EIDORS: Towards a community-based extensible software base for EIT

Andy Adler[1], William R.B. Lionheart[2]
[1] School of Information Technology and Engineering, University of Ottawa, Canada
[2] School of Mathematics, University of Manchester, U.K.

Abstract

EIDORS3D is an open source software suite for image reconstruction in electrical impedance tomography and diffuse optical tomography; designed to facilitate collaboration, testing and new research in these fields. This paper describes recent work to redesign the software structure in order to simplify its use and provide a uniform interface, permitting easier modification and customization. We describe the key features of this software, followed by examples of its use.

Introduction

EIDORS3D is a software suite for image reconstruction in electrical impedance tomography and diffuse optical tomography. The goal is to provide a freely distributable and modifiable software for image reconstruction of electrical or diffuse optical data. Such software facilitates research in these fields by providing a reference implementation against which new developments can be compared, and by providing a functioning software base onto which new ideas may be built and tested. The original EIDORS software (Vaukhonen et al., 2000) is based on software from the thesis of Vaukhonen (1997) It implemented a MATLAB package for two-dimensional mesh generation, solving of the forward problem and reconstruction and display of the images. In order to provide capability to solve 3D reconstruction models, a new project, EIDORS3D, was begun (Polydorides and Lionheart, 2002), based on the software developed for thesis of Polydorides (2002). The EIDORS software packages shared the same numerical foundations; media are modelled using a finite element representation, and the images are reconstructed using regularized inverse techniques.

In the three years since the publication of EIDORS3D, several patterns of use have been noted. Researchers typically download the software, run the provided demonstration examples, and make modifications in the demonstration examples and the software internals to meet their needs. Because of the lack of a modular software structure of EIDORS3D, changes tended to made into the code itself. This resulted in duplicated code which could not easily be *refactored* in order to be contributed back to EIDORS3D. Additionally, recent work has begun to move away from basic reconstruction algorithms, focussing on such issues as mesh generation, electrode modelling, visualization and electrode error detection. This research would be facilitated by using modular components which could be *plugged* into a selection of reconstruction algorithms.

To address these issues, the EIDORS3D software has been completely restructured with the goal of providing an *extensible software base*, designed to support *community* use, modification and contribution. This has been accomplished by the following:

- **Open-source license:** EIDORS3D is licensed under the GNU General Public License. Uses are free to use, modify, and distribute their modifications. All modifications must include the source code, or instructions on how to obtain it. EIDORS3D may be used in a commercial product, as long as the source code for EIDORS and all modifications to it are made available.
- **Sourceforge hosting:** EIDORS3D is hosted by `sourceforge.net` at `http://eidors3d.sf.net`. Software is available for download as packaged released versions (version 2.0 contains the modifications described in this paper), or the latest developments may be downloaded from the CVS version control repository. Sourceforge hosting allows for collaborative development for group members, while permitting read-only access to everyone. In order to become a member of the developer group, new contributors should contact the authors.
- **Language independence: (Octave and Matlab)** EIDORS3D was originally written for Matlab. However, the eventual goal is for language independence. The current version of EIDORS3D works with Octave

(`www.octave.org`, version ≥ 2.9.3) and Matlab (version ≥ 6.0). Since Octave is free software, EIDORS3D could be provided as part of an EIT system without incurring the additional licensing costs of Matlab.

- **Pluggable code base:** In order to facilitate user modifications, EIDORS3D has been designed to provide some of the benefits of Object-oriented (OO) software: *Packaging* and *Abstraction*. We have decided not to use the OO framework provided by Matlab, because it appears somewhat cumbersome, and may be intimidating to many mathematicians and engineers who do not regularly write OO software. This decision may be revisited in the future. Instead, EIDORS3D has been designed as "Pluggable" software. This design uses function pointers to allow adding new modules and controlling which parts of functions are executed. A detailed description of this capability is provided in the next section.
- **Automatic matrix caching:** In order to increase performance of image reconstruction software, it is important to cache computationally expensive variables, such as the Jacobian and image priors. Such caching complicates the software implementation. To solve this problem, EIDORS3D extracts caching to a separate module `eidors_obj` which is inherited by each functional module. This capability is described in the next section.
- **Generalized data formats:** In order to support the wide variety of EIT measurements and algorithms, a general EIT data format structure is developed (the `fwd_model` structure). This format specifies the electrode positions, contact impedances, and stimulation and measurement patterns.
- **Interface software for common EIT systems:** Functions to load data from some common EIT systems to the EIDORS3D data format have been developed. As coverage of more data formats is implemented, we hope that this will advance the ability of researchers to share data and results.
- **Usage examples:** It is observed that researchers typically base new software on demonstration examples. To facilitate this, some simple and more complex usage examples are provided.
- **Test suite:** Software is intrinsically difficult to test. While little work has been done specifically on testing numerical software, we believe that such tests are even more difficult. EIDORS3D has begun to implement a series of regression test scripts to allow automatic testing of code modifications.
- **Logo:** Since EIDORS3D images blobby objects in aqueous media, the logo (Fig. 1), was chosen to be a walrus; it is (also) a fat, blobby animal that lives in water.

Fig. 1: EIDORS3D logo

We hope that by providing a structure for community collaboration in EIT algorithms, we can produce robust, reliable and fairly portable software which draws on our collective expertise and facilitates innovations in the field.

Software Architecture

Example: Calculation of Jacobian

In order to illustrate the usage of EIDORS3D, we consider the calculation of the Jacobian, or sensitivity matrix, **J**. Given a finite element model (FEM) model of an EIT medium, *F*, we calculate the vector of voltages, **v**, at each FEM vertex as:

$$\mathbf{v} = F(\boldsymbol{\sigma})\ \mathbf{q}$$

where $\boldsymbol{\sigma}$ is the vector of element conductivities and **q** is the current stimulation pattern, a vector of current inputs to each vertex (Neumann boundary conditions). Depending on the electrode model used, the measured electrode voltage can be represented as a linear combination of vertex voltages, $\mathbf{V}_{2E}$. For each simulation pattern, $\mathbf{q}_i$, measurements $m_{i,j}$ are performed, each of which consist of a linear combination of electrode measurements, $\mathbf{m}_j$. Thus:

$$m_{i,j} = \mathbf{m}_j\ \mathbf{V}_{2E}\ F(\boldsymbol{\sigma})\ \mathbf{q}_i$$

Based on this model, **J** is calculated as:

$$\mathbf{J}_{i,j,k} = \left.\frac{\mathbf{m}_j \mathbf{V}_{2E} \partial F(\boldsymbol{\sigma})}{\partial \boldsymbol{\sigma}_k}\right|_{\boldsymbol{\sigma}=\boldsymbol{\sigma}_0}$$

where σ_0 is the "background" conductivity around which small changes are assumed to occur. In EIDORS3D, the Jacobian is calculated using `calc_jacobian`; this function needs as parameters the FEM model `fem`, of type `fwd_model`, and the image of the background `img_bkgnd` of type `image`.

Forward Model structure `fwd_model`

The following table shows a partial representation of the structure of an EIDORS3D `fwd_model` object.

Properties	
`fwd_model.name`	Model name (arbitrary)
`fwd_model.nodes`	Position of FEM vertices (*Nodes*×*Dims*)
`fwd_model.elems`	Node indices of FEM elements (*Elems*×*Dims+1*)
`fwd_model.boundary`	Node indices of element faces on the medium surface
`fwd_model.electrode`	Vector *(Num_elecs ×1)* of electrode models (**elec_model**)
`fwd_model.stimulation`	Vector *(Num_Stim ×1)* of stimulation patterns (**stim_model**)
Methods	
`fwd_model.solve`	Function to calculate solve FEM: *usage:* `data = fwd_solve( fwd_model, image )`
`fwd_model.jacobian`	Function to calculate Jacobian at *image* background *usage:* `J = calc_jacobian( fwd_model, image )`
`fwd_model.image_prior`	Function to calculate image_prior *usage:* `J = calc_image_prior( fwd_model, image )`

Each object is created using the `eidors_obj` function, which can fill in default attributes, and keeps track of cached properties. Given a `fwd_model` *fem* a background image may be defined as follows:

```
homg_conductivity= ones( length(fem.elems) ,1);
img_bkgnd= eidors_obj('image', 'homog background', ...
  'elem_data', homg_conductivity, 'fwd_model', fem );
```

where the object `name` is assigned (arbitrarily) to 'homog background'. In order to calculate the Jacobian, we call `J= calc_jacobian( fem, img_bkgnd )`, which 1) tests to see whether **J** has been previously calculated for `fem` and `img_bkgnd`; if so, the cached value is returned; otherwise, 2) loads and calls the function in `fem.jacobian`, which may be `np_calc_jacobian`, which calls the code from Polydorides (2002); the returned value is then stored in the cache and returned to the calling function.

Usage Examples

In this section two examples of the usage of EIDORS3D software functions are provided. The first illustrates the use of forward models to simulate data. The second illustrates how to modify the image reconstruction behaviour of an existing algorithm to add new features without needing to edit the original software.

Example: Simulate Data

In order to simulate data, a FEM model object `mdl_3d` and a conductivity image need to be created.

```
% mk_circ_tank(FEM_rings, FEM_levels, n_elec, elec_planes )
param= mk_circ_tank(8, [-1:.25:1], 16, 3);

% mk_stim_patterns(n_elec, elec_planes, stim, meas, opt,
stim_amplitude)
options = {'no_meas_current','no_rotate_meas'};
params.stimulation= mk_stim_patterns(16, 3, '{ad}','{ad}', opt, 10);

params.solve=      'np_fwd_solve';
```

```
mdl_3d = eidors_obj('fwd_model', params);
show_fem( mdl_3d );  % View model
% simulate using img_bkgnd above
homg_data=fwd_solve( mdl_3d, img_bkgnd);
```

The functions `mk_circ_tank` and `mk_stim_patterns` build regular cylidrical EIT FEM models. In order to use more sophisticated simulations, these functions can be replaced by the user. For example, code to build models using *netgen* (http://www.hpfem.jku.at/netgen/) is available in the `meshing/netgen/` directory.

Example: Reconstruct Images

This example shows how modify an existing algorithm. For example, we wish to modify the hyperparameter selection strategy of Polydorides (2002) to use instead the *Noise Figure* parameter introduced by Adler and Guardo (1996). The first step is to create a `fwd_model` structure `demo_mdl` in a similar way to the above (details are shown in `examples/demo_real.m`). Subsequently, an `inv_model`, `demo_inv` is created, and the image reconstructed from difference data: `data1` and `data2`.

```
demo_inv.name= 'Nick Polydorides - Modified';
demo_inv.solve= 'np_inv_solve';
demo_inv.reconst_type= 'differential';
demo_inv.fwd_model= demo_mdl;

% modify to use Tikhonov prior
demo_inv.image_prior.func= 'tikhonov_image_prior';

% modify to use Noise Figure of 2.0
demo_inv.hyperparameter.func = 'aa_calc_noise_figure';
demo_inv.hyperparameter.noise_figure = 2.0;
demo_inv= eidors_obj('inv_model', demo_inv);

% solve inverse
solve_img= inv_solve( demo_inv, data1, data2);
```

Discussion

In this paper, we have presented the software design principles and usage examples for the EIDORS3D software suite. The overall design principle has been to facilitate a community-extensible software collection, by providing a structure that is open-source, modular, and documented. Overall, the goals for EIDORS3D are to: 1) advance research by providing a set of reference algorithms against which new research can be compared; 2) facilitate new work in EIT by providing examples, and ways to quickly expand on existing work; and 3) allow experimental and clinical researchers to easily benefit from new research in reconstruction algorithms. We encourage others to participate by using the software, reporting on any bugs, and contributing their work to the community.

References

Adler A, Guardo R (1996) "Electrical Impedance Tomography: Regularised Imaging and Contrast Detection", *IEEE Trans. Medical Imaging*, **15** 170-179

Polydorides N, Lionheart W R B (2002) "A Matlab toolkit for three-dimensional electrical impedance tomography: a contribution to the Electrical Impedance and Diffuse Optical Reconstruction Software project", *Meas. Sci. Technol.* **13** 1871-1883

Polydorides N (2002), *Image reconstruction algorithms for soft-field tomography*, Ph.D. Thesis, University of Manchester Institute of Science and Technology

Vauhkonen M (1997) Electrical impedance tomography and prior information, PhD thesis, University of Kuopio, Finland.

Vauhkonen M, Lionheart W R B, Heikkinen L M, Vauhkonen P J, Kaipio J P (2000) "A MATLAB package for the EIDORS project to reconstruct two-dimensional EIT images" *Physiol. Meas.* **22** 107-111