Advances in open-source software for 3D electromagnetics using SimPEG

Lindsey J. Heagy
University of British Columbia, Geophysical Inversion Facility
lheagy@eoas.ubc.ca

Johnathan Kuttai
University of British Columbia, Geophysical Inversion Facility
jkutt@eoas.ubc.ca

Devin Cowan
University of British Columbia, Geophysical Inversion Facility
dcowan@eoas.ubc.ca

Joseph Capriotti
University of British Columbia Geophysical Inversion Facility
jcapriot@eoas.ubc.ca

Seogi Kang
Department of Geophysics, Stanford University
sgkang09@stanford.edu

Dominique Fournier
Mira Geoscience, Vancouver, BC
dominiquef@mirageoscience.com

Douglas W. Oldenburg
University of British Columbia, Geophysical Inversion Facility
doug@eoas.ubc.ca

SUMMARY

Open-source software is increasingly being adopted by the geophysics community. Their emergence has greatly reduced the time required for students and researchers to be able to implement and explore new ideas, and having new developments implemented in an open-source project facilitates technology transfer and collaboration between research and commercial organizations. SimPEG is an open-source project for geophysical simulations and inversions. In this abstract, we provide an overview of the capabilities and recent advancements in SimPEG that are relevant to the airborne electromagnetics community.

Key words: Numerical simulations, inversion, computational methods, software.

INTRODUCTION

There are now a number of simulation and inversion codes that are open-source and allow use and adaptation by academic and commercial groups. An open-source model of development and dissemination of scientific software facilitates reproducibility and extension of the code to new applications, reducing start-up time for students and researchers. These codes can also be valuable to groups with established codes, as they can be an additional tool for testing, or for extending functionality. Werthmueller, *et al.*, (2020) provides an overview of 4 open-source codes for simulating 3D electromagnetic (EM) problems. SimPEG is one such code that supports forward simulation and inversion of time and frequency domain EM data, including for controlled and natural sources (Cockett *et al.*, 2015). The SimPEG project was started in 2013 with the aim of accelerating research by building a modular, open-source code-base, and importantly, by fostering a community of researchers interested in solving inverse problems with geophysical data.

SimPEG supports a variety of data types, including gravity, magnetics, DC resistivity, induced polarization and electromagnetics. Methods for solving 3D EM forward and inverse problems were implemented 3D EM early in the project (Heagy *et al.*, 2017), but much progress has been made in recent years to improve the efficiency and expand the functionality of the codebase. In this abstract, we highlight the advances relevant to the airborne EM community. Using an example of a synthetic Z-Axis Tipper survey, we illustrate the forward simulation and inversion capabilities of SimPEG and compare the results to those obtained with UBC-GIF Fortran software.

ELECTROMAGNETICS IN SIMPEG

SimPEG provides a framework for forward simulation and gradient-based inversion of geophysical data. With respect to EM methods, it contains functionality for simulating and inverting Maxwell's equations in 1D and 3D (and for some problems, 2D).

The 1D forward simulations in time and frequency are based on semi-analytic solutions in the wavenumber-frequency domain and leverage empymod for the digital filters (Werthmüller, 2017). Inversions can be laterally or spatially constrained and use standard L2 norms or sparse / compact L1 or L0 norms. For a recent example, see Kang *et al.* (2022).

For 3D simulations and inversions, SimPEG uses a staggered-grid finite volume approach to discretize Maxwell's equations in space. Supported mesh types include tensor, cylindrical, and OcTree meshes as well as curvilinear meshes. Tetrahedral meshes are currently being developed. The meshing and finite volume operators are contained in the discretize package, which is a part of the SimPEG project. For both the frequency and time domain simulations, there are multiple formulations that can be used. The E and B formulations discretize the electric field on edges and the magnetic flux density on faces, while the H and J formulations discretize the magnetic field on edges and the current density on faces. Table 1 provides a summary of the different implementations. Having multiple implementations has proven valuable for testing of the code, as it enables us to check for internal consistency. Further, some discretizations are better-suited for a given problem than others. For example, if simulating or inverting with fully anisotropic physical properties, the choice of formulation and physical property (conductivity or resistivity) matters; we would use conductivity with the E-B formulation and resistivity with the H-J formulation to avoid the need to invert the physical property matrix, which becomes dense when properties are anisotropic. All implementations use a right-handed coordinate system with z-positive up.

In contrast to codes that are provided as an executable where the user interacts with the code primarily through input and output files (possibly through a graphical interface), SimPEG is provided as a Python library where the user has access to all components of the simulation and inversion programmatically. For example, in EM, this enables the user to access and visualize fields and fluxes for a given simulation. This ability has proven to be valuable for understanding the physics in research and educational applications (Oldenburg *et al.,* 2021).

Table 1: Summary of the E-B and H-J formulations of Maxwell's equations.

Formulation	Time domain equations	Frequency domain equations	Boundary conditions	Physical properties	Fields, fluxes
E-B	$\nabla \times \vec{e} = -\frac{\partial \vec{b}}{\partial t}$ $\nabla \times \mu^{-1}\vec{b} - \sigma\vec{e} = \vec{j_s}$	$\nabla \times \vec{E} = -i\omega\vec{B}$ $\nabla \times \mu^{-1}\vec{B} - \sigma\vec{E} = \vec{j_s}$	$\vec{b} \times \vec{n} = 0\|_{\partial\Omega}$ $\vec{B} \times \vec{n} = 0\|_{\partial\Omega}$	σ, μ^{-1} : cell-centers	$\vec{e}, \vec{E}$: edges $\vec{b}, \vec{B}$: faces
H-J	$\nabla \times \rho\vec{j} = -\mu\frac{\partial \vec{h}}{\partial t}$ $\nabla \times \vec{h} - \vec{j} = \vec{j_s}$	$\nabla \times \rho\vec{J} = -i\omega\mu\vec{H}$ $\nabla \times \vec{H} - \vec{J} = \vec{j_s}$	$\vec{j} \times \vec{n} = 0\|_{\partial\Omega}$ $\vec{J} \times \vec{n} = 0\|_{\partial\Omega}$	ρ, μ : cell-centers	$\vec{j}, \vec{J}$: faces $\vec{h}, \vec{H}$: edges

Accessing the full flexibility of SimPEG requires that the user be comfortable working with Python code. However, commercial groups have developed graphical interfaces that can be used to run the codes, making them more accessible to a wider community of users. The open-source MIT license facilitates commercial use and adaptation; as a project, we encourage collaboration and participation from commercial and academic groups.

Our initial implementation efforts focussed on developing the modular framework and structure of the codebase to facilitate research. As such, flexibility, and ease of extending the code are priorities. The codebase has matured with usage and contributions from the community, and in conjunction, there have been advances in the Python ecosystem that can be leveraged to improve efficiency. In particular, SimPEG now has capabilities to be parallelized using a domain-decomposition approach similar to that described in Yang *et al.* (2014). The forward simulation is broken up so that it can be run on a collection of smaller meshes that cover a subset of the survey, while the set of parameters we invert for are still represented on a global mesh as shown in Figure 1. Frequency domain problems can also be parallelized over frequency. We leverage parallelization frameworks in Python, such as Dask (Rocklin, 2015), to enable distributed computing on a cluster or on the cloud.

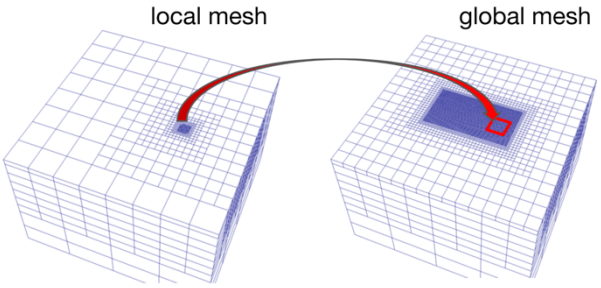

Figure 1. Domain decomposition approach where we invert for a model on the global mesh but solve the forward problem and compute sensitivities on the local mesh.

In the sections that follow, we describe the time, frequency, and natural source codes implementations in SimPEG, highlighting advancements relevant to the AEM community. Following this, we provide a synthetic example of an airborne Z-Axis Tipper EM (ZTEM) simulation and inversion. We compare the SimPEG results with those obtained using the UBC Fortran code E3DMT version 2 (https://e3dmt.rtfd.io). Finally, we conclude with a discussion of next steps for the SimPEG 3D EM codes.

Controlled Source Time Domain EM

The time domain implementation uses a backward Euler discretization in time. The user provides the time discretization. Sources can be airborne, on the ground or positioned in boreholes, and they can be inductive or galvanic sources. For wires or large-loop sources, the wire path can be provided. There are also other simple source types such as dipoles and circular loops that are implemented. The default simulation uses a step-off waveform, but the user can specify different source waveforms. Receivers can measure electric field, db/dt or magnetic fields, and these receivers can have arbitrary orientations.

Controlled Source Frequency Domain EM

Similar to the time-domain code, in the frequency domain code, the source can be galvanic or inductive, and again sources and receivers can be positioned arbitrarily. Relevant to AEM surveys, receivers can be at any orientation, which enables TMI data to be used, such as in SAM-type surveys. Receivers can measure total or secondary fields, such as for DIGHEM or RESOLVE surveys.

Natural source frequency domain EM

The natural-source EM code leverages the same forward simulation engine as the controlled source FDEM codes. The natural source EM code uses a primary-secondary approach, with a 1D primary field. A total field implementation, which uses boundary conditions to capture the effects of the inducing field is in progress. For natural source methods, the data are transfer functions. SimPEG includes common transfer functions such as MT or Tipper data and the ability to support arbitrary transfer functions. This includes surveys with many B-field measurements and a few E-field measurements, which is relevant to AEM surveys, as well as surveys with many E-field measurements and a few B-field measurements, which might be collected as a part of a DCIP survey. Note that since the same simulation code is shared for all frequency domain methods, it is straightforward to use a controlled source with data that are transfer functions, for example in a CSAMT survey where the wire location is known.

Inversion

We formulate the inverse problem as an optimization problem where we minimize a data misfit and model norm term

$$\min_m \phi(m) = \phi_d(m) + \beta\phi_m(m) \quad s.t. \quad \phi_d \leq \phi_d^*$$

SimPEG contains classes for defining the data misfit (ϕ_d) and regularization (ϕ_m); options for the regularization include standard L2 norms, as well as capabilities for sparse and compact norms (Fournier & Oldenburg, 2019). SimPEG also contains routines for performing optimization and updating parameters such as the trade-off parameter β during the inversion. We generally use second-order optimization methods, such as Inexact Gauss Newton, or Projected Gauss Newton when bound constraints are imposed on the model. To use such methods requires that we compute products of the sensitivity and its adjoint with a vector. For the EM simulations implemented in SimPEG, we have the ability to form and store the sensitivity matrix, which is useful for problems that are sufficiently small or when domain-decomposition is used to break up the problem. Alternatively, in memory-limited applications, we can opt not to form the sensitivity matrix and instead compute products of the sensitivity and its adjoint with a vector. With these pieces, we can then go ahead and perform an inversion.

EXAMPLE

Setup

As an example, we use the L-block model shown in Figure 2. The block is 1 Ωm and it is embedded in a 100 Ωm half-space, it extends from 400m to 1200m depth. We discretize the model using an OcTree mesh with core cells of size 200m x 200m x 100m. We consider a ZTEM survey with frequencies 10 Hz, 50 Hz, and 200 Hz. The data locations are indicated by the white dots in Figure 2.

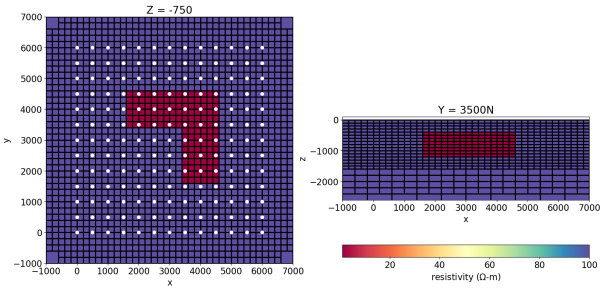

Figure 2. Model of a conductive L-shaped block in a halfspace. The block is 1 Ωm and the background is 100 Ωm. The block extends vertically from a depth of 400 m to 1200 m.

Forward Simulation

The forward simulation is performed using a primary-secondary approach with the E-formulation; a 1D primary is computed using a 100 Ωm halfspace. With SimPEG, users can choose to access the fields and fluxes that are computed as a part of the forward simulation. In Figure 3, we show the current density and anomalous magnetic field due to the conductive target for the at 10 Hz for a single source polarization. Currents are concentrated in the conductive target, producing an anomalous magnetic field, which is reflected in the data that we measure.

Figure 4 shows the simulated data. On the left we show the real and imaginary components of both Tzx and Tzy at 10 Hz; on the right, we show a corresponding profile of data at all three frequencies. The dashed lines show the data computed using the UBC E3DMT code. Both codes discretize electric fields on edges and magnetic flux density on faces. The exact same mesh and model were used in both codes. We can see the solutions are in good agreement.

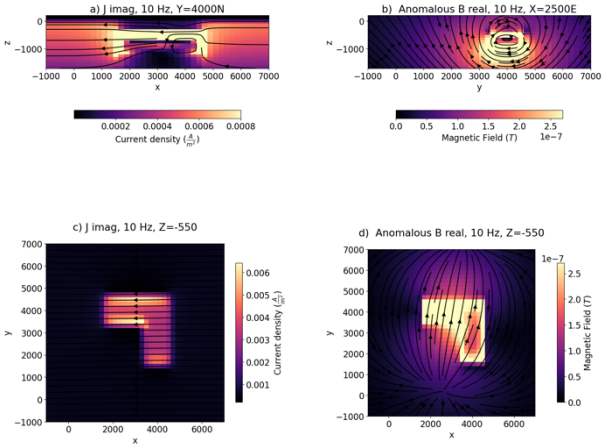

Figure 3. (a) Cross section of the imaginary part of the current density along the line Y=4000N for a single source polarization; (b) orthogonal cross section of the real part of the anomalous magnetic flux density along X=2500E. (c) Depth slice of the imaginary part of the current density and (d) depth slice of the magnetic flux density.

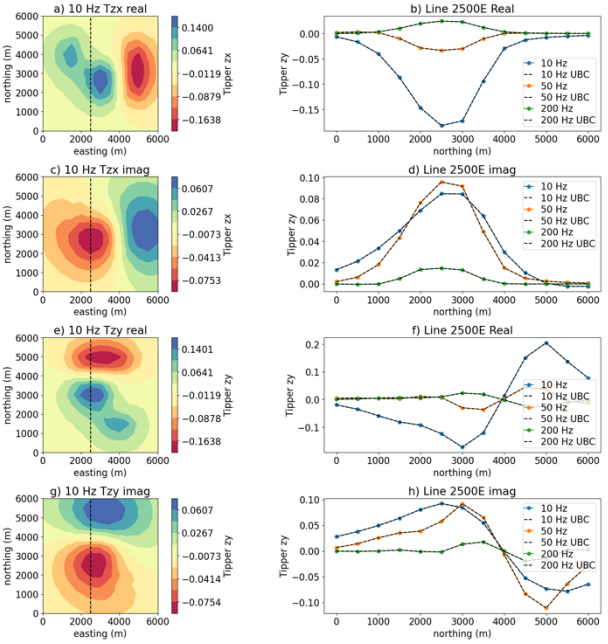

Figure 4. Simulated tipper data for the L-block model. Panels on the left show the 10 Hz data while panels on the right show a profile of data for all 3 frequencies. The colored lines show the SimPEG results while the black dashed lines show the UBC E3DMT simulation results.

Inversion

We invert the simulated data using both SimPEG and E3DMT. There are 2028 data total. A 5% relative error is assigned to all frequencies, a floor of 0.01 is used for the 10 Hz and 50 Hz data, while a floor of 0.005 is used for the 200 Hz data. We use a standard L2-norm regularization with smallness and first-order smoothness applied in each direction (α_s=2.5e-5; α_x = α_y= 1; α_z = 0.25). We use the same β-cooling schedules in both codes, starting with a value of 0.01 and decreasing by a factor of 5

every 3 iterations. The reference and starting models are a 100 Ωm half-space.

Both inversions converged to comparable misfits. The SimPEG inversion converged in 7 iterations to a final χ-factor of ϕ_d/ϕ_d^*= 0.47. The UBC E3DMT inversion converged in 8 iterations to a χ-factor of 0.55. The SimPEG inversion ran with the simulation parallelized over frequencies using Python's built-in multiprocessing library. It completed in 15 minutes. The UBC inversion was parallelized over frequencies using MPI and it took 11 minutes.

Figure 5 shows the inversion results obtained using SimPEG (left) and the UBC E3DMT code (right). The recovered models are comparable, resolving a smoothed version of our L-shaped target, as is expected.

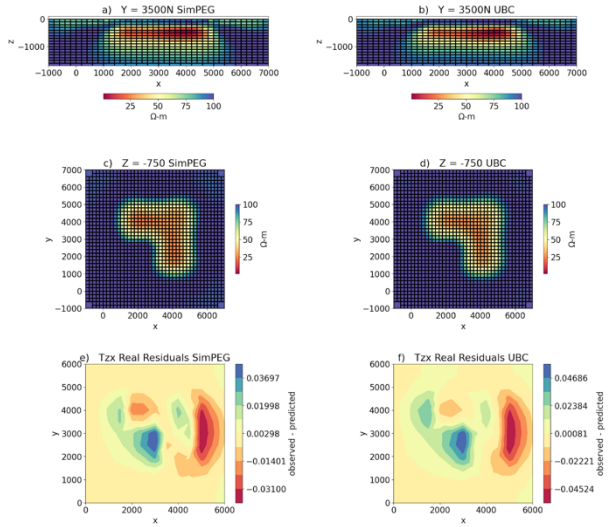

Figure 5. Recovered models obtained by inverting the synthetic ZTEM data using (a, c) SimPEG, and (b, d) the UBC E3DMT Fortran code. Panels (e), (f) show the data residual for the real part of Tzx for the SimPEG and UBC models, respectively.

CONCLUSIONS & NEXT STEPS

Work continues to advance the capabilities and efficiency of the SimPEG EM codes. Future releases of SimPEG will include support for multiple parallelization frameworks (e.g. Dask, MPI, and others) and utilities to support breaking up large simulations so that subsets can be run in parallel. Other areas of research include incorporating petrophysical and geologic information into the inversion as well as the joint inversion of multiple geophysical data sets. In conjunction, we are developing a suite of forward simulation and inversion examples that compare SimPEG results with those from the UBC GIF Fortran codes. These have proven to be valuable for testing and validating both codes and identifying areas where both code bases can be improved. Importantly, we are also taking steps to improve the usability of SimPEG. This includes an emphasis on the development of documentation and tutorials. Our hope is that by making 3D EM codes available and accessible, it will increase the use and value of EM geophysical data in solving geoscientific problems.

ACKNOWLEDGMENTS

The authors are grateful to members of the SimPEG community; those who use, contribute to, and support the project are critical to its success.

REFERENCES

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, 268–282. https://doi.org/10.1093/gji/ggz156

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, 1–19. https://doi.org/10.1016/j.cageo.2017.06.018

Kang, S., Knight, R., Goebel, M., 2022. Improved Imaging of the Large-Scale Structure of a Groundwater System With Airborne Electromagnetic Data. Water Resources Research 58, e2021WR031439. https://doi.org/10.1029/2021WR031439

Oldenburg, D.W., Heagy, L.J., Kang, S., 2021. Geophysical electromagnetics: A retrospective, DISC 2017, and a look forward. The Leading Edge 40, 140–148. https://doi.org/10.1190/tle40020140.1

Rocklin, M., 2015. Dask: Parallel Computation with Blocked algorithms and Task Scheduling. Presented at the Python in Science Conference, Austin, Texas, pp. 126–132. https://doi.org/10.25080/Majora-7b98e3ed-013

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

Werthmüller, D., Rochlitz, R., Castillo-Reyes, O., Heagy, L., 2021. Towards an open-source landscape for 3D CSEM modelling. Geophysical Journal International 1–18. https://doi.org/10.1093/gji/ggab238

Yang, D., Oldenburg, D.W., Haber, E., 2014. 3-D inversion of airborne electromagnetic data parallelized and accelerated by local mesh and adaptive soundings. Geophysical Journal International 196, 1492–1507. https://doi.org/10.1093/gji/ggt465