Introduction

Controlled-source electromagnetic (CSEM) methods have been used since decades in exploration geophysics, with a major hype in the early 2000s in the oil industry. These days it is one of the many established, non-seismic methods in the exploration of resources in the subsurface, not just for hydrocarbons but also for water and geothermal resources, mining, and civil engineering tasks. As a consequence various 3D CSEM codes have been developed since a long time, e.g. Oristaglio and Spies (1999). However, they were often proprietary or only available upon request from the author. Recently it has become possible to install different 3D CSEM modellers with a single command in the Python ecosystem, e.g., SimPEG (Cockett et al., 2015), PETGEM (Castillo-Reyes et al., 2018), custEM (Rochlitz et al., 2019), or emg3d (Werthmüller et al., 2019). This, together with increased computing power, makes it possible for anyone to compute CSEM responses for realistic real-world models. We tried to speed-up the computation of time-domain responses with a frequency-domain code by (a) improving existing methods, and (b) computing in the real-valued Laplace domain instead of the complex-valued frequency domain. While the former idea works fine the latter idea does only work with exact arithmetic results but unfortunately not when numerical approximations must be made.

For the 1D computations and for designing digital linear filters we use `empymod` (Werthmüller, 2017), and for the 3D computation we use the multigrid code `emg3d` (Werthmüller et al., 2019); both codes are released under the Apache License 2.0 and can be found on empymod.github.io. The multigrid solver `emg3d` solves the diffusive approximation of Maxwell's equation given by the second-order differential equation of the electric field,

$$s\mu_0\sigma\mathbf{E} - \nabla\times\mu_\mathrm{r}^{-1}\nabla\times\mathbf{E} = -s\mu_0\mathbf{J}_\mathrm{s}\,, \tag{1}$$

where σ is conductivity (S/m), $s = \mathrm{i}\omega$ is the Laplace parameter and $\omega = 2\pi f$ is angular frequency of frequency f (Hz), $\mu = \mu_0\mu_\mathrm{r}$ is magnetic permeability (H/m), $\mathbf{E}$ is the electric field (V/m), and $\mathbf{J}_\mathrm{s}$ the current source (A/m^2). The diffusive approximation neglects displacement currents by assuming that $\omega\varepsilon \ll \sigma$, where ε is electric permittivity (F/m). The advantage of a matrix-free multigrid solver is that it scales linearly with the number of cells for both CPU and RAM usage (Mulder, 2011). All examples shown here were computed on a laptop with an i7-6600U CPU @ 2.6 GHz (x4) and 16 GB of memory, using Ubuntu 18.04 and Python 3.7.

Time-domain modelling with a frequency-domain code

Computing time-domain data with a frequency-domain code requires the computation of a range of frequencies as input for the Fourier transform; for marine CSEM roughly from 0.001 Hz to 100 Hz (depending on the required offset one can get away with a much smaller range). The required computation grids vary a lot from low to high frequencies. An automatic, adaptive gridding scheme is therefore required, which is usually based on the skin depth $\delta \approx 503.3/\sqrt{f\sigma}$, the depth after which the signal attenuated to $1/e \approx 37\,\%$.

Our scheme is based on the automatic gridding as suggested by Plessix et al. (2007), and the adaptive frequency selection as presented by Mulder et al. (2008). Making two changes to their schemes resulted in substantial, model-dependent speed-up. The major change lies in the automatic gridding. Instead of looking for the optimal parameters (minimum cell width, stretching factor, domain size) for a given number of cells in each direction we are looking for the minimal number of cells required which still comply with the ranges we define for these parameters. Using a Fourier transform that works with logarithmically spaced values, such as digital linear filters (DLF, Ghosh, 1971) or the logarithmic fast Fourier transform (FFTLog, Hamilton, 2000) further improved the overall speed. Carefully selecting the frequency-range showed that 20 frequencies or less are often sufficient for a wide range of offsets.

Figure 1 shows an example, where we reproduced the fullspace model of Mulder et al. (2008) in less than 2 minutes, while their original computation took roughly 3.75 hours. The model is a

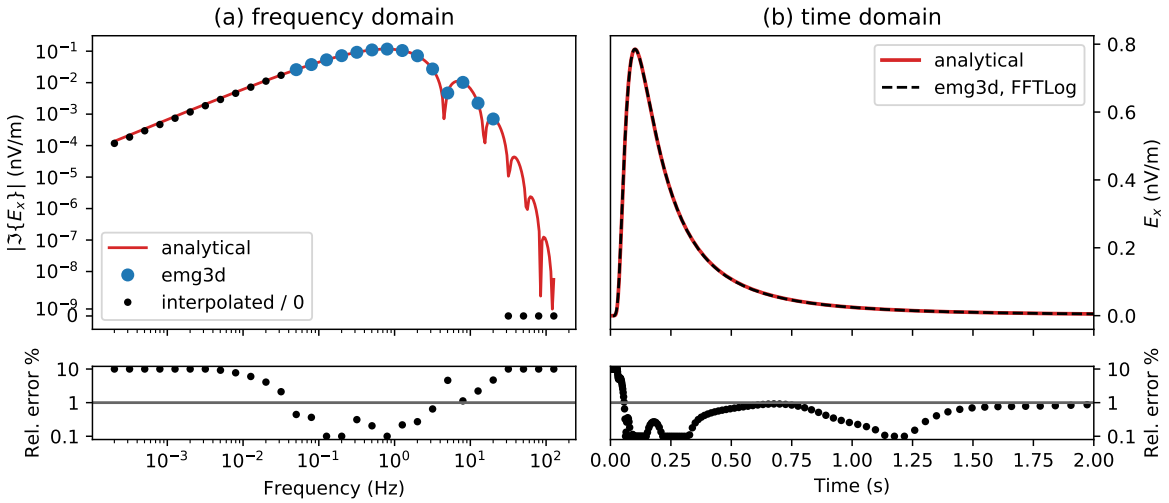

Figure 1: (a) Imaginary frequency-domain response, where the blue circles are computed with `emg3d`, and the black dots are interpolated or set to 0. (b) Corresponding time-domain response. The errors are clipped at $[0.1, 10]$ %

homogeneous fullspace of 1 S/m, an x-directed source at the origin, 1 Hz, and an x-directed inline receiver at an offset of 900 m. We defined the FFTLog with 30 frequencies, $0.0002 - 126$ Hz, but actually required to compute are only 14 frequencies, $0.05 - 20$ Hz (blue circles in the figure). Responses for $f > 20$ Hz were set to zero and responses for $f < 0.05$ Hz were interpolated by assuming that the imaginary part of E_x goes linearly to zero in log-frequency scale.

The time-domain result is shown in Figure 1b, where it can be seen that the relative error is always below 1 %, except for very early times. The frequency selection is important for the speed-up, as our scheme only required 14 frequencies in comparison to the 26 frequencies computed in the original publication. However, the most important change in terms of speed is in the adaptive gridding.

Laplace-domain computation

The complex-valued Laplace parameter $s = \mathrm{i}\omega$ can be replaced with a real-valued s to compute CSEM data in the space-Laplace domain instead of the space-frequency domain. Figure 2 shows a comparison of space-frequency and space-Laplace domain computations. The model is a diffusive halfspace with x-directed source and receiver, where the source is at 500 m depth, and the receiver at $x = y = 1$ km at a depth of 600 m; subsurface conductivity is 1 S/m. There are two main motivations to use Laplace-domain computations: (a) real-valued instead of complex operations, and (b) smoother, non-oscillating responses. The former should result in faster computation times of a single solver iteration, the latter is expected to result in faster convergence.

We implemented the possibility of space-Laplace domain computations into both `empymod` (v1.9.0) and `emg3d` (v0.8.0). To test if Laplace-domain computations are faster than frequency-domain computations we run a complex, isotropic model with roughly 2.1 million cells in both domains, as shown in Figure 3. We chose a frequency of 1 Hz, which yields in the frequency domain $\mathrm{i}\omega = \mathrm{i}2\pi\mathrm{f}$ and in the Laplace domain we took the equivalent of $s = 2\pi f$. An average iteration in the space-frequency domain took 9.9 seconds, and 16 multigrid cycles were required to reach the desired tolerance. In the space-Laplace domain it took 8.0 seconds per iteration, and 14 multigrid cycles were required. So both expectations, faster computation and faster convergence, could be confirmed. The overall speed-up in this case is a factor of 0.71.

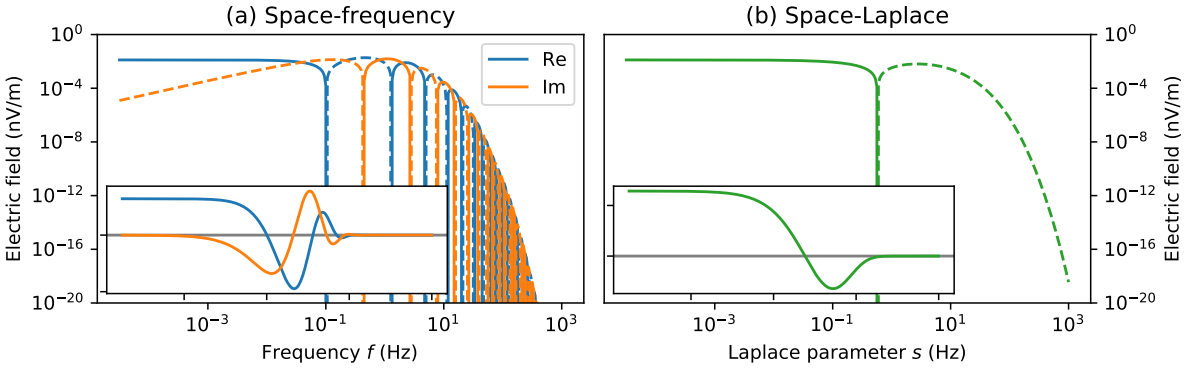

Figure 2: Comparison of a CSEM response in (a) the space-frequency and in (b) the space-Laplace domain; the inset figures show the same but with a linear scale for the y-axis. The motivation for Laplace-domain computations is real-valued operations and no oscillating behaviour in comparison to frequency-domain responses.

There is no analytical transformation from the Laplace domain to the time domain. However, we were able to derive a digital linear filter for the Laplace-to-time transformation using the `fdesign`-routine of `empymod` and the analytical Laplace- and time-domain functions as input values. The filter is shown in Figure 4, and a show-case of the filter in Figure 5a. The model is a diffusive, x-directed impulse response, inline, of a halfspace of 1 S/m conductivity and vertical transverse conductivity of $\sqrt{2}$. The source is located at the origin at a depth of 150 m, the receiver is at an offset of 2 km at a depth of 200 m. The time-domain result obtained through Laplace-domain computation followed by a DLF yields the time-domain response accurately.

Given these encouraging initial results we tried to apply it to 3D modelling results, however, without success. We therefore returned again to 1D modelling to analyze the problem: Figure 5b shows the exactly same data as Figure 5a. The only difference is that we multiplied the amplitude of $s = 7.043$ by 1.00001. This tiny error causes the whole DLF to fail.

Conclusions

We have shown that computing 3D time-domain CSEM data with computation in the frequency domain is very competitive if (a) the frequency-dependent adaptive gridding is optimized to use as few cells as possible and (b) a logarithmically-scaled Fourier transform is used such as DLF or FFTLog, with careful frequency selection. Computation in the Laplace domain results in faster

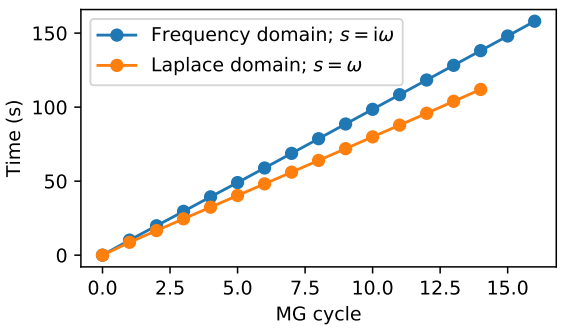

Figure 3: Runtime comparison between space-frequency and space-Laplace domain computations. The space-Laplace domain computation is faster per cycle and requires less cycles.

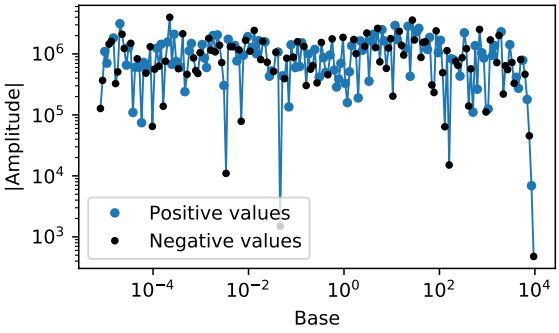

Figure 4: The derived 201 pt Laplace filter using the `fdesign`-routine of `empymod` with the analytical Laplace- and time-domain functions as input values.

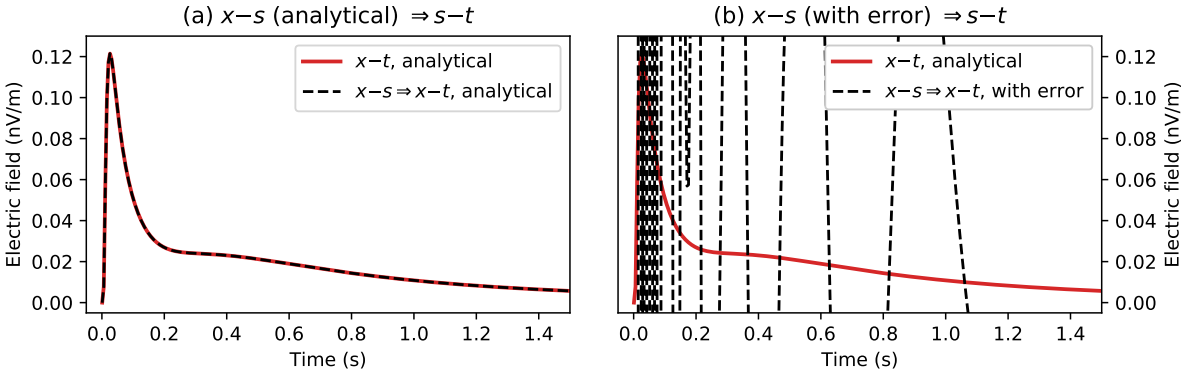

Figure 5: (a) Laplace-to-time DLF works fine for analytical responses. (b) As soon as the Laplace-domain has the slightest error it fails. The red curve, the analytical time-domain result, is the same in both plots.

computation time (real vs. complex operations) and faster convergence (smoother behaviour) when compared with computation in the frequency domain. However, using Laplace-domain computations to obtain time-domain CSEM responses fails, as the Laplace-DLF requires very precise and complete input data, unlike the sine- or cosine-DLF for the Fourier transform.

Acknowledgements

This research was conducted within the Gitaro.JIM project funded through MarTERA, a EU Horizon 2020 research and innovation programme (No 728053); martera.eu.

References

Castillo-Reyes, O., de la Puente, J. and Cela, J.M. [2018] PETGEM: A parallel code for 3D CSEM forward modeling using edge finite elements. *Computers & Geosciences*, **119**, 123–136. DOI: 10.1016/j.cageo.2018.07.005.

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

Ghosh, D.P. [1971] The application of linear filter theory to the direct interpretation of geoelectrical resistivity sounding measurements. **19**(2), 192–217. DOI: 10.1111/j.1365-2478.1971.tb00593.x.

Hamilton, A.J.S. [2000] Uncorrelated modes of the non-linear power spectrum. *Monthly Notices of the Royal Astronomical Society*, **312**(2), 257–284. DOI: 10.1046/j.1365-8711.2000.03071.x.

Mulder, W.A. [2011] *Numerical Methods, Multigrid*. Springer Netherlands, Dordrecht, 895–900. DOI: 10.1007/978-90-481-8702-7_153.

Mulder, W.A., Wirianto, M. and Slob, E. [2008] Time-domain modeling of electromagnetic diffusion with a frequency-domain code. *Geophysics*, **73**(1), F1–F8. DOI: 10.1190/1.2799093.

Oristaglio, M. and Spies, B. (Eds.) [1999] *Three-Dimensional Electromagnetics*. No. 7 in Geophysical Developments. SEG. DOI: 10.1190/1.9781560802154.

Plessix, R.E., Darnet, M. and Mulder, W.A. [2007] An approach for 3D multisource, multifrequency CSEM modeling. *Geophysics*, **72**(5), SM177–SM184. DOI: 10.1190/1.2744234.

Rochlitz, R., Skibbe, N. and Günther, T. [2019] custEM: customizable finite element simulation of complex controlled-source electromagnetic data. *Geophysics*, **84**(2), F17–F33. DOI: 10.1190/geo2018-0208.1.

Werthmüller, D. [2017] An open-source full 3D electromagnetic modeler for 1D VTI media in Python: empymod. *Geophysics*, **82**(6), WB9–WB19. DOI: 10.1190/geo2016-0626.1.

Werthmüller, D., Mulder, W.A. and Slob, E.C. [2019] emg3d: A multigrid solver for 3D electromagnetic diffusion. *Journal of Open Source Software*, **4**(39), 1463. DOI: 10.21105/joss.01463.