Geselowitz, D.B. and Schmitt, O.H., Electrocardiography, in *Biological Engineering*, Schwan, H.P., Ed., pp. 333–390. McGraw Hill, New York, 1969.

[11] Geselowitz, D.B., Dipole theory in electrocardiography, *Am. J. Cardiol.*, 1964;14:301–306.

[12] Henriquez, C.S., Simulating the electrical behavior of cardiac tissue using the bidomain model, *CRC Crit. Revs. in Biomed. Eng.*, 1993;21:1–77.

[13] Pollard, A.E., Hooke, N.F., and Henriquez, C.S., Cardiac propagation simulation, *Crit. Rev. in Biomed. Eng.*, 1992;20:319–358.

[14] Plonsey, R., Consideration of quasi-stationarity in electrophysiological systems, *Bull. Math. Biophysics*, 1967;29:657.

[15] Geselowitz, D.B., Use of time integrals of the ECG to solve the inverse problem, *IEEE Trans Biomed. Eng.*, 1985;32:73–75.

[16] Oostendorp, T.F. and Oosterom, A.van, Source parameter estimation in inhomogeneous volume conductors of arbitrary shape, *IEEE Trans Biomed. Eng.*, 1989;36:382–391.

[17] Greensite, F., Huiskamp, G., and Oosterom, A.van, New quantitative and qualitative approaches to the inverse problem of electrocardiology: their theoretical relationship and experimental consistency, *Medical Physics*, 1990;17:369–379.

[18] Mirowski, M., The automatic implantable cardioverter-defibrillation: An overview, *Journal of the American College of Cardiology*, 1985;6:461–466.

[19] Karlon, W., Eisenberg, S., and Lehr, J., Defibrillation current density distributions: a three-dimensional finite element model of the canine thorax, in *Proceedings of the 13th Annual Conference of the IEEE Engineering in Medicine and Biology Society*, 1991, vol. 13, pp. 770–771.

[20] Karlon, W., Defibrillation current density distributions: A three-dimensional finite element model of the canine thorax, M.s. thesis, Boston University, Boston, MA, 1991.

[21] Claydon, F., Pilkington, T., Tang, A., Morrow, M., and Ideker, R., A volume conductor model of the thorax for the study of defibrillation fields, *IEEE Transactions on Biomedical Engineering*, 1988;35.

[22] Deale, O. and Lerman, B., Intrathoracic current flow during transthoracic defibrillation in dogs, *Circulation Research*, 1990;67:1405–1419.

[23] Lerman, B. and Deale, O., Relation between transcardiac and transthoracic current during defibrillation in humans, *Circulation Research*, 1990;67:1420–1426.

[24] Schmidt, J.A. and Johnson, C.R., Defibsim: An interactive defibrillation device design tool, in *IEEE Engineering in Medicine and Biology Society 17th Annual International Conference*. 1995, IEEE Press.

[25] Mirowski, M., M., Mower, Reid, P., Watkins, L., and Langer, A., The automatic implantable defibrillator, *Pace*, 1982;5:384–399.

[26] Langer, A., Heilman, M., Mower, M., and Mirowski, M., considerations in the development of the automatic implantable defibrillator, *Medical Instrumentation*, 1976;10:163–167.

[27] Schmidt, J.A., Johnson, C.R., and MacLeod, R.S., An interactive computer model for defibrillation device design, in *International Congress on Electrocardiology*. 1995, pp. 160–161, ICE.

[28] Blilie, D., Fahy, J., Chan, C., Ahmed, M., and Kim, Y., Efficient solution of three-dimensional finite element models for defibrillation and pacing applications, in *Proceedings of the 13th Annual Conference of the IEEE Engineering in Medicine and Biology Society*, 1991, vol. 13, pp. 772–773.

[29] Hutchinson, S.A., Gao, S., Ai, L., Ng, K.T., Deale, O.C., Cahill, P.T., and Lerman, B.B., Three-dimensional modeling of electrical defibrillation on a massively parallel computer, in *Computers in Cardiology*. 1992, pp. 343–346, IEEE Computer Society Press.

[30] Hutchinson, S.A. and Ng, K.T., An optimization algorithm for electrode configurations in transcardiac defibrillation, in *Proceedings of the 15th Annual International Conference of the IEEE-EMBS*, 1993, vol. 15.

[31] Min, X., Wang, L., Hill, M., Lee, J., and Mehra, R., Detailed human thorax FEM model for the cardiac defibrillation study, in *Proceedings of the 15th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Piscataway, New Jersey, 1993, pp. 838–839, IEEE Press.

[32] Min, X. and Mehra, R., Finite element analysis of patch electrodes for cardiac defibrillation in a simulated tank, in *Proceedings of the 15th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Piscataway, New Jersey, 1993, pp. 826–827, IEEE Press.

[33] Yamashita, Y., Theoretical studies on the inverse problem in electrocardiography and the uniqueness of the solution, *IEEE Trans Biomed. Eng.*, 1982;29:719–725.

[34] Johnson, C.R. and MacLeod, R.S., Nonuniform spatial mesh adaptation using a posteriori error estimates: applications to forward and inverse problems, *Applied Numerical Mathematics*, 1994;14:311–326.

[35] Hansen, P.C., Analysis of discrete ill-posed problems by means of the L-curve, *SIAM Review*, 1992;34:561–580.

[36] Golub, G.H. and Loan, C.F. Van, *Matrix Computations*, Johns Hopkins, Baltimore, 1989.

[37] Horn, R.A. and Johnson, C.R., *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1994.

[38] Groetsch, C.W., *Inverse Problems in the Mathematical Sciences*, Vieweg, Wiesbaden, Germany, 1993.

[39] Tikhonov, A. and Arsenin, V., *Solution of Ill-posed Problems*, Winston, Washington, DC, 1977.

[40] Tikhonov, A.N. and Goncharsky, A.V., *Ill-Posed Problems in the Natural Sciences*, MIR Publishers, Moscow, 1987.

[41] Engl, H.W. and Groetsch, C.W., *Inverse and Ill-Posed Problems*, Academic Press, Boston, 1987.

[42] Glasko, V.B., *Inverse Problems of Mathematical Physics*, American Institute of Physics, New York, 1984.

[43] Brooks, D.H. and MacLeod, R.S., Imaging the electrical activity of the heart: direct and inverse approaches, in *Proc. ICIP '94*. 1994, pp. 548–552, ICIP.

[44] Brooks, D.H., Ahmad, G., and MacLeod, R.S., Multiply constrained inverse electrocardiography: Combining temporal, multiple spatial, and iterative regularization, in *Proc. 16th Ann. Int. Conf. IEEE EMBS*. 1994, pp. 137–138, IEEE Press.

[45] Rudy, Y. and Messinger-Rapport, B.J., The inverse solution in electrocardiography: Solutions in terms of epicardial potentials, *CRC Crit. Rev. Biomed. Eng.*, 1988;16:215–268.

[46] Greensite, F., Huiskamp, G., and Oosterom, A.van, The mathematical basis for imaging cardiac electrical function, *Crit. Rev. in Biomed. Eng.*, 1995;.

[47] Hansen, P.C., Regularization tools: A matlab package for analysis and solution of discrete ill-posed problems, Tech. Rep., Technical University of Denmark, 1992, Available via netlib in the library numeralgo/no4.

[48] Geselowitz, D.B. and Miller, W.T., A bidomain model for anisotropic cardiac muscle, *Annal. Biomed. Eng.*, 1983;1:191–206.

[49] Huiskamp, G.J. and Oosterom, A.van, The depolarization sequence of the human heart surface computed from measured body surface potentials, *IEEE Trans Biomed. Eng.*, 1989;35:1047–1059.

[50] Huiskamp, G.J. and Oosterom, A.van, Tailored versus standard geometry in the inverse problem of electrocardiography, *IEEE Trans Biomed. Eng.*, 1989;36:827–835.

[51] Oosterom, A.van and Huiskamp, G.J., The effect of torso inhomogeneities on body surface potentials quantified using "tailored" geometry, *J. Electrocardiol.*, 1989;22:53–72.

[52] Oostendorp, T.F. and Oosterom, A.van, The potential distribution generated by surface electrodes in inhomogeneous volume conductors of arbitrary shape, *IEEE Trans Biomed. Eng.*, 1991;38:409–417.

[53] Oster, H.S. and Rudy, Y., The use of temporal information in the regularization of the inverse problem of electrocardiography., *IEEE Trans Biomed. Eng.*, 1992;39:65–75.

[54] Rudy, Y. and Oster, H.S., The electrocardiographic inverse solution, in *High-Performance Computing in Biomedical Research*, Pilkington, T.C., B., Loftis, Thompson, J.F., Woo, S.L., Palmer, T. C, and Budinger, T.F., Eds., chapter 6, pp. 135–155. CRC Press, 1993.

[55] Rudy, Y. and Oster, H., The electrocardiographic inverse problem, *Crit. Rev. Biomed. Eng.*, 1992;20:22–45.

[56] Gulrajani, R.M., Roberge, F.A., and Mailloux, G.E., The forward problem of electrocardiography, in *Comprehensive Electrocardiology*, Macfarlane, P.W. and Lawrie, T.D. Veitch, Eds., pp. 197–236. Pergamon Press, Oxford, England, 1989.

[57] Scher, A.M. and Corbin, L.V., Present state of the electrocardiographic forward problem, *Adv. Cardiol.*, 1978;21:2–7.

[58] Shahidi, A.V., Savard, P., and Nadeau, R., Forward and inverse problems of electrocardiography: Modeling and recovery of epicardial potentials in humans, *IEEE Trans Biomed. Eng.*, 1994;41:249–256.

[59] Swihart, J.G., Numerical methods for solving the forward problem in electrocardiology, in *The Theoretical Basis of Electrocardiology*, Nelson, C.V. and Geselowitz, D.B., Eds., pp. 257–293. Claredon Press, Oxford, 1976.

[60] Walker, S.J. and Kilpatrick, D., Forward and inverse electrocardiographic calculations using resistor network models of the human torso, *Circ. Res.*, 1987;61:504–513.

[61] Ahmad, G.F., Brooks, D.H., Jacobson, C., and MacLeod, R.S., A feasability study of inverse electrocardiography by convex optimization, in *Proceedings of the 21st N.E. Bioengineering Conference*, Bar Harbour, ME, 1995, pp. 245–246.

[62] Barr, R.C., *Constrained Inverse Electrocardiography*, PhD thesis, Duke University, Durham, NC, 1968.

[63] Barr, R.C. and Spach, M.S., Inverse solutions directly in terms of potentials, in *The Theoretical Basis of Electrocardiography*, Nelson, C.V. and Geselowitz, D.B., Eds., pp. 294–304. Clarendon Press, Oxford, 1976.

[64] Franzone, P. Colli, Taccardi, B., and Viganotti, C., An approach to inverse calculation of epicardial potentials from body surface maps, *Adv. Cardiol.*, 1978;21:50–54.

[65] Franzone, P. Colli, Gassaniga, G., Guerri, L., Taccardi, B., and Viganotti, C., Accuracy evaluation in direct and inverse electrocardiology, in *Progress in Electrocardiography*, Macfarlane, P.W., Ed. 1979, pp. 83–87, Pitman Medical.

[66] Franzone, P. Colli, Guerri, L., Taccardi, B., and Viganotti, C., Finite element approximation of regularized solution of the inverse potential problem of electrocardiography and application to experimental data, *Calcolo*, 1985;22:91.

[67] Gulrajani, R.M., Savard, P., and Roberge, F.A., The inverse problem in electrocardiography: Solutions in terms of equivalent sources, *Crit. Rev. Biomed. Eng.*, 1988;16:171–214.

[68] Gulrajani, R.M., Roberge, F.A., and Savard, P., The inverse problem of electrocardiography, in *Comprehensive Electrocardiology*, Macfarlane, P.W. and Lawrie, T.D. Veitch, Eds., pp. 237–288. Pergamon Press, Oxford, England, 1989.

[69] Johnson, C.R., *The Generalized Inverse Problem in Electrocardiography: Theoretical, Computational and Experimental Results*, PhD thesis, University of Utah, Salt Lake City, Utah, 1989.

[70] Johnson, C.R. and Pollard, A.E., Electrical activation of the heart: Computational studies of the forward and inverse problems in electrocardiography, in *Computer Assisted Modeling on the IBM 3090*, Billingsley, K.R., III, H.V. Brown, and Durohanes, E.D., Eds., pp. 583–628. Baldwin Press, University of Georgia, Athens, Georgia, 1992.

[71] Johnson, C.R. and MacLeod, R.S., Inverse solutions for electric and potential field imaging, in *Physiological Imaging, Spectroscopy, and Early Diagnostic Methods*, Barbour, R.L. and Carvlin, M.J., Eds., pp. 130–139. SPIE, 1993.

[72] Johnson, C.R. and MacLeod, R.S., Nonuniform spatial mesh adaption using a posteriori error estimates: applications to forward and inverse problems, *Appl. Num. Anal.*, 1994;14:311–326.

[73] Johnson, C.R. and MacLeod, R.S., Local regularization and adaptive methods for the inverse Laplace problem, in *Biomedical and Life Physics*, Ghista, D.N., Ed., pp. 224–234. Vieweg-Verlag, Braunschweig, 1996.

[74] MacLeod, R.S., Johnson, C.R., Gardner, M.J., and Horáček, B.M., Localization of ischemia during coronary angioplasty using body surface potential mapping and an electrocardiographic inverse solution, in *IEEE Computers in Cardiology*. 1992, pp. 251–254, IEEE Press.

[75] Messinger-Rapport, B.J. and Rudy, Y., Regularization of the inverse problem in electrocardiography: A model study, *Math. Biosci.*, 1988;89:79–118.

[76] Yamashita, Y., Theoretical studies on the inverse problem in electrocardiography and the uniqueness of the solution, *IEEE Trans Biomed. Eng.*, 1982;29:719–725.

[77] Johnson, C.R., MacLeod, R.S., and Matheson, M.A., Computational medicine: Bioelectric field problems, *IEEE COMPUTER*, 1993;:59–67.

[78] Huiskamp, G.J., *Noninvasive Determination of Human Ventricular Activation*, PhD thesis, Catholic University of Nijmegen, Nijmegen, The Netherlands, 1989.

[79] Nielsen, P., Grice, I. Le, Smaill, B., and Hunter, P., Mathematical model of geometry and fibrous structure of the heart, *American Journal of Physiology: Heart Circulation and Physiology*, 1991;260:H1365–H1378.

[80] Hunter, P., Nielsen, P., Smaill, B. LeGrice, I., and Hunter, I., An anatomical heart model with applications to myocardial activation and ventricular mechanics, in *High-Perfomance Computing in Biomedical Research*, Pilkington, T., Loftis, B., Thompson, J., Woo, S., Palmer, T., and Budinger, T., Eds., chapter 1, pp. 3–26. CRC Press, Inc., Boca Raton, FL, 1993.

[81] Macfarlane, P.W. and Lawrie, T.D.V., Eds., *Comprehensive Electrocardiology: Theory and Practice in Health and Disease*, Pergamon Press, New York, 1988.

[82] Schmidt, J.A., Johnson, C.R., Eason, J.C., and MacLeod, R.S., Applications of automatic mesh generation and adaptive methods in computational medicine, in *Modeling, Mesh Generation, and Adaptive Methods for Partial Differential Equations*, Babuska, I., Flaherty, J.E., Henshaw, W.D., Hopcroft, J.E., Oliger, J.E., and Tezduyar, T., Eds., pp. 367–390. Springer-Verlag, 1995.

[83] Schmidt, J. A., Johnson, C. R., and Macleod, R. S., An interactive computer model for defibrillation, in *Proceedings of the 22nd International Congress on Electrocardiology*, Oosterom, A.van, Oostendorp, T. F., and Uijen, G. J. H., Eds. June 1995, pp. 160–161, University Press Nijmegen.

[84] Schmidt, J. A. and Johnson, C. R., Defibsim: An interactive defibrillation device design tool, in *IEEE Engineering in Medicine and Biology Society 17th Annual International Conference*, 1995, pp. 1285–1286.

[85] Jorgenson, D. Blilie, Haynor, D. R., Bardy, G. H., and Kim, Y., Computational studies of transthoracic and transvenous defibrillation in a detailed 3–D human thorax model, *IEEE Transactions on Biomedical Engineering*, 1995;42:172–184.

[86] Jorgenson, D. Blilie, Schimpf, P. H., Shen, I., Johson, G., Bardy, G. H., Haynor, D. R., and Kim, Y., Predicting cardiothoracic voltages during high energy shocks: Methodolgy and comparison of experimental to finite element model data, *IEEE Transactions on Biomedical Engineering*, 1995;42:559–571.

[87] Panescu, D., Webster, J. G., Tompkins, W. J., and Stratbucker, R. A., Optimization of cardiac defibrillation by three–dimensional finite element modeling of the human thorax, *IEEE Transactions on Biomedical Engineering*, 1995;42:185–192.

[88] Karlon, W. J., Eisenberg, S. R., and Lehr, J. L., Effects of paddle placement and size on defibrillation current distribution: A three–dimensional finite element model, *IEEE Transactions on Biomedical Engineering*, 1993;40:246–255.

[89] Camacho, M. A., Lehr, J. L., and Eisenberg, S. R., A three–dimensional finite element model of human transthoracic defibrillation: Paddle placement and size, *IEEE Transactions on Biomedical Engineering*, 1995;42:572–578.

[90] Mohammed, O. A. and Uler, F. G., Detailed 2–D and 3–D finite element modeling of the human body for the evaluation of derfibrillation fields, *IEEE Transactions on Magnetics*, 1993;29:1403–1406.

[91] Hutchinson, S.A., *Parallel Finite-Element Analysis and Optimization of Electrical Defibrillation*, PhD thesis, New Mexico State University, Dec. 1993.

[92] Hutchinson, S.A., Ng, K.T., Shadid, J.N., and Nadeem, A., Electrical defibrillation optimization – an automated, iterative parallel finite-element approach, *IEEE Trans. Biomed. Eng.*, 1995;.

[93] Newell, J.C., Gisser, D.G., and Isaacson, D., An electric current tomograph, *IEEE Trans Biomed. Eng.*, 1988;35:828–823.

[94] Gisser, D.G., Isaacson, D., and Newll, J.C., Electric current computed tomography and eigenvalues, *SIAM J. Appl. Math.*, 1990;50:1623–1634.

[95] Barber, D.C. and Brown, B.H., Applied potential tomography, *J. Phys. E. Sci. Instrum.*, 1984;17:723–733.

[96] Gisser, D.G., Isaacson, D., and Newell, J.C., Theory and performance of an adaptive current tomography system, *Clin. Phys. Physiol. Meas.*, 1988;9 (Supplement A):35–41.

[97] Kim, D.W., Baker, L.E., Pearce, J.A., and Kim, W.K, Origins of impedance change in impedance cardiography by a three-dimensional finite element model, *IEEE Trans. Biomed. Eng.*, 1988;35:993.

[98] J., Holsheimer, J.J., Struijk, and N.R., Tas, Effects of electrode geometry and combination on nerve fibre selectivity in spinal cord stimulation, *Med. Biol. Eng. Comp.*, 1995;33:676–682.

[99] Rattay, F., *Electrical Nerve Stilmulation, Theory, Experiments and Applications*, Springer-Verlag, New York, 1990.

[100] Koch, C. and Segev, I., *Methods of Neural Modeling*, MIT Press, Cambridge, Mass., 1989.

[101] Coburn, B., Electrical stimulation of the spinal cord: two dimensional finite element analysis with particular reference to epidural electrodes, *Med. Biol. Eng. Comp.*, 1980;18:573.

[102] Coburn, B. and Sin, W.K., A theoretical study of epidural electrical stimulation of the spinal cord. I. finite element analysis of stimulus fields, *IEEE Trans. Biomed, Eng.*, 1985;32:971.

[103] Buchthal, F., Electromyography, in *Handbook of Electroencephalography and Clinical Neurophysiology*, vol. 16. Elsevier Scientific, Amsterdam, 1973.

[104] Henneberg, K., Principles of electromyography, in *The Biomedical Engineering Handbook*, Bronzino, J.D., Ed., pp. 191–200. CRC Press, Boca Ratan, 1995.

[105] Henneberg, K. and Plonsey, R., Boundary element analysis of the directional sensitivity of the concentric EMG electrode, *IEEE Trans. on Biomedical Engineering*, 1993;40:621–631.

[106] Barry, D.T., Basic concepts of electricity and electronics in clinical electromyography, *Muscle Nerve*, 1991;14:937.

[107] Weinstein, D.M., Johnson, C.R., and Schmidt, J.A., Effects of adaptive refinement on the inverse EEG solution, in *Experimental and Numerical Methods for solving Ill-Posed Inverse Problems*, 2-11, , Ed., vol. 2570. SPIE, 1995.

[108] Nunez, P.L., *Electric Fields in the Brain: The Neurophysics of the EEG*, Oxford University Press, Oxford, England, 1981.

[109] Gevins, A., Le, J., Brickett, P., Reutter, B., and Desmond, J., Seeing through the skull: Advanced EEGs use MRIs to accurately measure cortical activity from the scalp, *Brain Topography*, 1991;4:125–131.

[110] Silva, F.H. Lopesda, A critical review of clinical applications of topographic mapping of brain potentials, *Journal of Clinical Neurophysiology*, 1990;7:535–551.

[111] Le, J. and Gevins, A., Method to reduce blur distortion from EEG's using a realistic head model, *IEEE Trans. Biomed. Eng.*, 1993;40:517–528.

[112] Peters, M.J. and Munck, J.C. De, The influence of model parameters on the inverse solution based on MEGs and EEGs, *Acta Otolaryngol*, 1991;491:61–69.

[113] Lutkenhoner, B., Menninghaus, E., Steinstrater, O., Wienbruch, C., Gissler, M., and Elbert, T., Neuromagnetic source analysis using magnetic resonance images for the construction of source and volume conductor model, *Brain Topography*, 1995;7:291–299.

[114] Tada, Y. and Nagashima, T., Modeling and simulation of brain lesions by the finite-element method, *Engineering in Medicine and Biology*, 1994;August/September:497–503.

[115] Homma, S., Musha, T., Nakajima, Y., Okamoto, Y., Blom, S., Flink, R., Hagbarth, K.E., and Mostrom, U., Location of electric current sources in the human brain estimated by the dipole tracing method of the scalp-skull-brain (SSB) head model, *Electroencephalography and clinical Neurophysiology*, 1994;91:374–382.

[116] Faux, S.F., McCarley, R.W., Nestor, P.G., Shenton, M.E., Pollak, S.D., Penhume, V., Mondrow, E., Marcy, B., Peterson, A., Horvath, T., and Davis, K.L., P300 topographic assymetries are present in unmedicated schizophrenics, *Electroencephalography and clinical Neurophysiology*, 1993;88:32–41.

[117] McCarley, R.W., Shenton, M.E., O'Donnell, B.F., Faux, S.F., Kikinis, R., Nestor, P.G., and Jolesz, F.A., Auditory P300 abnormalities and left posterior superior temporal gyrus volume reduction in schizophrenia, *Archives of General Psychiatry*, 1993;50:190–197.

[118] Bronzino, J.D., Principles of electroencephalography, in *The Biomedical Engineering Handbook*, Bronzino, J.D., Ed., pp. 201–212. CRC Press, Boca Ratan, 1995.

[119] Wells, W., Kikinis, R., Grimson, W., and Jolesz, F., Statistical intensity correlation and segmentation of magnetic resonance image data, in *Proceedings of the Third Conference on Visualization in Biomedical Computing*. SPIE, 1994.

[120] Crane, R., *A Simplified Approach to Image Processing: Classical and Modern Techniques in C*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1997.

[121] Gonzalez, R.C. and Woods, R.E., *Digital Image Processing*, Addison Wesley, Reading, Mass., 1992.

[122] Blake, A. and Zisserman, A., *Visual Reconstruction*, MIT Press, Cambridge, 1987.

[123] Sonka, Milan, Hlavac, Vaclav, and Boyle, Roger, *Image Processing, Analysis and Machine Vision*, Chapman and Hall, 1993.

[124] Tyan, S. G., Median filtering, deterministic properties, in *Two–Dimensional Digital Signal Processing*, Huang, T. S., Ed. Springer Verlag, Berlin, 1981.

[125] Shen, H.W. and Johnson, C.R., Semi-automatic image segmentation: A bimodel thresholding approach, Tech. Rep., UUCS-94-019, Dept. of Computer Science, Univ. of Utah, 1994.

[126] Kundu, A and Mitra, S. K., A new algorithm for image edge extraction using a statistical classifier approach, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1987;9:569–577.

[127] Rosenfeld, A, Hummel, R. A., and Zucker, S. W., Scene labeling by relaxation operations, *IEEE Trans. on System, Man and Cybernetics*, 1976;6:420–433.

[128] Zucker, S. W., *Relaxation Labeling, local ambiguity, and low level vision*, pp. 593–616, Academic Press, New York, 1976.

[129] Riseman, E. M. and Arbib, M. A., Computational techniques in the visual segmentation of static scenes, *Computer Graphics and Image Processing*, 1977;6:221–276.

[130] Hancock, E. R. and Kittler, J, Edge labeling using dictionary based relaxation, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1990;12:165–181.

[131] Hanson, A. R. and Riseman, E. M., Eds., *Computer Vision System*, Academic Press, New York, 1978.

[132] Prager, J. M., Extracting and labeling boundary segments in natural scenes, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1980;2:16–27.

[133] Nilsson, N. J., *Principles of artificial intelligence*, Springer Verlag, Berlin, 1982.

[134] Martelli, A., Edge detection using heuristic search methods, *Computer Graphics and Image Processing*, 1972;1:169–182.

[135] Tadikonda, S.K., Sonka, M., and Collins, S.M., Efficient coronary boarder detection using heuristic graph searching, in *Proceedings of the Annual International Conference of the IEEE EMBS*. 1992, vol. 14, pp. 1897–1899, IEEE.

[136] Hough, P. V. C, A method and means for recognising complex patterns, *U. S. Patent 3,069,654*, 1962;.

[137] J., Illingworth and Kittler, J., The adaptive hough transform, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1987;9:690–698.

[138] Haralick, R. M. and Shapiro, L. G., Image segmentation techniques, *Computer Vision, Graphics, and Image Processing*, 1985;29:100–132.

[139] Zamperoni, P., Analysis of some region growing operators for image segmentation, in *Advances in Image processing and Pattern Recognition*, Cappelini, V. and Marconi, R., Eds., pp. 204–208. North Holland, Amsterdam, 1986.

[140] Grimson, W. E. L. and Loranzo-Perez, P., Localizing overlapping parts by searching the interpretation tree, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1987;9:469–482.

[141] Pal, N. R. and K., Pal S., Segmentation based on contrast homogeneity and region size, *IEEE Trans. on System, Man and Cybernetics*, 1987;17:857–868.

[142] Johnson, C.R., Beazley, D.M., Livnat, Y, Parker, S.G., Schmidt, J.A., Shen, H.W., and Weinstein, D.M., Applications of large-scale computing and scientific visualization in medicine, *Int. J. of Supercomputer Apps.*, 1996;.

[143] Faugeras, O., *Three-dimensional computer vision*, MIT Press, Cambridge, 1993.

[144] Rosenfeld, Azriel, *Image Modeling*, Academic Press, 1981.

[145] Russ, J.C., *The Image Processing Handbook*, CRC Press, Boca Raton, 1992.

[146] Schmidt, J., *Mesh generation with applications in computational electrophysiology*, PhD thesis, Duke University, Durham, NC, 1993.

[147] Lawson, C.L., Software for C1 surface interpolation, in *Mathematical Software II*, pp. 161–194. Academic Press, New York, 1977.

[148] Lee, D.T. and Schachter, B.J., Two algorithms for constructing a Delaunay triangulation, *Int. J. Comp. Inf. Sci.*, 1980;9:219–242.

[149] Lewis, B.A. and Robinson, J.S., Triangularization of planar regions with applications, *Comp. J*, 1978;21:324–332.

[150] Christiansen, H., MOSAIC triangulation algorithm, in *MOVIE.BYU Program Manual*. Engineering Computer Graphics Laboratory, Brigham Young University, Provo, Utah, 1987.

[151] Vesely, I., Eickmeier, B., and Campbell, G., Automated 3-D reconstruction of vascular structures from high definition casts, *IEEE Trans Biomed. Eng.*, 1991;38:1123–1129.

[152] Mercer, R.R., McCauley, G.M., and Anjilvel, S., Approximation of surfaces in a quantitative 3-D reconstruction system, *IEEE Trans Biomed. Eng.*, 1990;37:1136–1146.

[153] MacLeod, R.S., Johnson, C.R., and Matheson, M.A., Visualization tools for computational electrocardiography, in *Visualization in Biomedical Computing*, Bellingham, Wash., 1992, pp. 433–444, Proceedings of the SPIE #1808.

[154] Hoppe, H., Progressive meshes, in *Proceedings of ACM Siggraph '96*, July 1996, pp. 99–108.

[155] Hoppe, H., T., DeRose, Duchamp, T., Halstead, M., Jin, H., McDonald, J., Schweitzer, J., and Stuetzle, W., Piecewise smooth surface reconstruction, in *Proceedings of ACM Siggraph '94*, July 1994, pp. 295–302.

[156] Hoppe, H., T., DeRose, Duchamp, T., McDonald, J., and Stuetzle, W., Surface reconstruction from unorganized points, in *Proceedings of ACM Siggraph '92*, July 1992, pp. 71–78.

[157] Oostendorp, T.F., Oosterom, A.van, and Huiskamp, G.J., Interpolation on a triangulated 3D surface, *J. Comp. Physics*, 1989;80:331–343.

[158] George, P.L., *Automatic Mesh Generation*, Wiley, New York, 1991.

[159] Knupp, P. and Steinberg, S., *Fundamentals of Grid Generation*, CRC Press, Boca Raton, 1993.

[160] Thompson, J. and Weatherill, N.P., Structed and unstructed grid generation, in *High-Performance Computing in Biomedical Research*, Pilkington, T.C., Loftis, B., Thompson, J.F., Woo, S.L-Y., Palmer, T.C., and Budinger, T.F., Eds., pp. 63–112. CRC Press, Boca Raton, 1993.

[161] Golgolab, A., Malilleur tridimensionnel automatique pour des geometries complexes, Tech. Rep., Rapport de Recherche no. 1004, INRIA, 1989.

[162] Shephard, M.S. and Yerri, M.A., An approach to automatic finite element mesh generation, *Comp. in Eng.*, 1982;3:21–28.

[163] Shephard, M.S., Guerinoni, F., Flaherty, J.E., Ludwig, R.A., and Baehmann, P.L., Adaptive solutions of the Euler equations using finite quadtree and octree grids, *Computers and Structures*, 1988;30:327–336.

[164] Gitlin, C.S. and Johnson, C.R., Meshview: A tool for exploring 3D unstructured tetrahedral meshes, in *5th International Meshing Roundtable*, 1996, pp. 333–345.

[165] Gitlin, C.S., Techniques for visualizing 3D unstructured meshes, Master's thesis, University of Utah, Dec. 1995.

[166] Bertrand, O., 3D finite element method in brain electrical activity studies, in *Biomagnetic Localization and 3D Modeling*, Nenonen, J., Rajala, H.M., and Katila, T., Eds., pp. 154–171. Helsinki University of Technology, Helsinki, 1991.

[167] Barber, C.B., Dobkin, D.P., and Huhdanpaa, H., The quickhull algorithm for convex hull, Tech. Rep., Geometry Center Technical Report GCG53, 1993, A public domain two- and three-dimensional Delaunay mesh generation code. Available via anonymous ftp from: geom.umn.edu:/pub/software/qhull.tar.Z. There is also a geometry viewer available from geom.umn.edu:/pub/software/geomview/geomview-sgi.tar.Z.

[168] Johnson, C.R., MacLeod, R.S., and Schmidt, J.A., Software tools for modeling, computation, and visualization in medicine, in *CompMed 94 Proceedings*. 1996, World Scientific.

[169] Watson, D.F., Computing the n-dimensional Delaunay tesselation with applications to Voronoi polytopes, *Computer Journal*, 1981;24:167–172.

[170] Chapman, R.M. and McCrary, J.W., EP component identification and measurement by principal component analysis, *Brain and language*, 1995;27:288–301.

[171] Christensen, G. E., Rabbitt, R. D., and Miller, M.I., Deformable templates using large deformation kinematics, *IEEE Trans. on Image Processing*, 1995;in press.

[172] Rabbitt, R. D., Weiss, J. A., Christensen, G. E., and Miller, M. I., Mapping of hyperelastic deformable templates using the finite element method, 1995;2573:252–265.

[173] Bowyer, A, Computing dirichlet tesselations, *Computer J.*, 1981;24:162–166.

[174] Hoole, S.R.H., *Computer-Aided Analysis and Design of Electromagnetic Devices*, Elsevier, New York, 1989.

[175] MacLeod, R.S., Johnson, C.R., and Matheson, M.A., Visualization tools for computational electrocardiography, in *Visualization in Biomedical Computing*, 1992, pp. 433–444.

[176] MacLeod, R.S., Johnson, C.R., and Matheson, M.A., Visualization of cardiac bioelectricity — a case study, in *IEEE Visualization '92*, 1992, pp. 411–418.

[177] Pilkington, T.C., Loftis, B., Thompson, J.F., Woo, S.L-Y., Palmer, T.C., and Budinger, T.F., *High-Performance Computing in Biomedical Research*, CRC Press, Boca Raton, 1993.

[178] Thompson, J.F., Warsi, Z.U.A., and Mastin, C.W., *Numerical Grid Generation*, North-Holland, New York, 1985.

[179] Henneberg, K. and Plonsey, R., Boundary element analysis in bioelectricity, in *Industrial Applications of the Boundary Element Method*, Brebbia, C.A. and Aliabadi, M.H., Eds., pp. 95–128. Springer-Verlag, 1993.

[180] Henriquez, C.S., Johnson, C.R., Henneberg, K.A., Leon, L.J., and Pollard, A.E., Large scale biomedical modeling and simulation: from concept to results, in *Frontiers in Biomedical Computing*, Thakor, N., Ed. IEEE Press, Philadelphia, 1995.

[181] Strikwerda, J.C., *Finite difference schemes and partial differential equations*, Wadsworth & Brooks/Cole Advanced Books & Software, Pacific Grove, 1989.

[182] Allen, M.B., Herrera, I., and Pinder, G.F., *Numerical Modeling in Science and Engineering*, Wiley, New York, 1988.

[183] Isaacson, E. and Keller, H.B., *Analysis of Numerical Methods*, Dover, New York, 1994.

[184] Young, D.M. and Gregory, R.T., *A Survey of Numerical Mathematics, Vols. I and II*, Dover, New York, 1973.

[185] Golub, G.H. and Ortega, J.M., *Scientific Computing: An Introduction with Parallel Computing*, Academic Press, Boston, 1993.

[186] Ames, W., *Numerical Methods for Partial Differential Equations*, Academic Press, New York, 1977.

[187] Twizell, E.H., *Computational Methods for Partial Differential Equations*, Halsted Press, New York, 1984.

[188] Botha, J.F. and Pinder, G.F., *Fundamental Concepts in the Numerical Solution of Differential Equations*, Wiley-Interscience, New York, 1983.

[189] Lapidus, L. and Pinder, G.F., *Numerical Solution of Partial Differential Equations in Science and Engineering*, Wiley-Interscience, New York, 1982.

[190] Akin, J.E., *Finite Element Analysis for Undergraduates*, Academic Press, New York, 1986.

[191] Johnson, C., *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, 1990.

[192] Tong, P. and Rossettos, J.N., *Finite-Element Method*, MIT Press, Boston, 1977.

[193] Silvester, P. and Ferrari, R.L., *Finite Elements for Electrical Engineers*, Cambridge University Press, Cambridge, England, 1983.

[194] Ciarlet, P.G. and Lions, J.L., *Handbook of Numerical Analysis: Finite Element Methods*, vol. 1, North-Holland, Amsterdam, 1991.

[195] Zienkiewicz, O.C. and Taylor, R.L., *The Finite Element Method, Vols. I and II*, McGraw-Hill, New York, 4th edition, 1988.

[196] Huebner, K.H., Thornton, E.A., and Byrom, T.G., *The Finite Element Method for Engineers*, Wiley, New York, 3rd edition, 1995.

[197] Jackson, J.D., *Classical Electrodynamics*, John Wiley, New York, 1975.

[198] Brebbia, C.A. and Dominguez, J., *Boundary Elements: An Introductory Course*, McGraw-Hill, Boston, 1989.

[199] Jawson, M.A. and Symm, G.T., *Integral Equation Methods in Potential Theory and Elastostatics*, Academic Press, London, 1977.

[200] Beer, G. and Watson, J.O., *Introduction to Finite and Boundary Element Methods for Engineers*, Wiley, New York, 1992.

[201] Barr, R., Pilkington, T., Boineau, J., and Spach, M., Determining the surface potentials from current dipoles with application to electrocardiography, *IEEE Transactions on Biomedical Engineering*, 1966;13:88–92.

[202] Plonsey, R., *Bioelectric Phenomena*, McGraw-Hill, New York, 1969.

[203] Gulrajani, R.M., Roberge, F.A., and Mailloux, G.E., The forward problem of electrocardiography, in *Comprehensive Electrocardiology*, Macfarlane, P.W. and Lawrie, T.D., Eds., pp. 197–236. Pergamon Press, Oxford, England, 1989.

[204] Meijs, J.W.H., Weier, O.W., Peters, M.J., and Oosterom, A.van, On the numerical accuracy of the boundary element method, *IEEE Trans Biomed. Eng.*, 1989;36:1038–1049.

[205] Steele, C.W., *Numerical Computation of Electric and Magnetic Fields*, Van Nostrand Reinholt, 1987.

[206] Brebbia, C.A., Telles, J., and Wrobel, L., *Boundary Element Techniques*, Springer-Verlag, New York, 1984.

[207] Gipson, G., Boundary element fundamentals - basic concepts and recent developments in the poisson equation, in *Topics in Engineering*, Brebbia, C.A. and Connor, J.J., Eds., vol. 2. Computational Mechanics Publications, Boston, 1987.

[208] Computational Mechanics Publications Inc., 25 Bridge Street, Billerica, MA 01821, Telephone: (508)667-5841, Fax: (508) 667-7582.

[209] Bradley, C.P and Pullan, A.J., A coupled cubic hermite finite element/boundary element procedure for electrocardiographic problems, *Computational Mechanics*, 1997;.

[210] Bradley, C.P, Pullan, A.J., and Hunter, P.J., Geometric modelling of the human torso, *Annals of Biomed. Eng.*, 1997;.

[211] Pullan, A.J., A high order coupled finite element/boundary element torso model, *IEEE Trans. on BioMed. Eng.*, 1996;:292–298.

[212] Pullan, A.J., A high order coupled finite element/boundary element torso model, *IEEE Trans. on BioMed. Eng.*, 1995;.

[213] Pullan, A.J., Bradley, C.P., and Hunter, P.J., A high order coupled finite element/boundary element torso model, in *Computers In Cardiology Conference*, 1993.

[214] Cuthill, E. and McKee, J., Reducing the bandwidth of sparse symmetric matrices, *ACM Proceedings of the 24th National Conference (New York)*, 1969;.

[215] Cuthill, E., Several strategies for reducing the bandwidth of matrices, in *Sparse Matrices and Their Applications*, Rose, D.J. and Willoughby, R.A., Eds. Plenum Press, New York, 1972.

[216] Liu, W.H. and Sherman, A.H., Comparative analysis of cuthill-mckee and the reverse cuthill-mckee ordering algorithms for sparse matrices, *SIAM J. Num. Anal.*, 1976;13:198–213.

[217] Jennings, A. and McKeown, J.J., *Matrix Computation*, Wiley, New York, 1992.

[218] Eisenstat, S.C., Gursky, M.C., Schultz, M.H., and Sherman, A.H., Yale sparse matrix package, Tech. Rep., Research Report 112, Dept. of Computer Science, Yale University, 1977.

[219] Saad, Y., SPARSKIT: a basic tool kit for sparse matrix computations, Tech. Rep., Dept. of Computer Science, Univ. of Minnesota, 1993, Available via Netlib.

[220] Duff, I.S., Erisman, A.M., and Reid, J.K., *Direct Methods for Sparse Matrices*, Claredon Press, Oxford, 1986.
[221] Saad, Y., *Iterative Methods for Sparse Linear Systems*, PWS, Boston, 1996.
[222] Saad, Y., ILUT: A dual threshold incomplete LU factorization, Tech. Rep., Dept. of Computer Science, Univ. of Minnesota, 1992.
[223] Demmel, J.W., *Numerical Linear Algebra*, SIAM Press, Philadelphia, 1997.
[224] Johnson, C.R. and MacLeod, R.S., High performance computing in medicine: Direct and inverse problems in cardiology, in *IEEE Engineering in Medicine and Biology Society 15th Annual International Conference*, 1993.
[225] Hutchinson, S.A., Hensel, E.C., Castillo, S.P., and Dalton, K.E., The finite element solution of elliptical systems on a data parallel computer, *Intl. J. for Numer. Meth. in Eng.*, 1991;32:347–362.
[226] Hutchinson, S.A., Ng, K.T., and Shadid, J.N., A finite element algorithm for elliptical equations over irregular domains on a data parallel computer, *Intl. J. for Numer. Meth. in Eng.*, 1994;37:3153–3167.
[227] Davis, P.J. and Rabinowitz, P., *Methods of Numerical Integration*, Academic Press, New York, 1984.
[228] Freeman, T.L. and Phillips, C., *Parallel Numerical Algorithms*, Prentice Hall, New York, 1992.
[229] Van de Velde, E.F., *Concurrent Scientific Computing*, Springer-Verlag, New York, 1994.
[230] Bertsekas, D.P. and Tsitsiklis, J.N., *Parallel and Distributed Computation: Numerical Methods*, Prentice-Hall, Englewood Cliffs, 1989.
[231] Gallivan, K.A. and al., et, *Parallel Algorithms for Matrix Computations*, SIAM, Philadelphia, 1990.
[232] Schendel, U., *Introduction to Numerical Methods for Parallel Computers*, Ellis Harwood, Chichester, 1984.
[233] Kumar, V., Grama, A., Gupta, A., and Karypis, G., *Introduction to Parallel Computing*, Benjamin/Cummings, New York, 1994.
[234] Hansen, P.B., *Studies in Computational Science*, Prentice Hall, New York, 1995.
[235] Loshin, D., *High Performance Computing Demystified*, Academic Press, New York, 1994.
[236] Morse, H.S., *Practical Parallel Computing*, Academic Press, New York, 1994.
[237] Gilbert, J. and Kershaw, D., Eds., *Large-scale Matrix Problems and the Numerical Solution of Partial Differential Equations*, Claredon, Oxford, 1994.
[238] Burnett, D.S., *Finite Element Method*, Addison Wesley, Reading, Mass., 1988.
[239] Zienkiewicz, O.C., *The Finite Element Method in Engineering Science*, McGraw-Hill, New York, 1971.
[240] Zienkiewicz, O.C. and Zhu, J.Z., A simple error estimate and adaptive procedure for practical engineering analysis, *Int. J. Num. Meth. Eng.*, 1987;24:337–357.
[241] Zienkiewicz, O.C. and Zhu, J.Z., Adaptivity and mesh generation, *Int. J. Num. Meth. Eng.*, 1991;32:783–810.
[242] Burnett, D., *Finite Element Analysis- From Concepts to Applications*, Addison-Wesley, Reading, MA, 1987.
[243] Lewis, R., Huang, H., Usmani, A., and Cross, J., Finite element analysis of heat transfer and flow problems using adaptive remeshing including application to solidification problems, *International Journal of Numerical Methods in Engineering*, 1991;32:767–781.
[244] Zienkiewicz, O. and Zhu, J., A simple error estimator and adaptive procedure for practical engineering analysis, *International Journal of Numerical Methods in Engineering*, 1987;24:337–357.
[245] Zienkiewicz, O. and Zhu, J., Adaptivity and mesh generation, *International Journal of Numerical Methods in Engineering*, 1991;32:783–810.
[246] Flaherty, J.E., *Adaptive Methods for Partial Differential Equations*, SIAM, Philadelphia, 1989.
[247] Gallopoulos, S., Houstis, E., and Rice, J.R., Computer as thinker/doer: Problem-solving environments for computational science, *IEEE Computational Science and Engineering*, 1994;:11–21.
[248] Johnson, C.R. and Parker, S.G., A computational steering model for problems in medicine, in *Supercomputing 94*. 1994, pp. 540–549, IEEE Press.
[249] Johnson, C.R. and Parker, S.G., Applications in computational medicine using SCIRun: A computational steering programming environment, in *Supercomputer '95*, Meuer, H.W., Ed. 1995, pp. 2–19, Springer-Verlag.
[250] Parker, S.G. and Johnson, C.R., SCIRun: A scientific programming environment for computational steering, in *Supercomputing '95*. 1995, IEEE Press.