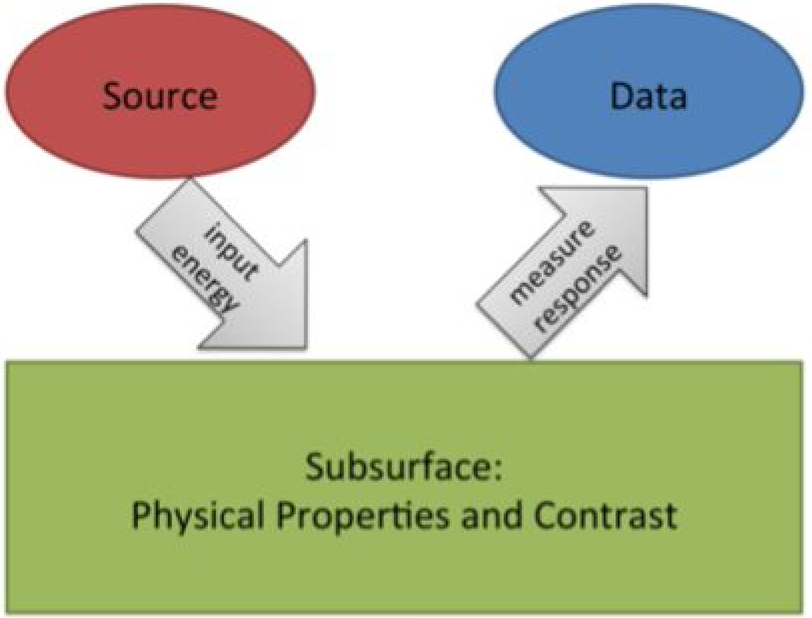
simpeg

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

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

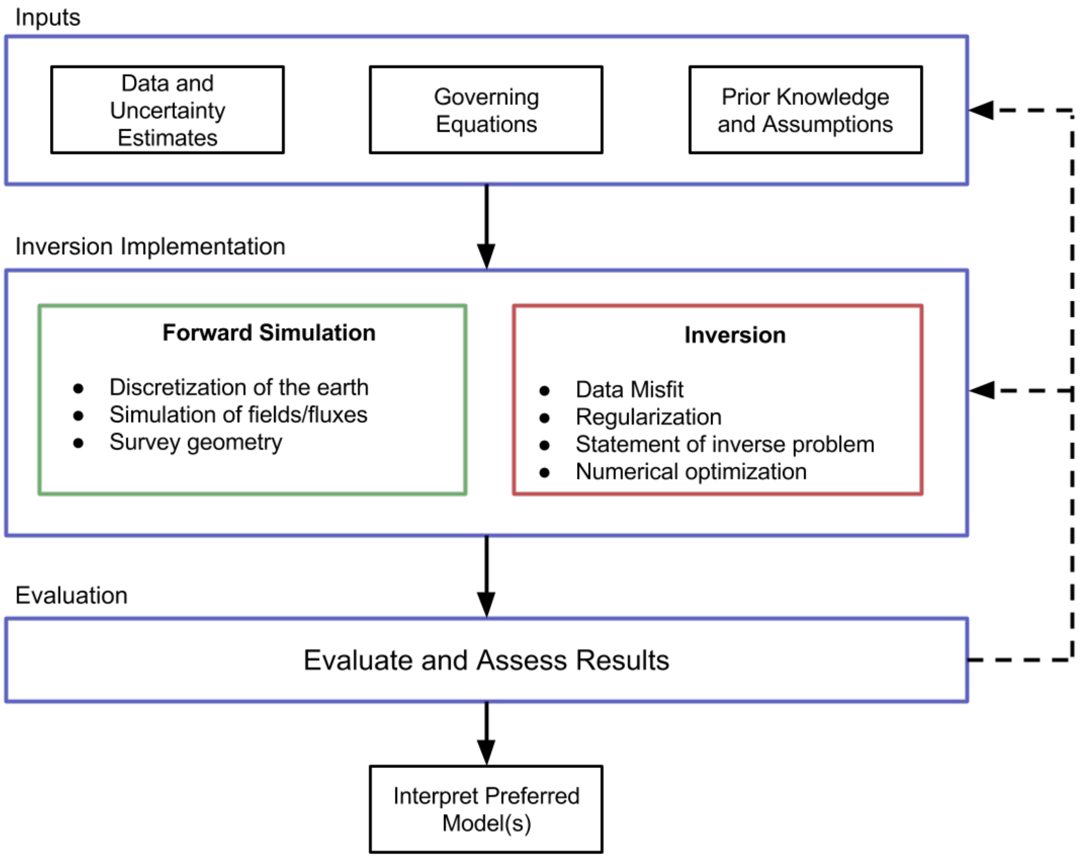
Geophysical Inversion Facility
University of British Columbia

Geophysics!



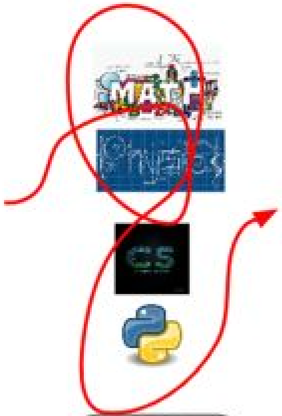
Source
Data
input energy
measure response
Subsurface:
Physical Properties and Contrast

Inversion Methodology



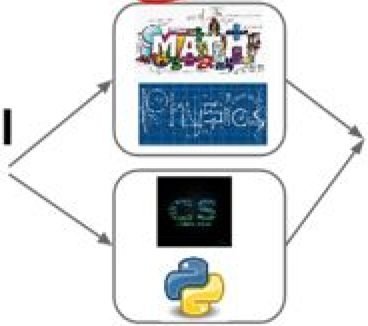
Implementation

Geophysical Ideas

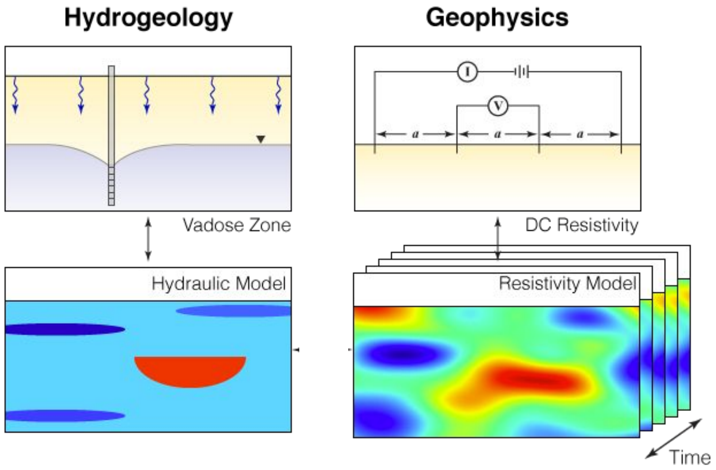


Realizations

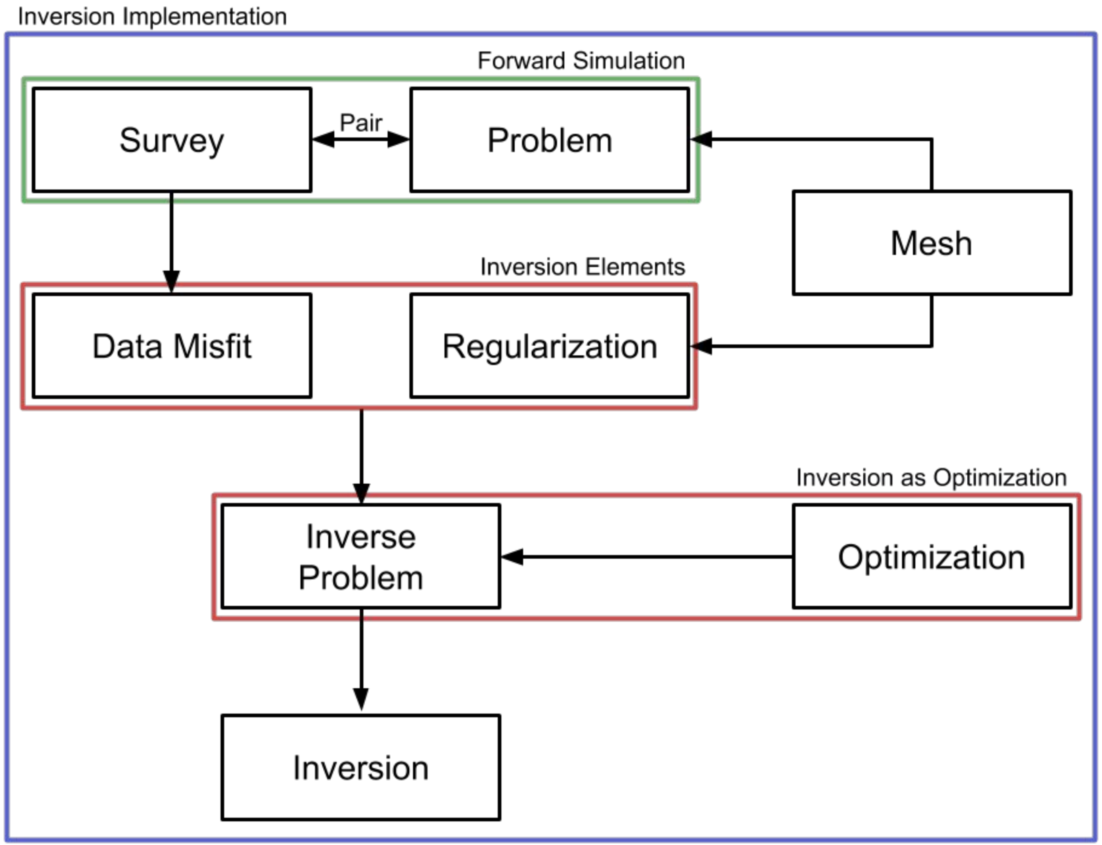
Geophysical Ideas



Realizations

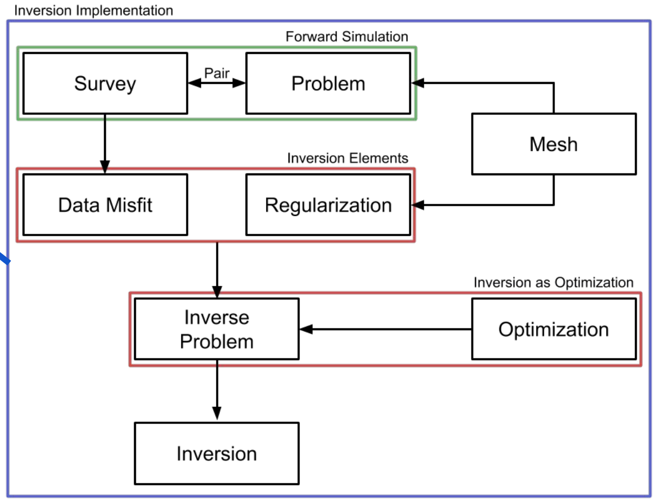


Implementation



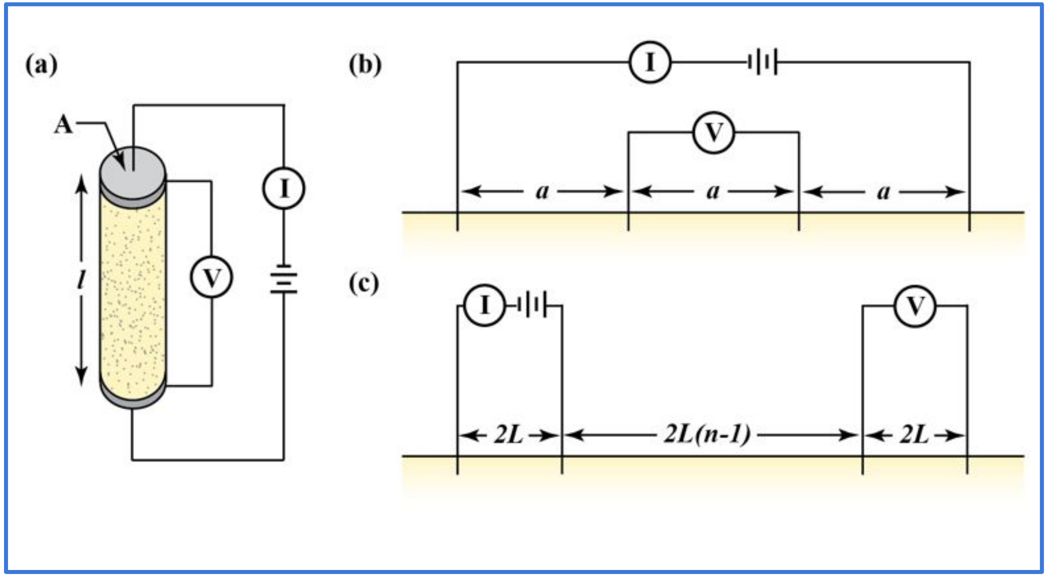
Interactive Geophysics in IPython

```
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
```

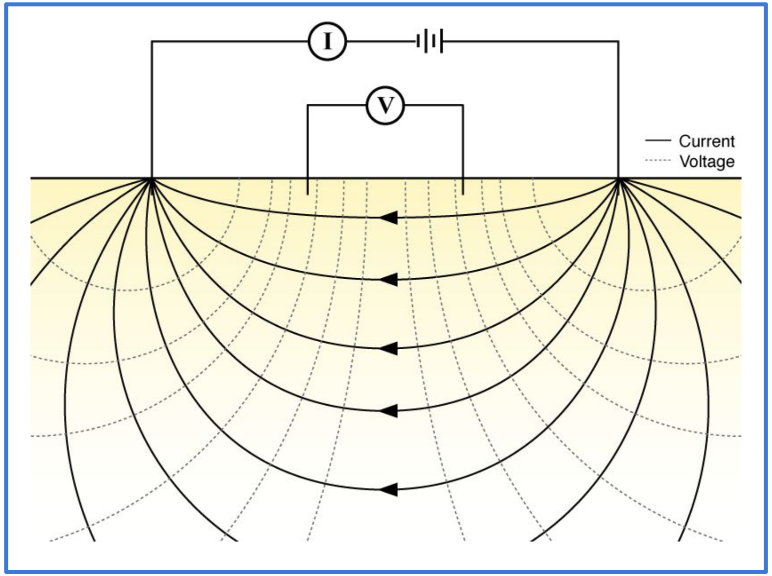


Survey & Problem

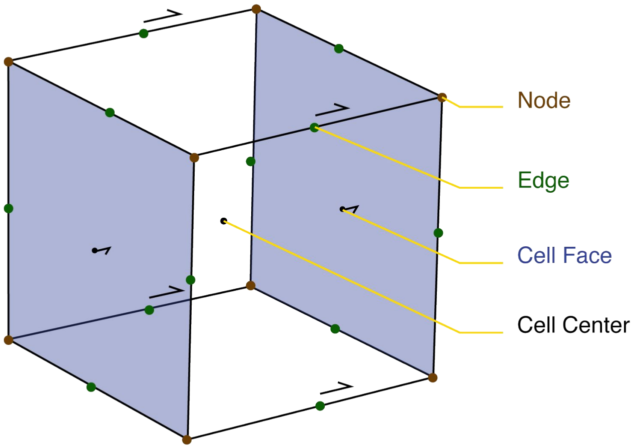
Data collection and geometry

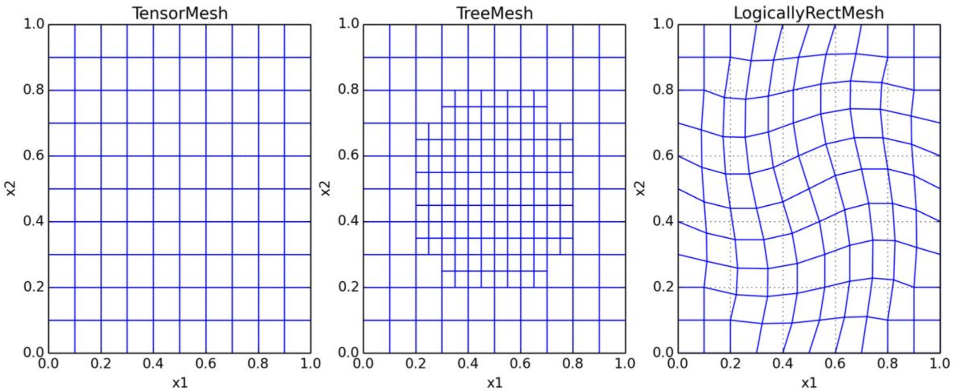


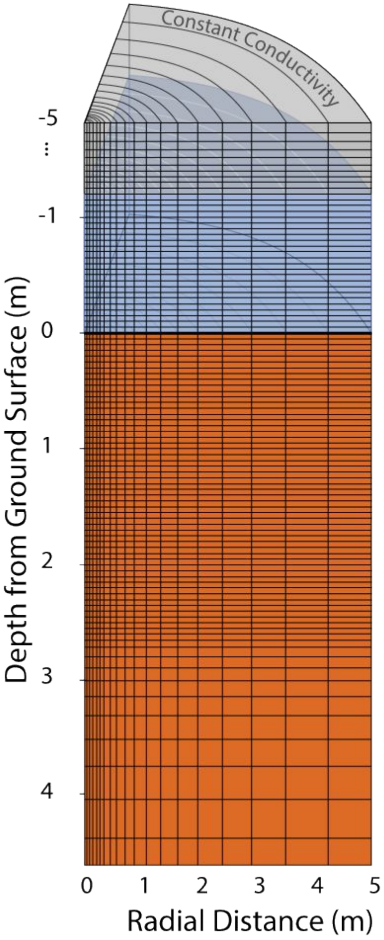
Physics



Finite Volume

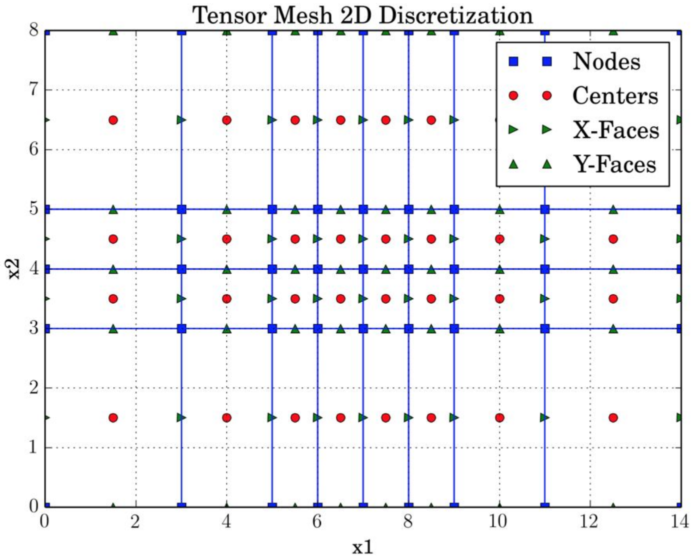






Creating a Mesh

```
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)
```

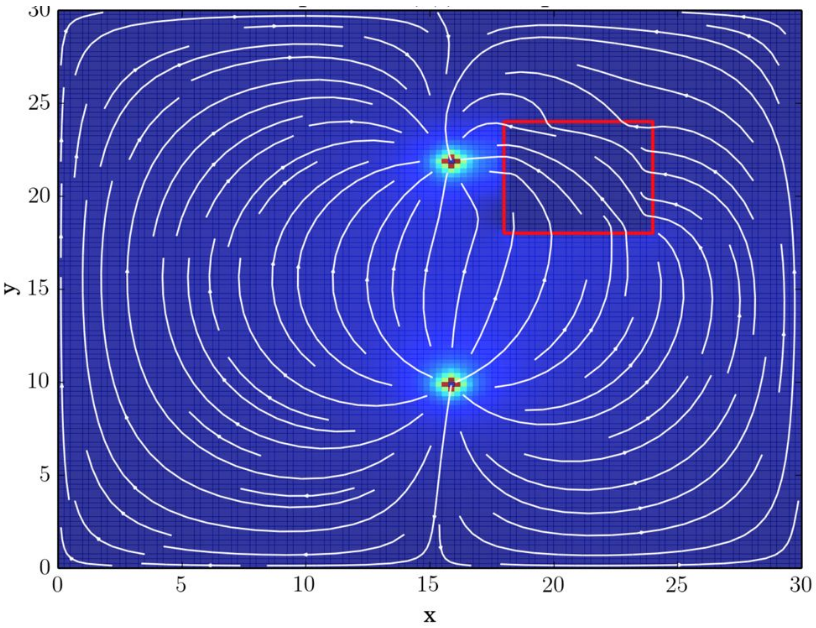


Property or Function
dim, x0
nC, nN, nF, nE
vol, area, edge
gridN, gridCC, ...
faceDiv, edgeCurl, cellGrad
aveF2CC, aveN2CC, etc.
getEdgeInnerProduct()
getInterpolationMat(loc)

DC Resistivity

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

```
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

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

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

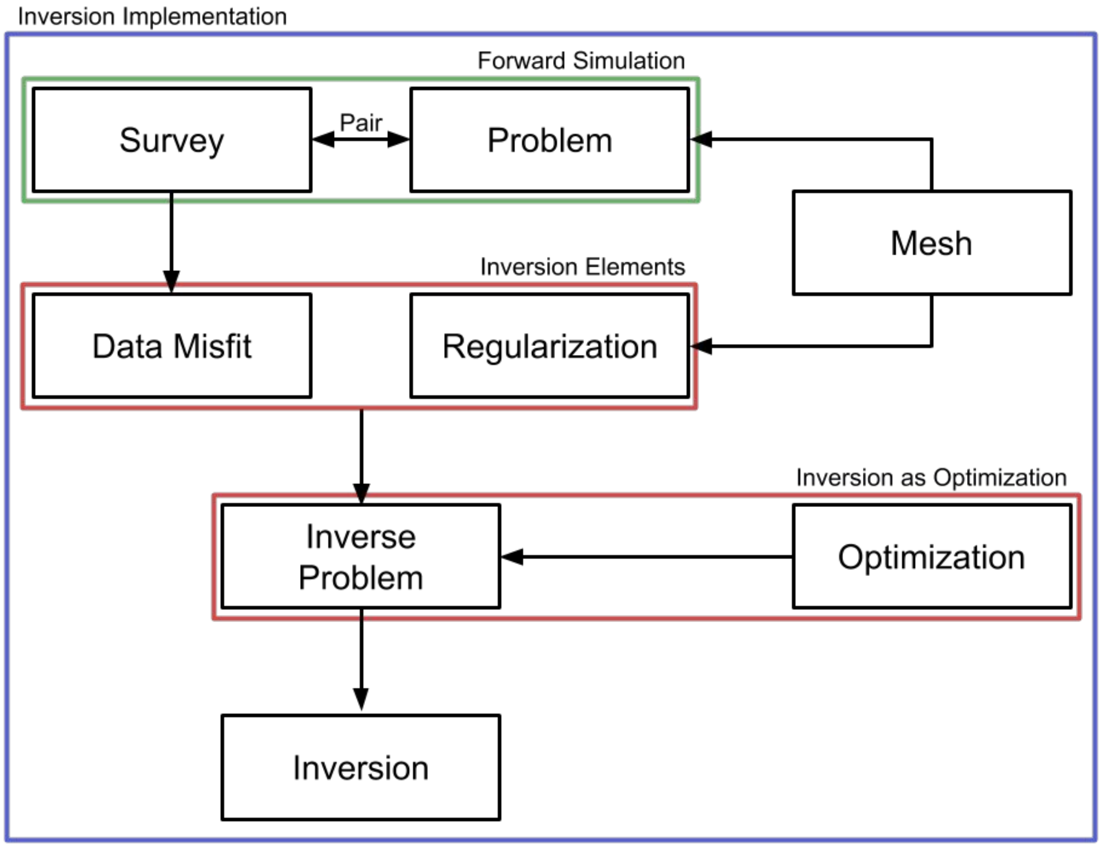
$$\mathbf{C}\vec{e}^{(t+1)} + \frac{\vec{b}^{(t+1)} - \vec{b}^{(t)}}{\Delta t} = 0$$

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

32 lines of code

```
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

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

Regularization

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

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

Inverse Problem

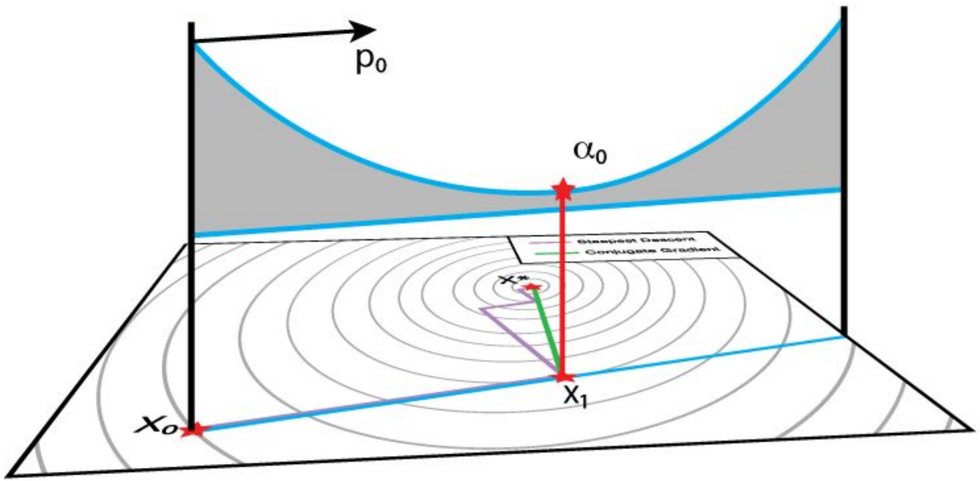
Optimization

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

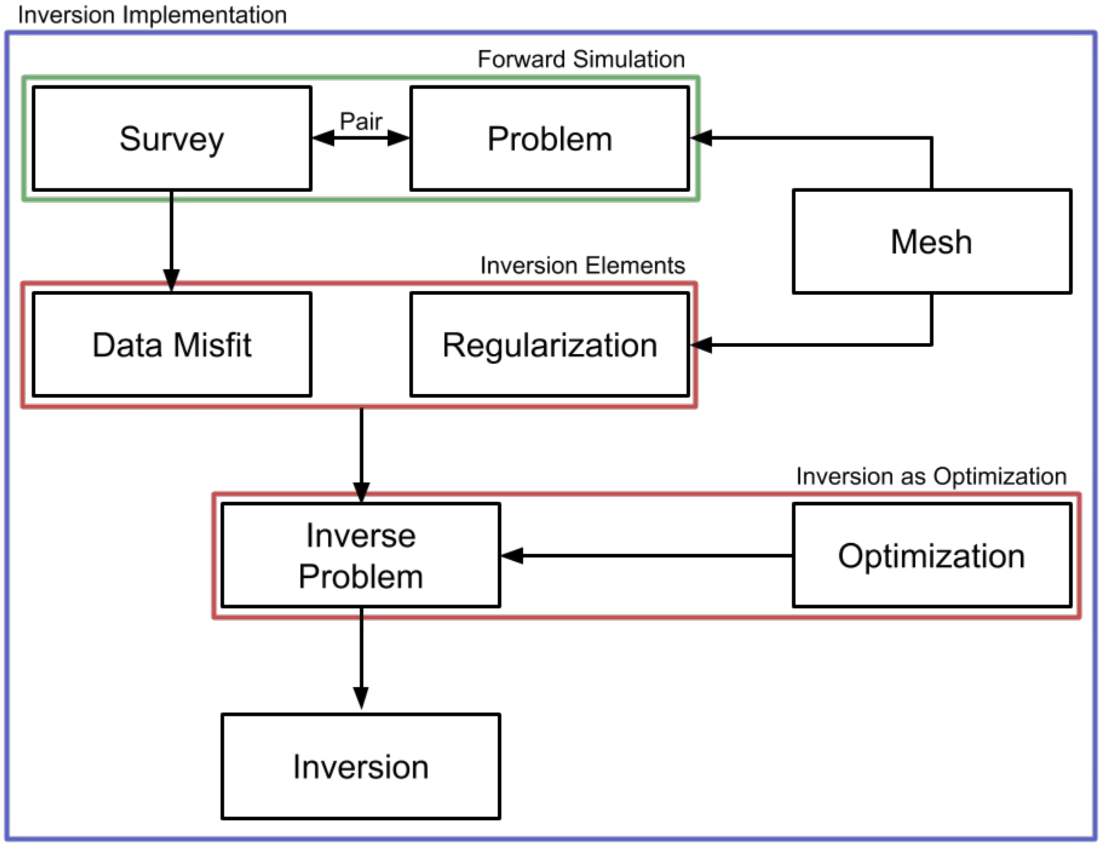
```
In [ ]: Optimization.
Optimization.BFGS
Optimization.GaussNewton
Optimization.InexactGaussNewton
Optimization.IterationPrinters
Optimization.Minimize
Optimization.NewtonRoot
Optimization.ProjectedGNCG
Optimization.ProjectedGradient
Optimization.Remember
Optimization.Solver
```

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

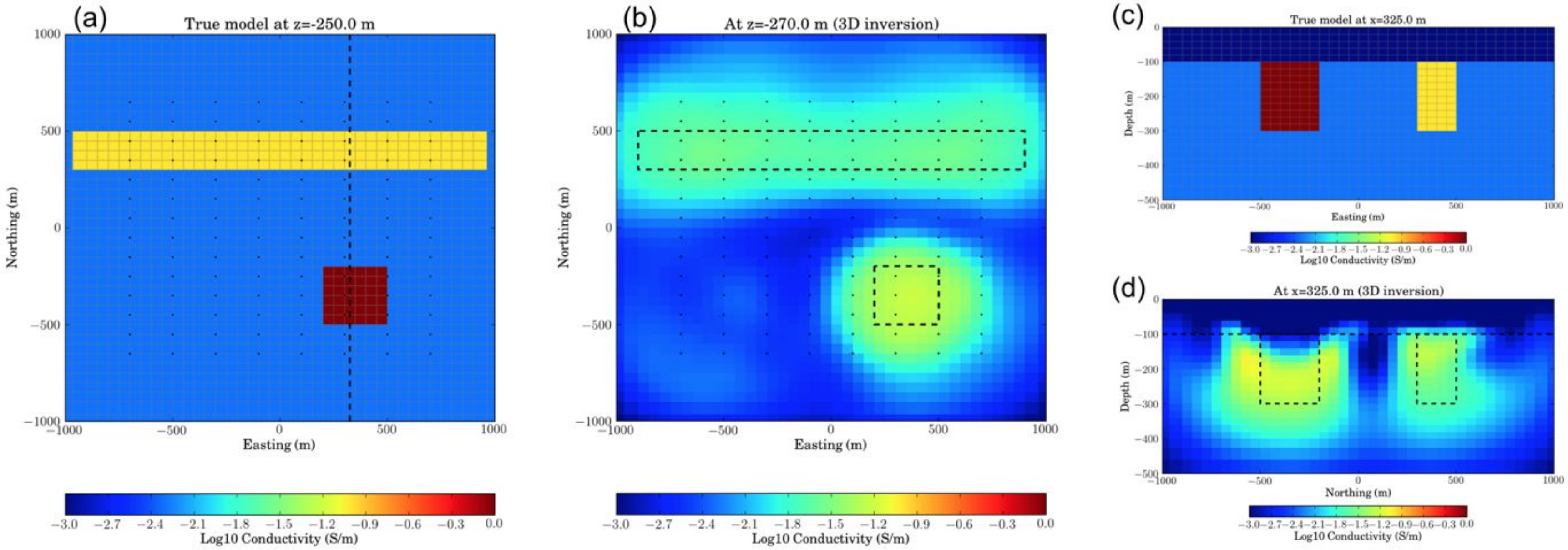
$$\text{s.t.} \quad \phi_\mathrm{d} \leq \phi_\mathrm{d}^*, \quad \mathbf{m}_\mathrm{i}^\mathrm{L} \leq \mathbf{m}_\mathrm{i} \leq \mathbf{m}_\mathrm{i}^\mathrm{H}$$



Bringing it together.



Airborne Time Domain EM Inversion



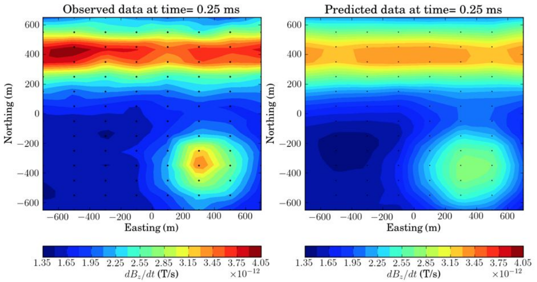
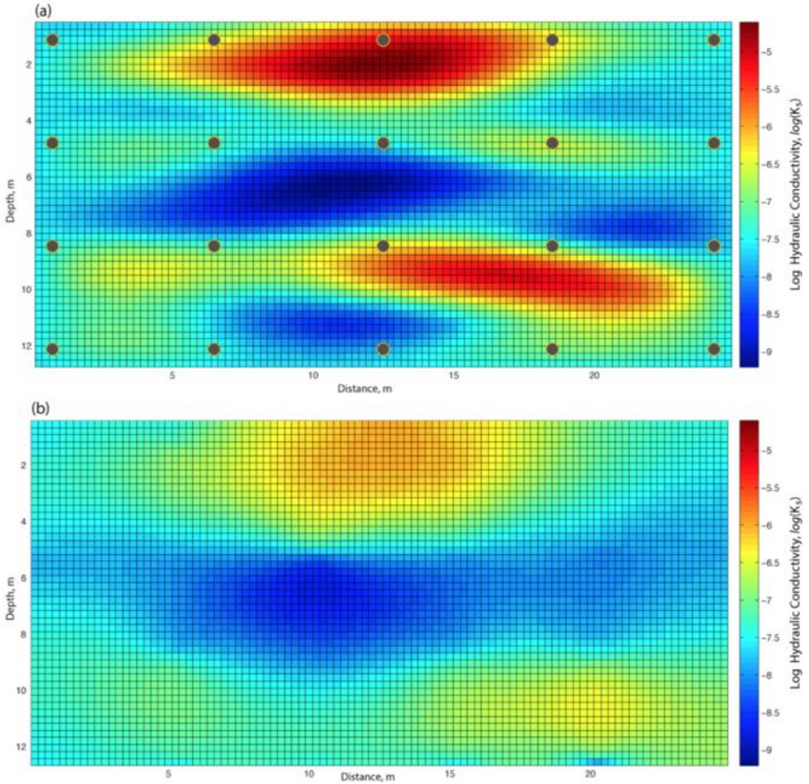
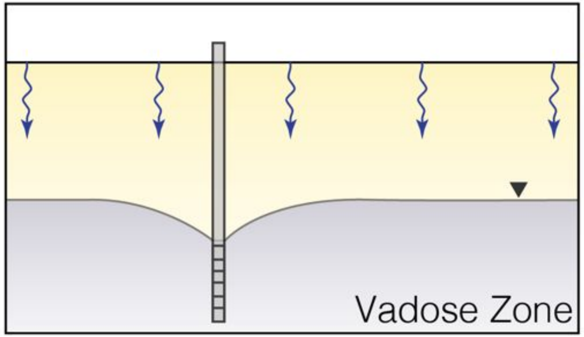


Figure above shows plan views of (a) the true conductivity model and (b) the recovered model, and section views of (c) the true and (d) the recovered conductivity models. Figure to the left shows observed and predicted data. Core cell size: 50×50×20 m, The number of cell: 50×50×48 = 120,000; Reference model: Half-space model with conductivity value, 0.005 S/m; Inexact Gauss-Newton: 13 iterations; Cpu time: 48hrs; Maximum memory usage: 51.2GB; Cpu:Intel(R)Xeon(R) CPU 2.80 GHz; Ram: 64 GB

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

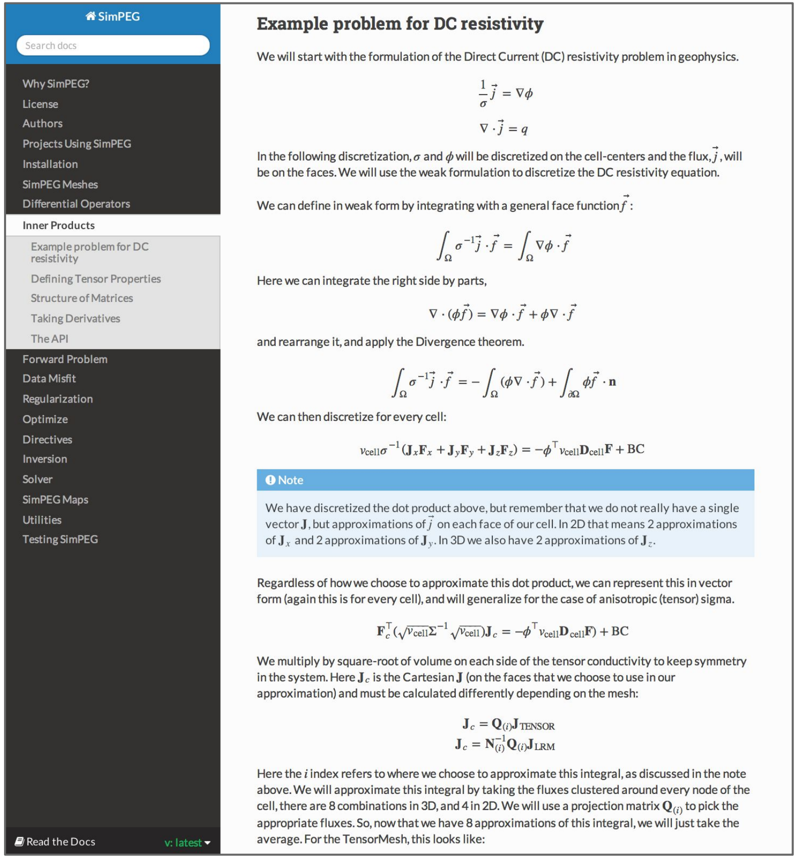
Boundary Conditions: top: -50.7 cm; bottom: -100.5 cm; sides: no-flow

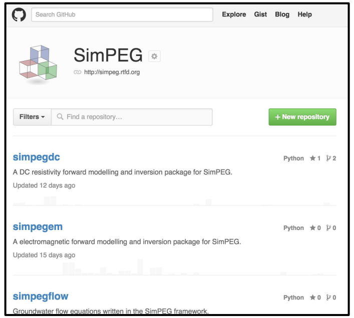
20 iterations, 4 PCG inner iterations; relative data misfit; 11%; CPU-time: 2 hrs

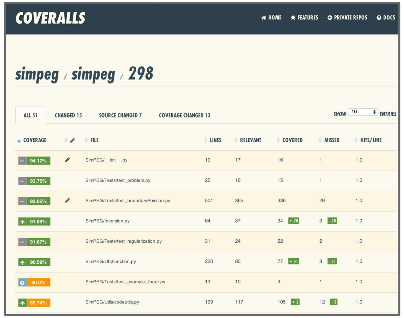
simpeg

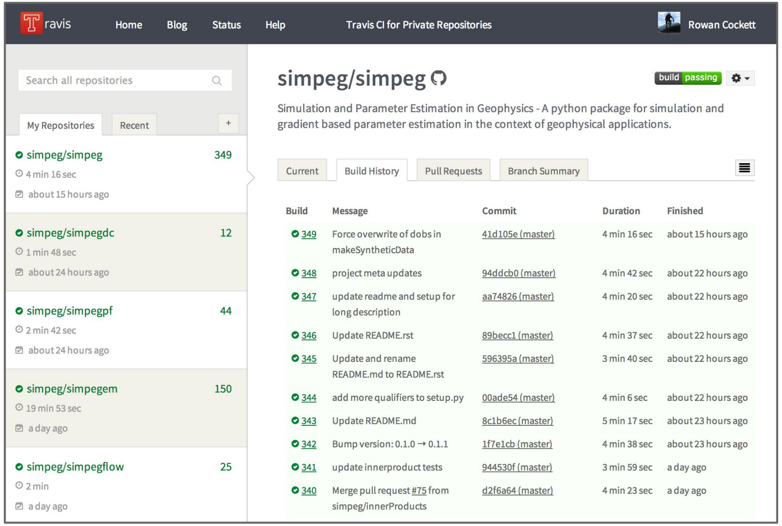
- Framework and toolbox for geophysics
- Motivated by inverse problem
- Guided by terminology
- Modular pieces that allow separation of concerns
- Built in the open to be open

Built in the open



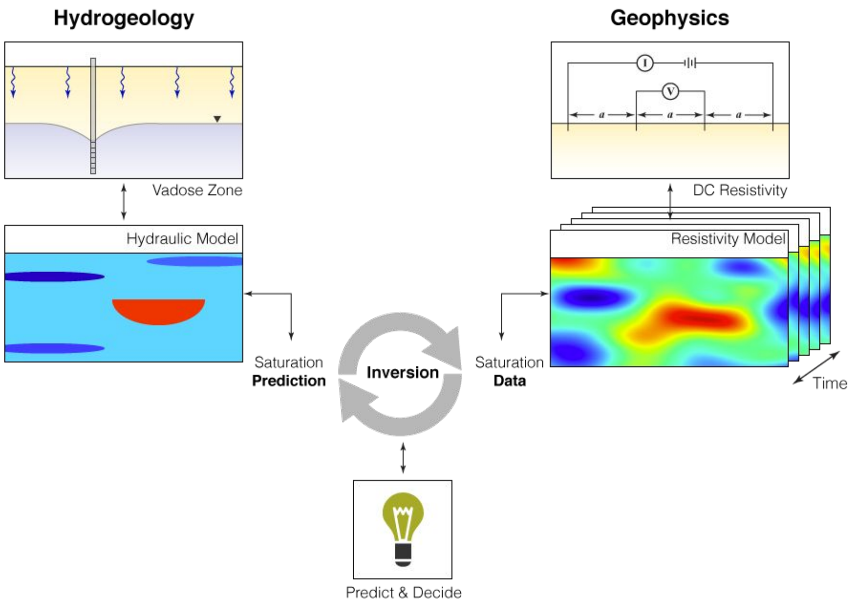


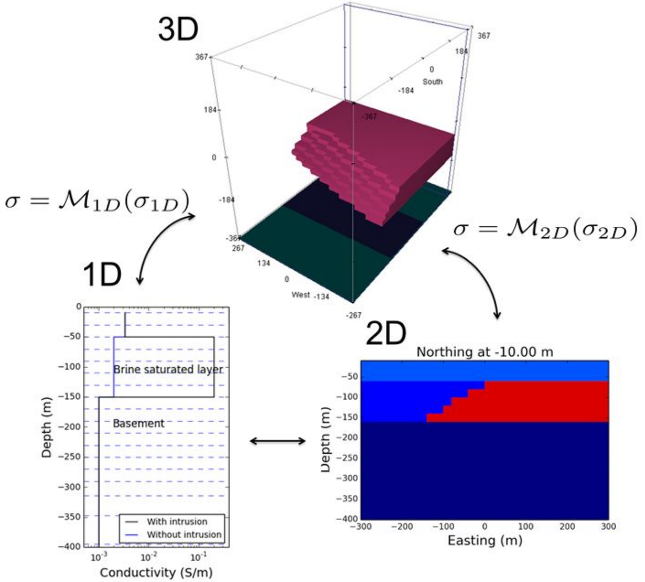




Future

Poster Tomorrow Morning! (GP31A-3682)

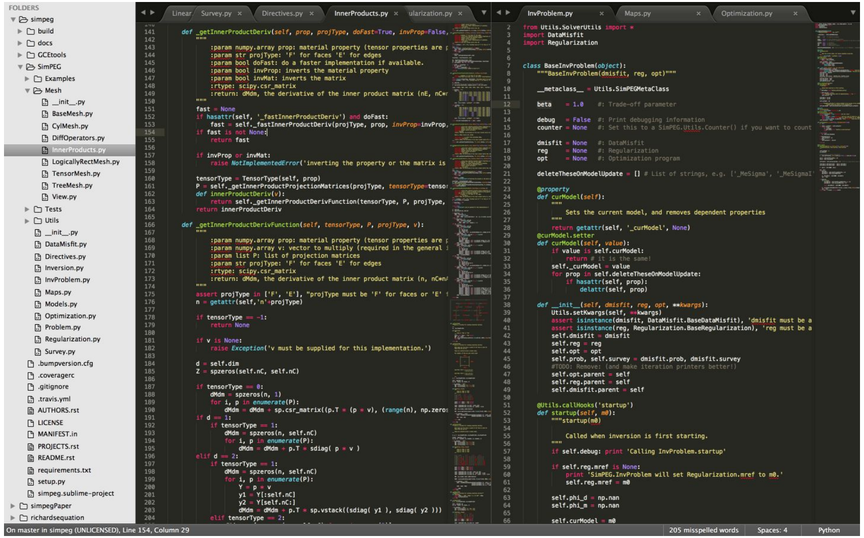


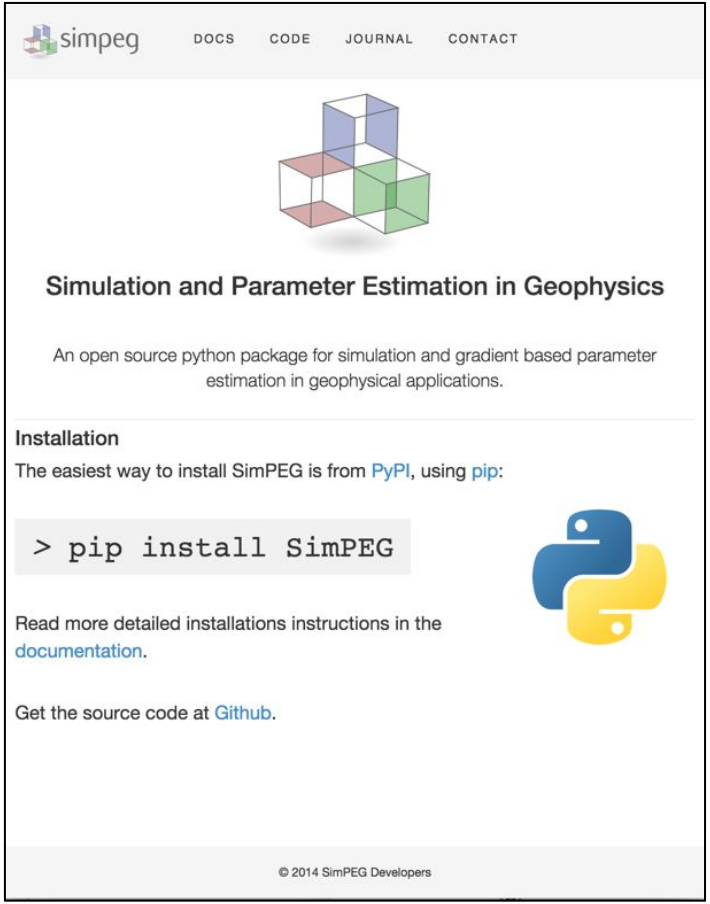


- Integration of information
- Moving between dimensions and physics
- Geophysics education

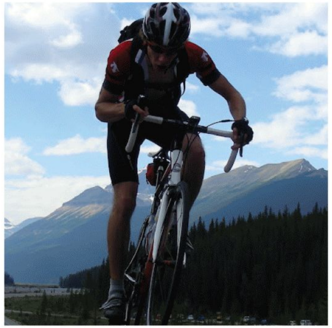
Look at code.

http://simpeg.xyz





People



Rowan Cockett

Seogi Kang

Lindsey Heagy

Dave Marchant

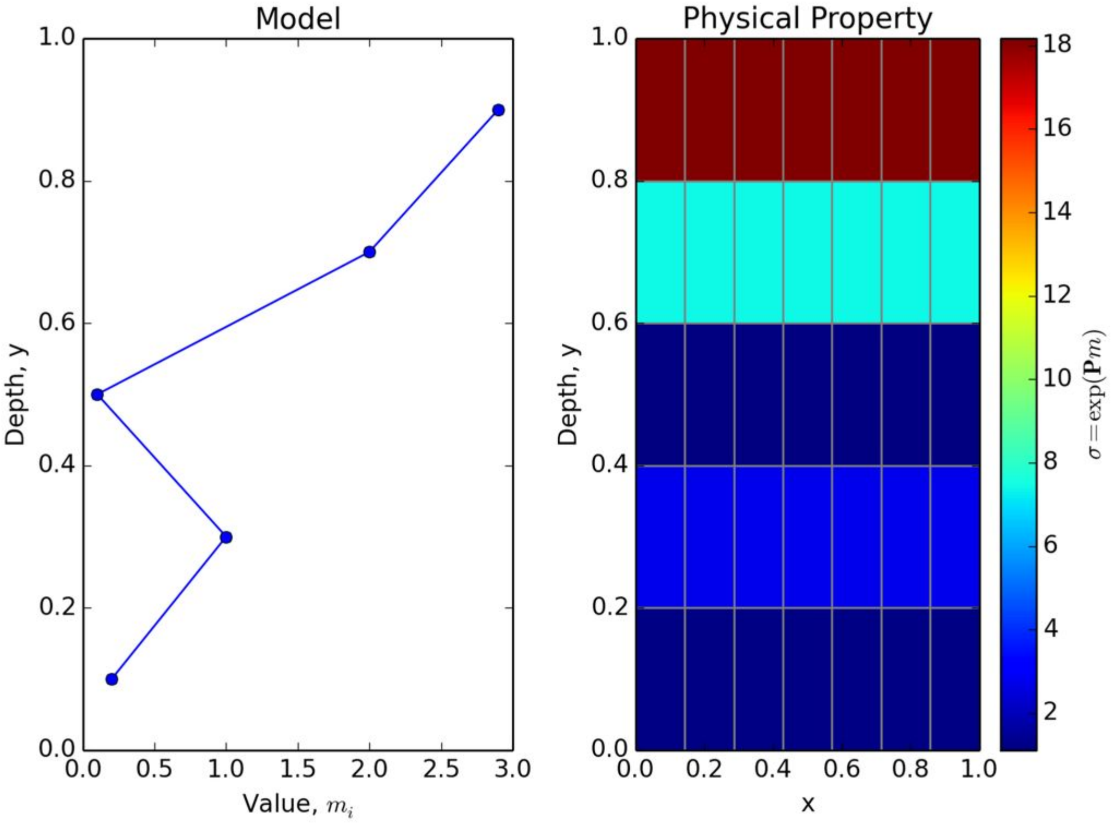
Doug Oldenburg

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
```