An open-source *1D* electromagnetic modeler in Python: empymod

Dieter Werthmüller[1]

Instituto Mexicano del Petróleo, Eje Central Lázaro Cárdenas Norte 152, Col. San Bartolo Atepehuacan C.P. 07730, Ciudad de México, México. E-mail: Dieter@Werthmuller.org.

(April 3, 2017)

GEO-2016-0626

Running head: **Open-source *1D* EM modeler in Python**

ABSTRACT

~~The free software empymod combines two earlier presented algorithms in this journal, creating a new code written in Python that is faster and leaner. The main objective is to present a three-dimensional layered-earth electromagnetic modeler with vertical transverse isotropy in an easy accessible programming language, both in terms of costs and learning curve, and in a collaborative way. The code is hosted in a web-based Git repository under a lax permissive license, allowing anyone to use it, even for commercial purposes, as well as to contribute to its development. Comparisons show that this code is as precise and faster than the codes it is based upon, thanks to the use of different Hankel transforms and the throughout calculation. This code might certainly be useful for professionals in the electromagnetic area, but I specifically hope this code to be useful for educational purposes.~~

The presented Python-code empymod computes the three-dimensional electromagnetic field in a layered-earth with vertical transverse isotropy by combining and extending two earlier presented algorithms in this journal. The bottleneck in frequency- and time-domain cal-

culations of electromagnetic responses derived in the wavenumber-frequency domain is the transformations from wavenumber to space domain and from frequency to time domain, the so-called Hankel and Fourier transforms. Three different Hankel transform methods (quadrature, quadrature-with-extrapolation, and filters) and four different Fourier transform methods (FFT, FFTLog, quadrature-with-extrapolation, and filters) are included in empymod, which allows to compare these different methods in terms of speed and precision. The best transform in terms of speed and precision depends on the modeled frequencies: available filters, for instance, are very fast and precise for frequencies in the range of controlled-source electromagnetic data, but fail in the frequency range of ground penetrating radar. Conventional quadrature, on the other hand, is in comparison very slow but can model any frequency. Examples comparing empymod with analytical solutions and with existing electromagnetic modelers illustrate the capabilities of empymod.

INTRODUCTION

The potential of electromagnetic methods for the detection of hydrocarbon reservoirs is known for some decades, see for instance Nekut and Spies (1989) or Chave et al. (1991). More recent, good overviews of the methodology and its applications are give by Edwards (2005) and Ziolkowski and Wright (2012). Whereas in the early days everything happened in idealized, isotropic one dimensional (1D) earth models, it is generally agreed that the often complex geology of hydrocarbon reservoirs requires two dimensional (2D) and three dimensional (3D), anisotropic forward modelers. However, 1D models are still very important. ~~, not last because they are very fast. Many problems can be simplified to and henceforth solved with 1D models. More importantly, 1D allows~~ *Besides that their calculation is very fast they allow* to study single, isolated effects to the electromagnetic field, which is a crucial foundation in understanding the electromagnetic field behaviour and a necessity for understanding the phenomena at higher dimensions. 1D models are furthermore often used in inversion routines of higher dimensions, for instance to generate a starting model or *to calculate the primary fields for 2D/3D modeling and inversion routines.* ~~embed a 3D body in a 1D background.~~

Solutions for the electromagnetic fields in a layered-earth model have been solved and published extensively using different approaches. The importance and widespread use of 1D models is shown in the continuous stream of publications in this area, even in recent years. Løseth and Ursin (2007) solved the problem with the scattering matrix formulation for a 1D earth with general anisotropy, spanning the frequency range from controlled-source electromagnetics (CSEM) to ground-penetrating radar (GPR). Chave (2009) presented a solution in terms of independent and unique transverse electric (TE) and tangential magnetic (TM)

modes for the electrical isotropic case using the diffusive approximation (without displacement currents, valid for low frequencies such as in CSEM). Key (2009) demonstrated why 1D models still matter by testing, for instance, the benefit of additional frequencies in an inversion routine~~. Key~~*; he* follows and extends the magnetic vector potential approach by Wait (1982) for forward modeling, using the isotropic, diffusive low-frequency approach. Hunziker et al. (2015) obtained the electromagnetic field in a layered earth with vertical transverse isotropy (VTI) by solving two equivalent scalar equations with a scalar global reflection coefficient. All of the four citations regarding 1D solutions have quite extensive reference lists about the history of 1D forward modeling, the last one featuring an interesting review of the history of 1D electromagnetic derivations spanning almost 200 years.

A crucial as well as very interesting part of electromagnetic modeling is, once one moves from the theoretical derivation to the numerical implementation, the Hankel transform involved in the transformation from the wavenumber-frequency domain to the space-frequency domain. *Because what is common in all these approaches is that they derive solutions in the wavenumber-frequency domain. This in turn requires a Hankel transform, which is a numerically expensive, infinite integral containing oscillating, slowly decaying Bessel functions (the Hankel transform is also known as the Fourier-Bessel transform).* The use of digital filters is quite common in geophysics, known as the fast Hankel transform method (FHT), as introduced by Ghosh (1971), and popularized by Anderson (1975, 1979, 1982) thanks to his freely available Fortran routines. Naturally, standard quadrature can be used as well for the Hankel transform, see for instance Chave (1983), and hybrid routines using both methods were published by Anderson (1984, 1989). The topic got picked up again recently with new filters being published by Kong (2007) and Key (2009, 2012)

In terms of freely available and open-source code there are a few examples. Key (2009)

published his forward modeling and inversion code Dipole1D, written in Fortran. The dipole can be placed anywhere in the stack of layers and can have arbitrary orientation and dip. Dipole1D computes the electric and magnetic fields to an electric source, using the FHT method. Key (2012) published additional code with his introduction of the quadrature-with-extrapolation method (QWE) and its comparison to the FHT method, for which he translated some of the Dipole1D-Fortran code to Matlab. Hunziker et al. (2015) published their code EMmod (Fortran and C), in which the source and receiver can also be placed anywhere in the stack of layers. They use a 61 pt Gauss-Kronrod quadrature for the Hankel transform.

With empymod I present a 1D forward modeling code that is based on Hunziker et al. (2015) for the wavenumber-frequency domain calculation, and *mainly* on Key (2012) for the Hankel and Fourier transforms. ~~As I will therefore refer to these publications quite a lot, I use the name Hun15 to denote Hunziker et al. (2015), and the names Key09 and Key12 to denote Key (2009, 2012). To denote the FHT filters as published by Key09 and Key12 I will add the corresponding filter-size, e.g. Key09-401.~~ In addition *to QWE and FHT*, empymod includes *an adaptive quadrature (QUAD) for the Hankel transform, and the standard fast Fourier transform FFT as well as* the logarithmic fast Fourier transform FFTLog of Hamilton (2000) ~~as a third~~ *for the* Fourier transform ~~possibility to FHT and QWE for the frequency-to-time transformation~~.

To my knowledge, empymod is the first code that is freely available which calculates the full wavefield for a layered-earth model with vertical isotropy and has ~~both quadrature and filters~~ *different types of Hankel transforms (QWE, QUAD, FHT) and various methods for the Fourier transform (FFTLog, sine/cosine-filters, QWE, FFT)*~~built in~~. ~~This makes~~ *An interesting use-case for* empymod ~~ideal not just for~~ *is therefore* comparison

studies of the ~~two~~ *different* methods, ~~but also to build~~ *up to building* hybrid inversion schemes. ~~Further advantages of empymod are:~~ *The code is published under the lax permissive* Apache Version 2.0 license*, which makes it available to everyonefor free, even for commercial purposes. It therefore might prove to be valuable for students and professionals alike. It is written in Python, a modern, cross-platform, free and open-source programming language. With its scientific libraries, mainly the numeric and scientific modules* `NumPy` *and* `SciPy`*, it creates an extremely powerful numerical calculation stack: Even though it is an interpreted language it can be very fast, as* `NumPy/SciPy` *use under the hood routines in Fortran or in C (using for instance the BLAS/Lapack libraries) for the linear algebra computations, and Python is merely the glue. The comparisons show that empymod is as fast as the existing codes. The code is vectorized (avoiding loops) as much as possible, which improves computation. The most time-consuming calculations have furthermore a flag to run parallelized. However, on a larger scale such as an inversion routine one would probably want to parallelize the kernel calls instead of the kernel itself. The code is hosted on GitHub, which makes it easy for anyone to improve and contribute to the code:* `https://github.com/prisae/empymod`*. The documentation is hosted on* `https://empymod.readthedocs.io`*.* Werthmüller (2017)*gives a tutorial-style introduction to EM modeling using empymod.*

~~**(1) Python:** Python is a modern, cross-platform, free and open-source programming language. With its scientific libraries, mainly the numeric and scientific modules `NumPy` and `SciPy`, it creates an extremely powerful numerical calculation stack.~~

~~**(2) Libre (free and open-source):** The code is published under the lax permissive *Apache Version 2.0 license*, which makes it available to everyone, even for commercial purposes. It therefore might prove to be valuable for students and professionals alike. Not~~

only the code is free, but also the underlying platform (Python), contrary to, for instance, Matlab. As such it is ideal for reproducible research.

(3) Lean: The codebase is very lean. It is built from scratch, and it therefore does not suffer the problems that sometimes come with the organic growth of a codebase. The code follows as much as possible the *DRY* coding paradigm, *Don't Repeat Yourself*.

(4) Fast: The code is vectorized *(avoiding loops)* as much as possible, which improves computation. The most time-consuming calculations have furthermore a flag to run parallelized. However, on a larger scale such as an inversion routine one would probably want to parallelize the kernel calls instead of the kernel itself. Even though Python is an interpreted language it can be very fast, as *under the hood routines* the underlying routines are in Fortran or in C (using for instance the BLAS/Lapack libraries) *are used for the linear algebra computations*, and Python is merely the glue. The comparisons show that empymod is as fast as the existing codes.

(5) Community: The code is hosted on GitHub, which makes it easy for anyone to improve and contribute to the code, and will allow empymod to grow: `https://github.com/prisae/empymod`.

(6) Well documented: The code is extensively documented, and large parts of it are extracted using automated tools (Sphinx) to create a manual: `https://empymod.readthedocs.io`.

These points make this a worthwhile extension to the existing codes from Hun15 and Key12. The existing codes are written in Fortran, which requires much more time than Python to get started for students or to develop for anyone, or Matlab, which is a proprietary language. And the existing codes are in static repositories where one can only download

~~the code, which makes interaction, contribution, or bug filing more difficult.~~

After introducing the code in the first part I will present some comparisons by reproducing results from the publications this code is based upon: First ~~a~~ *some* comparison*s* to the analytical half-space solution, followed by a comparison to EMmod by ~~Hun15~~Hunziker et al. (2015), and finally a comparison to some results presented by ~~Key12~~Key (2012). *Last is a time-domain example in which I compare the four different Fourier transforms.*

ABOUT THE CODE

The code consists of 5 files; these are 3 core modules plus *utils*, which contains input checks and other utilities, and *filters*, containing the FHT filter coefficients. The three core routines are: (1) *kernel*, where the wavenumber-domain calculation is carried out; (2) *transform*, where the Hankel and Fourier transforms are computed; and (3) *model*, which contains the actual modeling routines for end users. *The main modeling routine is* bipole*, which can calculate frequency- and time-domain responses for arbitrary oriented, electric or magnetic bipole sources and receivers of finite length.* ~~As of now there are two main modeling routines implemented, *frequency* and *time*, with which one can calculate the frequency- and time-domain responses for electric or magnetic point sources and receivers, directed along the three principal axis x, y, and z. More modeling routines can easily be added to *model*, such as finite bipoles or arbitrary source and receiver directions, as they do not affect the core of the calculation, hence not *kernel* nor *transform*.~~

The wavenumber-domain calculation in *kernel* follows ~~Hun15~~Hunziker et al. (2015), and calculates as such the complete wavefield for a layered VTI model. ~~, where the~~ *The* code makes no assumptions about the model~~.~~*:* Source and receiver can be placed anywhere

in the model~~.~~, *including first and last layer, and you have to define if the first layer is air or not. Bipoles can cross layer boundaries.* Depths, frequencies, and source-receiver configuration have to be defined, and each layer is characterized with its horizontal resistivity ρ_h, its electrical anisotropy λ, where $\lambda = \sqrt{\rho_\mathrm{v}/\rho_\mathrm{h}}$, its horizontal and vertical magnetic permeabilities μ_h and μ_v, and the horizontal and vertical electric permittivities ϵ_h and ϵ_v. The main differences between EMmod and empymod *is*~~are the programming language and~~ the Hankel transform, and a few additional fundamental differences: EMmod uses 2nd order Bessel functions of the first kind J_2. Published FHT filters generally provide coefficients for 0th and 1st order Bessel functions J_0, J_1 only. Therefore, the recurrence relation

$$J_2(kr) = \frac{2}{kr} J_1(kr) - J_0(kr) \tag{1}$$

is used, where k and r are the wavenumber and space-domain parameters, respectively. *Other differences are that empymod includes Fourier transforms, hence time-domain calculation, and can model arbitrary rotated, finite bipoles.* ~~Another difference is vectorization: EMmod carries out the calculation in the wavenumber-domain by looping over frequencies, offsets, and wavenumbers. This is carried out in a single calculation without looping in empymod. Another difference is that empymod is much leaner than EMmod. EMmod grew organically, as confirmed by the author, adding more and more features, where empymod was designed with all 36 source-receiver configurations from the beginning. As an example, the full-space solution in empymod is one function (107 lines of code) for all source-receiver configurations, whereas in EMmod it is split into no less than 16 functions (364 lines of code). This should help to maintain the code easier, and it should also be easier for new contributors to read into the code.~~

The *QWE and filter* Hankel and Fourier transforms in *transform* follow ~~Key12~~Key

(2012), with a few changes. The most important ones regarding speed is vectorization, *hence avoiding loops,* and a splined version for the filter method, in addition to the traditional lagged version: The lagged version samples the wavenumber from the minimum to the maximum required wavenumber given the required offsets and the chosen filter base with the spacing as defined by the filter. It subsequently carries out the Hankel transform, and interpolates for the required offsets in the space domain. The splined version, on the other hand, uses a user-specified number of values per decade from the minimum to the maximum wavenumber. It then interpolates for the required wavenumber values, and does the Hankel transform afterwards. The lagged version is *very* fast. However, with the splined version a speed-up can be achieved in comparison with the original FHT at higher precision if compared with the lagged version. All filters published *in the source codes of* ~~by~~ ~~Key09 and Key12~~Key (2009, 2012) are included in empymod, which includes Key's own filters as well as the filters by Anderson (1982) and by Kong (2007). *(I refer throughout to paper to a specific filter with author, year, and filter-size, for instance Key09-201.)*

The most import part of ~~Key12~~*GEO.12.Key* is the introduction of a new quadrature algorithm to geophysics, named quadrature-with-extrapolation (QWE). QWE is a fast quadrature method using the Shanks transformation (Shanks, 1955) computed with Wynn's epsilon algorithm (Wynn, 1956). The advantage of quadrature over filters is the ability to estimate the error. QWE continues until the absolute error, estimated by the difference of subsequent iterations n, satisfies the inequality

$$|S^*_n - S^*_{n-1}| \leq \varepsilon_r |S^*_n| + \varepsilon_a \ , \tag{2}$$

where S^* is the extrapolated result, and $\varepsilon_r, \varepsilon_a$ are the relative and absolute tolerance, respectively.

Whereas for the Hankel transform the ~~two~~*three* methods QWE, ~~and~~ FHT*, and QUAD, a standard adaptive quadrature using QAGSE from the Fortran QUADPACK library,* are implemented, for the Fourier transform there are ~~three~~*four* methods implemented: QWE, FHT with the sine and cosine filters, *the standard FFT,* and FFTLog (Hamilton, 2000). The logarithmic fast Fourier transform FFTLog is ideal for this operation, as generally a wide range of frequencies is required to go from *the* frequency to *the* time domain. The FFTLog can yield faster results than the QWE method and more precise than the FHT results, specifically for land impulse responses at early times.

~~In addition to~~ *Calculation time can be reduced by using* the splined version of the QWE and the splined and lagged versions of the FHT~~, a~~*. In addition, a*ll time-consuming calculations are set up to be able to run in parallel, with the help of the Python-module numexpr. This can significantly speed up calculations if you run big models with many layers and for many offsets and frequencies. However, if you include empymod in an inversion scheme then it might be better to parallelize the calls to empymod, instead of empymod itself.

It is important to note that calculations in the wavenumber domain depend *only* on offset $r = \sqrt{x^2 + y^2}$; the scaling factor, which depends on the angle $\varphi = \text{atan2}(\text{y}, \text{x})$, is multiplied afterwards. In order to calculate for instance a circle around the source, the kernel has to be called only once, and subsequently scaled by the angle-dependent factor for each source-receiver pair. This makes the splined version very powerful, as it can be used for irregularly distributed data*as long as all receiver depths are the same.*

~~The code distributed on GitHub contains Jupyter Notebooks to reproduce the results in this article, as well as some basic tests and benchmarks, and the LaTeX-source of the article itself.~~

The run-time comparisons tests were run on a Lenovo ThinkCentre running Ubuntu 16.04 64-bit, with 8 GB of memory and an Intel Core i7-4770 CPU @ 3.40GHz x 8 ~~(4 cores with hyper-threading)~~. Comparing run times is always a difficult task, and between different languages even more, here between Python *(empymod)*, Fortran *(EMmod, Dipole1D)*, and Matlab *(scripts from Key (2012))*. However, they do serve as comparison. I used the Matlab-`timing` and Python-`timeit` functions, which behave very similar. It is probably expected today, but still worth mentioning, that the calculation is carried out in double precision.

COMPARISON TO ANALYTICAL HALF-SPACE SOLUTION

Slob et al. (2010) published analytical frequency- and time-domain solutions for the diffusive electric field in a VTI half-space*, where source and receiver can be placed anywhere in the half-space*. As an example, I use the same model as ~~Hun15~~Hunziker et al. (2015) in their Figures 1 and 2: A half-space with horizontal resistivity $\rho_\mathrm{h} = 1/3\,\Omega\,\mathrm{m}$, anisotropy $\lambda = \sqrt{10}$, and for frequency $f = 0.5\,\mathrm{Hz}$. The source is located horizontally at the origin *($x_\mathrm{s} = y_\mathrm{s} = 0\ m$)* at a depth of 150 m, and the depth of the receivers is 200 m. The field is calculated on a regular grid with a spacing of 10 m. Figure 1 shows the analytical amplitude and phase results for the first quadrant (the other quadrants are simply symmetric copies of it). The figure shows the result for x-directed electric source and receiver dipoles (G^{ee}_{xx}), other configurations yield similar results (the choice is based on the insight that this is the most interesting of all electric source to electric receiver fields, as can be seen in ~~Hun15~~Hunziker et al. (2015) Figures 1 and 2).

The error of the amplitude is shown in Figure 2 for different settings regarding the Hankel transform: (a) for a 51 pt QWE with relative tolerance ε_r and absolute tolerance

ε_a of 10^{-12} and 10^{-30}, respectively; (b) for a 21 pt QWE with $\varepsilon_r = 10^{-8}$ and $\varepsilon_a = 10^{-30}$; (c) for a 15 pt QWE with $\varepsilon_r = 10^{-8}$ and $\varepsilon_a = 10^{-18}$; (d) using the standard FHT method with the filter Key09-4201; (e) using the splined FHT of the same filter with 40 points per decade; and (f) using the lagged FHT of the same filter. The results from the high precision QWE (a) and the standard FHT (d) are almost identical, even though they use completely different Hankel transforms. ~~This can be seen better in Figure ??, which shows a cross-section through subplots (a), (c), (d), and (f) of Figure 2 at a crossline offset of 4 km.~~ The error is, in this case, not due to the method used for the Hankel transform, but rather to the ~~limitations in computation: either because we reach the numerical noise level, or because of~~ distinct differences between the analytical solution, which uses the diffusive approximation, and the numerical code, which calculates the complete field. If the QWE is calculated for lower tolerance levels, as shown in (b) and (c), or the FHT is used with interpolation as in (e) and (f), artefacts seem to arise from the Hankel transform and the interpolation. The error of the phase is shown in Figure 3, with the same conclusion.

Figure 4 shows a comparison of the error for all nine included FHT-filters, for the amplitude (phase is not shown, but is very similar). They are all very accurate, and by choosing an optimal filter one has to decide between accuracy and speed, as shorter filters are faster. However, we have to keep in mind that the accuracy of the filters depends on the model, the offsets, and the frequencies. If we do not have an analytical solution, as in this case, we cannot check the accuracy.

COMPARISON TO HUNZIKER ET AL. (2015)

~~Hun15~~Hunziker et al. (2015) derive the electromagnetic fields in astoundingly simple equations for all 36 possible source-receiver combinations (electric and magnetic sources and

receivers in three directions x, y, z) by finding the solution for the vertical electric field and then applying the duality principle and reciprocity to derive all components. The corresponding code EMmod is published on the SEG website, and in Hunziker et al. (2016) they published with iEMmod an inversion routine for it.

EMmod is written in Fortran and C. On execution, all parameters are provided in an input file, and the resulting responses are written to an output file. The calculation is carried out for a regular grid in the space domain, with at least two points in each direction. The code calculates ~~in loops~~ the solution for all wavenumbers for the first quadrant, carries out the wavenumber-to-space transformation, and then copies the result for the other four quadrants, writing everything to the output file. This approach works very well and even for millions of cells. However, the usage is probably more academic. When it comes to actual measurements one deals with dozens to at most hundreds of offsets, and usually on an irregular grid. Dipole1D and empymod work more along the practical approach.

Figure 5 shows the error of amplitude and phase for EMmod. On the left side with a colorscale from 10^0 to 10^2 % as used in ~~Hun15~~Hunziker et al. (2015), on the right side with a colorscale from 10^{-8} to 10^0 % as used in Figures 2 and 3. Note that the error calculation in ~~Hun15~~Hunziker et al. (2015) was done slightly different. Here I show the relative error between the analytical result and the result from EMmod, where ~~Hun15~~Hunziker et al. (2015) shows the relative error between $\log_{10}$(analytical result) and $\log_{10}$(result from EMmod). Figure 5 shows a few interesting points. The responses from EMmod in this example have significantly less precision than the results from empymod. This does not mean in any way that EMmod is less precise, as EMmod can be adjusted to yield much preciser results. However, this would significantly increase the run time, so it has to be kept in mind for the run time comparison. The important point is that EMmod does *always* do an interpolation

in the space-frequency domain, by default, a linear interpolation. No interpolation is used in empymod *with QWE or FHT as Hankel transform*, unless one specifies it to speed up the calculations (splined QWE or splined and lagged FHT options), which will therefore yield preciser results. This can be seen very nicely in the results. Figures 5 (b) and (d) show white circles where the result is very precise. These are the offsets close to those where the fields were actually calculated. The black bands in-between are the areas where the result was interpolated. It also shows that the spacing between calculated offsets increases with increasing offsets. The error patterns from empymod in Figures 2 and 3 show *in (a) and (d)* ~~generally~~ a different pattern, displaying ~~where the code hits the numerical accuracy or~~ differences between the analytical, diffusive solution and the numerical, full wavefield solution. It is only for the splined and lagged versions that one sees the same, circular pattern*s* appearing, which are due to interpolation.

Table 1 shows run times for these models. A few notes worth mentioning: empymod carries out a lot of input checks, as it is quite forgiving in what you input. EMmod, on the other hand, reads the input file and writes the result to an output file, and copies the first quadrant to the other three. Both have therefore some overhead, and the comparison is not strictly 1:1. For the comparison the same model was used as shown above, on a regular grid with a spacing of 100 m. On the test-machine the standard ~~, vectorized,~~ QWE and FHT empymod would run into memory issues for denser spacing, hence arrays of more than some 10 000s of offsets. The splined and lagged versions of EMmod could handle it, however. QWE becomes faster for decreasing precision, not surprisingly, and the splined and lagged version of FHT are faster than the standard version. The lagged FHT is the fastest by quite a margin, and about 1/3 of that time is from the input checks, so it takes only ~~about 4 ms~~*a few milliseconds* to calculate the 11 025 offsets. What is interesting

here is that the lagged FHT is still more precise than EMmod with the given settings, which means a speed-up of a factor 1000 for the same result.

Many 1D CSEM codes use the diffusive approximation that is valid for low frequencies. The appealing part of the derivation of ~~Hun15~~Hunziker et al. (2015) is that it models the complete wavefield, hence it is *also* valid for high frequencies and therefore wave-phenomena. As an extreme case, ~~Hun15~~Hunziker et al. (2015) show a 1D example for ground-penetrating radar with a center frequency of 250 MHz. To do so, *they calculate* 2048 frequencies in the range of 1 MHz to 2048 MHz ~~are calculated~~ for the Fast Fourier transform from frequency to time domain. *Here I only calculated the FFT with 850 frequencies from 1 MHz to 850 MHz and then zero-padded up to 2048 MHz, for EMmod and empymod; tests have shown that this is sufficient.* Figure 6 shows *the model with the survey setup and* the *four*~~three~~ results for EMmod, *QUAD for relative and absolute tolerance of* 10^{-12} *and* 10^{-20}*, respectively,* empymod with a 21 pt QWE and relative and absolute tolerance of 10^{-10} and 10^{-18}, respectively, and empymod with FHT with the filter Key09-401, together with the analytical solution for the arrival of the direct wave (red), the wave refracted at the surface (~~cyan~~*yellow*), and the wave reflected at the subsurface interface (~~magenta~~*green*). ~~For the results with empymod I used the regular FFT as Hun15, not FFTLog.~~

~~EMmod does clearly do the best job. QWE has troubles at very short offsets (< 0.1 m), and a *noisy triangle* expanding from the origin. This triangle could be narrowed by increasing the precision of the QWE, at the cost of computation time. However, I was not able to completely get rid of it with QWE.~~ *EMmod and the QUAD and QWE transforms of empymod yield pretty much the same result. It is only at later times that some noise can be seen, different in each method.* FHT shows the first arrivals well, however, afterwards the result becomes extremely noisy. Interesting are the calculation times for these three

results. EMmod took ~~more than 32~~*roughly 9.5* hours to calculate, *empymod with QUAD about 8.8 hours,* empymod with QWE about ~~34 minutes~~*4.4 hours*, and empymod with FHT ~~a bit over 4~~*under a* minute~~s~~. *Note that QWE has internally a check: If the kernel is too steep within given intervals, it uses QUAD instead, as QWE cannot handle very steep functions. QWE uses in about 1/3 of the calculation of this example QUAD.* Surprising is how good the FHT result is, given that the filter was designed for CSEM data, hence frequencies 6 to 9 orders of magnitudes smaller *and much bigger offsets*. ~~One could implement the Gauss-Kronrod quadrature into empymod, and see how this would compare to EMmod for GPR data in terms of speed and precision. But, i~~*I*n any case, EMmod/empymod are not optimized for nor intended to model GPR data, and calculations in time-domain will be much faster and more precise. It is nevertheless an interesting proof of concept and shows that EMmod/empymod model the entire EM field. *This leaves the question if it would be possible to create a filter that works for the frequencies and the offsets required for GPR calculations.*

COMPARISON TO KEY (2009, 2012)

~~Key12~~*GEO.12.Key* not only introduces QWE to geophysics, but also compares QWE to FHT for various filters, some of which were specifically created for the comparison. In order to calculate the models he translates parts of Dipole1D (~~Key09~~Key (2009)) from Fortran to Matlab. The models are therefore isotropic and use the diffusive approximation. He summarizes succinct and clear the FHT method, ~~and~~ outlines the QWE method, and made the codes available from the SEG website.

The inclusion of the filter method FHT and the quadrature method QWE in empymod follows ~~Key12~~Key (2012), with one important exception: vectorization. The Matlab code of

~~Key12~~Key (2012) loops over frequencies, offsets, and wavenumbers; the code was developed to compare the different hankel transforms, not with speed in mind. In empymod both methods are vectorized, hence various ~~frequencies and offsets~~ *offsets, and in the case of FHT also various frequencies,* can be calculated for all required wavenumbers in one calculation, which changes the conclusion drawn in ~~Key12~~Key (2012) comparing the speed of QWE and FHT. The vectorized approach is much faster but is limited by the memory of the computer. If the model becomes too big, which depends on numbers of layers, frequencies, offsets, and wavenumbers, the calculation has to be carried out by looping over frequencies or offsets or both; the code never loops over wavenumbers.

In ~~Key12~~Key (2012), the standard QWE is always faster than the standard FHT. In the vectorized version of empymod, the standard QWE *can* be faster than the standard FHT if the model is big and many offsets are required, but often it is slower. Furthermore, the lagged FHT is generally much faster than the QWE method, splined or not.

Table 2 lists the run times for exactly the same models as ~~Key12~~Key (2012) compared in his Table 1~~.~~*: 1 km water layer, followed by a 1 km overburden of 1 Ω m, 100 m reservoir of 100 Ω m, and lastly 1 or 96 layers of underburden with a resistivity of 1 Ω m; frequency is 1 Hz, source depth 990 m, and receivers are on the sea-bottom; there are 1, 5, 21, 81, or 321 offsets between 0.5 and 20 km.* In this comparison, a 9 pt QWE is compared to a 201 pt (Key12-201) and a 801 pt filter (~~Key12~~*Anderson82*-801), where the absolute and relative tolerance are set so that the QWE achieves a similar error-level as the filters ($\varepsilon_r = 10^{-6}$ for the standard QWE and 10^{-2} for the splined version, $\varepsilon_a = 10^{-24}$ in both cases). In order to make a fair comparison I re-run the script from ~~Key12~~Key (2012), and the run times on the test-machine are roughly twice as fast as in the original paper. Next to it are the results from empymod. It can be seen from the results that empymod is significantly faster.

Important to note is, however, not the absolute speed of empymod, but the difference between the various Hankel transform methods. The standard QWE method, for a similar error level as FHT, is faster than the standard FHT mode only for many offsets. More importantly, the lagged FHT is generally much faster than any QWE. QWE is very useful to check the error level of a result. If many kernel evaluations have to be carried out, and speed is of importance, the lagged FHT is the preferred method. As ~~Key12~~Key (2012) loops over each wavenumber, the difference between the vectorized and non-vectorized version can be best seen in the long 801 pt filter.

These results regarding speed of calculation do not change the fact that the advantage of the QWE method is to have an estimate of the error. However, the filters designed for CSEM problems seem to be very good at solving them, as can be seen in the low error levels in Figures 2 and 3 or in ~~Key12~~Key (2012).

Table 3 lists the run times for the same model as before, but for empymod and Dipole1D using the FHT method with the filter Key09-201, the default filter in *both* Dipole1D *and empymod*. For the regular versions ~~(-)~~ empymod and Dipole1D perform very similar; empymod is a little faster for the smaller tests, Dipole1D is slightly faster for the bigger tests. It can be seen from the results that the parallel optimisation ~~(par)~~ in empymod can result in a significant speed-up. If the lagged convolution optimisation is chosen, empymod is significantly faster than Dipole1D. In empymod, additional offsets come at basically no cost in the lagged version, whereas Dipole1D still uses more time to calculate more offsets. This is most likely due to the writing of the output file, that becomes bigger with bigger offsets. Dipole1D was compiled using the free and open-source *gfortran* compiler. Compiling it with the (proprietary) *ifort* compiler might result in slightly faster run times. *Note that Dipole1D always calculates all six receiver components, whereas empymod only calculates*

the required component(s). On the other hand empymod computes the whole, anisotropic wavefield, whereas Dipole1D is isotropic and uses the diffusive approximation.

COMPARISON OF DIFFERENT FOURIER TRANSFORMS

The following is a comparison of different Fourier transforms with the analytical solution. The electromagnetic impulse response of an isotropic half-space model below air with interface at $z = 0\,m$ for an x-directed electromagnetic source at the origin ($x_s = y_s = z_s = 0\,m$) and a receiver at an inline offset of 6 km at the surface ($r = x_r = 6\,km$, $y_r = z_r = 0\,m$) is given by

$$E_x(\rho, r, t) = \frac{1}{8}\sqrt{\frac{\mu_0^3}{\pi^3 t^5 \rho}}\exp\left(-\frac{\mu_0 r^2}{4\rho t}\right) , \tag{3}$$

where ρ is the halfspace resistivity, μ_0 is the permeability of free space, and t is time (neglecting the impulse of the airwave at $t = 0\,s$), as given by Wilson (1997, eq. 5.38).

Figure 7 shows the result of this comparison, where FFTLog with 15 ms is the fastest method and FFT with 666 ms the slowest one. Interesting is to see how many frequencies are used internally, as listed in Table 4. Even though FFT uses the least frequencies it is the slowest (and least precise) method. The reason is that it only calculates the response for 61 frequencies, but for the actual FFT it afterwards interpolates them at $2^{20} = 1\,048\,579$ frequencies to have a regular array from 5e-5 Hz to 52 Hz. To get a better result we would have to get to lower and higher frequencies, but then the number of required frequencies would even grow bigger. Note that this example is a difficult one, with source and receiver at the interface between air and the subsurface. Models where source and receiver are not at the same depth and not at the air-interface, for instance in marine CSEM where source

and receiver are in the water layer, are generally much easier, in other words, faster and more precise.

CONCLUSIONS

The presented code empymod is *an*~~a fast, lean,~~ open-source electromagnetic modeler ~~, well documented and hosted in a way that makes collaboration easy. It~~*that* can model the full wavefield of a 3D source in a VTI layered-earth. *Three different Hankel transform methods (quadrature, quadrature-with-extrapolation, and filters) are included into empymod, as well as four different Fourier transform methods (FFT, FFTLog, quadrature-with-extrapolation, and sine/cosine-filters).* Additional to the obvious application of modeling and inversion of electromagnetic data, empymod can *therefore* be used to investigate into the differences between *different Hankel and Fourier transforms, or between different filters* ~~filters and quadrature~~ for full-wavefield electromagnetic calculations over a wide range of frequencies *as well as times*. ~~There are not many full-wavefield codes openly available and, as such, empymod with QWE and FHT is an important addition to EMmod, which uses a 61 pt Gauss-Kronrod quadrature.~~ *The examples show that empymod can compete with existing code in both speed and accuracy, even though it is written in a dynamically typed language.*

Outside of the traditional scope ~~I think~~ empymod can be very educative for someone who is just starting to get interested in electromagnetic modeling or even just in Hankel *and Fourier* transforms, hence for educational purpose: it is ~~brilliant~~*well suited* for students, specifically as Python is becoming more and more widespread in academia, and the number of Universities that teach geophysicist Python (mostly instead of Matlab) is growing annually. Python, like Matlab, has the advantage of very fast developing times, and considerably lower entry barriers than C or Fortran for beginners.

There are many possibilities to improve empymod~~,~~ or to add additional functionalities~~,~~ ~~as in any software (this is why it is important to keep open-source code in version-controlled repositories such as GitHub instead in static repositories). Possible additions are~~, *for instance* more modeling routines~~,~~ such as *loop sources*~~arbitrary source and receiver dipole lengths, arbitrary source and receiver rotations, or variable receiver depths within one calculation. Other features to include could be further Hankel and Fourier transforms, such as the Gauss-Kronrod as in EMmod~~. *Improving code abstraction or creating a graphical user interface are further possibilities. The code comes with a complete testing-suite, but the included* ~~The~~ benchmarks ~~and tests included in the code~~ could also be improved and extended *with regression tests*. ~~By the time this article is published some of them might already be implemented by me or by the community.~~

ACKNOWLEDGMENT

I would like to thank the *Consejo Nacional de Ciencia y Tecnología*, México (CONACYT) for funding this postdoc, the *Instituto Mexicano del Petróleo* (IMP) for allowing me to publish the code under a free software license, and the entire electromagnetic research group of Aleksandr Mousatov at the IMP for fruitful discussions. I owe a special thanks to Jürg Hunziker for answering all my questions regarding his code and publication. *I thank the assistant editor, C. Torres-Verdin, the associate editor, F. Broggini, and Jan Thorbecke, Vladimir Puzyrev, and a third, anonymous reviewer, as well as Jürg Hunziker and Kerry Key, whose feedback greatly improved the clarity of the manuscript.*~~, and to both Jürg Hunziker and Kerry Key for feedback regarding this manuscript that greatly improved the article.~~

REFERENCES

Anderson, W. L., 1975, Improved digital filters for evaluating Fourier and Hankel transform integrals: Technical report, U.S. Geological Survey. (https://pubs.er.usgs.gov/publication/70045426).

———, 1979, Numerical integration of related Hankel transforms of orders 0 and 1 by adaptive digital filtering: Geophysics, **44**, 1287–1305. (doi: 10.1190/1.1441007).

———, 1982, Fast Hankel transforms using related and lagged convolutions: ACM Trans. Math. Softw., **8**, 344–368. (doi: 10.1145/356012.356014).

———, 1984, On: "Numerical integration of related Hankel transforms by quadrature and continued fraction expansion" by Chave (1983): Geophysics, **49**, 1811–1812. (doi: 10.1190/1.1441595).

———, 1989, A hybrid fast Hankel transform algorithm for electromagnetic modeling: Geophysics, **54**, 263–266. (doi: 10.1190/1.1442650).

Chave, A. D., 1983, Numerical integration of related Hankel transforms by quadrature and continued fraction expansion: Geophysics, **48**, 1671–1686. (doi: 10.1190/1.1441448).

———, 2009, On the electromagnetic fields produced by marine frequency domain controlled sources: Geophysical Journal International, **179**, 1429–1457. (doi: 10.1111/j.1365-246X.2009.04367.x).

Chave, A. D., S. C. Constable, and R. N. Edwards, 1991, Electrical exploration methods for the seafloor, *in* Electromagnetic Methods In Applied Geophysics Vol. 2: SEG, Investigations in Geophysics, No. 3, 12, 931–966. (doi: 10.1190/1.9781560802686).

Edwards, R. N., 2005, Marine controlled source electromagnetics: principles, methodologies, future commercial applications: Surveys in Geophysics, **26**, 675–700. (doi: 10.1007/s10712-005-1830-3).

Ghosh, D. P., 1971, The application of linear filter theory to the direct interpretation of geoelectrical resistivity sounding measurements: Geophysical Prospecting, **19**, 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**, 257–284. (doi: 10.1046/j.1365-8711.2000.03071.x).

Hunziker, J., J. Thorbecke, J. Brackenhoff, and E. Slob, 2016, Inversion of controlled-source electromagnetic reflection responses: Geophysics, **81**, F49–F57. (doi: 10.1190/geo2015-0320.1).

Hunziker, J., J. Thorbecke, and E. Slob, 2015, The electromagnetic response in a layered vertical transverse isotropic medium: A new look at an old problem: Geophysics, **80**, F1–F18. (doi: 10.1190/geo2013-0411.1).

Key, K., 2009, 1D inversion of multicomponent, multifrequency marine CSEM data: Methodology and synthetic studies for resolving thin resistive layers: Geophysics, **74**, F9–F20. (doi: 10.1190/1.3058434).

———, 2012, Is the fast Hankel transform faster than quadrature?: Geophysics, **77**, F21–F30. (doi: 10.1190/GEO2011-0237.1).

Kong, F. N., 2007, Hankel transform filters for dipole antenna radiation in a conductive medium: Geophysical Prospecting, **55**, 83–89. (doi: 10.1111/j.1365-2478.2006.00585.x).

Løseth, L. O., and B. Ursin, 2007, Electromagnetic fields in planarly layered anisotropic media: Geophysical Journal International, **170**, 44–80. (doi: 10.1111/j.1365-246X.2007.03390.x).

Nekut, A. G., and B. R. Spies, 1989, Petroleum exploration using controlled-source electromagnetic methods: Proceedings of the IEEE, **77**, 338–362. (doi: 10.1109/5.18630).

Shanks, D., 1955, Non-linear transformations of divergent and slowly convergent sequences: Journal of Mathematics and Physics, **34**, 1–42. (doi: 10.1002/sapm19553411).

Slob, E., J. Hunziker, and W. A. Mulder, 2010, Green's tensors for the diffusive electric field in a VTI half-space: PIER, **107**, 1–20. (doi: 10.2528/PIER10052807).

Wait, J. R., 1982, Geo-Electromagnetism: Academic Press Inc. (ISBN: 978-0127308807).

Wilson, A. J. S., 1997, The equivalent wavefield concept in multichannel transient electromagnetic surveying: Ph.D., University Of Edinburgh. (uri: hdl.handle.net/1842/7101).

Wynn, P., 1956, On a device for computing the $e_m(S_n)$ tranformation: Math. Comput., **10**, 91–96. (doi: 10.1090/S0025-5718-1956-0084056-6).

Werthmüller, D., 2017, Getting started with controlled-source electromagnetic 1D modeling: The Leading Edge, **36**, 352–355. (doi: 10.1190/tle36040352.1).

Ziolkowski, A., and D. Wright, 2012, The potential of the controlled source electromagnetic method: A powerful tool for hydrocarbon exploration, appraisal, and reservoir characterization: Signal Processing Magazine, IEEE, **29**, 36–52. (doi: 10.1109/MSP.2012.2192529).

LIST OF TABLES

LIST OF FIGURES

1 Analytical solution for a half-space with $\rho_h = 1/3\,\Omega\,m$, $\lambda = \sqrt{10}$, $f = 0.5\,Hz$, $z_s = 150\,m$, and $z_r = 200\,m$, calculated on a regular grid with a spacing of 10 meters; (a) amplitude and (b) phase.

2 Error levels for different Hankel transform settings, compared with the analytical solution in Figure 1. The high precision QWE (a) and the standard FHT (d) have very low error levels, generally in the order of 10^{-6} % or less. The lagged FHT has much higher error levels, and the effects of interpolation can be seen clearly in the ring-like structure. However, almost the entire error is below 0.1 % (10^{-1} %), only the dark black parts have an error of 1 % or slightly more. *Hankel arguments for QWE: [rel. tol., abs. tol., nr of pts]; for splined FHT: [pts/decade].*

3 Same as Figure 2, but for the phase. Interesting is to see that interpolation in the wavenumber domain (e) yields slightly different patterns than interpolation in space domain (f).

4 Comparison of the nine included FHT filters. All of them have an error mostly far below 1 %. For other models and other frequencies the result will be different. (Phase is not shown but looks very similar.)

5 Error of EMmod for the half-space model in Figure 1. On the left side with the same colorscale as in ~~Hun15~~Hunziker et al. (2015), on the right side with the colorscale as in Figures 2 and 3.

6 GPR example for (a) EMmod, *(b) QUAD for relative and absolute tolerance of* 10^{-12} *and* 10^{-20}*, respectively, (c) empymod with 51 pt QWE for relative and absolute tolerance of* 10^{-8} *and* 10^{-15}*, respectively, and (d)*~~(b) empymod with 51 pt QWE for relative and absolute tolerance of 10^{-10} and 10^{-18}, respectively, and (c)~~ empymod with FHT (Key09-

401); run times are given in the subplot titles.

7 Comparison of different Fourier transforms for an impulse response at the interface between air and a halfspace. FFTLog is the fastest, and FFT the slowest in this example.

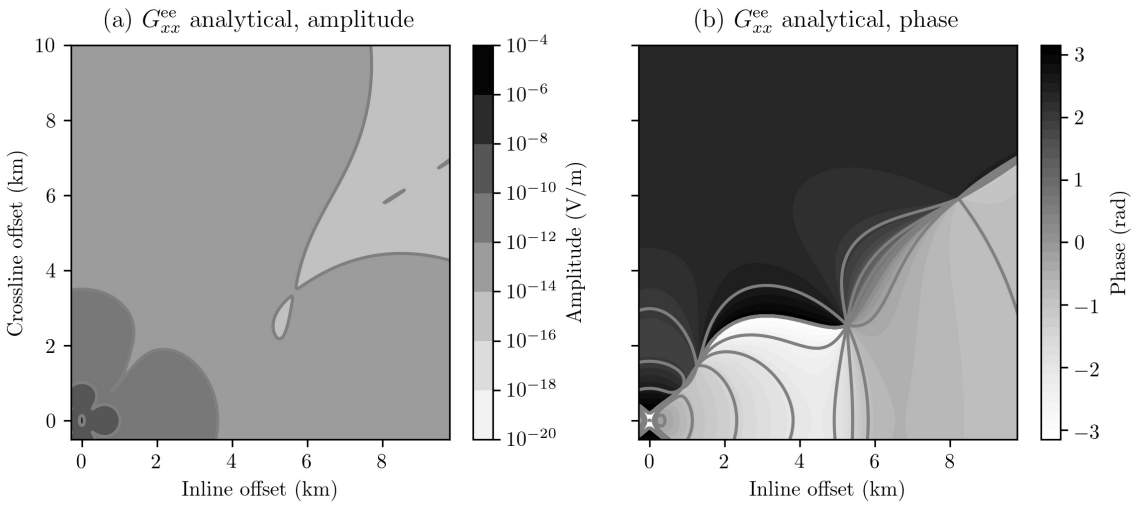

Figure 1: Analytical solution for a half-space with $\rho_\mathrm{h} = 1/3\,\Omega\,\mathrm{m}$, $\lambda = \sqrt{10}$, $f = 0.5\,\mathrm{Hz}$, $z_\mathrm{s} = 150\,\mathrm{m}$, and $z_\mathrm{r} = 200\,\mathrm{m}$, calculated on a regular grid with a spacing of 10 meters; (a) amplitude and (b) phase.

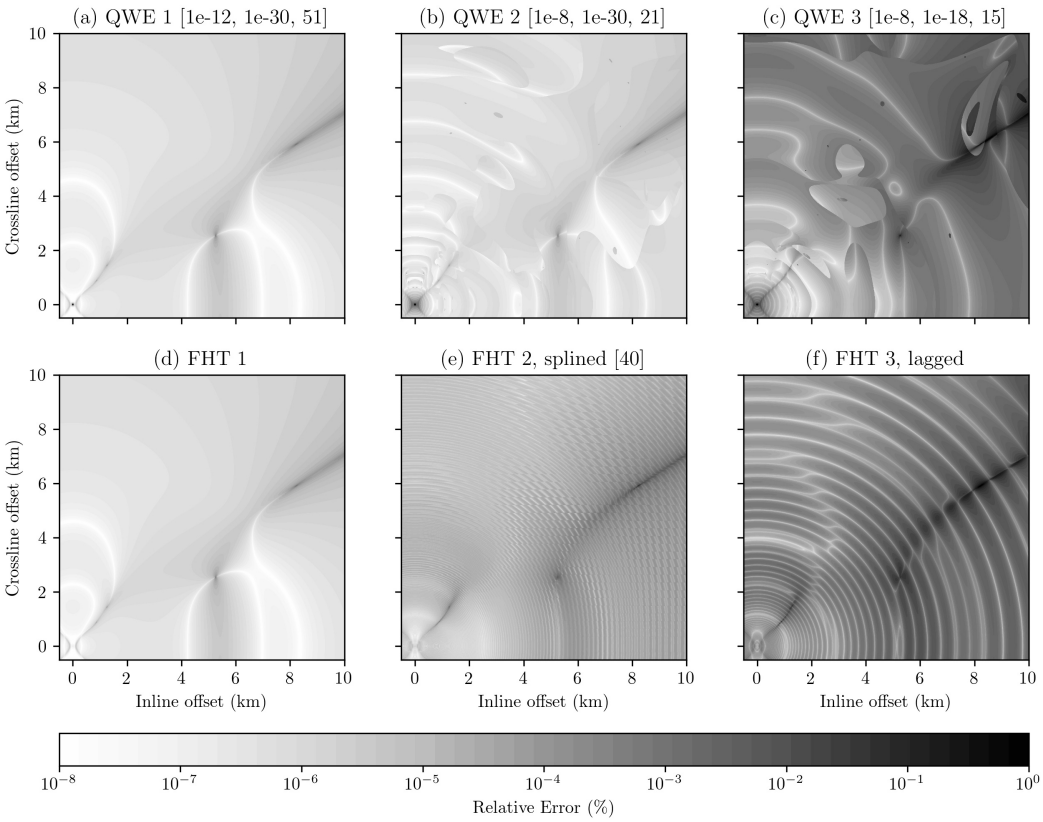

Figure 2: Error levels for different Hankel transform settings, compared with the analytical solution in Figure 1. The high precision QWE (a) and the standard FHT (d) have very low error levels, generally in the order of 10^{-6} % or less. The lagged FHT has much higher error levels, and the effects of interpolation can be seen clearly in the ring-like structure. However, almost the entire error is below 0.1 % (10^{-1} %), only the dark black parts have an error of 1 % or slightly more. *Hankel arguments for QWE: [rel. tol., abs. tol., nr of pts]; for splined FHT: [pts/decade].*

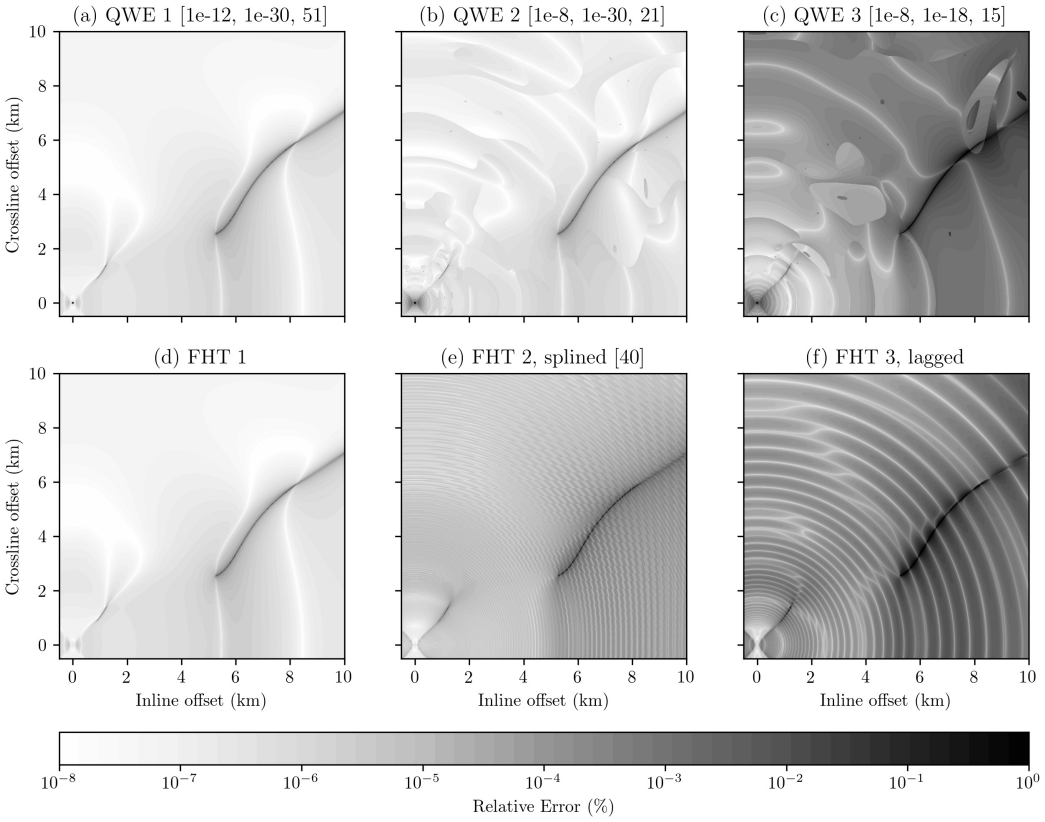

Figure 3: Same as Figure 2, but for the phase. Interesting is to see that interpolation in the wavenumber domain (e) yields slightly different patterns than interpolation in space domain (f).

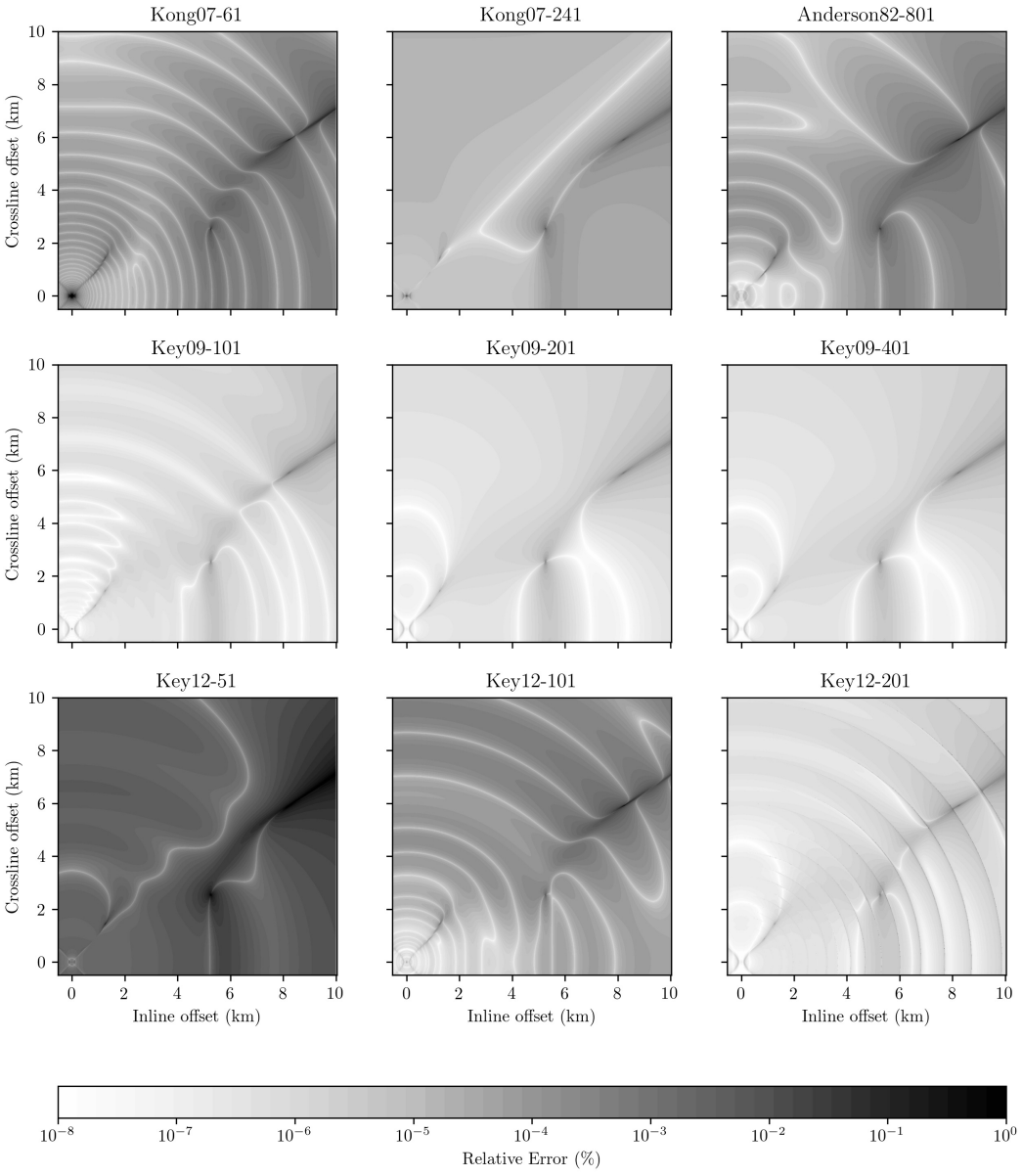

Figure 4: Comparison of the nine included FHT filters. All of them have an error mostly far below 1 %. For other models and other frequencies the result will be different. (Phase is not shown but looks very similar.)

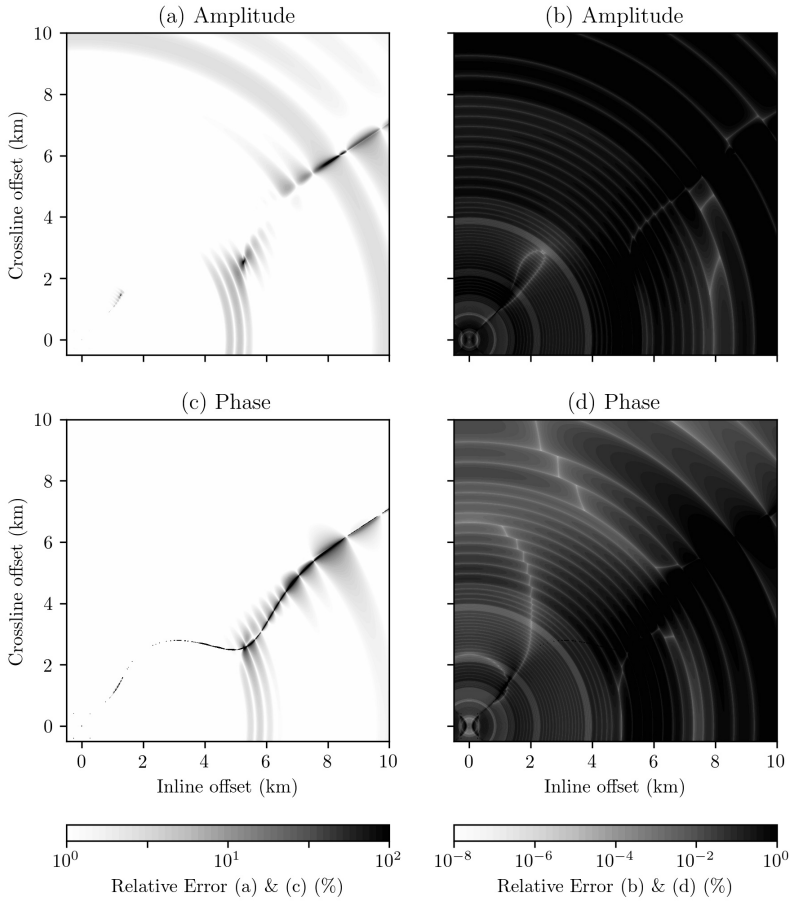

Figure 5: Error of EMmod for the half-space model in Figure 1. On the left side with the same colorscale as in ~~Hun15~~Hunziker et al. (2015), on the right side with the colorscale as in Figures 2 and 3.

EMmod	QWE			FHT		
	1	2	3	1	2	3
5040	10300	2530	1210	1480	875	6

Table 1: Run times for the half-space model shown in Figure 1, on a regular grid with spacing of 100 m, for EMmod and different Hankel transform settings of empymod (times in milliseconds).

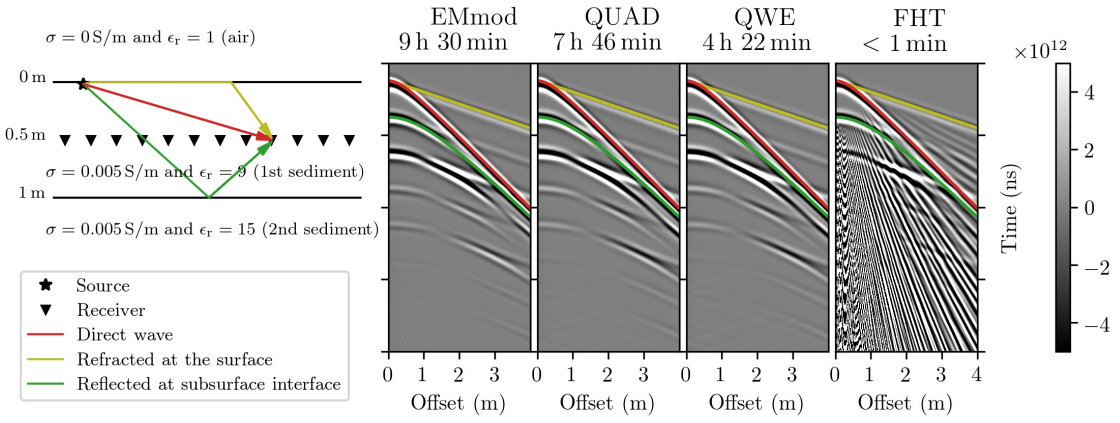

Figure 6: GPR example for (a) EMmod, *(b) QUAD for relative and absolute tolerance of* 10^{-12} *and* 10^{-20}*, respectively, (c) empymod with 51 pt QWE for relative and absolute tolerance of* 10^{-8} *and* 10^{-15}*, respectively, and (d)* ~~(b) empymod with 51 pt QWE for relative and absolute tolerance of 10^{-10} and 10^{-18}, respectively, and (c)~~ empymod with FHT (Key09-401); run times are given in the subplot titles.

		QWE: 9 pt				FHT: 201 pt filter				FHT: 801 pt filter			
		Normal		Splined		Normal		Lagged		Normal		Lagged	
Lay	Off	Wer	Key	Wer	Key	Wer	Key	Wer	Key	Wer	Key	Wer	Key
5	1	7	16	2	28	1	19	1	20	2	76	2	76
5	5	13	69	4	29	3	92	1	24	7	365	3	82
5	21	19	300	6	36	8	384	2	25	25	1522	3	83
5	81	37	1154	16	62	30	1475	2	28	100	5865	3	86
5	321	108	4581	58	165	124	5867	2	41	437	23165	3	99
100	21	118	463	13	51	92	604	8	37	304	2402	19	127
100	81	296	1792	23	75	337	2313	8	40	1234	9215	19	129
100	321	1021	7226	65	178	1408	9350	10	53	5448	36520	19	142

Table 2: Run times comparing *the codes from Key (2012)*~~Key12~~ (Key) and empymod (Wer) for the same cases as in (Key), Table 1 (in milliseconds).

		empymod			Dipole1D	
Lay	Off	-	par	spl	-	spl
5	1	1	2	1	4	4
5	5	3	3	2	6	5
5	21	9	5	2	12	8
5	81	30	10	2	36	18
5	321	124	34	2	130	56
100	21	91	53	9	90	13
100	81	340	106	9	333	23
100	321	1423	313	9	1321	62

Table 3: Run times comparing Dipole1D (~~Key09~~Key (2009)) and empymod, using the filter Key09-201 (default in Dipole1D; times in milliseconds). Optimisation: [-] None, [par] parallel, [spl] lagged FHT.

Method	# freq	min freq	max freq	time
		Hz	Hz	ms
FFTLog	70	1.8e-5	1.4e2	6
Sine-filter	128	5.3e-7	5.7e3	11
QWE	178	7.9e-5	6.3e4	298
FFT	61	5.0e-5	5.2e1	562

Table 4: Run-times for the four different Fourier transforms, and frequency range they require. More frequencies does not necessarily mean faster nor more precise. All Fourier transform do interpolation either in frequency or in time domain.

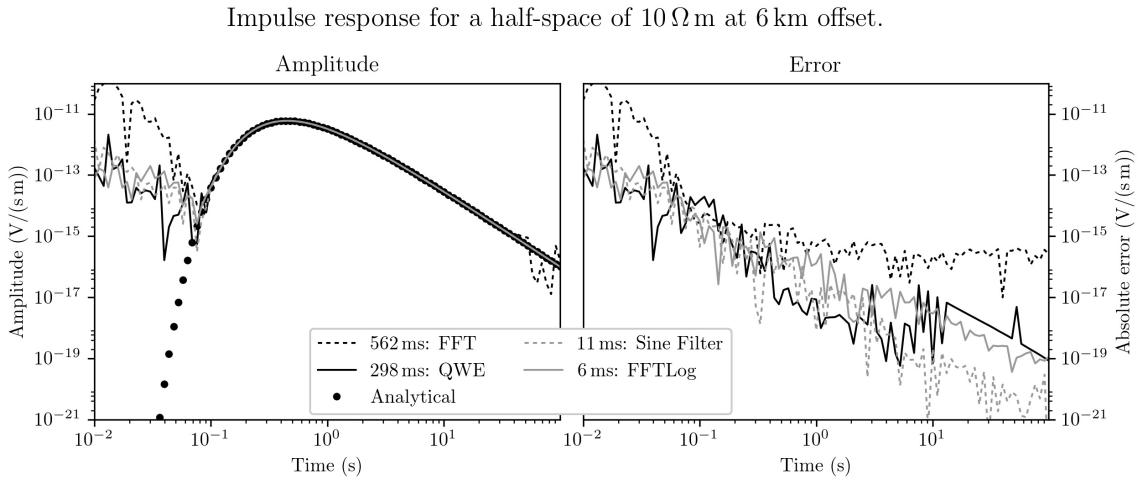

Figure 7: Comparison of different Fourier transforms for an impulse response at the interface between air and a halfspace. FFTLog is the fastest, and FFT the slowest in this example.