gerjoii

Imaging using DC currents

summer 2017

1 Forward

Using the finite volume method, solve for φ the DC process

$$-\nabla \cdot \sigma \nabla \varphi = s$$

over a rectangular region Ω (simulating a slice in depth of the earth) with Neumann boundary conditions on one edge, and Robin boundary conditions over the rest. Sources are assumed to be on the air-ground interface.

- Give grid size n, m.
- Build σ. ($n \times m$ matrix)
- Build source s. ($nm \times 1$ vector)
- Build observation matrix M. ($d \times nm$ matrix)
- Build grid edge lengths Δ. (two $n \times m$ matrices)
- Build boundary condition smoothers $\alpha = \alpha(s, \Delta)$. ($n \times m$ matrix)
- Build matrix $L = L(\sigma, \Delta, \alpha)$, b.c. are included. ($nm \times nm$ matrix)
- Compute $\varphi = L^{-1}s$. ($nm \times 1$ vector)
- Compute observations (data) $d = M\varphi$. ($d \times 1$ vector)

Matrix and vector form for writing the grid will be used interchangeably. Typically $[l, h]$ will refer to matrix form, and i or j to vector form.

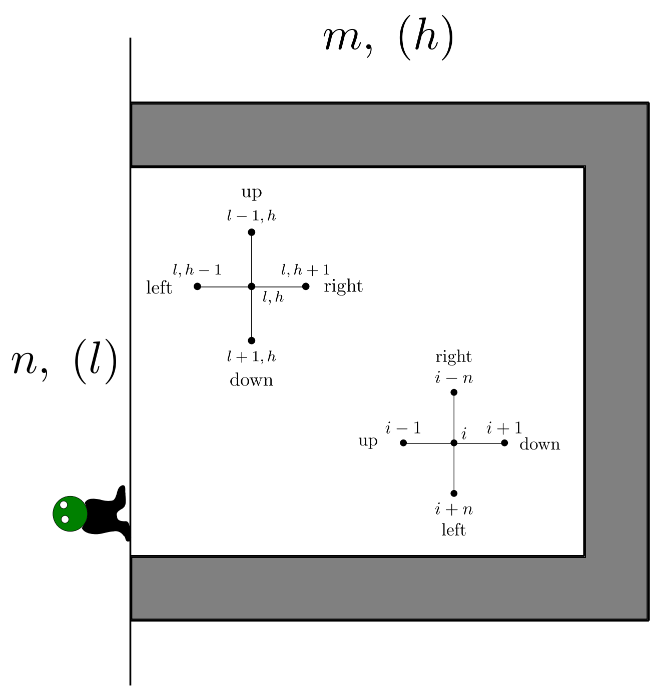

Figure 1: Computational domain of Ω. Orientation is *matrix* orientation. Different *up*, *right*, *down*, *left* depending on wether node is written in $[l,h]$ (matrix notation), or if it is written as index i (vector notation). The air-ground interface is where the black dude is. Gray area is where Robin b.c. take place.

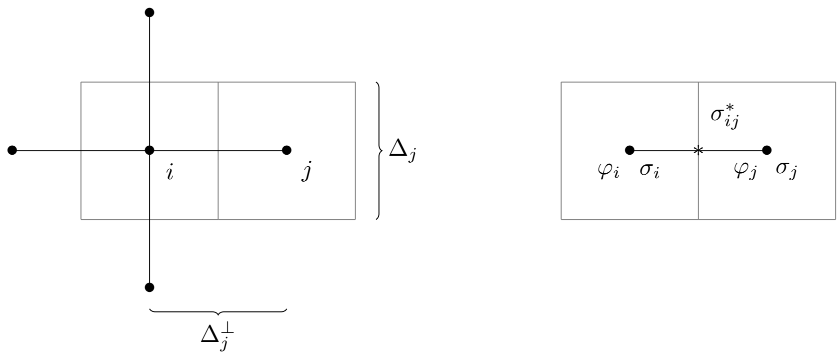

Figure 2: Regions inside Ω.

2 Inverse

Given data d^o, find $\sigma \approx \sigma^o$ that under the forward model recreates $d(\sigma) \approx d^o$. This is done by optimizing

$$\mathsf{E}(\sigma;\, d^o) = \sum_i \frac{e_i^2}{2} \qquad\qquad e = d - d^o,$$

with respect to σ.

- Build σ^o.
- Compute d^o.
- Introduce noise on d^o.
- Perform optimizing algorithm on E.

3 File names

Constructors

- `dc_sigma.m` builds σ.
- `dc_sosi.m` builds s_+, s_- in vector form.
  - `dc_sosi_compact.m` builds s_+, s_- in index form.
- `alphas.m` builds α.
- `dc_L.m` builds L.
- `dc_M.m` builds M.
- `dc_S.m` builds S.

Procedures

- `dc_fwd.m` computes φ, e, L.
- `dc_adj.m` computes adjoint λ.
- `dc_Jte.m` computes jacobian-transposed of φ times a vector.
- `dc_Jg.m` computes jacobian of φ times a vector.

- `dc_gd.m` performs gradient descent on E.
- `dc_bfgs.m` perfroms BFGS on E.
- `dc_armijo.m` computes simple back-track line search on E.
- `dc_gd_stoch.m` performs stochastic gradient descent on $\mathsf{E} = \sum_i \mathsf{E}_i$.

Examples

- `dc_run.m`
- `dc_run_many.m`
- `dc_inv.m`
- `dc_inv_many.m`

4 Building L

We want to discretize

$$L_{dc} \approx \underbrace{-\nabla \cdot \sigma \nabla}_{\mathcal{L}_{dc}}$$

$$L_{i,:} = \Big[\underbrace{a_{ik} \cdot \sigma_i + \sum_j a_{ij} \cdot \sigma^*_{ij}}_{i'th \text{ entry}} \qquad \underbrace{-b_{ij} \cdot \sigma^*_{ij}}_{j'th \text{ entries}}\Big], \qquad \sigma^*_{ij} = \frac{2\sigma_i \sigma_j}{\sigma_i + \sigma_j}$$

where,

$$\begin{aligned}
b_{ij} &= \frac{\Delta_j}{\Delta_j^{\perp}} && \text{for all nodes} \\
a_{ij} &= \frac{\Delta_j}{\Delta_j^{\perp}} && \text{inner nodes \& Neu nodes} \\
a_{ik} &= \Delta_{k_i} \cdot \alpha_{ik} && \text{Robin nodes } (0 \text{ otherwise}) \\
a_{ik} &= \Delta_{k_1} \cdot c_1 + \Delta_{k_2} \cdot c_2 && \text{corner nodes } (0 \text{ otherwise})
\end{aligned}$$

Different planes of Δ

$\Delta_{:,:,1}$ VERTICAL edges of node $(:,:)$
$\Delta_{:,:,2}$ HORIZONTAL edges of node $(:,:)$

Different planes of LL:

$LL_{:,:,1}$ entries for L of DOWN neighbor of $(:,:)$
$LL_{:,:,2}$ entries for L of UP neighbor of $(:,:)$
$LL_{:,:,3}$ entries for L of RIGHT neighbor of $(:,:)$
$LL_{:,:,4}$ entries for L of LEFT neighbor of $(:,:)$
$LL_{:,:,5}$ entries for L of GHOST neighbor of $(:,:)$
$LL_{:,:,6}$ entries for L of $-$STACK of all neighbors of $(:,:)$

Vertical (down) neighbor

$$LL_{l,h,1} = -\, 2\frac{\sigma_{l,h} \odot \sigma_{l+1,h}}{\sigma_{l,h} + \sigma_{l+1,h}} \odot \frac{\Delta_{l,h,2} + \Delta_{l,h+1,2}}{2\Delta_{l+1,h,1}}.$$

Vertical (up) neighbor

$$LL_{l,h,2} = -2\frac{\sigma_{l,h} \odot \sigma_{l-1,h}}{\sigma_{l,h} + \sigma_{l-1,h}} \odot \frac{\Delta_{l,h,2} + \Delta_{l,h+1,2}}{2\Delta_{l,h,1}}.$$

Horizontal (right) neighbor

$$LL_{l,h,3} = -2\frac{\sigma_{l,h} \odot \sigma_{l,h+1}}{\sigma_{l,h} + \sigma_{l,h+1}} \odot \frac{\Delta_{l,h,1} + \Delta_{l+1,h,1}}{2\Delta_{l,h+1,2}}.$$

Horizontal (left) neighbor

$$LL_{l,h,4} = -2\frac{\sigma_{l,h} \odot \sigma_{l,h-1}}{\sigma_{l,h} + \sigma_{l,h-1}} \odot \frac{\Delta_{l,h,1} + \Delta_{l+1,h,1}}{2\Delta_{l,h,2}}.$$

Ghost up

$$LL_{1,h,5} = -\sigma_{1,h} \odot \frac{\Delta_{1,h,2} + \Delta_{1,h+1,2}}{2} \odot \alpha_{1,h}.$$

Ghost right

$$LL_{l,m,5} = -\sigma_{l,m} \odot \frac{\Delta_{l,m,1} + \Delta_{l+1,m,1}}{2} \odot \alpha_{1,m}.$$

Ghost down

$$LL_{n,h,5} = -\sigma_{n,h} \odot \frac{\Delta_{n,h,2} + \Delta_{n,1,2}}{2} \odot \alpha_{n,h}.$$

Corner 3 up

$$LL_{1,m,5} = -\sigma_{1,m} \odot \frac{\Delta_{1,m,2} + \Delta_{1,1,2}}{2} \odot c_{1,1}.$$

Corner 3 right

$$LL_{1,m,5} = LL_{1,m,5} - \sigma_{1,m} \odot \frac{\Delta_{1,m,1} + \Delta_{2,m,1}}{2} \odot c_{2,1}.$$

Corner 4 down

$$LL_{n,m,5} = -\sigma_{n,m} \odot \frac{\Delta_{n,m,2} + \Delta_{n,1,2}}{2} \odot c_{1,2}.$$

Corner 4 right

$$LL_{n,m,5} = LL_{n,m,5} - \sigma_{n,m} \odot \frac{\Delta_{n,m,1} + \Delta_{1,m,1}}{2} \odot c_{2,2}.$$

5 Building S

We want to discretize S where,

$$-(\mathrm{d}_\sigma \mathcal{L}_{dc})\varphi \approx -(\nabla_\sigma L_{dc})\varphi = S^t.$$

Let $\partial_{\sigma_i} := \partial_i$,

$$\partial_i(\sigma^*_{ij}) = \frac{2\sigma_j^2}{(\sigma_i + \sigma_j)^2} = \partial_i(\sigma^*_{ji})$$

$$S_{i,:} = \Big[\underbrace{-a_{ik}\varphi_i + \sum_j (b_{ij}\varphi_j - a_{ij}\varphi_i) \cdot \partial_i(\sigma^*_{ij})}_{i'th \text{ entry}} \qquad \underbrace{(a_{ji}\varphi_i - b_{ji}\varphi_j) \cdot \partial_i(\sigma^*_{ji})}_{j'th \text{ entries}}\Big]$$

where,

$$\begin{aligned}
b_{ij} &= \frac{\Delta_j}{\Delta_j^\perp} && \text{for all nodes} \\
a_{ij} &= \frac{\Delta_j}{\Delta_j^\perp} && \text{inner nodes \& Neu nodes} \\
a_{ik} &= \Delta_{k_i} \cdot \alpha_{ik} && \text{Robin nodes (0 otherwise)} \\
a_{ik} &= \Delta_{k_1} \cdot c_1 + \Delta_{k_2} \cdot c_2 && \text{corner nodes (0 otherwise)}
\end{aligned}$$

and

$$\begin{aligned}
b_{ji} &= b_{ij} && \text{for all nodes} \\
a_{ji} &= a_{ij} && \text{for all nodes}
\end{aligned}$$

- The i'th entry of $S_{i,:}$ has information from $L_{i,:}$.
- The j'th entries of $S_{i,:}$ have information from $L_{j,:}$, where j has i as neighbor.
- $S_{i,:}$ has as many j'th entries as i is a neighbor of.

The structure of the code for building S is very similar to that of L, so just one example for each case is enough. See Figure (3) for a cool diagram explaining the code flow.

Vertical (down) neighbor

$$
\begin{aligned}
SB_{l,h,1} =& \frac{\Delta_{l,h,2} + \Delta_{l,h+1,2}}{2\Delta_{l+1,h,1}} \odot \varphi_{l+1,h} \\
SA_{l,h,1} =& \frac{\Delta_{l,h,2} + \Delta_{l,h+1,2}}{2\Delta_{l+1,h,1}} \odot \varphi_{l,h} \\
SD_{l,h,1} =& 2\left(\frac{\sigma_{l+1,h}}{\sigma_{l,h} + \sigma_{l+1,h}}\right)^2
\end{aligned}
$$

Ghost up

$$
SR_{1,h} = \frac{\Delta_{1,h,2} + \Delta_{1,h+1,2}}{2} \odot \alpha_{1,h} \odot \varphi_{1,h}
$$

Corner up & right

$$
SR_{1,m} = \left(\frac{\Delta_{1,m,2} + \Delta_{1,1,2}}{2} \odot c_{1,1} + \frac{\Delta_{1,m,1} + \Delta_{2,m,1}}{2} \odot c_{2,1}\right) \odot \varphi_{1,h}
$$

6 Iris data - in

Iris takes survey data in a `.txt` file with the format:

$$
\begin{matrix}
\text{r\#} & x & y & z \\
\vdots & \vdots & \vdots & \vdots \\
a & b & m & n \\
\vdots & \vdots & \vdots & \vdots \\
n_{sr} & & &
\end{matrix} \tag{1}
$$

where r# is the electrode number, (x, y, z) are its field coordinates, (a, b, m, n) are source-receiver pairs and n_{sr} is the number of source-receiver shots.

See routine `dc_gerjoii2iris.m` and script `gerjoii2iris_dc.m`.

7 Iris data - out

Iris performs experiments by picking a injecting current on the source and measuring on *some* of the receivers associated to that source, not all of them. The

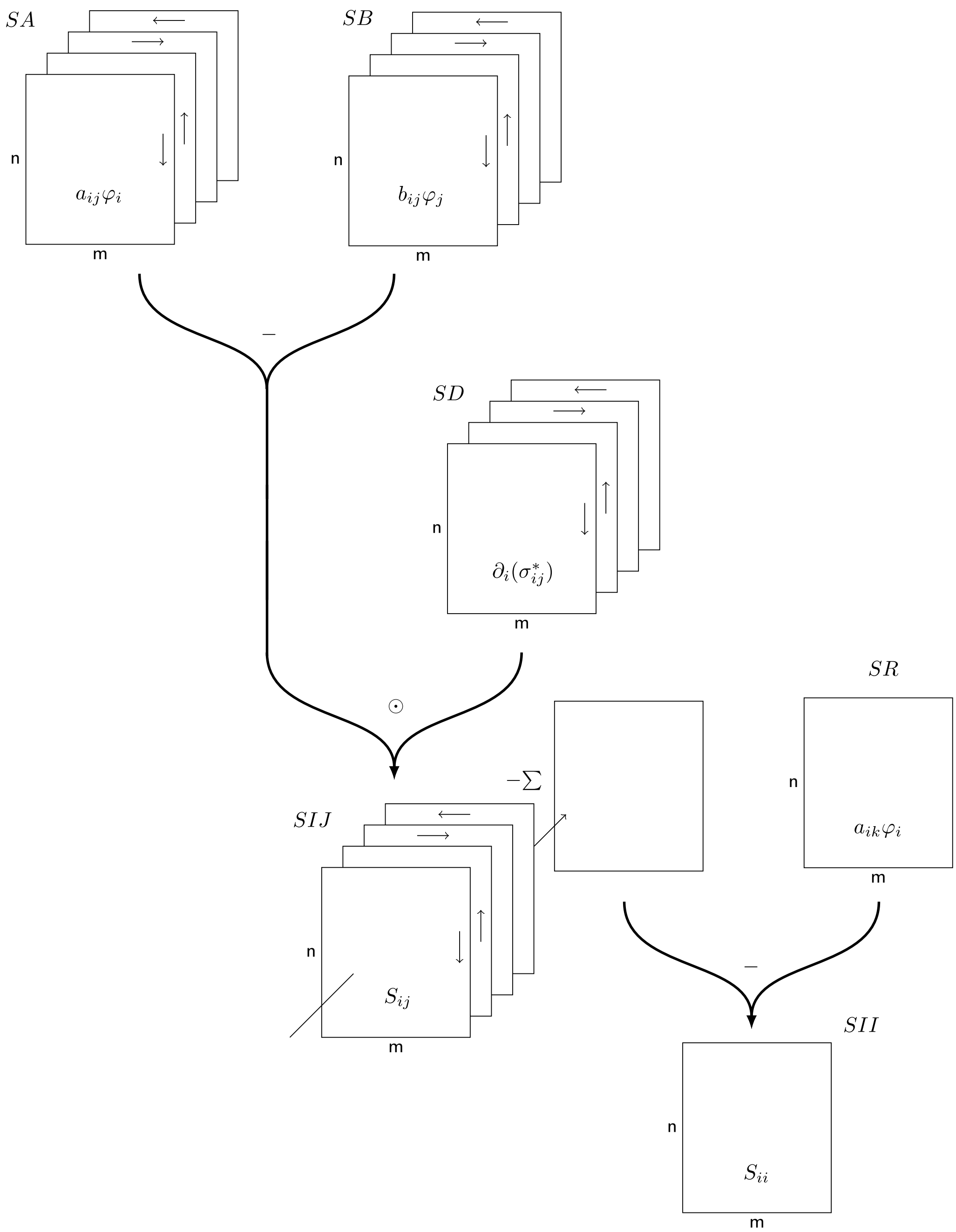

Figure 3: Building the entries of S. Matrices (and groups of) are labeled by their respective entry in the i'th node. Names used in the code are displayed next to the matrices.

missing receivers for a given source are then covered with a different current magnitude.

Routine dc_iris2gerjoii.m and script iris2gerjoii_dc.m bundle all common source shots in one cell type (s_i_r_d_std$\{i_e\}$) indexed by experiment number containing source, current, receivers, observed voltage and observed standard deviation. This bundle has sources and receivers in electrode number.

To use the bundle s_i_r_d_std$\{i_e\}$ in gerjoii the function dc_electrodes.m converts numbered real coordinate electrodes into the vectorized format dc_fwd.m needs:

Here goes a receiver diagram:
rectangle of size $(n_r \times 2)$ labeled by columns r_+, r_- and by rows # of receiver (these are the numbered receivers) $\rightarrow$ two rectangles each of size $(n_r \times 2)$ labeled by columns x, z, by rows # of receiver and one labeled + and the other − (these are the real coordinate receivers) $\rightarrow$ two rectangles each of size $(n_r \times 2)$ labeled by columns ix, iz, by rows # of receiver and one labeled + and the other − (these are the binned receivers). Then comes an arrow pointing down and an arrow pointing right, the down arrow goes to *expand to robin* and the right arrow goes to two rectangles each of size $(n_r \times 1)$ labeled r_+, r_- and by rows # of receiver (these are the vectorized receivers).

Here goes a source diagram:
rectangle of size (1×2) labeled by columns s_+, s_- (this is the numbered source) $\rightarrow$ two rectangles each of size (1×2) labeled by columns x, z and one labeled + and the other − (this is the real coordinate source) $\rightarrow$ two rectangles each of size (1×2) labeled by columns ix, iz and one labeled + and the other − (this is the binned source). Then comes an arrow pointing down and an arrow pointing right. The down arrow goes to *expand to robin*, then to the right $\rightarrow$ *clean source* and then up to two squares each of size (1×1) labeled s_+, s_- (this is the vectorized source). The right arrow goes to the vectorized source.