Modelling and inversion of electromagnetic data

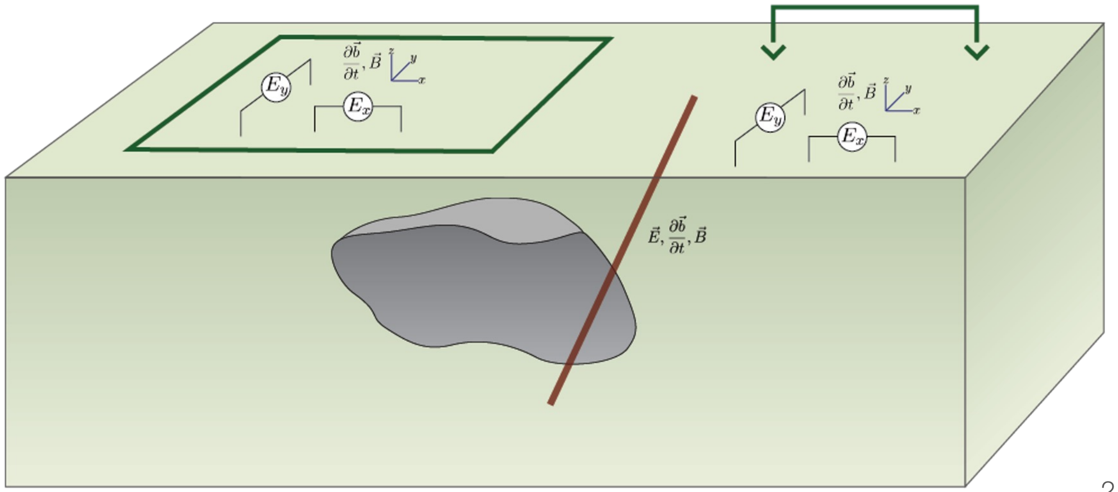
Lindsey Heagy
University of British Columbia - Geophysical Inversion Facility

January 19, 2025

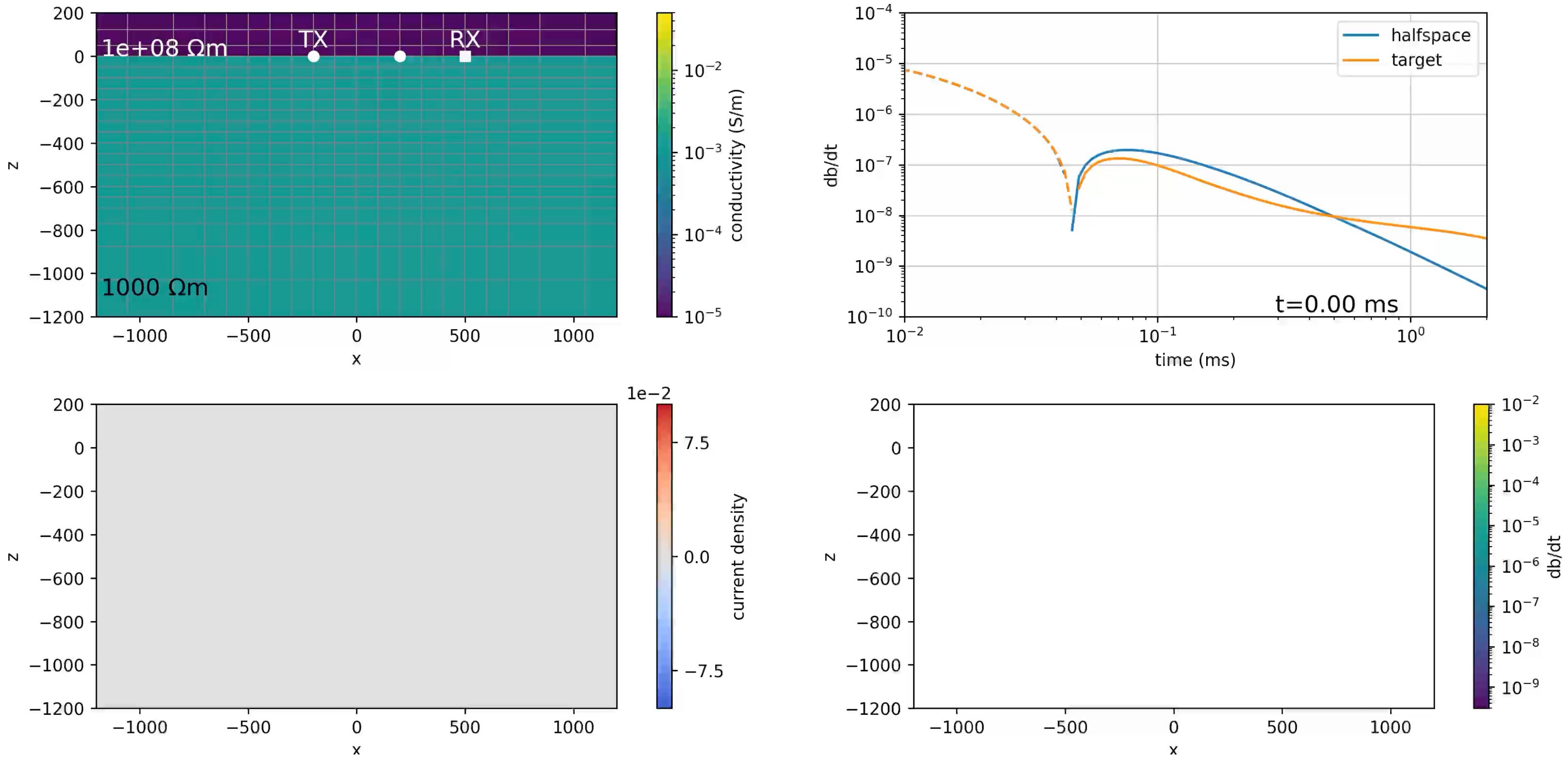
Ground and Borehole EM

setup:

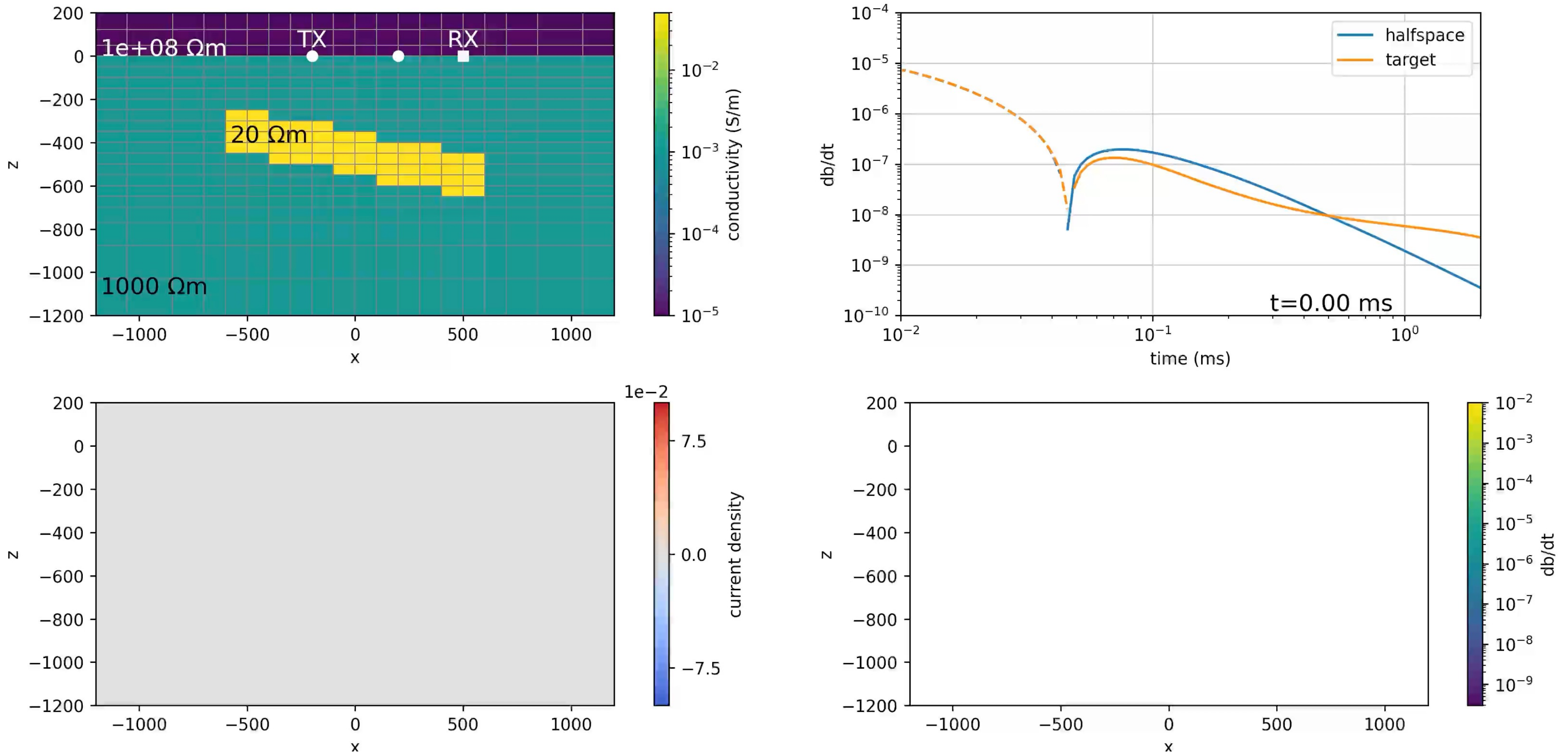
- grounded or inductive source
- measure E, B, db/dt
- x, y, z components
- on ground or in boreholes



Inductive source time domain response



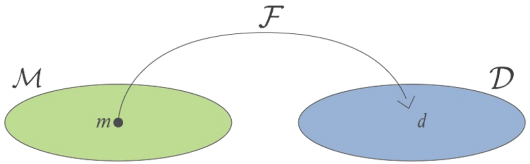
Inductive source time domain response



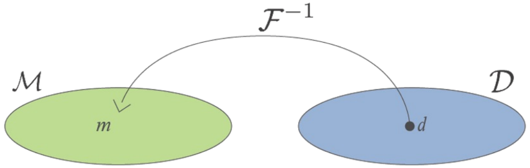
Statement of the inverse problem

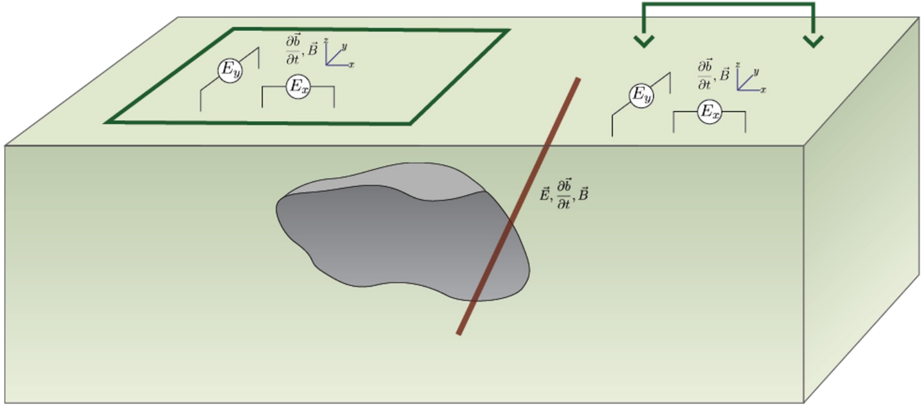
Given

- observations: $d_j^{obs}, \quad j = 1, \ldots, N$
- uncertainties: ϵ_j
- ability to forward model: $\mathcal{F}[m] = d$



Find the Earth model that gave rise to the data





Electromagnetics: basic equations (quasi-static)

	Time	Frequency
Faraday's Law	$\nabla \times \vec{e} = -\frac{\partial \vec{b}}{\partial t}$	$\nabla \times \vec{E} = -i\omega\vec{B}$
Ampere's Law	$\nabla \times \vec{h} = \vec{j} + \frac{\partial \vec{d}}{\partial t}$	$\nabla \times \vec{H} = \vec{J} + i\omega\vec{D}$
No Magnetic Monopoles	$\nabla \cdot \vec{b} = 0$	$\nabla \cdot \vec{B} = 0$
Constitutive Relationships (non-dispersive)	$\vec{j} = \sigma\vec{e}$ $\vec{b} = \mu\vec{h}$ $\vec{d} = \varepsilon\vec{e}$	$\vec{J} = \sigma\vec{E}$ $\vec{B} = \mu\vec{H}$ $\vec{D} = \varepsilon\vec{E}$

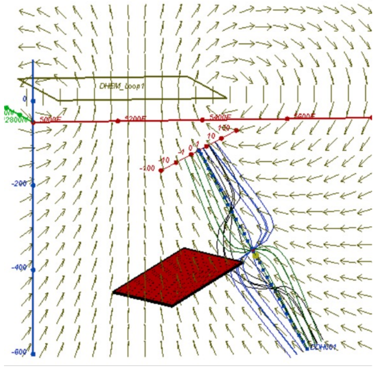
* Solve with sources and boundary conditions

Forward modelling

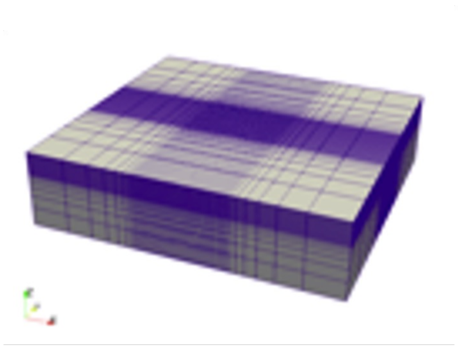
Lots of choices…

- plate modelling, 1D, cylindrically symmetric, 3D
- time or frequency
- different mesh choices
  - structured
  - semi-structured (e.g. OcTree)
  - unstructured (e.g. tetrahedral)
- discretization strategy
  - integral equation
  - finite difference
  - finite volume
  - finite element

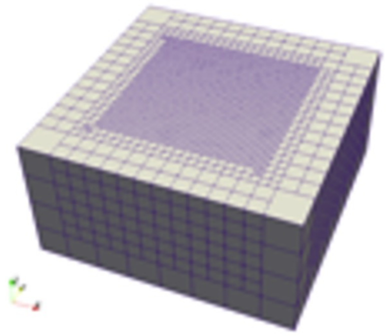
plate modelling



3D tensor



3D OcTree

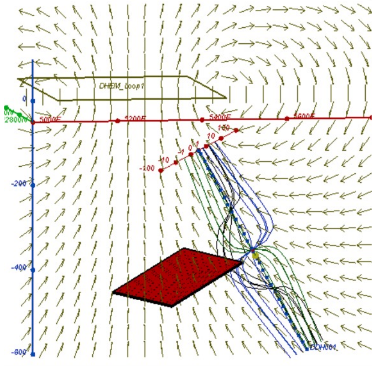


Forward modelling

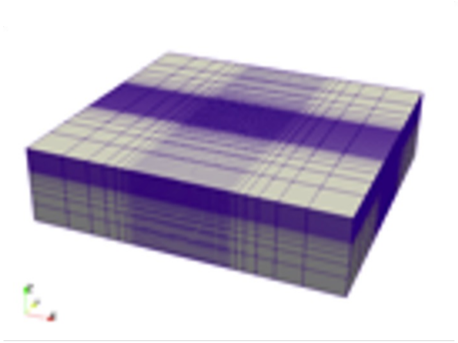
Our focus today

- plate modelling, 1D, cylindrically symmetric, 3D
- time or frequency
- different mesh choices
  - structured
  - semi-structured (e.g. OcTree)
  - unstructured (e.g. tetrahedral)
- discretization strategy
  - integral equation
  - finite difference
  - finite volume
  - finite element

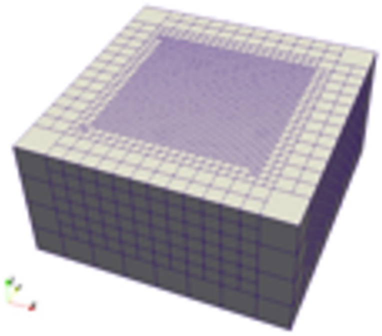
plate modelling



3D tensor



3D OcTree



Solving the frequency domain equations

Use constitutive relations, reduce to two equations, one field, one flux

$$\nabla \times \vec{E} + i\omega\vec{B} = 0$$

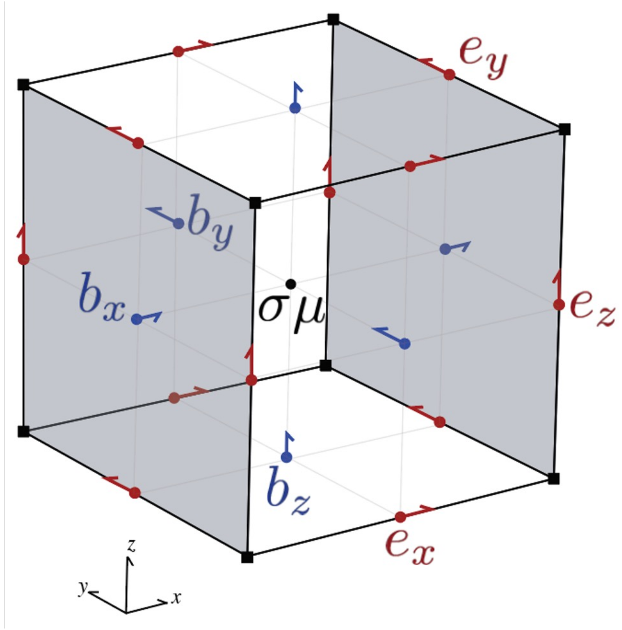
$$\nabla \times \mu^{-1}\vec{B} - \sigma\vec{E} = \vec{J}_s$$

Boundary conditions

$$\hat{n} \times \vec{B}|_{\partial\Omega} = 0$$

Staggered grid discretization

- Physical properties: cell centers
- Fields: edges
- Fluxes: faces



Solving the frequency domain equations

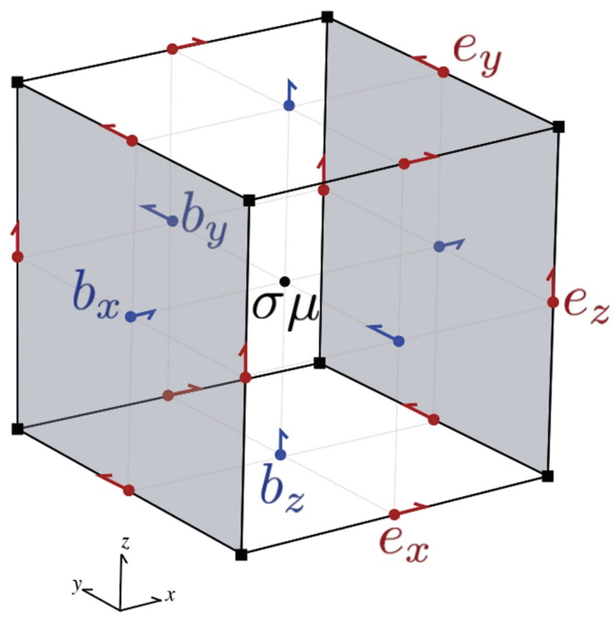
Discrete equations

$$\mathbf{C}\mathbf{e} + i\omega\mathbf{b} = 0$$

$$\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}}\mathbf{b} - \mathbf{M}^e_\sigma \mathbf{e} = \mathbf{M}^e \mathbf{j}_s$$

Eliminate **b** to obtain a second-order system in **e**

$$\underbrace{(\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}}\mathbf{C} + i\omega \mathbf{M}^e_\sigma)}_{\mathbf{A}(\sigma,\omega)} \underbrace{\mathbf{e}}_{\mathbf{u}} = \underbrace{-i\omega\mathbf{M}^e\mathbf{j_s}}_{\mathbf{q}(\omega)}$$



Solving the frequency domain equations

$$\underbrace{(\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{C} + i\omega \mathbf{M}^e_\sigma)}_{\mathbf{A}(\sigma,\omega)} \underbrace{\mathbf{e}}_{\mathbf{u}} = \underbrace{-i\omega \mathbf{M}^e \mathbf{j_s}}_{\mathbf{q}(\omega)}$$

$\mathbf{A}(\sigma,\omega)$

- Complex
- Symmetric
- Factor once for each frequency (solve for multiple sources)
- Need to refactor on each model update
- Separable over frequencies

Solving the time domain equations

Use constitutive relations, reduce to two equations, one field, one flux

$$\nabla \times \vec{e} + \frac{\partial \vec{b}}{\partial t} = 0$$

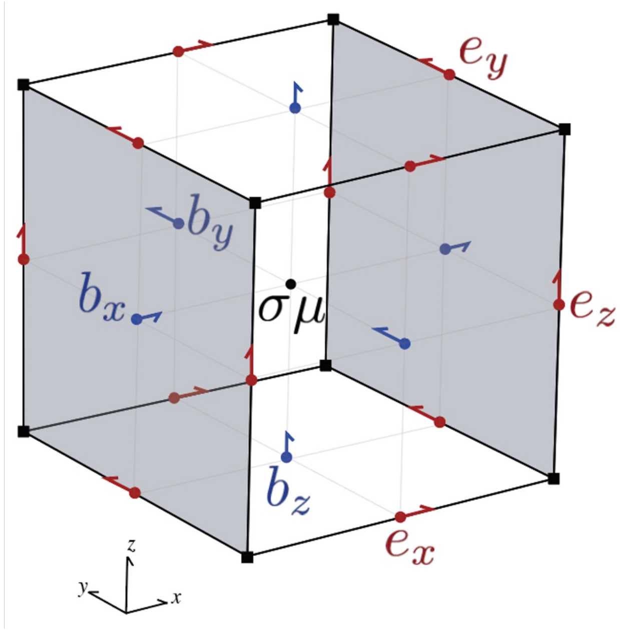
$$\nabla \times \mu^{-1}\vec{b} - \sigma\vec{e} = \vec{j}_s$$

Boundary conditions

$$\hat{n} \times \vec{b}|_{\partial\Omega} = 0$$

Staggered grid discretization

- Physical properties: cell centers
- Fields: edges
- Fluxes: faces

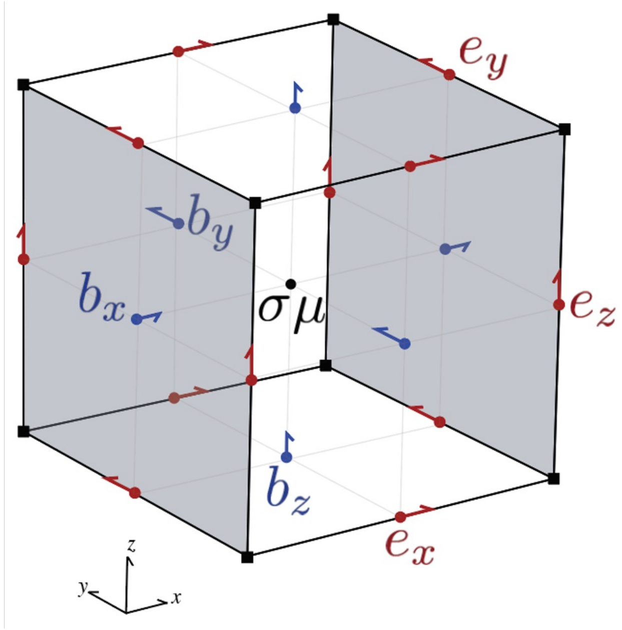


Solving the time domain equations

Semi-discretization in space

$$\mathbf{C}\mathbf{e} + \frac{\partial \mathbf{b}}{\partial t} = 0$$

$$\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{b} - \mathbf{M}^e_\sigma \mathbf{e} = \mathbf{M}^e \mathbf{j}_s$$



combine into a single, second order equation in **e**

$$\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{C}\mathbf{e} + \mathbf{M}^e_\sigma \frac{\partial \mathbf{e}}{\partial t} = -\mathbf{M}^e \frac{\partial \mathbf{j}_s}{\partial t}$$

to solve, need to discretize in time…

Solving the time domain equations

First order backward difference (backward euler)

$\mathbf{e}^{n+1}$ depends upon $\mathbf{e}^{n}$

$$\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{C} \mathbf{e} + \mathbf{M}^e_\sigma \frac{\partial \mathbf{e}}{\partial t} = -\mathbf{M}^e \frac{\partial \mathbf{j}_s}{\partial t}$$

Time step: $\Delta t = t^{n+1} - t^n$

$$\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{C} \mathbf{e}^{n+1} + \mathbf{M}^e_\sigma \frac{\mathbf{e}^{n+1} - \mathbf{e}^n}{\Delta t} = -\mathbf{M}^e \frac{\mathbf{j}_s^{n+1} - \mathbf{j}_s^n}{\Delta t}$$

Rearrange to solve for

$$\left(\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{C} + \frac{1}{\Delta t}\mathbf{M}^e_\sigma\right) \mathbf{e}^{n+1} = \frac{1}{\Delta t}\mathbf{M}^e_\sigma \mathbf{e}^n - \frac{1}{\Delta t}\mathbf{M}^e(\mathbf{j}_s^{n+1} - \mathbf{j}_s^n)$$

Solving the time domain equations

Discrete equations

$$\underbrace{\left(\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{C} + \frac{1}{\Delta t}\mathbf{M}^e_\sigma\right)}_{\mathbf{A}_{n+1}(\sigma,\Delta t)} \underbrace{\mathbf{e}^{n+1}}_{\mathbf{u}_{n+1}} = \underbrace{\frac{1}{\Delta t}\mathbf{M}^e_\sigma}_{\mathbf{B}_{n+1}(\sigma,\Delta t)} \underbrace{\mathbf{e}^n}_{\mathbf{u}_n} - \underbrace{\frac{1}{\Delta t}\mathbf{M}^e(\mathbf{j}_s^{n+1} - \mathbf{j}_s^n)}_{\mathbf{q}_{n+1}}$$

Arrange in a big matrix to solve

$$\begin{pmatrix} \mathbf{A}_0 & & & & & \\ \mathbf{B}_1 & \mathbf{A}_1 & & & & \\ & \mathbf{B}_2 & \mathbf{A}_2 & & & \\ & & \ddots & \ddots & & \\ & & & \mathbf{B}_{n-1} & \mathbf{A}_{n-1} & \\ & & & & \mathbf{B}_n & \mathbf{A}_n \end{pmatrix} \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{n-1} \\ \mathbf{u}_n \end{pmatrix} = \begin{pmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_{n-1} \\ \mathbf{q}_n \end{pmatrix}$$

Solving the time domain equations

Solve with forward substitution

- Solve initial conditions for $\mathbf{u}_0$

$$\mathbf{A}_0 \mathbf{u}_0 = \mathbf{q}_0$$

- Propagate forward by solving

$$\mathbf{A}_{n+1} \mathbf{u}_{n+1} = -\mathbf{B}_n \mathbf{u}_n + \mathbf{q}_{n+1}$$

$$\begin{pmatrix} \mathbf{A}_0 & & & & & \\ \mathbf{B}_1 & \mathbf{A}_1 & & & & \\ & \mathbf{B}_2 & \mathbf{A}_2 & & & \\ & & \ddots & \ddots & & \\ & & & \mathbf{B}_{n-1} & \mathbf{A}_{n-1} & \\ & & & & \mathbf{B}_n & \mathbf{A}_n \end{pmatrix} \begin{pmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{n-1} \\ \mathbf{u}_n \end{pmatrix} = \begin{pmatrix} \mathbf{q}_0 \\ \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_{n-1} \\ \mathbf{q}_n \end{pmatrix}$$

$\mathbf{A}_{n+1}(\sigma, \Delta t)$

- Symmetric
- Need to solve many times
- Only changes if $\sigma, \Delta t$ changes → store factors & re-use

$$\underbrace{\left(\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{C} + \frac{1}{\Delta t}\mathbf{M}^e_\sigma\right)}_{\mathbf{A}_{n+1}(\sigma, \Delta t)}$$

Frequency and Time domain simulations

Frequency

$$\mathbf{A}(\sigma,\omega)$$

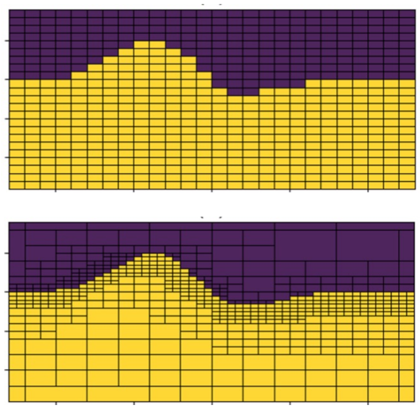
- Complex, symmetric
- Factor for each frequency
- Inversion: sensitivity derivation straight-forward

Time

$$\mathbf{A}_{n+1}(\sigma, \Delta t)$$

- Real, symmetric
- Factor for each unique Δt
- Inversion: sensitivity derivation more involved

Setting up a forward simulation & user decisions

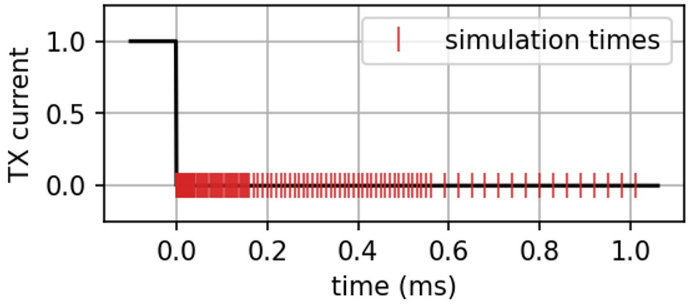


Mesh design

- minimum cell size
  - capture shortest time / highest frequency, highest conductivity
- domain extent
  - beyond skin depth / diffusion distance

Time discretization

- capture short-timescale variations near shut-off
  - needs to be finer than receiver times
- extend to latest time channel



trade-off between computation cost and accuracy

Forward modelling can address

Survey design:

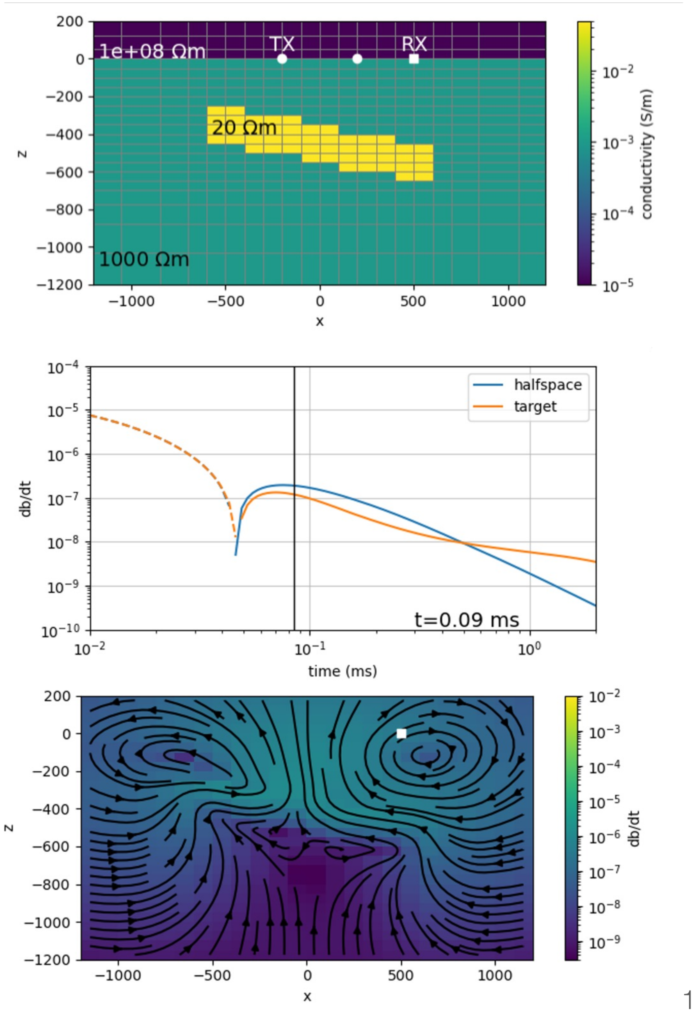
- source: coupling? base frequency?
- receivers: best to measure e, b, db/dt?

Complexities:

- overburden
- multiple targets, interactions?

Value in looking at:

- currents, magnetic field
- and simulated data



Practical notes

Computational time for this example

Tensor Mesh with…

- 62,000 cells
- 170 time steps, 4 unique Δt
- 1 source

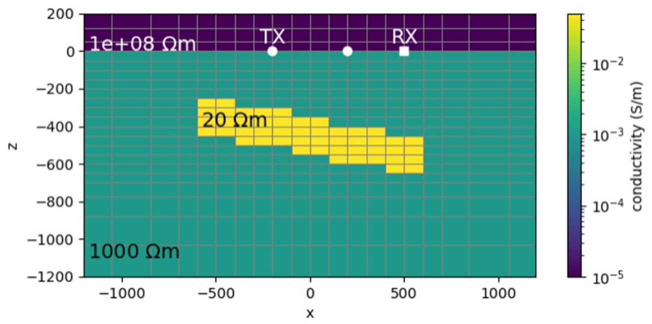
  → **Forward simulation time: 57s**

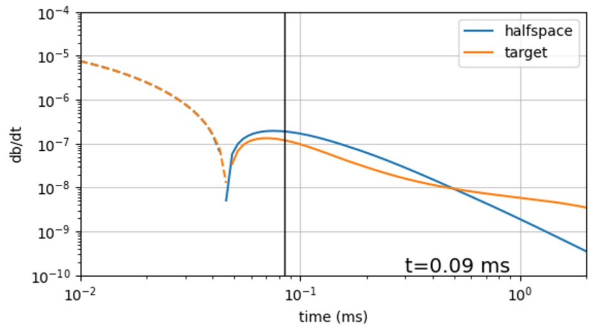
Refine the mesh (~double number of cells)

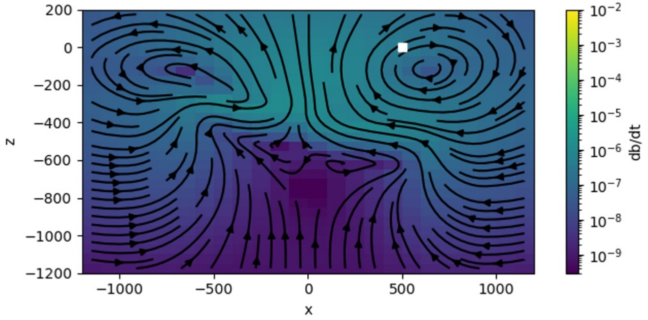
- 128,000 cells
- same time stepping

  → **Forward simulation time: 136s**

simpeg



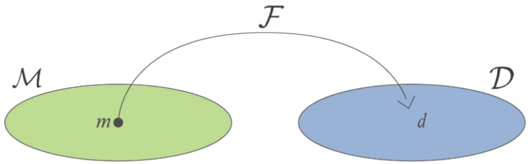




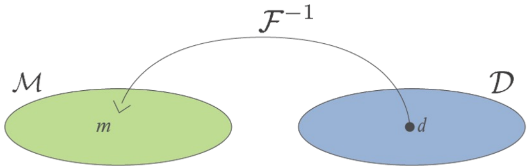
Statement of the inverse problem

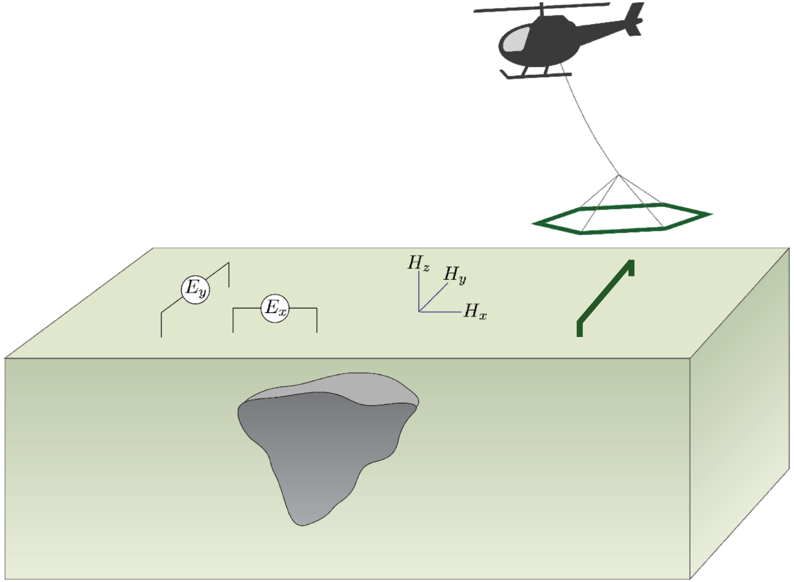
Given

- observations: $d_j^{obs}, \quad j = 1, \ldots, N$
- uncertainties: ϵ_j
- ability to forward model: $\mathcal{F}[m] = d$



Find the Earth model that gave rise to the data





Need a framework for inverse problem

Tikhonov (deterministic)

Find a single “best” solution by solving optimization

$$\text{minimize} \quad \phi = \phi_d + \beta\phi_m$$

$$\text{subject to} \quad m_L < m < m_H$$

$$\begin{cases} \phi_d\text{: data misfit} \\ \phi_m\text{: regularization} \\ \beta\text{: trade-off parameter} \\ m_L, m_H\text{: lower and upper bounds} \end{cases}$$

Bayesian (probabilistic)

Use Bayes’ theorem

$$P(m|d^{obs}) \propto P(d^{obs}|m)P(m)$$

$$\begin{cases} P(m)\text{: prior information about } m \\ P(d^{obs}|m)\text{: probability about the data errors (likelihood)} \\ P(m|d^{obs})\text{: posterior probability for the model} \end{cases}$$

Two approaches:

(a) Characterize $P(m|d^{obs})$

(a) Find a particular solution that maximizes $P(m|d^{obs})$
MAP: (maximum a posteriori) estimate

Flow chart for the inverse problem

What do we need for inversion?

$$\text{minimize} \quad \phi = \phi_d + \beta\phi_m$$

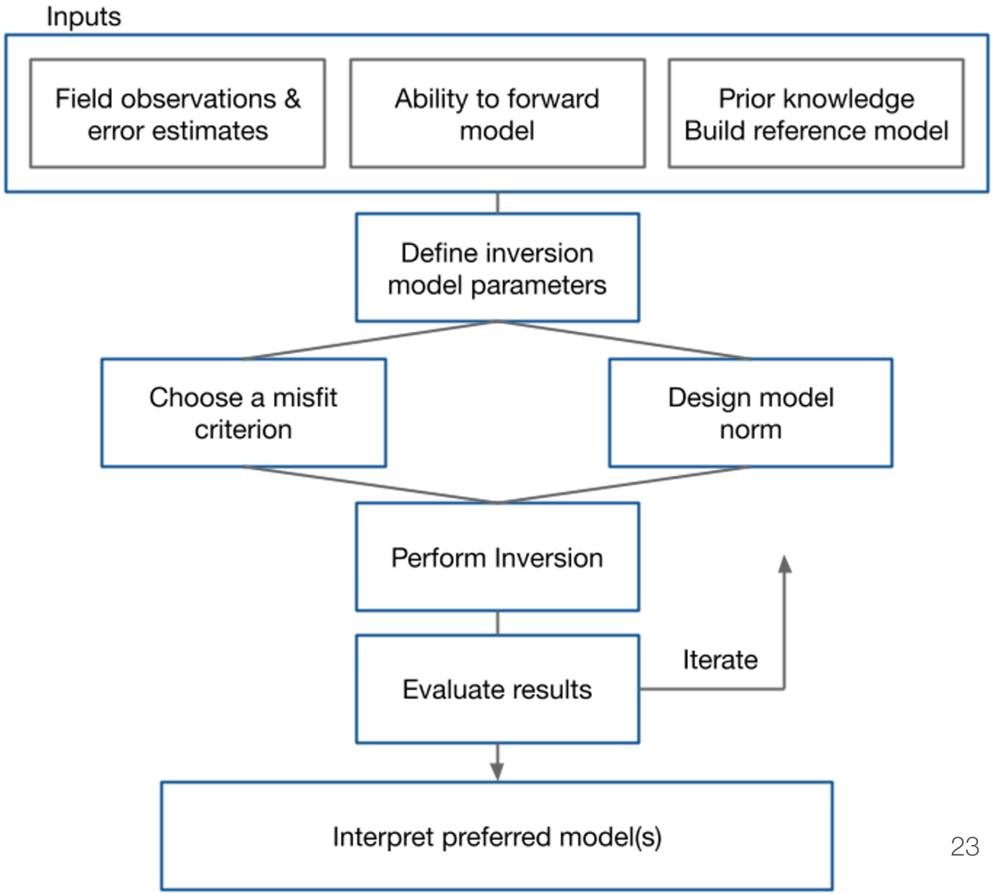
$$\text{subject to} \quad m_L < m < m_H$$

From the simulation

- adjoint sensitivity times a vector
- sensitivity times a vector

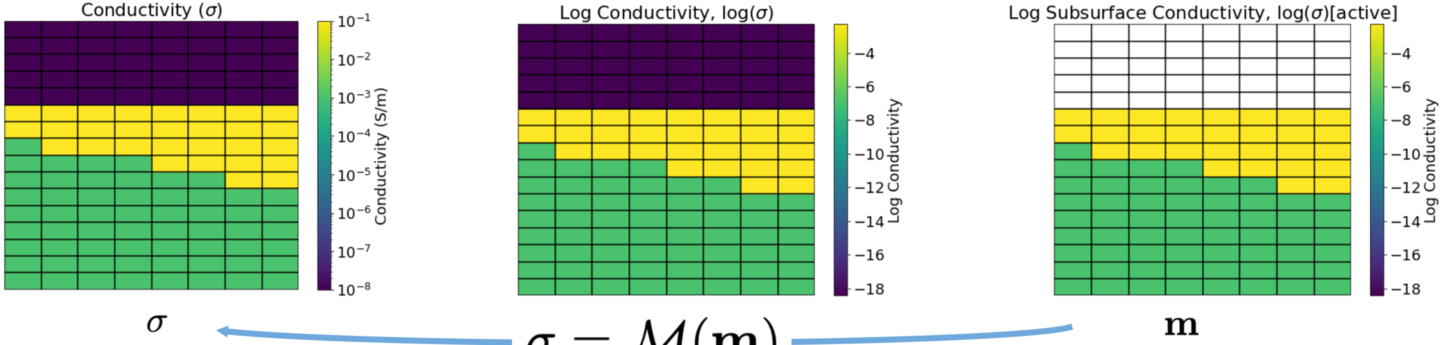
Inversion components:

- define a model norm
- perform optimization



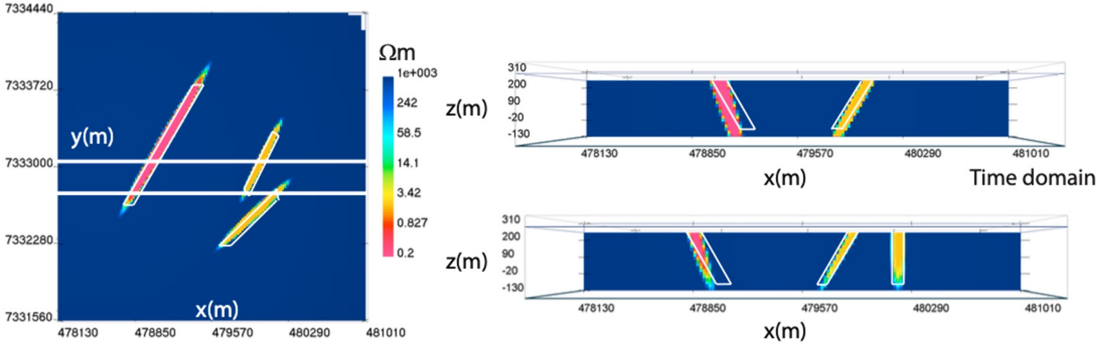
Defining the model model parameters

typically log conductivity below the surface of the earth



$$\sigma = \mathcal{M}(\mathbf{m})$$

can also work with parametric models (e.g. ellipsoids)



McMillan et al., 2017

Sensitivities

For inverse problem, also need sensitivities (and adjoint)

$$\mathbf{J} = \frac{\partial \mathbf{d}^{\text{pred}}}{\partial \mathbf{m}}$$

$$= \frac{\partial \mathbf{P}(\mathbf{u}, \omega)}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{m}}$$

where the derivative of the fields (**u**) is computed implicitly (requires a solve)

$$\frac{\partial \mathbf{A}(\sigma, \omega)\mathbf{u}^{\text{fixed}}}{\partial \mathbf{m}} + \mathbf{A}(\sigma, \omega)\frac{\partial \mathbf{u}}{\partial \mathbf{m}} = 0$$

J is a large, dense matrix → compute products with a vector if memory-limited

Inversion as an optimization problem

$$\min_{\mathbf{m}} \phi(\mathbf{m}) = \phi_d(\mathbf{m}) + \beta\phi_m(\mathbf{m})$$

$$\text{s.t. } \phi_d \leq \phi_d^* \quad \mathbf{m}_L \leq \mathbf{m} \leq \mathbf{m}_U$$

data misfit

$$\phi_d = \|\mathbf{W}_d(\mathcal{F}(\mathbf{m}) - \mathbf{d}^{\text{obs}})\|^2$$

uncertainties captured in **W**

$$\mathbf{W}_d = \text{diag}\left(\frac{1}{\epsilon}\right)$$

$$\epsilon_j = \%|d_j^{\text{obs}}| + \text{floor}$$

model norm (L2)

$$\phi_m = \underbrace{\alpha_s \int_V w_s (m - m_{\text{ref}})^2 dV}_{\text{smallness}} + \underbrace{\alpha_x \int_V w_x \frac{d(m - m_{\text{ref}})}{dx}^2 dV}_{\text{smoothness}}$$

discretize

$$\phi_m = \alpha_s\|\mathbf{W}_s(\mathbf{m} - \mathbf{m}_{\text{ref}})\|^2 + \alpha_x\|\mathbf{W}_x(\mathbf{m} - \mathbf{m}_{\text{ref}})\|^2$$

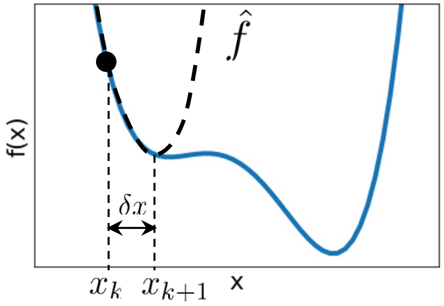
Gauss Newton approach

$$\min_{\mathbf{m}} \phi(\mathbf{m}) = \phi_d(\mathbf{m}) + \beta\phi_m(\mathbf{m})$$

$$\text{s.t. } \phi_d \leq \phi_d^* \quad \mathbf{m}_L \leq \mathbf{m} \leq \mathbf{m}_U$$

Gradient

$$\mathbf{g}(\mathbf{m}) = \mathbf{J}^\top \mathbf{W}_d^\top \mathbf{W}_d(F[\mathbf{m}] - \mathbf{d}^{obs}) + \beta \mathbf{W}_m^\top \mathbf{W}_m(\mathbf{m} - \mathbf{m}_{ref})$$



Taylor expand → Gauss newton equations (solve with CG)

$$(\mathbf{J}^\top \mathbf{W}_d^\top \mathbf{W}_d \mathbf{J} + \beta \mathbf{W}_m^\top \mathbf{W}_m)\delta\mathbf{m} = -\mathbf{g}(\mathbf{m})$$

Sensitivity (**J**) requires forward modelling

- large, dense matrix (expensive to store)
- compute matrix-vector products when memory-limited $\mathbf{J}\mathbf{v}, \mathbf{J}^\top\mathbf{v}$

Practical notes

One inversion iteration requires we solve:

$$\mathbf{g}(\mathbf{m}) = \underline{\mathbf{J}^\top} \mathbf{W}_d^\top \mathbf{W}_d (\underline{F[\mathbf{m}]} - \mathbf{d}^{obs}) + \beta \mathbf{W}_m^\top \mathbf{W}_m (\mathbf{m} - \mathbf{m}_{ref})$$

$$(\underline{\mathbf{J}^\top} \mathbf{W}_d^\top \mathbf{W}_d \underline{\mathbf{J}} + \beta \mathbf{W}_m^\top \mathbf{W}_m)\delta \mathbf{m} = -\mathbf{g}(\mathbf{m})$$

Number of forward calculations per iteration:

$$2(N_{CG} + 1) \sim 20$$

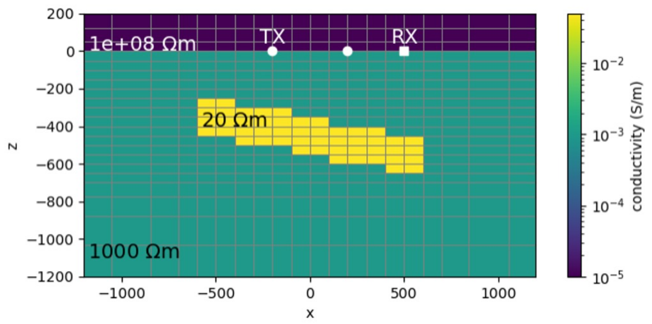
62,000 cells **~ 57s**

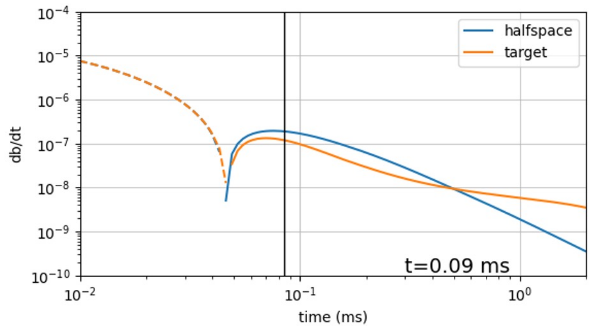
Total computation time:

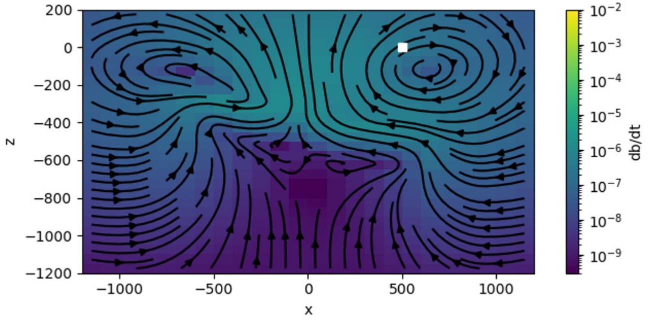
$$N_{iter} \cdot 2(N_{CG} + 1) \cdot T$$

~10 ~20 ~1 min (small prob)

~3.5 hours







3D inversions with EM

Challenges: computational cost

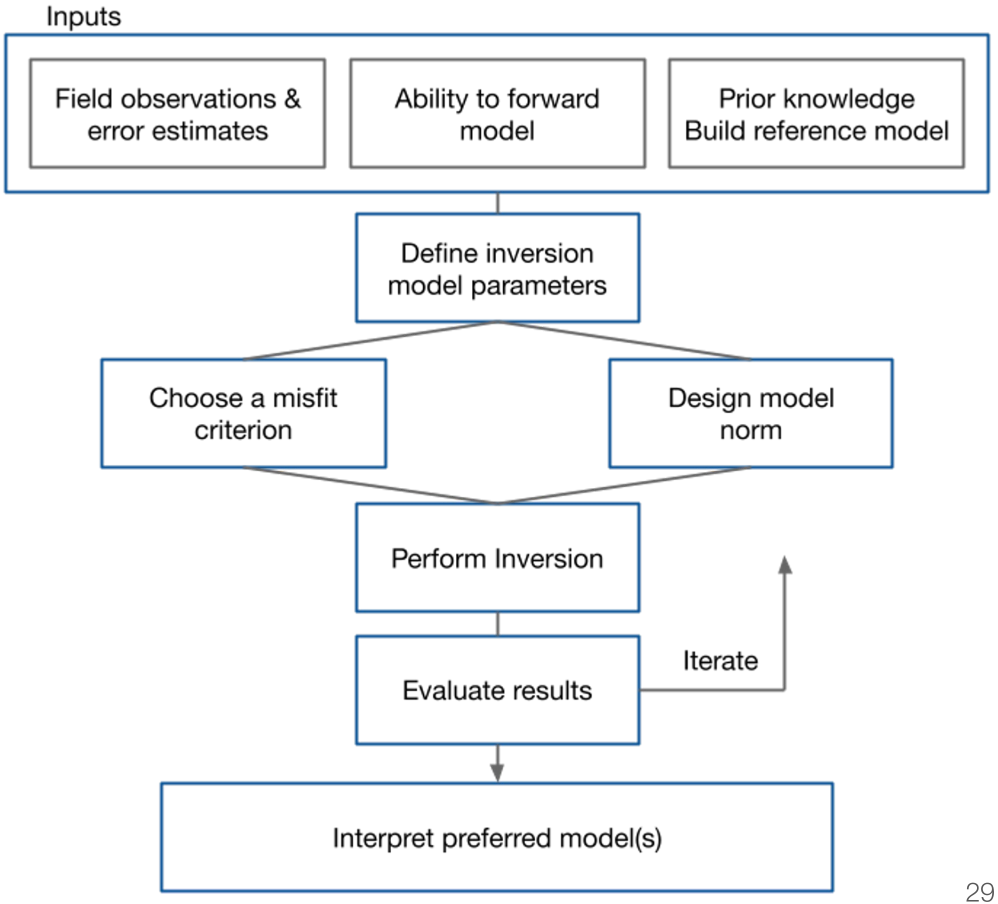
- need many forward simulations to run an inversion

Non-uniqueness

- all inversions non-unique
- impacted by survey geometry

Role of a-priori information

- reference models
- model parameterization
- …



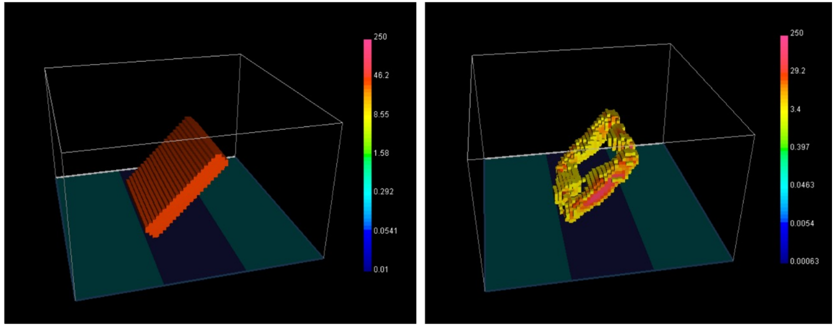
Research avenues

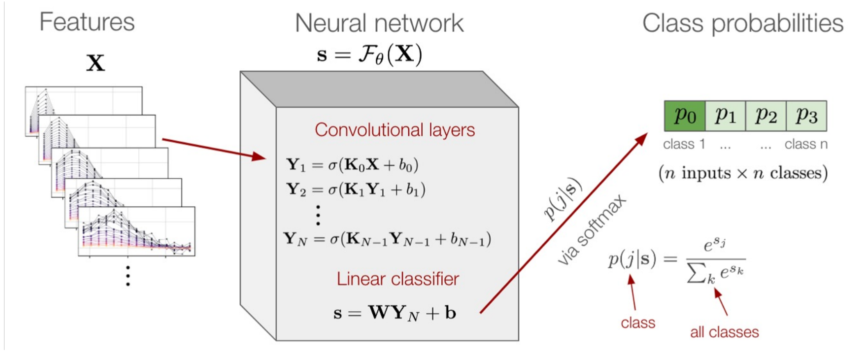
3D inversion with highly conductive targets

- Challenging sensitivities, modelling
- Role for parametric inversions

Incorporating in a-priori information

- Different norms in the regularization
- Petrophysically and Geologically guided inversions (PGI)
- Opportunities for machine learning





SimPEG

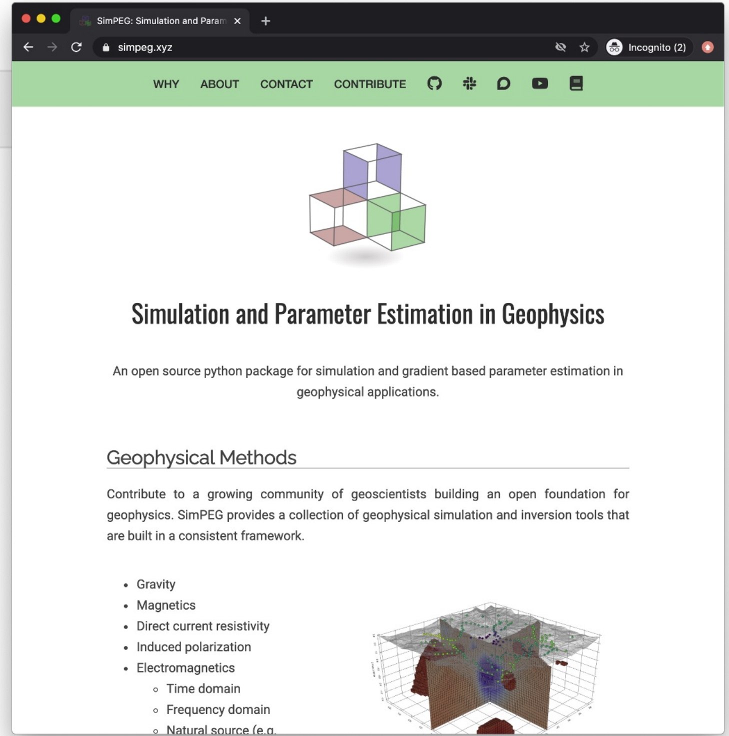
```
[1]: from simpeg import electromagnetics
```

forward simulation and inversion tools

- frequency and time
- 1D, 3D, cylindrically symmetric

inversions

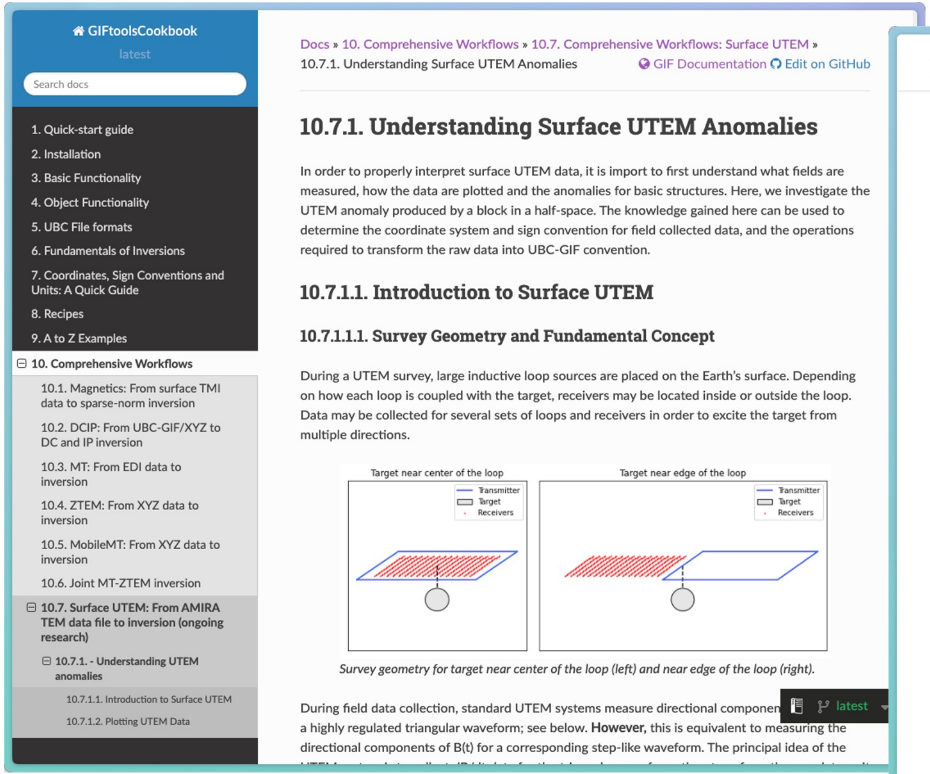
- standard L2
- different norms, PGI
- parametric approaches



https://simpeg.xyz

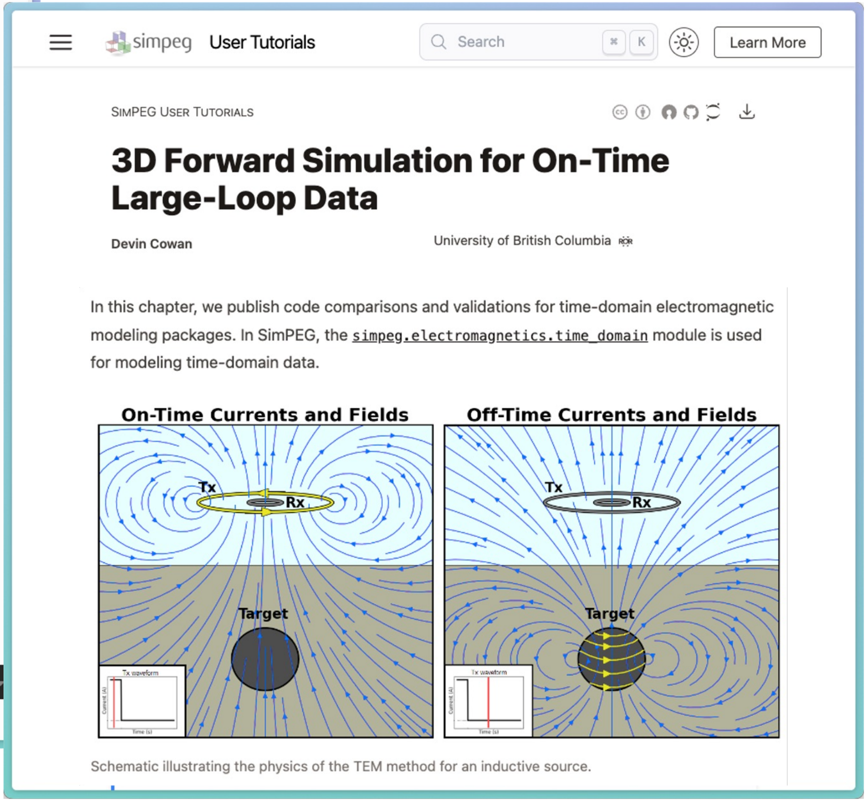
Resources

GIF tools cookbook



https://giftoolscookbook.readthedocs.io

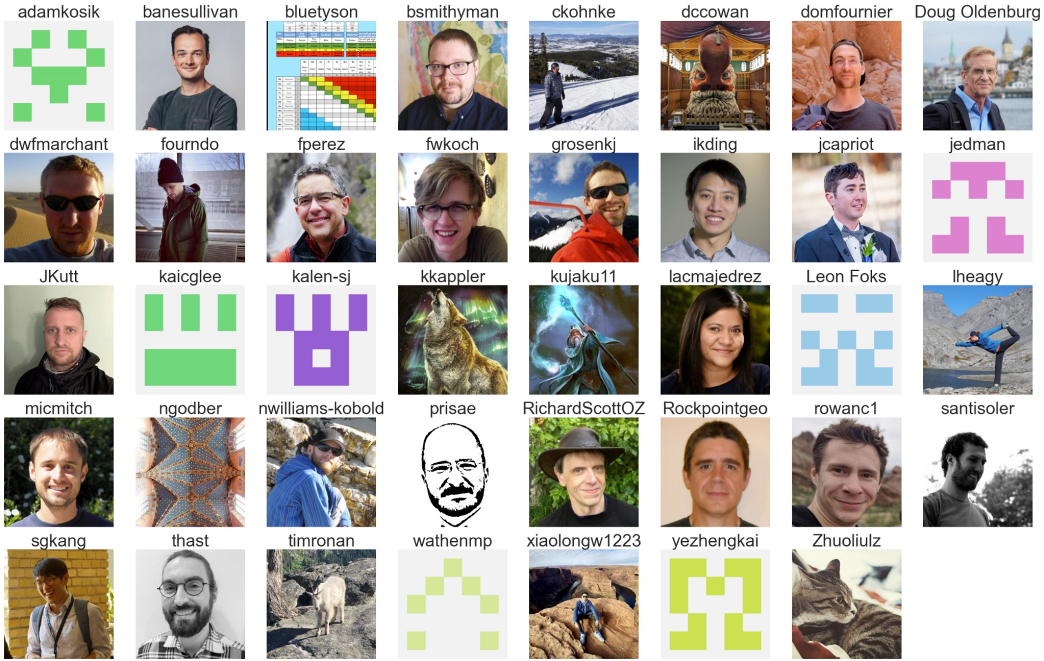
SimPEG user tutorials

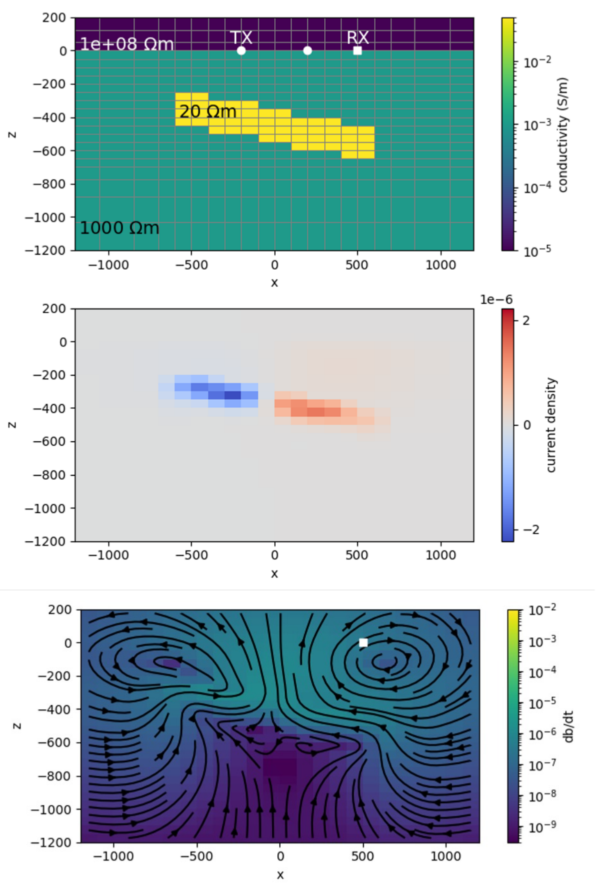


https://simpeg.xyz/user-tutorials

Thank you!

simpeg





lheagy@eoas.ubc.ca