capturing knowledge in code

simpeg and geosci

Lindsey Heagy, Seogi Kang, Doug Oldenburg
& the simpeg + geosci teams

collaborators

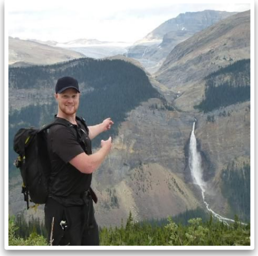
Seogi Kang
Stanford

Thibaut Astic
UBC GIF

Dom Fournier
Mira Geosciences

Joe Capriotti
UBC GIF

John Kuttai
Dias Geophysical



Devin Cowan
UBC GIF

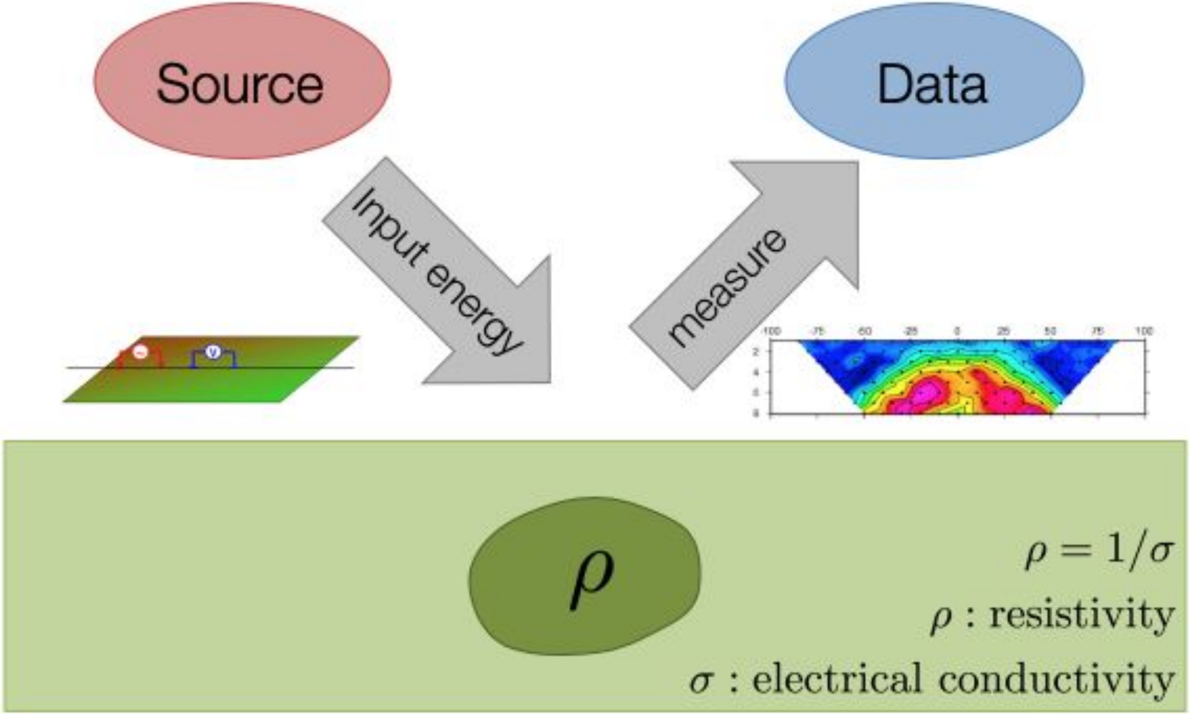
Rowan Cockett
Curvenote

Doug Oldenburg
UBC GIF

Last week:
Fundamentals of Inversion
Doug Oldenburg

http://www.mtnet.info/EMinars/EMinars.html

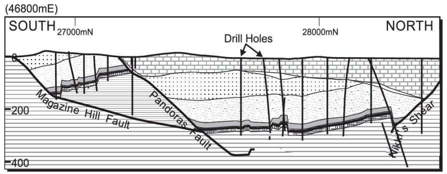
Geophysical experiments & physical properties

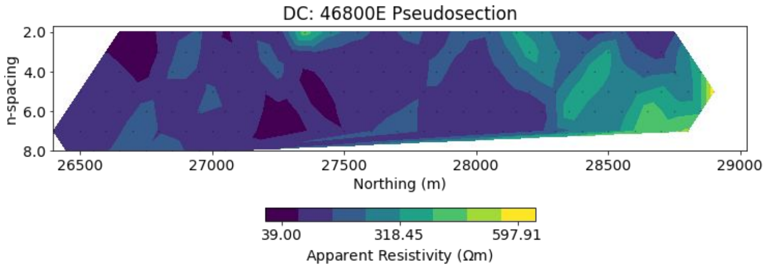


Last week: DC resistivity and IP at century

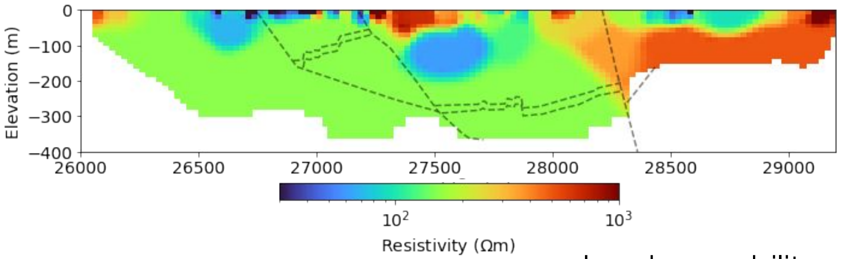
Last week: Century Deposit

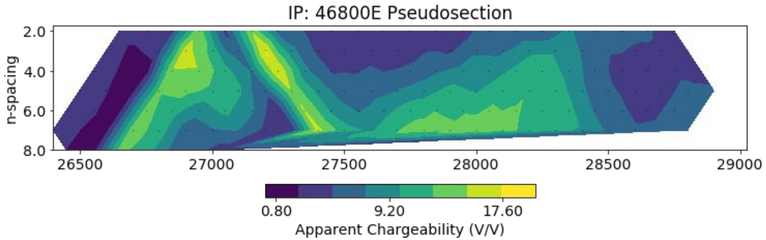
- IP: linear inverse problem
- DC: non-linear inverse problem



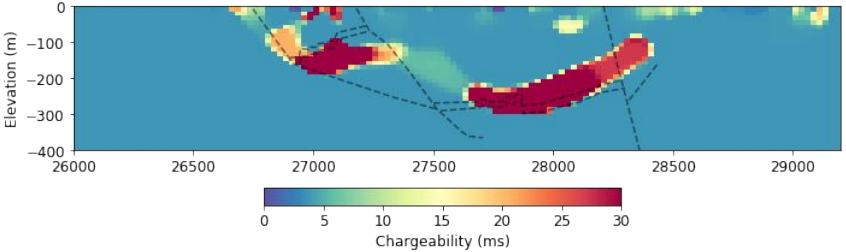


Lp resistivity





Lp chargeability



DC resistivity

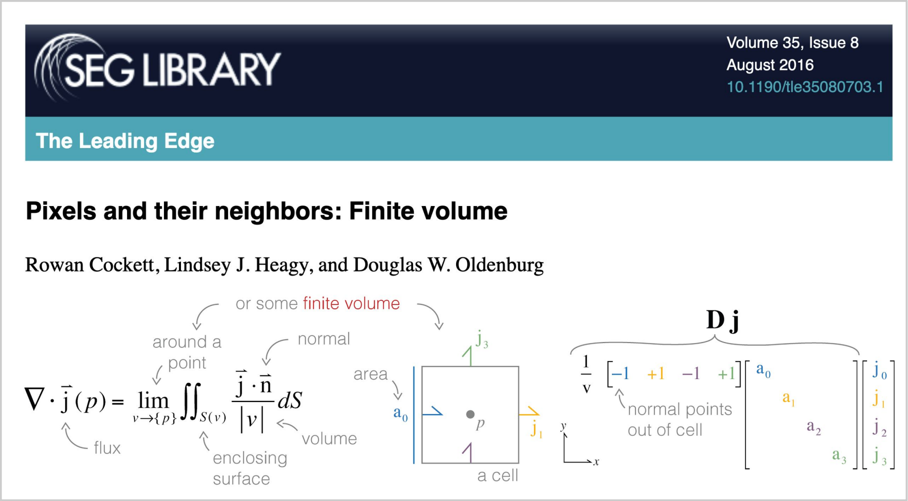
Governing PDE

$$\boxed{\nabla \cdot \frac{1}{\rho} \nabla \phi = -I\delta(r)}$$

DC is an illustrative problem

- foundation for the physics of EM
- poisson equation, starting point: numerical simulations, finite volume
- inverse problem: non-linear, ideal example for showing impacts of: model norm, constraints, …

Finite Volume Tutorial (Cockett et al, 2016)


SEG LIBRARY

Volume 35, Issue 8
August 2016
10.1190/tle35080703.1

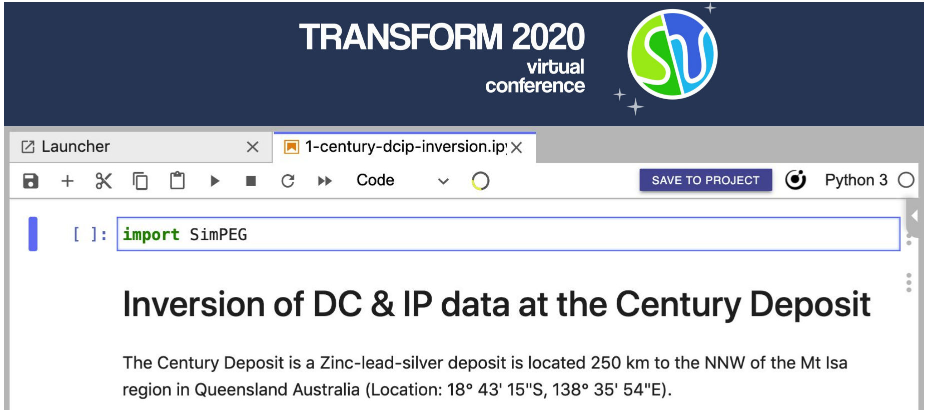
The Leading Edge

Pixels and their neighbors: Finite volume

Rowan Cockett, Lindsey J. Heagy, and Douglas W. Oldenburg

$$\nabla \cdot \vec{j}(p) = \lim_{v \to \{p\}} \iint_{S(v)} \frac{\vec{j} \cdot \vec{n}}{|v|} dS$$

Inversions with SimPEG (Heagy, 2020)




```
[ ]: import SimPEG
```

Inversion of DC & IP data at the Century Deposit

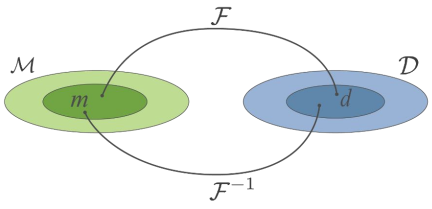
The Century Deposit is a Zinc-lead-silver deposit is located 250 km to the NNW of the Mt Isa region in Queensland Australia (Location: 18° 43' 15"S, 138° 35' 54"E).

Focus for today: electromagnetics

Theory: Maxwell's equations, inversion

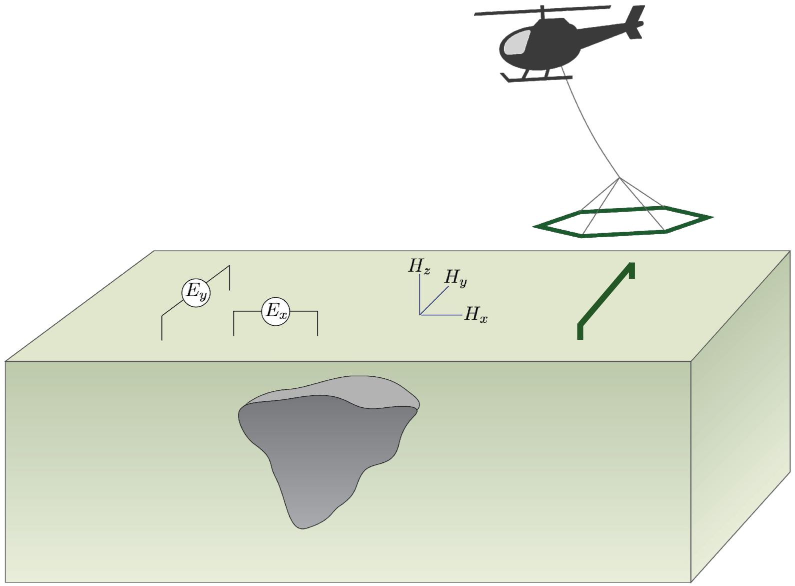
$$\nabla \times \vec{e} = -\frac{\partial \vec{b}}{\partial t}$$

$$\nabla \times \vec{h} = \vec{j} + \frac{\partial \vec{d}}{\partial t}$$

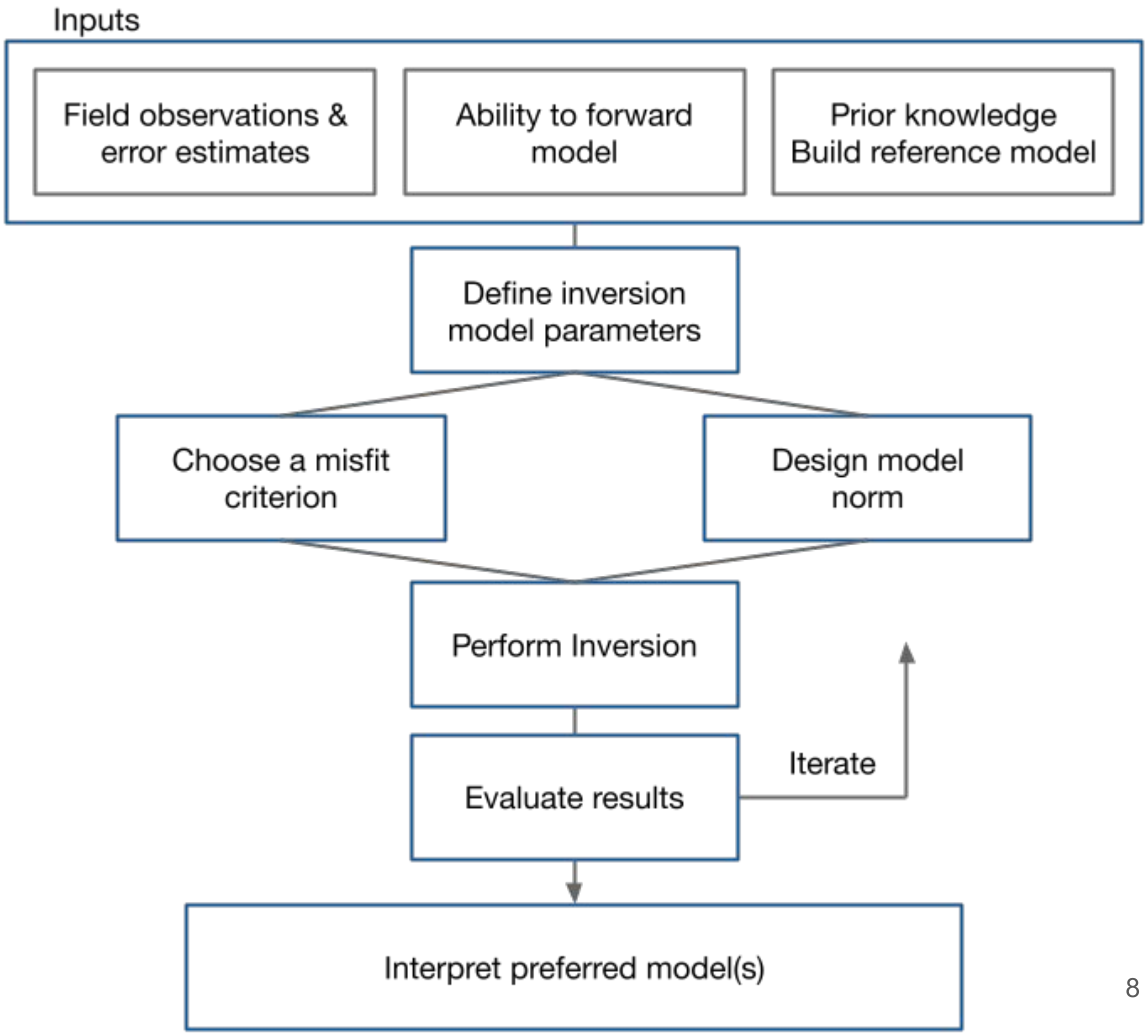


Captured in code

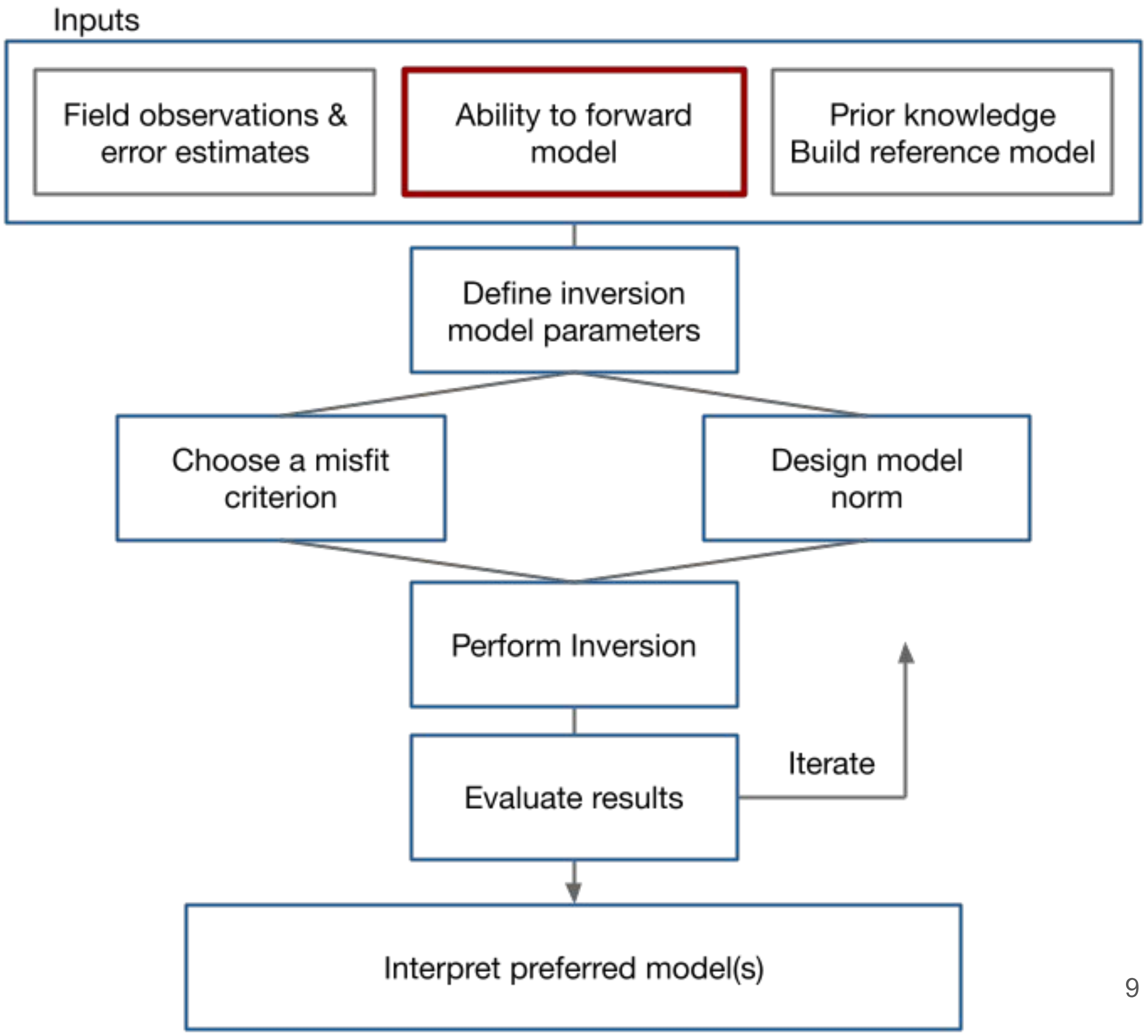
```
from SimPEG import electromagnetics
```



inversion flowchart



inversion flowchart



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

Electromagnetics: frequency domain equations

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

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

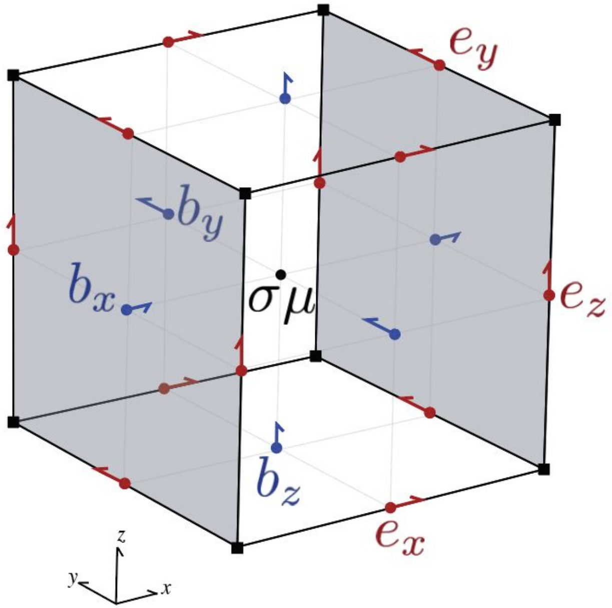
$$\nabla \times \mu^{-1}\vec{B} - \sigma\vec{E} = \vec{J}_s$$

Boundary conditions

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

Staggered grid discretization

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



Electromagnetics: frequency domain equations

Continuous equations

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

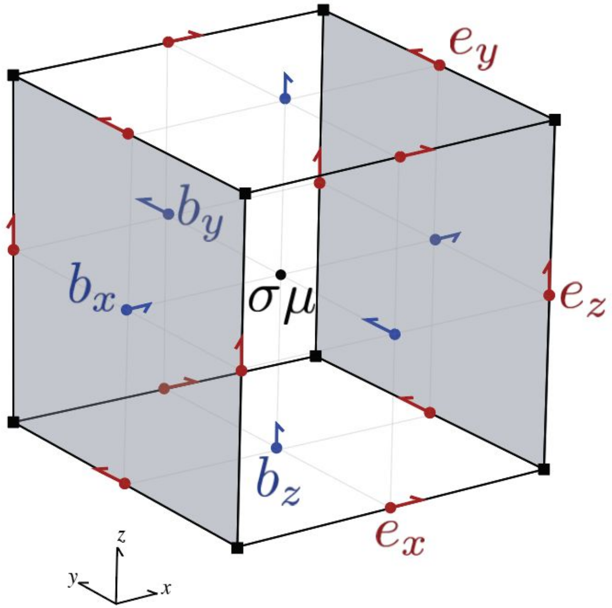
$$\nabla \times \mu^{-1} \vec{B} - \sigma \vec{E} = \vec{J}_s$$

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

Finite volume discretization (see: Haber, 2014; Cockett et al., 2016)

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

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



Electromagnetics: frequency domain equations

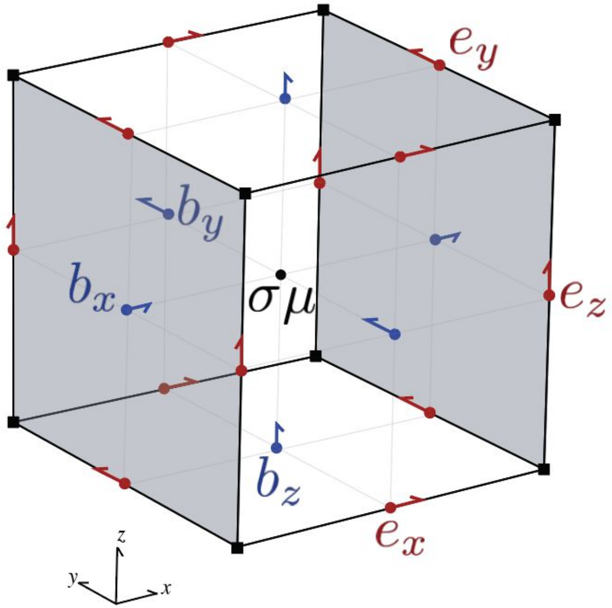
Discrete equations

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

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

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

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



Solving an FDEM problem

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

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

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

Solving an FDEM problem

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

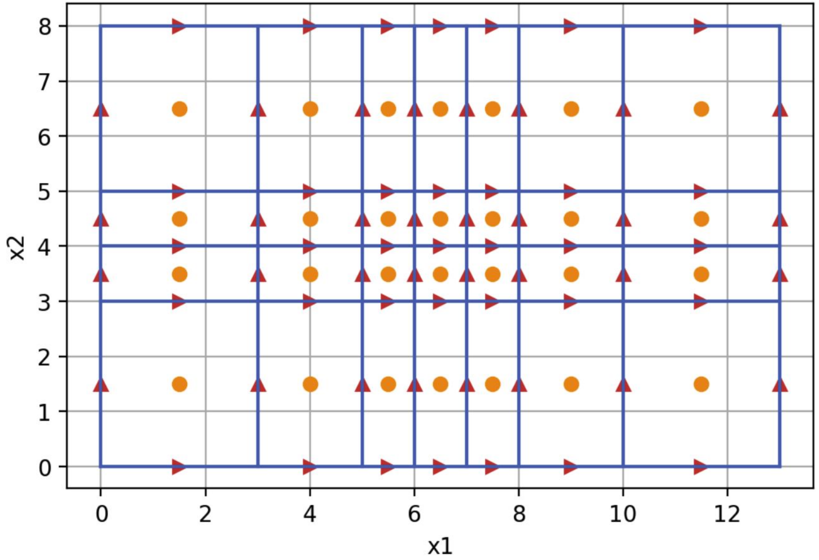
$$= \underbrace{-i\omega \mathbf{M}^e \mathbf{j_s}}_{\mathbf{q}(\omega)}$$

```
ω = 2 * pi * frequency

C = mesh.edge_curl
Mfμi = mesh.get_face_inner_product(1/mu_0)
Meσ = mesh.get_edge_inner_product(sigma)

A = C.T * Mfμi * C + i * ω * Meσ
Ainv = Solver(A)  # acts like A inverse

Me = mesh.get_edge_inner_product()
q = -i * ω * Me * js

u = Ainv * q
```

Create a mesh: the discretize package

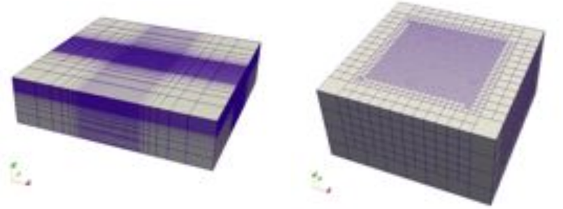
```python
import discretize
```

```python
hx = [3, 2, 1, 1, 1, 2, 3]
hy = [3, 1, 1, 3]
mesh = discretize.TensorMesh([hx, hy])
mesh.plot_grid(edges=True, centers=True);
```

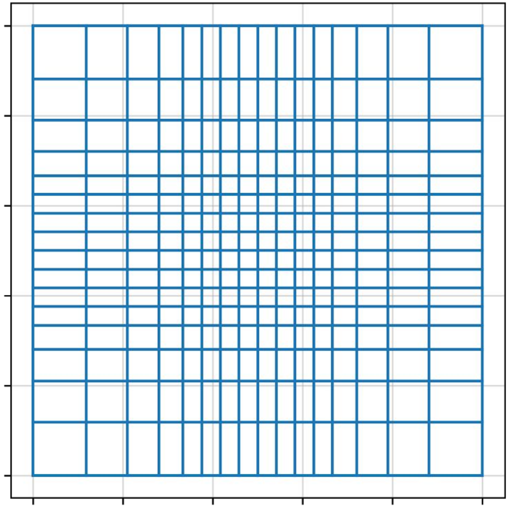


Properties or Methods
`dim, origin`
`n_cells, n_nodes, n_faces, n_edges`
`cell_volumes, face_areas, edge_lengths`
`cell_centers, nodes, edges, faces`
`nodal_gradient, face_divergence, edge_curl`
`average_edge_to_cell,` `average_node_to_cell, ...`
`get_edge_inner_product()`
`get_interpolation_matrix(location)`

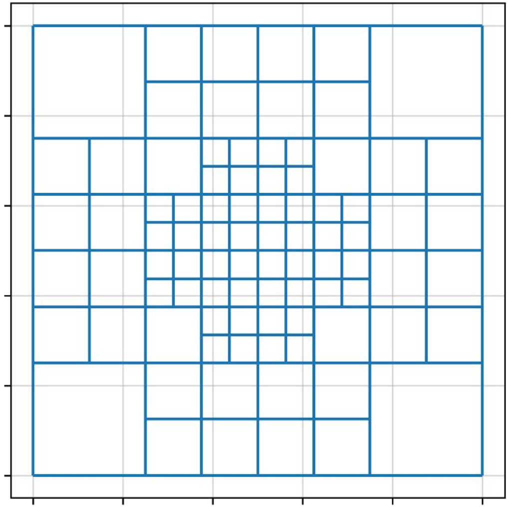
mesh types in simpeg



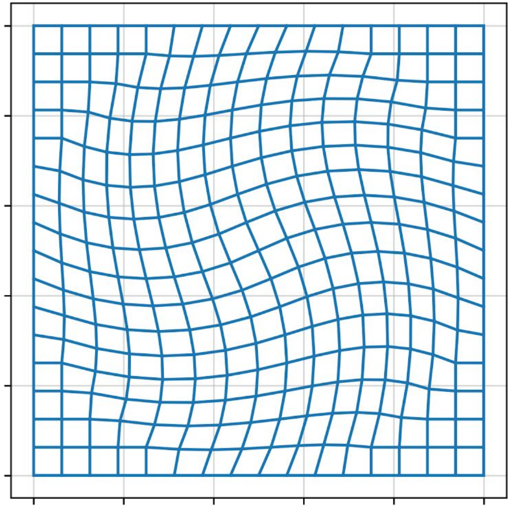
Tensor / Cylindrical



QuadTree / OcTree



Curvilinear



Joe
Capriotti

Survey: sources and receivers

Sources

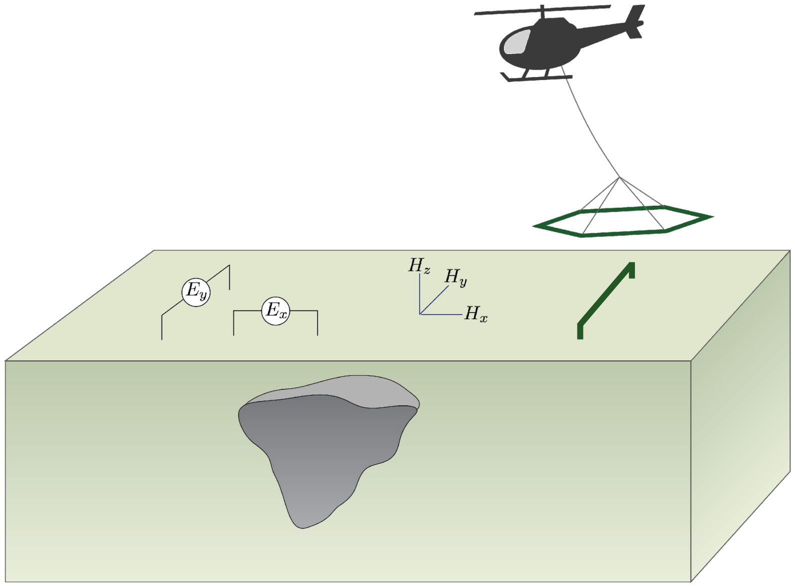
$$\underbrace{(\mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}} \mathbf{C} + i\omega \mathbf{M}^e_\sigma)}_{\mathbf{A}(\sigma,\omega)} \underbrace{\mathbf{e}}_{\mathbf{u}} = \underbrace{-i\omega \mathbf{M}^e \mathbf{j_s}}_{\mathbf{q}(\omega)}$$

- inductive
- grounded

Receivers

- electric
- magnetic
- interpolate to locations

$$\mathbf{d}^{\text{pred}} = \mathbf{P}(\mathbf{u}, \omega)$$




```
source_list = [
    fdem.sources.MagDipole(receiver_list, f, location)
    for f in frequencies
]
survey = fdem.survey(source_list)
```

Bring it all together: simulation

```
from SimPEG.electromagnetics import frequency_domain as fdem
```

```
simulation = fdem.Simulation3DElectricField(
    mesh=mesh, survey=survey, solver=Pardiso,
    sigmaMap=maps.IdentityMap()
)
```

For each frequency, solve

$$\mathbf{A}(\sigma, \omega)\mathbf{u} = \mathbf{q}(\omega)$$

```
u = simulation.fields(model)
```

Project to receiver locations

$$\mathbf{d}^{\text{pred}} = \mathbf{P}(\mathbf{u}, \omega)$$

```
dpred = simulation.dpred(model, fields=u)
```

Sensitivities (we will need them later!)

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

Demo

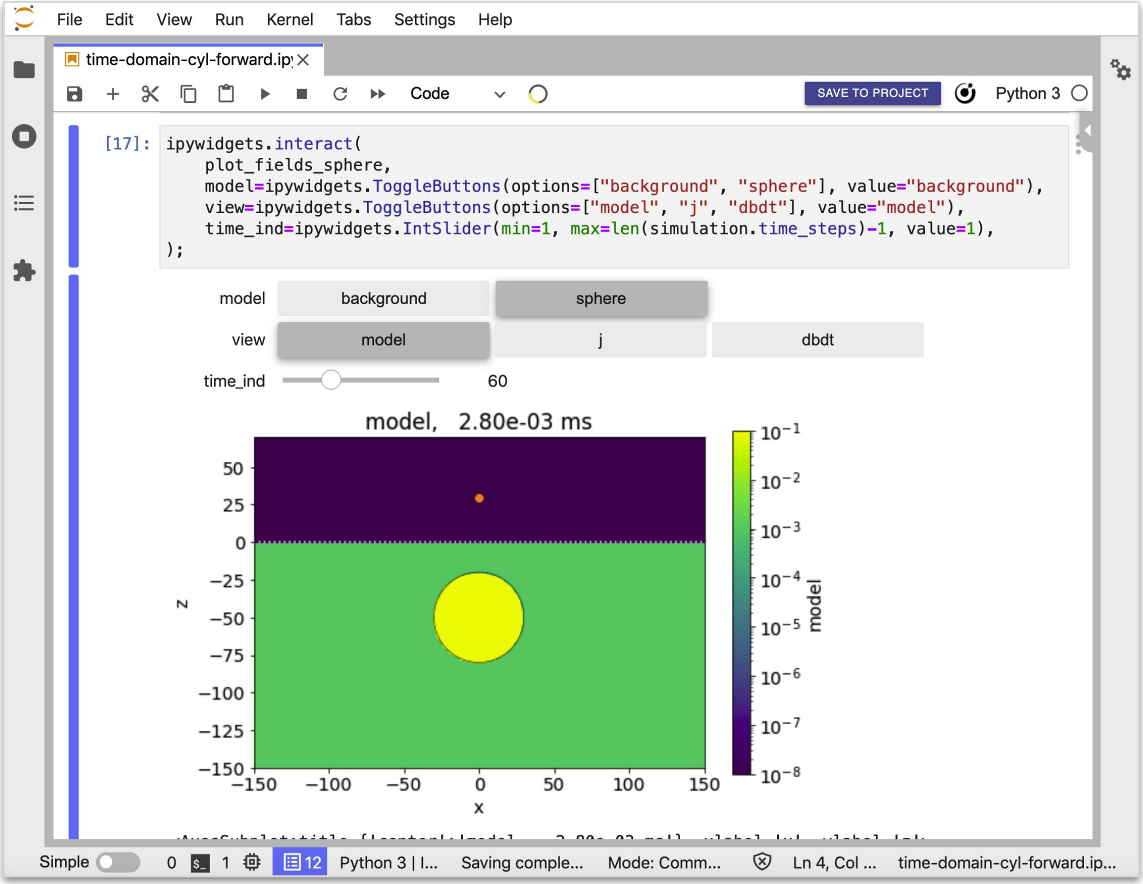
conductive sphere in a halfspace

cylindrical mesh

time-domain

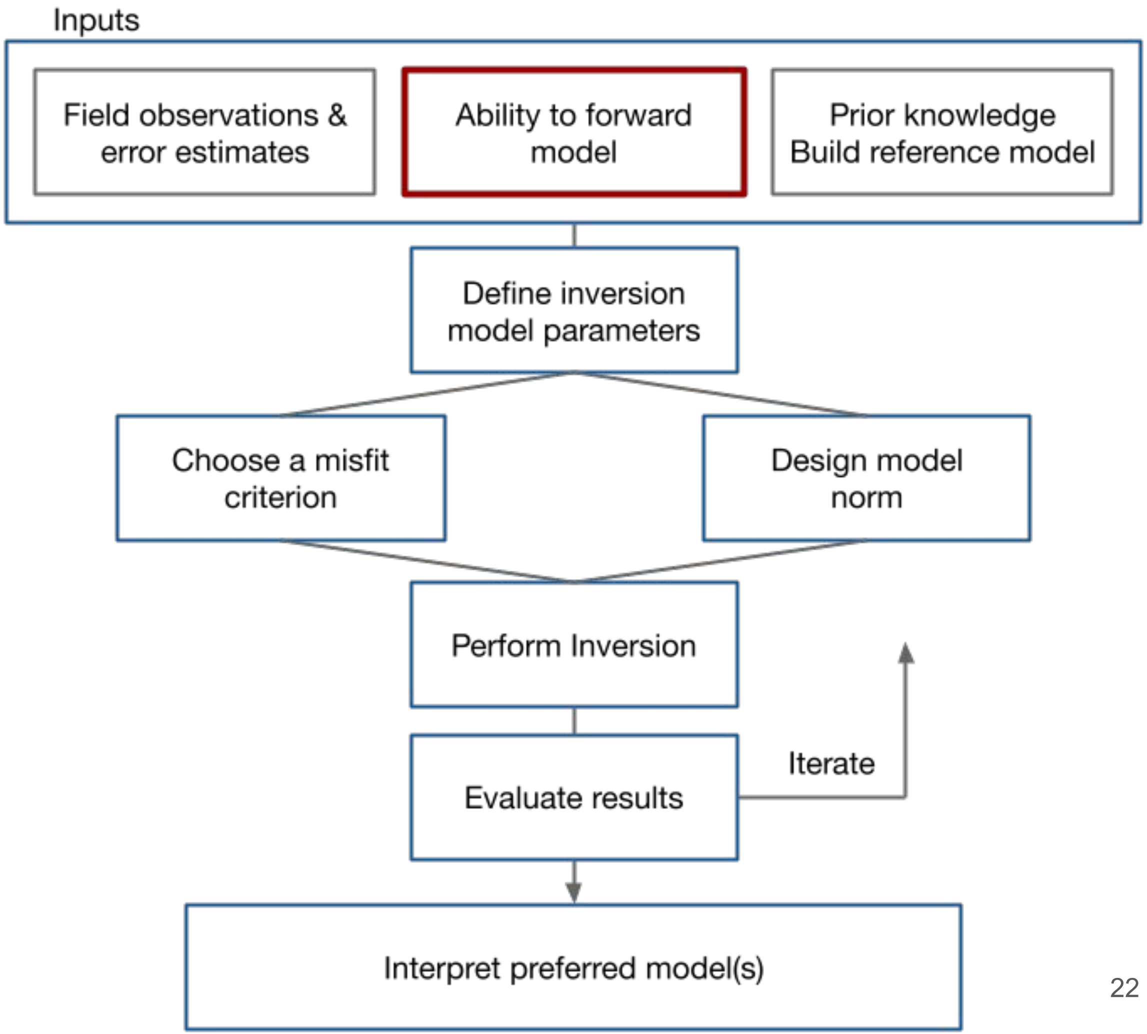
$$\nabla \times \vec{e} = -\frac{\partial \vec{b}}{\partial t}$$

$$\nabla \times \vec{h} = \vec{j}$$

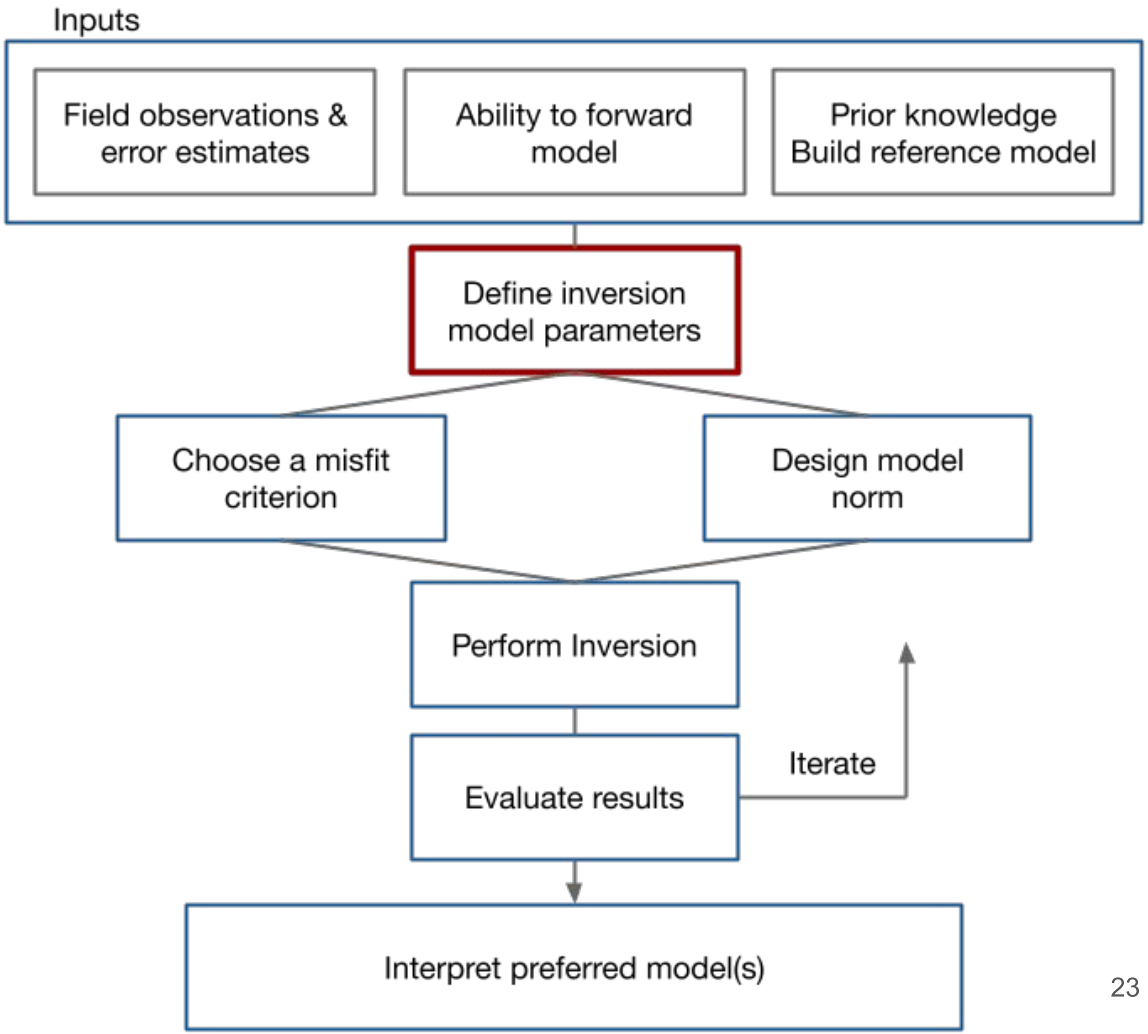


curvenote.com/@geosci/inversion-module

inversion flowchart

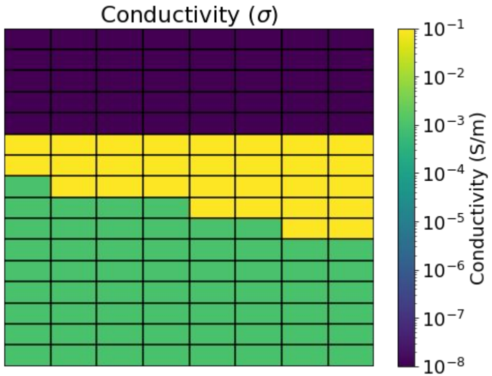


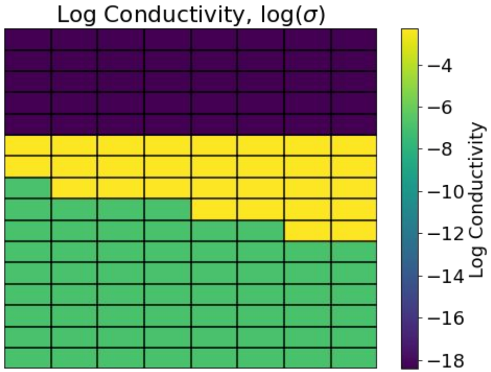
inversion flowchart

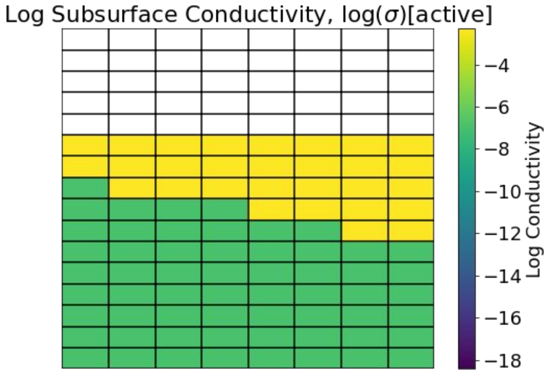


Inversion model parameters & mappings

What parameters are we inverting for?

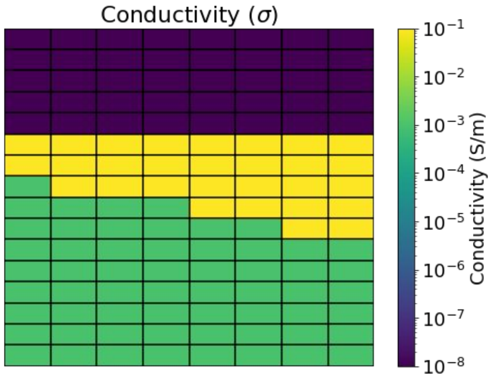


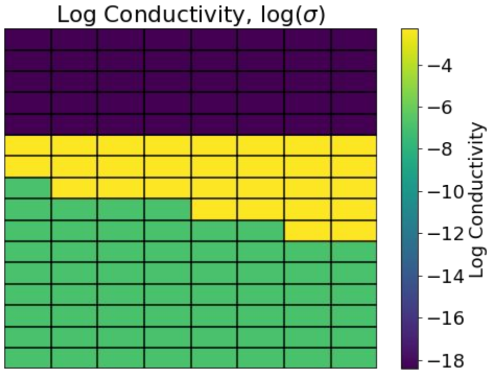


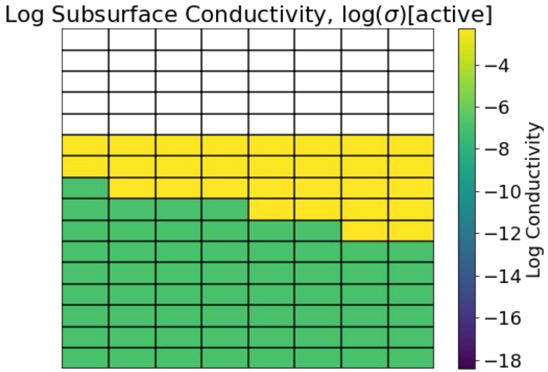


Inversion model parameters & mappings

What parameters are we inverting for?







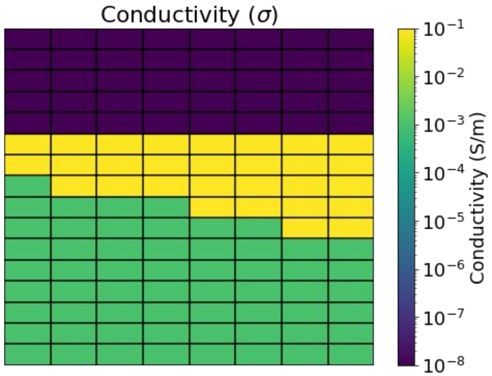
σ $\mathbf{m}$

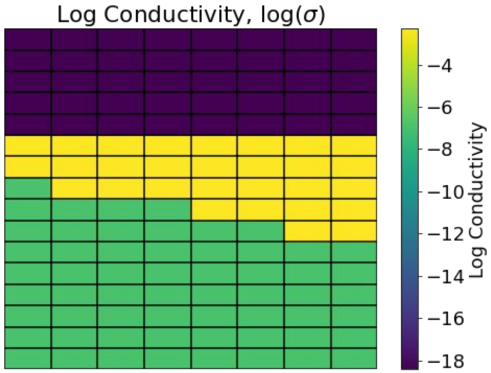
a mapping translates model parameters to physical properties on simulation mesh

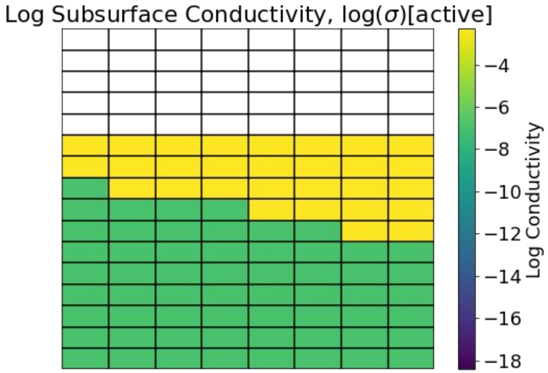
$$\sigma = \mathcal{M}(\mathbf{m})$$

Inversion model parameters & mappings

What parameters are we inverting for?





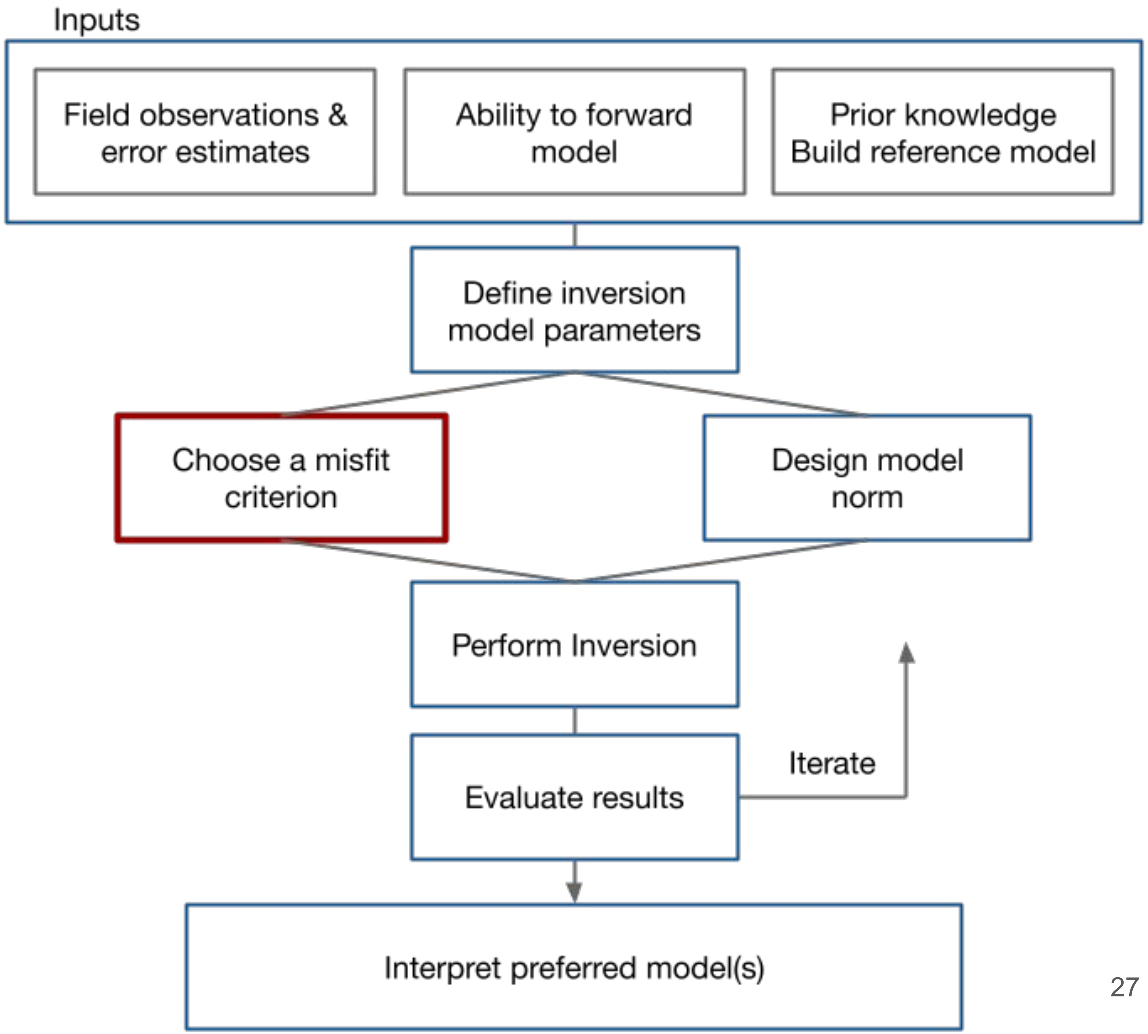


σ ← `maps.ExpMap` ← `maps.InjectActiveCells` ← $\mathbf{m}$

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

- Mappings can be composed
- Includes parametric models
- Keep track of derivatives (for sensitivities)

inversion flowchart



observed data, uncertainties, and data misfit

Data misfit term

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

uncertainties captured in W matrix

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

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

observed data, uncertainties, and data misfit

Data misfit term

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

uncertainties captured in W matrix

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

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

Data class: survey geometry, observed data, assigned uncertainties

```
from SimPEG import Data
data = Data(survey, dobs, relative_error, floor)
```

Data misfit instantiated with

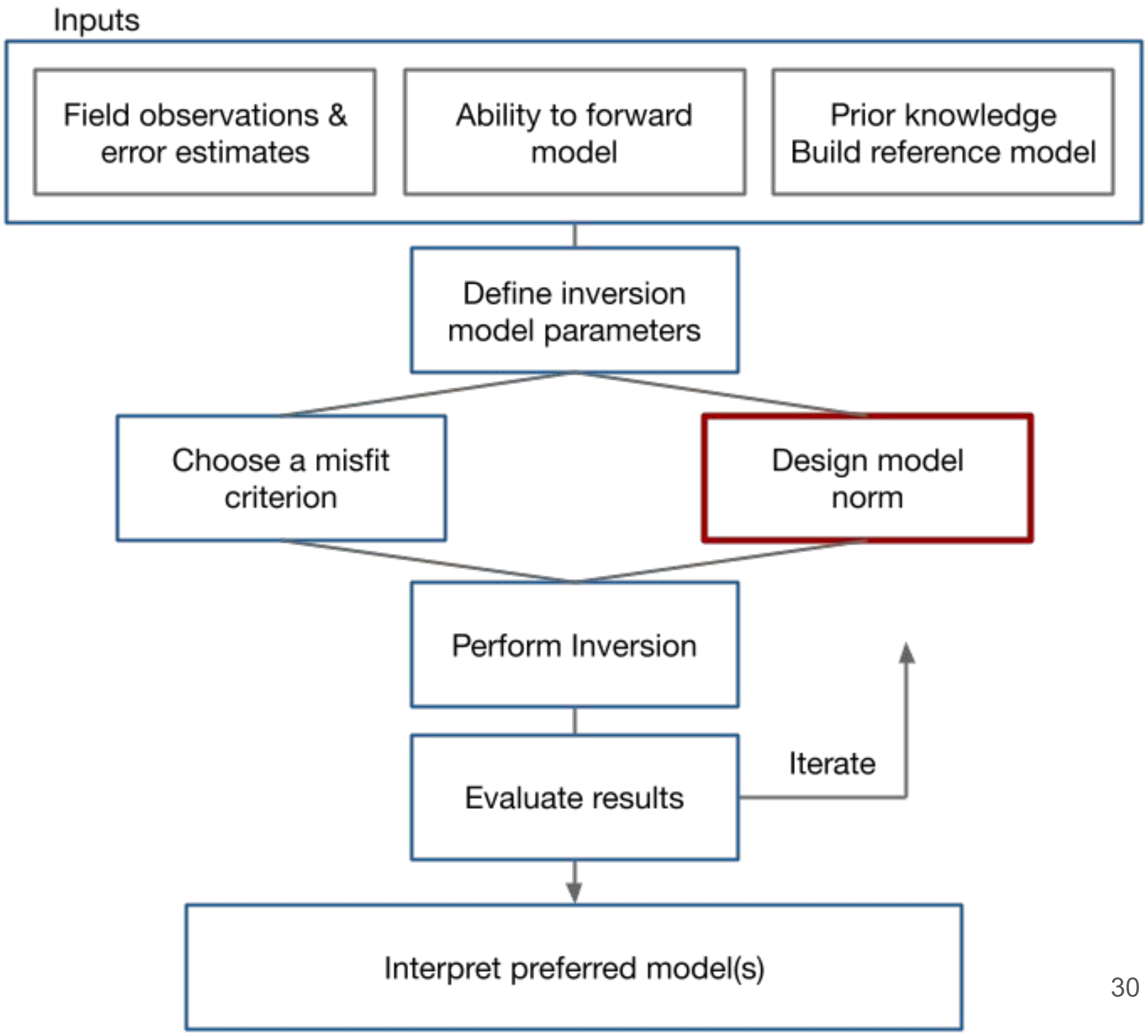
- simulation: to compute $\mathcal{F}(\mathbf{m})$
- data: defines $\mathbf{W}_d$, $\mathbf{d}^{\text{obs}}$

```
from SimPEG import data_misfit
phi_d = data_misfit.L2DataMisfit(data, simulation)
```

can now evaluate data misfit + derivatives

```
phi_d(m), phi_d.deriv(m), phi_d.deriv2(m, v)
```

inversion flowchart



Designing a model norm: regularization class

Basic Tikhonov regularization

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

discretize

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

Choices:

- α - parameter values
- reference model
- mref in the smoothness terms
- norm applied on each term

Designing a model norm: regularization class

Basic Tikhonov regularization

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

discretize

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

Choices:

- α - parameter values
- reference model
- mref in the smoothness terms
- norm applied on each term

Regularization instantiated with

- mesh: to evaluate spatial derivs
- alphas, mref have default values, can be replaced with user values

```
from SimPEG import regularization
```

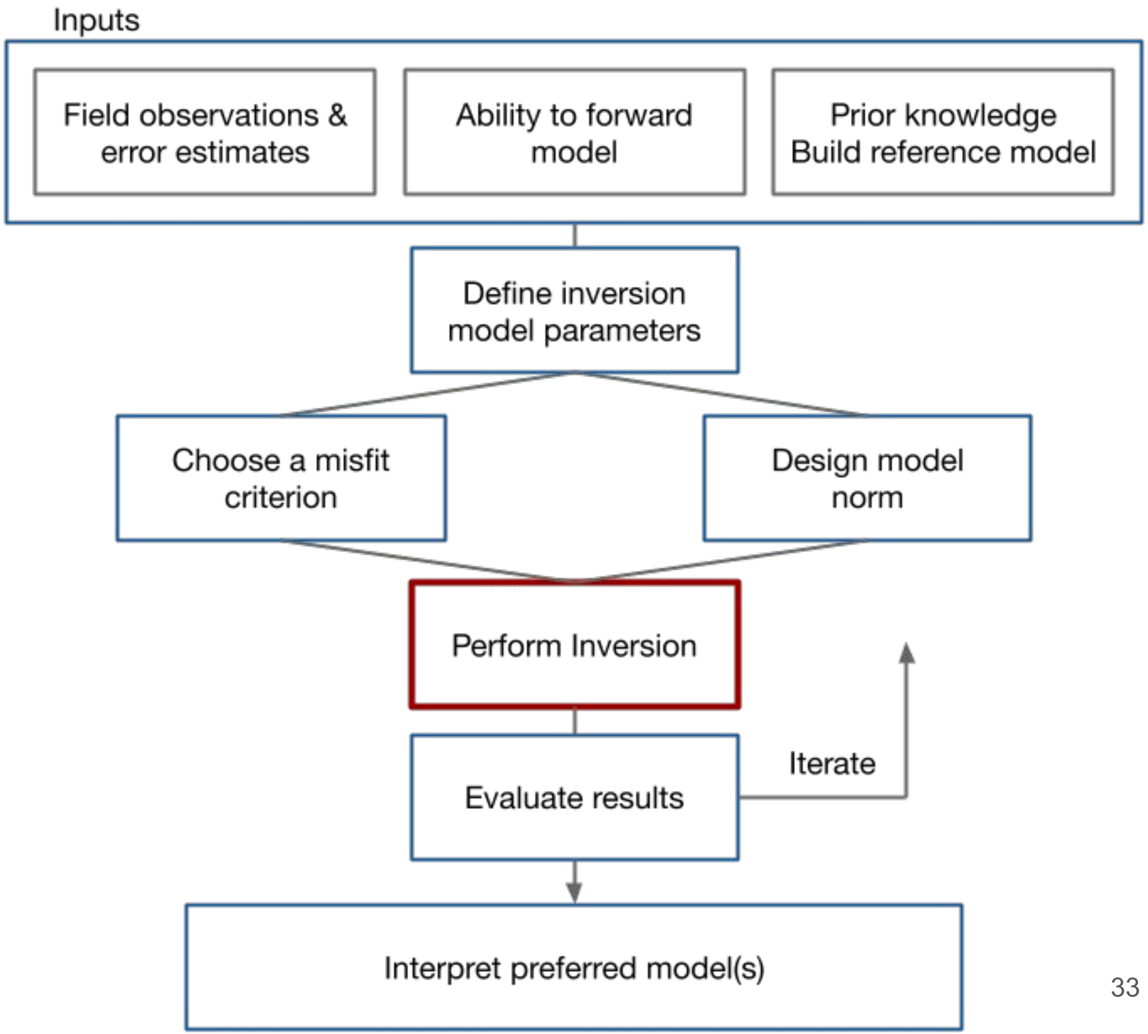
```
phi_m = regularization.Tikhonov(
    mesh, mref=mref, alpha_s=alpha_s, alpha_x=alpha_x
)
```

```
phi_m_sparse = regularization.Sparse(
    mesh, mref=mref, norms=[1, 1]
)
```

can now evaluate phi_m + derivatives

```
phi_m(m), phi_m.deriv(m), phi_m.deriv2(m, v)
```

inversion flowchart



Perform the inversion: stating the objective function

Inversion as an optimization problem (deterministic approach)

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

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

At this stage, we have specified

- parameters we are inverting for
- data misfit
- model norm

Perform the inversion: stating the objective function

Inversion as an optimization problem (deterministic approach)

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

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

Still to define

- optimization method
- upper and lower bounds
- choice of initial beta
- choice of beta-cooling schedule
- target misfit and stopping the inversion

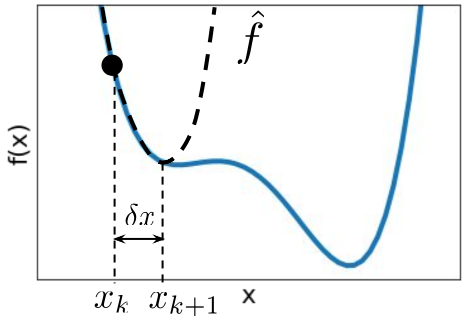
Perform the inversion: optimization approach

Inversion as an optimization problem (deterministic approach)

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

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

Second-order methods




```
from SimPEG import optimization
```

```
opt = optimization.InexactGaussNewton(maxIter=20, maxIterCG=20)
```

```
opt_with_bounds = optimization.ProjectedGNCG(lower=0)
```

Perform the inversion: inversion directives

Inversion as an optimization problem (deterministic approach)

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

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

We use **directives** to make parameter updates and orchestrate the inversion, e.g.

- estimating initial beta
- defining a beta-cooling schedule
- stopping the inversion when target misfit reached

Perform the inversion: inversion directives

Inversion as an optimization problem (deterministic approach)

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

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

Initial beta

- estimate "size" of data misfit and model norm by approximating eigenvalues of

$$\mathbf{J}^\top \mathbf{W}_d^\top \mathbf{W}_d \mathbf{J}, \ \mathbf{W}_m^\top \mathbf{W}_m$$

- take ratio, weight by a parameter controlling relative importance of each

```
from SimPEG import directives
beta0 = directives.BetaEstimate_ByEig(beta0_ratio=100)
```

Perform the inversion: inversion directives

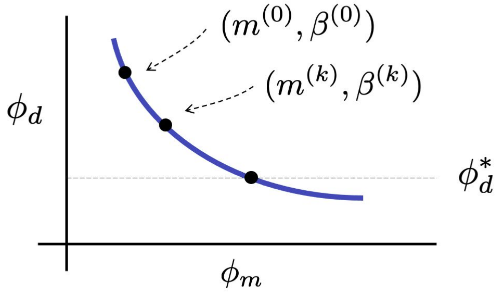
Inversion as an optimization problem (deterministic approach)

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

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

Beta-cooling

- Define how often beta is reduced (every N iterations)
- Define how much beta is reduced by




```
beta_cooling = directives.BetaSchedule(coolingRate=2, coolingFactor=4)
```

Perform the inversion: inversion directives

Inversion as an optimization problem (deterministic approach)

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

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

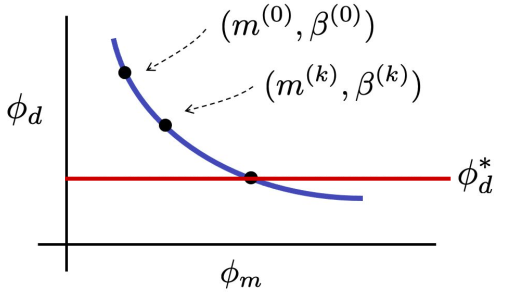
Target misfit

- Expected value of data misfit

$$E[\phi_d] \simeq N$$

- Define target misfit as (default χ= 1)

$$\phi_d^* = \chi N$$




```
target_misfit = directives.TargetMisfit(chifact=1)
```

Perform the inversion: inversion directives

Inversion as an optimization problem (deterministic approach)

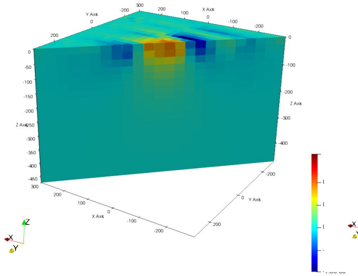
$$\min_{\mathbf{m}} \phi(\mathbf{m}) = \phi_d(\mathbf{m}) + \beta \phi_m(\mathbf{m})$$

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

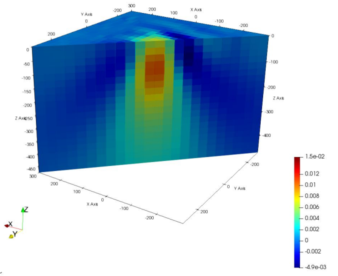
Other uses for directives

- saving inversion model at each iteration
- saving inversion progress (beta, data misfit, ...)
- including / updating sensitivity weighting
- updating values for norms (L2 → Lp)

Recovered susceptibility



Sensitivity weighted susceptibility



Perform the inversion: bringing it all together

Inversion as an optimization problem (deterministic approach)

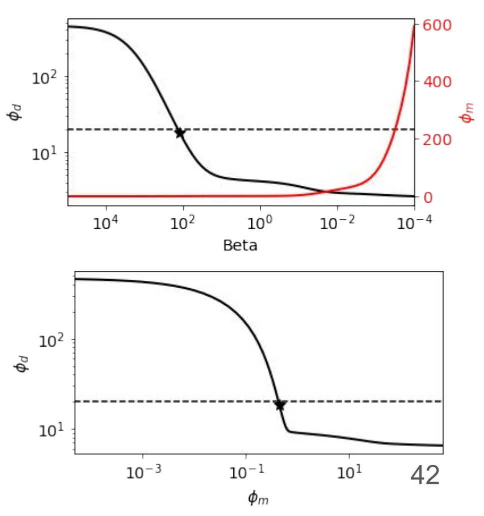
$$\min_{\mathbf{m}} \phi(\mathbf{m}) = \phi_d(\mathbf{m}) + \beta\phi_m(\mathbf{m})$$

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

```python
from SimPEG import inverse_problem, inversion
```

```python
inv_prob = inverse_problem.BaseInvProblem(phi_d, phi_m, opt)
inv = inversion.BaseInversion(inv_prob, [beta0, beta_cooling, target_misfit])
```

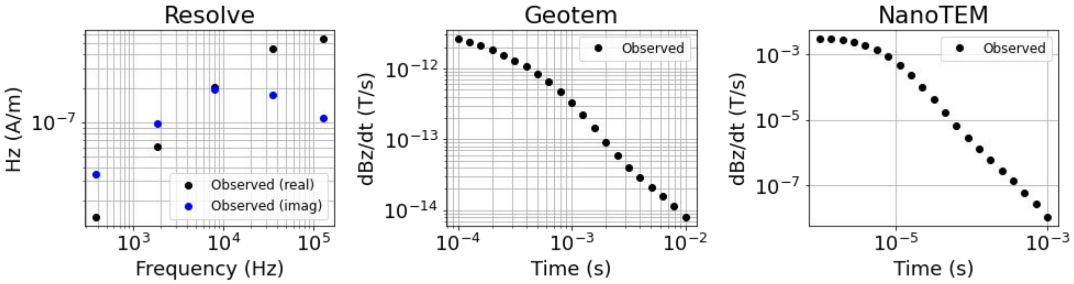
```python
inv.run(m0)
```

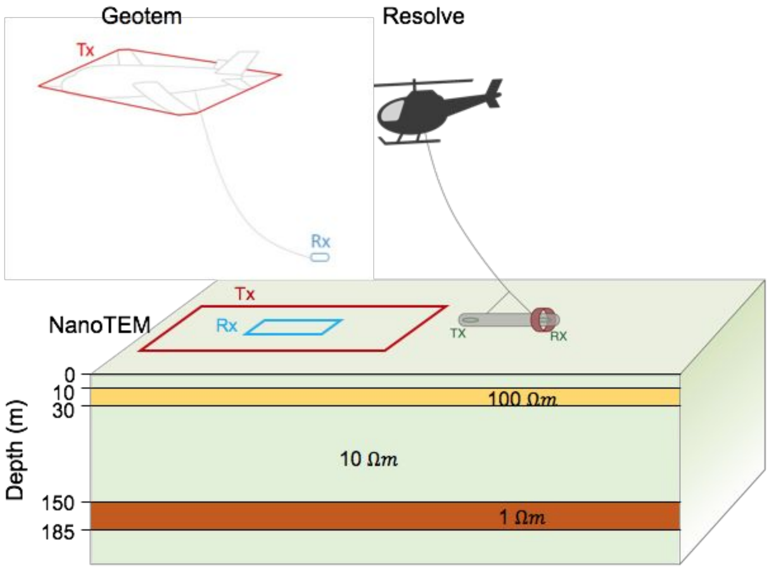


An example: 1D inversions

Layered earth, 3 different EM systems

- Resolve (airborne, frequency)
- Geotem (airborne, time-domain)
- NanoTEM (ground, time-domain)



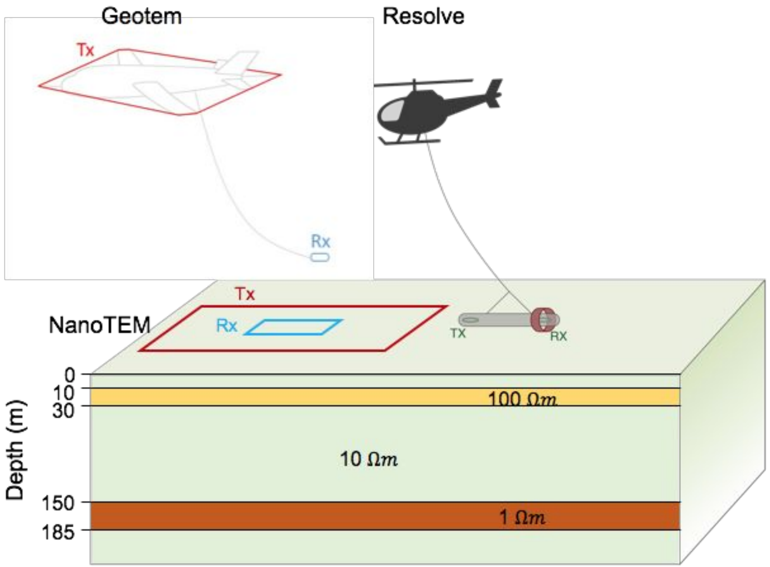


(Oldenburg et al, 2020)

SimPEG EM1D

Seogi Kang

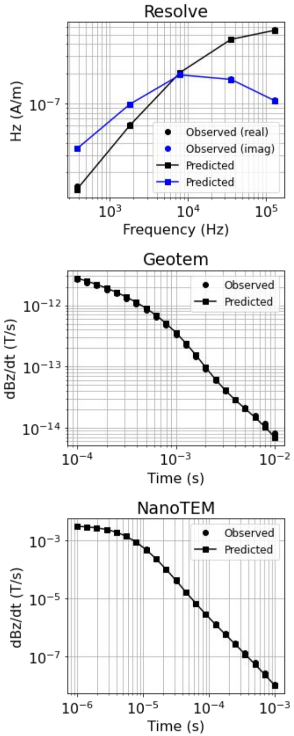
- Efficient forward simulation, sensitivity calculation using digital filters
    - relies on empymod (Werthmüller, 2017)
- Parallelized over soundings
- Common FDEM, TDEM system parameters implemented

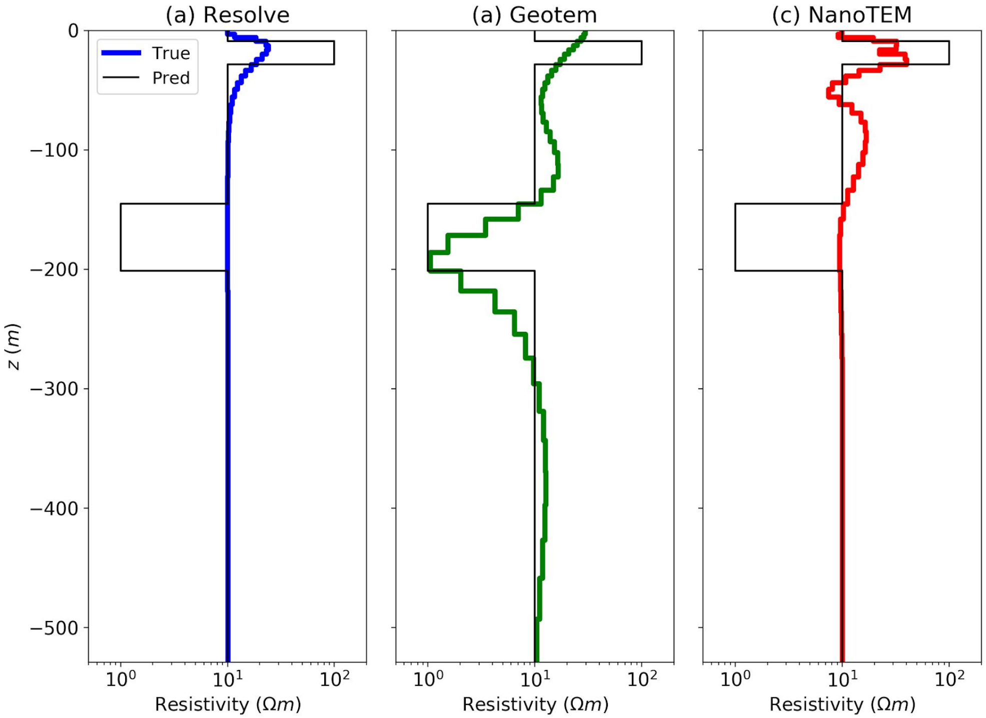


(Kang et al, 2018)

Individual inversions

L2 regularization



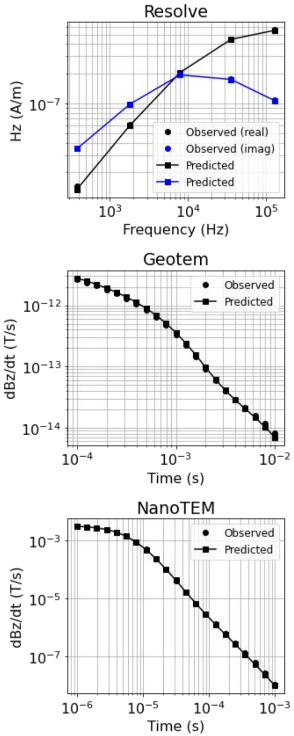


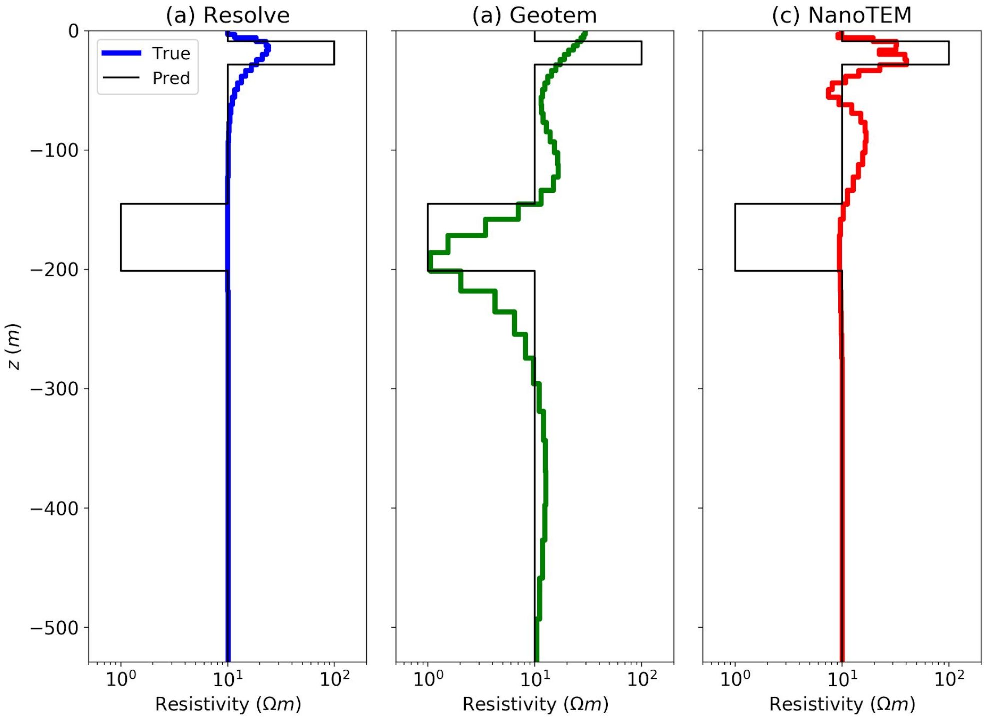
Joint Inversion

$$\phi(\mathbf{m}) = \underbrace{\phi_d^{\text{Resolve}} + \phi_d^{\text{Geotem}} + \phi_d^{\text{NanoTEM}}}_{\phi_d(\mathbf{m})} + \beta\phi_m(\mathbf{m})$$

L2 regularization

```
phi_d = phi_d_resolve + phi_d_geotem + phi_d_nanotem
```



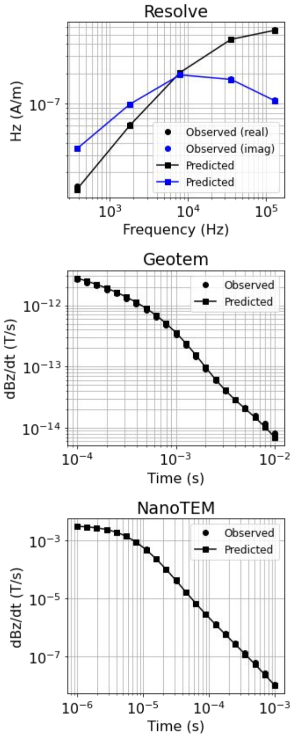


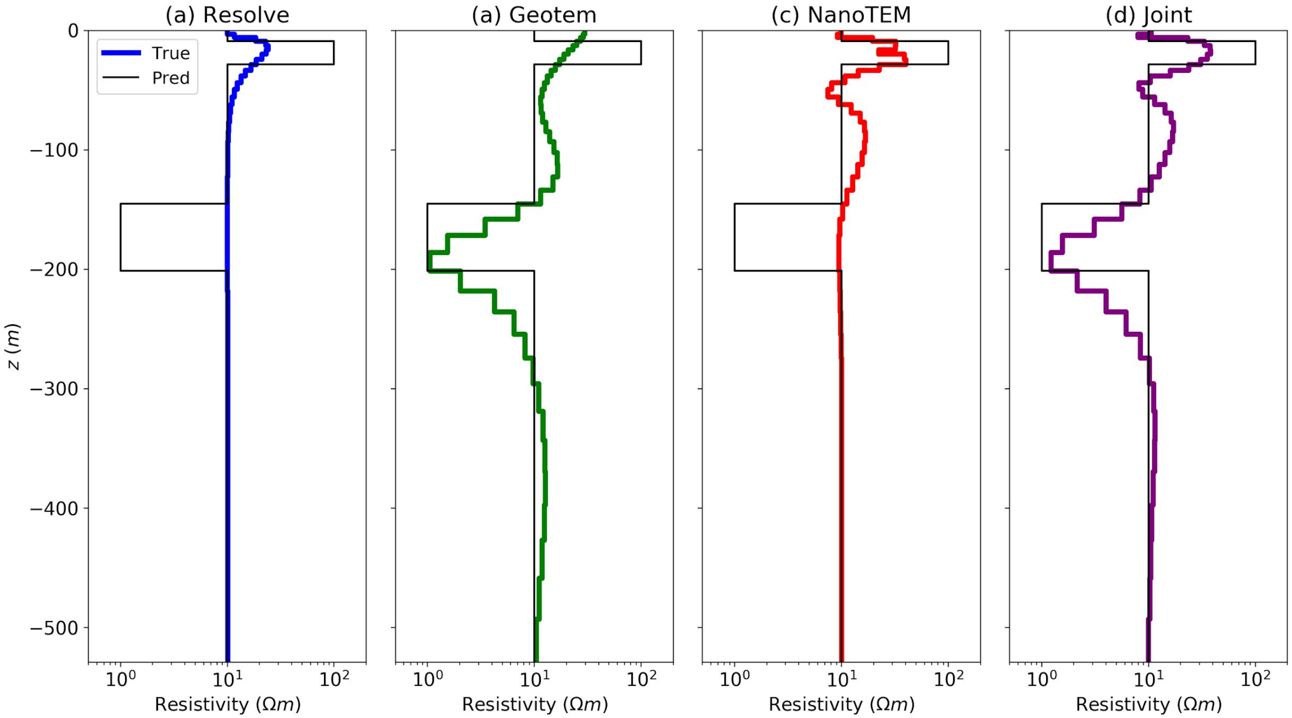
Joint Inversion

$$\phi(\mathbf{m}) = \underbrace{\phi_d^{\text{Resolve}} + \phi_d^{\text{Geotem}} + \phi_d^{\text{NanoTEM}}}_{\phi_d(\mathbf{m})} + \beta\phi_m(\mathbf{m})$$

L2 regularization

```
phi_d = phi_d_resolve + phi_d_geotem + phi_d_nanotem
```



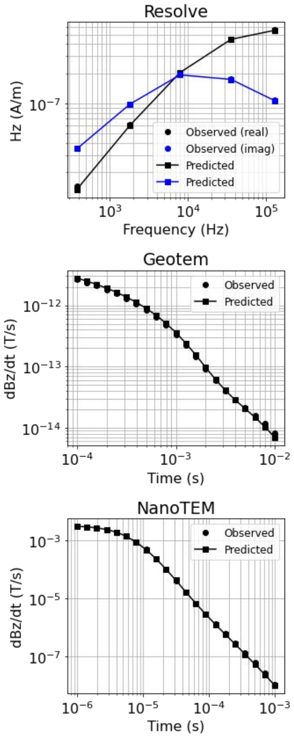


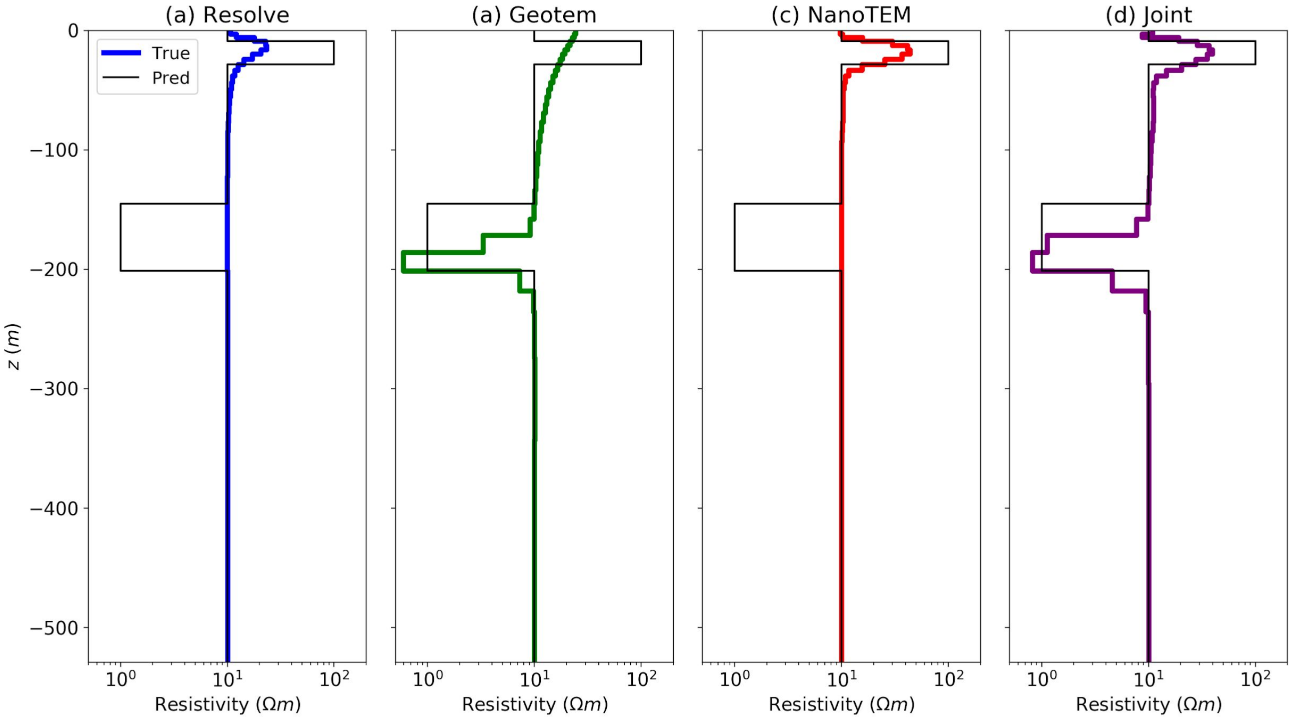
Joint Inversion

$$\phi(\mathbf{m}) = \underbrace{\phi_d^{\text{Resolve}} + \phi_d^{\text{Geotem}} + \phi_d^{\text{NanoTEM}}}_{\phi_d(\mathbf{m})} + \beta\phi_m(\mathbf{m})$$

L0 regularization

```
phi_d = phi_d_resolve + phi_d_geotem + phi_d_nanotem
```



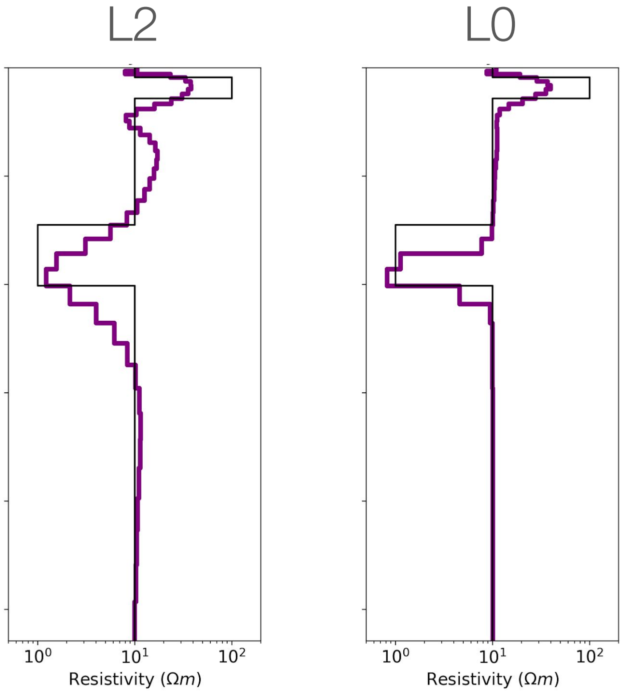


Joint Inversion

Flexibility to handle:

- multiple surveys / physics
- different model parameterizations
- different simulation mesh for each datum
- separate forward simulation and inversion meshes

```
phi_d = phi_d_resolve + phi_d_geotem + phi_d_nanotem
```



Example: Bookpurnong

Murray River Floodplain

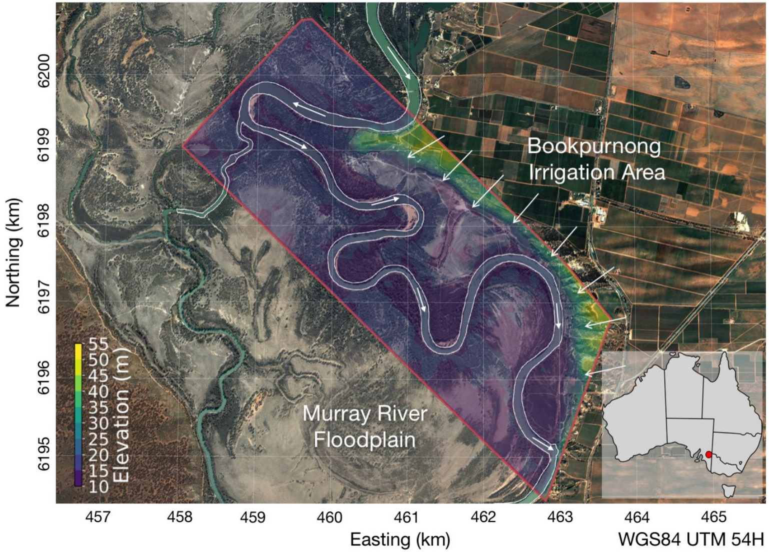
- over-irrigation and drought
- saline water recharges river
- floodplain salinization

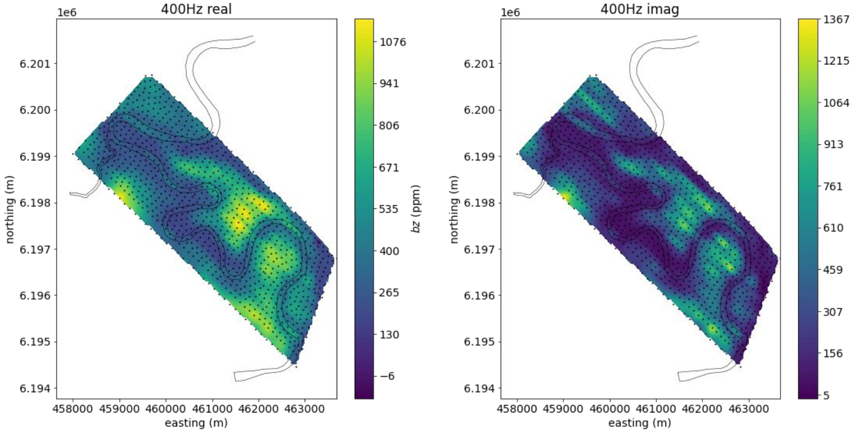
Data

- 2006: SkyTEM (time-domain)
- 2008: RESOLVE (frequency-domain)

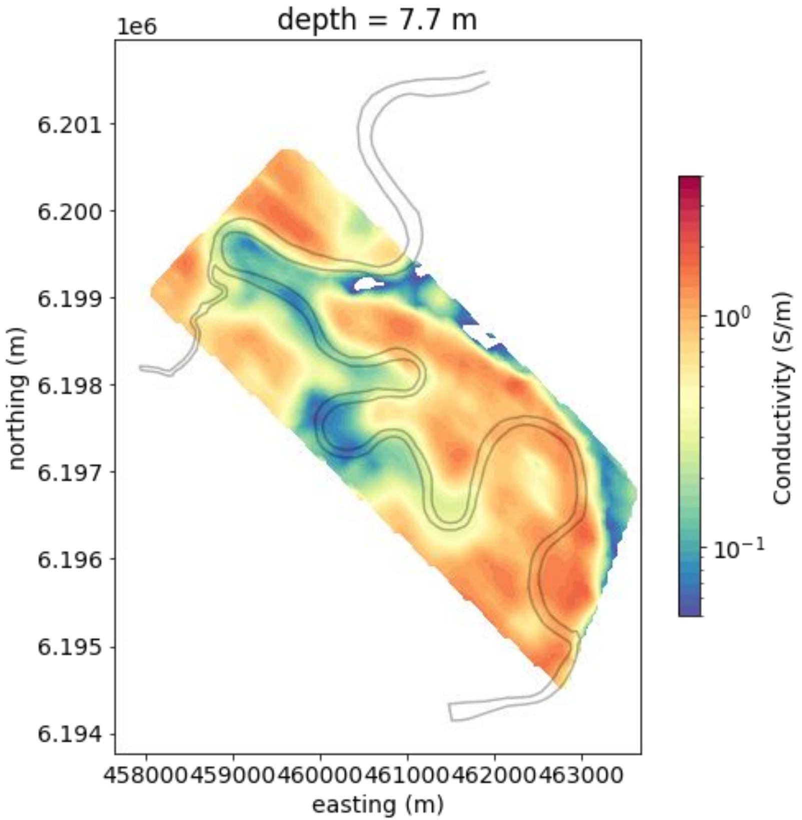
Inversion

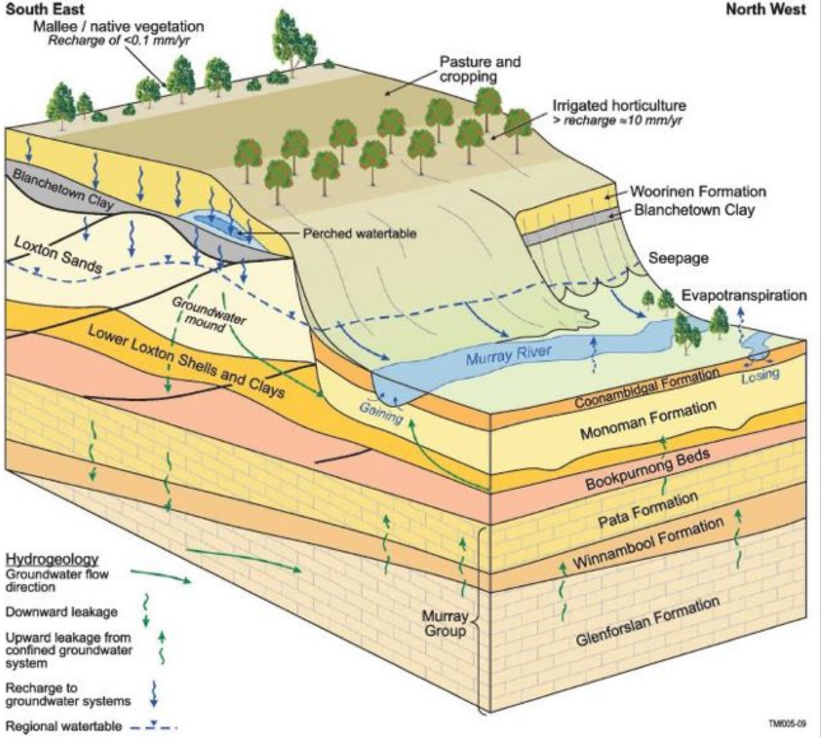
- Spatially constrained 1D





Example: Bookpurnong

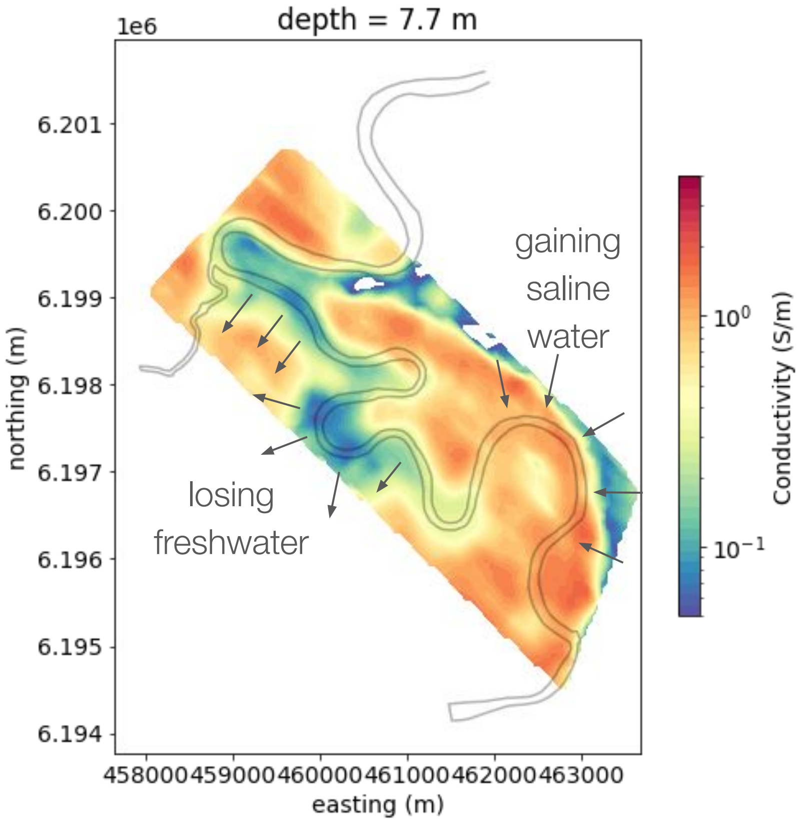




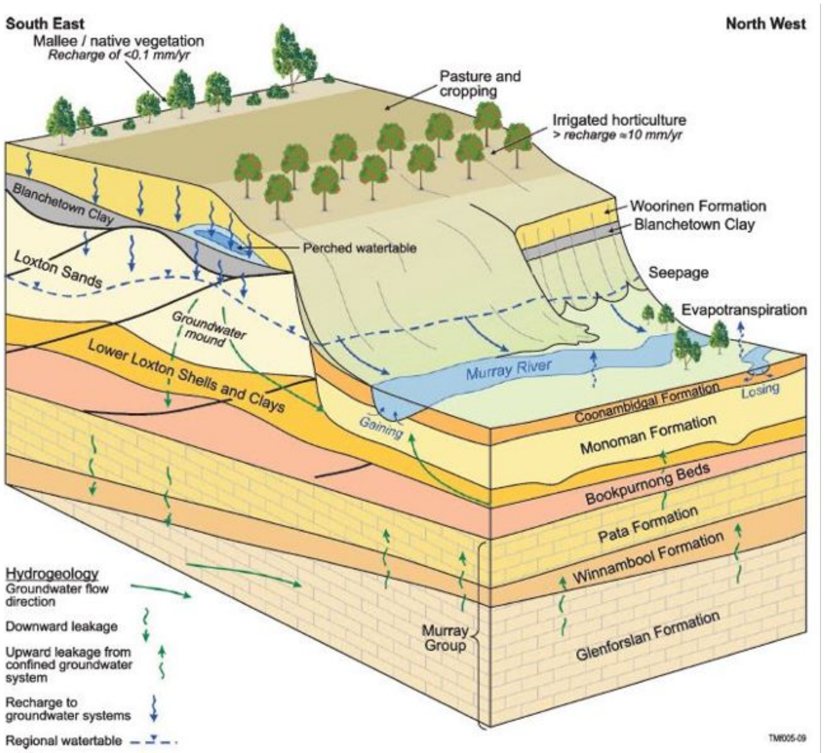
coming to the docs soon!

(Viezzoli et al., 2009)

Example: Bookpurnong



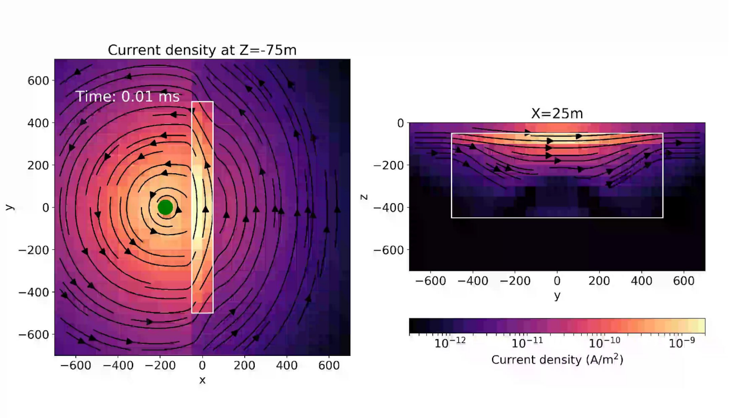
coming to the docs soon!

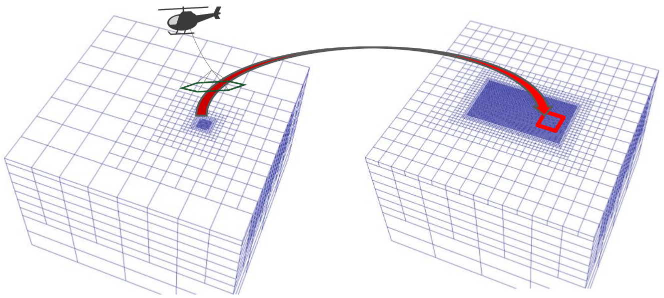


(Viezzoli et al., 2009)

geophysical methods in SimPEG

- Gravity
- Magnetics
- Direct current resistivity
- Induced polarization
- Electromagnetics
  - Frequency Domain
  - Time Domain
  - Controlled + natural sources
- Fluid Flow
  - Richards Equation

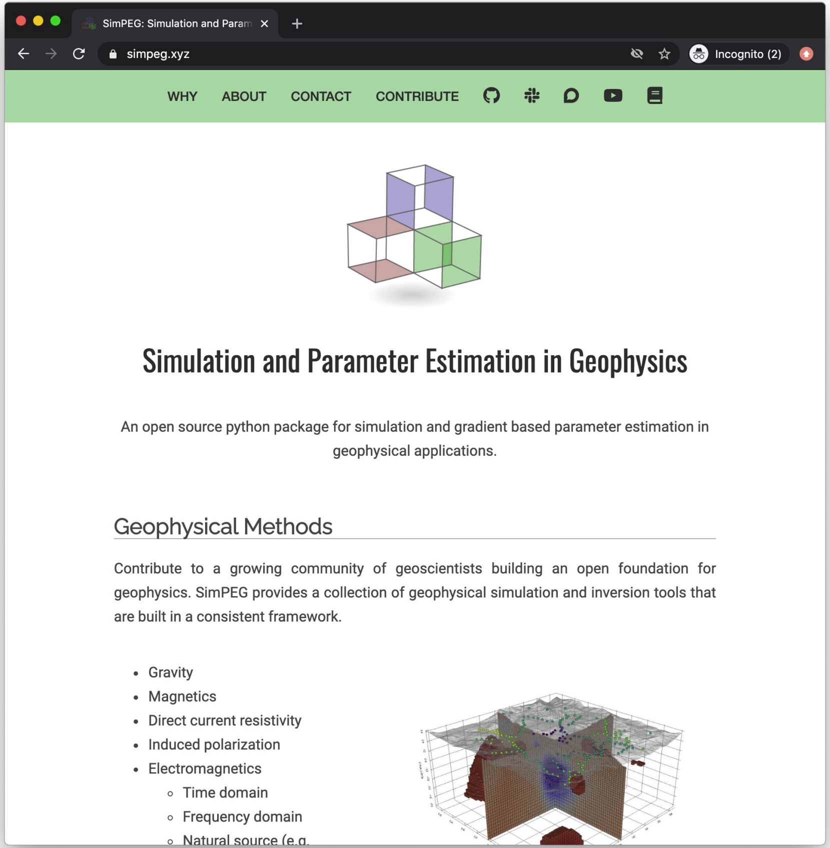


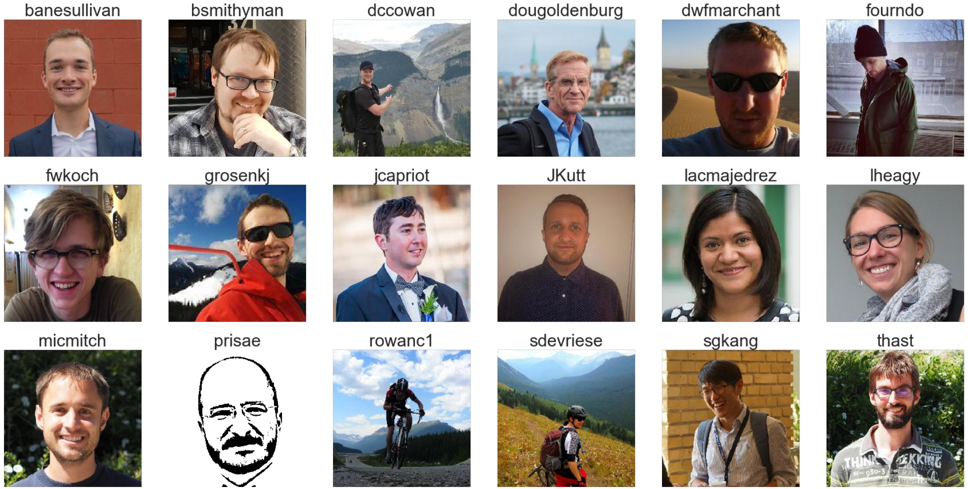


Tiled, parallelized inversions in progress

Dom Fournier

what is simpeg?





https://simpeg.xyz

code + community

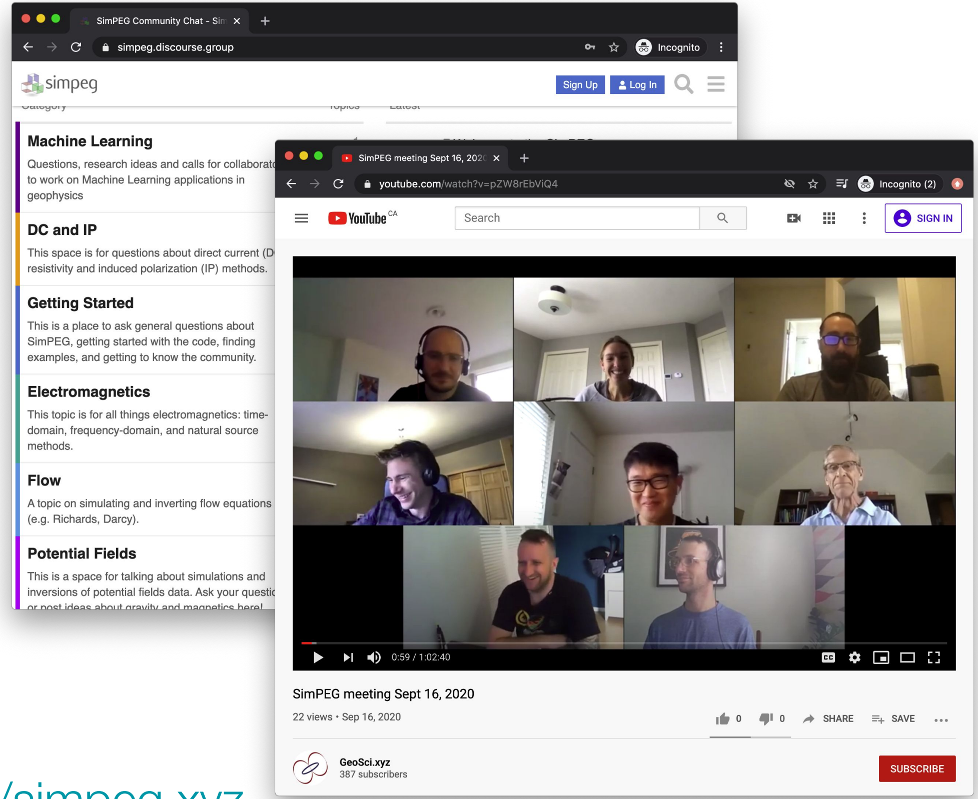
simpeg

Software practices

- Versioning code
- Tracking issues
- Testing code
- Suggesting changes
- Peer-reviewing changes

Communication

- Weekly meetings (recorded)
- Discourse forum for Q&A
- Chat with slack



https://simpeg.xyz

testing

mathematical properties

analytic solutions

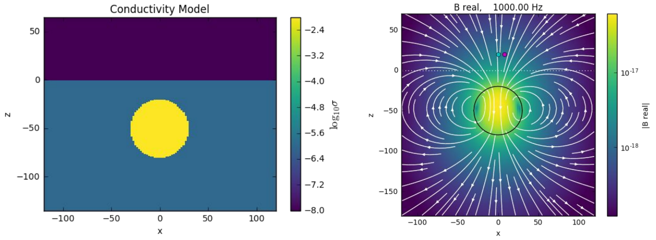
code comparisons

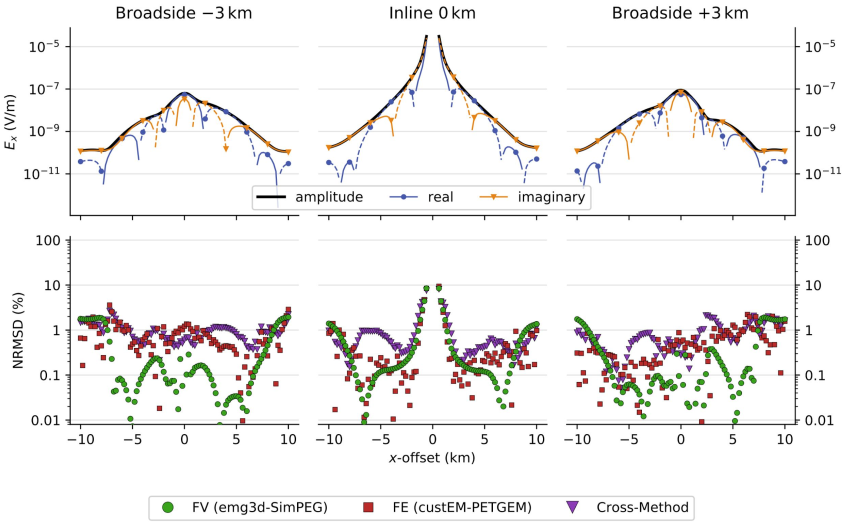
?

confidence

vector identity: $\nabla \cdot \nabla \times \vec{v} = 0$

```
[2]: v = np.random.rand(mesh.nE)
     np.all(mesh.faceDiv * mesh.edgeCurl * v == 0)

[2]: True
```

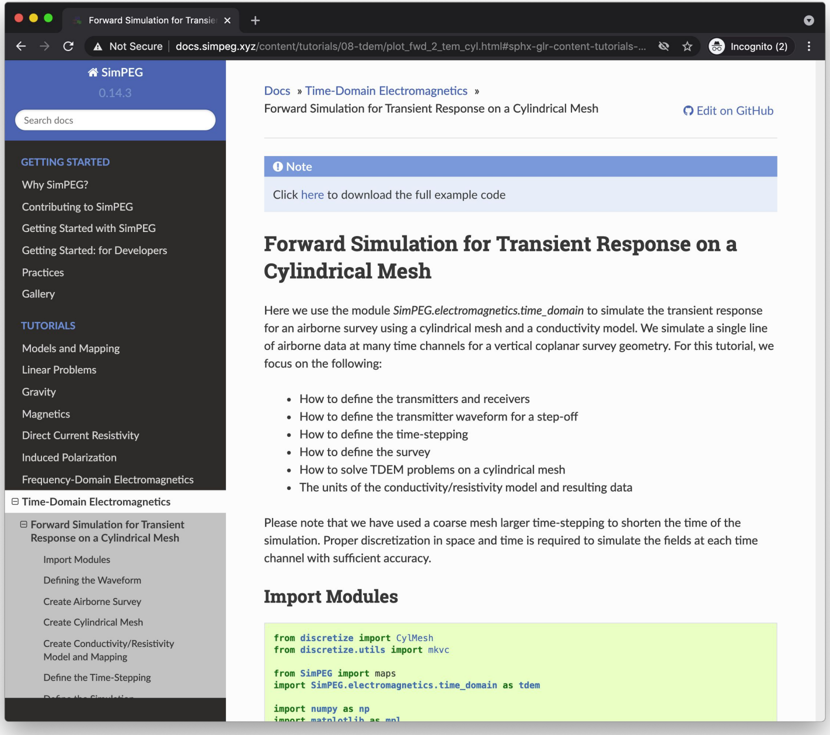




(Werthmüller et al., 2020)

community: connecting + resources

- documentation:
  docs.simpeg.xyz
- community forum:
  simpeg.discourse.group
- chat:
  slack.simpeg.xyz
- meeting notes + recordings:
  curvenote.com/@simpeg/meeting-notes



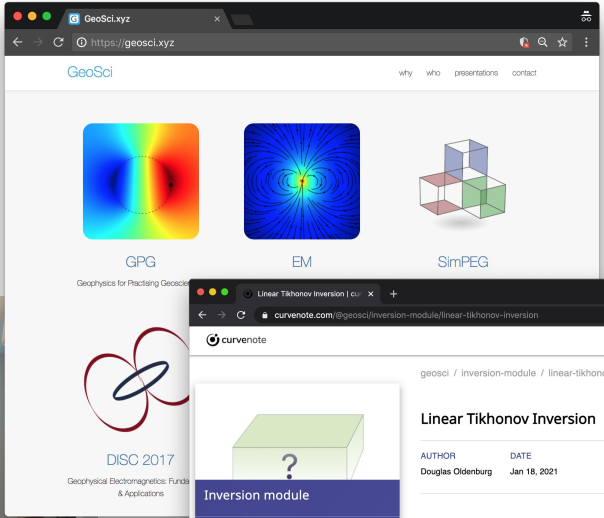
Devin Cowan

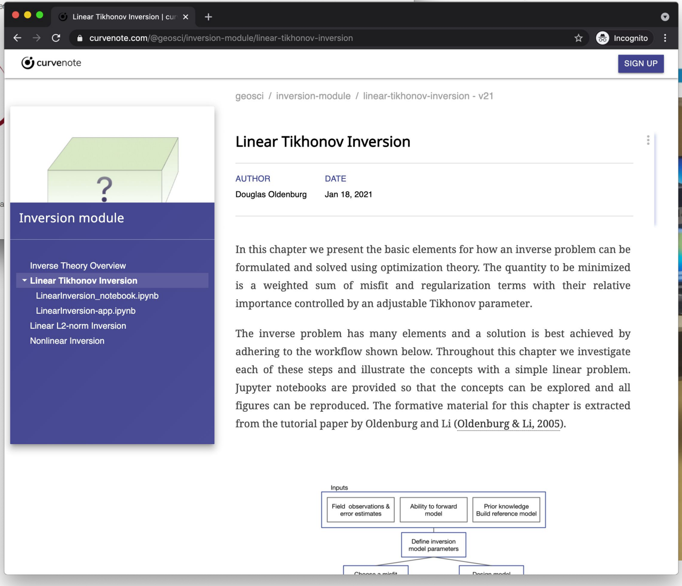
GeoSci.xyz

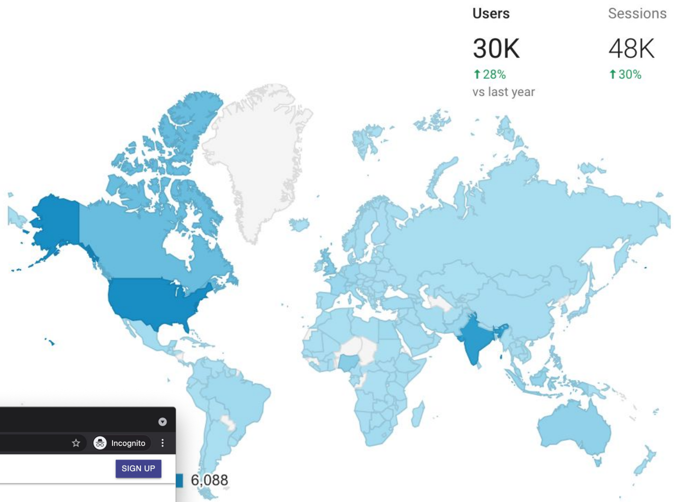
https://geosci.xyz

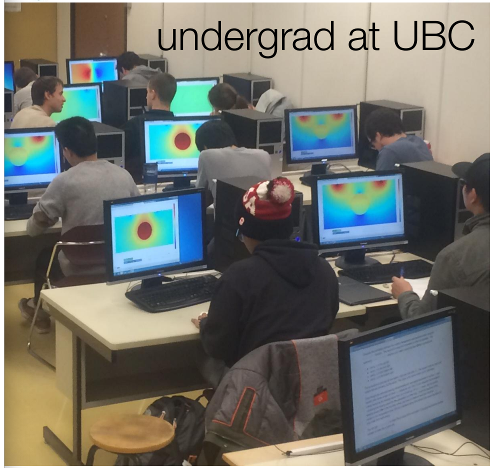
DISC 2017
Electromagnetics Fundamentals and Applications

26 locations worldwide









curvenote.com/@geosci/inversion-module

thank you!

simpeg

banesullivan

bsmithyman

dccowan

dougoldenburg

dwfmarchant

fourndo


fwkoch

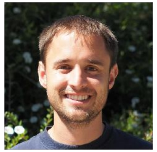
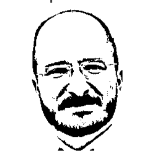
grosenkj

jcapriot

JKutt

lacmajedrez

lheagy


micmitch

prisae

rowanc1

sdevriese

sgkang

thast



Alan Jones


Max Moorkamp

simpeg.xyz

geosci.xyz

curvenote.com/@geosci

lheagy@eoas.ubc.ca

references

Astic, T., & Oldenburg, D. W. (2019). A framework for petrophysically and geologically guided geophysical inversion using a dynamic Gaussian mixture model prior. Geophysical Journal International. https://doi.org/10.1093/gji/ggz389

Cockett, R., Heagy, L. J., & Oldenburg, D. W. (2016). Pixels and their neighbors : Finite volume. The Leading Edge, 35(August), 703–706. https://doi.org/10.1190/tle35080703.1

Cockett, R., Kang, S., Heagy, L. J., Pidlisecky, A., & Oldenburg, D. W. (2015). SimPEG: An open source framework for simulation and gradient based parameter estimation in geophysical applications. Computers & Geosciences, 85, 142–154. https://doi.org/10.1016/j.cageo.2015.09.015

Fournier, D., & Oldenburg, D. W. (2019). Inversion using spatially variable mixed ℓp norms. Geophysical Journal International, 218(1), 268–282. https://doi.org/10.1093/gji/ggz156

Haber, E. (2014). Computational Methods in Geophysical Electromagnetics. Philadelphia, PA: Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9781611973808

Heagy, L. J., Cockett, R., Kang, S., Rosenkjaer, G. K., & Oldenburg, D. W. (2017). A framework for simulation and inversion in electromagnetics. Computers and Geosciences, 107(July), 1–19. https://doi.org/10.1016/j.cageo.2017.06.018

Kang, S., Cockett, R., Heagy, L. J., & Oldenburg, D. W. (2015). Moving between dimensions in electromagnetic inversions. In SEG Technical Program Expanded Abstracts 2015 (pp. 5000–5004). https://doi.org/10.1190/segam2015-5930379.1

references

Mutton, A. J. (2000). The application of geophysics during evaluation of the Century zinc deposit. Geophysics, 65(6), 1946–1960. https://doi.org/10.1190/1.1444878

Oldenburg, D. W., Heagy, L. J., Kang, S., & Cockett, R. (2020). 3D electromagnetic modelling and inversion: a case for open source. Exploration Geophysics, 51(1), 25–37. https://doi.org/10.1080/08123985.2019.1580118

Oldenburg, D. W., & Li, Y. (2005). 5. Inversion for Applied Geophysics: A Tutorial. In D. K. Butler (Ed.), Near-Surface Geophysics (Vol. 13, pp. 89–150). Society of Exploration Geophysicists. https://doi.org/10.1190/1.9781560801719.ch5

Viezzoli, A., Auken, E., & Munday, T. (2009). Spatially constrained inversion for quasi 3D modeling of airborne electromagnetic data - an application for environmental assessment in the Lower Murray Region of South Australia. Exploration Geophysics, 40(2008), 173–183.

Viezzoli, A., Munday, T., Auken, E., & Christiansen, A. V. (2010). Accurate quasi 3D versus practical full 3D inversion of AEM data the Bookpurnong case study. Preview, 2010(149), 23–31. https://doi.org/10.1071/PVv2010n149 p23

Werthmüller, D. (2017). An open-source full 3D electromagnetic modeler for 1D VTI media in Python: empymod. GEOPHYSICS, 82(6), WB9–WB19. https://doi.org/10.1190/geo2016-0626.1

Werthmüller, D., Rochlitz, R., Castillo-Reyes, O., & Heagy, L. (2020). Towards an open-source landscape for 3D CSEM modelling, 1–18. Retrieved from http://arxiv.org/abs/2010.12926