

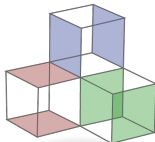
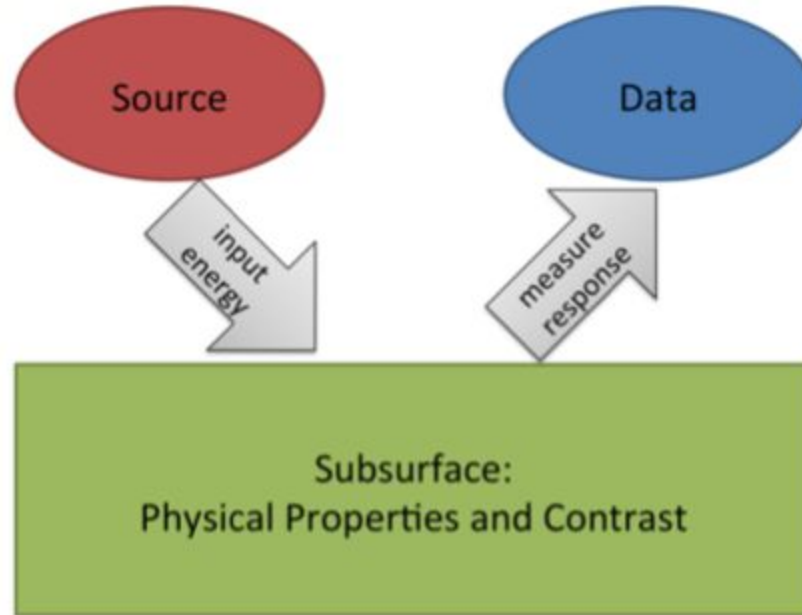


**An open-source framework for geophysical simulations and
inverse problems.**

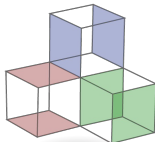
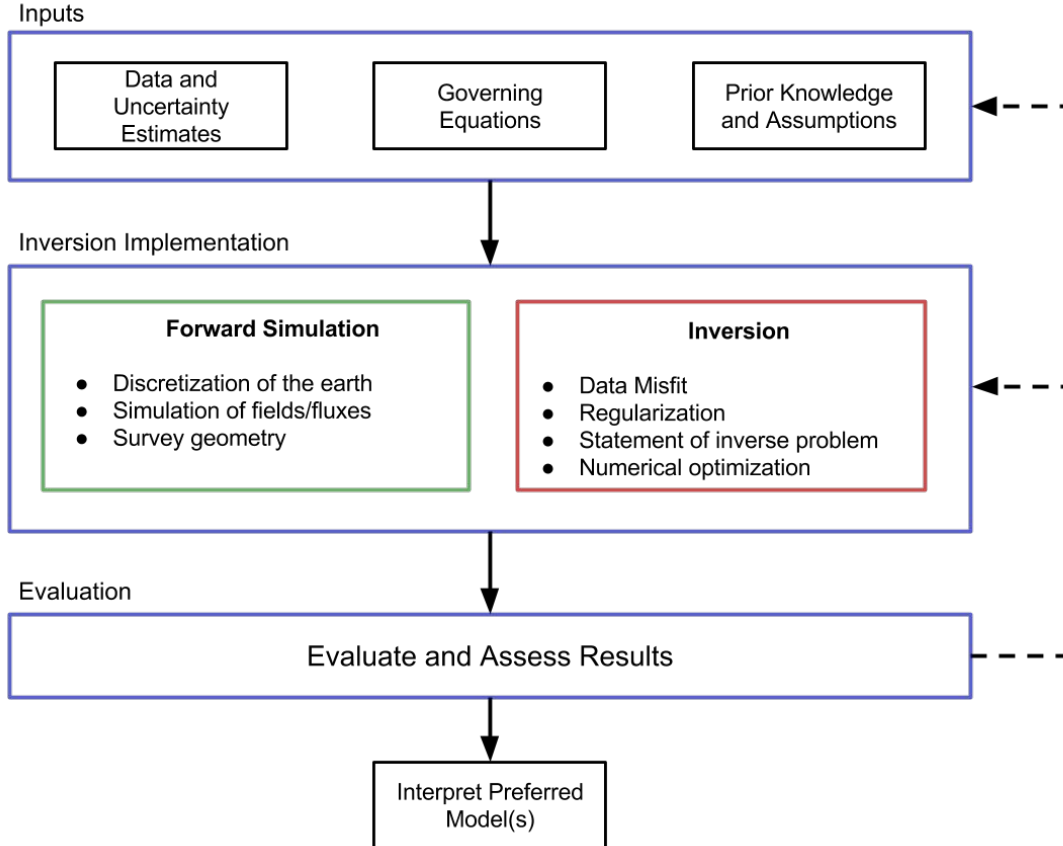
Rowan Cockett, Seogi Kang, Lindsey Heagy, et al.

Geophysical Inversion Facility
University of British Columbia

Geophysics!



Inversion Methodology



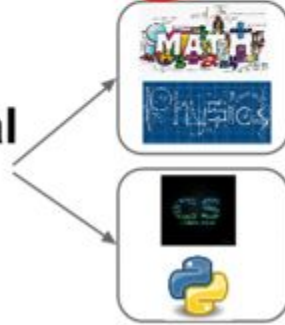
Implementation

**Geophysical
Ideas**



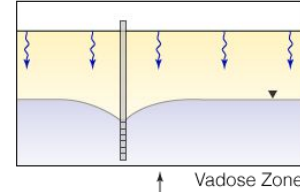
Realizations

**Geophysical
Ideas**

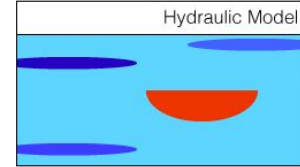


Realizations

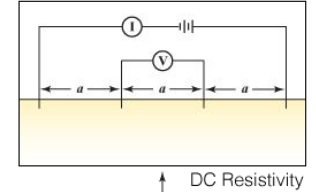
Hydrogeology



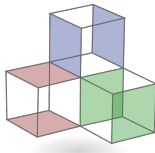
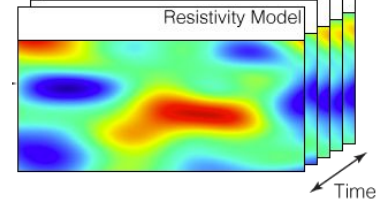
Hydraulic Model



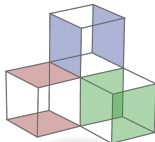
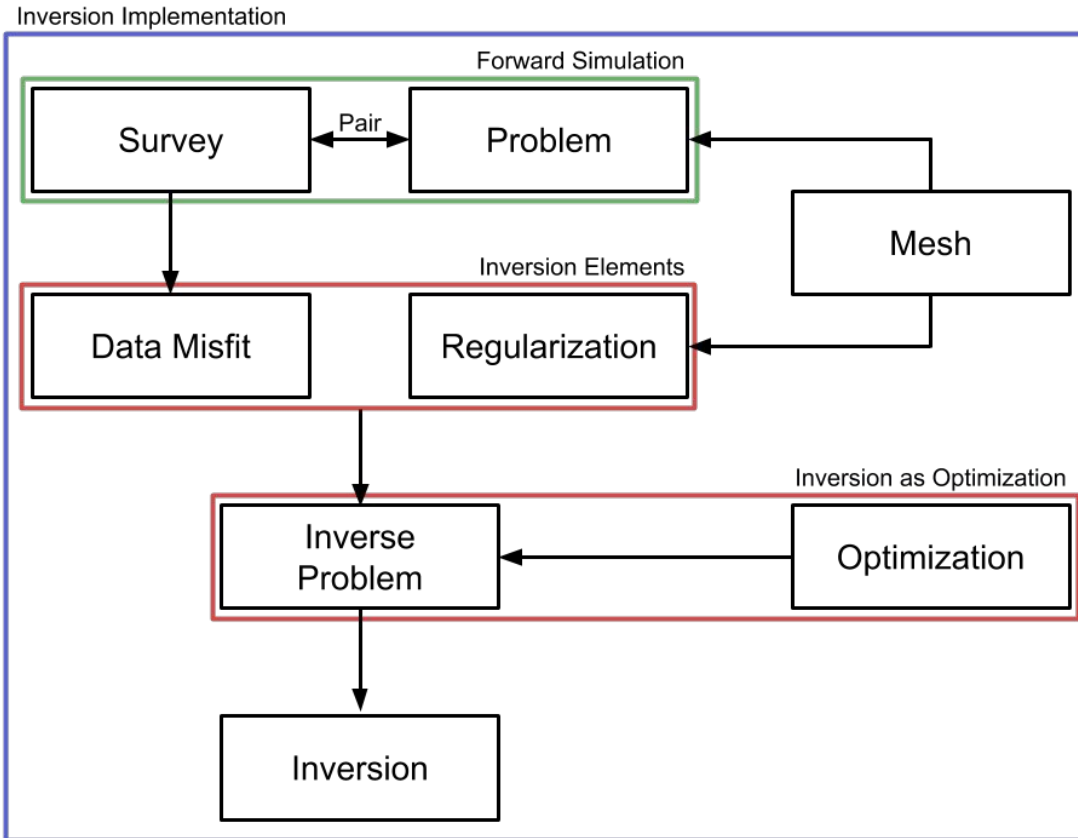
Geophysics



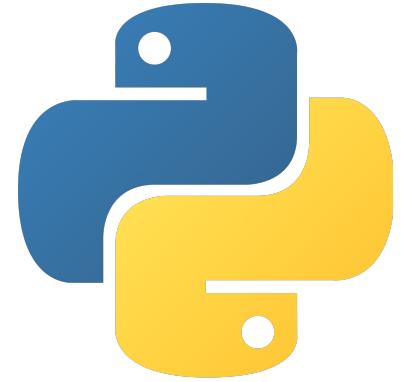
Resistivity Model



Implementation

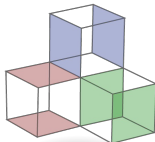
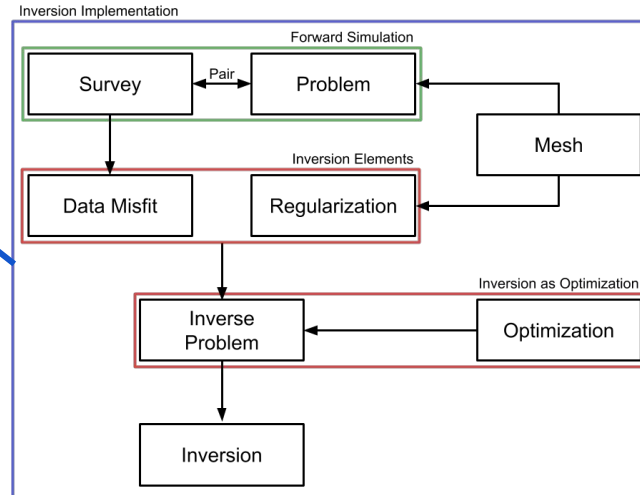


Interactive Geophysics in IPython



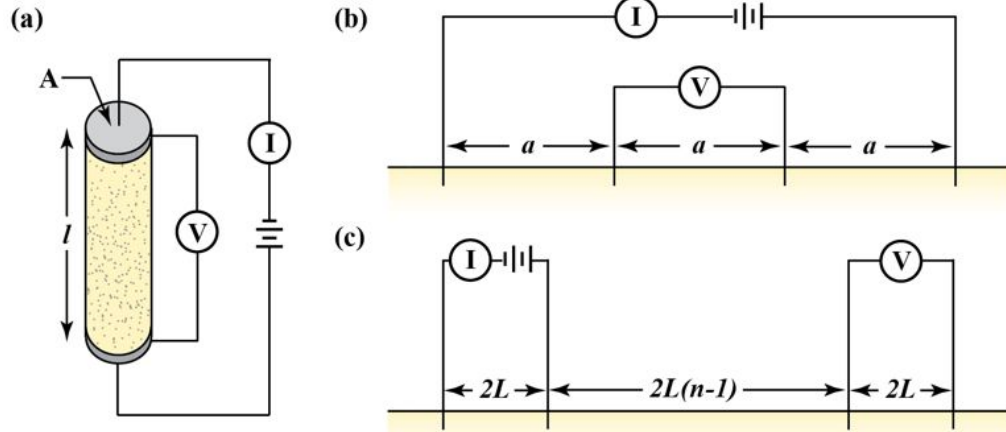
```
In [1]: import SimPEG
```

```
In [ ]: SimPEG.  
SimPEG.DataMisfit  
SimPEG.Directives  
SimPEG.InvProblem  
SimPEG.Inversion  
SimPEG.Maps  
SimPEG.Mesh  
SimPEG.Models  
SimPEG.Optimization  
SimPEG.Problem  
SimPEG.Regularization
```

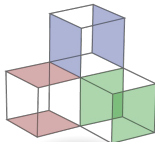
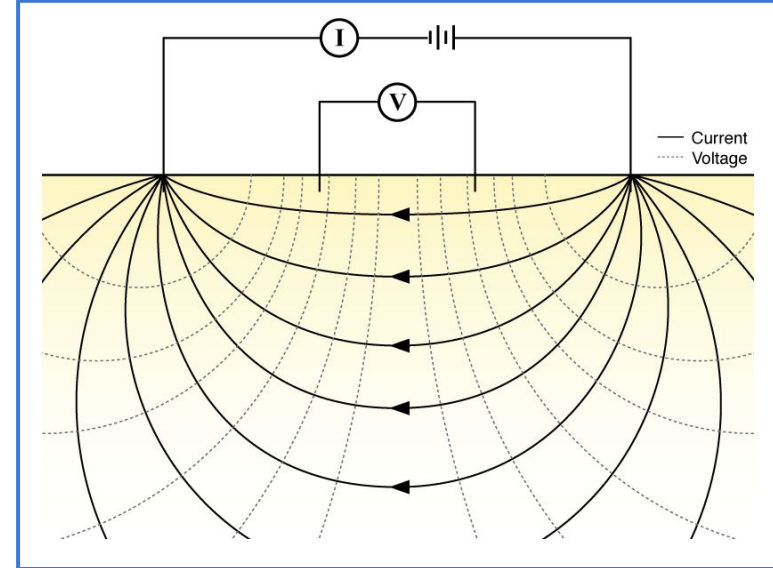


Survey & Problem

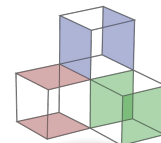
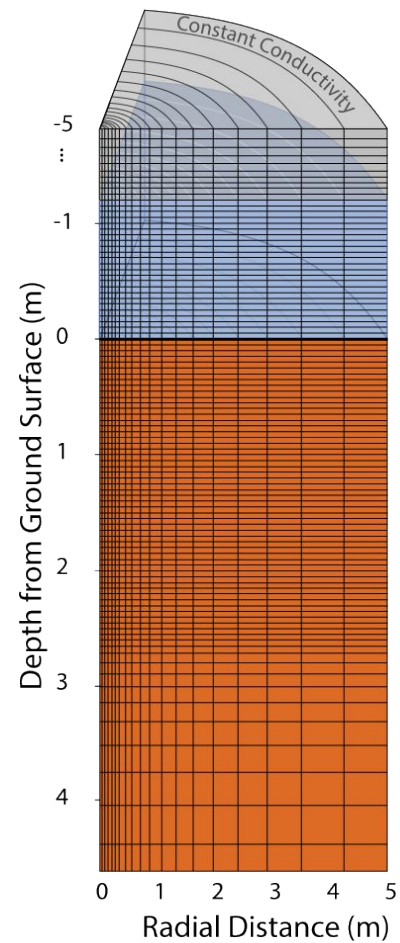
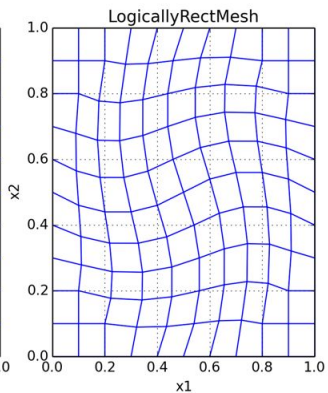
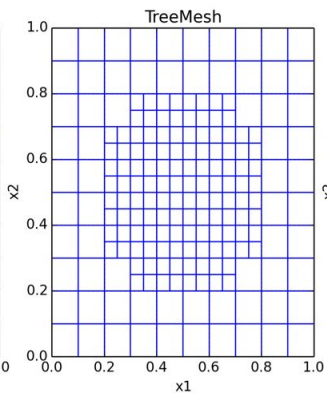
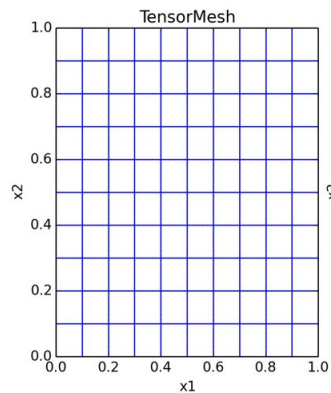
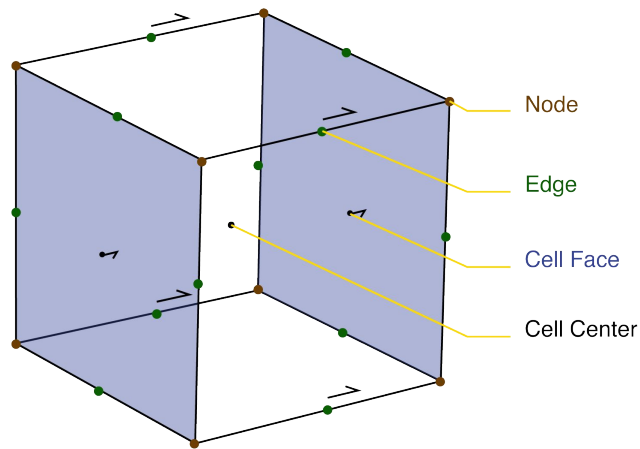
Data collection and geometry



Physics

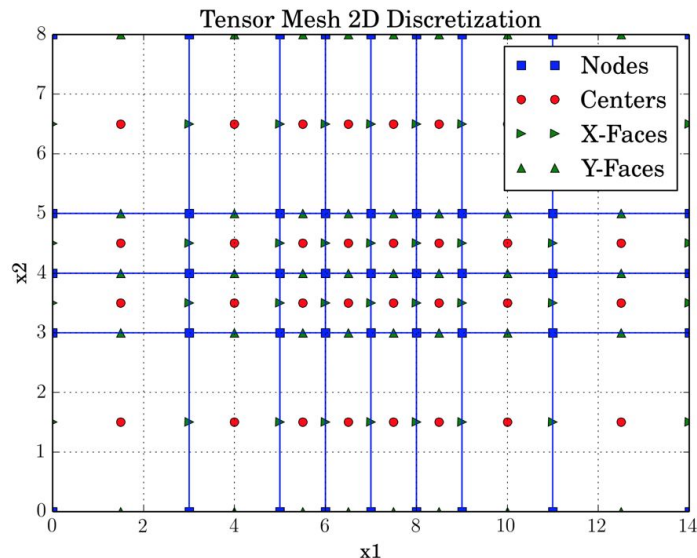


Finite Volume



Creating a Mesh

```
hx = [3,2,1,1,1,1,2,3]
hy = [3,1,1,3]
M = Mesh.TensorMesh([hx, hy])
M.plotGrid(faces=True, nodes=True, centers=True)
```



Property or Function

dim, x0

nC, nN, nF, nE

vol, area, edge

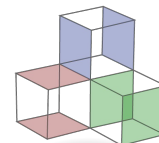
gridN, gridCC, ...

faceDiv, edgeCurl, cellGrad

aveF2CC, aveN2CC, etc.

getEdgeInnerProduct()

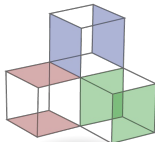
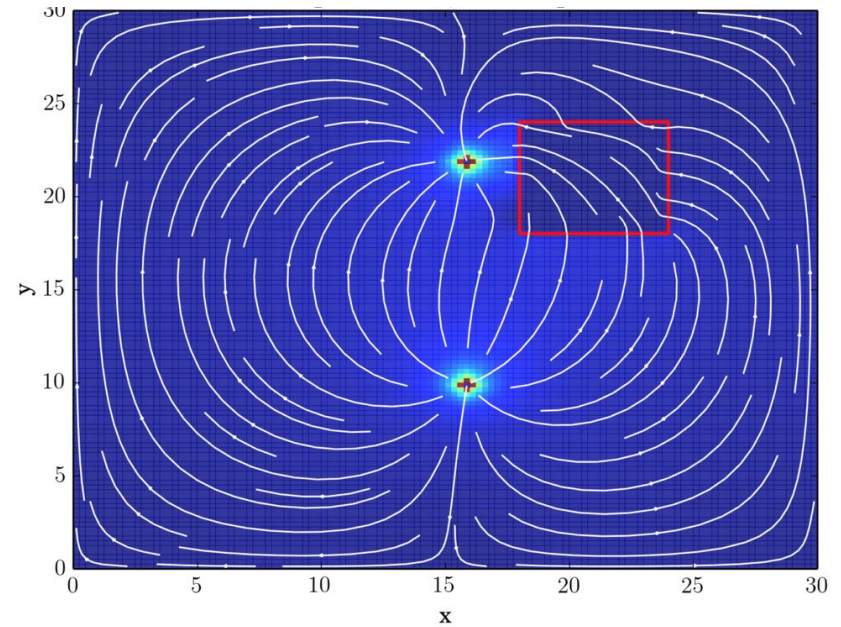
getInterpolationMat(loc)



DC Resistivity

$$\nabla \cdot (-\sigma \nabla \phi) = \mathbf{I}(\delta(\mathbf{r} - \mathbf{r}_{s^+}) - \delta(\mathbf{r} - \mathbf{r}_{s^-}))$$

```
D = M.faceDiv
G = M.cellGrad
# Harmonically average sigma
MsigI = sdInv(sdiag(M.aveF2CC.T*(1/sig)))
A = D*MsigI*G
A[0,0] *= 1/M.vol[0] # Remove the null space
Ainv = Solver(A) # Create a default Solver
phi = Ainv * ( - q )
```



Time Domain Electromagnetics

32 lines of code

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

$$\nabla \times \frac{1}{\mu_0} \vec{b} - \sigma \vec{e} = \vec{j}_s.$$

$$\mathbf{C} \vec{e}^{(t+1)} + \frac{\vec{b}^{(t+1)} - \vec{b}^{(t)}}{\Delta t} = 0$$

$$\mathbf{C}^\top \mathbf{M}_{\mu-1}^f \vec{b}^{(t+1)} - \mathbf{M}_\sigma^e \vec{e}^{(t+1)} = \mathbf{M}^e \vec{j}_s^{(t+1)}$$

```

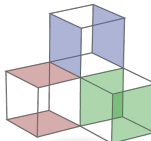
from SimPEG import *
import simpegEM as EM
from pymatsolver import MumpsSolver
from scipy.constants import mu_0

# Create the computational mesh
cs, nc, npad = 20., 20, 5 # cell sz, num cells/padding
h = [(cs,npad,-1.3),(cs,nc),(cs,npad,1.3)]
mesh = Mesh.TensorMesh([h,h,h], 'CCC')
# Create a half-space conductivity
sigma = np.ones(mesh.nC)*1e-8
sigma[mesh.gridCC[:,2] < 0] = 1e-3

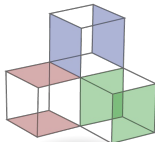
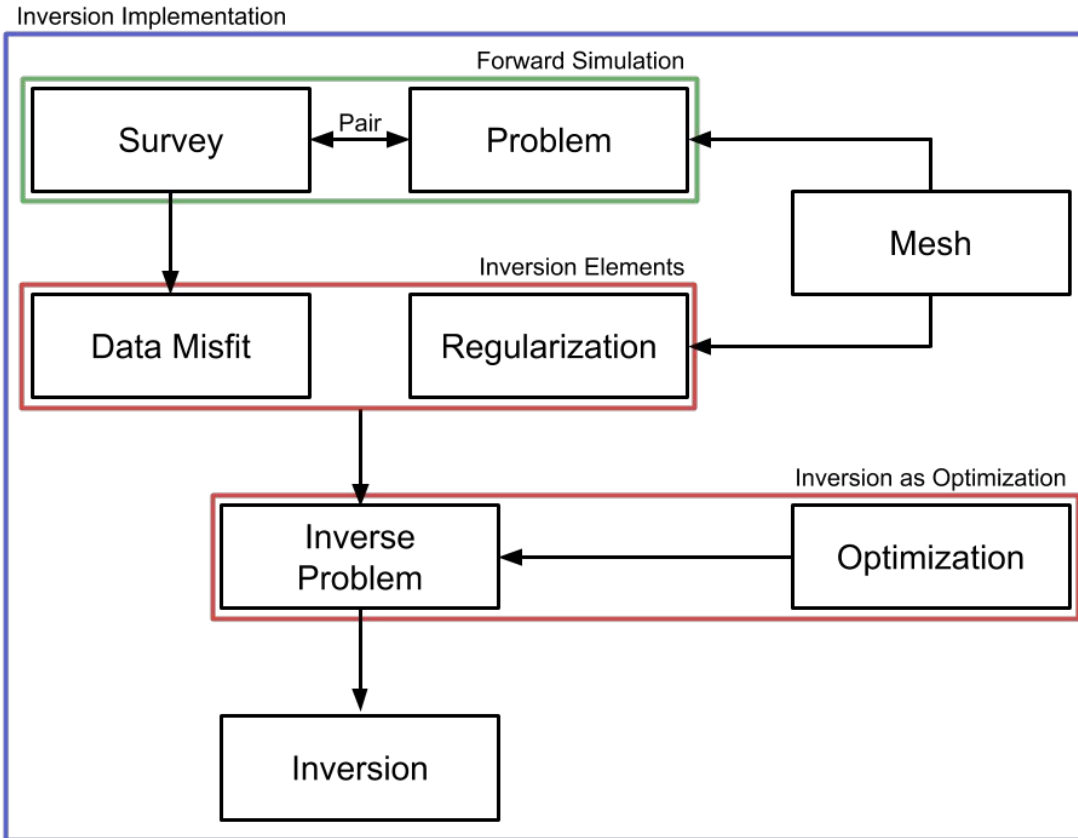
# Create a source in the center of our domain (0,0,0)
As = EM.Sources.MagneticDipoleVectorPotential(
    np.zeros(3), mesh, ['Ex','Ey','Ez'])
C = mesh.edgeCurl
b0 = C*As

# Create inner products
MesigI = mesh.getEdgeInnerProduct(sigma, invMat=True)
Mfmui = mesh.getFaceInnerProduct(1./mu_0)
Me = mesh.getEdgeInnerProduct()

# Maxwell's Equation (eliminate e, j_s=0)
dt = 1e-7 # Choose a time-step
A = Mfmui*C*MesigI*C.T*Mfmui + 1.0/dt*Mfmui
Ainv = MumpsSolver(A) # Factor the matrix!
# Solve for b using Backward Euler!
B = [b0] + range(299)
for i in range(len(B)-1):
    B[i+1] = Ainv * ( 1.0/dt*Mfmui*B[i] )
    
```



Implementation



Inversion Elements

Data Misfit

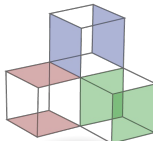
$$\phi_d(\mathbf{m}) = \frac{1}{2} \|\mathbf{W}_d(F[\mathbf{m}] - \mathbf{d}_{\text{obs}})\|_2^2$$

Regularization

$$\phi_m(\mathbf{m}) = \frac{1}{2} \|\mathbf{W}_m(\mathbf{m} - \mathbf{m}_{\text{ref}})\|_2^2$$

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

Inverse Problem



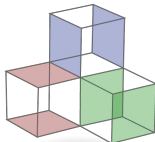
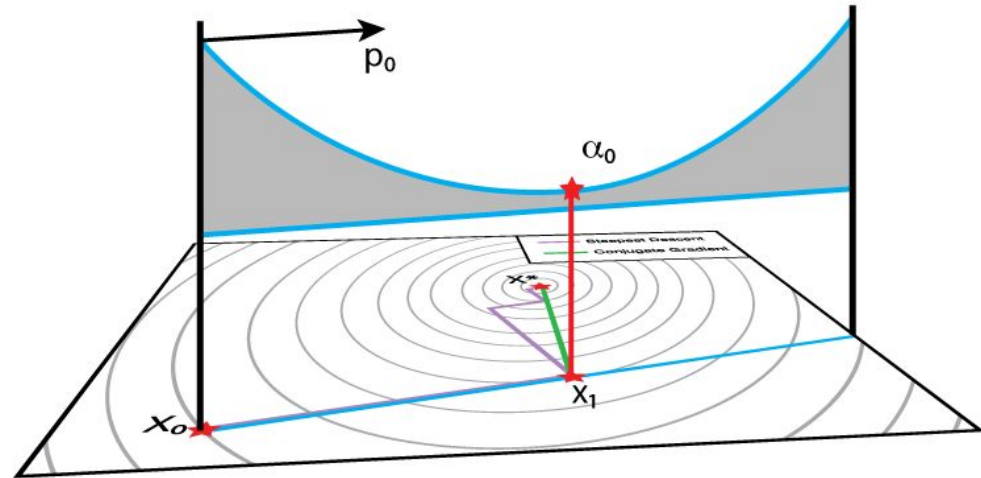
Optimization

```
In [1]: from SimPEG import Optimization
```

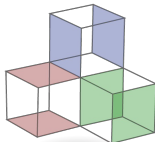
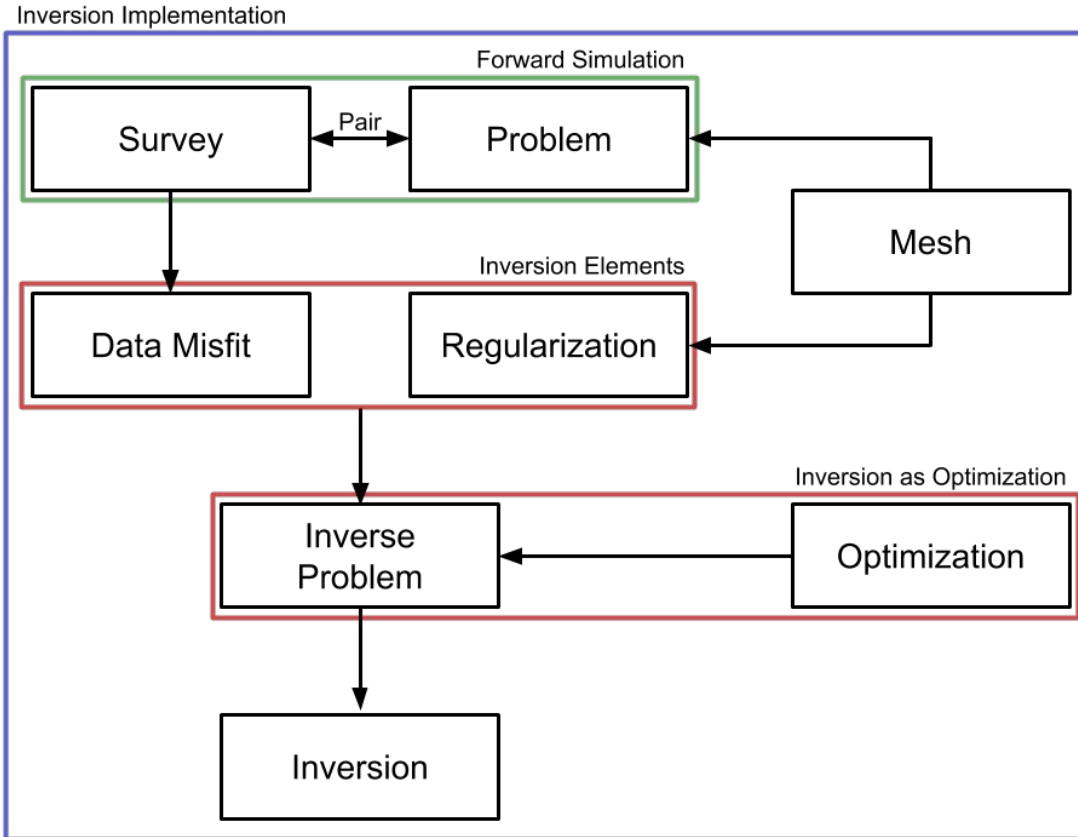
```
In [ ]: Optimization.  
Optimization.BFGS  
Optimization.GaussNewton  
Optimization.InexactGaussNewton  
Optimization.IterationPrinters  
Optimization.Minimize  
Optimization.NewtonRoot  
Optimization.ProjecteGNCG  
Optimization.ProjecteGradient  
Optimization.Remember  
Optimization.Solver
```

$$\underset{\mathbf{m}}{\text{minimize}} \quad \phi(\mathbf{m}) = \phi_d(\mathbf{m}) + \beta\phi_m(\mathbf{m})$$

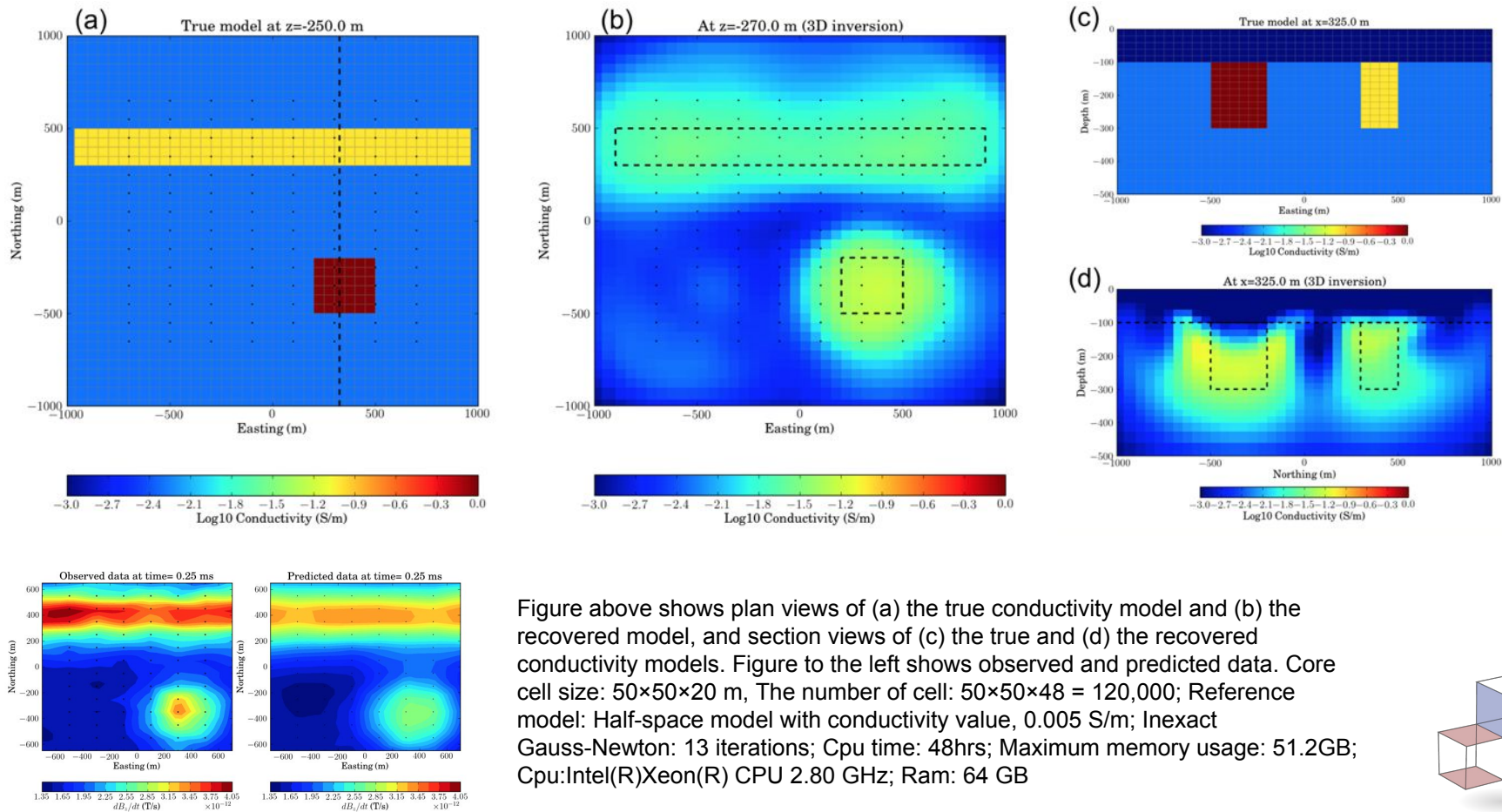
$$\text{s.t.} \quad \phi_d \leq \phi_d^*, \quad \mathbf{m}_i^L \leq \mathbf{m}_i \leq \mathbf{m}_i^H$$



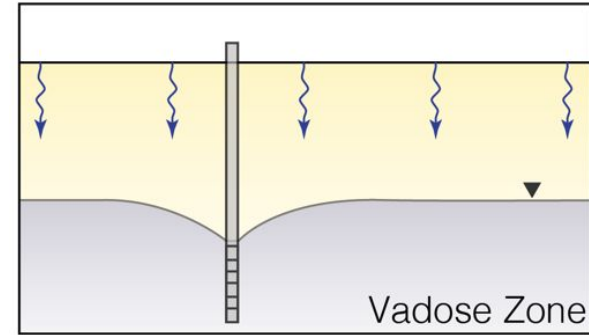
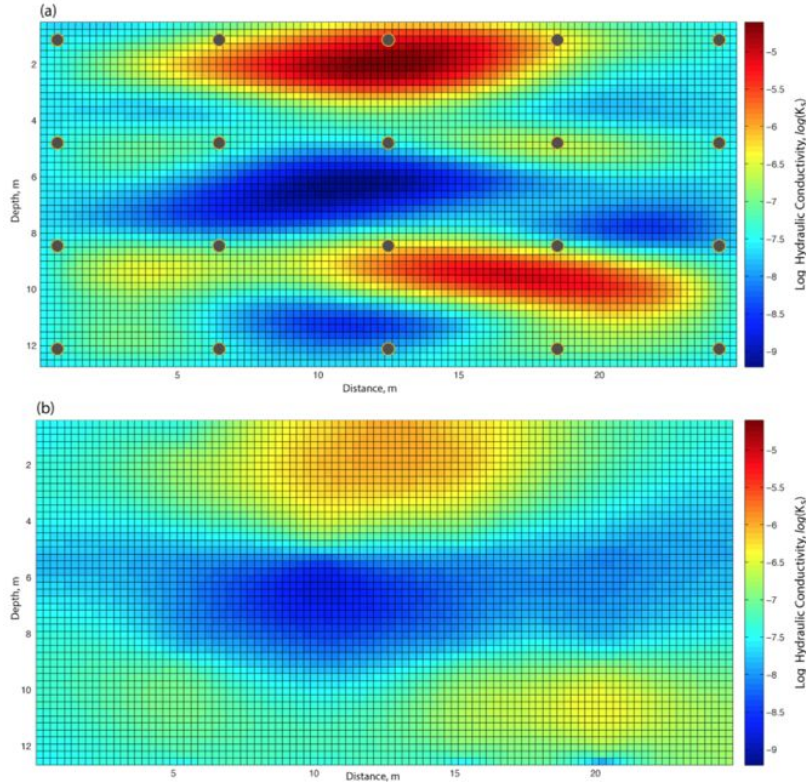
Bringing it together.



Airborne Time Domain EM Inversion



Hydraulic Conductivity Inversion (Richards Eqn.)



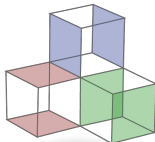
$$\frac{\partial \theta(\psi)}{\partial t} - \underbrace{\nabla \cdot K(\psi) \nabla \psi}_{\text{Diffusion}} - \underbrace{\frac{\partial K(\psi)}{\partial z}}_{\text{Gravity}} = 0$$

5 wells, each with 4 sampling ports: Simulate an infiltration experiment.

Van Genuchten parameters: α : 0.036; n : 1.56; θ_s : 0.43; θ_r : 0.078

Boundary Conditions: top: -50.7 cm; bottom: -100.5 cm; sides: no-flow

20 iterations, 4 PCG inner iterations; relative data misfit: 11%; CPU-time: 2 hrs





- Framework and toolbox for geophysics
- Motivated by inverse problem
- Guided by terminology
- Modular pieces that allow separation of concerns
- Built in the open to be open

Built in the open

pypi v0.1.1

downloads 188/month

license MIT

build passing

coverage 85%

SimPEG

Search docs

Why SimPEG?
License
Authors
Projects Using SimPEG
Installation
SimPEG Meshes
Differential Operators

Inner Products
Example problem for DC resistivity
Defining Tensor Properties
Structure of Matrices
Taking Derivatives
The API

Forward Problem
Data Misfit
Regularization
Optimize
Directives
Inversion
Solver
SimPEG Maps
Utilities
Testing SimPEG

Read the Docs

v: latest

Example problem for DC resistivity

We will start with the formulation of the Direct Current (DC) resistivity problem in geophysics.

$$\frac{1}{\sigma} \vec{j} = \nabla \phi$$
$$\nabla \cdot \vec{j} = q$$

In the following discretization, σ and ϕ will be discretized on the cell-centers and the flux, \vec{j} , will be on the faces. We will use the weak formulation to discretize the DC resistivity equation.

We can define in weak form by integrating with a general face function \vec{f} :

$$\int_{\Omega} \sigma^{-1} \vec{j} \cdot \vec{f} = \int_{\Omega} \nabla \phi \cdot \vec{f}$$

Here we can integrate the right side by parts,

$$\nabla \cdot (\phi \vec{f}) = \nabla \phi \cdot \vec{f} + \phi \nabla \cdot \vec{f}$$

and rearrange it, and apply the Divergence theorem.

$$\int_{\Omega} \sigma^{-1} \vec{j} \cdot \vec{f} = - \int_{\Omega} (\phi \nabla \cdot \vec{f}) + \int_{\partial \Omega} \phi \vec{f} \cdot \mathbf{n}$$

We can then discretize for every cell:

$$\mathbf{v}_{\text{cell}} \sigma^{-1} (\mathbf{J}_x \mathbf{F}_x + \mathbf{J}_y \mathbf{F}_y + \mathbf{J}_z \mathbf{F}_z) = -\phi^T \mathbf{v}_{\text{cell}} \mathbf{D}_{\text{cell}} \mathbf{F} + \mathbf{BC}$$

Note

We have discretized the dot product above, but remember that we do not really have a single vector \mathbf{J} , but approximations of \vec{j} on each face of our cell. In 2D that means 2 approximations of \mathbf{J}_x and 2 approximations of \mathbf{J}_y . In 3D we also have 2 approximations of \mathbf{J}_z .

Regardless of how we choose to approximate this dot product, we can represent this in vector form (again this is for every cell), and will generalize for the case of anisotropic (tensor) sigma.

$$\mathbf{F}_c^T (\sqrt{\mathbf{v}_{\text{cell}}} \Sigma^{-1} \sqrt{\mathbf{v}_{\text{cell}}}) \mathbf{J}_c = -\phi^T \mathbf{v}_{\text{cell}} \mathbf{D}_{\text{cell}} \mathbf{F} + \mathbf{BC}$$

We multiply by square-root of volume on each side of the tensor conductivity to keep symmetry in the system. Here \mathbf{J}_c is the Cartesian \mathbf{J} (on the faces that we choose to use in our approximation) and must be calculated differently depending on the mesh:

$$\mathbf{J}_c = \mathbf{Q}_{(i)} \mathbf{J}_{\text{TENSOR}}$$
$$\mathbf{J}_c = \mathbf{N}_{(i)}^T \mathbf{Q}_{(i)} \mathbf{J}_{\text{LRM}}$$

Here the i index refers to where we choose to approximate this integral, as discussed in the note above. We will approximate this integral by taking the fluxes clustered around every node of the cell, there are 8 combinations in 3D, and 4 in 2D. We will use a projection matrix $\mathbf{Q}_{(i)}$ to pick the appropriate fluxes. So, now that we have 8 approximations of this integral, we will just take the average. For the TensorMesh, this looks like:

Search GitHub

Explore

Get

Blog

Help

SimPEG

http://simpeg.github.io

Filters

Find a repository...

New repository

simpegdc

A DC resistivity forward modelling and inversion package for SimPEG.

Updated 12 days ago

Python 3.1 3.2

simpegem

A electromagnetic forward modelling and inversion package for SimPEG.

Updated 15 days ago

Python 3.8 3.9

simpegflow

Groundwater flow equations written in the SimPEG framework

Python 3.8 3.9

COVERALLS									
simpeg, simpeg / 298									
ALL 51	CHANGED 15	SOURCE CHANGED 7	COVERAGE CHANGED 13	SHOW 10 100 1000					
COVERAGE	FILE	LINE	RELEVANT	COVERED	MISSED	INFO/LINE			
94.12%	simpegdc/_init_.py	19	17	10	1	1.0			
93.75%	simpeg/Technical_problem.py	26	16	15	1	1.0			
93.24%	simpeg/Technical_boundaryProblem.py	301	300	300	0	1.0			
91.69%	simpeg/innerProducts.py	84	37	34	2	1.0			
91.6%	simpeg/Technical_reporter.py	31	24	23	2	1.0			
91.0%	simpeg/innerProducts.py	200	86	77	23	1.0			
90.74%	simpeg/Technical_example_inner.py	19	19	9	1	1.0			
89.34%	simpeg/Technical_example.py	100	117	105	12	1.0			

ravis

Home

Blog

Status

Help

Travis CI for Private Repositories

Rowan Cockett

Search all repositories

My Repositories

Recent

+

simpeg/simpeg

349

4 min 16 sec

about 15 hours ago

simpeg/simpegdc

12

1 min 48 sec

about 24 hours ago

simpeg/simpegpf

44

2 min 42 sec

about 24 hours ago

simpeg/simpegem

150

19 min 53 sec

a day ago

simpeg/simpegflow

25

2 min

a day ago

simpeg/simpeg

build passing

Simulation and Parameter Estimation in Geophysics - A python package for simulation and gradient based parameter estimation in the context of geophysical applications.

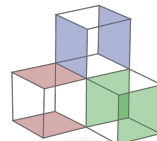
Current

Build History

Pull Requests

Branch Summary

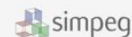
Build	Message	Commit	Duration	Finished
349	Force overwrite of docs in makeSyntheticData	41d105e (master)	4 min 16 sec	about 15 hours ago
348	project meta updates	94ddcb0 (master)	4 min 42 sec	about 22 hours ago
347	update readme and setup for long description	aa74826 (master)	4 min 20 sec	about 22 hours ago
346	Update README.rst	89becc1 (master)	4 min 37 sec	about 22 hours ago
345	Update and rename README.md to README.rst	596395a (master)	3 min 40 sec	about 22 hours ago
344	add more qualifiers to setup.py	00ade54 (master)	4 min 6 sec	about 22 hours ago
343	Update README.md	8c1b6ec (master)	5 min 17 sec	about 23 hours ago
342	Bump version: 0.1.0 → 0.1.1	1f7e1cb (master)	4 min 38 sec	about 23 hours ago
341	update innerproduct tests	944530f (master)	3 min 59 sec	a day ago
340	Merge pull request #75 from simpeg/innerProducts	d2f6a64 (master)	4 min 23 sec	a day ago



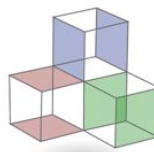
Look at code.



<http://simpeg.xyz>



DOCS	CODE	JOURNAL	CONTACT
------	------	---------	---------



Simulation and Parameter Estimation in Geophysics

An open source python package for simulation and gradient based parameter estimation in geophysical applications.

Installation

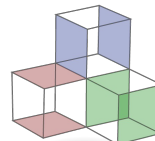
The easiest way to install SimPEG is from [PyPI](#), using [pip](#):

```
> pip install SimPEG
```

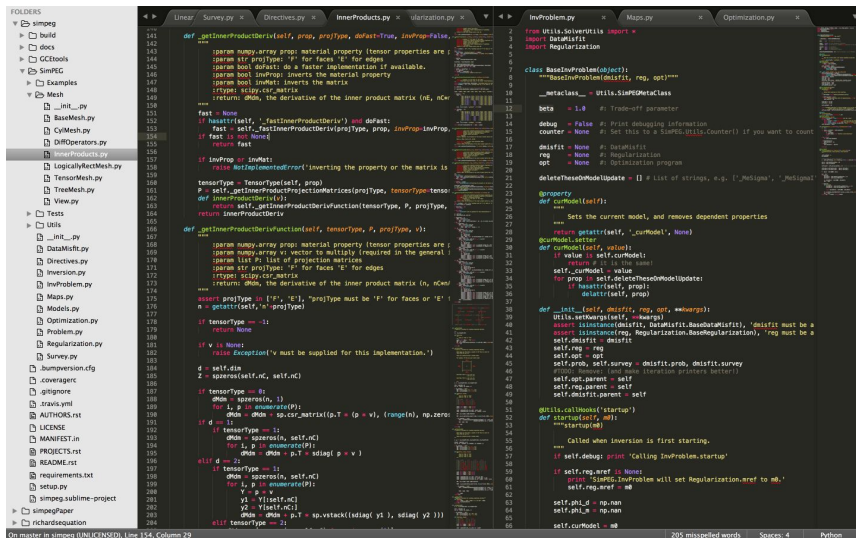


Read more detailed installations instructions in the [documentation](#).

Get the source code at [Github](#).



© 2014 SimPEG Developers



People



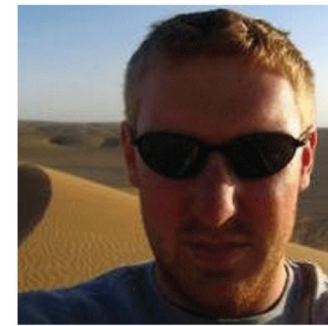
Rowan Cockett



Seogi Kang



Lindsey Heagy



Dave Marchant



Doug Oldenburg



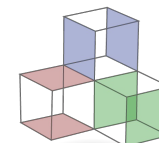
Eldad Haber



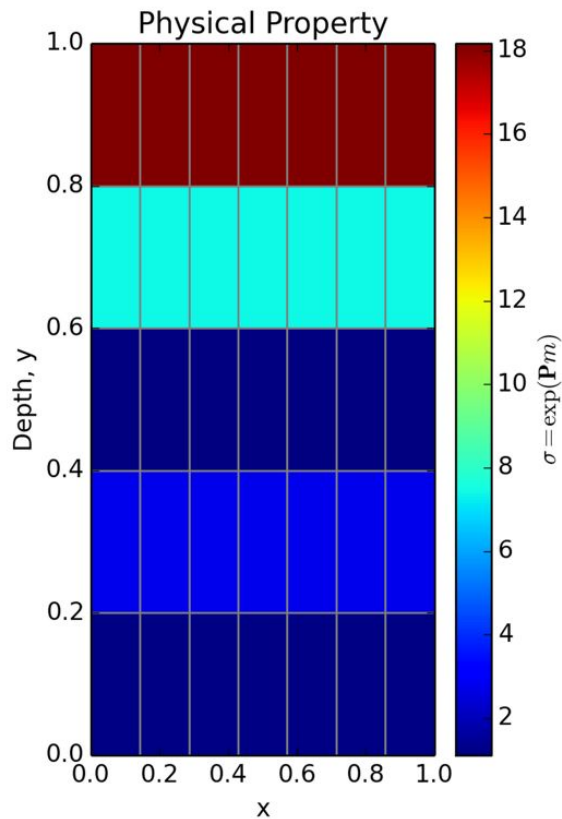
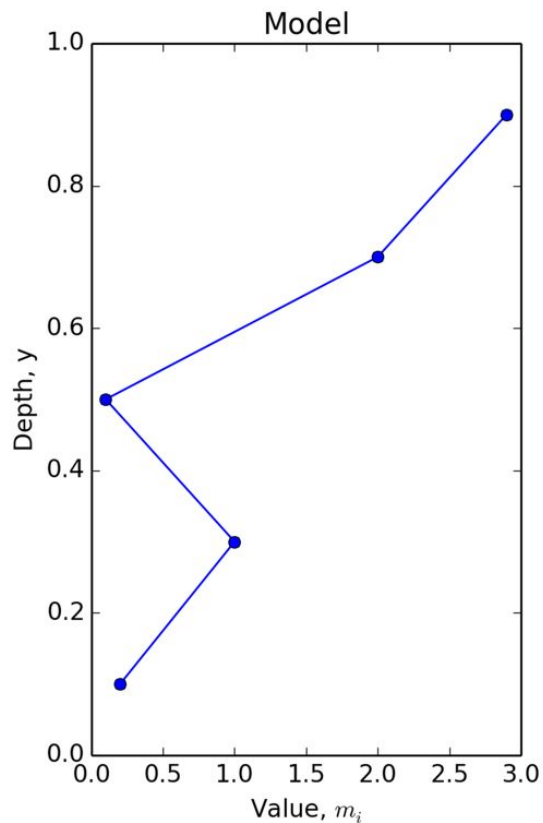
Adam Pidlisecky



you!?



Mapping between spaces



```
M = Mesh.TensorMesh([7,5])
v1dMap = Maps.Vertical1DMap(M)
expMap = Maps.ExpMap(M)
myMap = expMap * v1dMap
m = np.r_[0.2,1,0.1,2,2.9]
sig = myMap * m
```

