Using Distmesh as a Mesh Generating Tool for EIT

Solomon Fung, Andy Adler, Adrian D.C. Chan
Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
sfung@connect.carleton.ca

Abstract. We present a collection of automatic mesh generation software tools for electrical impedance tomography (EIT) finite element models (FEMs). Creating an image in EIT involves retrieving conductivity distribution from surface electrode measurements. The initial forward problem of this process requires estimates of electric potential according to the current sources. With non-trivial geometries, this task is difficult analytically and demands a numerical method such as FEM. The areas nearest to electrodes have the highest current density and are the most significant measurement points. For our FEM model, these areas should have the greatest mesh density for the purposes of usefulness and accuracy. The research focuses on Distmesh, an open source MATLAB mesh generator, for creating mesh shapes with non-uniform mesh densities for EIT software. The inherent simplicity of the Distmesh code was the rationale for its evaluation. A mesh shape is defined from specifying its signed distance function and its mesh size distribution is defined with an element length function. We created circular meshes with mesh refinement towards endpoints of distributed electrodes. In three dimensions, cylinders were produced with mesh refinement towards rings of points for each electrode. Finally, to accommodate organic shapes, meshes for closed contours were created using elliptic Fourier descriptors. Because Distmesh was not designed for efficiency, the generation of well-defined meshes is fairly slow. For the contours, Distmesh forms reasonable meshes but is unable to improve node locations upon successive iterations. Distmesh is not a suitable mesh generator for EIT software unless its code is optimized for speed.

1. Introduction

Electrical impedance tomography (EIT) is an imaging technology where images are constructed from impedance distributions. In contrast with conventional medical imaging technologies, EIT is a low-cost, portable and risk-free solution [1]. It does not involve any ionizing radiation, nor depend on limited materials or novel parts. Periphery electrodes are used to inject current across a medium and measure the resulting electric potential. A reconstruction algorithm relates the boundary values to the internal distribution of conductance or impedance. Several limitations in modern EIT technology such as low spatial resolution and noise sensitivity [2] have prevented its adoption in ordinary clinical use.

The EIT image reconstruction process consists of two parts. First, the current flow and voltage measurements are estimated for a current injection pattern. This is known as the forward model. The second part is the inverse solution, where the forward model is used to calculate the impedance distribution for constructing the image. The geometries used for the forward model should be as accurate as possible by considering irregularities and anisotropy [3]. An analytic solution is appropriate for simple geometries, as for a circle or sphere where current and voltage can be simply...
calculated. However for complex, realistic shapes, a unique and stable solution may not exist. In this case a numerical approximation may provide a solution for this ill-posed problem.

Finite element method (FEM) is a suitable numerical technique for solving impedance for EIT. A field variable normally assumes an infinite number values over a domain. FEM divides the domain into a manageable number of elements, forming a mesh. For each element an interpolation function approximates the value of the field variable. By assembling the element properties and imposing the boundary conditions, a system equation is constructed whose solution describes the behavior of the field variable over the entire domain. FEM is able to handle complex and irregular domains with ease.

Our research involves the use of Distmesh as a mesh generator for EIT models. Distmesh provides a set of MATLAB functions for creating high quality tetrahedral meshes in two or three dimensions. It is capable of accepting different shape boundaries and custom defined mesh densities. The tool was deliberately designed with short, simple code to encourage open source development [4]. Therefore, it is a worthy candidate as a mesh generator for free EIT software. Our research mainly focuses on experimenting with distance and edge length functions to define different shapes and mesh densities appropriate for EIT models.

2. Methods

The primary meshing function of Distmesh is composed of only a few dozen lines of code. At the beginning of the mesh process, the boundary outline is defined by a signed distance function [4]. The signed distanced function calculates the shortest distance between the shape boundary and any point. Its sign is positive for points outside the boundaries and negative for those inside. After the initial randomized distribution of nodes, Distmesh will eliminate nodes outside of the shape according to its sign. The inner space is then triangulated with a Delaunay algorithm. The resulting mesh can be analogously treated as a mechanical truss structure. Its topology is optimized by iterative node displacement and retriangulation to achieve force equilibrium [4]. The iterations halt once node displacements reach a minimum tolerance. A thorough explanation is given in [4].

The Distmesh package includes functions for simple shapes and is capable of producing mesh shapes given a signed distance function. Distmesh accepts an input value that specifies the element edge length in the initial mesh iteration. It also accepts an edge length function that dictates how node distances are modified upon further iterations. The default function is a uniform element size, producing a consistent density of nodes with the same edge length. In our case, we desire more elements at important regions. This increases accuracy since more variable estimates are calculated per unit area. For EIT, the electrode regions are the most significant areas of concern as they are the origins of the applied currents and locations of voltage measurements. With Distmesh, we passed on distance functions to produce mesh shapes and then passed on edge length functions dependant on point distances to refine mesh densities near electrode locations.

2.1. Circular meshes

The simplest distance function with Distmesh creates a circular mesh. A circle with its centre at the origin has a distance function of \( d - r \), with \( d \) being the distance of any point to the origin and \( r \) being the radius of the circle. We produced a circular mesh function including inputs for circular radius, edge length, number of electrodes and electrode length. These function handles were passed on as arguments in Distmesh’s mesh generator. The endpoints of each electrode arc length were used as refinement points and each pair was distributed evenly around the circumference according to the total number of electrodes. Element size changed proportionally to the distance from the refinement points using an edge length function. Optional inputs for this include a maximum and minimum value to control the size extremes for the edge lengths.

2.2. Cylindrical meshes

Distmesh supports \( n \)-dimensional geometries and shapes resulting from set operations of distance functions, allowing us to proceed from a circular mesh to a cylindrical one. A cylinder can be created
by the intersection of a circular distance function with two planes. The circular distance function is identical to the 2D one, since it is independent from the orthogonal \( z \)-axis. Two upper and lower planes are simply \( x-y \) planes of constant \( z \) values.

The ideal electrode arrangement for a cylindrical body is rows of vertically aligned rings [5]. We gave the option of inputting three values giving the number of rows and columns and the number of refinement points per electrode. The points for each electrode were distributed about the boundary of a disc curved along the cylindrical surface. The disc points were rotationally duplicated to create a row of electrode regions and then duplicated with vertical shifts for additional columns.

### 2.3. Meshes using elliptic Fourier descriptors (EFDs)

Circular boundaries are appropriate for experimental modeling but do not accurately depict the irregularities of organic shapes. Thus, we need a mathematical method of describing simple, closed curves in 2D. A series of elliptic Fourier descriptors can efficiently describe these types of contours [6]. EFD has been used for describing and classifying organic shapes in morphological analyses [7]. Joeng and Radke experimented with Fourier descriptors to model sampled slices for medical imaging [8].

The process of elliptic Fourier analysis is described as follows. The varying \( x \) and \( y \) coordinates about the contour are expressed as a periodic function for one complete revolution. These functions are based on a parameter \( t \) which begins at 0 and reaches \( 2\pi \) after one revolution. This parameter does not imply that the contour tracing arm progresses angularly. A Fourier transform of the periodic functions provides a set of harmonic values that can be used to faithfully reconstruct the contour through Fourier series expansion. Elliptic Fourier descriptors are rotationally invariant, making shape description independent of contour orientation.

We used David Thomas’ MATLAB functions of forward and inverse elliptic Fourier transforms for handling shape boundaries. The forward transform accepts a set of boundary outline points and calculates the values for a desired number of harmonics. The inverse transform reconstructs the boundary outline based on provided Fourier harmonic values for the parametric \( x \) and \( y \) expressions. These transforms enable us to convert between elliptic contours and Fourier harmonics, but cannot produce a mesh in Distmesh without first deriving a distance function.

For this distance function, we began with a general distance function for a reference point to any point on the contour given the value for parameter \( t \), which can be thought of as time from 0 to \( 2\pi \). The derivative of this function with respect to \( t \) gives the velocity of the contour point tracing the outline. A change from negative to positive velocity indicates a local minimum distance. Velocities about the contour were sampled at distributed \( t \) intervals and the adjacent negative-to-positive value pairs were selected as brackets for the false position method to find the roots. Each root is a value of \( t \) giving the point of zero velocity, indicating a local minimum distance. Using our general distance function, the distances at these roots were calculated and the lowest value is selected as the minimum distance. However, the sign of the distance values still needs to be resolved for node elimination. This is essentially a point-in-polygon problem. We decided to use a winding number calculation that measures the number of revolutions traversed around a point while tracing a contour line. Any point with a winding number less than one with respect to a shape must be outside that shape. This calculation for our contours is faster and more accurate than MATLAB’s built-in point-in-polygon function. Our mesh densities were uniform as we did not yet pursue electrode refinement points.

Elliptic Fourier descriptors can accurately construct boundaries that adhere to anatomical sections. This capability is valuable for our mesh generation since boundary shape errors adversely affect reconstructed EIT images [9]. Boundaries that are more irregular may require more harmonic descriptors for shape description.
Figure 1. (a) shows an outline formed by connecting points from mouse input. The point is the selected reference point for distance and the contour begins at $t = 0$ at the triangle and proceeds clockwise. (b) is the plot of contour distance from the point as $t$ reaches $2\pi$. (c) is the plot of contour velocity. Zero crossings during increasing velocity indicate $t$ values where a local minimum distance is found. Calculating distances at these $t$ values with (b) gives us a set to compare for smallest distance.

3. Results

The following figures show some of our Distmesh FEM models. Each one was produced by the separate functions previously described. The function producing EFD meshes uses a contour outline formed by user mouse input. Our EFD distance function is then passed onto Distmesh for mesh construction. Using twenty or more harmonics will produce shapes that accurately depict the original outline. Unfortunately, we were not able to reach the default iterative stop condition due to unremitting node displacement. Relaxing the displacement tolerance will comprise mesh quality for dense meshes.

Figure 2. A portion of a circular mesh with refinement towards a pair of electrode endpoints. The entire mesh contains over 10,000 elements with 8 electrode regions.

Figure 3. This close-up image shows the mesh surface of an electrode region with 5 endpoints on a 3D cylinder. This mesh has two rows and six columns of electrode regions (not shown).
4. Conclusion
Using Distmesh, we were able to create a variety of FEM meshes for EIT models. Our shape boundaries were based on providing signed distance functions. The ability of Distmesh to adjust element sizes based on edge length functions permitted meshes with very fine and dense elements near electrode points. We also used elliptic Fourier descriptors to express curved boundary contours. The signed distance function for EFD was based on finding roots of the contour velocity function then calculating the winding number to solve the point-in-polygon problem. Combining our EFD mesh function with a boundary extraction tool for medical images should produce useful FEM meshes for EIT forward models.

Since the design of Distmesh was intended for simplicity and ease of understanding, the code does not perform efficiently or quickly. Useful 2D EIT meshes contain tens of thousands of mesh elements and perhaps millions for 3D meshes. This can take days for Distmesh to produce quality mesh density refinement in circles and cylinders of many elements. Although our EFD mesh function creates reasonable shapes based on a custom outline, Distmesh is unable to make improvements after successive iterations. This may be caused by numerical approximations in our EFD signed distance function.

Without enhancements to the main code, Distmesh is prohibitively slow for producing complex meshes with many point refinements. Therefore it is not an ideal solution for an EIT meshing tool. Relaxing displacement tolerances will decrease the number of processed iterations at the expense of lower quality mesh triangulation. Porting a C++ version may perhaps bring substantial improvements in speed. However, this is a much more difficult undertaking than what was done so concisely and comprehensibly in MATLAB. We hope to implement our Distmesh functions in the open source EIT and diffuse optical tomography reconstruction software (EIDORS) for further development and experimentation.

5. Acknowledgements
I would like to thank the Natural Sciences and Engineering Research Council (NSERC) of Canada for their support through an undergraduate research award.

References
[1] Soleimani M 2006 Electrical impedance tomography imaging using a priori ultrasound data \textit{Biomed. Eng. Online} \textbf{5}
[2] Oh S, Tang T and Sadleir R J 2007 Quantitative analysis of shape change in electrical impedance tomography (EIT) \textit{IFMBE Proc.} \textbf{17} 424-27
[3] Bagshaw B, Liston A, Bayford R H, Tizzard A, Gibson A, Tidswell A, Sparkes M, Dehghani H, Binnie C and Holder D S 2006 Electrical impedance tomography of human brain function using reconstruction algorithms based on the finite element method Physiol. Meas. 27 S1-11
[4] Persson P and Strang G 2004 A simple mesh generator in MATLAB SIAM Rev. 46 329-45
[5] Graham B M, Adler A 2007 Electrode placement configurations for 3D EIT Physiol. Meas. 28 S29-44
[6] Kuhl F and Giardina C 1982 Elliptic Fourier features of a closed contour Computer Grapitics and Image Processing 18 236-58
[7] Rohlf F J and Archie J W 1984 A comparison of Fourier methods for the description of wing shape in mosquitos (diptera: culicidae). Syst. Zool. 33 302-317
[8] Jeong Y and Radke R J 2007 Reslicing axially sampled 3D shapes using elliptic Fourier descriptors Med. Image Anal. 11 197-206
[9] Lionheart W R B 1998 Boundary shape and electrical impedance tomography Inverse Problems 14 139-47