Software Tools for Modeling, Computation, and Visualization in Medicine

Christopher R. Johnson
Department of Computer Science, University of Utah
Salt Lake City, UT, USA
crj@cs.utah.edu

and

Robert S. MacLeod
Nora Eccles Harrison CVRTI, University of Utah
Salt Lake City, Utah, 84112, USA
macleod@cvrti.utah.edu

and

John A. Schmidt
Department of Computer Science, University of Utah
Salt Lake City, Utah, 84112, USA
jas@cs.utah.edu

ABSTRACT

In this paper we describe modeling, computation, and visualization software tools applied to problems in computational medicine. We focus the application of these software tools on a class of bioelectric field problems which arise in electrocardiography and electroencephalography.

Keywords: Computational Medicine, Modeling, Software Tools, Visualization.

1. Introduction

Computer modeling and simulation continue to grow in importantce in many fields of Medicine. The reasons for this growing importance are manifold. First, mathematical modeling has been shown to be a substantial tool for the investigation of complex biophysical phenomena [1,2]. Second, since the level of complexity one can model parallels existing hardware configurations and software tools, advances in computer architecture and application software have made it feasible to apply the computational paradigm to more accurate models of biophysical systems. Hence, while biological complexity continues to outstrip the capabilities of even the largest computational systems, the computational methodology has taken hold in medicine and has been used successfully to suggest physiologically and clinically important scenarios and results.

This paper provides an overview of software tools for solving problems in computational medicine that have been developed by the Scientific Computing and Imaging (SCI) research group at the University of Utah. We describe software tools for geometric modeling, computation, and visualization for problems in medicine. We focus the application of these software tools on bioelectric field problems which arise in electrocardiography and electroencephalography.

2. Modeling

In most computational science applications, considerable geometric modeling must take place prior to simulation and visualization. Modeling efforts usually take the form of estimating a continuous physical structure with a discrete collection of nodes connected into polygons (polyhedra).

In this section we describe two (often essential) modeling procedures, segmentation and mesh generation. We have constructed large scale torso and head models from MRI scans, within which, we solve bioelectric field problems. To classify the relevant tissues we have developed two semi-automatic segmentation algorithms [3,4]. To construct three-dimensional surfaces and/or volumetric meshes that are used for computation and visualization, we utilize two- and three-dimensional Delaunay triangulation algorithms [5,6,7].

2.1. Segmentation

Image segmentation, the process of defining boundary domains in 2D images, takes on an important role in biomedical model building applications. Given slices of 2D images, the boundaries of interesting regions must be defined before surface reconstruction, mesh generation, and other modeling operations begin. Although image segmentation and contour/edge detections have been investigated for quite a long time [8,9,10], there is still no algorithm that can automatically find region boundaries perfectly 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 background knowledge about the region(s) of interest.

To ensure the accuracy of the segmentation, many researchers still manually segment the image by inserting control points on the image by hand and then invoking data fitting algorithms to fit curves. Owing to the complexity and the large number of images needed to be processed for construction of a typical model, the segmentation process has been recognized as one of the most signficant challenges in the modeling process.

We have developed a semi-automatic image segmentation tool which combines conventional manual segmentation utilities with a novel automatic image segmentation algorithm. Manual segmentation is achieved by dropping control points and fitting cubic splines to selected points. Automatic segmentation is achieved by bimodally thresholding local windows of the target image to produce boundary pieces. The individual pieces are connected to yield the entire region boundary. By combining these two segmentation methods, a user can obtain accurate boundary descriptions with a minimum of effort.

The manual segmentation procedure is straightforward. Given a 2D image slice, the user places numerous control points on a visible region boundary. A data fitting algorithm is then invoked to interpolate between these control points to generate a region contour. Several different data fitting methods can be used. The methods differ based on either the degree of interpolation function used, i.e. linear, quadratic or on the basis function used, i.e. natural spline, Bezier spline, etc. In our implementation, we used a cubic spline interpolation method.

We developed an automatic algorithm that allows users to interactively steer the segmentation process. The program starts by prompting the user to select an initial starting point on the image. Local edge detection and contour following programs are then started from the initial point to find the region boundaries. Only a small contour segment piece is given at a time. The program then waits for feedback from the user. The user has the option to correct the identified contour segment based on a visual inspection. If no corrections are needed, the program continues to find the next contour piece.

We use a bimodal thresholding algorithm to determine the boundary segments in the local region. When the user selects an initial point, a small local window is placed around this point. A local histogram is then computed according to the values of pixels located inside the window. If part of a region boundary passes through the local window, the histogram should have a bimodal appearance, i.e, two peaks in the distribution. The local minimum between these two peaks determines the threshold value used to segment the local image.

After a boundary segment is found, the local window then moves in the direction of the contour to the next position. Currently, we position the window such that the end point of the previous contour segment is at the center of the next local window. Although this creates some overlap between consecutive local images, the overlap reduces the chance of discontinuities between contour segments. By combining automatic boundary detection for local regions with the manual corrections, we achieve a correct and effective means of segmentation.

While our original segmentation software tool used a technique that considered a single grey-scale value for each voxel, we have recently extended this segmentation technique to include an n-dimensional vector for each voxel. This technique requires multiple volume data sets as input, 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. In our example with MRI scans, the patient underwent two scans with different "weightings." 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. For this example, we have two volumes, each containing 81 256 x 256 MRI slices from the same patient.

This software module, called *VecSeg* reads in the data files and first passes the volumes to clipping modules. These modules are linked so that they will always define the same volume (i.e. if you move one volume's clipping planes, the other's move as well). If these modules were not included, segmenting the volume would take a long time, and the process of determining the best material ranges would not be nearly as interactive. By using the clipping modules, the user can interact with a small subset of the domain initially, and then, once the desired ranges have been determined, can pass the entire volume in to be segmented.

The next stage in the 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 this case, 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.

The result of the vector segmentation module is 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.

2.2. Mesh Generation

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 one can use 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 often 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 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 which matches the number of degrees of freedom of our fundamental element.

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 (eg., heart, lungs, skeletal muscle, vascularture and the body surface). 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 geometrical objects that comprise the geometrical model are described by segmented sets of 2-D MRI or CT images consisting of contours (polylines or splines) connecting discrete points at the boundaries between organs and/or regions of different conductivity. The resulting point set is then used to construct a polygonal representation of the solution domain.

For the construction and simulation of anatomically correct models we utilized unstructured meshes of triangular or tetrahedral elements. Unstructured grids tend to capture the complex structures of human and animal anatomy more efficiently than structured grids. Meshes used for the research presented in the Selected Applications section range in size from thousands of elements for 2D models to hundreds of thousands to millions of elements for 3D models.

The method we used to create meshes is based upon the Delaunay tessellation algorithm originally proposed by Watson [6] and later extended by Weatherhill [7]. The Delaunay criterion states that the circumsphere of any tetrahedron (triangle)* 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 [11,5]. A more detailed description of the mesh generation algorithm can be found in [5].

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 tesselation until certain spacing criteria are satisfied.

The final step is to classify the tetrahedra by their material properties, e.g.,

*In describing the algorithm, we use triangle or tetrahedron interchangeably.

which tetrahedra are considered to be in the heart, lungs, et cetera. Since the material property of interest in bioelectric problems, electric conductivity, can be anisotropic, each tetrahedron must be assigned a tensor describing local conductivity. To classify a tetrahedron we had to localize the position of its centroid relative to the surfaces separating regions of different conductivity. To localize the element, we utilized a ray tracing approach [6] which 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.

3. Computation

Over the past several years we have developed several programs to solve computationally intensive problems in medicine (focusing primarily on bioelectric field problems), such as (1) interactive programs to construct, manipulate, and display large scale, three-dimensional surface and volumetric meshes [12,13,14,15]; (2) three-dimensional, adaptive, finite element programs for solving partial differential equations with general boundary conditions and source terms [12,16,17,5]; and (3) programs that solve the resulting large system of equations and simultaneously apply regularization to ill-posed inverse problems [12,18]. Because our work involves a large-scale problems, much of the software we have developed has been designed to take advantage of parallel processing via high performance architectures.

3.1. Approximation Methods

Due to the geometrical complexity of, and numerous inhomogeneities inherent in, anatomical structures in physiological models, solutions of bioelectric field problems (and many other problems in physiological modeling and simulation) 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), or the Boundary Element (BE) approach to solve the governing field equation(s).

Since we have limited space, we will not be able to give a quantitative description of each of these methods, rather we give an abbreviated summary of the applicability of each method in solving different types of bioelectric field problems.

The FD, FE, and BE methods can all be used to approximate the boundary value problems which 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, a BE program requires less book-keeping 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.

3.2. Adaptive Methods

Thus far we have mentioned how one discretizes the 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 simulation results? This is a topic that is often ignored by computational scientists (in particular, it is the opinion of the authors that practitioners of computational medicine have been slow to address this issue), thus we include a brief introduction to adaptive methods. To provide an intuitive feel for the problem (and possible solution), consider the approximation of a two-dimensional region discretized into triangular elements. Assume the finite element method will be used 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):

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

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. This means that the error due to discretization (from using linear elements) is proportional to the third term of the Taylor's expansion:

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

where Φ is the exact solution. Thus the error due to discretization for first order linear elements is proportional to the second derivative such that the gradient must be constant over each element. If the gradient is not constant over an element, the second derivative is not zero and is proportional to the error incurred due to "improper" discretization. Examining (2) 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 gradients decreases the error.

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

An approach to mesh refinement that we have investigated is to estimate the solution accuracy by performing an error analysis on the finite element approximation. We then utilize this *a posteriori* error estimate to refine the mesh in an iterative fashion until the errors drop below a pre-defined tolerance.

Briefly, suppose we want the accuracy of the finite element approximation to be within some given tolerance, δ, of the true solution, i.e.,

$$|\Phi - \Phi_h|_{H^1(\Omega)} \ \leq \ \delta. \tag{3}$$

We can utilize an error estimator, [19,20,17], to refine the mesh so that the resulting error decreases to within the tolerance given in (3). This error estimator is

$$\sum_{K \in T_h} \left(h_K |\Phi_h|_{H^2(K)}\right)^2 \ \leq \ \frac{\delta^2}{C^2}, \tag{4}$$

where h_K is a measure of size of an element, $|\Phi|_{H^m(\Omega)} = (\sum_{|\alpha|=m} \int_\Omega |D^\alpha \Phi|^2 dx)^{\frac{1}{2}}$ measures the $L_2(\Omega)$-norm of the partial derivatives of Φ of order equal to m and signifies the H^m seminorm of the finite element approximation, and C is a positive constant that depends on the size of the second partial derivative of Φ and the smallest angle formed by the sides of the elements in V_h.

We apply this error estimator to all the K elements in our tessellated geometry, T_h, and compare the error to the *normalized* tolerance, which is defined to be the right hand side of (4). We note that the constant C is often difficult to estimate, especially for non-uniform meshes, and we have utilized approximations based on simplified meshes [17]. If the value of the error estimate in a given element exceeds the tolerance, then the element is 'flagged' for further refinement. The refinement can be realized by subdividing the egregious element (so-called h-refinement), by increasing the order of the basis function of the element (p-refinement), or by doing both (hp-refinement). Obviously, with each iteration of the mesh refinement algorithm, the number of unknowns increases as does the time for each subsequent iteration, so that, once again, the principle of *conservation of complexity* applies. Achieving greater accuracy increases computational costs; realistic trade-offs need to be made based on the size of the problem and computational resources available [21,19,17,22,23,5].

4. Visualization

Visualization systems play an integral role in all phases of computational modeling, and we have developed a variety of application programs based on several common visualization paradigms [13,14,15,24]. Map3d is an interactive three-dimensional GL based graphics system for viewing surface and volume geometry and simulation results. For higher quality ray-traced rendered images, we have developed graphics systems which exploits parallel processors and/or clusters of workstations. A recent development is a differential volume rendering module which allows nearly real-time rendering of three-dimensional flows for models with hundreds of thousands of elements [25].

4.1. Map3d

Early in our model development, access to flexible, powerful visualization tools arose as necessity in dealing with complex geometries and large datasets. In particular, we first needed an interactive means of viewing and manually editing the three-dimensional geometric models that segmentation of medical images provided. A supplementary task, which followed as soon as we began to apply electric potential data to the models, was to display scalar and vector information on the model geometry. While we were convinced that graphics hardware at the time (early 1990's) was adequate for our needs, the visualization software was still relatively immature and we found no packages capable of handling the large geometrical models the project was generating. Out of this environment, the program *Map3d* was developed [26,27]. In the interim, *Map3d* has continued to evolve in order to encompass new features and utilize improvements in hard- and software.

4.1.1. Program structure

The basic organizational element of *Map3d* is the 'metasurface', minimally defined as a set of nodes, which need not form a closed surface but are in some sense logically connected. To the nodes of a metasurface can be added connectivities, in the form of line segments, triangles, tetrahedra, or hexahedra. The capabilities of *Map3d* include display of the geometry as a set of nodes, connected polygons (triangles or tetrahedra), or rendered surface, with indication of the orientation of the surface facets ("outward" or "inward" facing). In order to analyze and edit the geometry, we needed to be able to interrogate elements of the geometry, to link nodes into triangles (tetrahedra), remove existing triangles (tetrahedra), and to flip the orientation of surface facets. Managing complex geometries also required that we be able to combine multiple surfaces in the same display and control them both synchronously and independently, rotate and translate the geometry, and apply adjustable clipping planes, depth cueing and perspective.

We were able to maintain acceptable interactive performance by dividing large geometries (80,000 nodes and 500,000 tetrahedra) into smaller, independent units, which *Map3d* could manipulate individually, or in groups. *Map3d* provides a very simple, efficient interface that allows users to select the best compromise of features to balance display quality and interactive responsiveness.

4.1.2. Data Visualization

Geometrical models provide a mesh for computation, but also define the underlying structure for visualization of both experimental and modeling results [27]. Recording electrode sites can either map directly to nodes of the mesh or interpolation can provide a means of assigning sparsely recorded data to the entire mesh. Within the surface-based paradigm of most bioelectrical measurements in cardiology (body surface potential mapping, epicardial heart mapping, *etc.,*), we have used an interpolation scheme based on minimzation of an approximation of the three-dimensional, surface Laplacian operator [28]. For data recorded or computed in a volume (*e.g.,* heart potentials recorded with intramyocardial needle electrodes or finite element computation of torso volume potentials), one can also define artificial surfaces based on proximity or other natural relationships between nodes.

Map3d incorporates data visualization by extending the metasurface paradigm to include data values, *e.g.,* electric potential, activation times, or current vectors, by linking them to the elements of the geometry. To each element (node or connected polygonal element) of the metasurface can be associated any number of scalar, vectorial, or tensorial quantities. Scalar values are bilinearly interpolated over the triangular surface elements and displayed as isocontour lines, or Gouraud shaded surfaces. Vectors are drawn as colored arrows anchored at either the nodes or centroids of elements. Variable settings determine the color map and scaling parameters, and control the display of textual labels with absolute data (or node number) values at the appropriate sites in the display. To provide a temporal context for the spatial displays, *Map3d* displays selected individual time signals (scalar lead) in separate windows. The scalar lead window also provides an interactive (via the mouse) means of selecting the instant in time at which to display the spatial distributions in the other windows.

The visualization of data values in *Map3d* is based entirely on surfaces, *i.e.,* there is no volume rendering. This reflects both performance concerns and the fact that most bioelectric measurement systems record values at relatively few (in comparison to imaging modalities like MRI and ultrasound), discrete sites, which are often bound to a physical surface, as, for example, in heart, body, and cranial surface mapping. With scalar data provided on tetrahedralized volume meshes, *Map3d* can derive and display multiple isosurfaces.

In order to capture the temporal continuity of datasets in some form of hardcopy output, we have devised a video animation driver interface for *Map3d*. The animator allows complete control of *Map3d*, generating sequences of frame files or even controlling directly a video output card and, via a video local area network (V-LAN), a frame-accurate video recorder. Hard copy output of individual frames makes use of both the image capture and graphics pipeline feedback features of the Silicon Graphics graphics engine. At any time, or under control of an animation script, the user can capture the contents of all active *Map3d* windows either as a raw image file, or a postscript page description.

4.2. Volume rendering

Direct volume-rendering techniques are effective tools for exploring 3D scalar data. Unlike surface-rendering methods, direct volume-rendering methods can be used to visualize 3D scalar data without converting intermediate geometric primitives. By assigning appropriate colors and opacities to the scalar data, one can render objects semi-transparently to expand the amount of 3D information available at a fixed position. Volume-rendered images can also be superimposed upon surface-oriented icons or textures, thus allowing for simultaneous scalar and vector field composite visualizations.

We were motivated to develop a more efficient way to visualize scalar fields by our attempts to visualize simulation data from a model of electrical wave propagation within the complex geometry of the heart and in large scale models of unsteady compressible fluid flow [29,30]. Because of the regular structures used to characterize this particular set of simulation data, we can use direct volume-rendering techniques to visualize the states at each time step. We characterize states within the model by assigning different colors and opacities. By animating the volume-rendered images at each time step, we can effectively investigate the propagation of waves throughout the volume.

Direct volume-rendering methods employing ray casting algorithms have become the standard methods to visualize 3D scalar data. To characterize the dynamic behavior of the flow field, one generates a series of volume rendered images at different time steps and then records, stores, and animates the sequence. Because direct volume rendering is so time consuming to realize animations for models of any significant size (i.e. realistic problems in science and engineering), standard volume-rendering techniques prove prohibitive for animating hundreds of time steps interactively. Moreover, the disk space required for storing hundreds or thousands of sets of volumetric simulation data could be overwhelming.

We have developed of an algorithm which significantly reduces the time to create volume-rendered flow animations of scalar fields. Furthermore, our algorithm reduces the amount of disk space needed for storing volume data. We achieve these reductions by implementing a differential volume-rendering method. The method utilizes data coherency between consecutive time steps of simulation data to accelerate the volume animation and to compress the volume data. The method is independent of specific volume-rendering techniques and can be adapted into a variety of ray casting paradigms which can be used to further accelerate the visualization process [31,32,33].

From preliminary studies of our electrical wave propagation simulations in the heart, we noticed that the only elements which changed values between consecutive time steps, when the time steps were small, were the activated cells and their neighbors. We hypothesized that only a fraction of elements in the volume change from any given time step to the next in simulations of physical flow phenomena. In addition, when a sequence of propagating images is animated, the viewing parameters usually don't change. In our ray casting method, we are able to cast rays along a path corresponding only to changed data elements. Therefore, the pixels in the new image keep the same colors that they had previously unless they correspond to changed data elements. Retaining the color values from the non-changing pixels results in a significant time savings. The differential volume-rendering algorithm thus exploits the temporal coherence between sets of volume data from different time steps in order to speed up volume animation of the 3D flow.

The differential volume rendering method separates the data generation and data visualization processes. Scientists perform simulations to obtain sequences of data at different time steps. The differential volume-rendering algorithm extracts the differential information which contains the differences between the data files at each consecutive time step. According to the specified viewing direction, the pixel positions where new rays need to be cast can be computed from the differential information and then the ray casting process is invoked to produce the updated image. Because the variation between consecutive time steps is small, the differential information file, which replaces the whole sequence of volume data, can yield tremendous savings in terms of disk space [34].

5. Selected Applications

In the next section we describe two specific applications of computer modeling in the field of cardiology. One is a diagnostic method utilizing noninvasive electrical measurements from the body surface to infer the electrical state of the heart. The other simulates a therapeutic approach used in cases of severe electrical disfunction of the heart, defibrillation. Finally we end with an overview of a new software architecture we have developed that integrates modeling, computation, and visualization into a single framework.

5.1. Cardiac Defibrillation

To stop the aberrant rhythms that characterize fibrillation, electrical shocks are delivered to the heart, the process of defibrillation. The design and placement of defibrillation electrodes is a critical issue for effectively restoring normal rhythm. Inefficient electrode design and placements can require larger quantities of current that can cause significant damage to the heart. In the past, researchers would use several series of animal trials to estimate effective electrode locations. Computer models have recently become sophisticated enough to assist in studying the defibrillation problem. Early computer models usually required significant amounts of time to construct and change model paramaters such that relatively few electrode configurations have been simulated [35]. We have developed a defibrillation modeling system that allows a researcher to interactively design and place electrodes into a high-resolution finite element model of the human thorax (the Utah Torso Model [12,16]). The advantages of this system are that different electrode combinations can be tested computationally, reserving the most promising candidates for animal studies. By reducing the number of experiments, there is the potential for tremendous savings in device design and development costs.

5.1.1. Model Construction and Electrode Design

The Utah Torso Model [12,13] was derived from MRI scans and digitized 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, subclavian, innominate and azygous veins). Figure 1 depicts a cutaway of the model highlighting the different anatomical regions.

Rectangular shapes were chosen for the subcutaneous patch electrodes and cylinders were chosen for the internal electrodes. The electrodes were deformed to fit inside the major vessels and/or to conform to the ribs with an interactive modeling tool. Once the electrodes were defined, a volumetric mesh was generated by first creating a surface mesh consisting of triangles used to place the electrodes within the thorax surface triangles. Following the surface construction, volume points were automatically created and tesselated into tetrahedral elements using a Delaunay mesh generation algorithm [35]. Dirichlet boundary conditions were then assigned to the electrode surfaces. The model was defined by over 20,000 surface points (including the electrodes). The resulting volume mesh (we created nearly 50 unique meshes to handle over 170 different electrode combinations) ranged from 1.2 million to 1.5 million elements. The finite element method was then used to solve for the potential and current density distributions.

5.1.2. Defibrillation Criteria

Five different defibrillation criteria were used to determine the efficacy of various electrode configurations. These included *electrode voltage threshold (EVT)*, *efficiency (Eff)*, *damage*, *uniformity (Unif)*, and *lead impedance* $\mathrm{Z_L}$. The EVT was defined as the voltage difference across the electrodes necessary to yield a gradient of either 5 V/cm, (monophasic shocks) or 3.5 V/cm, (biphasic shocks) in 95% of the heart tissue. The **efficiency** of the delivered shock was based on the ratio of the energy reaching the heart to the total energy delivered. The **damage** level was determined to be the percentage of heart tissue above a particular gradient level. The gradient levels used were 50 V/cm for monophasic shocks and 70 V/cm for

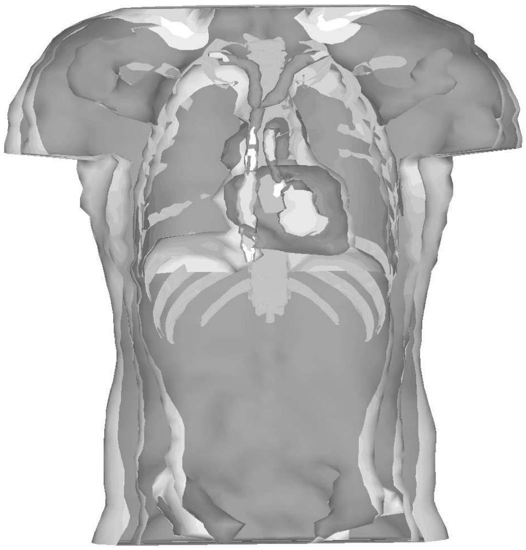

Figure 1: Cutaway of the Utah Torso Model showing the different anatomical regions included in the model.

biphasic shocks. The **uniformity** of the delivered shock was based on the ratio of the peak electric field in the heart volume to the average electric field in the heart volume. It ranges from 1 for a perfectly uniform field to ∞ for a very non-uniform electric field. The **lead impedance** is the impedance across the electrodes and is commonly determined in actual experimental and clinical settings. This serves as an additional check of the model to experimental data.

5.1.3. Defibrillation Results

Over 170 different electrode combinations were implemented and computationally tested with this interactive modeling tool. Preliminary results showed that several of the defibrillation criteria parameters were within the ranges measured in clinical and experimental settings. Shown in Figure 2 is the current density distribution on the heart surface for a particular defibrillation electrode configuration. The darker regions indicate high current density magnitudes. Table 1 compares two different electrode configurations that we explored. The first case refers to two cylindrical leads positioned in a vessel and blood cavity. The second case refers to the same lead system with an additional anodal subcutaneous patch. Case 2 with a subcutaneous patch has a significantly lower threshold voltage compared to case 1. Based on the criteria that smaller electrode voltage thresholds are better, case 2 could possibly merit further testing.

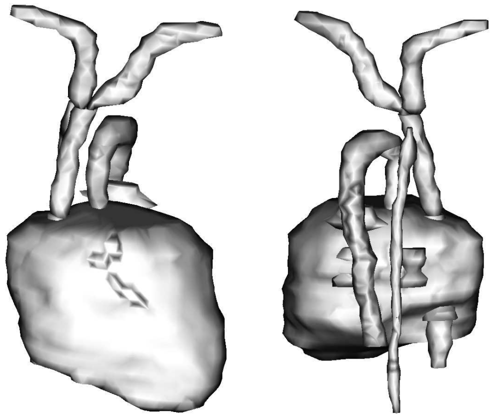

Figure 2: Figure showing the current density magnitude distributions for both the anterior and posterior views of the mediastinum for a typical defibrillation configuration.

	EVT			Damage			
	Mono	Bi	Eff	Mono	Bi	Unif	Z_L
1	801.8	561.3	18.7	14.3	3.0	14.0	48.5
2	389.7	272.8	17.2	3.5	0.02	7.8	34.6

Table 1: A sample comparison of the defibrillation criteria results for two different cases. Case 1 is for an anode and cathode lead system. Case 2 uses the same lead system as case 1, but it also includes an anodal subcutaneous patch electrode.

5.2. The Inverse ECG Problem

The intrinsic bioelectric source currents in the heart gives rise to electric, and thus electric potential, fields within the volume of the thorax and upon the torso surface. The voltages on the torso surface are related to voltages upon the heart surface via the resistive properties of the intermediary tissues of the thoracic cavity (volume conductor). The general inverse problem in electrocardiography can be stated as follows: given a subset of electrostatic potentials measured on the surface of the torso and the geometry and conductivity properties of the thorax, calculate the potential fields on, and source currents within, the heart. Mathematically this can be posed as an inverse source problem in terms of the primary current sources within the heart and described mathematically by Poisson's equation for electrical conduction:

$$\nabla \cdot \sigma \nabla \Phi = -I_v \quad \text{in } \Omega \tag{5}$$

with the boundary condition:

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

where Φ are the electrostatic potentials, σ is the conductivity tensor, I_v are the cardiac current sources per unit volume, and Γ_T and Ω represent the surface and the volume of the thorax, respectively. The goal of the inverse solution is to recover the magnitude and location of the cardiac sources. Alternatively, it is possible to formulate the problem in terms of the electrostatic potentials on a surface bounding the heart. This form of the inverse problem does have a unique solution. Thus, instead of solving Poisson's equation, we solve a generalized Laplace's equation with Cauchy boundary conditions:

$$\nabla \cdot \sigma \nabla \Phi = 0 \tag{7}$$

with boundary conditions:

$$\Phi = \Phi_0 \quad \text{on } \Sigma \subseteq \Gamma_T \quad \text{and} \quad \sigma \nabla \Phi \cdot \mathbf{n} = 0 \quad \text{on } \Gamma_T. \tag{8}$$

While this version of the inverse problem has a unique solution [36], the problem is still mathematically ill-posed in the Hadamard sense; i.e., because the solution does not depend continuously on the data, small errors in the measurement of the voltages on the torso can yield unbounded errors in the solution.

An accurate solution to the inverse problem in electrocardiography would provide a noninvasive procedure for the evaluation for myocardial ischemia [37], the localization of ventricular arrhythmias and the site of accessory pathways in Wolff-Parkinson-White (WPW) syndrome, and, more generally, the determination of patterns of excitation and recovery of excitability.

5.2.1. Regularization of the ECG Inverse Problem

Traditional schemes for solving the inverse problem, (7), have involved reformulating the linear equation, $A\xi = b$, into $\xi_T = K\xi_E$, where ξ_T and ξ_E are the voltages on the body surface (torso) and heart's surface (epicardium) respectively, and K is the $T \times E$ transfer matrix of coefficients relating the measured torso voltages to the voltages measured on the heart. From this statement of the forward problem, the inverse problem can then be expressed as $\xi_E = K^{-1}\xi_T$

Unfortunately, because K is ill-conditioned, regularization techniques are necessary to restore continuity of the solution back onto the data. Briefly, one wishes to find an approximate vector, $\xi_{E\varepsilon}$, such that $\xi_{E\varepsilon} \to \xi_E$ as $\varepsilon \to 0$, where ε is the error between the approximate and true values. It is possible to reformulate the K matrix into an expression involving sub-matrices of K. We have found that it is then possible, and advantageous, to apply regularization not globally, but at the *local* level of the submatrices.

Researchers have long noticed that the discontinuities in the inverse solution appear irregularly distributed throughout the data. Tikhonov and other regularization schemes are, in effect, filters, which restore continuity by attenuating the high (spatial) frequency components of the solution. Since regularization is usually applied globally, the results can leave some regions overly damped or smoothed while others remain poorly constrained. Our idea is then to apply regularization only to sub-matrices that require it and to apply different amounts of regularization to different sub-matrices.

We begin by expressing $A\xi = b$ for the Cauchy problem in (7) as

$$\begin{pmatrix} A_{TT} & A_{TV} & A_{TE} \\ A_{VT} & A_{VV} & A_{VE} \\ A_{ET} & A_{EV} & A_{EE} \end{pmatrix} \begin{pmatrix} \xi_T \\ \xi_V \\ \xi_E \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}. \tag{9}$$

We can then rearrange the matrix to solve directly for the epicardial voltages in terms of the measured body surface voltages,

$$\xi_E = (A_{VE}^{-1} A_{VV} A_{TV}^{-1})(A_{TT} - A_{TV} A_{VV}^{-1} A_{VT})\xi_T. \tag{10}$$

where the subscripts T, V, and E stand for the nodes in the regions of the torso, the internal volume, and epicardium, respectively. Since torso and epicardial nodes are always separated by more than one element, the A_{TE} submatrix is zero.

Note that we now have inverses of three sub-matrices, A_{VE}, A_{TV}, and A_{VV}, as well as several other matrix operations to perform. If one estimates the condition number, κ, of the three sub-matrices using the ratio of the maximum to minimum singular values from a singular value decomposition (SVD), one finds that the condition number varies significantly (from $\kappa \approx 200$ for A_{VV} to $\kappa \approx 1 \times 10^{16}$ for A_{VE} in the two-dimensional model described below). Thus, one can regularize each of the sub-matrices differently, even leaving some of the other sub-matrices untouched. The overall effect of this *local regularization* is more control over the regularization process.

5.2.2. Inverse Problem Results

To study the direct and inverse problems in electrocardiography, we developed a series of two- and three-dimensional finite element models based upon magnetic resonance images from a human subject. Each of 116 MRI scans were segmented into contours defining torso, fat, muscle, lung, and heart regions. Additional node points were added to digitize each layer and pairs of layers were tessellated into tetrahedra using a Delaunay triangulation algorithm [12,16]. The resulting model of the human thorax contained approximately 675,000 volume elements, each with a corresponding conductivity tensor. For the initial studies on the effects of adaptive control of errors and local regularization, we utilized a two-dimensional finite element model extracted from our three-dimensional model of the human thorax. The two-dimensional model was constructed from a single transaxial magnetic resonance image located approximately 4 cm. above the apex of the heart.

As a test of the adaptive algorithm, a simulated source distribution was placed on the surface of the heart model. Using the procedure described previously, direct solutions were computed for different levels of mesh refinement and compared at the outer torso boundary. Figure 3 shows the voltage at the outer boundary versus distance around the two-dimensional contour. The original mesh contained approximately 1500 elements while the final mesh, after five iterations of the adaptive algorithm, contained approximately 7000 elements. The maximum estimated error in the calculated potential was over 30% greater in the original mesh compared to the final mesh and the maximum estimated error in the potential gradient was over 13% larger in the original mesh compared to the refined mesh. Increased accuracy does not come without a price and the h-adaption increases the number of degrees of freedom, and thus, the computational costs. However, since the relative error falls off rapidly with increasing degrees of freedom and then levels out after only a few iterations of the adaptive algorithm, it is not difficult to choose a reasonable balance between accuracy and computational cost.

As a test of the local regularization we applied epicardial potentials recorded during open chest surgery from a patient diagnosed with WPW syndrome as the Dirichlet boundary conditions of the two-dimensional model described above. The tissue conductivities were assigned as follows: fat = .045 S/m, fatpad = .045 S/m, lungs = .096 S/m, skeletal muscle (along the fiber) = .3 S/m, skeletal muscle (across the fiber) = .1 S/m, and an average thorax value = .24 S/m. Forward solutions calculated using the adapted mesh served as torso boundary conditions, $\Phi = \Phi_0$ on $\Sigma \subseteq \Gamma_T$ for the inverse solution.

For the sub-matrices A_{VE} and A_{TV} the *L-curve* algorithm determined the optimal *a priori* regularization parameter. The local Tikhonov regularization technique was then applied to the two sub-matrices and the inverse solution matrix computed. Because the size of the two-dimensional finite element model was relatively small, the calculations stayed in dynamic memory on an IBM RISC-6000 model 580. The resulting inverse solutions were then compared to those calculated using standard global Tikhonov techniques and the original measured epicardial potentials.

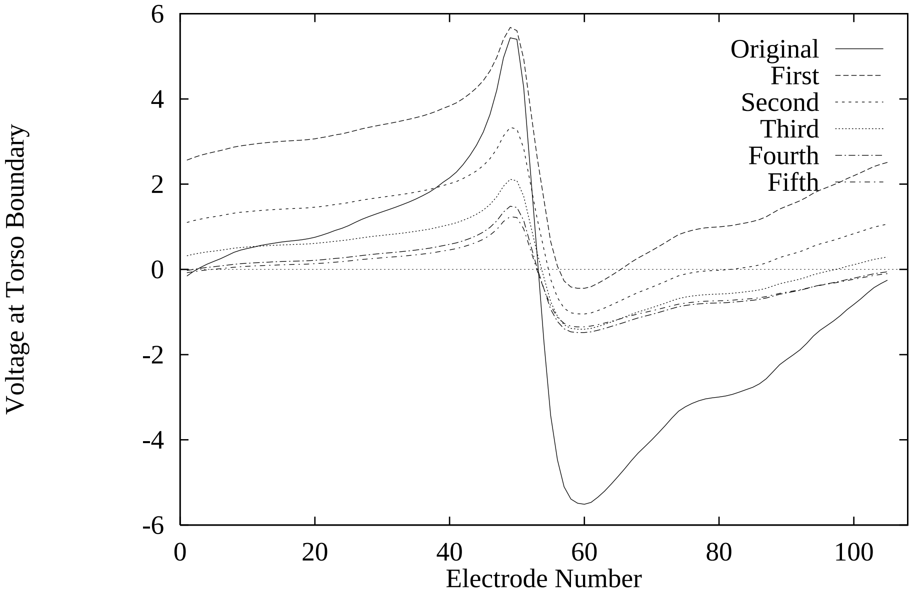

Figure 3: Effects of Automatic Mesh Refinement

The application of the local regularization technique recovered the voltages to within 12.6% RMS error. Previous studies in three-dimensioanl problems have reported the recovery of epicardial potentials with errors in the range of 20-40%, [38,39,40,41]. Figure 4 shows the inverse solution calculated using the local regularization technique compared with the recorded heart voltages as a function of position on the epicardium. The global solution tended to be smoother, not able to follow the extrema as well as the local solution could. The local solution also showed areas of local error which suggest that a different partitioning of the sub-matrices may provide even better accuracy.

In interpreting the results of the local regularization method, we must plead guilty to several *inverse crimes*. Namely, we have used the same finite element mesh to calculate solutions for both the forward and inverse problem, and we have not, at this point, added noise to the input torso boundary values. Currently, we are investigating the effects of added noise on a variety of experimental data as well as developing more general and computationally efficient ways to decompose the transfer matrix.

5.3. Computational Steering

The current computational engineering and science modeling process is a sequential one – create, or make modifications to, a geometric model, input initial conditions and/or boundary conditions, numerically approximate solutions to the

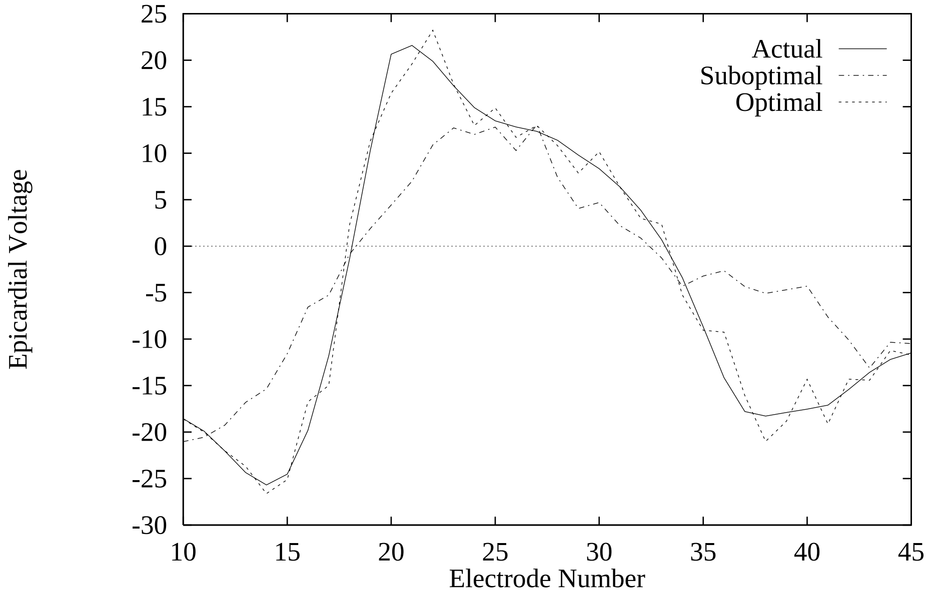

Figure 4: Local Regularization Technique

governing equation(s), store data off to disk and/or tape, visualize the results via a separate visualization package, decide upon appropriate changes, and repeat the whole process *ad nauseam.* Usually, the modifications made to the model, input parameters, numerical and computing methods, and visualization parameters are accomplished using separate in-line, command based instructions, and each change made in the model and/or input parameters causes all of the other steps to be repeated. Even for experienced scientists who may employ scripts and conversion programs to facilitate the task, the process is seldom streamlined. Ideally, scientists and engineers would be provided with a system in which all these computational components were linked, so that all aspects of the modeling and simulation process could be controlled interactively through a graphical interface 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 because it is often difficult and time consuming.

To reduce the time the scientist spends in using this standard modeling paradigm and to provide a tool for exploration of computational science and engineering problems, we have developed SCIRun[†42,43]. SCIRun is a framework in which large scale computer simulations can be composed, executed, controlled and tuned interactively. Composing the simulation is accomplished via a visual programming interface to a dataflow network. To execute the program, one specifies parameters with

† Pronounced "ski-run." SCIRun derives its name from the Scientific Computing and Imaging (SCI) research group at the University of Utah.

a graphical user interface rather than with the traditional text-based datafile. Controlling a simulation involves steering the simulation interactively as it progresses.

Computational steering has been defined as "the capacity to control the execution of long-running, resource-intensive programs" [44]. In the field of computational science, we apply this concept to link visualization with geometric design and computational phases to interactively explore (steer) a simulation in time and/or space. As knowledge is gained, it can be used to change the input conditions and/or other parameters of the simulation. Implementation of a computational steering framework requires a successful integration of the many aspects of scientific computing, including geometric modeling, numerical analysis, and scientific visualization, all of which need to be effectively coordinated within an efficient computing environment.

SCIRun is a framework in which large scale computer simulations can be composed, executed, controlled and tuned interactively. Composing the simulation is accomplished via a visual programming interface to a dataflow network. Software systems such as AVS from Application Visualization Systems Inc. [45], Iris Explorer from Silicon Graphics, and Visualization Data Explorer from IBM [46] have made this archetype popular for scientific visualization [47]. Our work has extended this paradigm into the realm of scientific computation. In SCIRun, the typical components of the computational paradigm – geometric modeling, numerical analysis, and scientific visualization – are integrated into a visual programming environment with the ability to interactively steer any one phase of the process and to see the effects propagate throughout the system automatically.

As a simple example we consider a group of visualization modules within a dataflow network, illustrated in Figure 5. The boxes represent computational algorithms (modules) and the lines represent data pipes between the modules. Scalar field data enters through the pipe at the top, passing into an isosurface extraction algorithm and a gradient field computation. Streamlines are computed on the resulting gradient vector field, and colored according to the scalar values. Geometry objects for the isosurfaces and streamlines are passed out the bottom.

When the user changes a parameter in any of the module user interfaces, the module is re-executed, and all changes are automatically propagated to all connecting modules. The user is freed from worrying about details of data dependencies and data file formats. Some changes can be made without stopping the computation, thus "steering" the computational process. Other changes cause the computations to be cancelled and automatically re-started, improving the efficiency of this "computational workbench."

6. Acknowledgements

This research was supported in part by awards from the Whitaker Foundation, NIH, NSF, Pacesetter Inc., the Nora Eccles Treadwell Foundation, and the Richard A. and Nora Eccles Harrison Fund for Cardiovascular Research. The authors would like to thank S. Parker, H.W. Shen and D. Weinstein for their valuable comments

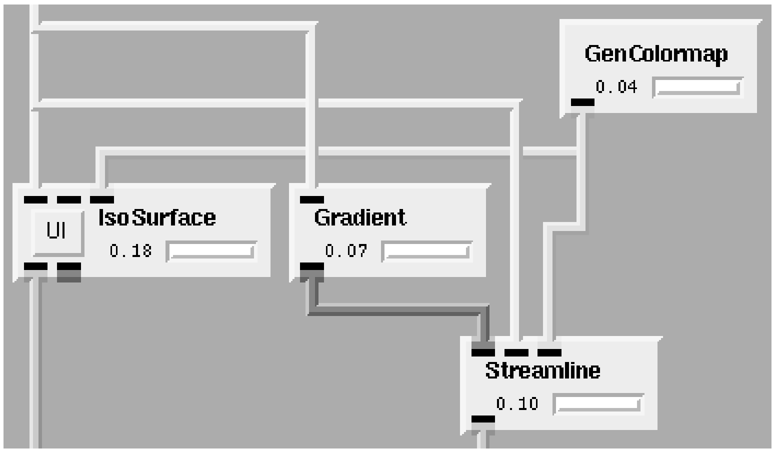

Figure 5: Dataflow Network for a Subset of Visualization Modules

and suggestions. Additional information regarding the research efforts of the Scientific Computing and Imaging group, including postscript versions of several papers, can be found via our web page, http://www.cs.utah.edu/˜sci.

7. References

1. C.E. Miller and C.S. Henriquez. Finite element analysis of bioelectric phenomena. *Crit. Rev. in Biomed. Eng.*, 18:181–205, 1990.
2. T.C. Pilkington, B. Loftis, J.F. Thompson, S.L-Y. Woo, T.C. Palmer, and T.F. Budinger. *High-Performance Computing in Biomedical Research.* CRC Press, Boca Raton, 1993.
3. H.W. Shen. Semi-automatic image segmentation: A bimodel thresholding approach. Technical report, UUCS-94-019, Dept. of CS, Univ. of Utah, 1994.
4. D. Weinstein and C.R. Johnson. Hierarchical data structures for interactive volume visualization. *IEEE Visualization '95*, 1995 (submitted).
5. J.A. Schmidt, C.R. Johnson, J.C. Eason, and R.S. MacLeod. Applications of automatic mesh generation and adaptive methods in computational medicine. In J.E. Flaherty and I. Babuska, editors, *Modeling, Mesh Generation, and Adaptive Methods for Partial Differential Equations.* Springer-Verlag, 1994 (to appear).
6. D.F. Watson. Computing the n-dimensional Delaunay tesselation with applications to Voronoi polytopes. *Computer Journal*, 24(2):167–172, 1981.
7. J. Thompson and N.P. Weatherill. Structed and unstructed grid generation. In T.C. Pilkington, B. Loftis, J.F. Thompson, S.L-Y. Woo, T.C. Palmer, and T.F. Budinger, editors, *High-Performance Computing in Biomedical Research*, pages 63–112. CRC Press, Boca Raton, 1993.
8. J.C. Russ. *The Image Processing Handbook.* CRC Press, Inc, 1992.

9. R.J. Schalkoff. *Digital Image Processing and Computer Vision.* John Wiley & Sons, Inc., 1989.
10. A. Rosenfeld and A.C. Kak. *Digital Picture Processing.* Academic Press, Inc., 1982.
11. C.R. Johnson. *Numerical Methods for Bioelectric Field Problems.* CRC Press, Inc., Boca Raton, FL, 1995 (to appear).
12. C.R. Johnson, R.S. MacLeod, and P.R. Ershler. A computer model for the study of electrical current flow in the human thorax. *Computers in Biology and Medicine*, 22(3):305–323, 1992.
13. C.R. Johnson, R.S. MacLeod, and M.A. Matheson. Computational medicine: Bioelectric field problems. *IEEE COMPUTER*, pages 59–67, Oct., 1993.
14. R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualization tools for computational electrocardiography. In *Visualization in Biomedical Computing*, pages 433–444, 1992.
15. R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualization of cardiac bioelectricity — a case study. In *IEEE Visualization '92*, pages 411–418, 1992.
16. C.R. Johnson, R.S. MacLeod, and M.A. Matheson. Computer simultions reveal complexity of electrical activity in the human thorax. *Comp. in Physics*, 6(3):230–237, May/June 1992.
17. C.R. Johnson and R.S. MacLeod. Nonuniform spatial mesh adaption using a posteriori error estimates: applications to forward and inverse problems. *Applied Numerical Mathematics*, 14:311–326, 1994.
18. C.R. Johnson and R.S. MacLeod. Inverse solutions for electric and potential field imaging. In R.L. Barbour and M.J. Carvlin, editors, *Physiological Imaging, Spectroscopy, and Early Dectection Diagnostic Methods*, volume 1887, pages 130–139. SPIE, 1993.
19. C. Johnson. *Numercial solution of Partial Differential Equations by the Finite Element Method.* Cambridge University Press, Cambridge, 1990.
20. P.G. Ciarlet and J.L Lions. *Handbook of Numerical Analysis: Finite Element Methods*, volume 1. North-Holland, Amsterdam, 1991.
21. D.S. Burnett. *Finite Element Method.* Addison Wesley, Reading, Mass., 1988.
22. J.E. Flaherty. *Adaptive Methods for Partial Differential Equations.* SIAM, Philadelphia, 1989.
23. O.C. Zienkiewicz. *The Finite Element Method in Engineering Science.* McGraw-Hill, New York, 1971.
24. R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualizing bioelectric fields. *IEEE Computer Graphics and Applications*, pages 10–12, 1993.
25. H.W. Shen and C.R. Johnson. Differential volume rendering: A fast alogrithm for scalar field animation. In *Visualization '94*, pages 180–187. IEEE Press, 1994.
26. R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualization tools for computational electrocardiography. In *Visualization in Biomedical Comput-*

ing, pages 433–444, Bellingham, Wash., 1992. Proc. SPIE 1808.
27. R.S. MacLeod, C.R. Johnson, and M.A. Matheson. Visualization of cardiac bioelectricity — a case study. In *Proc. IEEE Visualization 92*, pages 411–418. IEEE CS Press, 1992.
28. R.S. MacLeod, R.L. Lux, and B. Taccardi. Translation of body surface maps between different electrode configurations using a three-dimensional interpolation scheme. In P.W. MacFarlane, editor, *Electrocardiology '93: Proceedings of the International Congress on Electrocardiology, XXth Annual Meeting*, pages 179–182. World Scientific, Singapore, 1993.
29. P. Ghapure and C.R. Johnson. A 3d cellular automata model of the heart. In *Proceedings of 15th Annual IEEE Engineering in Medicine and Biology Society International Conference*. IEEE Press, 1993.
30. K.-L. Ma and K. Sikorski. A distributed algorithm for the three-dimensional compressible navier-stokes equations. *Transputer Research and Applications*, 4, 1990.
31. R Yagel and Z. Shi. Accelerating volume animation by space-leaping. In *Proceedings of Visualization '91*, pages 62–69. IEEE Computer Society Press, Los Alamitos, CA, October 1993.
32. R Yagel and A. Kaufman. Template-based volume viewing. In *Proceedings of EUROGRAPHICS '92*, pages 153–157. Blackwell, Cambridge, England, September 1992.
33. M.F. Cohen K.-L. Ma and J.S. Painter. Volume seeds: A volume exploration technique. *The Journal of Visualization and Computer Animation*, 2:135–140, 1991.
34. H.W. Shen and C.R. Johnson. Differential volume rendering: A fast volume visualization technique for flow animation. In *IEEE Visualization '94*, pages 180–187, 1994.
35. J. Schmidt. *Mesh generation with applications in computational electrophysiology*. PhD thesis, Duke University, Durham, NC, 1993.
36. Y. Yamashita. Theoretical studies on the inverse problem in electrocardiography and the uniqueness of the solution. *IEEE Trans Biomed Eng*, BME-29:719–725, 1982.
37. R.S. MacLeod, C.R. Johnson, M.J. Gardner, and B.M. Horáček. Localization of ischemia during coronary angioplasty using body surface potential mapping and an electrocardiographic inverse solution. In *Computers in Cardiology*, pages 251–254. IEEE Press, 1992.
38. F.P. Colli Franzone, G. Gassaniga, L. Guerri, B. Taccardi, and C. Viganotti. Accuracy evaluation in direct and inverse electrocardiology. In P.W. Macfarlane, editor, *Progress in Electrocardiography*, pages 83–87. Pitman Medical, 1979.
39. P. Colli Franzone, L. Guerri, S. Tentonia, C. Viganotti, S. Spaggiari, and B. Taccardi. A numerical procedure for solving the inverse problem of electrocardiography. Analysis of the time-space accuracy from *in vitro* experimental data. *Math Biosci*, 77:353, 1985.

40. B.J. Messinger-Rapport and Y. Rudy. Regularization of the inverse problem in electrocardiography: A model study. *Math Biosci*, 89:79–118, 1988.
41. P.C. Stanley, T.C. Pilkington, and M.N. Morrow. The effects of thoracic inhomogeneities on the relationship between epicardial and torso potentials. *IEEE Trans Biomed Eng*, BME-33:273–284, 1986.
42. C.R. Johnson and S.G. Parker. A computational steering model for problems in medicine. In *Supercomputing '94*, pages 540–549. IEEE Press, 1994.
43. S.G. Parker and C.R. Johnson. Scirun: A scientific programming environment for computational steering. In *Supercomputing '95*. IEEE Press, 1995. (submitted).
44. W. Gu, J. Vetter, and K. Schwan. An annotated bibliography of interactive program steering. *Georgia Institute of Technology Technical Report*, 1994.
45. C. Upson and et al. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics & Applications*, 9(4):30–42, July 1989.
46. B. Lucas and et al. An architecture for a scientific visualization system. In *Proceedings of Visualization '92*, pages 107–114. IEEE Press, 1992.
47. C. Willams, J. Rasure, and C. Hansen. The state of the art of visual languages for visualization. In *Proceedings of Visualization '92*, pages 202–209. IEEE Press, 1992.