SCIRun Forward/Inverse ECG Toolkit

SCIRun 4.6 Documentation

Center for Integrative Biomedical Computing
Scientific Computing & Imaging Institute
University of Utah

SCIRun software download:

http://software.sci.utah.edu

Center for Integrative Biomedical Computing:

http://www.sci.utah.edu/cibc

This project was supported by grants from the National Center for Research Resources **(5P41RR012553-14)** and the National Institute of General Medical Sciences **(8 P41 GM103545-14)** from the National Institutes of Health.

Author(s):
Jaume Coll-Font, Moritz Dannhauer, Michael Steffen, Darrell Swenson, Dafang Wang, Burak Erem, Jess Tate, Jeroen Stinstra, Dana Brooks and Rob MacLeod

Contents

Chapter 1

Overview

This guide describes and documents the SCIRun ECG Forward/Inverse Toolkit. This toolkit is a collection of modules and networks within the SCIRun system, which can be used to solve forward and inverse electrocardiography problems. The guide assumes that the reader has basic user-level familiarity with SCIRun; in particular that the reader is familiar with placing modules into a SCIRun network, connecting modules, and visualizing data. If the reader is not familiar with these operations, we suggest you first consult the SCIRun Basic Tutorial, also distributed in the SCIRun documentation.

The purpose of this toolkit is to advance the state of forward and inverse solutions in this field by offering a common platform for investigators and others to compare geometries and geometric representations, forward and inverse algorithms and data, with the maximum degree of both flexibility and commonality that we can achieve. Thus, we hope to increase the "reproducibility" of developments in this field and help move these technologies closer to useful clinical applications. As such, the software (like all of SCIRun) is open-source, modular, and extensible. We follow modern open-source software engineering practices to maximize portability, extensibility, and reliability of our software. As we explain below, SCIRun has a built-in facility for connecting with Matlab so that processing can be carried out jointly using both programs. In addition, the ECG Forward/Inverse Toolkit features capabilities which allow the user to interface with ECGSim (http://www.ecgsim.org), a popular software package for solving certain types of forward ECG problems with a high degree of interactive control over source models).

We welcome inquiries from users about this toolkit, and very much encourage contributions to the toolkit from the community. For more information about the toolkit or the underlying algorithms, or to discuss contributing to its development, please contact *scirun-users@sci.utah.edu*.

This toolkit is a product of the Center for Integrative Biomedical Computing (CIBC), an NIH supported Biomedical Technology Research Center, and we gratefully acknowledge the support from NIH that has made the development of this work possible. The CIBC is housed in the Scientific Computing and Imaging (SCI) Institute at the University of Utah and the environment and personnel in SCI have also been integral to the development of the toolkit.

1.1 Toolkit overview and capabilities

Computational modeling of bioelectric fields often requires the solution of electrocardiographic or electroencephalographic forward and inverse problems in order to non-invasively analyze electrophysiological events that are otherwise inaccessible or unethical to explore. All solutions to both problems require a common set of components, each of which is then customized or optimized for the particular problem formulation. Bioelectric activity begins from a biophysical source of voltage or current, which for computational purposes must be modeled in an appropriate, problem-specific form. For example, when the clinical mission is to identify focal activation in the brain, one or more current dipoles may be a suitable source model. However, when the goal is to reconstruct the sequence of activation across the

ventricles of the heart, the sources are better modeled in more complex form to capture the wavefront of this activation. Each type of source then requires an approximation in mathematical and then numerical form. Such a source model can then be used (in a solution to the *forward* problem) to computationally generate the associated voltages on the surface of the body (or wherever measurements might be made). If the goal is to recover sources from measured potentials, *i.e.* to solve the *inverse* problem, such as in the activation sequence reconstruction just mentioned, the solution strategy must include appropriate numerical techniques that can incorporate constraints and recover useful solutions, even when the inverse problem is poorly posed and hence the forward problem numerically ill-conditioned.

Creating complete software solutions to such problems from scratch is a daunting undertaking, requiring both considerable resources and considerable breadth of expertise. It is in order to make such tools more accessible to a broader array of researchers, and to enable validation and comparison across solution approaches, that the Center for Integrative Biomedical Computing (CIBC) has created this ECG Forward/Inverse toolkit. It is designed to provide a generalized software environment, based on the open-source SCIRun system, to aid researchers to construct, execute, visualize, and compare such computational models.

SCIRun is based on a data flow-visual programming paradigm, in which distinct computational (and software) modules are linked by connections ("data pipes") to form functional networks. The network editor offers the user interactive control and flexibility to assemble the specific network needed to solve the computational task at hand. The ECG Forward/Inverse toolkit provides a diverse array of modules, with associated data types, for bioelectric field problems, together with sample networks and data sets with which to explore its functionality. An additional key feature of using the SCIRun system is that rather than requiring custom code for all its functionality, the toolkit leverages SCIRun's ability to connect directly to Matlab through a bi-directional Matlab interface, and has capabilities to read in data created by ECGSim, as mentioned above.

Table 1.1 lists currently available forward and inverse approaches explicitly provided within the toolkit. The toolkit contains full simulation example networks that illustrate potential and activation-based forward models using both finite element (FEM) and boundary element (BEM) approximation techniques as well as selected potential and activation-based inverse methods. As indicated by the table, simulation specific tools, such as lead field and stiffness matrix generators, as well as a variety of regularization modules, are also included, and can easily be included in user-built networks.

Forward tools	Inverse Tools
Potential-Based FEM/BEM	Activation-Based[&*]
Activation-Based FEM[†]	Potential-Based Regularization Methods
FEM Lead Field Calculation	- Tikhonov
Stiffness Matrix Calculation	- Tikhonov SVD
	- Truncated SVD
	- Isotropy method[*]
	- Gauss-Newton Method[*]
	- Wavefront based potential reconstruction[*]
	- Total Variation

Table 1.1. Algorithms currently included in the CIBC ECG Forward/Inverse toolkit
[†]Activation-based BEM forward solution is currently unavailable in the toolkit
[&]Based on a Gauss-Newton optimization
[*]Requires Matlab

1.2 Software requirements

1.2.1 SCIRun Compatibility

The modules demonstrated in this tutorial are available in SCIRun version 4.5 and higher. This tutorial is not compatible with earlier versions of SCIRun. If you have an existing SCIRun installation, we strongly encourage you to update your SCIRun version to the latest build, available from the SCI software portal (http://software.sci.utah.edu), which will include the latest bug fixes and ensure that your version is up to date with the capabilities demonstrated in this guide.

1.2.2 Required Datasets

This tutorial relies on several datasets that are freely distributed as part of the SCIRunData bundle. To obtain these datasets, please go to the SCI software portal at http://software.sci.utah.edu and click on **Download SCIRun** to download the SCIRunData zip files. The SCIRunData dataset is provided in both zip and gzip file formats for your convenience.

Chapter 2

Mathematical Background

2.1 Overview

In this chapter we describe the basic mathematical formulations, and some solution approaches to the forward and inverse problems of electrocardiography. The goal of solutions to the forward problem is to predict potentials that could be measured in any accessible location (usually the surface of the torso) given a description of the cardiac electrical sources as well as the geometry and conductivities of the torso involved. The goal of solutions to the inverse problem is to predict cardiac sources given a set of measurements and the same geometry and conductivity information. It is important to note that a solution to the inverse problem presupposes an available solution to the forward problem. There exists a large literature on both of these problems as well as tutorial articles, and many excellent textbook and reference chapters. Here we briefly summarize the basic background to facilitate our description of the toolkit capabilities in subsequent chapters.

Both forward and inverse solutions require a specific model formulation of the cardiac electrical sources. This toolkit treats two different equivalent source models. (These two models are arguably the two dominant formulations in current forward and inverse problem research.) One model, which we will refer to as the "activation-based" source model, assumes that the dominant feature of cardiac electrical activity is the timing of the arrival of the depolarization wavefront (known as "activation times") at each location in the heart. (A similar problem, in which the equivalent sources called "recovery times" are the timing of repolarization at each location, is solved in a similar fashion.) Classical results have shown that, under assumptions of isotropy and homogeneity of the myocardium, activation-based models can be reduced to the activation times on the surface of the heart. In this context, this is usually taken as the epicardial (outer) and endocardial (inner) heart surfaces, connected across the base of the heart by an imaginary connecting surface. The source in this case can be modeled by a moving set of current dipoles aligned along the "activation wavefront"; that is, the curve where activation is taking place on this surface at any given time.

The second source model treated here, which we will refer to as the "potential-based" source model, assumes that the cardiac sources can be represented by the time-varying electrical potentials present on a surface, enclosing all the electrical sources. Gauss' Law implies that any such set of potentials is unique, and that the closer the surface is to the myocardial surface the more useful the model is. So, the surface is typically taken as the

epicardium, closed off by an imaginary "top" surface at the base of the heart or, alternatively, the same joint epicardial/endocardial surface used for activation-based models.

In the rest of this chapter we describe solutions to the forward and inverse problem concentrating on the specific tools currently provided in this toolkit. Again we refer the reader to the literature for more complete background.

2.2 Solutions to the Forward Problem in the Forward/Inverse Toolkit

The temporal frequencies which are relevant to electrocardiographic biolelectricity are relatively low, and the wavelengths many orders of magnitude larger than the dimensions of the human body. So, from a biolelectricity viewpoint, the governing partial differential equation (PDE) is Laplace's equation:

$$\nabla \cdot (\boldsymbol{\sigma} \nabla \Phi) = 0, \tag{2.1}$$

where $\boldsymbol{\sigma}$ contain the relevant conductivities and Φ the electrical potentials. The boundary conditions are given by:

$$\Phi(x, y, z)|_{\Omega_k} = V_k \tag{2.2}$$

$$\left.\frac{\partial \Phi}{\partial n}\right|_{\Omega} = 0 \tag{2.3}$$

where Ω_k is the surface on which the sources are located.

As an alternative to solving this problem directly, a (weighted) integral can be taken over the solution domain on both sides of Eq. (2.1) and the resulting integral equation solved for Φ at locations of interest. This is usually referred to as the "weak form" of the PDE solution, and (aside from discontinuities which theoretically could be possible in the weak form solution but are not of practical importance here) is equivalent to solving the original, or "strong" form.

In a tractable geometry, such as a set of concentric or even eccentric spheres, this PDE can be solved via analytical expansions. However in complex geometries such as realistic torso models, numerical solutions must be applied. Again there is a large literature on such numerical methods. Two of these methods have predominated in the literature for forward electrocardiography: the Finite Element Method (FEM) and the Boundary Element Method (BEM). It is these two methods which have been implemented in this toolkit. Thus, in the rest of this subsection, we give a very brief description of these two methods.

The major difference between FEM and BEM, from the practical application point of view, is in the way in which they discretize the solution domain. FEM relies on a volume discretization. The geometry of the solution domain is describe by a mesh of small three dimensional volume elements, each with its own conductivity parameters. BEM, on the other hand, is a surface discretization method, with the geometry represented as a collection of bounding surfaces separating regions in the volume with different conductivities. Each of these surfaces is then discretized into a mesh. In other words, in both FEM and BEM there exists a collection of points, called nodes, which define the respective volume or surface elements. The potential Φ (and the current, $\boldsymbol{\sigma} \nabla \Phi$), is approximated by interpolation of potential and current across those elements, based on its value at the nodes, using known (usually polynomial) interpolation functions. Thus, numerical integration can be applied

to the weak form, and the node values come out of the integrals, leaving subintegrals over known functions which result in a set of weights. The result in either case is a system of linear equations.

The boundary conditions in Eq. (2.3) are applied differently in FEM and BEM, another important difference between the two methods.

We note that either BEM or FEM can be applied to both activation-based and potential-based source models, although there are important implementation details. We briefly note these differences here, and some specifics of the applications in the context of the relevant SCIRun modules will be presented in Ch. 3.

2.2.1 FEM in the Forward/Inverse Toolkit

The finite element method begins by subdividing the geometry into a set of volume elements with vertices at a set of nodes, and then approximating the potential in the volume by a basis expansion:

$$\bar{\Phi}(x, y, z) = \sum_i \Phi_i N_i(x, y, z), \tag{2.4}$$

where $\{N_i\}$ are a set of basis functions, one for each node in the volume element discretization, and $\{\Phi_i\}$ are the corresponding (unknown) coefficients at those nodes. Note that if the $\{\Phi_i\}$ can be determined, then the potential everywhere in the volume can be approximated via the basis expansion in Eq. (2.4). Usually, the basis functions are (Cartesian product) low-order polynomials, most commonly tri-linear functions, designed so that each function is 1 at its "own" node and decays to 0 at the other nodes of all elements which share that node (in which case Φ_i becomes a direct approximation of the potential at node i).

The Galerkin method is applied to solve this Laplace equation. In particular, the basis expansion in Eq. (2.4) is substituted into Eq. (2.1), both sides of the equation are multiplied by a set of "trial" or "test" functions (typically taken to be the same family of functions as the basis functions $\{N_i\}$), and then the equation is integrated over the solution domain, resulting in a weak form of the PDE.

Manipulation of the resulting integral equations yields, in the case where the test and basis function sets are the same,

$$\sum_i \Phi_i \int_{\Omega - \bar{\Omega} - \bar{\Omega}_k} \boldsymbol{\sigma} \nabla N_i \nabla N_j \, d\mathbf{V} = 0. \tag{2.5}$$

This can be rewritten as the matrix vector equation:

$$\mathbf{K}\boldsymbol{\Phi} = 0 \tag{2.6}$$

where $\mathbf{K}_{ij} = \int \boldsymbol{\sigma} \nabla N_i \nabla N_j \, d\mathbf{V}$ is the stiffness matrix, and $\boldsymbol{\Phi} = [\Phi_1, \ldots, \Phi_n]^T$ is the vector of unknown coefficients. The critical point is that the coefficients in $\mathbf{K}$ depend only on the geometry and the choice of basis and test functions, and thus can be computed ahead of time. We note that this equation is clearly singular, and an additional condition must be imposed (biophysically equivalent to taking some potential value as a reference) to reduce the number of degrees of freedom by one.

Once the equations are written, the boundary conditions must be imposed. In the case of bioelectric field problems such as forward electrocardiography, this reduces to replacing

the 0's on the right hand side by known currents or fixing some values of the vector $\mathbf{\Phi}$ to correspond to known voltages. There are a variety of ways to accomplish this to preserve certain numerical properties of the matrix equation, and again we refer the reader to the vast literature on this subject for details. If the measurement electrodes are treated as being larger than one node in size, this can lead to additional boundary conditions which in turn leads to additional modifications of the equations, and again we refer the reader to the literature.

The result is a matrix equation whose size is the total number of nodes in the volume, which tends to result in a relatively large system. However, as long as the basis and test functions have local support (for example, with linear basis functions, the support is restricted to all first-order neighboring nodes of the given node), most of the integrals defining the elements of $\mathbf{K}$ will involve non-overlapping functions and thus be equal to 0. Therefore, $\mathbf{K}$ will be a very sparse matrix with strong structure, leading to the possibility of both efficient storage and efficient solution by iterative solvers.

We note that every time a different set of source currents (activation-based model) or potentials (potential-based model) is applied on the heart surface, the modifications of the stiffness matrix are different, and thus the system of equations must be solved *de novo*. However, there exist certain problems where only the values of the potential at a subset of nodes is needed; In our applications of forward electrocardiography, we are generally only interested in the solution on the measurement surface, *e.g.*, on the body surface. Additionally, these applications often require multiple solutions performed for the given geometry, *e.g.*, when a time series of body surface potentials needs to be generated from a time series of sources. In this case, it can be useful to extract a "transfer matrix" from Eq. (2.6), which directly relates the known sources to the unknown and desired measurement potentials. Once this is done, solving the forward problem reduces to matrix-vector multiplication rather than the solution of a linear system.

There are several approaches to this problem. In this toolkit we have provided an example SCIRun network to implement one of them, the so-called "lead field" method, which solves the matrix equation repeatedly for source vectors consisting of a 1 at each source node in turn and 0 at all other source nodes. From this collection of solutions we can obtain the desired transfer matrix, which we will denote a $\mathbf{A}$. This network is described below in Ch. 3 and once again the details are available to the interested reader in the literature.

2.2.2 BEM in the Forward/Inverse Toolkit

Expanding on the initial description above, the boundary element method starts with the assumption that the domain can be divided into a (relatively) small number of (relatively) large subdomains in which the conductivities are isotropic (scalar) and constant. In addition there are other conditions on the subdomains, principally that they be bounded by closed surfaces. They can be simply nested or can have a more complex arrangement. Given that assumption, the surfaces of those subdomains become a sufficient domain upon which to solve the problem for the entire domain.

Briefly (and once again we refer the reader to the literature for the details and complications), one of the Green's Theorems from vector calculus is applied to an integrated form of Laplace's equation to transform the differential problem into a Fredholm integral problem. The surfaces are each subdivided (tessellated) into a collection of small surface (or boundary)

elements. Then (two-dimensional) basis functions (again usually low-order polynomials) are used to approximate the quantities of interest between the nodes of the resulting surface meshes. Given this discretization, after manipulation of the resulting integral equation, the integrals required can be computed through a series of numerical integrations over the mesh elements. In the BEM method, these integrals involve as unknowns the potential and its gradient. The integration involves the computation of the distance between each node within the surface and to all others surfaces. In complicated geometries, and in all cases when the node is integrated against the points on its "own" surface, there are numerical difficulties computing these integrals. In those cases there are a number of sophisticated solutions which have been proposed in the literature (and some of them are adopted in the SCIRun implementation). The result of all these integrals is a transfer matrix, which again we will denote $\mathbf{A}$, relating the source potentials or currents to the unknown measurement potentials. In the BEM case, because of the all-to-all nature of the integrations required, this matrix will be dense, not sparse. On the other hand, the size of the equation will be directly determined by the number of measurements and sources rather than the number of nodes in the entire domain. (We note that there is an alternative formulation of the BEM method which retains the potentials at all nodes on all surfaces, and which can be reduced to the transfer matrix described here, but we omit the details as usual.)

2.3 Solutions to the Inverse Problem in the Forward/Inverse Toolkit

To describe the solution to the inverse problem in a manner useful for this toolkit, we start with two different equations, depending on whether the activation-based or potential-based source models were used. Both approaches assume the availability of a forward transfer matrix $\mathbf{A}$, calculated by any appropriate method, including either FEM or BEM.

In the activation-based case, the source model is that the unknowns are an activation surface. (That is, activation times as a function of position on the heart surface, which we denote as $\tau(x)$, where x indicates position on the heart surface.) The assumptions required for the activation-based model imply that the temporal waveform of the potential (and current) at each source node has a fixed form, the same at all locations on the surface. This is assumed to be either a step function or a smoothed version of a step function (using piecewise polynomials or inverse trigonometric functions). We denote this function as $u(t)$. Thus the relevant forward equation can be written as

$$y(p,t) = \int_x \mathbf{A}_{p,x} u(t - \tau(x))\, dx \tag{2.7}$$

where the integral is over the heart surface, $\mathbf{A}_{p,x}$ is the element of $\mathbf{A}$ relating source node x to measurement node p, and $y(p,t)$ is the measurement surface potential at any time t and at a position p on the body surface. One advantage of the activation-based formulation is that the number of unknowns over an entire cardiac cycle is the number of solution nodes taken on the heart surface. On the other hand, as can be seen in Eq. (2.7), the forward equation is non-linear in the unknown activation times.

In the potential-based case, the forward matrix can be applied in a more straightforward manner. If we collect all measurements at a given time t into a vector $\mathbf{y(t)}$ and the potential

at all desired heart surface locations into a vector $\mathbf{x}(\mathbf{t})$, then we have

$$\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t). \tag{2.8}$$

The resulting equations over a time series can be collected into a matrix-matrix equation (with columns indexing time samples) or a single block matrix equation with a block diagonal matrix which has $\mathbf{A}$ repeated along the diagonals. The number of unknowns for the potential-based inverse problem is then the product of the number of surface nodes and the number of time samples. The time waveforms are left unconstrained, but the equations remain linear in the unknowns.

Both approaches result in ill-conditioned systems of equations, a direct result of the fact that the inverse problem itself is intrinsically ill-posed. Thus effective numerical solutions need to impose additional *a priori* constraints to achieve useful solutions, usually through a technique known as "regularization." Much of the research on the inverse problem over the last 30+ years has concerned methods of regularizing this problem.

2.3.1 Activation-based Inverse Solutions in the Forward/Inverse Toolkit

Since this problem is non-linear, iterative solutions are employed. An "initial guess", or starting point, is required. Then solutions are iteratively re-computed until a desired convergence criterion is met. Currently in the toolkit, we have a Matlab version of a Gauss-Newton iterative solver which can be called from within SCIRun. The starting solution must be supplied by the user. (We refer the reader to Ch. 4 for details.) Regularization is done using a constraint on the ℓ_2 norm of the Laplacian of the solution. See the next subsection for an explanation of Tihkhonov regularization.

2.3.2 Potential-based Inverse Solutions in the Forward/Inverse Toolkit

To combat the ill-posedness of the (linear) potential-based inverse problem, some form of what is known as "regularization" is typically applied. A number of such solution methods are available through the toolkit, either native in SCIRun or through the Matlab interface. Here we describe the basic formulation behind these approaches, while usage details are in Ch. 4.

Standard Tikhonov regularization

The Tikhonov regularization minimizes $\|Ax - y\|$ in order to solve $Ax = y$, where x is the unknown solution and y the given measurement. The forward solution matrix $\mathbf{A}$ is of size $m \times n$, where m is the number of measurement channels and n is the number of source reconstruction points). The system Ax can be overdetermined ($m > n$), underdetermined ($m < n$) or $m = n$. A is often ill-conditioned or singular, so it needs to be regularized. The Tikhonov regularization is often used to overcome those problems by introducing a minimum solution norm constraint, such as $\lambda\|Lx\|_2^2$. The solution of the problem can be found by minimizing the following function:

$$f(x) = \|P(y - Ax)\|_2^2 + \lambda^2\|Lx\|_2^2, \tag{2.9}$$

where λ is the regularization parameter, which is a user defined scalar value. The matrix $\mathbf{P}$ represents the *a priori* knowledge of the measurements. The matrix $\mathbf{L}$ describes the property of the solution x to be constrained. Conceptually, λ trades off between the misfit between predicted and measured data (the first term in the equation) and the *a priori* constraint. An approximate solution $\hat{x}$ of 2.9 is given for the overdetermined case ($n > m$) as follows:

$$\begin{aligned} \|P^{-1}(y - Ax)\|_2^2 + \lambda^2 \|Lx\|_2^2 &= f(x) \\ (A^T P^T P A + \lambda^2 (L^T L)^{-1})\hat{x} &= A^T P^T P y \end{aligned} \tag{2.10}$$

$$\tag{2.11}$$

and for the underdetermined case ($n < m$) as:

$$\begin{aligned} \|C^{-1}(y - Ax)\|_2^2 + \lambda^2 \|Wx\|_2^2 &= f(x) \\ WW^T A^T (AWW^T A^T + \lambda^2 (CC^T)^{-1})^{-1} y &= \hat{x} \end{aligned} \tag{2.12}$$

$$\tag{2.13}$$

with source space weighting $W, L \in n \times n$ and sensor space weighting $C, P \in m \times m$.

When $n = m$, either of the above formulations can be used. Both solutions are equal but the computational effort may differ. Furthermore, both representations can be solved numerically by a direct method (*e.g.* Gaussian elimination) or an iterative method (*e.g.* conjugate gradients). Both methods are implemented in SCIRun.

An alternative approach to solve the Tikhonov minimization involves a singular value decomposition (SVD) of the matrix $\mathbf{A}$. The SVD results in several factor matrices, from which one then obtains the inverse solution. The SVD approach is also implemented in SCIRun and will be described below.

Choosing the regularization parameter:

Choosing an appropriate value as a regularization parameter is critical for achieving useful solutions. The solution typically depends on it in a sensitive fashion, and good choices vary with the size, smoothness, etc. of the problem/solution as well as on the choice of regularization constraint. One commonly-used method to choose the regularization parameter automatically is to run the inverse solution for a large variety of parameters. For all regularization parameters, the residual norm (fit of the data, $\|y - A\hat{x}\|_2^2$) and the weighted solution norm (solution properties, $\|L\hat{x}\|_2^2$) is plotted in a log-log plot. Often, the resulting curve shape looks similar to the letter "L". The corner of the "L" represents a good trade off between both constraints. The automatic parameter selection used in this L-curve approach can be used for underdetermined and the overdetermined case as in option in the module `SolveInverseProblemWithTikhonov`.

Truncated SVD Tikhonov regularization

Another approach to regularization is to approximate $\mathbf{A}$ with a low-rank substitute by truncating the low-order modes of its SVD. The choice of rank is equivalent to the choice of regularization parameter.

Isotropy Method

This inverse method was introduced in the literature by Huiskamp and Greensite. It attempts to decorrelate the time series of the source waveforms prior to applying spatial regularization. However, since the temporal correlation of the source potentials is unknown, the isotropy method approximates this by the temporal correlation of the measurements. Under certain conditions (termed "isotropy" by the authors of this method, but also known as "separability" in the random field literature) this produces an equivalent decorrelating basis. Once this decorrelation is achieved, the resulting set of equations is truncated and then standard Tikhonov regularization is applied to each remaining system. After solving these systems, the decorrelation is reversed to restore the temporal correlation which had been estimated and removed previously.

Total Variation Method

The effect of the ℓ_2-norm regularization in Eq. (2.9) is typically to produce smooth solutions, where the smoothness is taken as a worthwhile cost to increase reliability in the face of ill-conditioning. However, cardiac wavefronts are relatively sharp (non-smooth) in space, and thus Tikhonov methods typically produce overly smooth wavefronts. One approach to address this problem is to replace the typical regularization constraints with a "total variation" (TV) constraint, which is to constrain the ℓ_1 norm of the gradient of the solution. Minimization of ℓ_1 norms favors a small number of relatively large values in the solution compared to ℓ_2 norm minimization, so using TV constraints will favor a sparse set of rapid spatial changes, which may allow reconstruction of wavefronts more accurately.

Specifically, total variation regularization can be formulated as follows:

$$\hat{\mathbf{x}} = \text{argmin } \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda TV(\mathbf{x}) \tag{2.14}$$

$$TV(\mathbf{u}) = \|\mathbf{L}\mathbf{u}\| = \sum_i |\mathbf{L}\mathbf{u}|_i \tag{2.15}$$

where $\mathbf{L}$ is a matrix approximation of the spatial gradient and the last sum is over the elements of the matrix-vector product.

To solve this minimization problem, Eq. (2.14) is differentiated with respect to $\mathbf{x}$ and the gradient is set to zero. Because the total variation term, $TV(u)$, is not differentiable at zero, a positive constant β is added:

$$TV(\mathbf{u}) = \sum_i \sqrt{(|\mathbf{L}\mathbf{u}|_i)^2 + \beta^2} \tag{2.16}$$

This results in the need to solve

$$(\mathbf{A}^T\mathbf{A} + \lambda \mathbf{L}^T \mathbf{W}_\beta(\mathbf{x})\mathbf{L})\mathbf{x} = \mathbf{A}^T\mathbf{y} \tag{2.17}$$

where the diagonal weight matrix $\mathbf{W}_\beta$ is defined as:

$$\mathbf{W}_\beta(u) = \frac{1}{2}\text{diag}\left[1/\sqrt{|[\mathbf{L}\mathbf{x}]_i|^2 + \beta^2}\right] \tag{2.18}$$

It can be seen that the weight matrix $\mathbf{W}$ depends on the local derivative. When the local derivative is small, the weight becomes a large value, imposing greater smoothness on

the solution. When the local derive is large, the weight becomes a small value, making the solution less constrained.

The total variation method is non-linear because the weight matrix $\mathbf{W}$ relies on the solution. We solve this iteratively, with an all-zero initial guess.

The total variation regularization contains two parameters to be tuned: β and λ. β controls the smoothness of the computed solution. A small β allows sharp gradients in the solution. λ controls the amount of regularization.

Wavefront-based potential reconstruction (WBPR) method

The WBPR method, like TV, also attempts to impose a constraint which encourages the presence of wavefronts in the solution. Simply put, the idea is to locate the wavefront by an appropriate "jump-detection" algorithm at each time instant. Then the solution is approximated with one that has three regions: a non-activated region (which is flat), an activated region (which is also flat but at a different potential), and a "wavefront" region, which is described in two dimensions by a smooth but sharp (across the wavefront) spatial function such as those used (in one dimension) for the time waveforms in activation-based methods. This three-region surface is then used as a constraint in an otherwise typical Tikhonov solution. The solution can be iterated forward or backward in time. The size of the potential jump across the wavefront must be known. Also any drift in the potential of the regions far from the activation wavefront, typically caused by drift in the reference potential used in the measurements, must be identified and eliminated. This method has shown considerable early promise but is included here as a "research" approach and should be treated as such.

Chapter 3

Forward Solutions

3.1 Overview

As described previously, the solution to the forward problem deals with the calculation of the projection of properties from a source, through some medium, to a set of collection locations. In this application, we will demonstrate that the forward problem specifically means the calculation of torso surface potentials from known cardiac source parameters. There are two aspects of the forward problem that one must consider in order to solve it: how to model the source and how that source maps to the torso surface. If presented as the classic $y = A(x)$, with y as the torso surface and x as the cardiac source, choosing a model that is represented by x is the source formulation. A is then dependent on the source model used, which must represent all relevant information about the torso geometry.

In regard to the source formulation, the two most common models are cardiac potentials and activation times. The use of cardiac potentials is a more intuitive formulation in that the cardiac cells interact to change the extracellular potentials of the myocardium. These potentials then project to the torso surface. The activation-time-based source formulation deals with the fact that when there is inactive and active tissue next to each other, a source much like a dipole or layer of dipoles is generated, which then projects to the surface. Both of these source models contain several assumptions, but each provides a generally accurate model that can be computationally reasonable.

With the source formulation considered, one must now determine the pattern with which the source projects to the torso surface. As mentioned, this relationship is dependent on the source model as well as the various properties of the torso. But in both source models presented, the function A can be represented by an $N \times M$ matrix. N is the number of points on the cardiac surface and M the number of points on the torso surface, which is called the lead field matrix. Calculating this matrix can be a very computationally intensive task, and there are varying ways to do this. The two examples given here find the lead field, or transfer matrix, are based on finite element method (FEM) and boundary element method (BEM). The theory and mathematics behind these two methods are presented in Ch. 2.

We provide in this toolkit three methods to calculate the forward problem. Potential based models using both FEM and BEM are demonstrated, as well as an activation-time based model using FEM. The BEM activation-time forward problem is not yet supported in SCIRun, but it is performed in ECGSim, the interactive simulation software developed

by Peacs in the Netherlands.[1].

An equivalent dipole layer is approximated as the source in the activation based forward problem.

3.2 Module Descriptions for Boundary Element Solutions

The boundary element solution utilizes many common modules within the SCIRun framework to read in files, visualize data, and manipulate geometry. Descriptions of these modules can be found in the SCIRun documentation, whereas the modules that are relatively unique to the boundary element solution are outlined below.

The first module to be introduced is `SetFieldProperties`. This module is thought to generate and modify properties of an input field, such as the potentials and conductivities of the epicardial surface. Figure 3.2 shows an example of how epicardial and torso surfaces should be defined. Since the heart surface is associated with epicardial potentials, it must have conductivity set to 0. On the other hand, the torso and the lungs are measurement surfaces and should have their respective conductivity value.
With the `SetFieldProperties` GUI, the user can define a set of desired properties of the input field and the corresponding values.

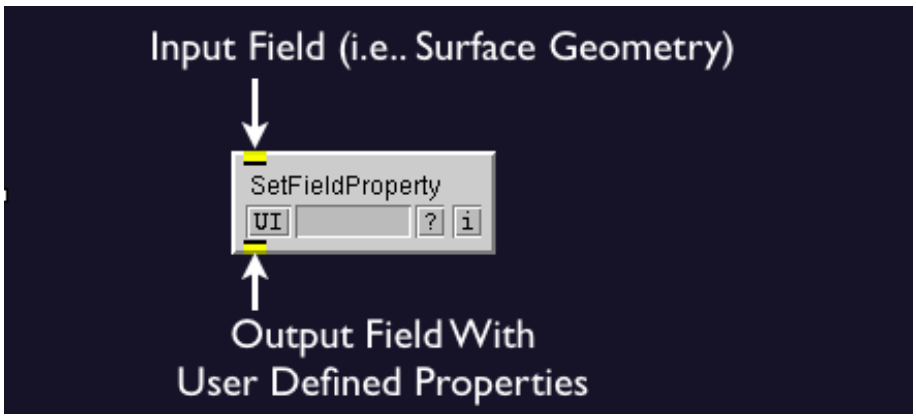

Figure 3.1. Module to set properties such as conductivity and if it is a source or boundary surface for the boundary element method.

[1]To obtain this free software, visit http://www.ecgsim.org/. Any references to ECGSIM herein are referring to version 1.3 beta

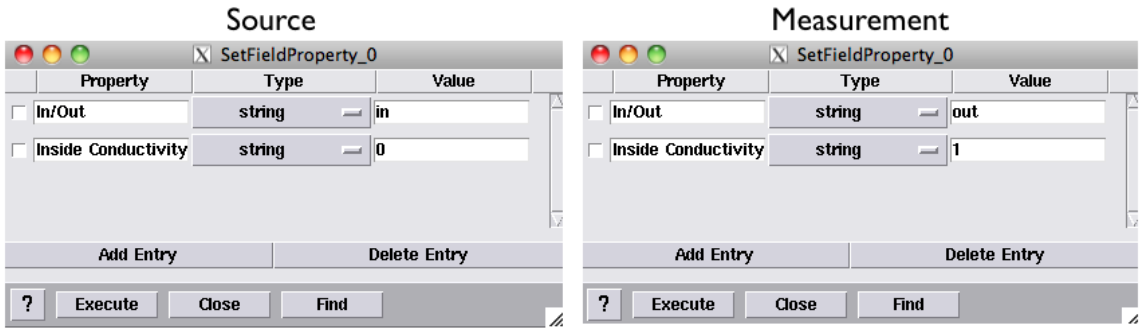

Figure 3.2. Shows the properties and values needed to create a source surface, left, or a measurement surface, right.

The forward matrix can then be generated by joining all the different surfaces in `BuildBEMatrix` module. This module will take all the surfaces inputed in the in fields and generate a transfer matrix from the surfaces with the In/Out field set to the last of the inputed surfaces. Note that the module dynamically adds as many inputs as necessary.

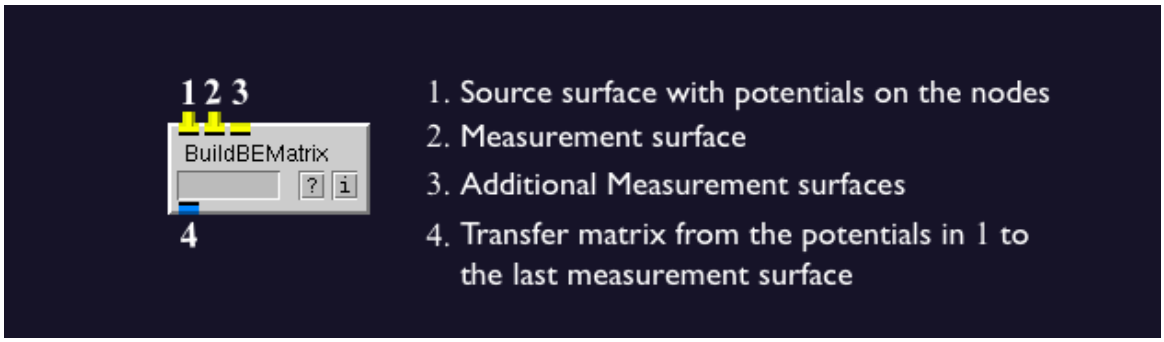

Figure 3.3. Shows the module that computes the transfer matrix between the source surface and the outermost measurement surface.

`BuildBEMatrix` assumes that, for any closed surface, the surface normals will be pointing outward. The surface normal for each element is determined using the node order from the connectivity definition of the mesh. The normals are defined using the right hand rule (counterclockwise) of the node order in each element. The module `FlipSurfaceNormals` can be used to flip all the element normals in the surface. It is important to check the normals of the surfaces being used because incorrect normals will produce erroneous answers.

3.3 Module Descriptions for Finite Element Solutions

The two most important modules for the forward finite element solution are the `BuildFEMatrix` and the `AddKnownsToLinearSystem` modules. These allow you to compute a stiffness matrix and add boundary conditions. The module `BuildFEMatrix` inputs a finite element

mesh with the conductivities set on each element, or a lookup table may be used for the conductivities. The result is a stiffness matrix based on the Galerkin method.

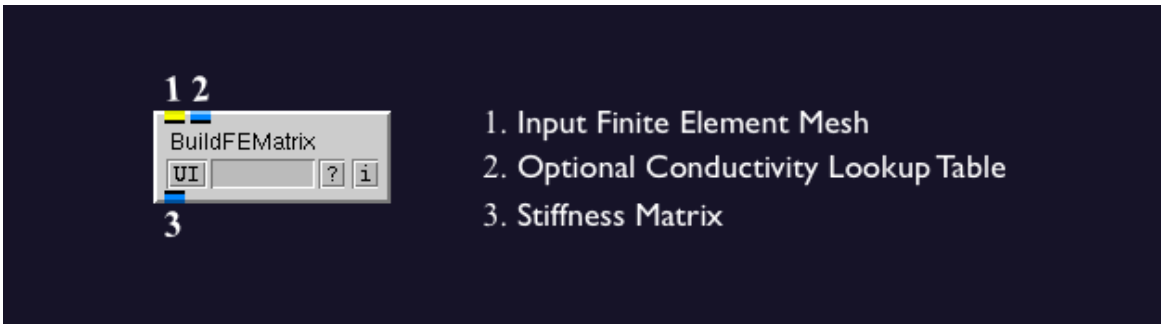

Figure 3.4. Shows the module that computes the stiffness matrix for a FE solution.

`AddKnownsToLinearSystem` makes it possible to add known values as boundary conditions to the linear system $Ax = b$, where $\mathbf{A}$ is the stiffness matrix, $\mathbf{b}$ is the right hand side vector, and $\mathbf{x}$ is known in the forward problem and unknown in the inverse problem. The module must have stiffness matrix along with an $\mathbf{x}$ matrix. If no right-hand-side matrix is provided, then it is assumed to be all zeros. Know parameters are input along with the unknown values, while the unknown values are set to 'nan'.

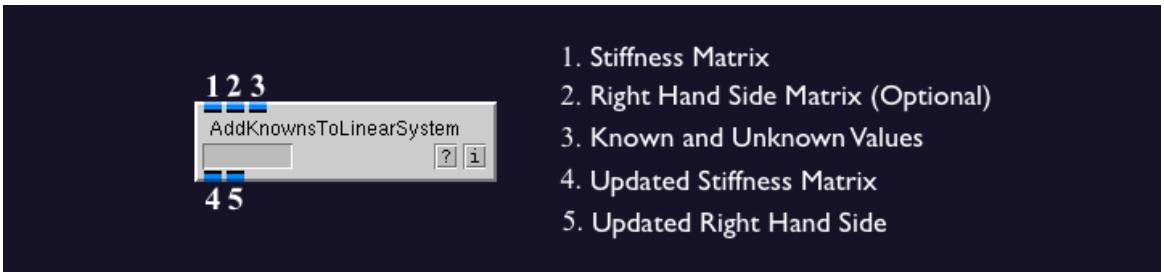

Figure 3.5. Shows the module that computes the stiffness matrix for a FE solution.

3.4 Example Networks for Boundary Element Solutions

The following network shows the most basic implementation of the boundary element method along with visualizing the results. This network reads in an epicardial surface mesh that has an associated matrix of epicardial potentials at each node. The conductivities of the torso and heart are set, along with the differentiation between source and measurement surfaces in the `SetFieldProperties` module. Next, a boundary element transfer matrix is computed and multiplied with the epicardial potentials. This results in a torso potential field that can be visualized with SCIRun's `ShowField` module.

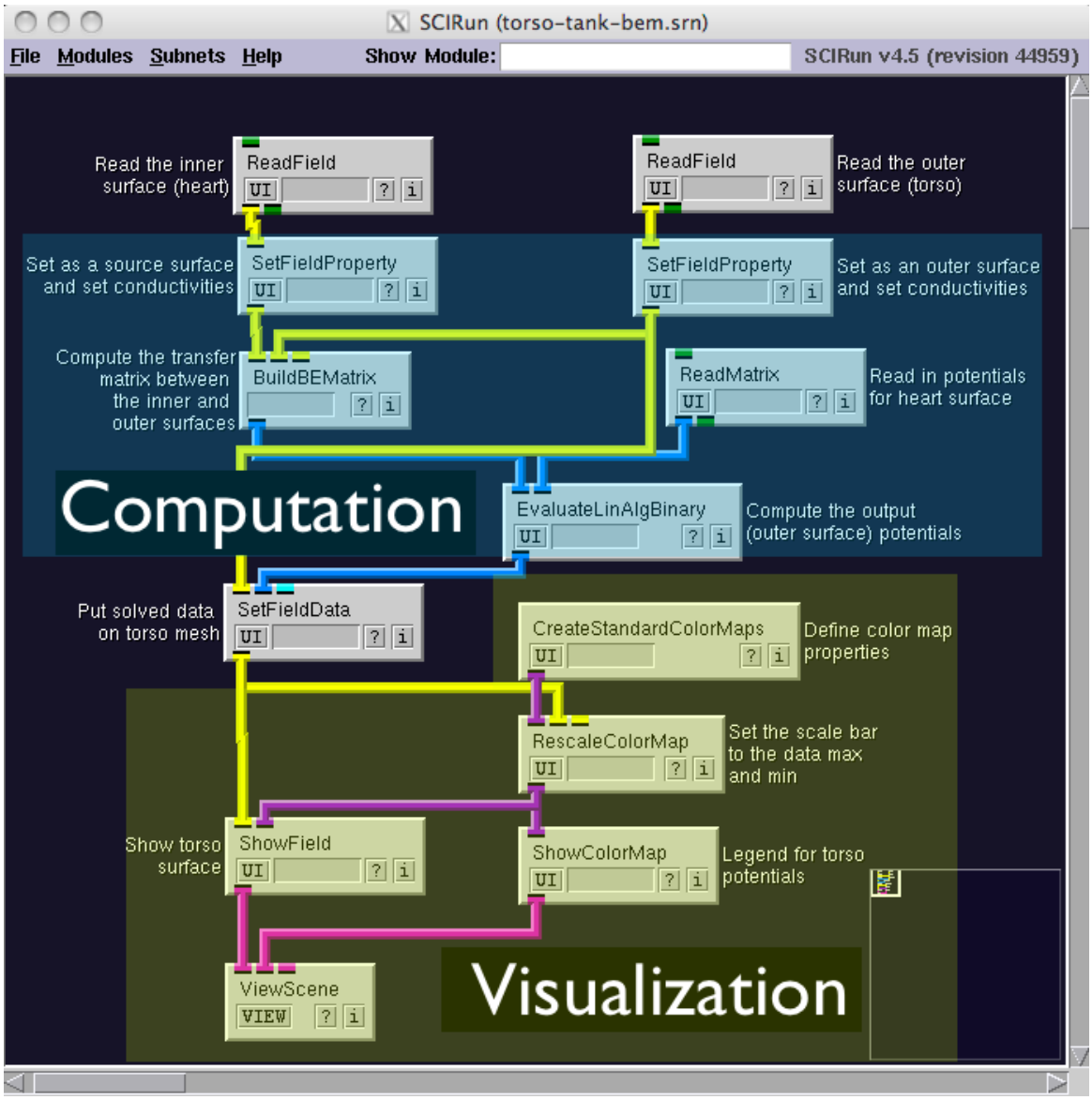

Figure 3.6. Shows the implementation of the boundary element method.

A similar SCIRun network for this example can be found at:
`src/nets/FwdInvToolbox/potential-based-bem/torso-tank-bem.srn`
in the SCIRun source code directory.

3.5 Example Networks for Finite Element Solutions

3.5.1 Potential Based Forward FEM simulation

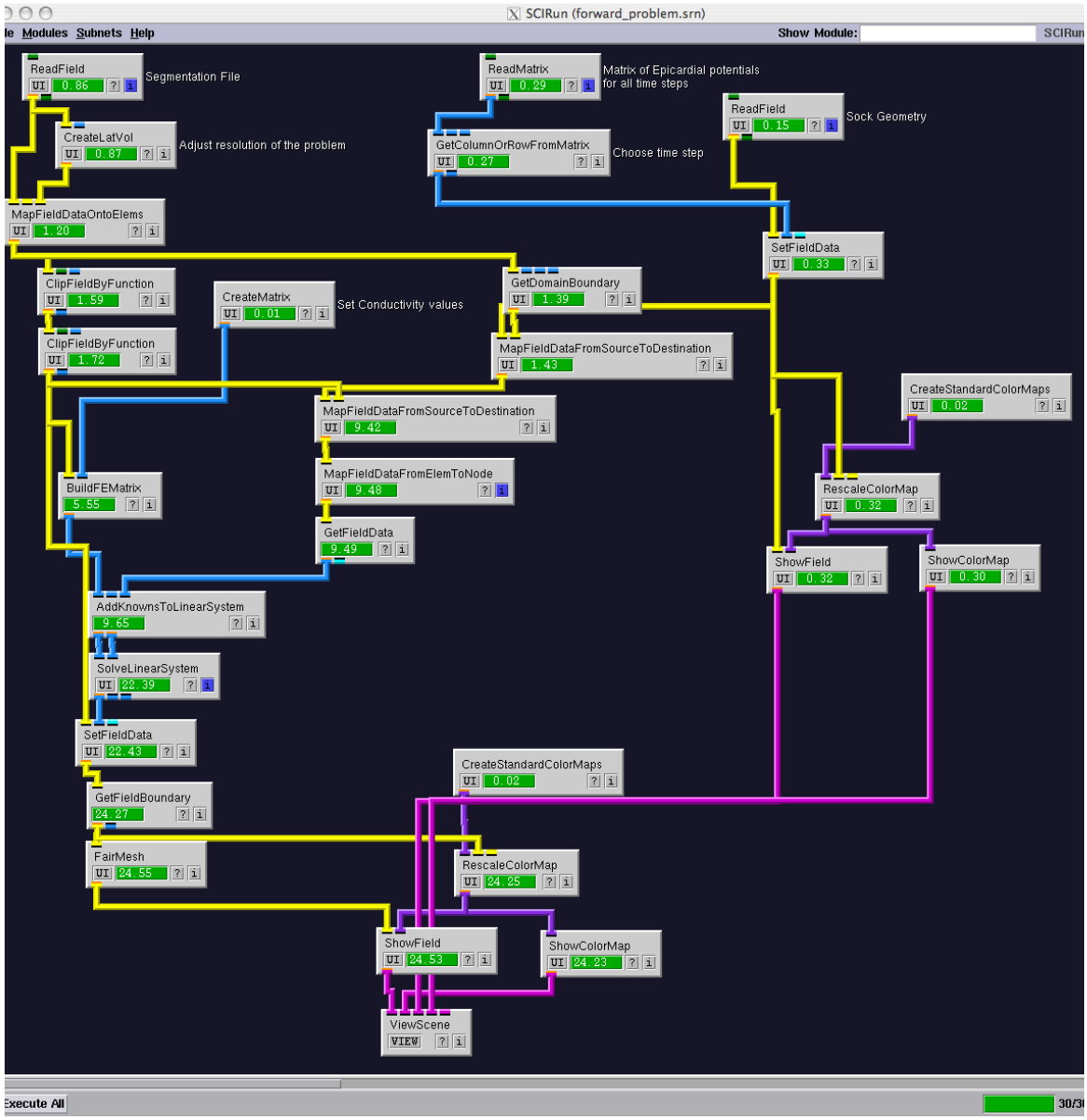

Figure 3.7. Simple potential based forward problem using FEM.

There are two examples given to calculate the potential-based forward problem using finite element method. One example involves the generation of the lead field matrix, or transfer matrix, then performing the forward calculation. The other example is a direct calculation of the projection of the cardiac potentials onto the torso surface, satisfying Laplace's equation.

The inputs to the example networks in this section are all the same: a torso segmentation including segmentation of the heart volume with a closed epicardium, a cardiac sock geometry registered to the heart volume from the segmentation, and a matrix of epicardial recordings that were obtained with an epicardial sock. To calculate the lead field matrix, an identity matrix is also used for input vectors.

The example network
`src/nets/FwdInvToolbox/potential-based-fem/forward_problem.srn` provides an example of an easy implementation of the projection of the cardiac potentials onto the torso surface (Figure 3.7). Though this problem is not formulated in the classic sense of the forward problem, it is useful because it does not require calculating the lead field matrix, which can take longer to compute in most cases. The example presented is useful if a forward calculation requires a few time-instances for a given geometry. Different time points may be selected in the `GetColumnOrRowFromMatrix` module.

As mentioned above, the method implemented in this network uses Laplace's equation to find the torso potentials. That is, all the potentials in the torso satisfy the expression:

$$\nabla\sigma\nabla\phi = 0$$

where ϕ are the potentials and σ are the conductivity values. Using SCIRun, one is able to use the cardiac surface potentials as known values and solve the rest of the torso potentials to satisfy Laplace's equation as a linear system.

The example network
`potential-based-fem/forward_problem_with_lead_field_matrix.srn` is a network that computes the forward problem in a more traditionally understood method (Figure 3.8), in that it is a simple matrix multiplication to calculate the torso potentials. This network simply uses a pre-computed lead field matrix and multiplies it by the cardiac potentials to yield the surface torso potentials. Different time points may be selected in the `GetColumnOrRowFromMatrix` module.

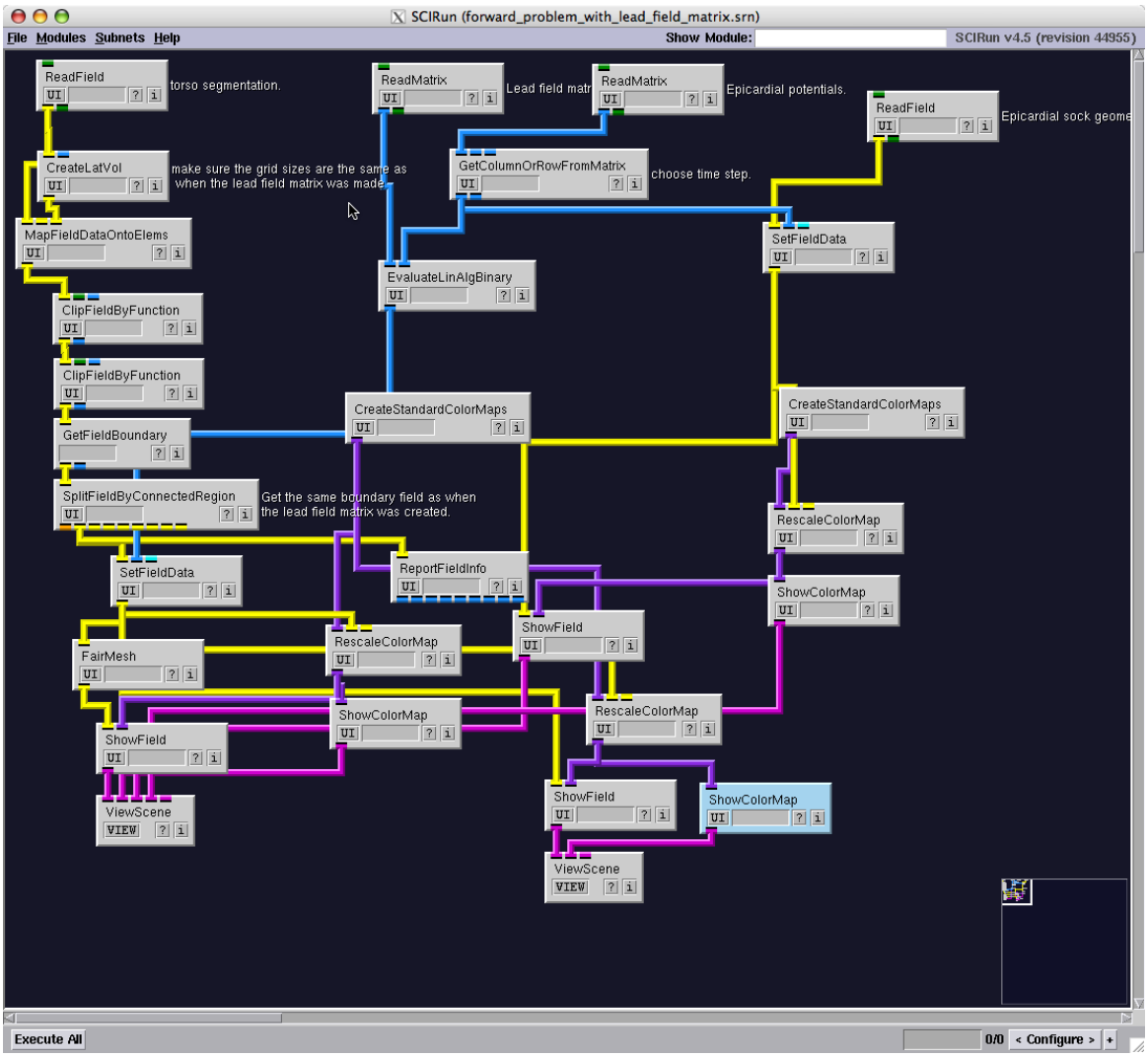

Figure 3.8. Simple potential based forward problem using a pre-computed lead field matrix.

The lead field matrix provided was generated by the network `src/nets/FwdInvToolbox/potential-based-fem/Make_Lead_field_matrix.srn` (Figure 3.9). This network solves and collects the solution vectors relating isolated points on the heart to the torso. The solution vector is calculated in the same manner as in the network `src/nets/FwdInvToolbox/potential-based-fem/forward_problem.srn` described above, using orthogonal unit vectors as the cardiac potential (value of 1 at the point of interest and 0 elsewhere). Computing the lead field matrix this way is very useful if several forward calculations are needed on the same geometry.

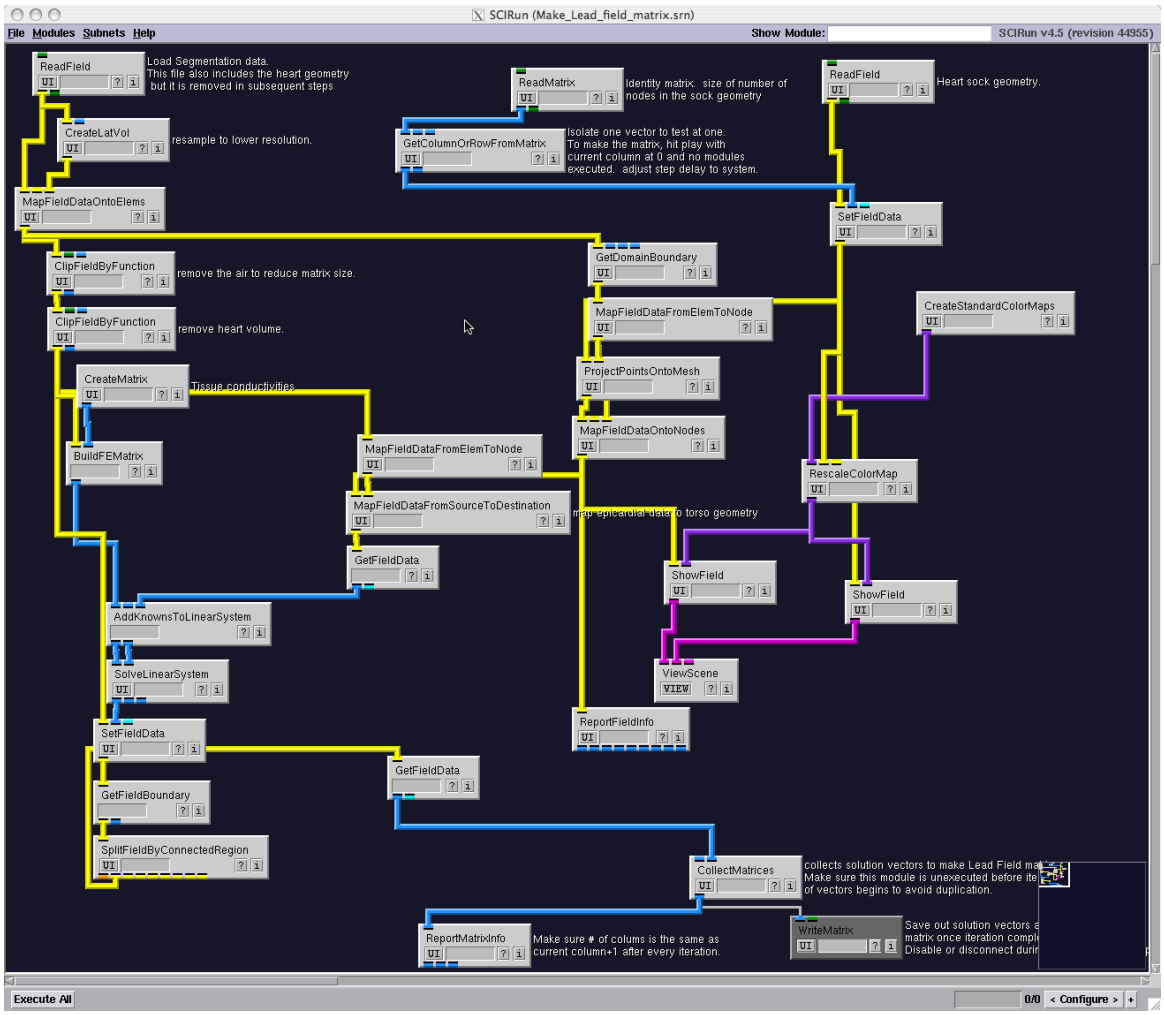

Figure 3.9. Computing the FEM based lead field matrix.

To calculate the lead field matrix with `src/nets/FwdInvToolbox/potential-based-fem/Make_Lead_field_matrix.srn`:

1. Make sure the `CollectMatrices` module has not been executed.

2. Set step delay in the `GetColumnOrRowFromMatrix` module to a sufficient interval so that all the modules can fully execute before the next iteration.

3. Set the current column to 0 and press "plays" in the `GetColumnOrRowFromMatrix`.

4. After the network finishes iterating, enable the `WriteMatrix` module (by right clicking on the module), choose the place and name of the matrix you would like, and press save.

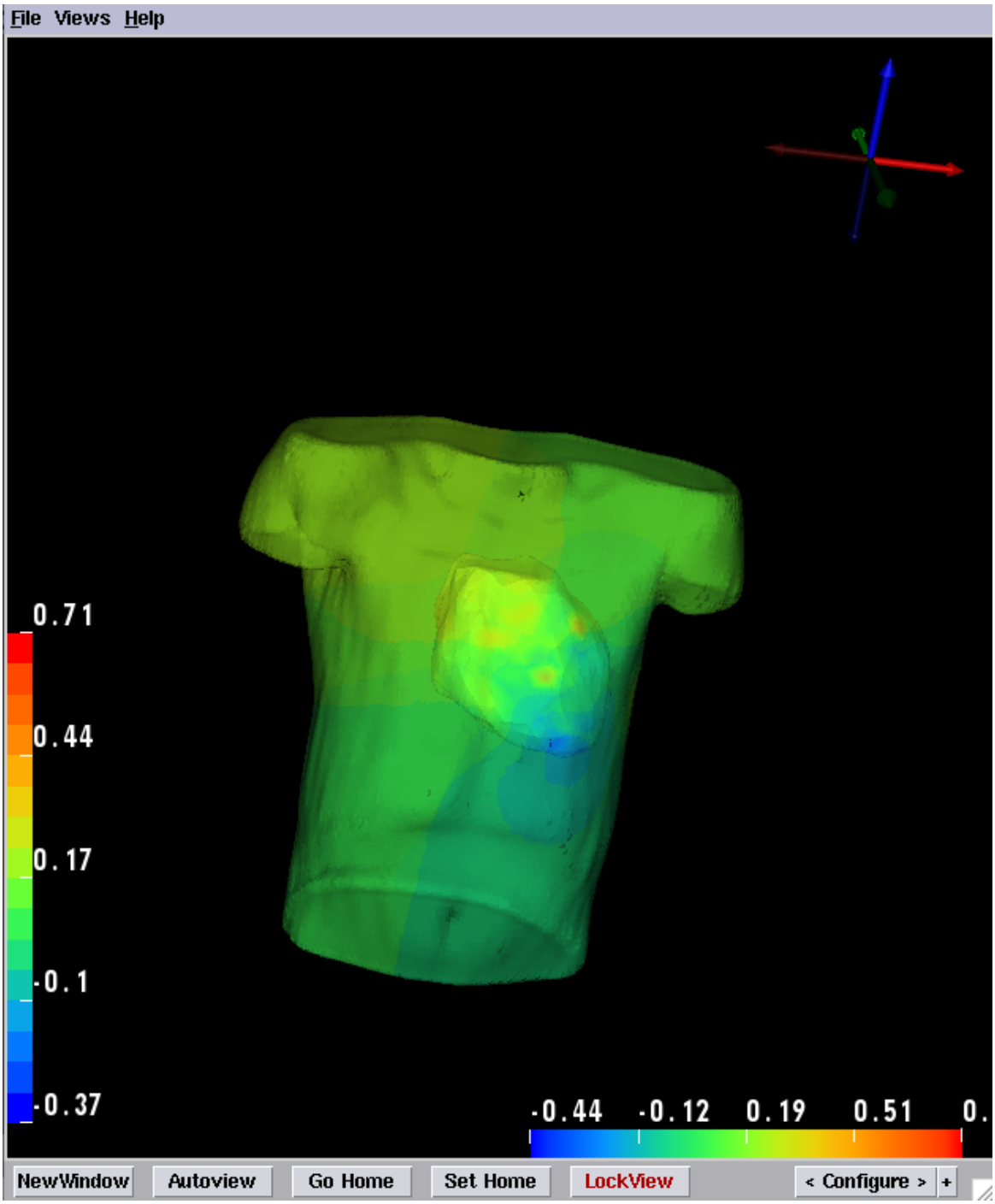

Figure 3.10. Solution of the potential based forward problem using FEM.

3.5.2 Activation Based Forward FEM simulation

The technique used by the activation-based model is rather different than the potential-based one. These methods, instead of searching solely for the potentials, consider whether a region of the heart is "active" or not and which is the profile of its potential. This is very different than the activation profile on a cellular level, as each surface node represents a large region of the heart. The activation profile is used to specify the strength, in time, of dipoles located on the surface.

Since FEM is a volume method, SCIRun approximates the volume potentials by first clipping it with the surface mesh of the inner and outer boundaries. With this, it obtains a volume mesh that roughly follows the surface. Then, the potentials of this volume mesh are set so that they approximate the dipoles on the heart surface. This representation is known as the equivalent dipole layer.

There are three networks located in `src/nets/FwdInvToolbox/activation-based-fem` which provide an example of activation-based forward calculation using FEM. The three networks are similar in nature to the potential-based FEM forward problem networks explained in Sec. 3.5.1. The activation-based forward problem networks use data derived from ECGSIM. The surface mesh and source parameters are also obtained from the ECGSIM project.

The primary inputs to this network are the surface mesh representing the heart, a torso surface mesh which is made a volume mesh (refined near the surface of the heart) in the network, and a matrix specifying the activation profile for each point in the heart surface mesh at each time to be simulated. This matrix is then of size $N \times M$ where N is the number of surface points and M is the number of time-steps represented. A very good example of the input expected can be obtained from the ECGSIM software by choosing *Save* $\rightarrow$ *Source parameters* from the *File* menu.

Though the `activation-based-fem.srn` networks seem quite complicated, they perform some simple tasks. The loaded torso surface is made into a volume and refined in nearer to the heart. Then, two cardiac surfaces are made from the first, one slightly inside the heart, and one slightly outside. The corresponding points on the two surfaces are set to either active or inactive. If the point is inactive, then it is set as an unknown and will be solved for later. If the point is active, the two surfaces are set to opposing values to approximate a dipole layer. The remaining torso potentials are solved using a linear-systems solver similar to Sec. 3.5.1.

The network `activation-based-fem_lead_field.srn` produces similar results as the previous network, but it utilizes a pre-computed lead field matrix. This network allows for the activation times to be directly related to the torso surface and the computation is relatively quick. This method is also more like the traditionally formulated forward problem with a simple matrix multiply.

The lead field matrix is calculated using the `make_lead_field.srn` network. Similar to the potential-based approach (Sec. 3.5.1), the solution vectors for each point on the cardiac surface are calculated and collected as the lead field matrix. This is essentially a pre-computation of the contribution of each point. This calculation is very useful if the geometry is going to be used many times.

To calculate the lead field matrix with `make_lead_field.srn`:

1. Make sure the `CollectMatrices` module has not been executed.

2. Set step delay in the `GetColumnOrRowFromMatrix` module is set to a sufficient interval so that all the modules can fully execute before the next iteration.

3. Set the current column to 0 and press "play" in the `GetColumnOrRowFromMatrix`.

4. After the network finishes iterating, enable the `WriteMatrix` module (by right clicking on the module), choose the place and name of the matrix you would like, and press save.

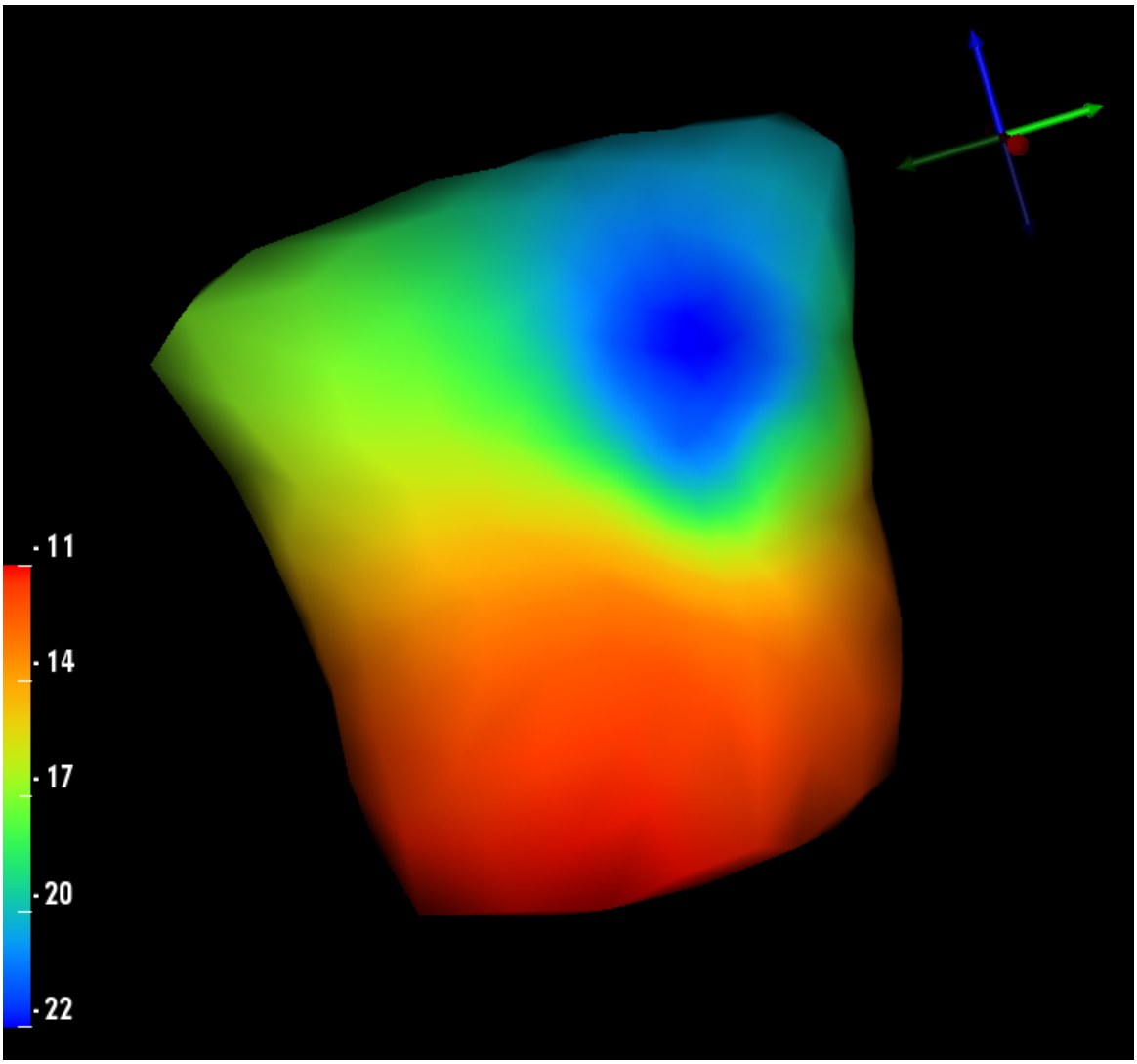

Figure 3.11. Solution of the activation based forward problem using FEM.

Chapter 4

Inverse Solutions

4.1 Overview

The inverse problem of electrocardiography is to find suitable electrical source parameters on the heart that adequately describe the observed body surface potentials. Regularization is typically employed in solution methods to reduce the sensitivity of the problem to relatively small errors in the observed body surface potentials, thereby stabilizing it. As a result, solution methods may be supplied body surface potentials, a forward model, and method-specific regularization parameters as input. Specific use of each method is described below. As output, modules primarily produce the resulting solution with extra outputs, depending on the specific method.

4.2 Module Descriptions for Inverse Solution Methods

4.2.1 Tikhonov Regularization

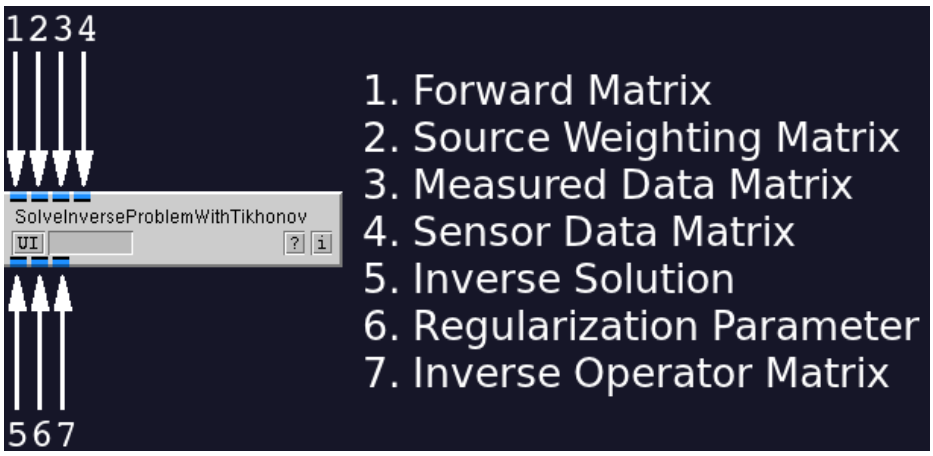

Figure 4.1. Revised Tikhonov module: `SolveInverseProblemWithTikhonov`.

In order to perform inverse ECG, the Tikhonov regularization can be used, which is implemented in SCIRun as the module `SolveInverseProblemWithTikhonov`. The module requires a forward matrix and a vector of observed/measured body surface potentials as inputs (see figure 4.1, module input ports 1 and 3). Weighting schemes for the measurements and source space (see figure 4.1, module input ports 2 and 4) can be fed in optionally. These *a priori* weighting matrices are **NOT** inverted in the module - if needed they must be provided already inverted as input. The outputs of the module are the computed solution, the used regularization parameter and the used inverse operator matrix (see figure 4.1, module output ports 5, 6 and 7). The inverse operator G is defined for the overdetermined case as:

$$G = (A^T P^T P A + \lambda^2 L L^T)^{-1} A^T P^T P \tag{4.1}$$

and for the underdetermined case:

$$G = W W^T A^T (A W W^T A^T + \lambda^2 C C^T)^{-1}. \tag{4.2}$$

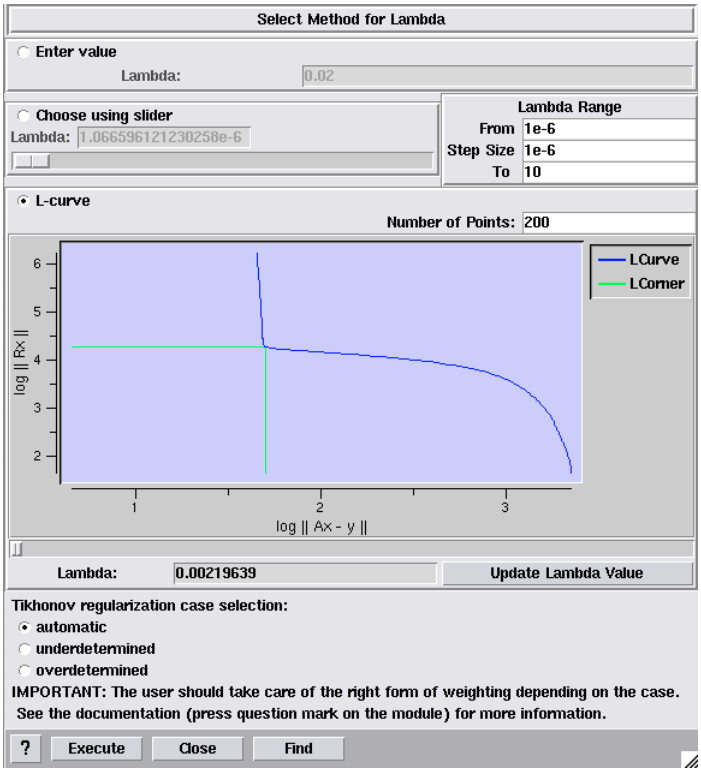

Figure 4.2. GUI from revised Tikhonov module: `SolveInverseProblemWithTikhonov`.

Manual selection of the regularization parameter "λ"

The user is able to enter a specific regularization parameter to be used in the inverse calculation. Further, the user can also specify a range of equally distributed regularization parameters and select a particular one using the slider, as shown in figure 4.3.

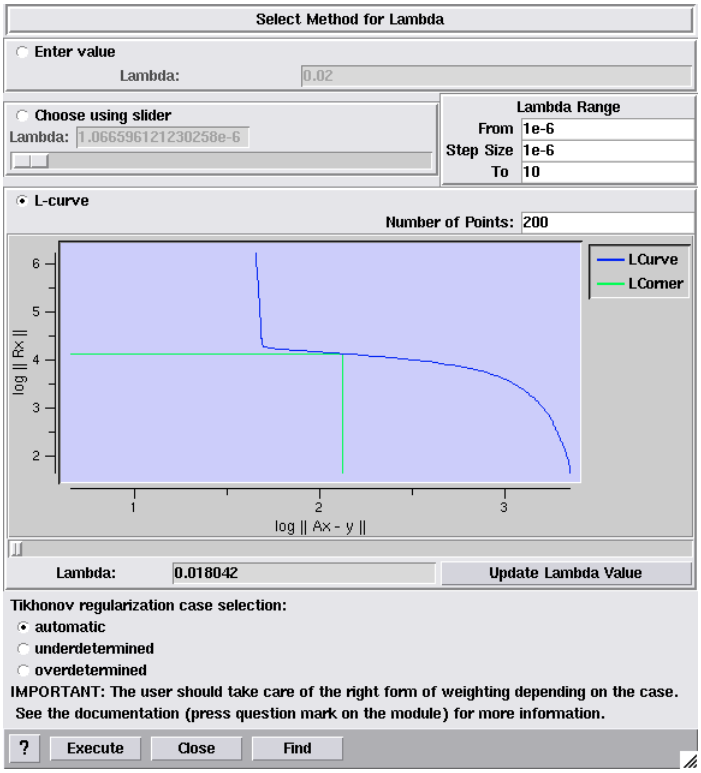

Figure 4.3. Select regularization parameter using the slider in `SolveInverseProblemWithTikhonov`.

Automatic selection of the regularization parameter "λ" - the L-curve

In figure 4.2, an example of a L-Curve is shown. Initially, the automatic procedure to find the L-Curve corner evaluates the curvature at each of the discrete points of the L-curve and estimates the maximal absolute curvature. In some cases, the automatically chosen regularization parameter is not optimal. Because of this, a slider under the L-Curve is available; it allows the user to choose a better regularization parameter. While moving the slider, the chosen L-Curve-Corner depicted in green, is updated. If a more reasonable regularization parameter is found by the user, it can be used later on for a single inverse computation by pressing the button, **Update Lambda Value**. This will load the lambda value in the uppermost input field, which can later be activated with the radio button next to it.

Computational efficiency

The module can automatically (by default) decide which case (overdetermined or underdetermined) should be used to benefit from the most computationally efficient approach. The

user can manually decide which case should be used as well (clicking to one of the radio buttons under **Tikhonov regularization case selection**).

To start any computation, the **Execute** button must be clicked.

4.2.2 Tikhonov Singular Value Decomposition (SVD)

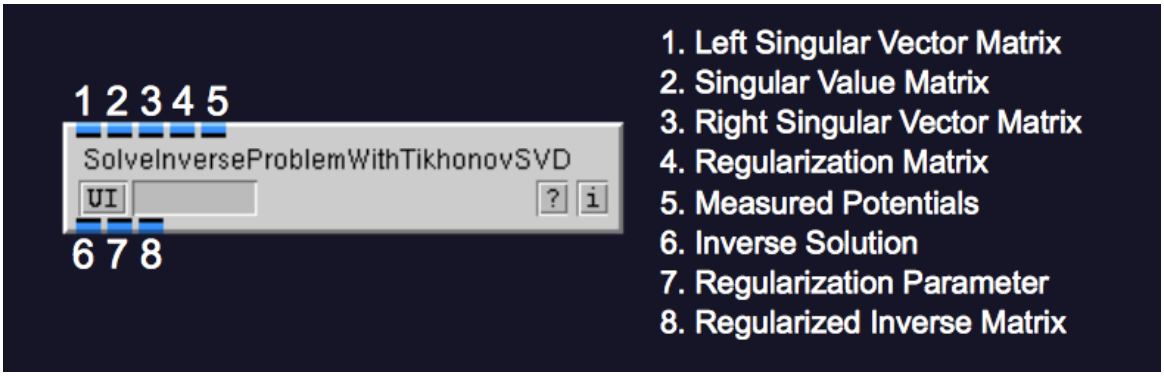

Figure 4.4. Tikhonov SVD Module.

This is a method of solving the potential-based inverse problem.

This module implements Tikhonov regularization (closed-form solution) using the singular value decomposition (SVD) in SCIRun, and is called `SolveInverseProblemWithTikhonovSVD`. The module requires that the SVD of a forward solution matrix (left/right singular vector matrices and singular value matrix), regularization matrix (such as a surface Laplacian approximation), and a vector (single-column matrix) of observed/measured body surface potentials are supplied as input.

In the user interface (UI), one can explicitly specify a scalar regularization parameter that weights the influence of the regularization matrix on the solution. The module can also automatically select a regularization parameter using the L-curve method.

Upon completion, the module will output the inverse solution. In addition, it will either output the specified or automatically-chosen regularization parameter, as well as the regularized inverse matrix (a closed-form Tikhonov inverse operator, computed using the provided SVD).

4.2.3 Truncated SVD

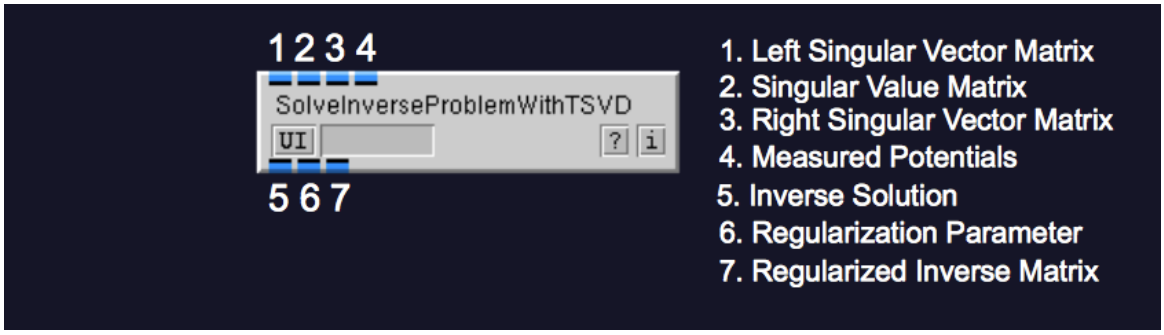

Figure 4.5. Truncated SVD Module

This is a method of solving the potential-based inverse problem.

This module, called `SolveInverseProblemWithTSVD`, computes an inverse solution by creating a pseudo-inverse operator and multiplying that with the measured body surface potentials. The module requires that the SVD of a forward solution matrix (left/right singular vector matrices and singular value matrix) and a vector of observed/measured body surface potentials are supplied as input.

In the user interface (UI), one can explicitly specify a scalar-integer regularization parameter to control the amount of regularization imposed on the problem. Assuming the singular values are sorted from largest to smallest (and therefore most significant to least significant), the regularization parameter controls how many significant singular values to consider (truncation degree) when calculating the pseudo-inverse. Alternatively, the module will use the L-curve method to automatically obtain a suitable truncation degree.

The primary output of this module is a vector containing the inverse solution. In addition, it will output the regularization parameter and regularized inverse operator used to obtain the inverse solution from the measured body surface potentials.

4.2.4 Matlab Interface

The *InterfaceWithMatlab* module allows a SCIRun network to make use of Matlab for some of its calculations. In Figure 4.6, the user interface for this module shows how SCIRun data types from the module's input pipes are converted to Matlab variables for use in a Matlab script. Upon completion of the Matlab script, Matlab variables are converted back to SCIRun data types to be sent to the module's output pipes.

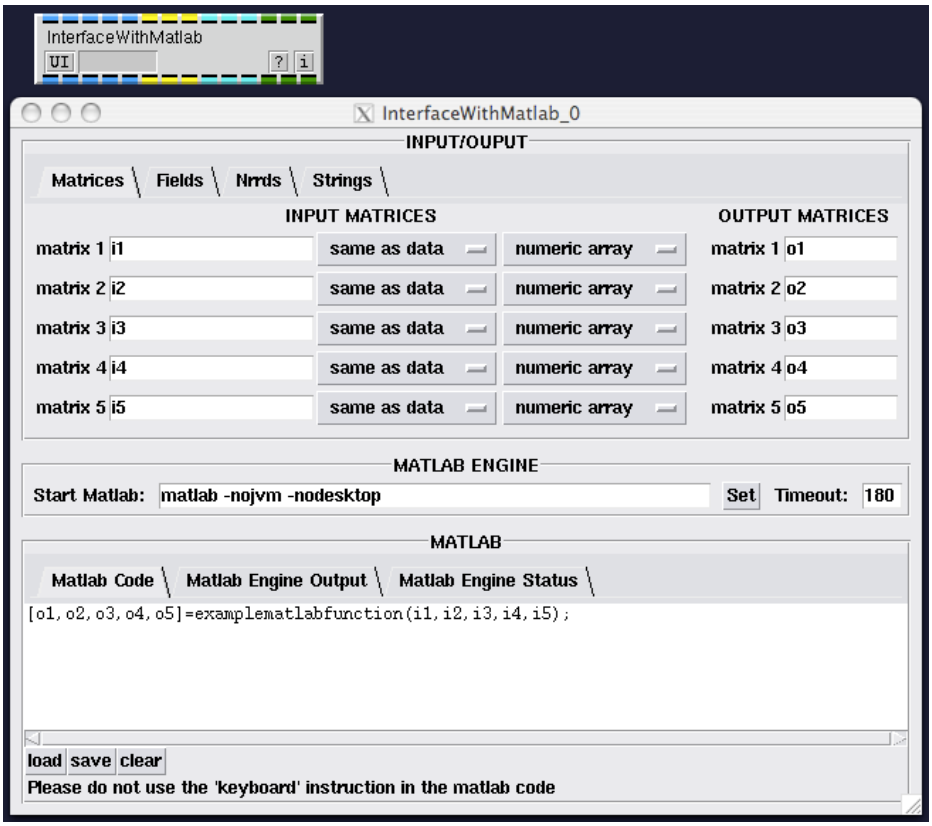

Figure 4.6. The InterfaceWithMatlab Module and its corresponding user interface

In the rest of this section, we will describe some of the inverse solution methods distributed with SCIRun that have been implemented in Matlab.

Isotropy Method

This is a method of solving the potential-based inverse problem.

```
function X_reg=greensite(A,Y,trunc_deg)
% Function to calculate the Greensite inverse solution
%   A - forward matrix
%   Y - data
%   trunc_deg - truncation degree
%   X_reg - inverse solution
```

The isotropy method (a.k.a. Greensite method) takes as input a forward solution matrix, measured body surface potential data, and an optional integer regularization parameter (truncation degree). In this method, the truncation degree controls the number of significant singular values of the measured body surface potential data to be used when computing the inverse solution. Consequently, the truncation degree partitions the measured body surface potential data into independent "signal" and "noise" components, ignoring the "noise"

component in its calculation of the inverse solution. The only output of the method is the inverse solution.

Gauss-Newton Method

This is a method of solving the activation-based inverse problem.

```
function tau = ActGaussNewton(A,Y,L,tauinit,lambda,w,minstep)
% Implements the Gauss-Newton algorithm for solving the activation-based
% inverse problem of electrocardiography.
% => minimizes the objective function ||Y-A*X||^2+lambda*||L*X||^2 where
% X is parameterized by the C^1 polynomial approximation to a step function
% as explained in "The Depolarization Sequence of the Human Heart Surface
% Computed from Measured Body Surface Potentials" by Geertjan Huiskamp and
% Adriaan van Oosterom.
%
% Input Variables:
% A: Forward matrix
% Y: Observations (columns index time from 1 to T=size(Y,2))
% L: Regularization matrix (typically a surface Laplacian approximation)
% lambda: Regularization parameter
% w: Width parameter in step function approximation
% tauinit: Initial phase shifts for starting the algorithm
%
% Output Variables:
% tau: Solution phase shifts of the step functions
```

The activation-based inverse problem of electrocardiography is to solve for electrical activation times (i.e. depolarization times) on the heart, given body surface potentials during the QRS complex. This results in a nonlinear least-squares optimization problem that this method solves using the Gauss-Newton algorithm. Solutions are typically highly dependent on the choice of the initial guess. This method requires a forward solution matrix, body surface potentials, regularization matrix, initial guess, regularization parameter, activation waveform transition width, and convergence parameter to be specified as input. The body surface potentials should be provided as a $M \times T$ matrix, where M is the number of leads from which potentials are recorded and T is the number of time samples recorded during the QRS complex of the heart, whose activation times are to be estimated. The regularization matrix and corresponding parameter influence the inverse solution as in Tikhonov regularization (see above). The initial guess is the set of activation times from which the Gauss-Newton algorithm starts pursuing a more suitable inverse solution. The transition width controls the number of time samples (not necessarily integer) taken by each source to transition from inactivated to activated in this method. The convergence parameter is the minimum norm of the step taken by the iterations within the Gauss-Newton algorithm before the method is deemed to have converged to a suitable solution. The only output of the method is the inverse solution (in this case: an array/vector of activation times).

Wavefront-Based Potential Reconstruction

This is a method of solving the potential-based inverse problem.

```
function [x_WBPR_forward,x_WBPR_backward] = WBPR(A,y,heart,first_act,last_act)
% Function to calculate the WBPR solutions
% Inputs:
%   A: forward matrix
%   y: torso data
%   heart: heart geometry
%   first_act: first activated node
%   last_act: last activated node
%
% Outputs:
%   x_WBPR_forward: inverse solution using forward WBPR
%   x_WBPR_backward: inverse solution using backward WBPR
```

The Wavefront-Based Potential Reconstruction (WBPR) method imposes prior knowledge about the spatial patterns of electric potentials on the heart during the QRS complex to reconstruct an inverse solution. This method requires a forward solution matrix, measured body surface potentials, a structure containing the triangles and nodes of the heart geometry mesh, and the first/last nodes to activate during the QRS complex of the heart beat. The body surface potentials should be provided as a $M \times T$ matrix, where M is the number of leads from which potentials are recorded, and T is the number of time samples recorded during the QRS complex of the heart beat, whose electric potentials are to be estimated. The heart geometry is a Matlab "struct" that must contain a matrix of node coordinates and a matrix of indices for triangles that connect the nodes on the surface of the heart. The first/last nodes to activate are specified as the indices of the nodes.

As output, WBPR produces two inverse solutions. The procedure by which the inverse solutions are obtained may be run both forward and backward in time. Therefore, the "forward WBPR" solution is that obtained by applying the method starting from the earliest sample time and ending at the latest. On the other hand, the "backward WBPR" solution is obtained by starting from the latest sample time and traversing the samples in reverse until ending at the earliest.

4.3 Example Simulations

4.3.1 Tikhonov Regularization

The SCIRun network for this example can be found at:
`src/nets/FwdInvToolbox/potential-based-inverse/tikhonov-inversion.srn`
in the SCIRun source code directory.

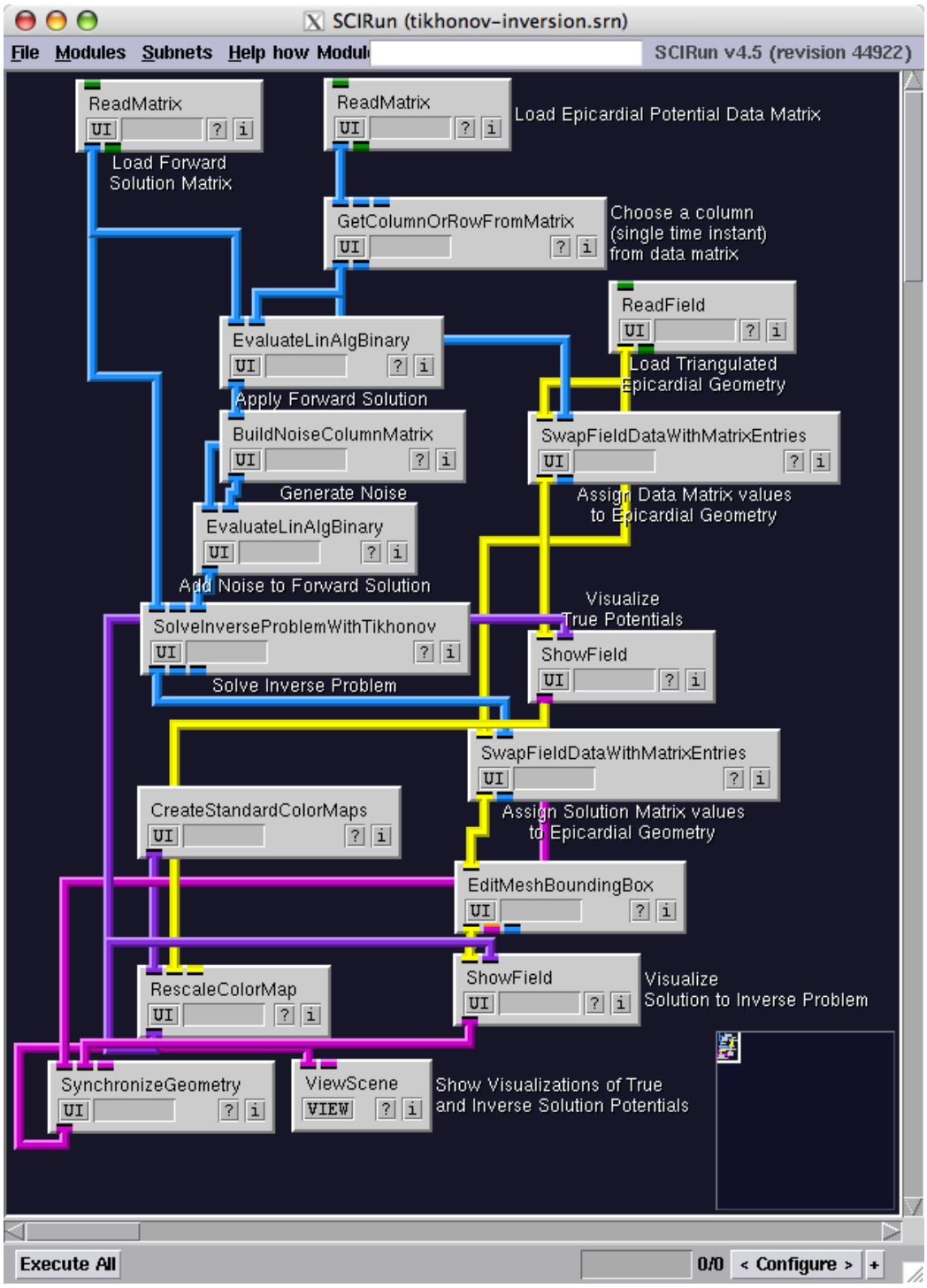

Figure 4.7. The SCIRun network for the Tikhonov inverse solution example.

This example shows how to use the `SolveInverseProblemWithTikhonov` module in a SCIRun network to solve a potential-based ECG inverse problem using Tikhonov regularization. Specifically, this example simulates "measured" body surface potentials by applying

a forward solution and adding pseudorandom noise with a specified signal-to-noise ratio (SNR). The forward solution is obtained by multiplying a known vector of epicardial potential data by a pre-computed forward solution matrix. The simulated measurements and forward solution matrix are the inputs to the `SolveInverseProblemWithTikhonov` module. The regularization parameter of this method is controlled via the user interface (UI) in this example. The output from the module is the inverse solution. This example compares the inverse solution to the known epicardial potentials from which the measurements were simulated. A visualization of both can be seen in the UI of the `ViewScene` module.

4.3.2 Gauss-Newton Method

The SCIRun network for this example can be found at:
`src/nets/FwdInvToolbox/activation-based-inverse/actgaussnewton-inversion.srn`
in the SCIRun source code directory.

This example shows how to use the `InterfaceWithMatlab` module in a SCIRun network to solve an activation-based ECG inverse problem using the Gauss-Newton method. This example uses the "normal male" dataset and geometries from ECGSIM. Specifically, this method compares measured body surface potentials with body surface potentials simulated from activation times. We search the set of possible activation times for ones that minimize the squared error residual between the measured and simulated body surface potentials. The Gauss-Newton method in this example is implemented as a Matlab function and is used in SCIRun via the `InterfaceWithMatlab` module. As a numerical optimization method, Gauss-Newton will look for a local minimizer of the squared error residual starting from a suitable initialization point. In this example, we initialize the method with the activation times reported for the dataset in ECGSIM, but perturbed by noise.

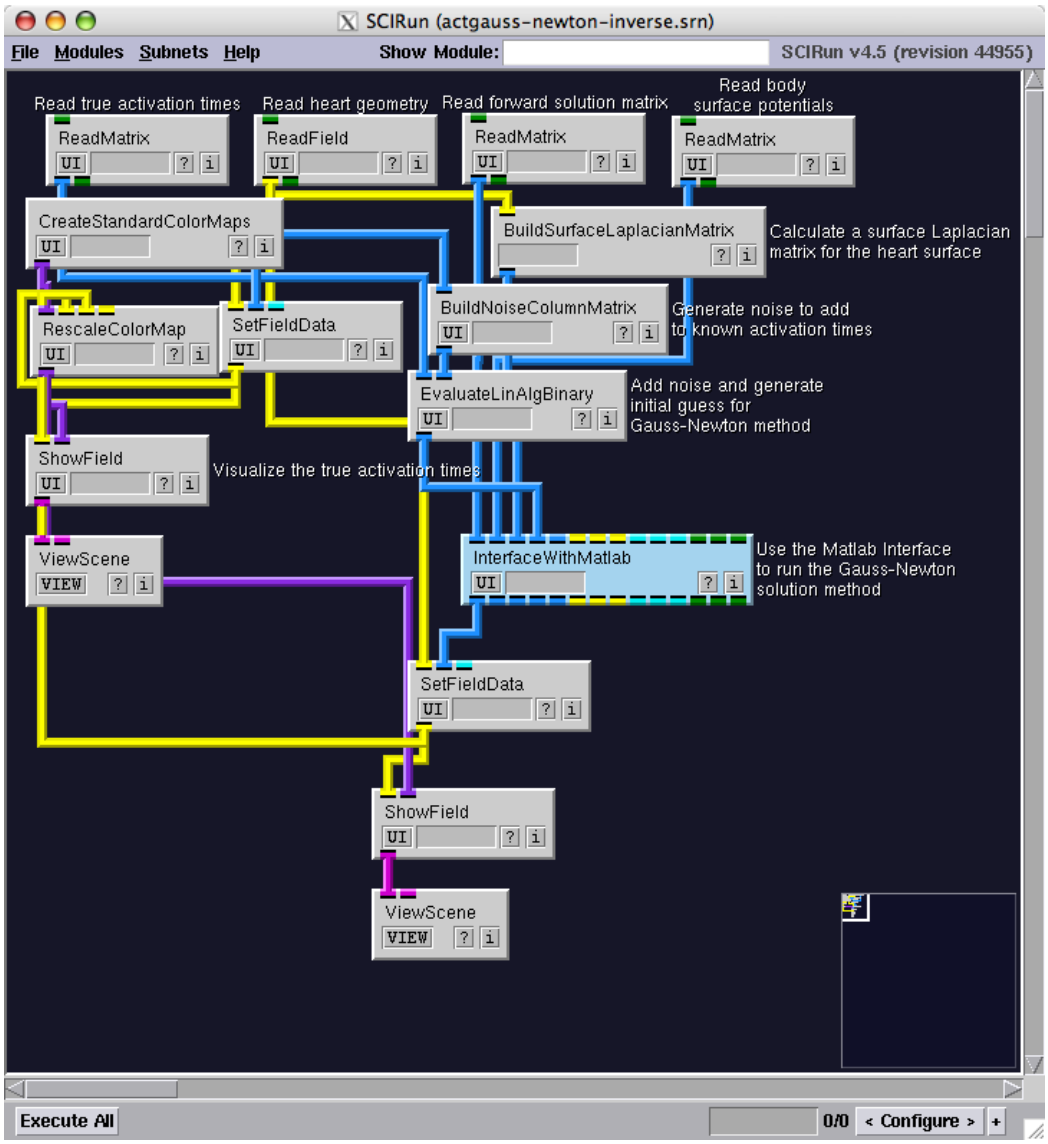

Figure 4.8. The SCIRun network for the activation-based Gauss-Newton inverse solution example.