COMMON PROJECT FOR MATH 5750/6880 OPTIMIZATION FALL 2008

Changes

- In §3, item 2, the regularization term used the wrong matrix.
- Added section on how to compute the derivative of the forward map.
- Added section on the Newton-CG method
- Added section on other regularization terms.
- New files are: `cg.m`, `tvfun.m`, `tvfun_hess.m`, `tvfun_hessmult.m`, `test_tvfun.m`, `tvfun_init.m`, `fobj_GNHessian.m`.
- Modified files: `fobj.m`
- Fixed typo in §4. In the description of the adjoint state method, the computation of the residual (b) should be: $r(\sigma) = Cu - V$.
- Fixed typos in §5. In the description of the Gauss-Newton Hessian approximation computation (3) should be $A^T(\sigma)\psi = -QC^TC\delta u$ and (4): $w = \text{diag}\left(|T_1|, |T_2|, \cdots, |T_{\mathtt{nt}}|\right)(\nabla\psi \cdot \nabla u)$.

1. Introduction

In this project you will use electrical tomography problem (EIT) to image the (electrical) conductivity $\sigma(\mathbf{x})$ for $\mathbf{x}$ in some domain Ω (say the human body), based only on measurements at the boundary $\partial\Omega$ (i.e. the skin). The equation describing electric conduction is

$$\begin{aligned} \nabla \cdot [\sigma \nabla u] &= 0, \text{for } \mathbf{x} \in \Omega \\ \mathbf{n} \cdot \nabla u &= I, \text{for } \mathbf{x} \in \partial\Omega. \end{aligned} \tag{1}$$

This problem only makes sense if the compatibility condition

$$\int_{\partial\Omega} I(\mathbf{x}) dS(\mathbf{x}) = 0 \tag{2}$$

holds, and the solution u is defined up to an additive constant (or grounding potential). We chose this constant so that,

$$\int_{\partial\Omega} u(\mathbf{x}) dS(\mathbf{x}) = 0. \tag{3}$$

Data is acquired in N_{exp} experiments, where in each experiment a different current pattern is applied to the boundary $\partial\Omega$ (the skin of the patient). Thus the data consists of several currents $I^{(1)} \dots I^{(N_{exp})}$ and the resulting voltages $V^{(1)} \dots V^{(N_{exp})}$ recorded at the boundary only (since we only have access to the skin of the patient). The EIT problem can then be formulated as a non-linear least squares problem:

$$\min_{\sigma} \frac{1}{2} \sum_{k=1}^{N_{exp}} \int_{\partial\Omega} \left| u(\mathbf{x}; \sigma, I^{(k)}) - V^{(k)}(\mathbf{x}) \right|^2 dS(\mathbf{x}) + J_{reg}(\sigma) \tag{4}$$

here $u(\mathbf{x}; \sigma, I^{(k)})$ solves (1) with conductivity σ and boundary current $I^{(k)}$, and J_{reg} is a regularization term that makes the problem well-posed (we discuss this later).

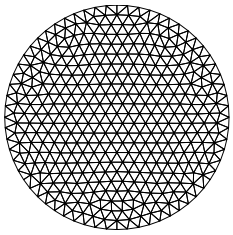

FIGURE 1. Triangulation of the unit disk.

.

The first term in the functional measures how close the data for the conductivity σ is to the measured data. Some good references for EIT are [3] and [2] (for the adjoint state method for EIT).

2. Finite element discretization

Matlab code implementing a finite element discretization of this problem with piecewise linear (P1) triangular elements is provided in `http://www.math.utah.edu/math5750_f08/eit.tgz`.

The domain we take is the unit disk (think of it as a slice of the human body) and we first need to generate a mesh:

```
% Generate the grid
grid = gengrid(32,0.1);
```

The grid is generated by the distmesh package (see [4] and `http://www-math.mit.edu/~persson/mesh/`) which is included in `eit.tgz` (`addpath distmesh` makes sure Matlab knows how to find the grid generator). The output of `gengrid` is a struct describing the grid with members

- `p`: a `np` $\times$ 2 matrix where `p(i,1)` and `p(i,2)` are the x and y coordinates of the $i-$th vertex in the grid. Here `np` is the number of vertices (a.k.a. nodes) in the grid.
- `t`: a `nt` $\times$ 3 matrix where `t(i,:)` contains the indices (in `p`) of the vertices of the $i-$th triangle in the discretization.
- `be`: a `nbe` $\times$ 2 matrix where `be(i,:)` contains the indices (in `p`) of the endpoints of the $i-$th boundary edge.

For this particular grid see Figure 1.

Then we take the conductivity function `phantom1`, which is supposed to model lungs and heart conductivity and evaluate it at the centers of the triangles of the discretization.

```
c=triangle_centers(grid);
truecond = phantom1(c(:,1),c(:,2));
```

The vector `truecond` (size: `nt` $\times$ 1) is the approximation to the conductivity that is needed for the finite element method.

The system for the finite element is setup and solved in the following lines:

```
% assemble system
A = stiffmat(truecond,grid);
```

```
% change last equation to impose grounding condition: that the
    integral over
% the boundary of the potential should be 0
A(end,:) = 0; A(end,1:nbe) = boundary_int(grid);

% assemble right hand side (the current patterns are predefined)
I = currents(grid);
I(end,:) = 0; % to go with last equation above

% solve the system
u=A\I;
```

We used the "short finite element implementations" described in [1] (`http://www.math.hu-berlin.de/~cc/english/software/shortFE.html`).

- The matrix `I` has `nbe` columns, each one corresponding to a different current pattern applied to the boundary. Thus we can solve all `nbe` systems with the expression `u=A\I`. Then `u(:,i)` is the potential resulting from the current pattern `I(:,i)`.
- The expression `g = boundary_int(grid)` generates a vector that comes handy when we want to approximate the integral of a function over the boundary and we know only the values at the boundary nodes. Here it is used to impose the grounding condition (3), and make the matrix `A` invertible, but in general if vector `v` contains values at the boundary nodes of some function $v(\mathbf{x})$, then `g'*v` $\approx \int_{\partial\Omega} v(\mathbf{x}) dS(\mathbf{x})$.

Discrete gradients can be obtained with `[Gx,Gy]=gradmat(grid);`. Given some vector `v` of node values (length `np`), `Gx*v` gives the partial derivative with respect to x of the piecewise linear interpolation of `v`:

$$v(\mathbf{x}) = \sum_{i=1}^{\texttt{np}} v_i \phi_i(\mathbf{x}),$$

where the $\phi_i(\mathbf{x})$ are functions linear on each triangle such that $\phi_i(\mathbf{x}_j) = \delta_{ij}$. Thus $\partial v/\partial x$ is constant on each triangle, and `Gx` is a `nt` $\times$ `np` matrix. Likewise, `Gy` approximates $\partial/\partial y$.

The stiffness matrix `A` (as generated by `A = stiffmat(truecond,grid);`) can also be obtained using the discrete gradient matrices `Gx` and `Gy`. If we let:

`S = spdiags(truecond.*triangle_area(grid),0,nt,nt);`

then `A == Gx'*S*Gx + Gy'*S*Gy`.

3. Project plan

1. Compute the gradient of the objective function using the adjoint state method. Check numerically that your gradient is indeed the gradient of the objective function (with your `check_gradient` function). For this purpose, you can use any matrix for the data, for example `pars.V = randn(nbe,nbe);`
2. Use BFGS to solve the EIT problem.
   - Try first to do this with data `pars.V = obsop(grid)*u`, where `u` has been obtained from the true conductivity as described above. This is not very realistic data, but makes the problem easier to solve.

- For the regularization term, take for the moment L^2 regularization: $J_{reg}(\sigma) =$ `alpha*sigma'*(ta.*sigma)` where `alpha` is a regularization parameter that you have to adjust (trial and error!) to get acceptable images, and `ta` is the vector of areas of each triangle in the discretization that can be obtained with: `ta = triangle_areas(grid)`. (**note:** multiplying element by element a vector by `ta` is the same as multiplying by the diagonal matrix with diagonal `ta`)
- **note:** It should be possible to avoid using limited memory BFGS, since the number of parameters is relatively small.

3. Compute the Fréchet derivative of the forward map (conductivity to measurements at the boundary map) and use it to compute the action of the Gauss-Newton approximation of the Hessian on some vector. We do not need to store the full Hessian to solve the Newton systems with conjugate gradient.
4. Use the Newton-CG approach to solve the EIT problem.
5. Try other regularization methods: H^1 and TV regularization.

4. Code and instructions for the adjoint state method

Please see the files `fobj.m` and `test_fobj.m`. The first file defines the objective function which is a discrete version of the objective function in (4):

$$\begin{aligned} J(\sigma) &= J_{misfit}(\sigma) + \alpha J_{reg}(\sigma) \\ &= \frac{1}{2}\sum_{i=1}^{\texttt{nbe}} (r^{(i)}(\sigma))^T Q_{bdry} r^{(i)}(\sigma) + \alpha \sigma^T D \sigma, \end{aligned} \tag{5}$$

where:

- σ is a vector of length `nt`, giving the value of the conductivity at the center of each triangle.
- $r^{(i)}(\sigma) = Cu^{(i)}(\sigma) - V^{(i)}$, $i = 1, \ldots,$ `nbe` is the residual for the $i-$th experiment.
- The observation matrix C restricts a vector defined on all the nodes to the nodes at the boundary (essentially taking measurements on the skin of the patient, file: `obsop.m`).
- The matrix D is a `nt` $\times$ `nt` diagonal matrix with $D = \text{diag}\,(|T_1|, |T_2|, \ldots, |T_{\texttt{nt}}|)$. Here $|T_i|$ is the area of the $i-$th triangle in the triangulation. The diagonal `ta` of this matrix can be obtained by `ta=triangle_area(grid);`
- The matrix Q is a symmetric positive semi-definite matrix of size `np` $\times$ `np` with entries (file: `bmassmat.m`):

$$Q_{i,j} = \int_{\partial\Omega} \phi_i(\mathbf{x})\phi_j(\mathbf{x}) dS(\mathbf{x}).$$

  Since the entries $Q_{i,j}$ are boundary integrals, the only nonzero entries are when i and j are boundary nodes. The restriction of Q to the boundary is the symmetric positive definite matrix $Q_{bdry} = CQC^T$. If we are given a vector $\mathbf{v}$ with values of some function $v(\mathbf{x})$ at the boundary nodes, then

$$\int_{\partial\Omega} |v(\mathbf{x})|^2 dS(\mathbf{x}) \approx \mathbf{v}^T Q_{bdry} \mathbf{v}.$$

Follow these steps to compute the gradient with the adjoint state method. We assume for simplicity a single experiment where current I is applied to the boundary

$\partial\Omega$ (skin of patient) and V is the voltage measured on $\partial\Omega$. The matrix $A(\sigma)$ is the stiffness matrix for σ where the last row has been adjusted to make it invertible (enforcing grounding condition), and for simplicity we compute the gradient of the misfit part of (5), that is $\nabla J_{misfit}(\sigma)$.

(a) **Solve the Forward Problem**

$$A(\sigma)u = I$$

(b) **Compute residual** $r(\sigma) = Cu - V$

(c) **Solve the Adjoint Problem**

$$A^T(\sigma)\psi = -QC^T r(\sigma)$$

(d) **Compute the gradient** (using "discrete gradients" `[Bx,By]=gradmat(grid);`).

$$\nabla J(\sigma) = \text{diag}\,(|T_1|, \ldots, |T_{\mathtt{nt}}|)(\nabla\psi \cdot \nabla u),$$

where $|T_i|$ is the area of the $i-$th triangle. The reason why we multiply by the triangle areas is that in this way:

$$\nabla J(\sigma)^T \delta\sigma = \int_\Omega \nabla J(\sigma)(\mathbf{x})\delta\sigma(\mathbf{x})d\mathbf{x}.$$

where the functions in the integrand are piecewise constant interpolations on the triangulation of the vectors $\delta\sigma$ and $\nabla J(\sigma)$.

5. Computing the Gauss-Newton approximation to the Hessian

For simplicity we consider one single experiment where current I is applied to the boundary $\partial\Omega$ and V is measured voltage. The multiple experiment case is left to you. Let $F : \mathbb{R}^{\mathtt{nt}} \to \mathbb{R}^{\mathrm{nbe}}$ be the forward map, that is the map taking a conductivity σ (vector of size `nt`) and giving the voltage resulting from the experiment setting the current at the boundary to I. With the forward map the misfit part of the objective function in (5) can be written as:

$$J_{misfit}(\sigma) = F^T(\sigma)Q_{bdry}F(\sigma)$$

The Gauss-Newton approximation to the Hessian of J_{misfit} is:

$$\nabla^2 J_{misfit}(\sigma) \approx DF[\sigma]^T Q_{bdry} DF[\sigma].$$

All we need for the CG Newton method is to know how to compute the action of the G-N approximation on some vector v, that is $w = DF[\sigma]^T Q_{bdry} DF[\sigma]v$. Here are the steps to follow, using the same notation as in §4.

(1) **Solve the Forward Problem** (already done in §4)

$$A(\sigma)u = I$$

(2) **Solve the Linearized Problem** (computes $\delta u = DF[\sigma]v$)

$$A(\sigma)\delta u = -A(v)u$$

(3) **Solve the Adjoint Problem**

$$A^T(\sigma)\psi = -QC^T C\delta u.$$

(4) **Compute** $w = DF[\sigma]^T Q_{bdry}\delta u$

$$w = \text{diag}\,(|T_1|, |T_2|, \ldots, |T_{\mathtt{nt}}|)(\nabla\psi \cdot \nabla u).$$

5.1. **Implementation notes.**

- The forward problem solution u and $A(\sigma)$ have already been computed in §4. So if Hessian information is needed, your function `fobj` should return a struct `Hinfo` that contains these precomputed quantities. Then another function uses this information to compute the matrix vector products if needed.
- To compute $A(v)$, you only need to call `Av=stiffmat(v,grid);`.
- The two last steps very similar to those in §4.
- In order to make sure you are computing the action of $DF[\sigma]^T Q_{bdry} DF[\sigma]$ right, the file `fobj_GNHessian.m` does the computation of the whole Gauss-Newton approximation to the Hessian. This is not feasible/recommended when the problem is large, but it is possible to do here because of the relatively small size.

6. Newton-CG method

The idea of this method is to replace the linear system solve that is used in Newton's method to find the step by a few steps of the conjugate gradient algorithm. The file `cg.m` implements the conjugate gradient algorithm in a way suitable for optimization methods.

There are a few parameters to play with: the number of iterations and the tolerance for the steps solutions. An accuracy of say 10^{-3} and 50 iterations maximum can give good results, but you may want to experiment with the tolerance $\min(0.5, \sqrt{\|\nabla f_k\|})\,\|\nabla f_k\|$ which guarantees super linear convergence (see Algorithm 7.1 p169).

If CG is not converging or backtracking fails to find a descent direction, you can add a small (positive) multiple of the identity to the diagonal of the Hessian. Please document as well as possible the choices of parameters you make.

7. Computing regularization terms

7.1. L^2 **regularization.** The regularization term is a multiple of the L^2 norm of the solution. This favors solutions that have a small norm, here

$$J_{reg}^{(L2)}(\sigma) = \int_\Omega |\sigma(\mathbf{x})|^2 d\mathbf{x} = \sigma^T \mathrm{diag}\,(|T_1|, \ldots, |T_{\mathtt{nt}}|)\sigma,$$

where $\sigma(\mathbf{x})$ is the interpolation of the vector $\sigma \in \mathbb{R}^{\mathtt{nt}}$ which is piecewise constant on the elements (triangles) of the discretization. The gradient and Hessian of this regularization term are trivial to compute.

7.2. **Total Variation regularization.** The regularization term is a multiple of the Total Variation (TV) of the solution. This favors solutions that are piecewise constant (or blocky):

$$J_{reg}^{(TV)}(\sigma) = \int_\Omega |\nabla \sigma(\mathbf{x})| d\mathbf{x}.$$

Recall that our ansatz for the conductivity is a piecewise constant function, so the gradient is not defined. To compute a "gradient" for such functions, we find a "dual" grid by triangulating the centers of the triangles of the original grid, and then using `gradmat.m`.

Unfortunately the TV function is not smooth because $|x|$ is not smooth. Since we need derivatives, we replace the TV functional by a smooth approximation:

$$J_{reg}^{(TV)}(\sigma) = \int_\Omega \sqrt{\|\nabla\sigma(\mathbf{x})\|_2^2 + \beta^2} d\mathbf{x},$$

where $\beta > 0$ is yet another parameter to adjust.

The files related to the (approximate) TV function are:

- `tvfun.m` computes smoothed TV functional, its gradient and additional information needed to computed the Hessian.
- `tvfun_hess.m` computes Hessian of the TV functional
- `tvfun_hessmult.m` applies the Hessian of TV functional to a vector
- `test_tvfun.m` tests gradient and Hessian computation on a randomly generated unstructured grid.
- `tvfun_init.m` initializes data structure for TV functional

7.3. H^1 **regularization (optional).** This regularization favors smooth solutions:

$$J_{reg}^{(H1)}(\sigma) = \int_\Omega \|\nabla\sigma(\mathbf{x})\|_2^2 \, d\mathbf{x}.$$

For the discrete version of this function, we use the same trick as for TV regularization. We define a "gradient" by using `gradmat.m` to build a "dual" grid out of the centers of the triangles:

```
c = triangle_centers(grid);  nc=size(c,1);
t = delaunay(c(:,1),c(:,2)); ndt=size(t,1);
dgrid.p=c; dgrid.t=t;
[Gx,Gy]=gradmat(dgrid);
```

Then for some vector `s` the H^1 functional is: `s'*Gx'*D*Gx*s + s'*Gy'*D*Gy*s`, where `D = spdiags(triangle_area(dgrid),0,ndt,ndt);`. A simpler expression for the H^1 functional is: `s'*L*s` where `L = stiffmat(ones(ndt,1),dgrid)` is the Laplacian on the "dual" grid.

References

[1] J. Alberty, C. Carstensen, and S. A. Funken. Remarks around 50 lines of Matlab: short finite element implementation. *Numer. Algorithms*, 20(2-3):117–137, 1999. ISSN 1017-1398. doi:10.1023/A:1019155918070.

[2] L. Borcea. Electrical impedance tomography. *Inverse Problems*, 18:R99–R136, 2002. doi:10.1088/0266-5611/18/6/201. Topical Review.

[3] M. Cheney, D. Isaacson, and J. C. Newell. Electrical impedance tomography. *SIAM Rev.*, 41(1):85–101 (electronic), 1999. ISSN 1095-7200. doi:10.1137/S0036144598333613.

[4] P.-O. Persson and G. Strang. A simple mesh generator in Matlab. *SIAM Rev.*, 46(2):329–345 (electronic), 2004. ISSN 0036-1445. doi:10.1137/S0036144503429121.