SCIRun Forward/Inverse ECG Toolkit

SCIRun 5.0 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 the National Institute of General Medical Sciences of the National Institutes of Health under grant number **(P41GM103545)**

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 Python, or Matlab through Python, so that processing can be carried out jointly using both programs. In addition, the ECG Forward/Inverse

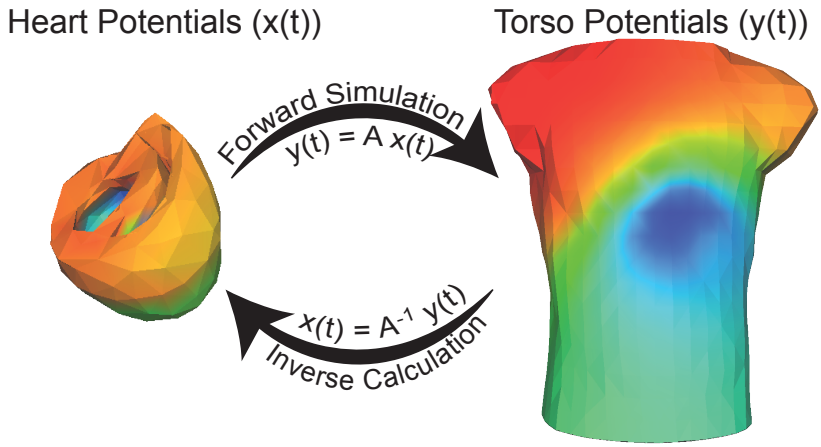

Figure 1.1. Forward and inverse problems in electrocardiology. The forward simulation involves computing the torso potentials with know cardiac sources and torso geometry. The inverse solution, called ECG Imaging (ECG) consists of inverting the forward model to calculate cardiac sources from body surface recordings.

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$^{\&*\ddagger}$
FEM Lead Field Calculation	Potential-Based Regularization Methods
Stiffness Matrix Calculation	- Tikhonov
	- Tikhonov SVD
	- Truncated SVD
	- Isotropy method*
	- Gauss-Newton Method*‡
	- Wavefront based potential reconstruction*
	- Total Variation*
	- Method of Fundamental Solution (MFS) ‡

Table 1.1. Algorithms currently included in the CIBC ECG Forward/Inverse toolkit
†Activation-based forward solution is not yet available
$^{\&}$Based on a Gauss-Newton optimization
* Matlab implementation ‡ Python implementation

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, also called ECG forward Simulation, 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, also called ECG imaging or ECGI, 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 ECG Imaging presupposes an available solution to the forward simulation. Here we briefly summarize the basic background to facilitate our description of the toolkit capabilities in subsequent chapters. Please refer to the list of literature on the CEI webpage for more information (http://www.ecg-imaging.org/home/publications).

Solving both forward and inverse problems requires a specific model formulation of the cardiac electrical sources. This toolkit contains examples from the two arguably most dominant equivalent source models used 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 [1]. 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 bioelectricity are relatively low, and the wavelengths many orders of magnitude larger than the dimensions of the human body. So, from a bioelectricity 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:

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

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

where Ω_k is the surface on which the sources are located, V_k are the potentials on that surface, and $\hat{n}$ are the surface normals of the surface.

As an alternative to solving this problem directly, a (weighted) integral can be taken over the solution domain on both sides of Equation 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. 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 interpolating the 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 Equation 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 Equation 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 Equation 2.4 is substituted into Equation 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. 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.

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 Equation 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 [2]. From this collection of solutions we can obtain the desired transfer matrix, which we will denote as $\mathbf{A}$ as used in Equation 2.8. This network is described below in Ch. 3.

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, 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 [3]. 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 re-

sulting 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 Equation 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 $\mathbf{y}(\mathbf{t})$ 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 **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 the acitvation-based inverse 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 $Ax = y$, which is the same as Equation 2.8 with x replacing $x(t)$ for the unknown solution and y replacing $y(t)$ for the given measurement. The forward solution matrix **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 **P** represents the *a priori* knowledge of the measurements. The matrix **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 Equation 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 **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 **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.

Spline-Based Inverse Method

As in the Isotropy method, the Spline-Based method enforces prior knowledge in time. This method, introduced by Erem *et.al.*, enforces a smooth temporal behavior of the potentials on the heart through the paramaterization with a spline curve. The main assumption behind this model is that the sequence of multi-dimensional potentials, discretetized on the body surface by electrodes and heart by nodes on the geometry, across time lie on a 1D manifold embedded in a high dimensional space. Thus, position along this curve is equivalent to phase of cardiac cycle. As in the Isotropy method, the spline-inverse first attempts to estimate the temporal structure of the heart potentials by fitting a smooth-varying spline on the measured torso potentials. Then, it solves the inverse problem for the reduced set of knot points that characterize the fitted spline with a first order Tikhonov inverse method. The full temporal sequence of heart potentials is finally reconstructed by combining the inverse solutions obtained for each knot point with the temporal structure obtained from the torso potentials.

Total Variation Method

The effect of the ℓ_2-norm regularization in Equation 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, Equation 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:

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

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:

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

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 forward problem predicts the potentials observed along some measurement surface (commonly the body surface) given a set of known electrical sources and the geometry through which those source fields conduct. In this chapter, we will provide an overview of the forward problem as it is implemented in the Fwd/Inv toolkit, which generally calculates torso surface potentials given a set of known cardiac source parameters. Of course, these toolkit networks can be manipulated to incorporate other source models (*e.g.,* brain potentials or transmural cardiac potentials) or solution representations (*e.g.,* scalp or epicardial potentials) as needed.

The forward problem can be broken down into two fundamental concepts: 1) how to represent the source and 2) how that source maps to desired observation locations (*e.g.,* torso surface). One can conceptualize these two concepts in the classic sense by considering the function $\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t)$, where $\mathbf{y}(t)$ (Equation 2.8) represents a set of unknowns on the torso surface (observation locations) and $\mathbf{x}(t)$ is the cardiac source or source formulation. The function A, therefore, represents how source potentials are to be mapped to the observation field and is dependent on the choice of source model, the solution approach (*i.e.,* FEM/BEM), and relevant torso specific requirements such as conductivities and geometry.

The two most common source formulation models used in cardiac forward problems are cardiac potentials and activation times. The use of cardiac potentials is arguably a more intuitive formulation in that it is typically potentials that are recorded from cardiac-specific electrodes. Activation times must be *derived* from recorded signals and are, therefore, one step removed from the original recordings. In terms of abbreviated physiology, cardiac potentials relate the interaction of cardiac cells to changes in the extracellular potential fields of the myocardium. These extracellular cardiac potentials are then projected to the torso surface. The activation-time-based source formulation, on the other hand, deals with the fact that when inactive and active tissues exist in close proximity, a source much like a dipole or layer of dipoles can be generated that is representative of the propagating activation wavefront, which then projects to the torso 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 that source projects to the torso surface. As mentioned, this relationship is dependent on

the choice of source model, the solution approach, and the various properties of the torso. In both source models presented, the function A can be represented by an $N \times M$ matrix where N is the number of points on the cardiac surface and M is the number of points on the torso surface. This matrix representation is called the lead field, or transfer, matrix. Calculating this matrix can be computationally intensive, and there are several ways to approach it. The two examples provided in the Fwd/Inv toolkit find the lead field, or transfer matrix, based on finite element (FEM) and boundary element methods (BEM). The theory and mathematics behind these two methods are presented in chapter 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 activation based model in its current state, however, is incomplete. The network is provided, though it is still undergoing a series of improvements. The BEM activation-time forward problem is not yet supported in SCIRun, though a working version is available through ECGSim, the interactive simulation software developed by Peacs in the Netherlands. ECGSim assumes an equivalent dipole layer as the approximated source in the activation based forward problem and is compatible with SCIRun outputs.[1]

The general network structure of all forward problems in the Fwd/Inv toolkit can be broken down into 6 component phases: Phase 1: Data Inputs, Phase 2: Parameter Manipulation & Transfer Matrix Computation, Phase 3: Solution Calculation, Phase 4: Data Mapping, Phase 5: Visualization, and Phase 6: Data Output (though most networks provided omit the final phase). Descriptions using these phases will be utilized throughout each network description below.

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.

Following the format proposed above, the BEM Solutions network in the Fwd/Inv toolkit is broken up into only the first five phases.

- **Phase 1: Data Inputs**
  Cardiac (source) and torso (measurement/solution space) geometries as well as cardiac source potentials are read into the network. Additional fields are provided in the actual Fwd/Inv toolkit network that are unique to validating the specific model as well. This is because both heart and torso potentials are known for this example, both torso and heart data are available to be read in and used to simulate and validate the problem. As such, alternative geometries representing recorded torso potential sites with their associated recorded potentials are also included.

- **Phase 2: Parameter Manipulation & Transfer Matrix Computation**
  Modules assign source and measurement surfaces within the formulation and the conductivities between these respective surfaces as well as (if necessary) provide a means for defining/re-assigning surface normals.

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

- **Phase 3: Solution Calculation**
  With Data Inputs representing vector x of our linear system and the Transfer Matrix A being computed, the next step is to solve for b using the equation $Ax = b$.

- **Phase 4: Data Mapping**
  The solution to our problem involves a full time series of data inputs that were read in as part of Phase 1. Extracting a single time step from the solution and mapping it to the respective torso geometry is necessary for subsequent visualization.

- **Phase 5: Visualization**
  SCIRun offers an interactive visualization environment that is more thoroughly covered in the Basic SCIRun Tutorial documentation. Briefly, visualizations in the Fwd/Inv toolkit allow the user to display the source and solution geometries with data values mapped and colored onto each respective surface. The view window allows the user to then interactively manipulate the view orientation in order to generate the optimal visualization for the purposes of explaining the data.

Unique to the BEM implementation in SCIRun is the `BuildBEMatrix` module (Figure 3.1). This module accepts only surface inputs into a dynamic port structure, meaning that as additional inputs are added, additional input ports will appear (*notice the 3rd input port in Figure 3.1 even though only 2 inputs are provided*). A minimum of two surfaces must be used, though support for multiple surfaces is available. Generally, the initial input (input port 1) will accept the inner surface representing the source. The last output is reserved for the final outer measurement field (usually the torso) with all additional fields representing regions internal to the measurement field surface but external to the source surface (*e.g.*, lungs, bone, and other internal torso structures).

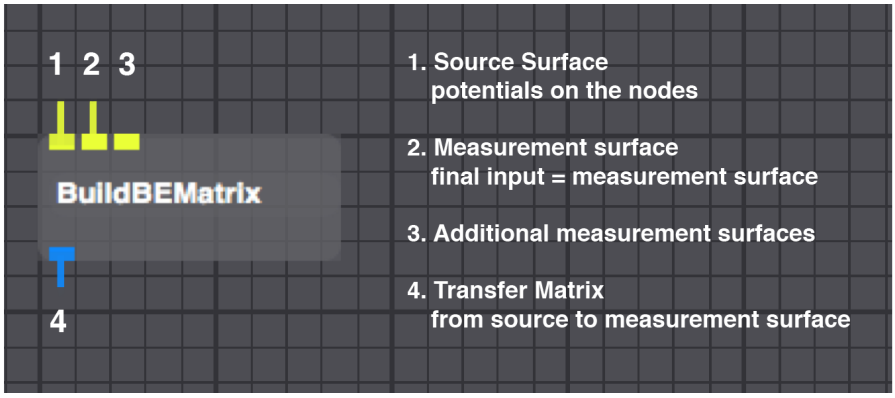

Figure 3.1. **BuildBEMatrix Module.** The `BuildBEMatrix` module computes the transfer matrix between the source surface and the outermost measurement surface.

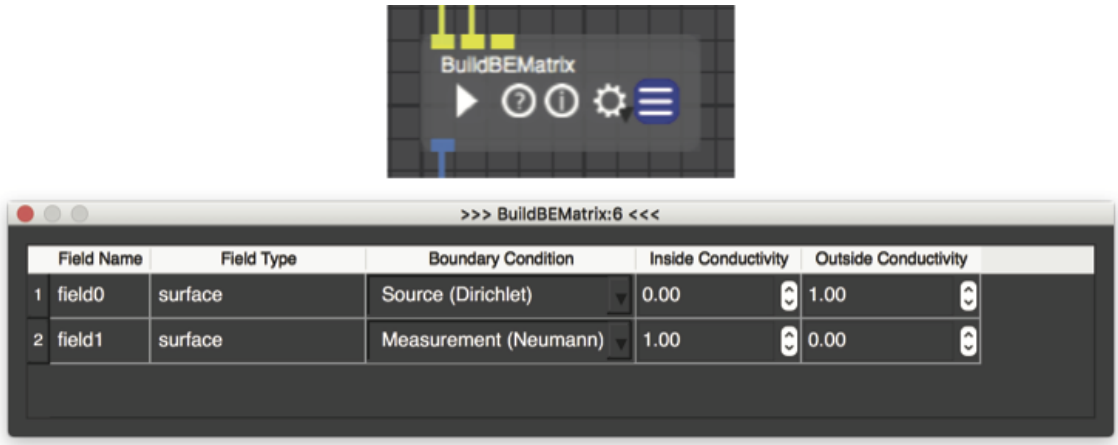

Figure 3.2. **BuildBEMatrix Module Interface.** The `BuildBEMatrix` UI allows the user to define boundary conditions and conductivities within the mesh.

Hovering over the `BuildBEMatrix` module produces a display of icons used to investigate and manipulate the module. The user interface (UI) icon is the right-most icon comprising three horizontal lines. Accessing the UI, as shown in Figure 3.2, allows the user to define surface specific parameters such as boundary conditions and conductivity values within the surfaces. Users should be careful to match conductivity values with respect to each surface. For example, in Figure 3.2, the outside source conductivity matches the inside conductivity of the torso surface.

Contrary to traditional BEM approaches, the `BuildBEMatrix` assumes that, for any closed surface, the surface normals will be pointing *inward* – not outward. The surface normal for each element is 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 on the surface. It is important to check the normals of model surfaces because incorrect normals will produce erroneous answers. Figure 3.3 shows a simple way to check surface normals in SCIRun using the module chain, or snippet.

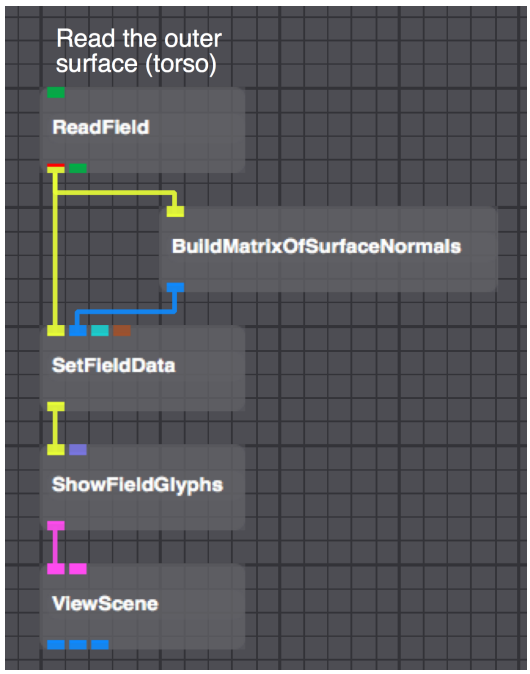

Figure 3.3. **Checking Surface Normals.** Checking surface normals requires the creation of a surface normal matrix that is then mapped back onto the original surface geometry. Showing field vectors as arrows allows the user to determine surface normal directions.

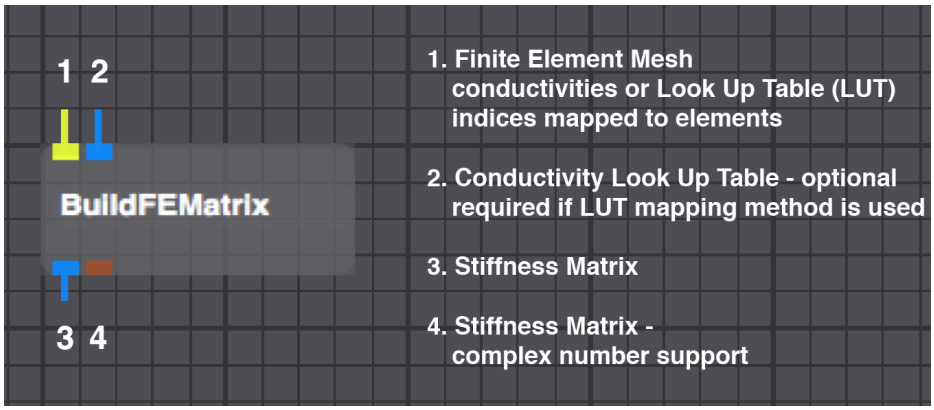

Figure 3.4. **BuildFEMatrix Module.** The `BuildFEMatrix` module that computes the stiffness matrix for a FEM solution.

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. `BuildFEMatrix` allows the user to compute a stiffness matrix while `AddKnownsToLinearSystem` provides a means for adding boundary conditions to the system. The `BuildFEMatrix` module receives a finite element mesh and an optional look up table (LUT) matrix of conductivity values. To simplify the assignment of conductivity values, the user can simply define conductivity values directly onto each element in which case, the LUT input port should be left unconnected. If the the lookup table method is preferred, the user must make sure that integer values are assigned to the mesh that correspond to the index numbers within the lookup table that contain the proper conductivities for each respective region. The resulting output of the `BuildFEMatrix` module is a stiffness matrix for the mesh that was generated using the Galerkin method with linear basis functions (Equation 2.5).

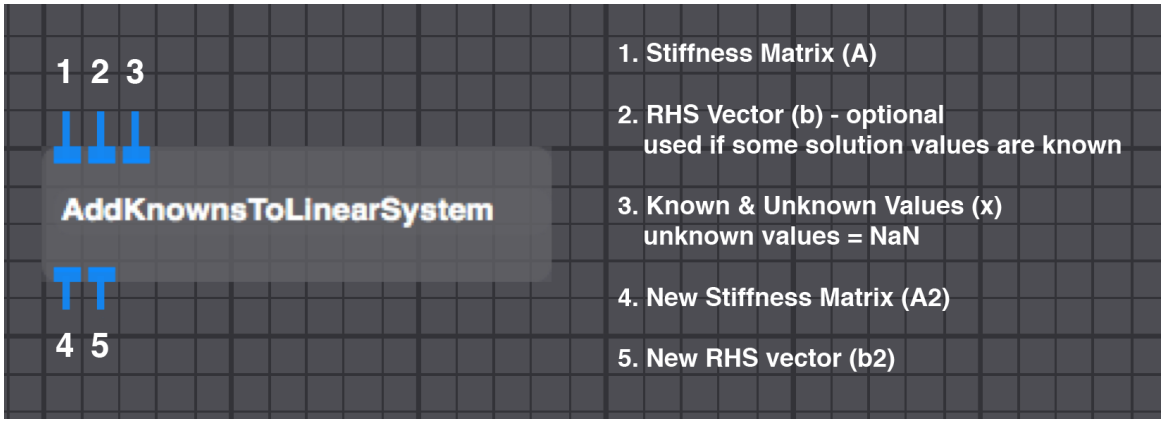

Figure 3.5. **AddKnownsToLinearSystem Module.** The `AddKnownsToLinearSystem` module recalibrates the stiffness matrix and RHS vector for a FE solution given known values in $\mathbf{x}(t)$ and (optionally) $\mathbf{y}(t)$.

`AddKnownsToLinearSystem` makes it possible to add known values as boundary conditions to the linear system $\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t)$, where some values of $\mathbf{x}(\mathbf{t})$ are already known. The module receives the stiffness matrix $\mathbf{A}$ along with an $\mathbf{x}(\mathbf{t})$ matrix with known values imposed at the specific indices where data is known and NaN values occupying the remaining vector entries. A right-hand-side (RHS) matrix may also be provided if any of the solution values are also known, but it is not required. If an RHS field is not included then it is assumed that the vector contains all zeros. Using the provided inputs, the module adjusts the linear system according to the known values. This means that $\mathbf{A}$ and the RHS vector $\mathbf{y}(t)$ are altered in such a way that the product of the new stiffness matrix and RHS $(\mathbf{A2})^{-\mathbf{1}}\mathbf{b2}$ (referring to Figure 3.5) will produce an $\mathbf{x}(\mathbf{t})$ that contains the defined values at the specified locations.

3.4 Example Networks for Boundary Element Solutions

The following network shows the most basic implementation of the boundary element method (Figure 3.6) along with a visualization of the results (Figure 3.7). To follow our previous standard:

- **Phase 1: Data Inputs**
  A torso mesh and epicardial surface mesh with associated data matrix of epicardial potentials at each node is read into the network.
- **Phase 2: Parameter Manipulation & Transfer Matrix Computation**
  The conductivities of the torso and heart are set, along with the differentiation between source and measurement surfaces as shown in Figure 3.2, and a transfer matrix is computed.
- **Phase 3: Solution Calculation**
  The transfer matrix is applied to the vector representing the potentials along the epicardium in the form of the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$.
- **Phase 4: Data Mapping**
  The solution is then mapped to the torso surface.
- **Phase 5: Visualization**
  Finally, the results are visualized and displayed using SCIRun's `ShowField` module.

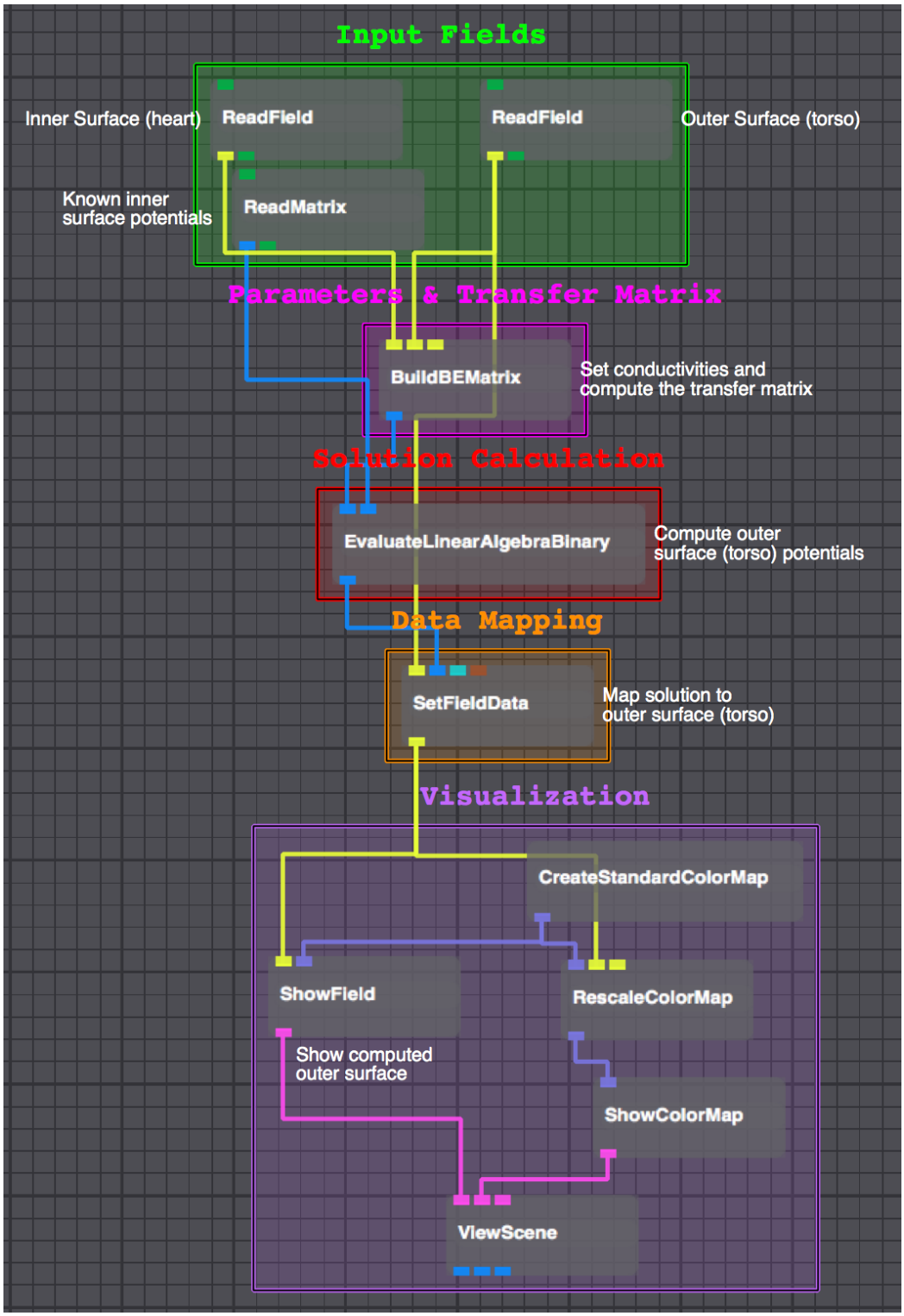

Figure 3.6. **BEM Network Example.** A simple implementation of the boundary element method in SCIRun.

A similar SCIRun network for this example can be found in the Fwd/Inv toolkit at: `[ToolkitPath]/Networks/potential-based-bem/torso-tank-bem.srn5`

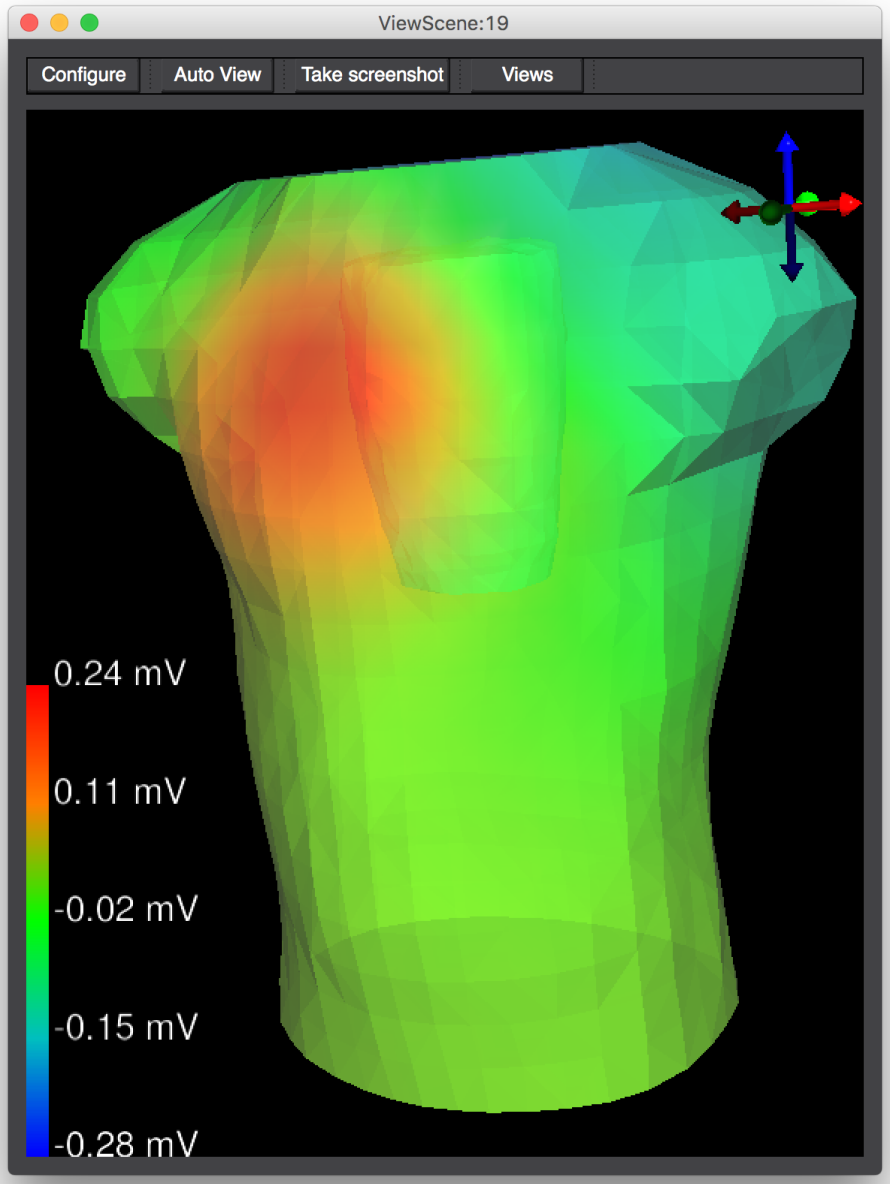

Figure 3.7. **BEM Solution Example.** Potentials from the internal surface (representing the heart potentials) were computed to the torso surface. An additional `ShowField` module was used in the above network (Figure 3.6) to display the cardiac region.

3.5 Example Networks for Finite Element Solutions

There are two examples in the Fwd/Inv toolkit for calculating the potential-based forward problem using finite element method. The first example is a direct calculation of the projection of the cardiac potentials into the torso mesh, satisfying Laplace's equation (Equation 2.1), and from thence to the torso surface. The second example involves the generation of the lead field matrix, or transfer matrix, that is then used to perform the forward calculation that very much resembles the $\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t)$ formulation seen in the BEM method above.

3.5.1 Potential Based Forward FEM Simulation: Laplacian Solve

The following example network shows a basic implementation of the finite element method used to solve Laplace's equation in order to, in essence, project cardiac surface potentials throughout the torso and onto the torso surface (Figure 3.8). Though this problem is not formulated in the classic sense of the forward problem, it is useful because it does not require calculating a lead field matrix, which is often limited to a low resolution domain due to the extensive time required to compute the lead field matrix. In each example it is assumed that the input meshes need no additional manipulation. The actual networks housed in the Fwd/Inv toolkit use torso segmentation files as meshes that need moderate regional geometric parsing in order to obtain the desired mesh. In this network time-series data is provided, but a solution for the time-series cannot be computed in a single step as was done in the BEM solution. Using the FEM approach with time-series input requires that the potential values at each desired time instant be extracted using the `GetFieldSlice` module. These data are subsequently mapped to the heart before an FEM solution is generated (Figure 3.8).

As mentioned previously, the Laplacian Solve method implemented in this network uses Laplace's equation to solve for torso potentials. That is, all the potentials in the torso satisfy the expression:

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

where ϕ are the potentials within the torso mesh 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 network is organized, again, in the following manner.

- **Phase 1: Data Inputs**
  A labeled torso mesh and epicardial surface mesh with associated time-series data matrix of epicardial potentials over several time instances is read into the network.
- **Phase 2: Parameter Manipulation & Transfer Matrix Computation**
  Potential data are mapped into the torso mesh (right). Notice the need for geometric manipulation (left) in order to remove the heart from the torso. With the mesh prepped a LUT approach is used to generate map conductivities into the FEM matrix.
- **Phase 3: Solution Calculation**
  Known values are added to the linear system before an iterative solver is used to compute the solution.

- **Phase 4: Data Mapping**
  The solution is then mapped to the torso and its surface is extracted.

- **Phase 5: Visualization**
  The results are smoothed using the `FairMesh` module and visualized.

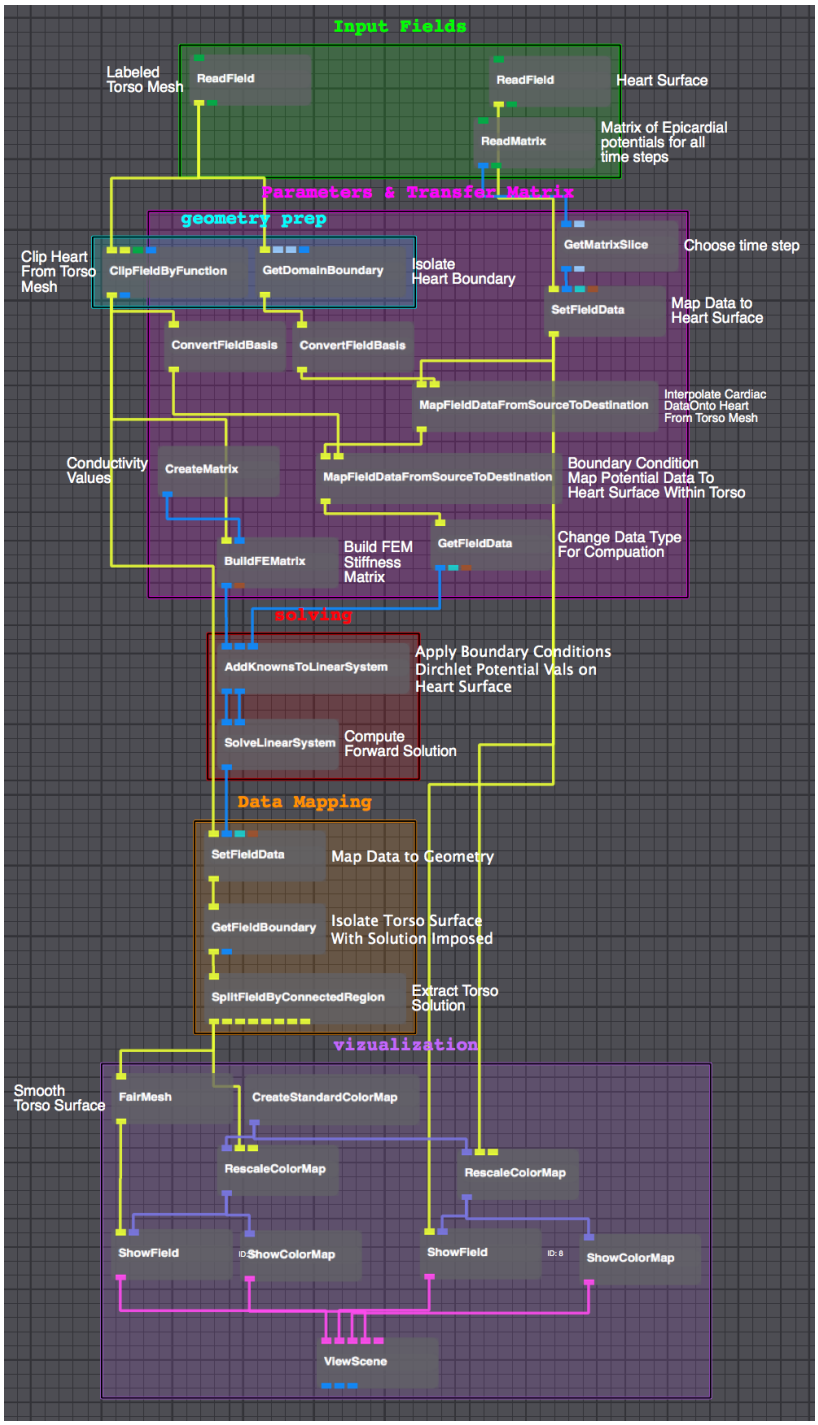

Figure 3.8. **FEM Network Example.** A simple implementation of the finite element method in SCIRun.

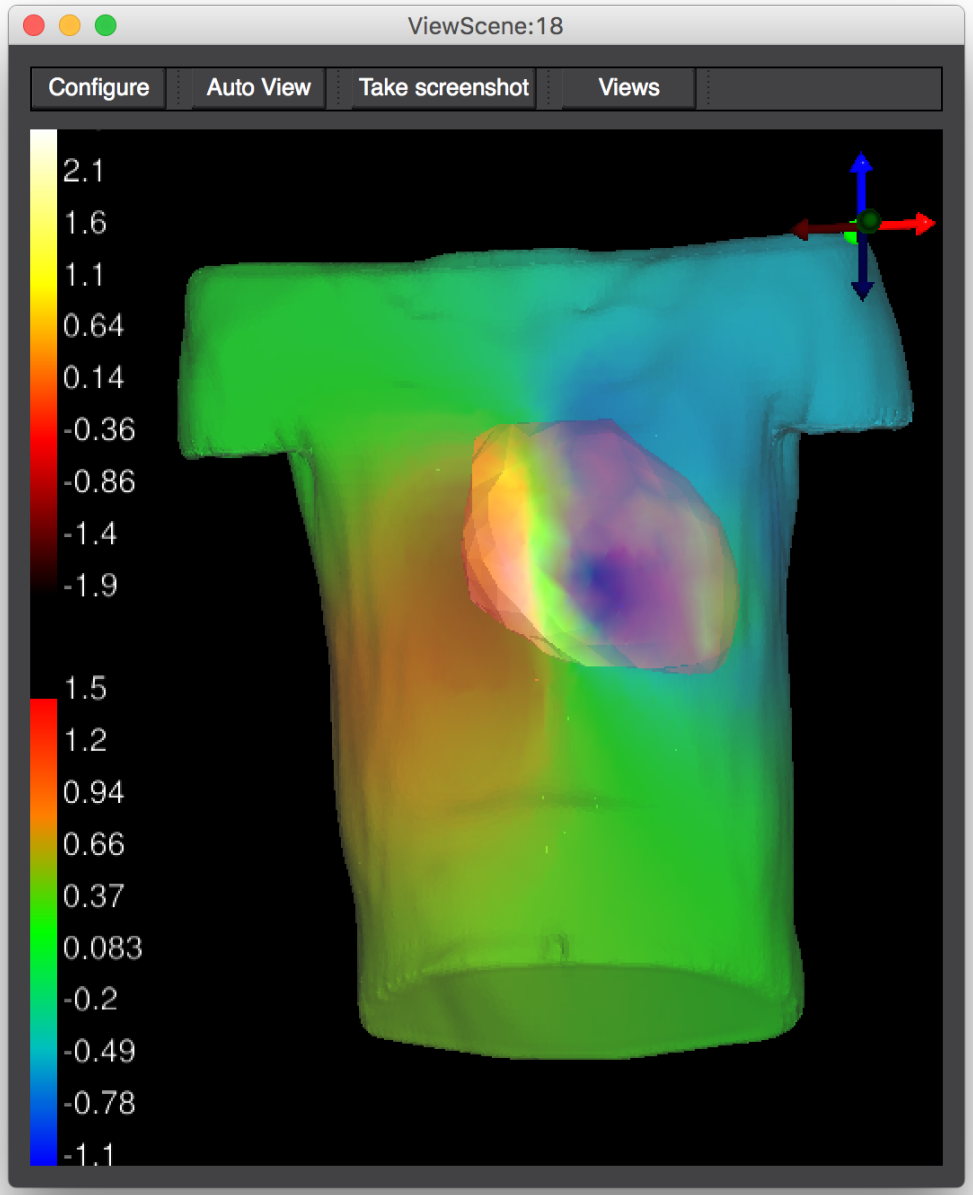

Figure 3.9. **FEM Laplace Solution Example.** Laplace's equation was solved within the torso to project potentials from the heart throughout the torso domain. The torso surface was subsequently extracted and visualized. Different colormaps have been assigned to the heart (source) and the torso (solution) surfaces.

A similar SCIRun network for this example can be found in the Fwd/Inv toolkit at: `[ToolkitPath]/Networks/potential-based-fem/forward_problem.srn5`

3.5.2 Potential Based Forward FEM Simulation: Lead Field Calculation

The following example utilizes two networks in order to produce 1) a lead field matrix with which to solve 2) a traditional $\mathbf{y}(t) = \mathbf{A}\mathbf{x}(t)$ forward solution. Figure 3.10 shows a basic implementation of the finite element method used to calculate a lead field matrix and Figure 3.11 shows how that lead field matrix can be applied to generate a forward solution. These networks also show the complete networks available in the Fwd/Inv toolkit in all their complexity. Input meshes are segmentations that include multiple label masks (including one for the heart volume with a closed epicardium), which require manipulation prior to use.

It is worth noting that the initial network, shown in Figure 3.10, is remarkably similar to the Laplacian Solve network used in the previous section (Figure 3.8). Apart from the additional geometric manipulation steps, that isolate the heart and torso regions, the major difference between these two networks is the definition of source potentials. The Laplacian solution (Figure 3.8) applies a potential field to the nodes along the surface of the heart. That is, it applies a complete data set in which each node has a specified potential value that is representative of the potential measurements along the epicardium at a given time instance. The motives for using the heart potentials as a source is to be able to interpolate the response of the torso to potentials measured on the cardiac surface. The goal of the lead field matrix creation approach (Figure 3.10) is different. The lead field matrix aims to define the relationship between the source and the solution spaces in global terms – not just as a response to potentials during a single step in time. This global relationship is defined by identifying the influence that each source node has on the entire system – similar to the generation of a transfer matrix in the BEM approach. First, a single source node is "activated" (given a value of 1) while the other nodes are "inactive" (given a value of 0). The Laplacian is computed via an FEM approach, producing a solution vector that defines the influence of the active node on the system. The data makes up one column of the lead field matrix. A second node is then activated while all others are left inactive, creating another column in the lead field matrix. This process is repeated for every source node and the results are collected producing an N x M matrix, where N is the number of nodes defining the source, and M is the number of nodes in the complete solution space.

Because we are computing the lead field matrix separately from the actual forward computation, a 6th phase is added to the process in which we save out the data field. A practical implementation of data collection when executing the lead field matrix generation network is the need to:

1. Set the three `CollectMatrices` settings to "Column", "Append", and "PostPend"
2. Clear any previous data in the `CollectMatrices` module using the "Clear Output" button in the module UI.
3. Set the `GetMatrixSlice` module at 0 before pressing play.

Once a lead field matrix is generated and saved, it can be used in a relatively simple network to generate a forward solution for any set of data points that are applied to the heart (Figure 3.11). Notice that in Phase 2 (Parameter Manipulation & Transfer Matrix Computation) no transfer matrix is computed. This is because the lead field acts as the transfer matrix, which is now one of the Input fields. As one would expect, comparing the solution of the Lead Field Solution (Figure 3.12) to those of the Laplacian Solution (Figure 3.9) – taken at the same time step – yields very similar results with only slight discrepancies in the upper range of the matrix.

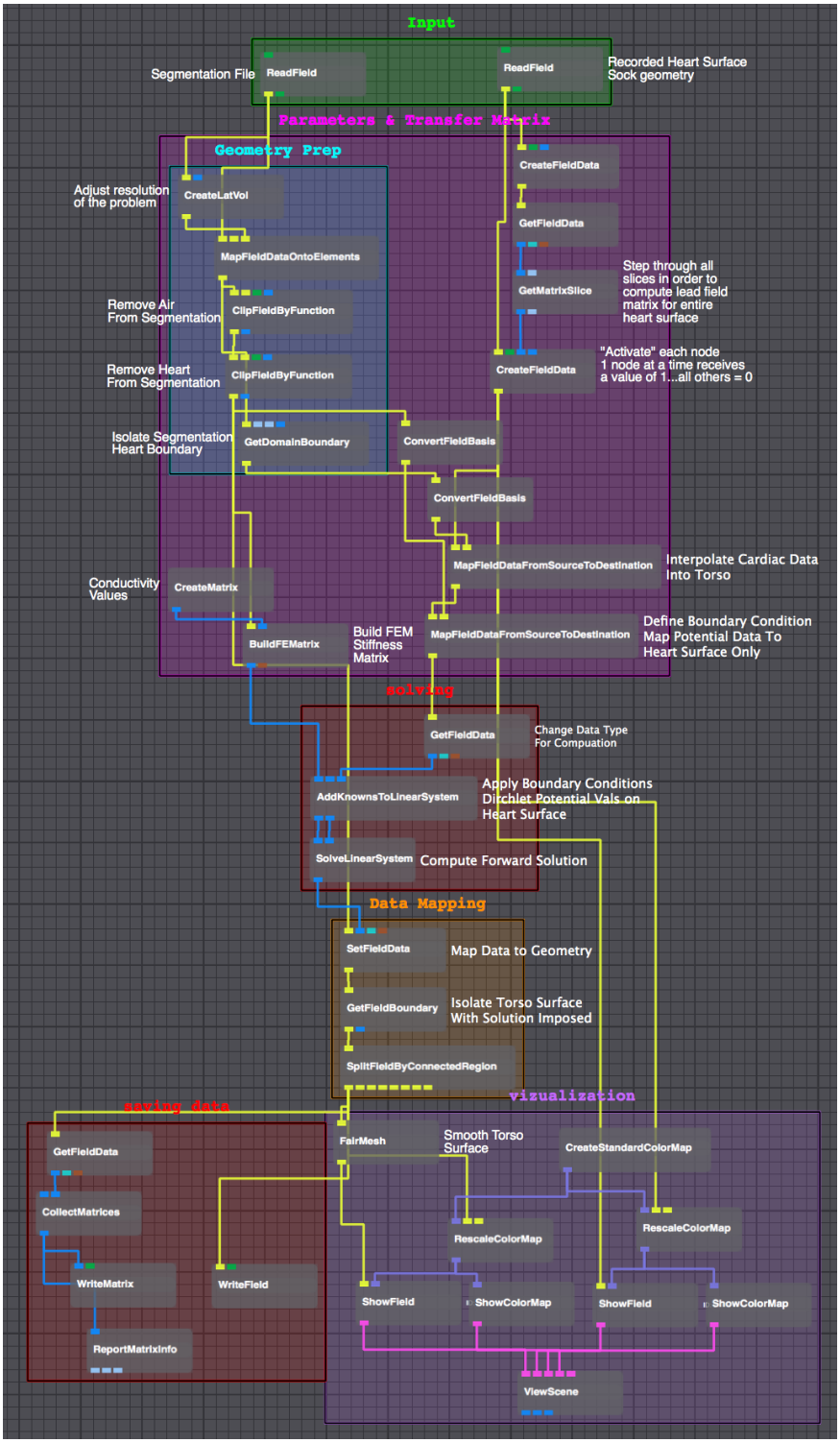

Figure 3.10. **FEM Lead Field Generation Example.** A network that demonstrates the creation of a lead field matrix in SCIRun.

The lead field matrix provided was generated by the network `[ToolkitPath]/Networks/potential-based-fem/make_lead_field_matrix.srn5`.

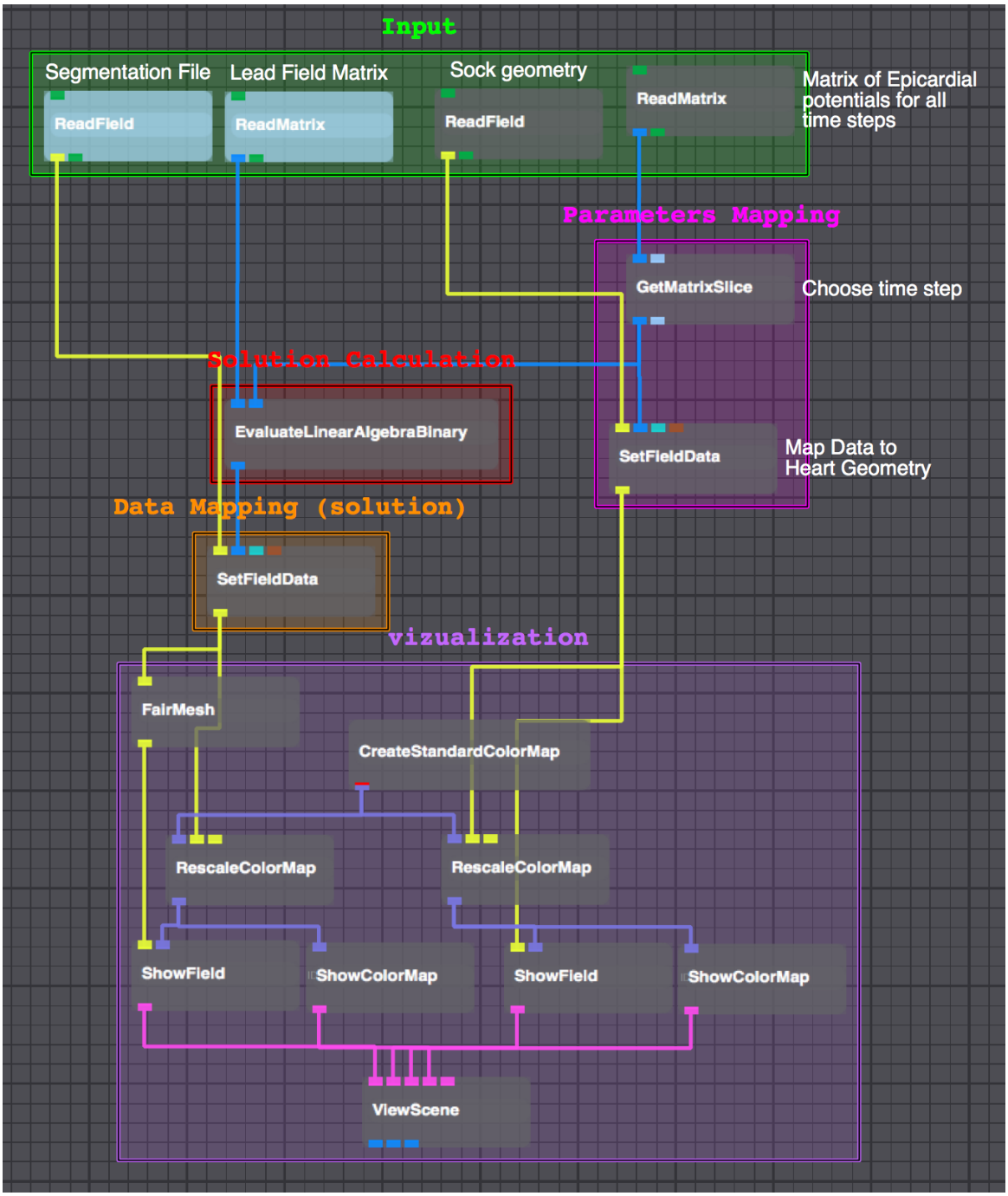

Figure 3.11. **FEM Lead Field Solution Example.** A network that demonstrates how to use a lead field matrix to solve the cardiac forward problem.

The FEM solution with lead field matrix provided was generated by the network `[ToolkitPath]/Networks/potential-based-fem/...Forward_problem_with_lead_field_matrix.srn5`.

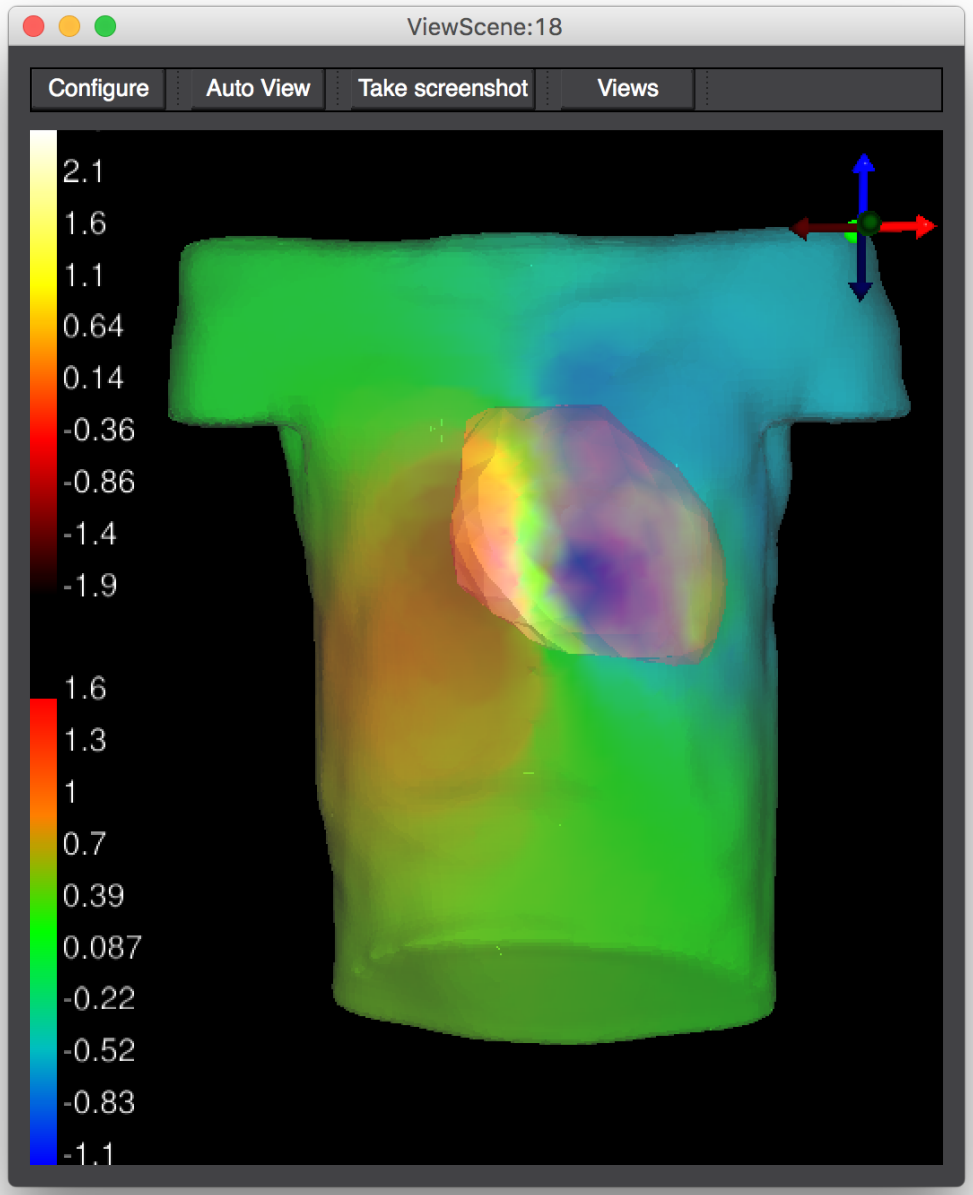

Figure 3.12. **FEM Lead Field Solution Example.** The product of Figures 3.10 and 3.11 the lead field solution produces similar results to those produced by the Laplacian solution (Figure 3.9) when considering the same time step.

3.5.3 Activation Based Forward FEM simulation

As stated above, the activation-based model is incomplete and under review. If the reader has any practical experience and/or would like to assist in the development and implementation of this network, please contact us using the contact information provided in the introduction.

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 Descriptions of the Inverse Solution Methods Implemented in SCIRun

4.2.1 Tikhonov Regularization

As described in Sec. 2.3.2, Tikhonov regularization is a classical inverse method that solves the following least squares problem:

$$min_X \|P(Y - AX)\|_2^2 + \lambda^2 \|RX\|_2^2, \tag{4.1}$$

where Y are the measured ECG potentials, A is the forward matrix, X are the unknown potentials on the heart, λ is the regularization parameter, R is a regularization matrix and P is the sensor covariance matrix. All these variables can be found as inputs or outputs of the module `SolveInverseProblemWithTikhonov`, shown in Figure 4.1. **Inputs:**

1. Forward Matrix ($A \in \Re^{N,M}$)
2. Weights in Source Space ($R \in \Re^{L,M}$ or squared $R^2 \in \Re^{M,M}$ o)
3. Measured Potentials ($Y \in \Re^{N,T}$)
4. Weights in Sensor Space ($P \in \Re^{F,N}$ or squared $P^2 \in \Re^{N,N}$),

Figure 4.1. Revised Tikhonov module: `SolveInverseProblemWithTikhonov`.

Outputs:

1. Inverse Solution ($X \in \Re^{M,T}$)
2. Regularization Parameter (λ)
3. Regularized Inverse ($G \in \Re^{M,N}$)

An example of the usage of the module `SolveInverseProblemWithTikhonov` can be found in the network "potential-based-inverse/tikhonov-inverse.srn5", which is shown in Figure 4.3. This example network is composed of four main blocks:

- **Loading Data (green):** These are the modules that load the data into SCIRun.
- **Forward Solution (white):** These modules compute simulations of ECG potentials that would be measured on the body surface.
- **Tikhonov Inverse (black):** This module computes the Tikhonov inverse solution.
- **Visualization (purple):** These modules provide visualization of the solution and the ground truth.

Options and Modes of Operation

The `SolveInverseProblemWithTikhonov` module allows for multiple modes of operation that provide different functionalities and computational advantages. These modes of operation, can be found in the two panels of the module user interface, shown in Figure 4.2. The options within each of the two panels are described in the following:

Problem Description:

There are two formulations for the solutions of least squares problem in Equation 4.1. These are the overdetermined formulation:

$$\hat{x} \quad = \quad \left(A^T P^T P A + \lambda^2 R^T R\right)^{-1} A^T P^T P Y, \tag{4.2}$$

and the underdetermined:

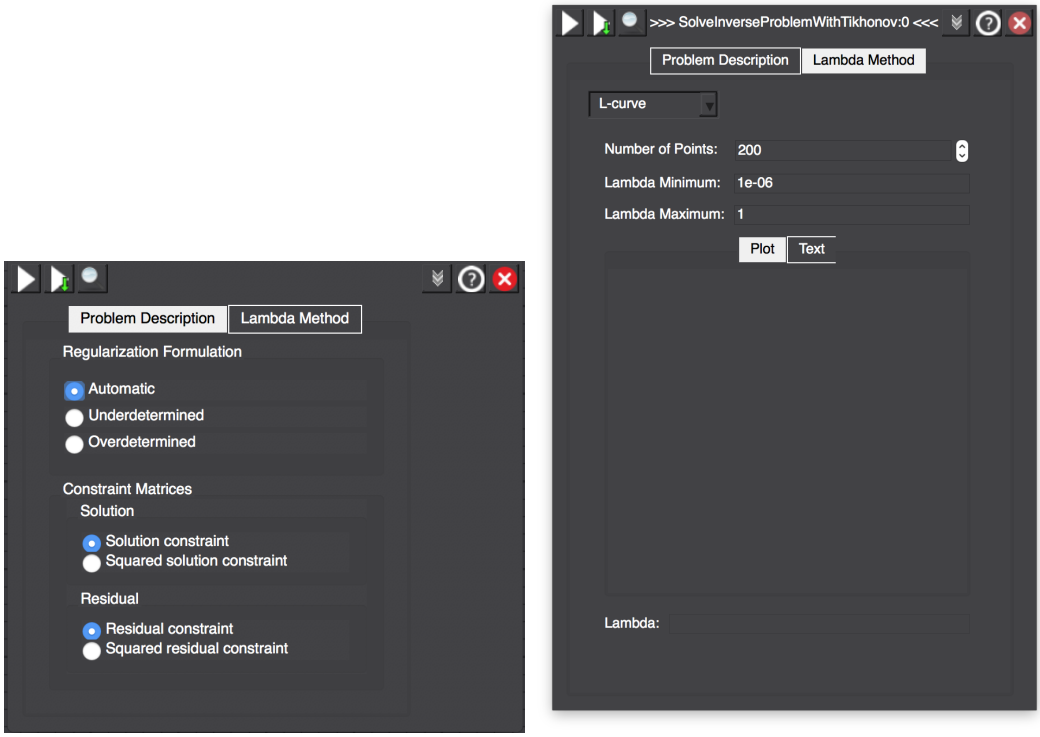

Figure 4.2. GUI from revised Tikhonov module. Left: problem description. Right: Lambda selection method `SolveInverseProblemWithTikhonov`.

$$\hat{x} \quad = \quad (R^T R)^{-1} A^T \left(A (R^T R)^{-1} A^T + \lambda^2 (P^T P)^{-1} \right)^{-1} Y. \tag{4.3}$$

The difference between these two formulations is computational. It can be observed from equations 4.5 and 4.3 that the size of the inverse matrix that needs to be computed is N in the overdetermined case and M in the underdetermined. Thus, when the number of ECG measurements is smaller than the number of sources on the heart ($N < M$), it is computationally desirable to use Equation 4.5. On the other hand, if the number of ECG measurements is larger than the number of sources ($N > M$), it is preferable to use Equation 4.3. `SolveInverseProblemWithTikhonov` is prepared to use either formulation upon request of the user or choose it automatically based on the size of the forward matrix. To choose the desired option, the user must select the appropriate radial button in the "Regularization Formulation" option within the "Problem Description" panel.

The "Problem Description" panel also allows to modify how the source and sensor weight matrices (R and P) are defined. As can be observed in equations 4.5 and 4.3, these two matrices appear in quadratic form. For this reason, many users might prefer to define them in the quadratic form (i.e. $R^2 = R^T R$ and $C^2 = C^T C$). `SolveInverseProblemWithTikhonov` allows users to change the default definition of these input matrices to the squared form by selecting the appropriate radial buttons in the "Constraint Matrices" options within the "Problem Description" panel.

Lambda Method:

The solutions to the Tikhonov regularization problem depend on the selection of the regularization parameter (λ). There are multiple approaches in the literature that allow for its selection. `SolveInverseProblemWithTikhonov` is currently implemented so that the user can choose between a direct entry, selection using a slider and the L-curve method, described in Sec. 2.3.2. These methods can be found by selecting the appropriate option in the drop-down menu within the "Lambda Method" panel.

4.2.2 Tikhonov SVD Method

The Tikhonov SVD method effectively solves the same least squares problem as the Tikhonov method described above. The difference between these two implementations is computational. The `SolveInverseProblemWithTikhonovSVD` module uses the following formulation to solve the least-squares problem:

$$\hat{X} = \sum_{k=1}^{K} \frac{\sigma_k}{\lambda^2 + \sigma_k^2} v_k u_k^T Y, \tag{4.4}$$

where Y are the ECG measurements, X are the unknown potentials on the heart, λ is the regularization parameter and σ_k, u_k and v_k are the singular values, left and right singular vectors of the forward matrix A and K is its rank.

The `SolveInverseProblemWithTikhonovSVD` module is shown in Figure 4.6. The inputs and outputs of this module are a superset of `SolveInverseProblemWithTikhonov`. The new additions permit the user to use pre-computed singular values and vectors of the forward matrix. **Inputs:**

1. Forward Matrix ($A \in \Re^{N,M}$)
2. Weights in Source Space ($R \in \Re^{L,M}$ or squared $R^2 \in \Re^{M,M}$ o)
3. Measured Potentials ($Y \in \Re^{N,T}$)
4. Weights in Sensor Space ($P \in \Re^{F,N}$ or squared $P^2 \in \Re^{N,N}$)
5. Left Singular Vectors ($U \in \Re^{N,N}$)
6. Singular Values ($S \in \Re^{K,K}$ or in vector form $s \in \Re^{K,1}$)
7. Right Singular Vectors ($V \in \Re^{M,M}$),

Outputs:

1. Inverse Solution ($X \in \Re^{M,T}$)
2. Regularization Parameter (λ)
3. Regularized Inverse ($G \in \Re^{M,N}$)

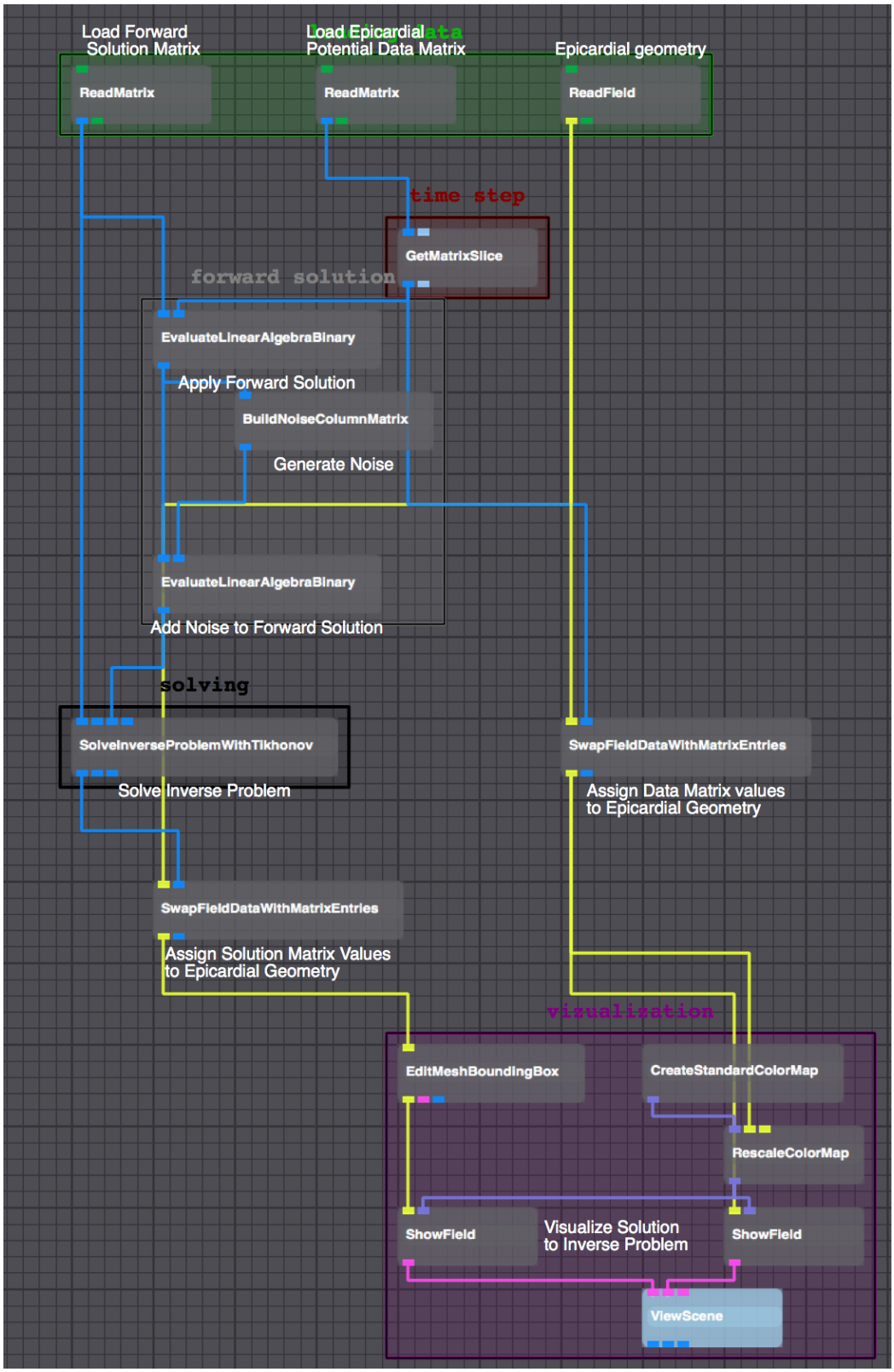

Figure 4.3. The SCIRun network for the Tikhonov inverse solution example.

Figure 4.4. Revised TikhonovSVD module: `SolveInverseProblemWithTikhonovSVD`.

Options and Modes of Operation

The options allowed in SolveInverseProblemWithTikhonovSVD include the use of a pre-computed singular value decomposition of the forward matrix and the method to select the regularization parameter λ.

Pre-computed Singular Value Decompositions

When needed, SolveInverseProblemWithTikhonovSVD automatically computes a Singular Value Decomposition of the input forward matrix using sub-routines found in the Eigen library. However, the module is also prepared to use a pre-computed singular value decomposition. This option is selected by connecting ALL the input ports of the left and right singular vectors and singular values (ports 5, 6 and 7). In this case, the algorithm skips the computation of the SVD and uses the provided matrices.

Lambda Method

The solutions to the TikhonovSVD regularization problem depend on the selection of the regularization parameter (λ). There are multiple approaches in the literature that allow for its selection. SolveInverseProblemWithTikhonovSVD is currently implemented so that the user can choose between a direct entry, selection using a slider and the L-curve method, described in Sec. 2.3.2. These methods can be found by selecting the appropriate option in the drop-down menu within the "Lambda Method" panel as done in SolveInverseProblemWithTikhonov (see Figure 4.2).

4.2.3 Truncated SVD Method (TSVD)

The truncated SVD method solves the classical least-squares problem with a low-rank approximation of the forward matrix. Solutions to this method require the decomposition of the forward matrix A into its left singular vectors U, the right singular vetors V and the corresponding singular values S. Then, these are computed following the formula:

$$\hat{X} = \sum_{k=1}^{Q} \frac{1}{\sigma_k} v_k u_k^T Y, \tag{4.5}$$

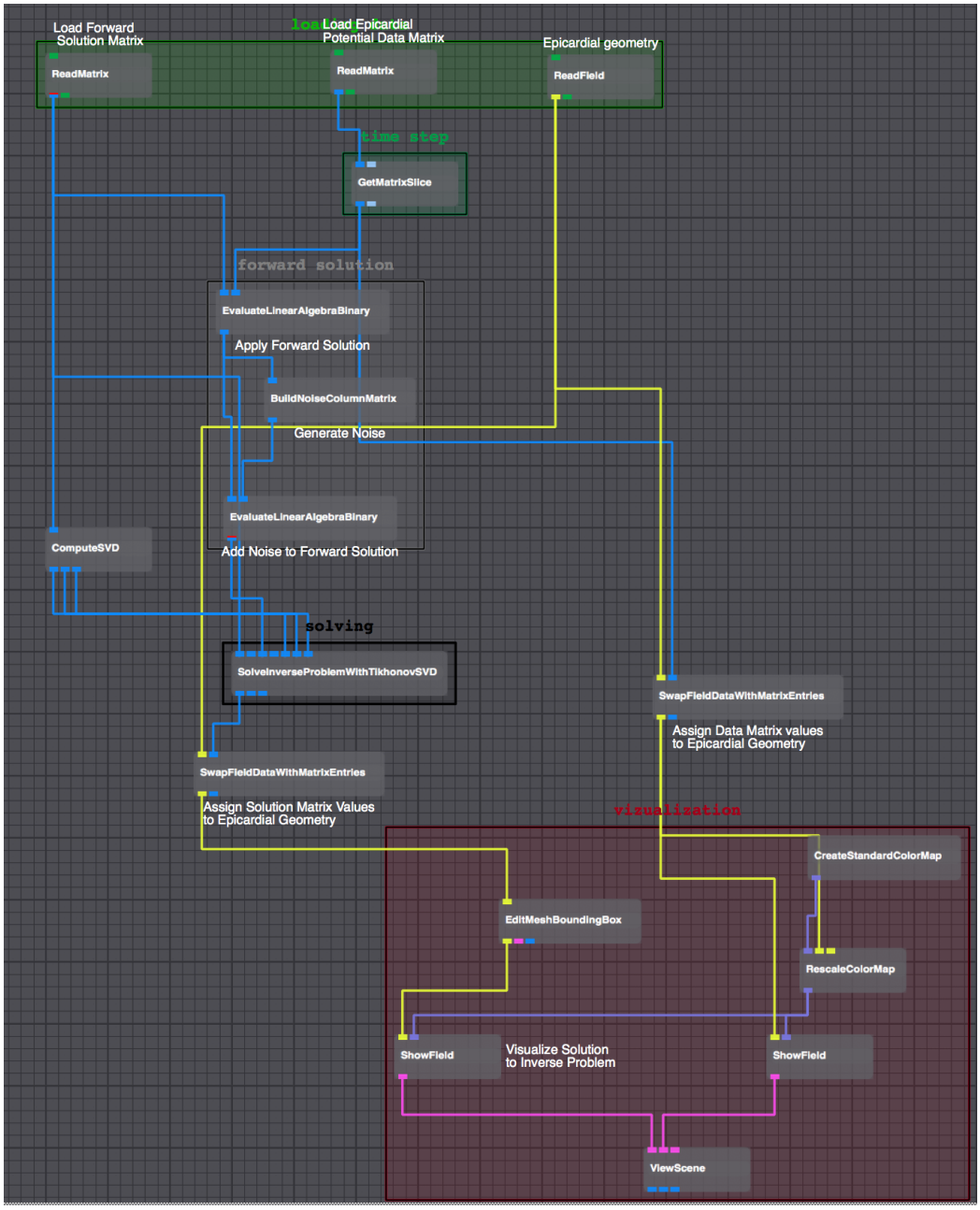

Figure 4.5. The SCIRun network for the TikhonovSVD inverse solution example.

Figure 4.6. Revised TSVD module: `SolveInverseProblemWithTSVD`.

where Y are the ECG measurements, X are the unknown potentials on the heart, λ is the regularization parameter and σ_k, u_k and v_k are the singular values, left and right singular vectors of the forward matrix A and Q is the truncation point.

The module `SolveInverseProblemWithTSVD` is shown along its inputs and outputs in **??** The inputs and outputs of this module are a superset of `SolveInverseProblemWithTikhonov`. The new additions permit the user to use pre-computed singular values and vectors of the forward matrix. **Inputs:**

1. Forward Matrix ($A \in \Re^{N,M}$)
2. Weights in Source Space ($R \in \Re^{L,M}$ or squared $R^2 \in \Re^{M,M}$ o)
3. Measured Potentials ($Y \in \Re^{N,T}$)
4. Weights in Sensor Space ($P \in \Re^{F,N}$ or squared $P^2 \in \Re^{N,N}$)
5. Left Singular Vectors ($U \in \Re^{N,N}$)
6. Singular Values ($S \in \Re^{K,K}$ or in vector form $s \in \Re^{K,1}$)
7. Right Singular Vectors ($V \in \Re^{M,M}$),

Outputs:

1. Inverse Solution ($X \in \Re^{M,T}$)
2. Regularization Parameter (λ)
3. Regularized Inverse ($G \in \Re^{M,N}$)

Options and Modes of Operation

The options allowed in `SolveInverseProblemWithTikhonovSVD` include the use of a pre-computed singular value decomposition of the forward matrix and the method to select the truncation point (or regularization parameter) Q.

Pre-computed Singular Value Decompositions

When needed, SolveInverseProblemWithTSVD automatically computes a Singular Value Decomposition of the input forward matrix using sub-routines found in the Eigen library. However, the module is also prepared to use a pre-computed singular value decomposition. This option is selected by connecting ALL the input ports of the left and right singular vectors and singular values (ports 5, 6 and 7). In this case, the algorithm skips the computation of the SVD and uses the provided matrices.

Lambda Method

The solutions to the TSVD regularization problem depend on the selection of the regularization parameter (Q). There are multiple approaches in the literature that allow for its selection. SolveInverseProblemWithTSVD is currently implemented so that the user can choose between a direct entry, selection using a slider and the L-curve method, described in Sec. 2.3.2. These methods can be found by selecting the appropriate option in the drop-down menu within the "Lambda Method" panel as done in SolveInverseProblemWithTikhonov (see Figure 4.2).

4.2.4 Method of Fundamental Solutions

Method of Fundamental Solutions (MFS) is a method to approximate solutions to partial differential equations, such as Laplace's equation used in bioelectricity problems. MFS is similar to BEM in that it is formulated to take into account boundary conditions and solutions, but not volumetric ones. However, MFS is considered meshless, even though points representing the heart and torso regions are needed, because the points do not need tessellation, as do BEM and FEM. In MFS the points representing these regions must be outside the computational domain, *i.e.,* in the myocardium and outside the torso, but as close to the boundary as possible. Also, unlike BEM and FEM methods, computing a forward matrix is not necessary. Instead, the MFS computes the coefficients representing pericardial potential values based on analytical evaluation of the potential distribution and the boundary condition [4, 5]. These coefficients are are usually solved for with Tikhonov regularization [5, 6]. In this toolkit, we implemented MFS as described by Wang and Rudy [5] with the methods to choose the regularization parameters described by Johnston [6]. It is implemented in a python library ('PythonLibrary/MFS_inverse/mfs_inverse.py') that can be called from SCIRun using the InterfaceWithPython module, as shown in 'MSF_inverse_Cage.srn5' and 'MSF_inverse_python.srn5'.

Inputs:

1. Heart boundary ($H \in \Re^{N,3}$)
2. Torso boundary ($T \in \Re^{M,3}$)
3. Heart recording points ($P \in \Re^{F,3}$)
4. Measured Potentials ($Y \in \Re^{K,T}$)
5. Torso Electrodes (C list of K indices)
6. options

Outputs:

1. Inverse Solution ($X \in \Re^{F,T}$)

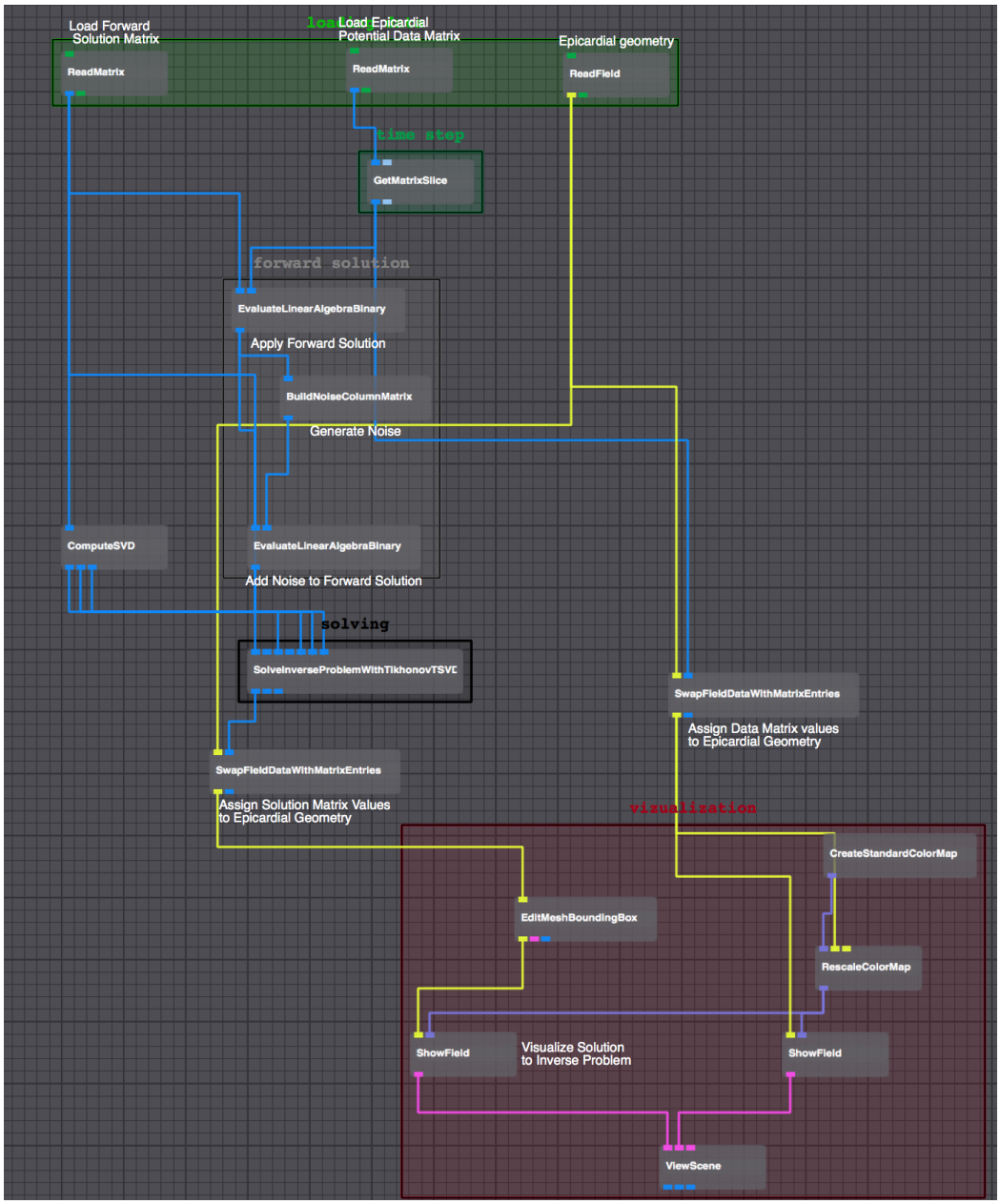

Figure 4.7. The SCIRun network for the TSVD inverse solution example.

2. Regularization Parameter (λ)
3. Regularized Curve

Options and Modes of Operation

Options for the algorithm are the scale method (along the normals or by scaling the point cloud), scale factor, regularization parameter choosing method [6], lambda range, gamma (for some techniques). The implementation has an option to plot the regularization function.

4.2.5 Isotropy Method

The Isotropy Method is an approach that considers temporal correlations to imrove the conditioning of the inverse taken. In SCIRun we have implemented this approach with a combination of modules and SolveInverseProblemWithTikhonov. Our implementation of the Isotropy Method can be found in the example network "potential-based-inverse/greensite-inverse-IsotropyMethod.srn5", shown in Figure 4.8.

The network is composed of the standard sections of a simulation network to loading data, visualizing and synthesizing the ECG measurements. The specific blocks that correspond to the Isotropy Method are the "Temporal Decomposition", "solving" and "Temporal Reconstruction". Here we describe this blocks in more detail:

1. The **temporal decomposition** block computes the singular value decomposition of the input data, truncates the right singular vectors (corresponding to time) and projects the input data into this lower dimensional space.

2. The **solving** block consists on a Tikhonov module that solves the inverse problem for the input data projected onto the low dimensional temporal space.

3. The **temporal reconstruction** block, uses the truncated right singular vectors from (1) to reconstruct the full temporal behavior of the inverse solutions obtained in the Tikhonov solver.

Options and Modes of Operation

As implemented, the Isotropy Method does not have many options for operation. The most important parameter that needs to be determined by the user is the truncation point of the right singular vectors, which can be accessed in the "SelectSubMatrix" module within the "temporal decomposition" block. The GUI for this module, shown in **??**, allows to select a submatrix from an input matrix. In the case of the isotropy method, the users are only interested in the "end" parameter from the column range selector (lower right entry) since it determined where to truncate the right singular vectors.

This implementation of the Isotropy Method has extra parameters that determine the operation of the Tikhonov inverse. These parameters are specific to the SolveInverseProblemWithTikhonov module and we refer the user to subsection 4.2.1 for more details.

4.2.6 Spline-Based Inverse Method

The spline-based inverse method is another approach that takes into account the temporal characteristics of the signal. In this particular case, it projects the input data onto a 1D manifold characterized with a spline []. The current implementation of this method uses the MATLAB-python interface capabilities as well as the implemented inverse methods. Similarly to the Isotropy Method, the spline-based inverse is composed of three main blocks "temporal decomposition", "inverse solving" and "temporal recontruction". The main difference concerns the temporal decomposition and reconstruction steps, which now are non-linear and involve the estimation fo a 1D manifold. In the following, we will describe these three blocks:

Manifold Estimation:

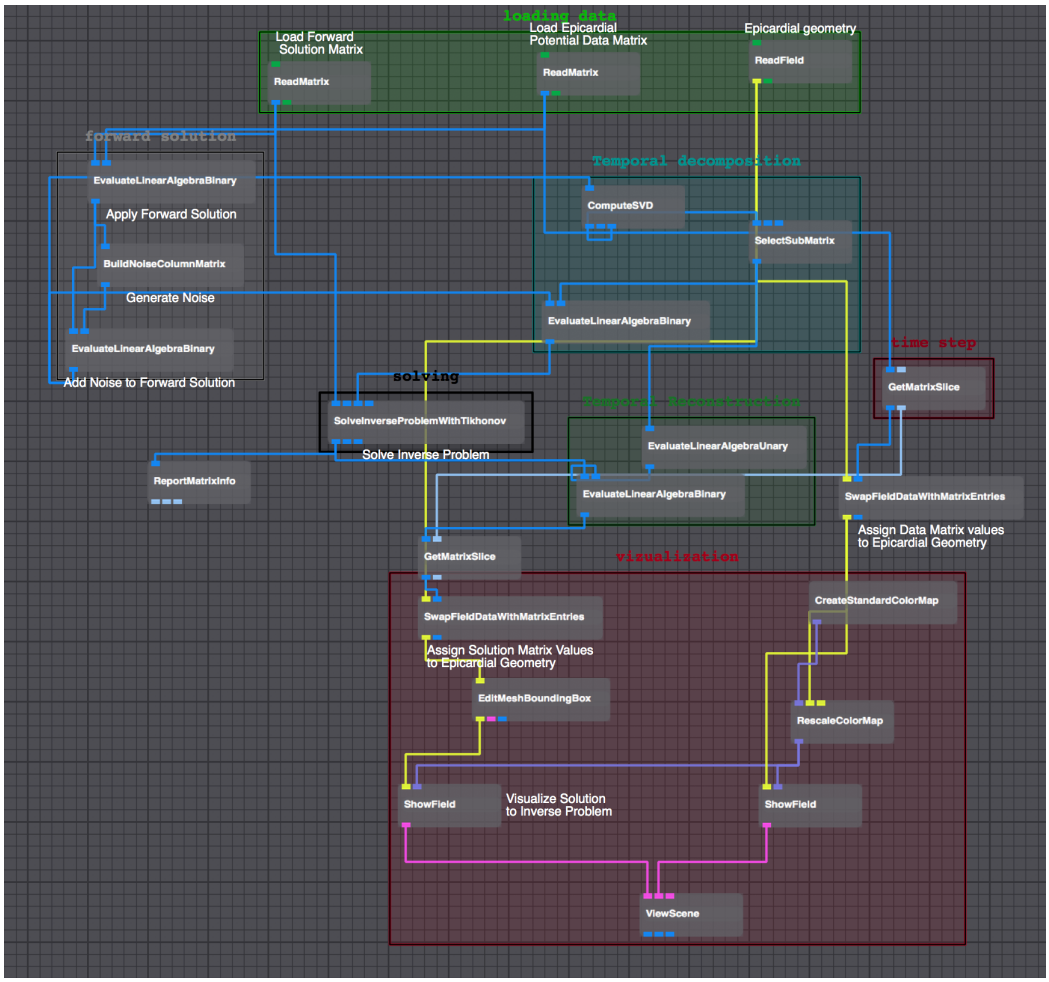

Figure 4.8. The SCIRun network for the Isotropy Method inverse solution example.

```
% This code implements the inverse solutions pipeline presented in
% the paper:
%         Erem, Coll-font, Martinez Orellana - 2013 -
%         Using Transmural Regularization and Dynamic Modeling
%         for Non-Invasive Cardiac Potential Imaging of
%         Endocardial Pacing Sites with Imprecise Thoracic Geometry.
%
% Inputs
%   i1 - <N,T>double - measured potentials on the torso.
%   i2 - <N,M>double - forward matrix.
%   i3 - <L,M>double - regularization matrix.
%   i4 - <3,1>int - regularization constant params.
%                   i4(1) - log10 min lambda.
%                   i4(2) - log10 max lambda
%                   i4(3) - num lambda.
```

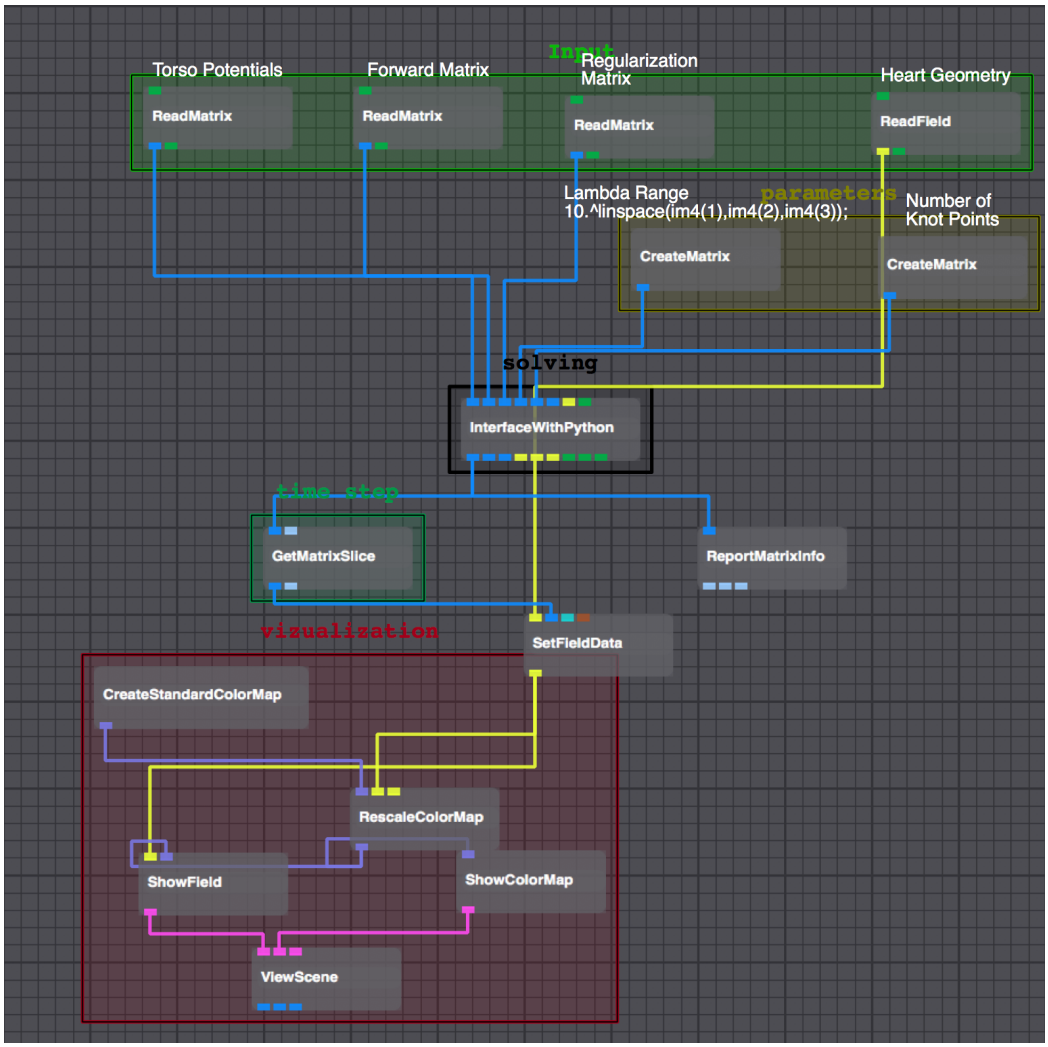

Figure 4.9. The SCIRun network for the Spline-Based Method inverse solution example.

```
% Outputs
%   o1 - <M,T>double - estimated heart potentials.
```

Inverse Solution:

4.2.7 Non-Decreasing Inverse Method

The Non-Decreasing Inverse Method estimates the transmembrane potentials by solving a constrained Tikhonov problem. Thus the function to be optimized over is a standard

Tikhonov problem but restricts the solutions to always be non-decreasing. Thus, the problem to be solved is:

$$\begin{aligned} \min_{x(t),t=1\ldots T} & \|y(t) - Ax(t)\|_2^2 + \lambda * \|Rx(t)\|_2^2 \\ s.t. & \\ & x(1) >= minB \\ & x(t+1) >= x(t), \;\; t = 1 \ldots T-1 \\ & x(T) <= maxB \end{aligned} \tag{4.6}$$

Where $minB$ and $maxB$ are the minimum and maximum bounds for the TMP. For memory efficiency, this problem is implemented with an ADMM solver.

This model assumes that the electrical sources of the heart model the transmembrane potentials on each node of the geometry. The inputs and ouputs of the MATLAB implementation can be seen here:

```
%% MESSNARZ INVERSE SCRIPT FOR SCIRUN
%
%           (...)
% INPUT:
%  - A - <N M>double - forward operator of the linear system.
%  - R - <H,M>double - regularization matrix.
%  - ECG - <N,T>double - Body surface measurements.
%  - lambda - double - regularization term.
%  - initialx - <M,T>double - initial guess for the TMP
%  solution.
%  - rho - double - augmented term weight.
%  - min_r - double - stopping criteria for the primal
%  residual.
%  - min_s - double - stopping criteria for the dual residual.
% - margin - <2,1>double - minimum and maximum bounds for
% the transmembrane potentials.
%  - verbose - boolean - print ADMM iteratinons information.
%
% OUTPUT:
%  - xk - <M,T>double - estimated TMP.
%  - zk - <M,T+1>double - estimated slack variable.
%           (...)
%
```

Inputs:

1. Forward Matrix ($A \in \Re^{N,M}$)
2. Regularization matrix ($R \in \Re^{F,N}$)
3. Measured Potentials ($Y \in \Re^{N,T}$)
4. Regularization parameter ($\lambda \in \Re$)

5. Initial guess for the transmembrane potentials ($X_0 \in \Re^{N,1}$)

6. ADMM parameters: weight of augmented lagrangian, stopping paramter for primal and dual objectives ($\rho, min_r, min_s \in \Re^{+}$)

7. Lower and upper bounds (margin) of the transmembrane potentials ($marg \in \Re^{2}$)

Outputs:

1. Heart potentials ($xk \in \Re^{M,T}$)

The resulting potentials have sharp increases of potentials similar to the typically observed in TMPs for most of the nodes on the heart geometry. In general, the final solution is insensitive to the initial guess but it will affect the time of convergence. The script is set up such that the initial guess for the first lambda is a simple ramp, after each initial guess is the final result of the previous lambda. The decision of the correct lambda is done automatically using an L-corner detection. This algorithm is implemented as the SCIRUN network shown in **??**

NOTE: This code requires the software package reguTools (http://www.imm.dtu.dk/ pcha/Regutools/) to be incorporated in the default path from MATLAB.

4.2.8 Activation-Based Method

The activation-based inverse method solves the inverse problem in electrocardiography through the estimation of the activation times (*i.e.* the depolarization times) on the heart. This inverse method assumes that the sources on the heart are modeled with a potentials over time at each node are parameterized by the time of activation. This assumption allows the inverse method to resolve the inverse problem by estimating the activation times at every node on the heart geometry.

The inverse problem being solved is thus a non-linear optimization problem for which there is no closed-form solution. The implementation of the activation-based inverse in SCIRun is done with a Gauss-Newton iterative method that solves:

$$min_X \|Y - A\Phi(\tau, \omega)\|_2^2 + \lambda \|R\Phi(\tau, \omega)\|_2^2, \tag{4.7}$$

where $\Phi(\cdot)$ is the pre-defined temporal waveform at each node and is paramaterized by τ and ω, which correspond to the time of activation and the slope of the depolarixzation wave respectively. Y are the measured body surface potentials, A the forward matrix and R the regularization matrix.

The inputs and outputs of this inverse method can be observed in the "InterfaceWithPython" module in the network example shown in Figure 4.10 and in the header of the MATLAB implementation:

```
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
```

Inputs:

1. Forward Matrix ($A \in \Re^{N,M}$)
2. Measured Potentials ($Y \in \Re^{N,T}$)
3. Regularization matrix ($R \in \Re^{F,N}$)
4. Initial guess for the activation times ($\tau \in \Re^{N,1}$)
5. Regularization parameter ($\lambda \in \Re$)
6. Activation waveform transition width ($\omega \in \Re^{+}$)
7. Convergence parameter ($\epsilon \in \Re^{+}$)

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.

Outputs:

1. Activation times ($\tau \in \Re^{M,1}$)

The only output of the method is the inverse solution in the form of an array of activation times.

4.2.9 Wavefront-Based Potential Reconstruction (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 inverse method assumes a source model characterizing the extracellular potentials on the heart surface and introduces a regularization term that imposes them to be similar to a pre-specified shape.

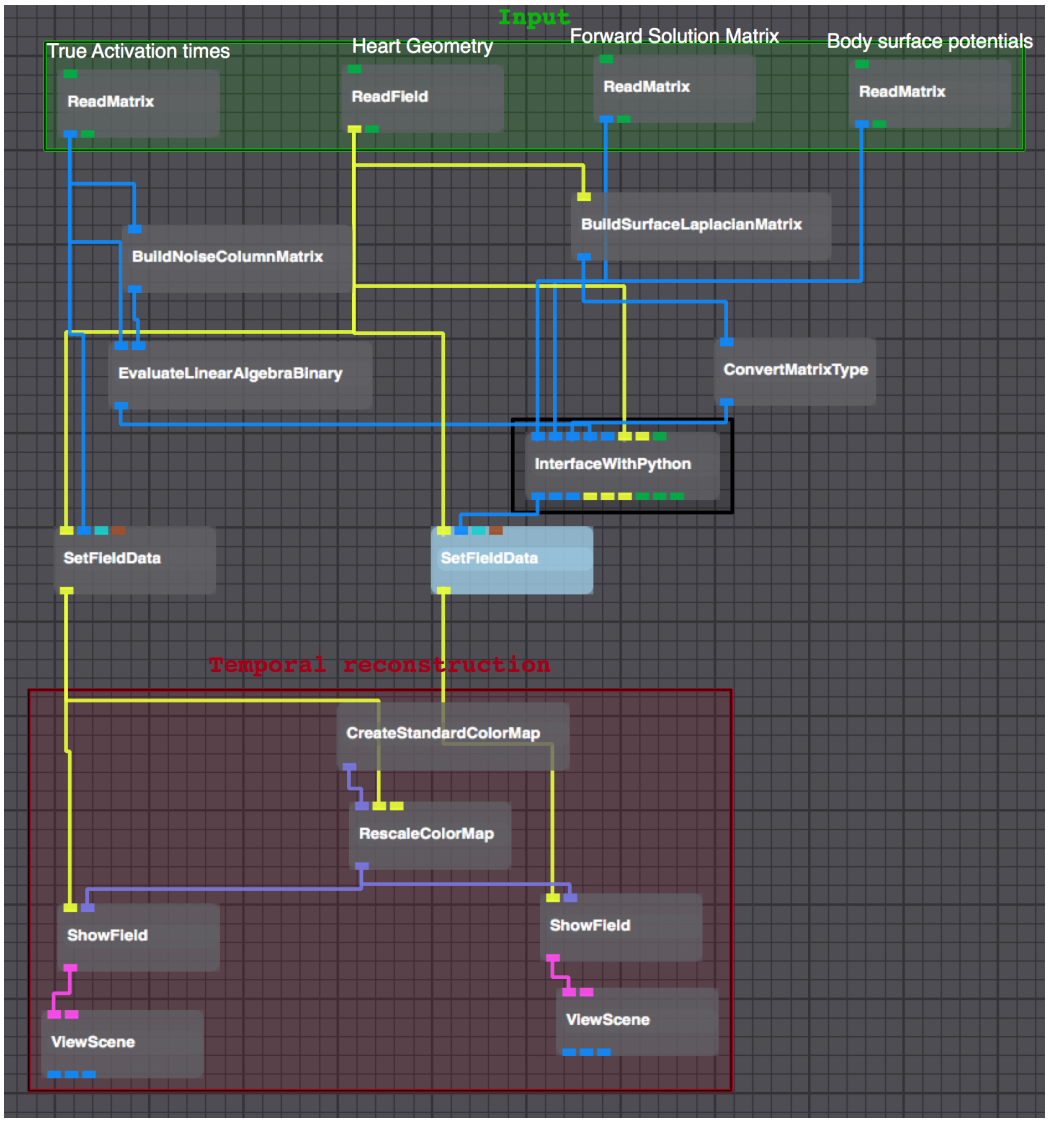

Figure 4.10. The SCIRun network for the Activation-Based Method inverse solution example.

The inputs and outputs of this inverse method can be observed in the "InterfaceWithPython" module in the network example shown in Figure 4.11 and in the header of the MATLAB implementation:

```
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
```

Inputs:

1. Forward Matrix ($A \in \Re^{N,M}$)
2. Measured Potentials ($Y \in \Re^{N,T}$)
3. Heart geometry (struct containing nodes and triangles)
4. First and last nodes to activate ()

Outputs:

1. forward WBPR solution ($X_f \in \Re^{M,T}$)
2. backward WBPR solution ($X_b \in \Re^{M,T}$)

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.

NOTE: This code requires the software package reguTools (http://www.imm.dtu.dk/ pcha/Regutools/) to be incorporated in the default path from MATLAB.

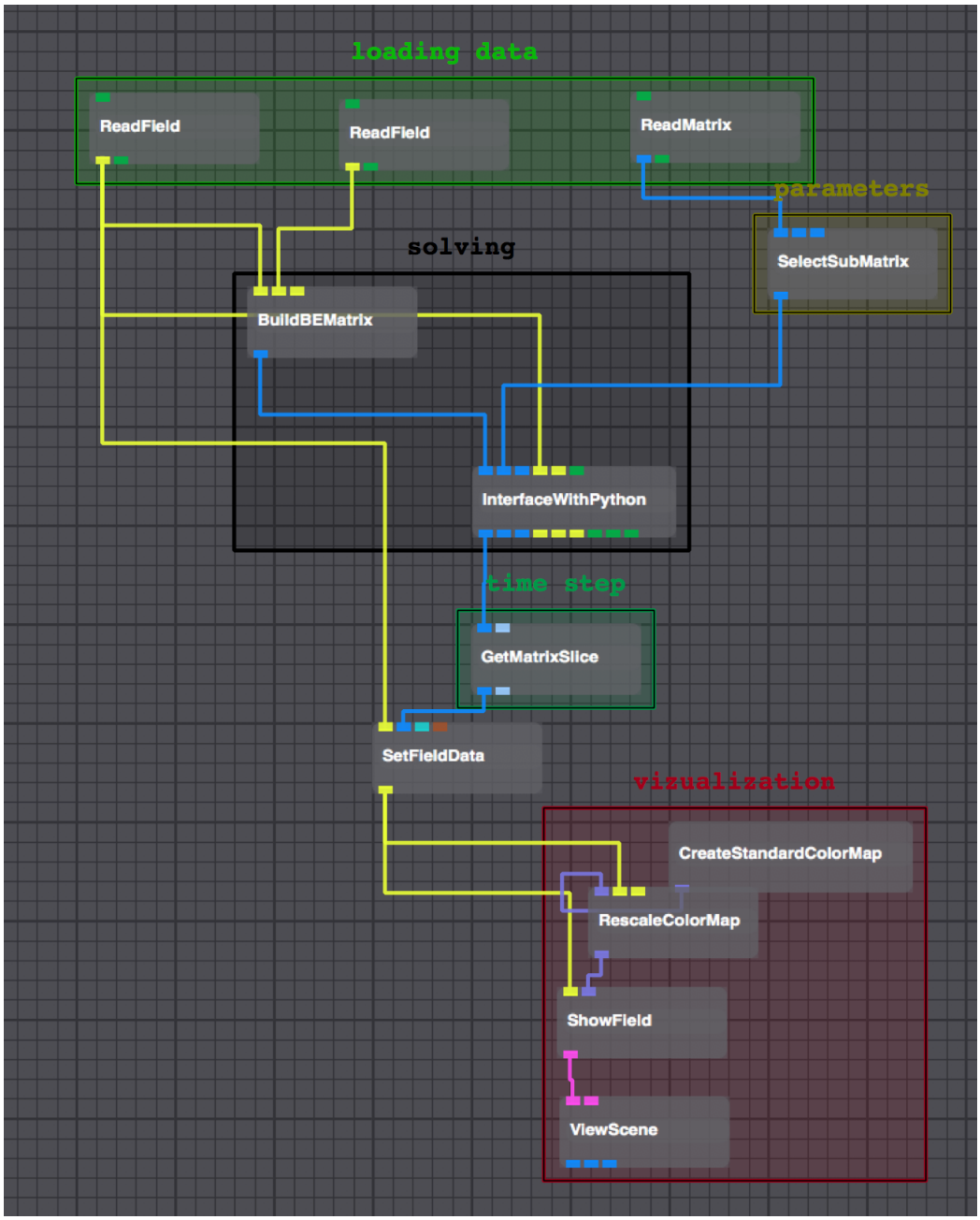

Figure 4.11. The SCIRun network for the WBPR Method inverse solution example.

Bibliography

[1] A. van Oosterom and T.F. Oostendorp. ECGSIM: an interactive tool for studying the genesis of QRST waveforms. *Heart*, 90(2):165–168, February 2004.

[2] R. M. Gulrajani. The forward problem of electrocardiography: from heart models to body surface potentials. In *EMBS, 1997. Proceedings of the 19th Annual International Conference of the IEEE*, volume 6, pages 2604–2609 vol.6, Oct 1997.

[3] R.C. Barr, M. Ramsey, and M.S. Spach. Relating epicardial to body surface potential distributions by means of transfer coefficients based on geometry measurements. *IEEE Trans. Biomed. Eng.*, 24:1–11, 1977.

[4] R. Mathon and R. Johnston. The approximate solution of elliptic boundary-value problems by fundamental solutions. 14(4):638–650, 1977.

[5] Yong Wang and Yoram Rudy. Application of the method of fundamental solutions to potential-based inverse electrocardiography. 34(8):1272–1288, 08 2006.

[6] Peter R. Johnston. Accuracy of electrocardiographic imaging using the method of fundamental solutions. *Computers in Biology and Medicine*, 102:433 – 448, 2018.