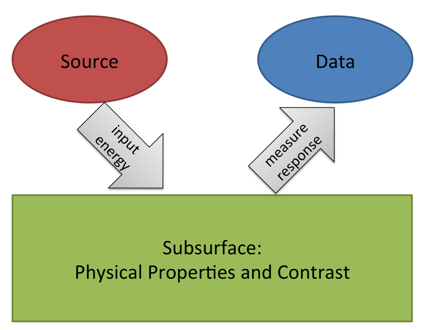
SimPEG: A framework for Simulation and Parameter Estimation in Geophysics

Rowan Cockett[1,3], Seogi Kang[1], Lindsey Heagy[1], Adam Pidlisecky[2,3], Eldad Haber[1], Douglas W. Oldenburg[1]
(1) The University of British Columbia, (2) The University of Calgary, (3) 3point Science Inc.

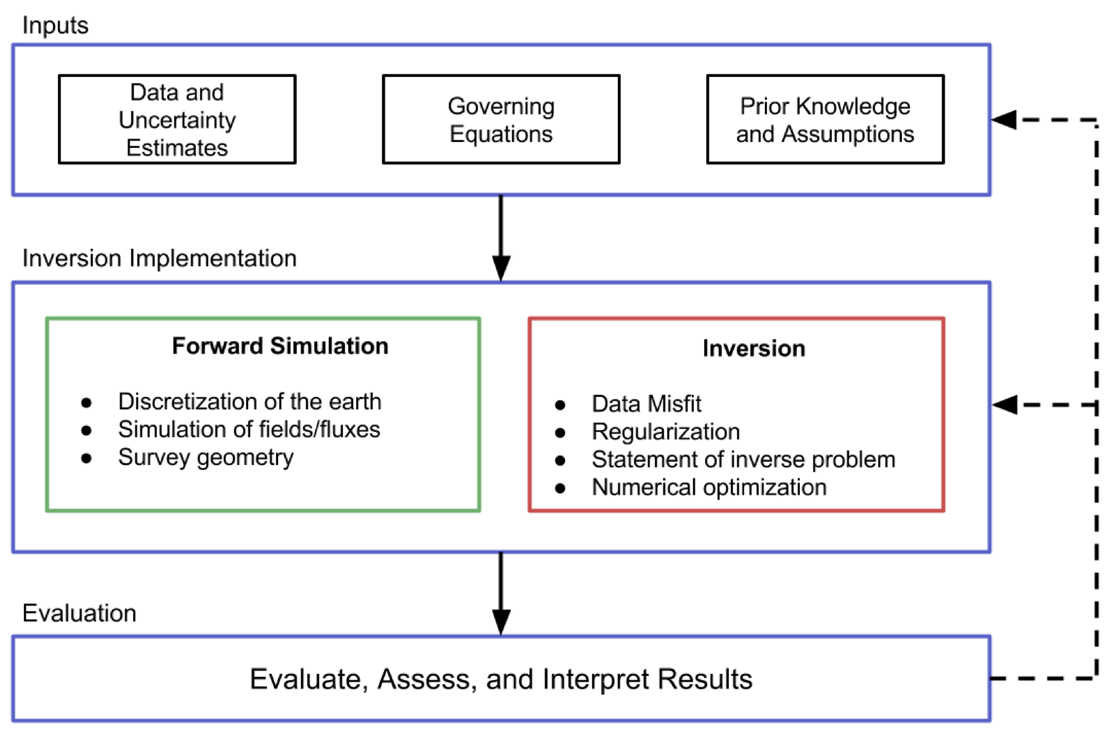
Introduction

What is geophysics?
In many situations, we need to understand and characterize the subsurface in a non-invasive manner



- **Resource exploration**: minerals or oil & gas
- **Environmental**: locating contamination plumes
- **Geotechnical**: mine stability, finding sink holes
- **Monitoring**: aquifers and salt water intrusions, subsurface storage of waste-water, radioactive materials, carbon-dioxide

Why inversions? We measure data, but want information about the physical property distribution that gave rise to those data.



Motivation & Outline

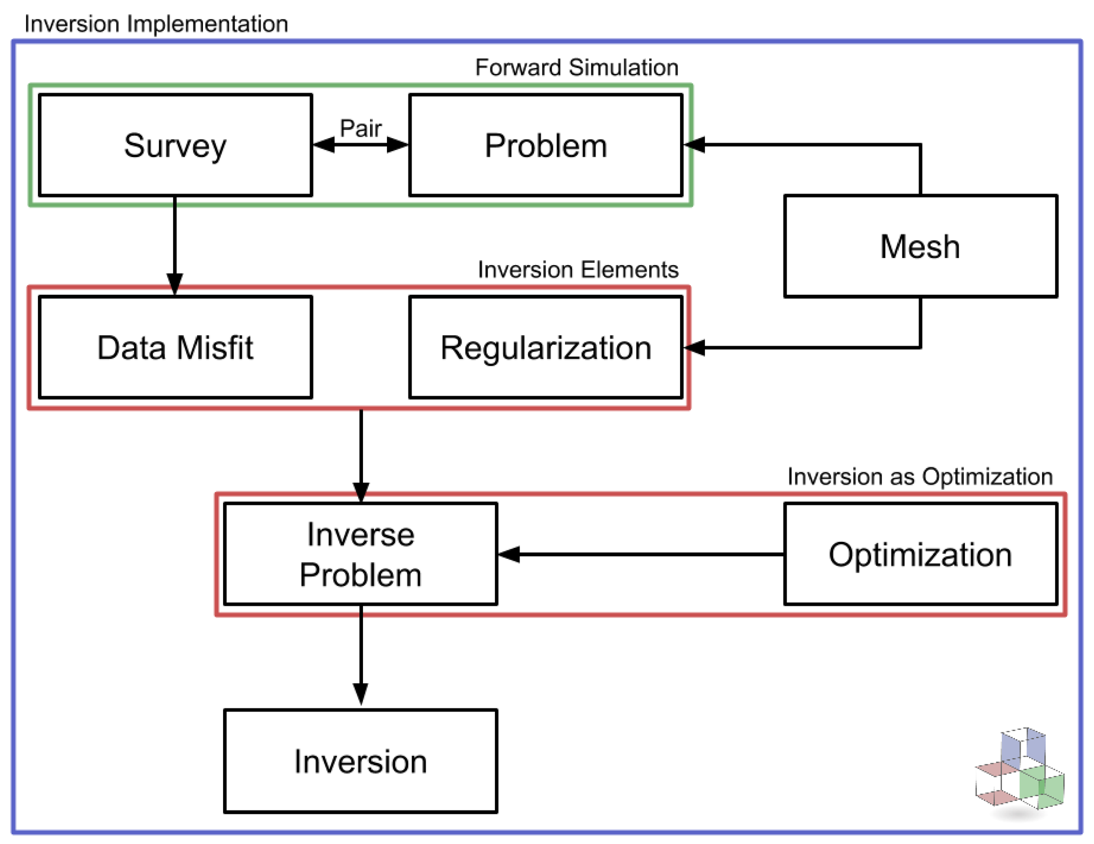
Why?
- Want to **interactively** design inversions
- Want **consistency** between applications
- Moving to **large scale** inversions
- Few well documented **open source** choices

How?
- Everything in **Python**
- Provide **documentation**, **test** everything
- Make everything **modular** and **extensible**
- Provide a **framework** and a **toolbox**

What?
- Interactive finite volume **simulation**
- **Forward** and **inversion** frameworks
- Building **applications**: DC, EM, Flow, ...!

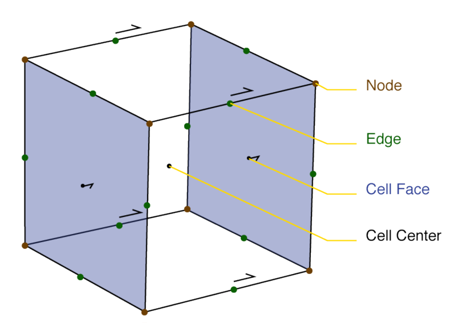
SimPEG Framework

Identify key pieces and build a modular framework. The framework is extensible and has many toolbox style plug-ins that aid in geophysical code development.



Framework was developed by writing multiple inversion packages and cherry picking common features into SimPEG.

Numerical Discretization



Creating a finite volume mesh should be easy, with all numerical operators accessible as matrices!

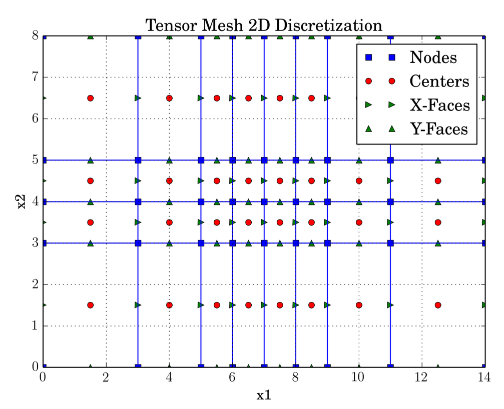
For example, the DC resistivity equations:

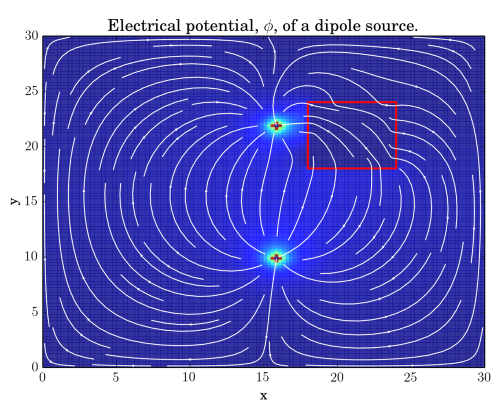
$$\nabla \cdot (-\sigma \nabla \phi) = \mathbf{I}(\delta(\mathbf{r}-\mathbf{r}_{\mathbf{s}+}) - \delta(\mathbf{r}-\mathbf{r}_{\mathbf{s}-}))$$

Creating a finite volume, tensor mesh:

```python
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)
```

```python
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 )
```





Property or Function	Explanation
dim, x0	Dimension and origin of the mesh
nC, nN, nF, nE	The number of cells, nodes, faces, or edges.
vol, area, edge	Geometric measurements for the mesh
gridN, gridCC, ...	Array of grid locations
faceDiv, edgeCurl, cellGrad	Differential operators as matrices
aveF2CC, aveN2CC, etc.	Averaging (e.g. F→CC, averages face variables to cell-centers)
getEdgeInnerProduct()	Inner product operators for material properties
getInterpolationMat(loc)	Interpolation matrix for xyz locations

3D Airborne Time Domain ElectroMagnetic Case Study

We use coincident loop Tx-Rx geometry with db_z/dt component measured at 112 stations with 24 time channels ranging from 0.01-2 ms. We compute solutions of time domain Maxwell's equations:

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

$$\nabla \times \frac{1}{\mu_0}\vec{b} - \sigma\vec{e} = \vec{j}_s. \qquad \mathbf{C}^\top \mathbf{M}^f_{\mu^{-1}}\vec{b}^{(t+1)} - \mathbf{M}^e_\sigma \vec{e}^{(t+1)} = \mathbf{M}^e \vec{j}_s^{(t+1)}$$

```python
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] )
```

Simulating Maxwell's equations, 'from scratch'!

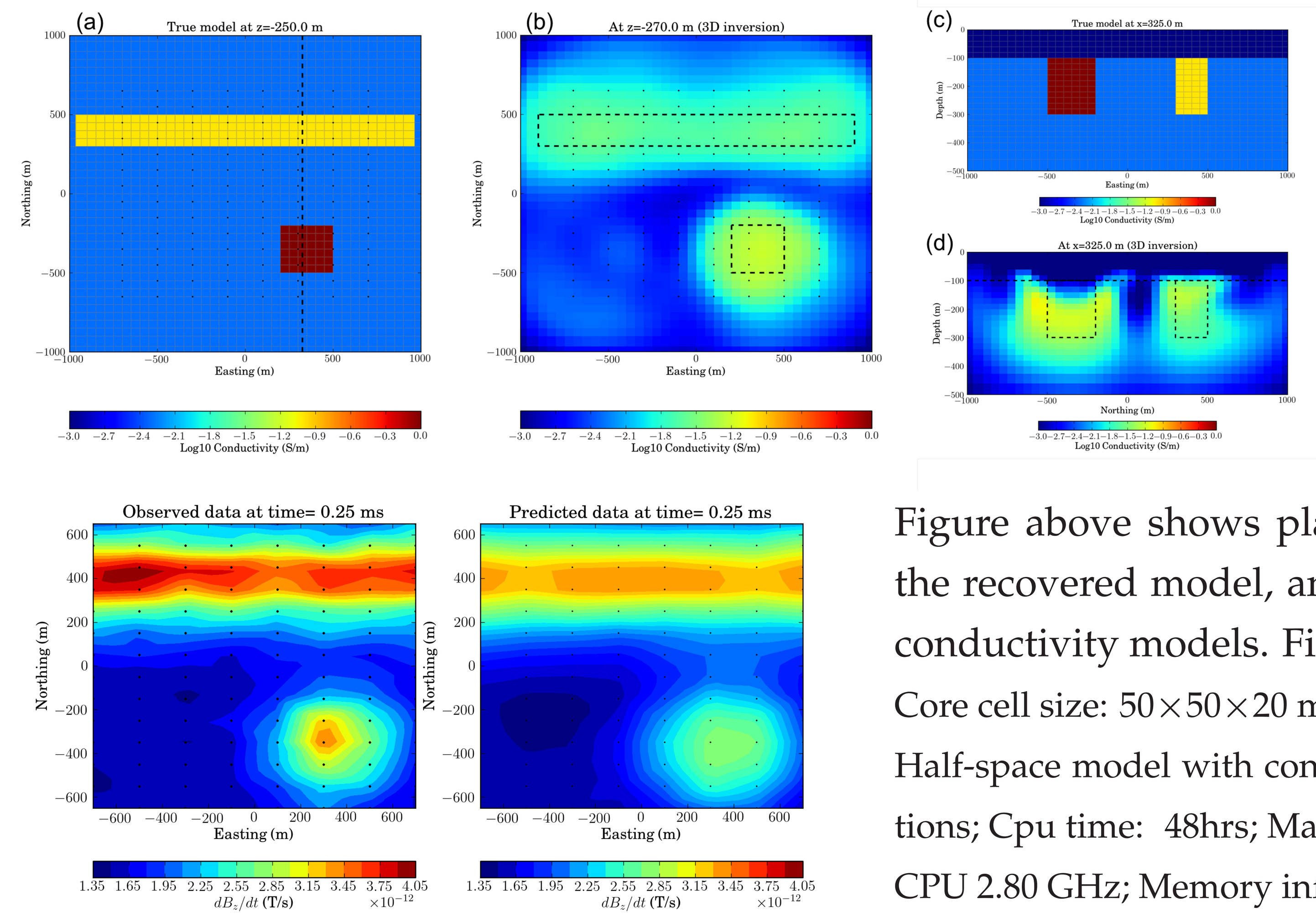


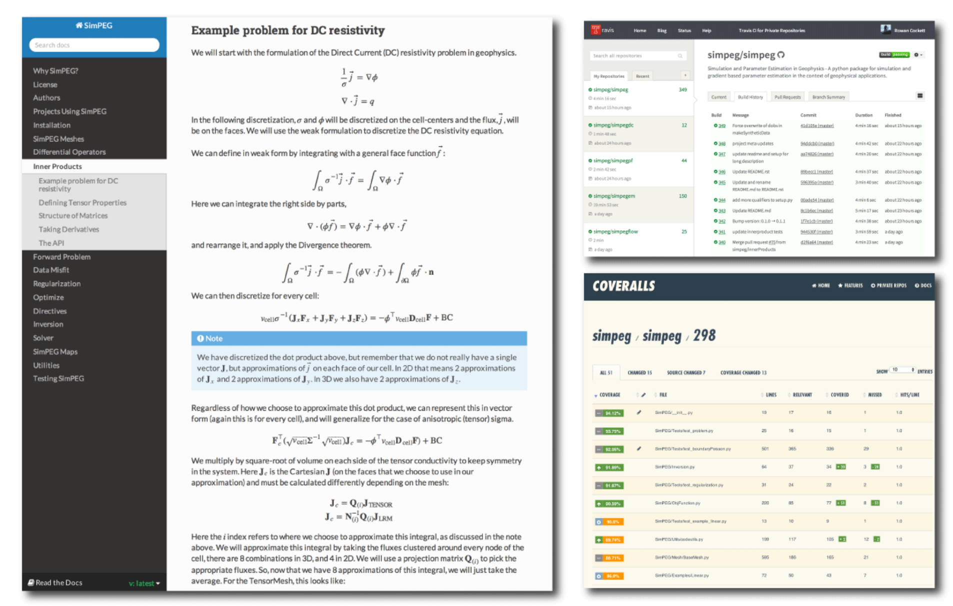
Figure above shows plan views of (a) the true conductivity model and (b) the recovered model, and section views of (c) the true and (d) the recovered conductivity models. Figure to the left shows observed and predicted data.

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: 51.2 GB; Cpu info: Intel (R) Xeon (R) CPU 2.80 GHz; Memory info: 64 GB

Lessons Learned

Where we focused and what we learned:
- Make it fast for the researcher. Docs, Tests, and UI/UX!
- Build a modular framework that doesn't lock you in.
- Minimal interfaces & fight to keep the package small.
- Focus on toolbox rather than any one geo-application.
- Writing the code should aid in understanding the geophysical inversion methodology. If it doesn't, refactor.

What was difficult?
- Creating a general and extensible framework.
- Access to robust large-scale matrix solver packages.
- Memory consumption for large-scale inversion.



GitHub, Travis, Coveralls, Sphinx, ReadTheDocs, MathJax, ...

Getting Started

Install: `pip install SimPEG`
Documentation: `http://simpeg.rtfd.org/`
Source: `http://github.com/simpeg/`

Moving Forward

- Publish the paper and port it to the documentation. Focus on geophysics education!
- Continue to develop application specific code (simpegEM, simpegFLOW, simpegDC, etc.)
- Extend and investigate joint-inversion techniques in SimPEG (e.g. DC resistivity and fluid flow)

Take Home Points

- Rapid development of geophysical methods.
- Consistency leading towards integration.
- Play with SimPEG! ☺