Inversion and Uncertainty Quantification for Airborne Electromagnetic Data: The `AEMpyX` Toolbox

Duygu Kiyan[1], Volker Rath[4,1], Somaye Bayat[1,2] & Mark Muller[3]

[1]Dublin Institute for Advanced Studies (DIAS)
[3]Geophysical Consultant, Chatteris, United Kingdom
[4]Geophysical Consultant, Chambery, France
[2]National University of Ireland, Galway (NUIG)

November 13, 2023

© 2023, Dublin Institute for Advanced Studies (DIAS). This work is released under the Creative Commons BY-NC-SA 4.0 license

Contents

Summary

The general aim of the project RAFTA (**R**esolution **A**nalyses for **F**requency- and **T**ime-Domain **A**irborne Electromagnetic Data of the Irish Tellus Programme) was to further develop the existing `aempy` toolbox for a better understanding of the uncertainties of AEM inversion.

From a computational viewpoint, we have transferred practically all of the original toolbox to Python 3.8+, taking advantage of the much improved external toolboxes of this ecosystem (as `numpy`, `scipy`, `matplotlib`, `gdal`, and more). The toolbox can now be used with Anaconda-like setups on Windows systems. In addition, we have implemented some new methods, which are particularly adapted for the quantification of uncertainties, both, based on the traditional inversions (Jackknife, RoI), and ensemble based (RTO, EKI). The general use of line-search is now available. Internally, considerable rewriting took place, with many improvements, the most important of which is the general availability of data and model parameters, which turned out to be highly relevant for time-domain data.

For the first time, we were successful in using our tools on the GENESIS TDEM data. The forward modelling here is much more time consuming (at least an order of magnitude), and some special solutions, such as data transforms were required. Further analysis demonstrated that in general only the vertical field can be inverted, because of the frequent occurrence of negative data in the in-line component. We performed a comparison of the two methods on synthetic data, and on two field areas (Bonduran and Clara), where ERT data existed or were measured within our project. However, the comparison of TD and FD field inversions do not produce consistent models. We were not able to identify the reason for this unfortunate result.

As weakest point relevant to model uncertainty we identified the lack of reliable error models. Data errors must always be interpreted as assumptions, which may be unsuitable for a give data set. While the additive-plus-multiplicative model used works well in the case of FDEM, it performs much worse for the TDEM data. As we think that the real data errors have meteorological origins, we see the hypothesis that error is constant, i.e., independent in time and space, is highly questionable, and needs further research.

Methods of uncertainty quantification as a general rule require much higher computing costs than inversions, and can even with parallel implementations only be used on small data sets, a typical for field projects. We have implemented and started testing novel, methods which hybridise the deterministic approach “inversion”) with stochastic approaches via randomised ensemble inversions.

1 Introduction

The development of the `aempy` toolbox started in early 2016, funded by the Geological Survey of Ireland (GSI). The objectives of this project were to develop and implement Bayesian methods for the quantitative interpretation of Tellus AEM data and provide open-source tools for inversion of these data for further research, and in particular useful for improving scientific training as well as build expertise for the interpretation and optimal use of the Tellus airborne data.

The extended toolbox, now renamed to `AEMpyX` (“aempy extended”), has been realised with a number of regularised inversion methods, which can be used for frequency- and time-domain data. Beyond the capability for resistivity inversion, which is possible with most existing software, it allows for the estimation of magnetic susceptibilities and Cole-Cole induced polarisation parameter within all implemented algorithms. In particular, we provide tools for the systems used in the

Tellus surveys, namely the frequency-domain system originally developed by the Finish Geological Survey (GTK), now used by Sander Geophysics [89], and the GENESIS system developed by Fugro/CGG as a derivative of CSIRO's TEMPEST system [71], adapted for shallow (hydro-geophysical) targets.`AEMpyX` was developed and implemented in project RAFTA (**R**esolution **A**nalyses for **F**requency- and **T**ime-Domain **A**irborne Electromagnetic Data of the Irish Tellus Programme), and can be seen as a follow-up to the original `aempy` toolbox development. It grew from the need to proceed from the simple inversion of the AEM data to the further analysis, in particular the characterisation of uncertainties (UQ).

We implemented a highly flexible toolbox using the Python language for the 1-D inversion of frequency- and time-domain AEM data,which is aimed mainly at a better understanding and easy experimentation with inversion methods. The computational core is based on an adapted forward model derived from the well-tested open-source code AirBeo. This code was originally developed by Australia's CSIRO and the AMIRA consortium, and the latest version is available from `https://sourceforge.net/projects/p223suite`. Since the original `aempy` implementation, the code was migrated from Python 2.7 to 3.9+ (currently being developed under 3.9). The choice of Python for the development was motivated by the high flexibility of this language, and the vast ecosystem of useful toolboxes available. As of 2023, the weighing of advantages and drawbacks would favour the Julia language `https://julialang.org/`, because of the impressive advantage in computational speed, and ease of programming, in particular when introducing parallelism [e. g., 135].

Using this computational core (currently for 1-D layered forward models), we implemented several inversion techniques. Though we concentrate on methods usually classified as "highly parametrized" in the sense of [37], few-layer inversions are still possible. In particular:

- Tikhonov-type inversion including optimal regularisation methods employing different choices of the regularisation parameter(s) [6, 158];
- Bayesian MAP inversion in parameter and data space [142, 143, 141];
- Minimum (Gradient) Support (MGS) variants [112, 158, 157];
- Fully Bayesian inversion [99, 98, 100, 123, 46] by Markov Chain Monte Carlo using the existing home-made Metropolis-Hastings-implementation (MH).
- New in *AEMpyX*: Jackknife estimation of variance - a simple resampling method based on a leave-out-n approach [42, 40, 145, 117, 41];
- New in *AEMpyX*: Randomise-then-Optimise (RTO) and related methods [106, 8, 9, 16, 17];
- New in *AEMpyX*: Ensemble Kalman Inversion and similar ensemble methods [104, 67, 18, 49, 19, 20, 34].
- New in *AEMpyX*: Bayesian inversion [99, 98, 100, 123, 46] by Markov Chain Monte Carlo calling parallel open source libraries like `PyMC` [93], `emcee` [47],`DRAM` [57, 96], or MT-DREAM(ZS) [80]. These toolboxes represent the experience of many years of development, and are much more efficient than our vanilla implementation. They also contain advanced computational approcahes, such as parallelism, and specialised visualisation tools.

Apart from the reformulated and added high-level inverse algorithms, there were many internal algorithmic and structural changes, often not visible to users. The most important of these are:

- All algorithms now allow very general transformations of both, data and parameter vectors. Data transforms are commonly used for TDEM, in particular to deal with the large bandwidth (log-transform), and the presence of negative values (asinh-transform)[128, 129, 61]. Parameter transforms include of course the logarithm of resistivity or conductivity (used generally in EM inversions because of the huge spread of possible values over more than six decades), but also alternative parametrisations for induced polarization [e.g., 45]. As before, data and parameters can be activated or deactivated.

- The deterministic inversion algorithms mentioned above (Tikhonov, MAP, MGS) have been complemented by a line search algorithm, as suggested e.g., in Tarantola [141]. This led to much improved convergence in nearly all cases.

- As the prior/initial model influences inversion results considerably, we have introduced the option of reading the priors from a pre-calculated half space model. This is important if dealing with flight lines crossing different geological zones, where the depth-averaged resistivities may differ over more than a decade, e. g., limestone and quaternary sediments. This also may speed up the convergence of the inversion procedure.

- Minor novelties include a revision of the difference regularization operators (different null spaces), explicit covariance matrices (Matérn), and new choices of regularisation parameter, as the L-Curve [60, 58].

- Much of often-used visualisation code has bee been moved to a dedicated module `viz.py`. The different high-level algorithms are now exclusively in the `alg.py`, and they now can be called in a homogeneus API.

The algorithms are exemplified high-level scripts including a full work flow from loading of raw data over pre-processing to inversion. They import modules, which contain functions providing numerical procedures, constraint methods, processing tools, and utilities. In the tutorial (Section 5), we give an overview of the typical work flow for an inversion which is constructed from these building blocks. The idea behind this approach is to be able combining the existing functions for fast prototyping new and improved work flows. This can be on the level of inversion algorithms, but also for designing more complicated procedures, such as using the output of MAP inversion for the prior in MH simulations, or a two-stage procedure for obtaining susceptibility using the results from prior resistivity-only inversion. The setup for these inversions is highly flexible, as activity indicator matrices are used to choose the data to be used for inversion, and more important, the parameters can be determined likewise. The pre-processing functions include the masking of non-physical data, interpolation, PCA filtering of data [97]. Inversion algorithm can be combined with different type of regularisation such as covariance matrices in MH and MAP, or differential operators in Tikhonov inversions. Utilities include data managing tools, such as reformatting raw data to an internal format, data choice from a polygon, or a profile projection, and various visualisation methods.

Apart from moving from Python 2.7 to 3.9+, `AEMpyX` was ported to the Windows 10 operating system (Windows 11 not available for testing). The installation procedure is slightly more difficult as under Linux, as some of the tools used are not native to the Windows environment (Fortran compiler, git, and more), and a few additional installations are needed.

In the following, Section 6 gives some necessary information and results concerning the data for both systems, including the approach taken for the preprocessing of the data. Section 3 describes the scientific work done with respect to the different new methods and other accomplishments,

including the comparison with co-located ERT measurements. Section 4 will give a some technical information on the toolbox, in particular it's prerequisites and installation under the linux and windows operating systems. Finally, Section 5 will present a hands-on guide through the most important steps when using `AEMpyX`.

2 Data

In this section, we will concentrate on our recent study of the Tellus data, in particular on the problem of defining reasonable models for the error present in the input data. A few steps for the FD data have already been taken in [75, 76]. The inclusion of the much more problematic TD data made it necessary to investigate further.

2.1 Tellus AEM systems

As of today, the Tellus Programme includes the Airborne EM surveys given in Table 5. One of the important features of this huge data set is the use of two systems, or veven three if the Northern Ireland part is included.The North Midlands survey comprises acquisition of time-domain EM (TDEM) data, which was carried out by CGG Airborne Survey (Pty) Ltd. with the GENESIS system, while the other surveys include acquisition of frequency-domain EM (FDEM) data, which were carried out by Sanders Geophysics Ltd. with the GTK airborne system (AEM-95), and the Joint Airborne Geoscience Capability (JAC) airborne system (AEM-05). This situation is one of the motivations for this research project.

2.2 Frequency-Domain EM Systems - AEM-05 and AEM-95

The Northern Ireland part of the Tellus Programme was carried out with GTK AEM-95 system which had two frequencies 3125 and 14368 Hz), and the vertical, co-planar coils had a wing tip separation of 21.4 m. The other FD surveys were completed using the improved AEM-05 system which was developed as Joint Airborne Geoscience Capability (JAC) by a partnership between the Finnish and British Geological Surveys. The AEM-05 system operates at four frequencies, 912 Hz, 3005 Hz, 11962 Hz, and 24510 Hz, and subsequently operated by Sanders Geophysics Ltd, (SGL). Their DHC-6 Twin Otter is configured with a four-frequency, wing-tip mounted FDEM system. The transmitter-receiver coil pairs are mounted in a vertical-coplanar orientation which reduces noise by minimizing coupling with the wing tip surface. Additionally, the coils in any one set (transmitter or receiver) are axially offset and are kept adequately separated from each other. The system also comes equipped with a 50/60 Hz power line monitor which becomes particularly useful in identifying cultural interference when surveying in urban settings. The system has a 40 Hz sampling rate which is later decimated to 10 Hz in the processing. Survey acquisition parameters for the FDEM surveys are summarised in Table 6.

Table 1: State of the Tellus AEM survey as of January 2022. Several blocks were reprocessed and delivered to SFI by SGL several times.

Block	System	Report	#Sites	Remark
NI	AEM-05 AEM-95			< 2006, Tellus Northern Ireland
CV	AEM-05			2006, Joint Airborne Geoscience Capability (JAC) pilot studies (Cavan-Monaghan, Silvermines and Castleisland)
TB	AEM-05			2013, Tellus Border (Donegal, Sligo, Leitrim, Cavan, Monaghan, Louth)
NM	CGG GENESIS			2015 North Midlands (Roscommon, Longford, Westmeath)
A1	AEM-05			2016, A1 block, (Meath, Kildare, Offaly, rural Dublin and parts of Laois, Wicklow)
WF	AEM-05			2016, Waterford (Waterford, parts of southern Tipperary and Kilkenny)
A2	AEM-05			2017, A2 block (Galway, including parts of Mayo, Tipperary and Offaly
A3	AEM-05			2018, A3 block (Mayo)
A4	AEM-05			2018 A4 block (Donegal
A5	AEM-05			2019 A5 block (Limerick)
A6	AEM-05			2019 A6 block (West Cork)
A7	AEM-05			2019 A7 block (SE Ireland)
A8/A9	AEM-05			2020/21 A8/A9 blocks (Cork and neighbouring parts of Limerick, Waterford and Tipperary)

Table 2: Summary of FD survey data acquisition parameters.

Flight Line Spacing:	200 m
Flight Line Direction:	SE-NW (≈345°)
Tie Line Spacing:	2000 m
Nominal flying height:	60 m above ground level (rural), up to 240 m (urban)
Data sample intervals:	<10 m (10 Hz)

Apart from the data proper, the latest deliveries provided by SGL include a new Power Line Monitor (PLM). This magnetometer power line monitor data channel included is derived from the 160 Hz magnetic data. These measurements are fed through a frequency-domain band pass filter centred on 3 samples (i.e., 0.01875 s). This step extracts the 50 Hz power line signal that is observed in the magnetometer while suppressing all other signal. The absolute value is taken from the output of that band pass filter and is subsequently passed through a

Figure 1: Airplane operated by SGL for the FDEM surveys.

median slope time-domain filter with a window of 8 samples (i.e.,0.05 s), effectively estimating the noise envelope. The magnetometer power line monitor channel is not as susceptible to interference from spurious sources such as radio transmitters and is also able to detect power lines with less current. The magnetometer power line monitor data channel is included with the frequency-domain electromagnetic data. This PLM is supposed to be better than the one provided by SGL previously. Any indicators of potential noisy data is highly desirable, given the dense net of electrical installations in Ireland. We used threshold values between 2 and 10 for this parameter when qualifying the data as "bad", and potentially excluding them from the inversion process. We find, however, that the use of this parameter has disadvantages, which are connected to the choice of threshold. In may cases it is difficult to find a level which at the same time leads to reasonably short exclusion intervals, and connected gaps. This is due to the sometimes complicated structure of the signatures, and the presence of trends, often at the beginning of flight lines (see Figure **??**). In many case we were not able to see observational effects in zones with high PLM or vice versa - which may be due to the underlying physics (geometry of noise source, and their activity). This has also been shown in the recent GSI report on data variability [102], which gives a comprehensive study on potential error sources based on nearly co-located test lines. A further study on the behaviour of this data quality indicator on synthetic and field cases would be strongly indicated.

2.3 Time-Domain EM System - CGG GENESIS

For the Northern Midlands (NM) Survey of the Tellus Programme, the AEM data were acquired by CGG Airborne Survey Ltd. between September 2014 and June 2015 using the GENESIS system described by CGG (2015). It is essentially a TEMPEST system, adapted to shallow surveys. The GENESIS 3-axis towed bird assembly provides accurate low noise sampling of the X (horizontal in-line), Y (horizontal transverse) and Z (vertical) components of the electromagnetic field. However, only the vertical and in-line components were processed and delivered. While this makes sense with respect to modelling, it impedes the application of corrections for potentially important errors related to bird movement, as described for the TEMPEST system [134, 133, 21, 22].

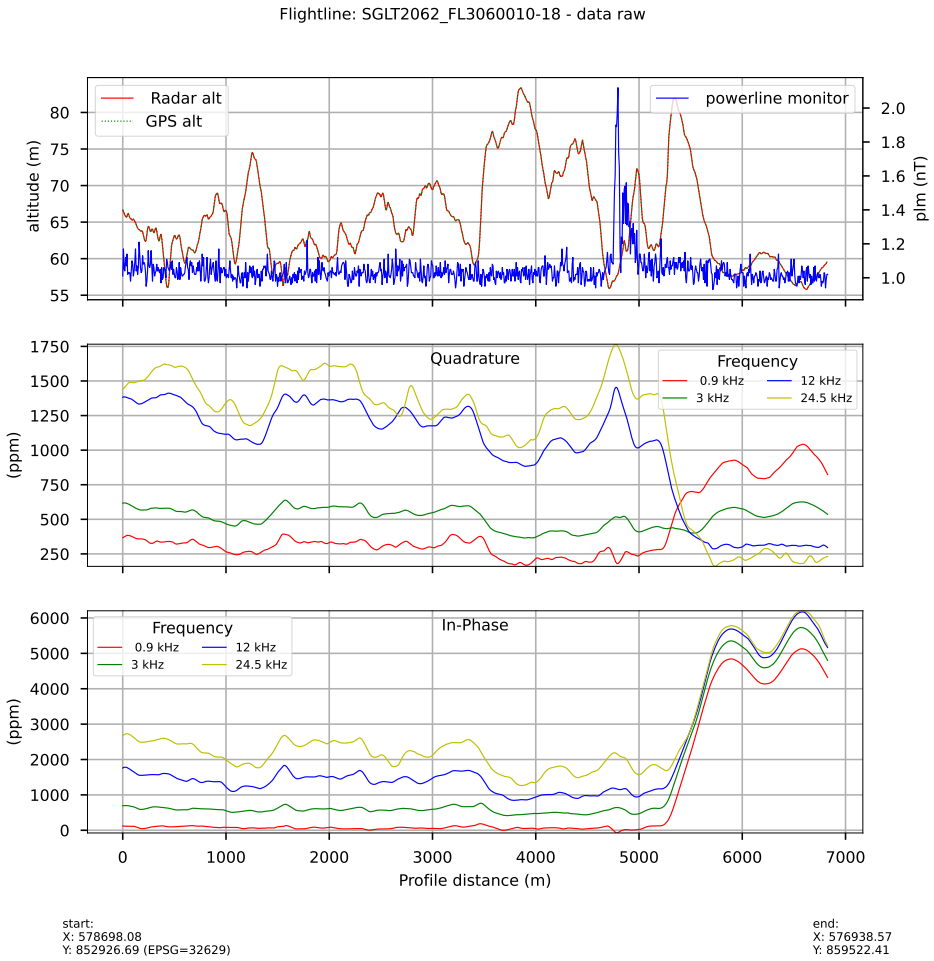

Figure 2: Field data originating from the SGL AEM05 system, from the Bundoran test lines, including the new magnetometric PLM. In this case it is not easy to decide whether the PLM correctly indicates intervals of disturbed data, which need to be left out for inversion.

The receiver coils measure the time derivative of the magnetic field, which is integrated during the processing, such that the outcome is in fT. Signals from each axis are transferred to the aircraft through a tow cable specifically designed for its electrical and mechanical properties. The EM transmitter (TX) was mounted above the aircraft with a loop running between the wings and tail mount. The EM receiver (RX) was deployed on a bird 45 m below the aircraft. Survey acquisition parameters are summarised in Table 7.

When working with the GENESIS data, there are considerable problems obtaining correct technical information from CGG, who, moreover, have sold their full airborne business (including staff) to a new company, Xcalibur Multiphysics (`https://www.xcaliburmp.com`). The information in report CGG [28], literature [39, 133], and the AirBeo input files received from CGG do not agree. We believe, however, that we have found the correct setup, reproducing the original AirBeo

Figure 3: Airplane operated by CGG for the TDEM survey in the Northern Midlands.

modelling provided by CGG (Figure 36). Furthermore, careful re-checks of the TD Jacobian calculations (including the asinh transform suggested in the literature) show stable behaviour for a wide range of perturbation parameters. Thus we conclude that from the technical viewpoint our implementation of the GENESIS system is correct.

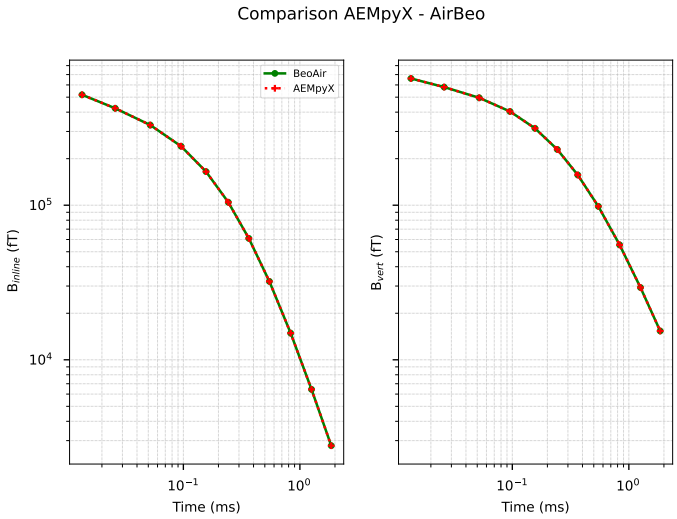

Figure 4: Comparison of original AirBeo forward modelling with AEMpyX core module results. The results are reproduced to full accuracy (14 digits).

The first observation with the GENESIS data is the extraordinary amount of negative values

Table 3: Summary of TD survey data acquisition parameters.

Flight Line Spacing:	200 m
Flight Line Direction:	SE-NW (≈345°)
Tie Line Spacing:	2000 m
Nominal flying height:	90 m above ground level (rural), 240 m (urban)
Data sample intervals:	≈12 m (5 Hz)

in the data, particularly in the in-line horizontal component, as seen in Figure 37. A simple counting statistics for Block NM is presented in Table 8. Negative values in TDEM may result from a variety of reasons. They may occur when conditions are not 1-D, which will produce negatives mainly in the X component, as seen in Table 8. For this reason we concentrate on the vertical component (Z) for the 1-D inversions of field data. One-dimensional layered, pure conductivity models can not produce sign-reversals in the transients.

Noise effects apply mainly to the early and late times of the transients, but particularly when negative values in the transient occur a late times, noise may be the reason. If no strong perturbances (as power lines) are present, there may exist effects of induced polarisation (IP), producing negatives all over the time windows. These effects can be used for further characterisation of the subsurface, and are important in ore exploration, or wherever clay minerals are present. In Ireland, black shales are common [see 121]. In principle, the corresponding parameters can be inverted for, though this increases the number of parameters considerably (three more per layer for the Cole-Cole parametrisation). Though their effects have been known for some time [26, 84, 85], super-paramagnetic (SPM) rocks/soils have only lately come into the view of AEM [124, 91, 92]. As with IP, they could be used for further characterisation of the shallow subsurface.

In the case shown in Figure 37 some observations can be made: In both, the vertical (Z) and the in-line (X) component, the last two channels are already noisy, though they in principle characterize deeper structures and should therefore show a smoother character. As the in-line (X) component is considerably smaller than the vertical(Z), the negative values occur mainly in X component. This is consistent with the correlation of the negative data with the flight altitude, resulting from the exponential decrease of subsurface signal with system flight altitude. While it seems to be possible to work with the Z component (with an appropriate channel-dependent error model), the last two X channels are not usable in most cases. It can also be observed that the flight altitude changes considerably (in all test lines), which could be taken as an indication of problematic atmospheric conditions (see Section 6.4).

It comes clear from the basic physics of the two systems, that their potential depth of investigation and resolving powers are largely different. Figure 38 shows the Jacobian matrices for both systems, which display large difference in structure and values.

2.4 Noise in field surveys

The success of Bayesian and other inverse approaches depends strongly on our ability to obtain a realistic noise model. As shown later in Section 3, this choice influences not only the weighting of data-fit and regularisation terms in the objective function (Equation **??**), but also the relative importance within the data set. As the data span several magnitudes, particularly in time-domain

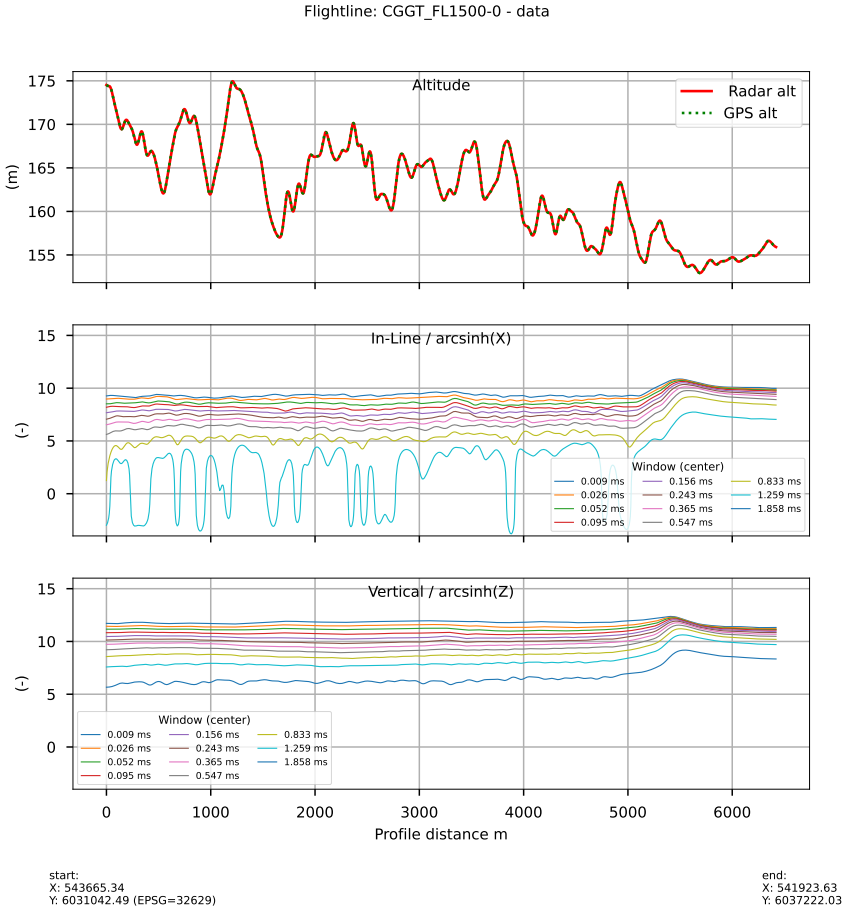

Figure 5: This is an example for field data originating from the CGG GENESIS system. Note that because of the large bandwidth of the data, and the presence of negative values in the in-Line (X) component, the data were transformed using the *asinh* function [129]. Data were taken from the Bonduran test site.

methods, a simple additive error will not suffice, and even a combined additive and multiplicative model, as implemented in `AEMpyX` may not be enough. It has to be kept in mind that any error model used must be seen as a hypothesis, which will require revisions whenever new information is available, as also corroborated by the recent variability study [102].

[76] has shown that the errors estimated from the Bundoran high-fly experiment are spatially (and thus also temporally) correlated [76]. The correlation length for these perturbation is in the order of a few 100 m, corresponding to some seconds in time. No real frequency dependency could be detected, and for the real inversions we originally assumed a constant additive error of 30 ppm to 70 ppm, which always led to reasonable convergence an a good data fit. In some cased this was complemented with a 3% to 5% multiplicative error, in order to inhibit over-fitting the largest values. In view of the more recent study by Muller [102], this may be slightly low, with about 100 ppm more realistic within this framework.

Table 4: Summary of GENESIS TD survey data: negative values. The total values are 28.% for X (in-line component), 2.9% for Z (vertical component), and 15.5% for all data.

Window (ms)	X (%)	Z (%)
0.026	7.26	0.96
0.052	15.2	0.13
0.095	5.1	0.075
0.156	12.0	0.28
0.243	19.6	0.47
0.365	31.7	1.12
0.547	43.2	1.89
0.833	41.8	2.56
1.259	47.9	4.92
1.858	84.0	19.8

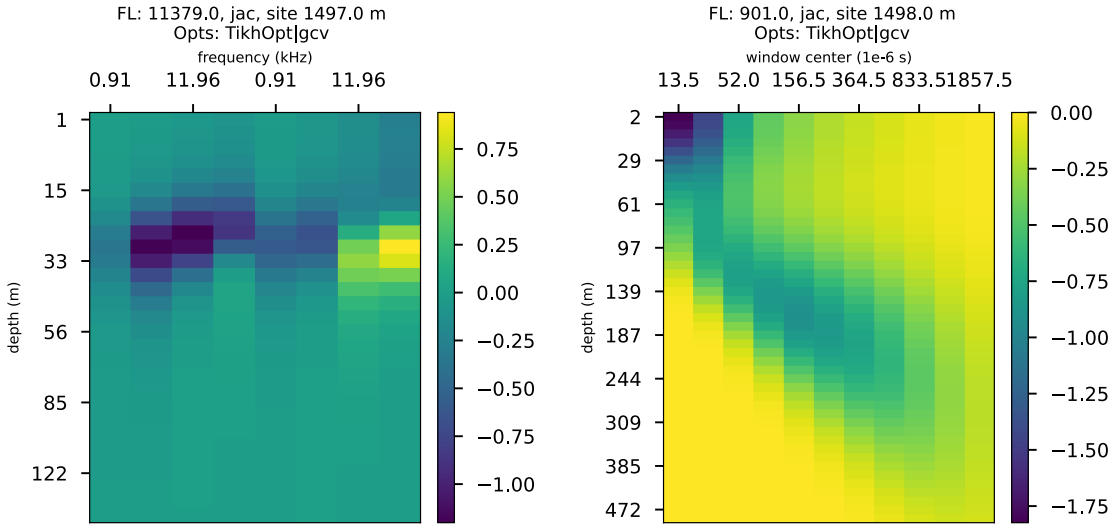

Figure 6: This figure shows an exemplary comparison of Jacobian matrices for TD- and FD systems. Data were taken from the Bonduran test site (TD), and the A1 block (FD).

In the Tellus project, error estimation was attempted by flying test lines for both systems in the area of Bundoran (Co Sligo), and later for the FD systems in the Waterford area. Kiyan and Rath [75] and Kiyan et al. [76] present an analysis of the so-called high-fly experiment for the AEM05 system at Bundoran. In contrary to a commonly made assumption [70, 62], we think that the noise is not purely random, but is the result of neglected, unconsidered, or even unknown physical processes. Here, it is plausible to think that properties of the atmosphere are involved. Heterogeneities of atmospheric or wind properties are the probably best candidates, which may in turn will be influenced by surface conditions (e. g., canopy). As already suggested by Muller [102], a detailed surface field study may be useful.

The very procedure of the high-fly experiment [54] has some inherent weaknesses. It depends on the assumption that noise is independent of the flight altitude, and can thus be separated from

the signal part originating from the subsurface. As we think the "noise" is related to physical processes, this is certainly not true. The altitude range between 50 m and 300 m covers the lower part of the atmospheric boundary layer (ABL), is well-known to be highly heterogeneous in space and time [e.g., 137, 136, 139, 86] . While a direct influence on the measurements is not easy to conceive, the heterogeneities in atmospheric conditions including turbulence may produce perturbations in movements of the plane (i.e., pitch, roll, yaw) for both systems, and the relative motion of the bird for GENESIS. These parameters unfortunately are not available in the Tellus database. If our hypothesis is correct, they might be used for a correction during the pre-inversion processing. As far as we know, only the plane movement has been corrected for the GENESIS system [28]. The problem is aggravated by the choice of the test line area, which is in a land/sea breeze zone, which is particularly prone to changes in ABL structure and turbulences [149, 5]. This can be nicely observed in the GENESIS data shown in Figure **??**, where the difficulty of keeping a constant altitude is obvious. The spatial nd temporal correlation found for the FD observations [76] is not far from the scales of turbulence in the lower ABL [1]. No local weather station are available for both test line zones. For these reasons, a repeatability approach [70, 62] may be more adequate for determining a "realistic" noise model, as proposed by Green and Lane [54] for the TEMPEST system. High-fly estimates of additive noise seem possible for both systems, and a number of repeats for determining the multiplicative errors are available, though their number is low.

3 Methods

In this section we summarise the methods used in the frame of `AEMpyX`, as far as it is necessary to understand the subsequent setions. As Inversion builds upon the repeated solving of a forward problem, i. e., calculating data from a set of parameteres, we will shortly describe the analytical methods used in forward modelling in Section 3.1, and the inversion approaches in Sections 3.2 and 3.3.

Inverse techniques may be classified following different paradigms. From a more philosophical point of view, the most important division is between the Bayesian approach [141, 120, 142, 143, 98, 99], and deterministic, optimisation-related methods [109, 90, 110, 58, 59, 38, 7]. While the former usually are realised through "large" model ensembles (as in McMC, EKI, or RTO implementations), the latter require the solution of an optimisation problem.

Before describing any method used here in more detail, a few general remarks are necessary. Inversion is the production of parameter estimates from a particular combination of an underlying physical model, and measured data. Uncertainty, as resolution, is not very well-defined concept (see, e.g. Section 3.2.1), and in consequence there is no single method covering all aspects. Fortunately, some of the methods described below will give answers to important but better defined questions, giving a somewhat arbitrary and incomplete collection:

- Which size can be assigned to the error bars which correspond to a parameter estimate?
- Which parameter estimates are independent, which are correlated? How many parameters can be estimated from the data?
- What are the minimum and maximum values to be expected?
- How are any of the mentioned estimates distributed spatially (and potentially, with time)? How deep can we look into the subsurface?

In particular none will inform us on the physical origin of the measurement errors, or the appropriateness of the model used for inversion.

3.1 Forward problem

The theory of electromagnetic induction in layered media is based on the excellent and detailed presentations in [152, 53, 72, 153, 63, 154, 71]. In this section, we will give a short summary of the physical setup for the models, and the necessary information on the forward modelling technique.

3.1 Physical Set-up

The 1-D (layered) forward model assigns 6 physical properties to each layer n of thickness $\delta z_n = thk_n$. This general set-up makes it possible to model a layered response in many situations, though it has been used mainly for electrical resistivity. These are:

- the electrical resistivity ρ_n (Ωm);
- the electrical permittivity $\epsilon_n = \epsilon_{r,n}\epsilon_0$ (Fm^{-1});
- the magnetic permeability $\mu_n = \mu_{r,n}\mu_0$ (NA^{-2});
- the IP parameters defined by the Cole-Cole parametrisation described below, namely the chargeability m_n (-), the time constant t_n (s), and the frequency constant c_n (-).

Here the index 0 marks the corresponding free space value of the magnetic permeability and the dielectric permittivity, the r denotes the relative value used for the parametrisation in the respective layer. These possibilities make `AEMpyX` a highly versatile tool for many geoscientific problems.

While the usefulness of magnetic properties is straightforward, the IP parameters require some more explanation. The origins of Induced Polarisation effects (IP), which may be relevant in areas with high clay content or disseminated metallic particles (e. g., ores) have been described in detail in [138, 13, 111]. Due to their strong effects in AEM ("negative transients"), they are currently a very active area of research. The most common parametrisation for the frequency-dependence of electrical resistivity ρ related to induced polarisation effects, was introduced already in [35], and is therefore called the Cole-Cole model and is implemented within the forward model calculations by the following equation:

$$\rho(\omega) = \rho_0 \left[1 - m\left(1 - \frac{1}{1 + (j\omega\tau_\rho)^c}\right)\right] \quad , \tag{1}$$

With the low- and high-frequency limits, ρ_0 ρ_∞, the chargeability can also be expressed as:

$$m = \frac{\rho_\infty - \rho_0}{\rho_0} = \frac{\sigma_0 - \sigma_\infty}{\sigma_\infty} \quad . \tag{2}$$

However, other parametrisations are possible, which may be better suited for certain petrophysical conditions in the subsurface Fiandaca et al. [45], Binley and Slater [14].

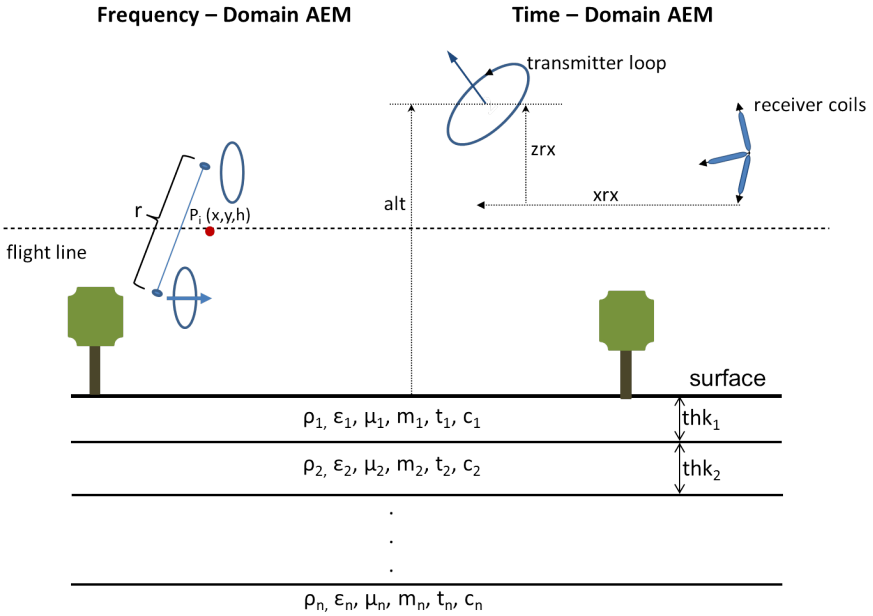

Figure 7: Schematic view of a vertical-coplanar loop configuration over a layered-earth model. The AEM response is positioned at the centre between Transmitter (TX) and Receiver (RX). The spacing (r) between them is kept constant. For the frequency-domain Tellus Surveys, r= 21.4 m.

3.1 Frequency-Domain EM methods

The FDEM methods are based on the EM induction where time varying magnetic field generated by the sinusoidal current of the transmitter (TX) generates currents in the subsurface conductors. Resulting from the primary EM fields, these generate secondary EM fields, where the eddy currents induced in the conducting subsurface are included. The total magnetic field that will be measured at the receiver (RX) loop is the sum of the primary (B_p) and secondary (B_s) magnetic fields. The FDEM response is defined as the secondary magnetic field (B_s) normalized by the primary (free space) magnetic field ($B_p = B_0$). Note that due to the complex nature of the fields in the frequency domain a phase shift between B_p and B_s exists. Thus the AEM response can be defined as in-phase (real) and quadrature (90^o out of phase or imaginary) components:

$$\mathcal{I} = \left(\frac{\Re(B_{tot})}{B_0} - 1\right), \quad \mathcal{Q} = \left(\frac{\Im(B_{tot})}{B_0}\right) \tag{3}$$

where $\Re(B_{tot})$ is the part of the total field that is in the same phase as the primary field and $\Im(B_{tot})$ is the part that has 90^o phase shift with the primary field. AEM data are presented in units of parts per million (ppm). Figure 7 shows a layered-earth model used for AEM computations. The response is computed at data location $P_i(x, y, h)$, which is the centre point between TX and RX. x, and y are the coordinates of the data location, and h is the flight altitude.

The AEM response of a dipole source and a receiver above a layered-earth is computed using

the solutions given in [72] as

$$\frac{B_s}{B_0} = r^2 \int_0^\infty R(\lambda)\lambda e^{-2\lambda h} J_1(\lambda r) d\lambda \quad , \tag{4}$$

where r is the coil separation, λ is the variable of integration, h is the elevation of the TX and RX coils, and J_1 Bessel function of the first kind of order one. The term $R(\lambda)$ can be written as

$$R(\lambda) = \frac{Y_0 - \hat{Y}_1}{Y_0 + \hat{Y}_1} \quad , \tag{5}$$

where $\hat{Y}_1$ is the surface admittance and $Y_0 = u_0/(i\omega\mu_0)$ intrinsic admittance, i is the imaginary unit ($i^2 = -1$), ω is the angular frequency ($\omega = 2\pi f$), μ_0 is the magnetic permeability of free space. For an N-layered earth, $\hat{Y}_1$ can be obtained by the following recurrence relation [152]:

$$\hat{Y}_n = Y_n \frac{\hat{Y}_{n+1} + Y_n \tanh(u_n t_n)}{Y_n + \hat{Y}_{n+1} \tanh(u_n t_n)}, n = 1, 2, ...N \quad , \tag{6}$$

where $Y_n = u_n/(i\omega\mu_n)$ and $\hat{Y}_N = Y_N$ in the n^{th} layer, t_n is the layer thickness and $\mu_n = \mu_0\mu_{rn}$ is the magnetic permeability of the n-th layer, and $u_n = (\lambda^2 + k_n^2)^{1/2}$. k_n is the wave number of the n-th layer given by

$$k_n = \sqrt{-i\omega\mu_n\sigma_n - \omega^2\varepsilon_0\mu_0\mu_n}, \tag{7}$$

where σ_n is the conductivity of the n^{th} layer, and ε_0 is the dielectric permittivity of free space.

The necessary Hankel transform integrals are realized using the filtering technique introduced by [4], which with some more recent improvements [3, 32, 114, 115] has turned out as an accurate and fast method which is nearly exclusively used in EM methods, though alternatives have recently been proposed [73].

3.1 Time-Domain EM methods

In principle, the time-domain response could be calculated by a direct numerical solution to the Maxwell equations. However, this is only rarely done in 1-D, as in this case it can be calculated more efficiently by solving the frequency domain problem as above for a predetermined, optimized set of frequencies. In the CSIRO codes this currently achieved with a predefined set of 67 frequencies, which seem to be sufficient for the problems encountered in AEM. It is clear from this fact, that the calculation of time-domain responses is considerably slower than in the frequency-domain case. The thus obtained solution is then transformed to the time domain and convolved with the system responses. Details can be found in [23].

Time-domain AEM has produced a large number of systems, which differ in their technical realisation and, consequently, in the necessary processing. Here, we are here referring to the TEMPEST system [82], assuming that the GENESIS system used in the Tellus projects is a modification of this system [71, 88], with less windows and a higher base frequency of 225 Hz. Figure 8 summarizes the basic principles of this TDEM System. The transmitter waveform is a series of positive and negative current pulses which by design are long enough to create secondary magnetic field in the ground. Current flows in the transmitter loop for a period of time (on-time) and is then abruptly terminated, and the rate change of the secondary field due to induced eddy currents is ideally recorded at the receiver coil during the off-time in order to minimize the primary transmitter field. In the case of TEMPEST/GENESIS, a full-time work

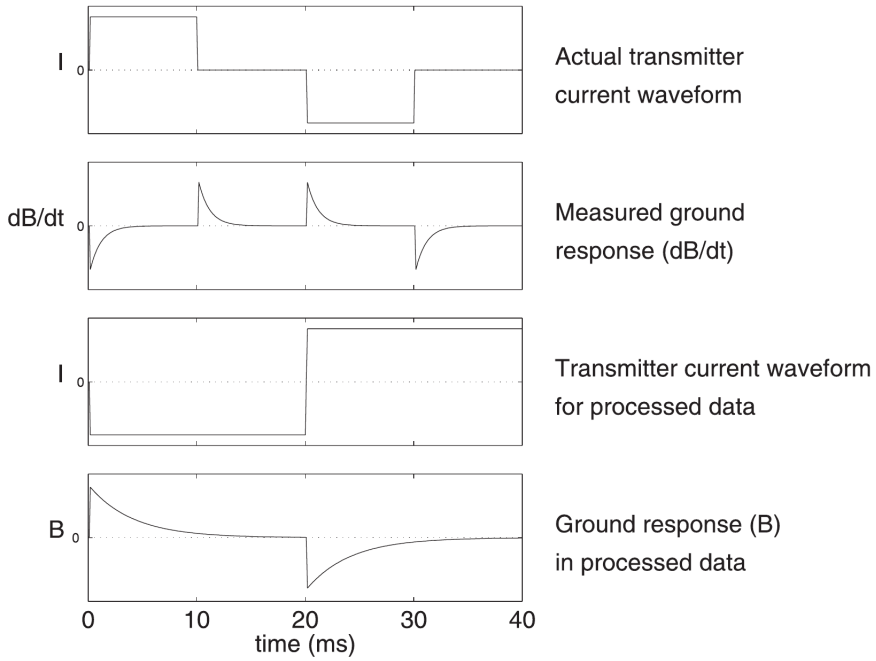

Figure 8: Schematic representation of the time-domain current waveform and the corresponding raw and processed receiver coil response for the TEMPEST system [82].

cycle is employed (Figure 8). Conversion from the measured $\delta B/\delta t$ data to a square-wave B-field response is applied during processing to improve the response characteristics.

Most scientists working with TEMPEST agree that a critical point for this type of system is the treatment of the primary field, which is needed to obtain the pure subsurface response and the normalisation. In contrast to the fixed-wing FDEM systems, the relative position of the receivers with respect to the transmitter is only approximately known. TEMPEST/GENESIS uses an approximate geometry and consequently primary field as described in [27]. This may be a serious source of error [21, 33]. However, no correction can be obtained for the Tellus survey, as some important parts of the required data are not available.

3.2 Deterministic inversion methods

Traditional inversion methods of the deterministic kind include (amongst others):

- Classical Tikhonov inversion can be used with constant or optimally chosen regularization parameters. Included are function-minimizing methods for estimation the optimal value of these parameters, as GCV, L-Curve, MLE, U-Curve and more.
- Tikhonov inversion can be combined with an cooling approach to find the best regularization parameter. This is often called Occam inversion, if realised as a variant of Morozov's principle.
- In order to avoid the smooth solutions produced by the former methods, several focussing variants have been proposed [83, 112, 158, 157].

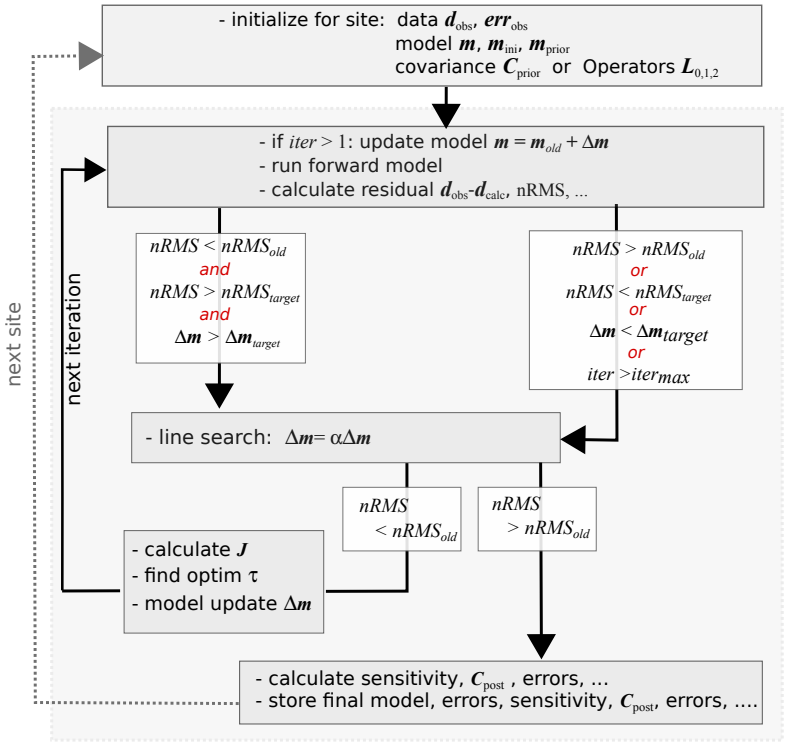

Figure 9: Flowchart for the `TikhOpt` inversion. Other implemented traditional inversion schemes look very similar. Further information can be found in the text.

- Maximum A-Posteriori (MAP) methods derive from a Bayesian approach [see, 142, 143, 120]. Here we have implemented both, data-space and parameter space versions.

For obtaining solutions in those deterministic approaches, an optimisation is employed, that is finding the minimum of

$$\theta = \mathcal{D}(\mathbf{d}, \mathbf{m}) + \mathcal{R}(\mathbf{m}), \tag{8}$$

with respect to $\mathbf{m}$. $\mathcal{D}$ measures the data fit, while $\mathcal{R}$ is called regulariser, and may contains one or more regularisation operator with corresponding weighting parameters $\boldsymbol{\tau}$, which need to be determined during optimisation process.

This framework is very general, and the result of this inversion depends strongly on the type of operator chosen for $\mathcal{D}$, $\mathcal{R}$ and the regularisation parameters. While the choice of $\mathcal{D}$ is usually a weighted Euclidean norm of the residual $\mathbf{r} = \mathbf{d} - \mathbf{g}(\mathbf{m})$, i.e., $\|\mathbf{r}\|_2^2$, other choices are possible which may increase the robustness of this approach to outlier data [64, 44]. In the toolbox described here, currently only the Euclidean approach is used, leading to

$$\mathcal{D} = (\mathbf{d} - \mathbf{g}(\mathbf{m}))^T \mathbf{W}_d^T \mathbf{W}_d (\mathbf{d} - \mathbf{g}(\mathbf{m})). \tag{9}$$

The most common types of operators $\mathcal{R}$ are the identity $\mathcal{I}$, or a discrete representation of

first or second order derivatives, each multiplied by an appropriate weighting τ, leading to

$$\mathcal{R} = \sum\nolimits_i \tau_i \left(\mathbf{m} - \mathbf{m}_a\right)^T \mathbf{W}_{m,i}^T \mathbf{W}_{m,i} \left(\mathbf{m} - \mathbf{m}_a\right), \tag{10}$$

where we have added a reference ("prior") model for convenience. For instance, in one dimension $W_{m,1}$ can be defined as

$$\mathbf{W}_{m,1} = \frac{1}{\Delta} \begin{bmatrix} -1 & 1 & & \cdots & & 0 \\ & -1 & 1 & & & \\ \vdots & & \ddots & & & \vdots \\ & & & -1 & 1 & \\ 0 & & \cdots & & -1 & 1 \end{bmatrix}. \tag{11}$$

The most straightforward way is by finding the minimum of θ as defined in Equation (8) by taking its derivative with respect to the parameters $\mathbf{m}$, and solving the corresponding normal equations implicitly or explicitly [e. g., 7], though many other algorithms are available. In this equation, $\mathbf{g}(\mathbf{m})$ represents the forward solution at the data points $\mathbf{d}$. The weighting $\mathbf{W}_d$ commonly is constructed from an estimation of the associated data error.

With these settings, and using the definition $\tilde{\mathbf{J}} = \mathbf{W}_d\mathbf{J}$, the common procedure of differentiation with respect to the model parameters and setting the result to zero leads to an iteration scheme

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \delta\mathbf{m}_k,$$

with

$$\delta\mathbf{m}_k = \left(\tilde{\mathbf{J}}^T\tilde{\mathbf{J}} + \sum_i \tau_i \mathbf{W}_{m,i}^T \mathbf{W}_{m,i}\right)^{-1} \left(\tilde{\mathbf{J}}^T \left[\mathbf{d} - \mathbf{g}(\mathbf{m}_k)\right] - \sum_{i=0}^{1} \tau_i \mathbf{W}_{m,i}^T \mathbf{W}_{m,i} \left[\mathbf{m}_k - \mathbf{m}_a\right]\right). \tag{12}$$

For convergence, it turns out that a change of the above iteration to

$$\mathbf{m}_{k+1} = \mathbf{m}_k + \mu\delta\mathbf{m}_k, \tag{13}$$

is crucial. A simplified line search uses a μ with $0 < \mu < 1$.

From the inversion process commonly the generalized inverse is calculated, which is defined as:

$$\mathbf{G}^\dagger = \left(\tilde{\mathbf{J}}^T\tilde{\mathbf{J}} + \tau\mathbf{W}^T\mathbf{W}\right)\tilde{\mathbf{J}}^T \quad , \tag{14}$$

which is often used to define uncertainty- and resolution-related parameters.

Within the paradigm of Bayesian inversion deterministic approaches have been developed. While depending on strong assumptions, they can produce useful results in much shorter computing times than sampling in parameter space. In [142] and [143] several flavours of these algorithms have been derived. The most common is the maximum a posteriori probability (MAP) estimate which leads to theoretically equivalent iterations:

$$\mathbf{m}_{k+1} = \mathbf{m}_a + \left(\mathbf{C}_m^{-1} + \mathbf{J}_k^T \mathbf{C}_d^{-1} \mathbf{J}_k\right)^{-1} \left[\mathbf{J}_k^T \mathbf{C}_d^{-1} \left(\mathbf{d} - \mathbf{g}(\mathbf{m}_k)\right) - \mathbf{C}_m^{-1} \left(\mathbf{m}_k - \mathbf{m}_a\right)\right] \tag{15}$$

$$\mathbf{m}_{k+1} = \mathbf{m}_a + \mathbf{C}_m \mathbf{J}_k^T \left(\mathbf{C}_d + \mathbf{J}_k \mathbf{C}_m \mathbf{J}_k^T\right)^{-1} \left[\mathbf{d} - \mathbf{g}(\mathbf{m}_k) + \mathbf{J}_m \left(\mathbf{m}_k - \mathbf{m}_a\right)\right] \tag{16}$$

$$\tag{17}$$

The former is usually called parameter-space, the later data-space formulation. A discussion of these iterations can be found in [120]. As above, a simplified line search as in Equation (13) improves convergence considerably.

Note that these iterations are very similar to the ones mentioned before (Equation 12). Here the prior covariances play a central role. The data covariance C_{ij}^d is commonly defined as diagonal, with the squared error (the variance) on the diagonal. This is not a necessary assumption, as data may not be mutually independent. The parameter covariance, C_{ij}^p plays a similar role as the differential operators in constraining the model search. Indeed, it has been shown that the inverses of some commonly used covariance matrices can be expressed by a weighted combination of differential operators [140, 141, 156]. In practice, the exponential (or Markovian) matrix is conveniently defined as $C_{ij}^{exp} \propto \exp -d_{ij}/L$, where i and j are parameter indices, d is their distance (in space or time),while L determines the associated correlation scale. The commonly used Exponential and Gaussian covariances can be unified in the Matérn form:

$$C_{ij}^{mat} \propto \frac{2^{\nu-1}}{\Gamma(\nu)} \left(\sqrt{2\nu}\frac{d_{ij}}{L}\right)^{\nu} K_\nu \left(\sqrt{2\nu}\frac{d_{ij}}{L}\right) \quad , \tag{18}$$

where ν and L determine the sharpness function. In this definition, Γ is the Gamma function, and K_ν is a modified Bessel function of the second kind. We have left out the leading factors σ_m^2, as this is usually unknown, and could be estimated during the inversion, e.g. by generalised cross-validation (GCV). The case of $\nu = 1/2$ will produce an exponential form, while a very large value (e.g., $\nu > 100$) leads to a Gaussian.

As a by-product of iterations 15 or 16, corresponding a-posteriori covariance matrices can be derived as:

$$\hat{\mathbf{C}}_m^p = \left(\mathbf{C}_m^{-1} + \mathbf{J}_k^T \mathbf{C}_d^{-1} \mathbf{J}_k\right)^{-1} \tag{19}$$

$$\hat{\mathbf{C}}_m^d = \mathbf{C}_m - \mathbf{C}_m \mathbf{J}_k^T \left(\mathbf{C}_d + \mathbf{J}_k \mathbf{C}_m \mathbf{J}_k^T\right)^{-1} \mathbf{J}_k \mathbf{C}_m \tag{20}$$

$$\tag{21}$$

respectively. They can in turn be used in defining the covariances used in a further MCMC simulation, though the assumptions made on priors and data errors should be taken into consideration. In this framework, the generalized inverse can be calculated as:

$$\mathbf{G}^\dagger = \left(\mathbf{J}^T \mathbf{C}_d^{-1/2} \mathbf{C}_d^{-1/2} \mathbf{J} + \mathbf{C}_m^{-1}\right) \mathbf{J}^T \mathbf{C}_d^{-1/2} \quad ., \tag{22}$$

Many uncertainty and resolution quantifiers are by-products of the inversion itself, as generalised inverses, data/parameter resolution, or covariance matrices. All of these methods depend strongly on the assumptions of error normality, and, being based on the Jacobian, are only valid near the current favourite model. They do not exclude models far away from this

reference. It should be noted, that though in sloppy talk often used quasi-synonymously, they are not. While the former is related to the question of how sure we are that a certain model is correct (i.e., images the truth), while the second would describe the spatial or temporal uncertainty of a particular parameter.

Detailed descriptions can be found in the referenced literature. We have also added some more ad-hoc methods for uncertainty or resolution favoured in the community. These include the simple determination of a Depth-of-Investigation (DoI) using different priors [105], and the direct use of (cumulative) sensitivities [130, 48, 56, 55, 131, 148], and Jackknife estimates of variance [40, 145, 125, 118]. Some remarks on the understanding of these indicators follow.

3.2 Sensitivity

The use of sensitivities (in a variety of flavours) is comparatively easy, but needs some clarification, as it does not really conform to the everyday use of the concept. Sensitivities are derived from the final model Jacobian matrix, which often is available from the inversion algorithm itself. It needs to be kept in mind that this implies any conclusions drawn are valid in the domain of validity for the Taylor expansion involved only. This may be a grave disadvantage in highly non-linear settings. However, it can be argued that the AEM problems are only mildly non-linear, as suggested by the very fast convergence of the different inversion algorithms.

The Jacobian of a (potentially transformed) data and parameter set is defined as

$$J_{ij} = \frac{\delta f_d(d_i)}{\delta f_m(m_j)} = \frac{\delta m_j}{\delta f_m(m_j)} \frac{\delta f_d(d_i)}{\delta d_i} \cdot \frac{\delta d_i}{\delta m_j}. \tag{23}$$

This Jacobian is first normalized with the (transformed) data error (i. e., multiplied by $\mathbf{C}_d^{-1/2}$) to obtain $\tilde{\mathbf{J}}$. This is not controversial, particularly as in TDEM errors change considerable with time gate and component.

The definition of sensitivity is not unique, and various forms can be found in the literature, and `AEMpyX` calculates several of them, shown in Figure 10):

- Raw sensitivities, defined as

$$S_j = \sum_{i=1}^{n_d} \tilde{J}_{ij} \quad , \tag{24}$$

  can be used. No absolute value is involved, hence there will be both, positive and negative, values. This does not conform to what we expect of sensitivity, and makes it unusable for shading or blanking model plots. It carries, of course the most direct information on the role of parameter j in the inversion.

- The commonly used form is defined as

$$\mathbf{S}_j^2 = \sum_{i=1}^{n_d} \left|\tilde{J}_{ij}\right|^2 = diag\left(\tilde{\mathbf{J}}^T \tilde{\mathbf{J}}\right) \quad . \tag{25}$$

  This solves the positivity issue of raw sensitivities. The square root of this sensitivity is preferred in most cases [55, 36] and implemented in many popular inversion codes [e. g., 14, 15]. Also in use is the form $\mathbf{S}_j = \sum_{i=1}^{n_d} \left|\tilde{J}_{ij}\right|$, sometimes called coverage [55].

- A choice more appropriate for 1-D models can be the cumulative sensitivity, as suggested by Vest Christiansen and Auken [148]. They propose summing up the layer sensitivities from below, as

$$S_j^{cum} = \sum_{k=N_l,-1}^{j} S_j \quad . \tag{26}$$

It can easily be seen that the first two forms may lead to possibly unwanted effects if used for a definition of any depth of investigation (DoI). The cumulative sensitivity is monotonous decreasing with depth by construction. This entails that the maximum value of sensitivity is always at the surface, which is not guaranteed form other forms. While this is appropriate for a DoI definition, there is unfortunately no obvious way to generalise this for future multidimensional models. In the figures shown later in this report, we have always used the square root of the second form.

When moving from the error-normalised Jacobian $\mathbf{J}_d$ to sensitivity, there are again several choices, depending on the understanding and use: If sensitivity is to be interpreted as an approximation to a continuous field over the volume of the model, which is independent of the discretisation, it seems useful to be normalized by the layer thickness [130, 131]. On the other hand, effect of the "cell size" (here layer thickness) is important when investigating the true role of this layer in the inversion. Finally, for comparing different data (sub)sets, it is convenient to do a final normalization by the maximum value in the model. All these options are implemented in the `AEMpyX` toolbox. An example for the different options is given in Figure 10.

When interpreting sensitivities, it has to be kept in mind that this is a linear approximation near the favourite model $\mathbf{m}_0$. Sensitivity being small means that while small perturbances in $\mathbf{m}$ will produce small changes in the response, equivalent models far away from $\mathbf{m}_0$ may exist, making it a sometimes questionable measure of uncertainty.

3.2 Posterior covariances

The posterior model covariances can be used to characterise the uncertainty of models. The square root of their diagonal elements will be linearly related to the a-posteriori parameter standard deviations. With the usual assumptions of normality, these are

$$\sigma_i \approxeq nRMS\sqrt{\hat{\mathbf{C}}_{ii,m}} \quad , \tag{27}$$

and are often plotted with the corresponding models. These errors, however, are in practice often misleading. One possible reason for this lies in the method of inversion. Posterior covariances are heavily influenced by the value(s) of the regularisation weights. Searching for the smallest value possible while keeping the solution stable is at the base of several methods mentioned above, but in practice often produce too small error estimates. An example for a full covariance matrices can be found in Section 5.4.

3.2 Resolution matrices

The parameter and data resolution matrices are usually defined as

$$\mathbf{R}_m = \mathbf{G}^{\dagger}\tilde{\mathbf{J}} \quad \text{and} \quad \mathbf{R}_d = \tilde{\mathbf{J}}\mathbf{G}^{\dagger}, \tag{28}$$

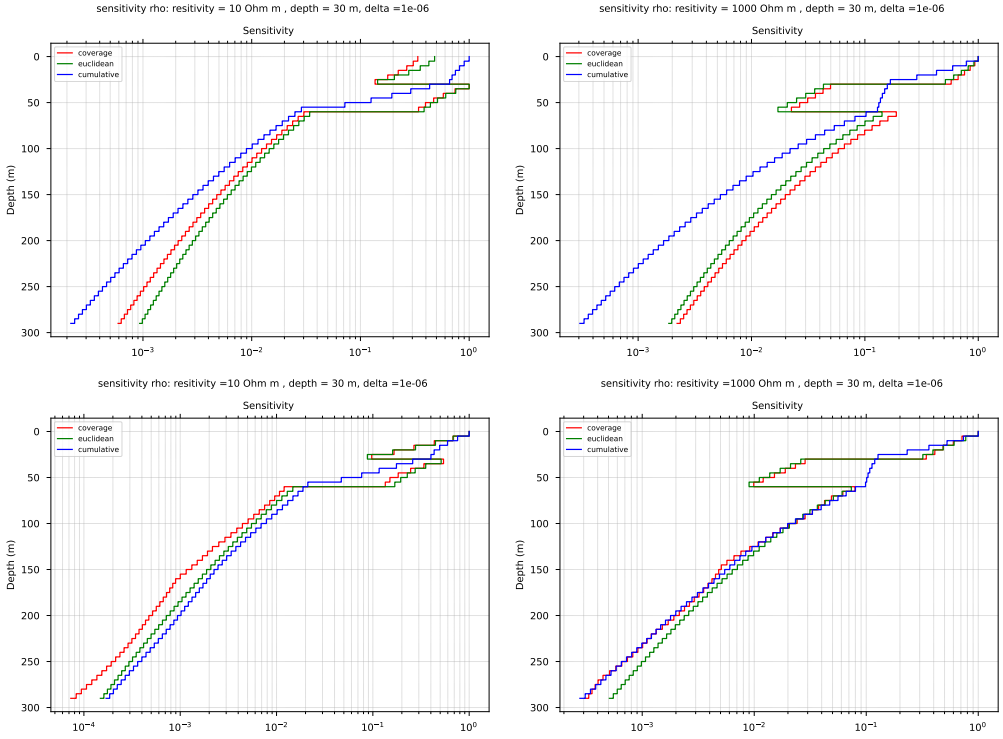

Figure 10: Different forms of sensitivity as described in the text. Top row: GENESIS TD system. Bottom row: AEM05 FD system. Left column: Conductor with resistivity of 10 Ωm, at a depth of 30 m, and a thickness of 30 m; Right column: Same geometry for a resistor of 000 Ωm. It can easily be seen that the first two forms may lead to possibly unwanted effects if used for a definition of a depth of investigation. The cumulative sensitivity is monotonous decreasing with depth by construction. This entails that the maximum value of sensitivity is always at the surface, which is not guaranteed form other forms.

respectively. Several textbooks and articles discuss their interpretation [108, 48, 94, 7, 2, 144]. The basic idea is that they establish linear relations between the estimated data and parameters to the "true" ones, $\mathbf{m}_{est} \approx \mathbf{R}_m \mathbf{m}_{true}$. Note that both resolution matrices do not depend on the data themselves, only their errors. Figure **??** in tutorial Section 5.4 shows an example for a simple three-layer model (FD and TD).

In the case of a non-regularized inversion, both of these are unit matrices, however, in the case of ill-posed problems as common in geophysics, the inversion will diverge, or lead to unphysical model. Thus, without a regularization or prior covariance term, they are useless. Resolution matrices will only give information of the whole inversion, including the chosen regularisation weigh. Once accepting this caveat, relevant information can be extracted from them. Note that in contrast to the already mentioned posterior covariances, they are not required to be symmetric.

$\mathbf{R}_d$ shows the spread of information used in the inversion process, i.e., gives an indication of the importance and correlation of data. $\mathbf{R}_m$ can be used to further characterise the inversion results. Detailed accounts are given in some recent publications [2, 144]. As it is cumbersome to show these matrices for all sites, one needs to condensate the information. For this purpose,

Figure 11: Here we show inversion results for an ensemble (N_e= 10000) for a three-layer model, embedding a conductor (C) within a more resistive half space (B). As in the following figures, the left column represents FD (AEM05) inversions, the right TD (GENESIS) results. The model always contains 32 layers (incl. underlying half space). Shown are percentiles (10, 20,... 50,...90 %) for the ensemble inversions. More information in the text.

Figure 12: Inversion results for the same ensemble for a three-layer model, embedding a resistor (R) within a more conductive halfspace (B).

both, columns and rows for a particular parameter (of interest) can be plotted as point spread functions (PSF) or smoothing kernels (SK). Further details are given [144]. However, the most commonly used parameter is directly derived from the diagonals as sum of all diagonal elements, $N_p = \sum_{i=1,M} R_{m,ii}$. This quantity will give an estimate for the number of independently determinable parameters.

An example for full parameter and data resolution matrices for moth Tellus sytems can be found in Section 5.4.

3.3 Ensemble methods

We use the concept of an ensemble in a sloppy way. The methods described her span a range between "hybrid" methods (as the Jackknife or RTO methods), which build on deterministic methods, and the fully Bayesian McMC methods.

In order to get a feeling of what information can be extracted from AEM data, studies on simple, few-layer models, comparable to the ones shown by Bedrosian et al. [12] are instructive. Here we show only few examples, while more numerical of this kind can be found in Bayat [10]. In all cases shown here, ensembles (N=1000) of perturbed synthetic data. The examples shown here are three-layer models, embedding a conductor (C) or resistor (R) within a more moderately resistive half space (B). This simple approach is not an inversion, but is still usefull when starting to set up inversions.

3.3 Randomise-then-Optimise

This method (RTO) is a technique quite similar to Randomised Maximum Likelihood (RML) [106]. It has been first proposed by Bardsley et al. [8], further developed and applied by several authors [9, 16, 17]. It is particularly attractive as it is non-intrusive in the sense that in only needs the output of a given inversion procedure. Thus, it can be easily combined with any inverse software, even if the source code is not available. This feature is invaluable for comparison of methods, and even joint inversions in the future. Furthermore, the inner runs of the inversions

Figure 13: Inversion results for the same ensemble for a three-layer model, embedding a resistor (R) within a more conductive halfspace (B).

Figure 14: McMC inversion results for ensembles of three-layer models. Top row: BCB. Bottom row: BRB.

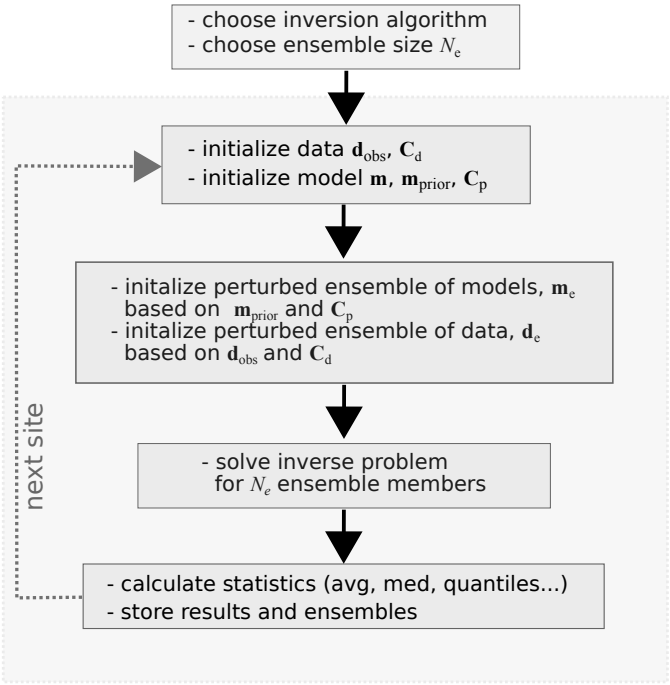

Figure 15: Schematic diagram of the Randomize-then-Optimize algorithm. Explanations in the text.

are independent, in the sense that they do not require any interaction. This allows not only to exploit parallelism, but also to enlarge the ensemble later if required. The final statistics can be made off-line.

3.3 Jackknife estimates

The Jackknife estimate is an comparatively old approach [113, 147], and subsequently further developed in [42, 40, 41]. It was introduced to geophysics by Tichelaar and Ruff [145] as a simple resampling estimate, and has been applied successfully in practice many times [103, 117, 122, 126, 127]. In our context this means that after having inverted the full data set, an ensemble is generated by leaving out one data point, $\mathbf{d}_i = [d_1, \ldots, d_{i-1}, d_{i+1} \ldots d_N]$. Then, the inverse problem is solved for all ensemble members, $\mathbf{m}_i = \mathbf{g}(\mathbf{d}_i)$. Pseudo-models can be defined as linear combinations $\tilde{\mathbf{m}}_i = N\mathbf{m}_{full} - (N-1)\mathbf{m}_i$. Then the jackknife estimate of the models is expressed as

$$\mathbf{m}_{jcn} = \frac{1}{N}\sum_{i=1}^{N} \tilde{\mathbf{m}}_i \quad , \tag{29}$$

with the variance

$$\mathbf{var}_{jcn} = \frac{\sum_{i=1}^{N} \tilde{\mathbf{m}}_i^2 - 1/N \sum_{i=1}^{N} (\tilde{\mathbf{m}}_i)^2}{N(N-1)} \quad . \tag{30}$$

3.3 Ensemble Kalman inversion

EKI can be interpreted as an derivative-free optimiser, as no Jacobian calculations are necessary. In many cases Jacobians are not easily accessible, and even Jacobian-vector-products, usually calculated via adjoints are not available. This may not only be advantageous with respect to computational resources when models get multidimensional and large, but also when coupling different inversions, as when multi-physics is required, or spatially constraints. We are aware that the case of AEM inversion as implemented here is not challenging enough to expose these advantages, though this may become te case in the future.

EKI borrows ideas from the (immensely successful) Ensemble Kalman Filter (EnKF), [43], replacing deterministic covariance estimates by their ensemble-based counterparts. Assuming we have a mode ensemble of size N_e which is described by its ensemble mean $\bar{\mathbf{m}}$ and prior covariance $\mathbf{C}_m^{prior}$, and the corresponding ensemble of responses $\mathbf{g}_i = \mathbf{g}(\mathbf{m}_i)$ with mean $\bar{\mathbf{g}}(\mathbf{m})$ the following ensemble-based covariances can be defined :

$$\mathbf{G}^{mm} = \frac{1}{N_e} \sum_{i=1}^{N_e} (\mathbf{m}_i - \bar{\mathbf{m}}) (\mathbf{m}_i - \bar{\mathbf{m}})^T \tag{31}$$

$$\mathbf{G}^{gm} = \frac{1}{N_e} \sum_{i=1}^{N_e} (\mathbf{g}_i - \bar{\mathbf{g}}) (\mathbf{m}_i - \bar{\mathbf{m}})^T \tag{32}$$

$$\mathbf{G}^{gg} = \frac{1}{N_e} \sum_{i=1}^{N_e} (\mathbf{g}_i - \bar{\mathbf{g}}) (\mathbf{g}_i - \bar{\mathbf{g}})^T \tag{33}$$

$$\tag{34}$$

Starting from the data space update given in Equation 16, the EKI algorithm can be derived in the following way. iterative ensemble update equation [67, 30, 43]:

$$\mathbf{m}_i^{k+1} = \mathbf{m}_i^k + \mathbf{G}^{gm} (\mathbf{G}^{gg} + \mathbf{H})^{-1} (\mathbf{d} - \mathbf{g}_i)^k \quad , \tag{35}$$

where the covariances a understood to be evaluated at iteration k. While looking similar to Equation 16, it operates on an ensemble of size N_{ens}, with i representing the ensemble member.

The term $\mathbf{H}$ needs further explanation. In the simplest, unregularised approach $\mathbf{H} = \mathbf{C}_d$

Here we have implemented only one of the basic flavours of this method. Many different algorithms have ben proposed, including explicit regularisation [66, 30, 31, 65, 87], and multiscale approaches [e. g., 30, 74]. While some interesting field studies have been published [e. g., 101, 146], none of the many variants has acquired production status. Many theoretical and practical questions are open (e. g., convergence and stopping criteria), in particular with non-linear problems [29].

3.3 McMC methods

McMC methods are supposed to sample the full parameter space, though in practice this may not be achieved. Since their introduction into computational statistics [95], there have been many further developments, driven by the interplay of the enormous requirements of computational resources ("curse of dimensionality"), and the fast development of computer architectures and algorithms [24, 77, 50, 51].

Though our original straightforward implementation of the Metropolis-Hastings scheme is still available, it is not recommended for use in production. We have created interfaces to existing

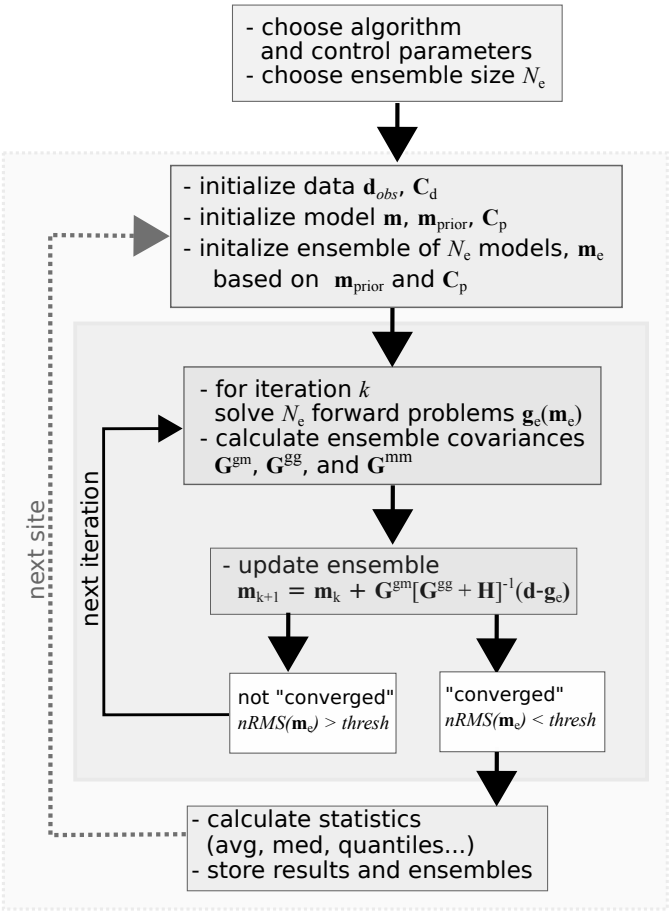

Figure 16: Schematic diagram of the Ensemble Kalman Inversion. Explanations in the text.

highly sophisticated open-source libraries, as `PyMC` [93], `emcee` [47], or `zeus` [69]. They include many flavours of McMC methods, and are already developed for parallel processing options.

4 The `AEMpyX` toolbox

We implemented an easy-to-use and highly flexible toolbox using the Python language for the 1-D inversion of frequency- and time-domain AEM data. The computational core is based on an adapted forward model derived from the well-tested open-source code AirBeo. This code was originally developed by Australia's CSIRO and the AMIRA consortium, and the latest version is available from `https://sourceforge.net/projects/p223suite`.

The toolbox is implemented are implemented as high-level scripts including a full work flow from loading of raw data over pre-processing to inversion. They rely on functions implemented in callable modules, containing functions providing numerical procedures, constraint methods, processing tools, and utilities. Here, we give an overview of the typical work flow for an inversion which is constructed from these building blocks. The idea behind this approach is to be able to combine the existing functions for fast prototyping new and improved work flows. This can be on

the level of the inversion algorithm, but also for designing more complicated procedures, such as using the output of MAP inversion for the prior in MH simulations, or a two-stage procedure for obtaining susceptibility using the results from prior resistivity-only inversion. The setup for these inversions is highly flexible, as activity indicator matrices are used to choose the data to be used for inversion, and more important, the relevant parameters can be determined likewise. The pre-processing functions include the masking of non-physical data, interpolation, PCA filtering of data [97]. Inversion algorithm can be combined with different type of couplings (covariance matrices in MH or differential operators in Tikhonov inversions), or regularisation. In contrast to the `aempy` toolbox, now data and parameter transformations are allowed. which includes the use of different parametrisations, in particular in the modelling of induced polarisation effects. Utilities include data managing tools, such as reformatting raw data to an internal format, data choice from polygons or their intersection/unions, profile projection, and various visualisation methods.

The `AEMpyX` ("aempy-extended") toolbox is written in python 3.9+ (currently developed under 3.11) with an adapted computational core which was extracted from previously mentioned BeoAir code, written in Fortran 90. In the following we describe the main steps for the use of `AEMpyX` with the anaconda package under Ubuntu Linux and Windows 10. The move from Python 2.7 (which is no longer maintained since 2021) to Python 3.9 required a complete overhaul of the software, and allowed us an easier inclusion of the methods mentioned above.

We have not chosen to implement a single call for calculating the forward model Jacobian, but we have encapsulated the system-specific settings within the Fortran core modules. It seemed to be advantageous, as in practice, you rarely mix many different types of data. As the full capabilities of the BeoAir core are still active, additional wrappers can easily be implemented by changing a few lines in the corresponding AEM05 or GENESIS drivers. This approach also makes the calling of these routines much simpler.

The scripts described here can be run in different ways. While developing, it is advantageous to run the code from within the `spyder` IDE because of the debugging abilities and the integrated graphics preview. However, they also can be run from the command line or shell script by entering `python script.py`, or, even easier, by `./script.py`. The last variant requires that the executable script contains the corresponding interpreter directive (see below, options are possible with the `sys.getopt` command within the script) in the first line.

4.1 Install Python

Though it is in principle possible to work within the system python, we strongly advise to install and use the freely available open-source anaconda package, which is self-contained, and comes with flexible and easy-to-use administration tools. Note that we recommend anaconda not only because of the easy management and high-performance implementation, but also because the self-contained environment provided by this package does not interfere with the system-provided python (on which many other packages depend), which is a common source of unexpected problems for less experienced users. Installers for Linux and Windows are available at `https://www.anaconda.com/products/distribution`, or, in case you want to work with the alternative mamba-based distributions, at `https://github.com/conda-forge/miniforge`.

In this text, we describe our current minimum setup for python, however, as mentioned in the introduction, many other packages will be useful when working with the `AEMpyX` tools in practice. From the link above, download the latest Miniconda (or Mambaforge) version, and install by running the downloaded bash script (or executable in Windows). Check whether this version is

for python 3.9, as some tools are still missing from 3.10. For instance

```
> bash Miniconda3-latest-Linux-x86_64.sh
```

will install the necessary python distribution under Linux. We suggest installing it in user space, i.e., keep the standard settings, and agree to put this path into the `.bashrc` initialisation file. In case you do not want to be within a miniconda environment by default, you should copy the initialisation bock created by the installation within `.bashrc` to a stand-alone bash script. similar to the one in the `environment` directory within `AEMpyX`. Under Windows, execute the installer executable and follow the instructions.

After successful installation, `conda` should be updated to the most recent version of the distribution. This will be achieved by entering the following from the base environment (Linux) or the Anaconda prompt (Windows):

```
conda update conda
conda update --all
```

The current version is being developed under Python 3.11. We also suggest to install the `mamba` replacement for the `conda` tool, namely:

```
conda install mamba -c conda-forge
mamba update --all
```

4.2 Install AEMpyX

Go to an appropriate place, and get your working copy via `git` from the command line:

```
git clone https://github.com/RAFTA-AIRBORNE/AEMpyX/
```

Alternatively, the package can be downloaded from `https://github.com/RAFTA-AIRBORNE/AEMpyX` as `AEMpyX.zip` or `AEMpyX.tgz`, which should be unpacked in an appropriate place. However, the zip file will not include the git information necessary for easy update via the git version control system. The most current git manual can be found in the info subdirectory of the toolbox tree, or downloaded from `https://git-scm.com/book/en/v2`.

The resulting AEMPyX repository (or directory tree) contains the subdirectories `aempy`, and `environment`. While the former contains the toolbox itself, in the latter some useful helper files for working within the `conda` environment can be found. In particular, the `conda` environment description files `AEMpyX.yml` and `AEMpyX.txt` are found here, useful to set up a working environment for `AEMpyX`. The current environments contain a lot of packages which are not strictly necessary for running the inversion software, but useful for related geoscientific work. There will be will be a streamlined `AEMpyX` environment (Python 3.9) at a later time.

In `aempy`, the following subdirectories are found:

aempy/core1d: This directory contains the Fortran 90 source code for the computational core run by the Python toolbox. Currently it contains working wrappers for the two systems used in Tellus: AEM05 and GENESIS. The numerics is derived from the AMIRA/CSIRO AirBeo software. Wrappers for GEOTEM and TEMPEST are work in progress.

aempy/modules: Contains the modules `aesys.py`(system-specific code), `prep.py` (data preparation), `post.py` (post-processing), `(e)viz.py` (visualisation), `inverse.py` (general inversion-related procedures), `alg.py` (high-level algorithms), `ensemble.py` (ensemble-related), and `util.py` (general utilities), which are called from the Python scripts in aempy/tutorial, aempy/scripts and the notebooks in aempy/notebooks, accomplishing different tasks of AEM inversion.

aempy/tutorial: Contains tutorial scripts for preprocessing, visualization, and one-dimensional inversion, and uncertainty analysis of AEM data. The scripts here should cover a full workflow, from ingesting the data to uncertainty analysis.

aempy/scripts: Contains a collection of further scripts for manipulating, preprocessing, visualization, and one-dimensional inversion of AEM data. Also included is a workflow for static shift correction of MT observations (work in progress). These scripts can be used as templates for a large number tasks related to practical AEM. The user should be aware, that these are scripts written for some special task at a certain state of the toolbox, and thus will not be at the most recent state. We include them at the convenience of the user.

aempy/notebooks: Contains `jupyter` notebooks for some typical work flows in preprocessing, visualization, one-dimensional inversion, including uncertainty quantification of AEM data.

aempy/test: Contains scripts for testing various aspects of `aempy`.

aempy/util: Contains useful non-python helper shell scripts etc.

aempy/manual: LaTeX sources for the toolbox's documentation, as well as publications and slides related to it.

aempy/info: This directory contains some useful documentation for python, including the most important extensions, numpy, scipy, and matplotlib.

The following steps are required for the installation of the toolbox:

1. Create an appropriate `conda` environment (including the necessary prerequisites) from the linux or windows version of the files `AEMpyX.yml` or `AEMpyX.txt` found in the `environment` directory by:

```
conda env create -f AEMpyX.yml
```

or:

```
conda create --name AEMpyX --file AEMpyX.txt
```

This will set up a Python 3.9 environment with all dependencies for `AEMpyX`. In case you want to use a different python version, corresponding files are still available. Don't forget to update also `AEMpyX` regularly, using:

```
conda update --name AEMpyX --all
```

Activate this environment by:

```
conda activate AEMpyX
```

2. Under Linux,once you have checked that `gfortran` exists in the environment, go to the `aempy/core1d/` directory, where a file called `Makefile_linux` should be found. It includes all the commands for the compilation of the Fortran component of the toolbox. The compilation and installation of the resulting library is done by entering

```
make -f Makefile_linux
```

in the command line. This will result in a dynamic library file `core1d.so` in the correct subdirectory, `aempy/modules/`. This directory contains the modules and external libraries necessary for running the script from the `aempy/scripts/` or any other place in the system. Under Windows, open the Anaconda command window, activate `AEMpyX`, and enter

```
make -f Makefile_windows
```

3. Currently we have defined two environmental variables, `AEMPYX_ROOT` and `AEMPYX_DATA`. These need to be set in your `.bashrc` file pointing to the place where `AEMpyX` is installed, and where you keep your AEM data, respectively. Keeping to this scheme makes life much easier if more than one person work on the tools. Example:

```
export AEMPYX_ROOT='\${HOME}/AEMpyX/'
export AEMPYX_DATA='\${HOME}/AEM_Data/Tellus/data/'
```

4. In order to make updates secure and avoid inconsistencies, copy `.condarc` from the `environment` directory to your home. For having identical behaviour of matplotlib, you should also copy the included `matplotlibrc` file to the appropriate directory. Under Linux (Ubuntu), this should be `$HOME/.config/matplotlib/matplotlibrc`. Pertinent changes should be made there, or have to be made within the scripts/modules using the `mpl.rcParams[name]=value` mechanism.

5. Finally, the remaining toolboxes you want to use within the `AEMpyX` environment need to be installed, either via the `conda` framework, or the `pip` command, if not in the `conda` ecosystem. Use of `pip` is discouraged, as it may lead to inconsistencies of the environment.

4.3 Running AEMpyX scripts

Once in the activated conda environment `AEMpyX`, there are several ways to start python scripts or jupyter notebooks `https://jupyter.org/`.

For getting started with python **scripts**, we suggest to use the `spyder` IDE `https://www.spyder-ide.org/`, which is already installed within `AEMpyX`. It has been developed for easy development of python software, including visualisation with `matplotlib` and derived packages. Current versions (5.X) also allow to work with JULIA, R, or other languages, or with jupyter notebooks by installing the appropriate plug-in.

However, as we have defined the environmental variables above, the scripts can be run from the command line anywhere in the system, either from the activated `AEMpyX` environment, or using the command `conda run -n AEMpyX` (see,`https://docs.conda.io/projects/conda/en/latest/commands/run.html` from an initialized conda (i.e., base environment). Running python scripts directly from the command line is done by several calls:

```
conda run -n AEMpy mypythonscript.py
python -u mypythonscript.py > output.log
mypythonscript.py > output.log
```

The last call can be used, if the magic first line "`#! python`" exists in the script.

The usual way to work with **notebooks** is connecting to your favourite web browser, or specialized software as `jupyterlab`. If using your browser (as set in your system as default), the calls are:

```
jupyter notebook mynoytebook.ipynb
jupyter lab mynoytebook.ipynb
```

Both calls will open a new browser window, in which you can edit and run the notebook. However, `jupyterlab` is the future interface for notebooks, and there are many options here not present in the classical notebook interface (`https://jupyterlab.readthedocs.io`). Remote clusters often offer a specialized jupyter server (`JupyterHub`) to develop and run notebooks.

5 Tutorial: Introduction to the `AEMpyX` workflow scripts

Here we will give a walk-through of the new `AEMpyX` toolbox, following a typical work flow from the raw data (as received from the Tellus working group), via preprocessing, inversion, to uncertainty quantification. The general structure of a full workflow is summarised in Figure 17. It must be stressed, however, that this is only one possible workflow, and the actual procedure can be adapted to the needs of a particular project, e.g., into one single script.

The typical approach for uncertainty-enabled inversions would be the following: Keeping in mind the vast amount of data, it still seems impracticable to use a fully Bayesian treatment, except in limited data sets, e.g., relevant for a particular geoscientific project. One could also reduce the highly redundant data along a flight line by decimating by an appropriate factor, accepting the corresponding loss of in-line resolution. The deterministic inversions mentioned above will, however, produce parameters relevant for the evaluation of results, as a-posteriori

covariances, parameter errors, resolution matrices, or spread functions. New algorithms introduced in *AEMpyX* alleviate the problem of high numbers of data, but still require very high computing times.

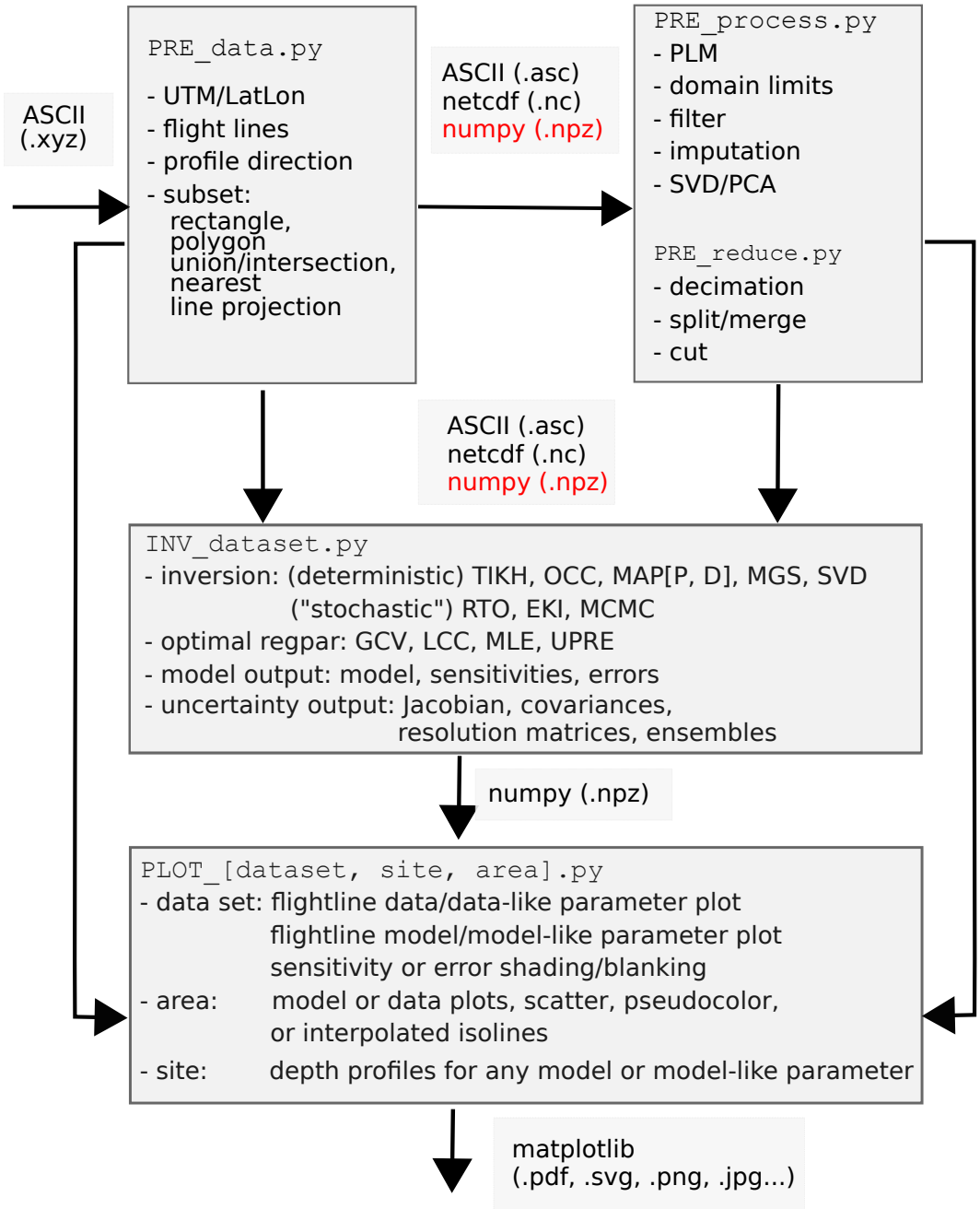

Figure 17: This diagram shows a simplified `AEMpyX` workflow. Detailed information can be found in the text.

All scripts begin with header lines that are required to load the necessary modules from the python standard libraries, and the aforementioned numerical modules, and `AEMpyX` modules proper. The following header is common to most scripts in the toolbox.

```
# System modules.
import os
import sys
from sys import exit as error
from datetime import datetime
import warnings
# Modules for numerical computing.
import numpy
import scipy
import scipy.linalg
import scipy.signal
# Matplotlib and other visualisation modules
import matplotlib.pyplot
import matplotlib
# AEMPYX modules
AEMPYX_ROOT = os.environ["AEMPYX_ROOT"]
mypath = [AEMPYX_ROOT+"/aempy/modules/", AEMPYX_ROOT+"/aempy/scripts/"]
for pth in mypath:
     if pth not in sys.path:
     sys.path.insert(0,pth)
from version import versionstrg
import util
import aesys
import prep
import core1d
import inverse
import alg
```

Here, we have fully imported numpy and the other modules, allowing any callable procedure or class referring to 'numpy.function'. Alternatively, any procedure `func` needed in the script could be imported individually with the `from numpy import func1, func2, ...` syntax, which can make the scripts more readable, though may make it difficult to locate the functions, in particular if they are within some submodules. Within the individual scripts, the imported modules may differ, and if required, other modules may be added to the respective blocks.

For importing the `AEMpyX` modules, we use two environmental variables, as already mentioned in 4.2. This appeared as an easy and transparent way to make the scripts callable from everywhere within the system, once the corresponding environment is loaded, e.g., in `.bashrc` or any startup script. This is particularly useful when it is necessary to set up special versions associated with particular data sets. For reproducibility reasons, it is good practice to store the adapted scripts together with the data.

A block shared by most scripts is the definition of parameters related to the system, and the task of the script. A minimum would be a call to `aesys.get_system_params(System=AEM_system)`. Here some of the output is replaced by `_`, which is a common convention for python dummy variables.

```
AEM_system = "aem05"
if "aem05" in AEM_system.lower():
    FwdCall,NN, _, _, _, = aesys.get_system_params(System=AEM_system)
    nL = NN[0]
    ### add pertinent parameter settings for scriptand system ####

if "genesis" in AEM_system.lower():
    FwdCall,NN, _, _, _, = aesys.get_system_params(System=AEM_system)
    nL = NN[0]
### add pertinent parameter settings for script and system ####
```

The full set of possible output parameters includes:

- `FwdCall`. A string defining the forward modelling call, giving a high degree of freedom for the integration of new code.
- `NN`. A list containing the lengths of different parameter sets in the internal format for this system. It is defined as `[line, meta, data,other]`, where meta includes flight line, position (UTM), height asl, radar altitude, and DTM, data is the length of the data set, and other could be a power line monitor where available, or other available quality related information.
- `Fmt`. Format string for the (legacy) ASCII data.
- `Col`. List of strings containing the column names for data.
- `MiscPars`: List of other objects which are necessary or useful, as arrays of frequencies or windows, waveforms, scale factors, and physical units.

In contrast to parameters specifying model discretisation, priors, or similar, pertinent parameters could be any changing rarely in production, such as data and model transforms, (de)activated data, error models, or similar data- or task-specific settings.

In the following, we will describe the most important steps leading from (1) the ingestion of data, via (2) preparation of the data sets (most often flight lines); (3) inversion of the data; to (4) uncertainty analysis of the data. Though we believe it is good practice to split the workflow as described below, where it is possible to store intermediate results with different settings, nothing forbids using the required steps within only one script, which is left to the user.

5.1 Ingesting data

Survey data will be read and transformed to the internal storage format by the tutorial script `Tutorial1_PRE_data.py`, or the corresponding jupyter notebook `Tutorial1_PRE_data.ipynb`.

`AEMpyX` can currently work with the following systems: *AEM05*, *AEM95*, and *GENESIS*. In addition, the data from different surveys come in different formats, including additional information, as the Power Line Monitor (PLM) channel, or other parameters of interest. Thus we have tried to make the work with different systems/surveys easier. We define two control variables `AEM_system` and `AEM_Survey`, which determine the setup for specific case.

The system setup (specified by `AEM_system`) looks as following, shown for *AEM05* here:

```
AEM_system = "aem05"
FwdCall, NN, Fmt, Col, MiscPars = aesys.get_system_params(AEM_system)
```

`FwdCall` is a string which determines the call to forward modelling, which is executed in the modules/scripts as `data_calc = eval(fwdcall)`, thus allowing for passing additional parameters to the Fortran core. `NN` contains the structure of the `Data` block as `NN =[n_meta+n_data+n_option, n_meta, n_data, n_option]`, where the elements give the column numbers in the full data record. The following output parameters are usually not used, and thus set to "`_`".

A specific survey specified through `AEM_survey` is read by the call:

```
Data, Header, AEM_system = aesys.read_survey_data(DatFile=File,
Survey="A5", Missing = "1.e30",
EPSG_in=2157, EPSG_out=32629)
```

The script `Tutorial1_PRE_data.py` will read the input file in XYZ format as generated from the (former) Geosoft products (see `https://www.seequent.com/`). The structure is defined in the `Survey` block, and write an output file in the internal`aempy` format, where unused information has been stripped out. It is straightforward to generalise it for other data. In this case, the input data contain the position in the Irenet 95 system (EPSG 2157), but is transformed to UTM 29N (EPSG 32629). It is desirable to use the regular UTM system, as this standard with GPS and can easily be used with google-earth and other software. In `AEMpyX`, internally UTM 29N is used.

As it has turned out that in practical projects only subsets of all data are necessary, we have included several methods of extracting spatial subsets: rectangles (defined by lower left and upper right corner points), circles (center and radius), and polygons (including their union or intersection).

```
Corners = ([638968.67,641519.17, 5922331.93, 5924940.46]),
Data_rect = util.extract_data_rect(Data, Corners)

Polygon = [[x1,y1],[x2,y2]...]
Data_poly = util.extract_data_polygon(Data, Polygon, method="shp")
```

In the polygon case, first and last points will be connected for closure. Other method options are using the convex hull ("con") or the envelope ("env"). Polygonal choices allow, e.g., the construction of pseudo-profiles in any direction by projection of these data onto a virtual flight line. An important case is the modification of polygons:

```
Polygon = util.modify_polygon([Polygon1, Polygon2], Operator="union")
Data_union = util.extract_data_poly(Data, Polygon, method="shp")

Polygon = util.modify_polygon([Polygon1, Polygon2], Operator="inter")
Data_intersection = util.extract_data_poly(Data, Polygon, method="shp")
```

A final option is the rotation of all profiles into the same direction, making their optical comparison easier. For Tellus the approximate angle is 345°. This means that all profiles start in the SE:

```
CorrectDirection = True
Ang, Spread = 345., 5.
```

The script can produce an output containing all data, and/or separate files for each flight line, which will be written by:

```
aesys.write_aempy(File=f, Data=Data, System=AEM_system, Header=Head, OutInfo=OutInfo)
```

to external files. Format in this case is chosen by file extension (".npz"), but can also be set explicitly by adding `Format=".npz"` to the call. Though options to write the data in text (".asc", ASCII), or netcdf/hdf5 (".nc") files exist, we suggest to use numpy's NPZ format `https://numpy.org/doc/stable/reference/generated/numpy.savez_compressed.html`. It combines the low storage (internal compression), portability between operating systems, with the easy manipulation of a python dictionary. This means that it is easy to extract, overwrite, delete or add new fields as considered be useful during the processing. In the future, it will be useful to integrate the `GSPy` library (`https://github.com/usgs/gspy`, [68] recently developed at USGS.

Before writing data to an internal file, a header needs to be defined. This string is useful for finding bugs, and identifying what was done with the data. In `Tutorial1_PRE_data.py`, the header consists of a version string, the name of script, and date. The sections of this potentially quite long single string are marked with one ore more `|`, which will be replaced by line feeds for printing. It is good practice to update this header whenever the data change. To give an example:

```
aesys.write_aempy(File=f, Data=Data, System=AEM_system, Header=Head, OutInfo=OutInfo)

```

5.1 Visualisation

Data written by `Tutorial1_PRE_data.py` can be visualised by `Tutorial1_PLOT_area.py`, either as a coloured scatter plot or interpolated high-resolution images or contour plots (Figure 18). This script offers also a range of interpolation methods, using `scipy.interpolate.griddata` for nearest neighbour, linear or cubic interpolation, or `scipy.interpolate.RBFInterpolator` using radial basis functions [25, 155] offering a large flexibility in the choice of kernels related to particular spatial covariances. While the different interpolation methods for the profile data are used for visualisation, it goes without saying that they may also be useful for generating smoother data sets for inversion.

`Tutorial1_PLOT_area.py` can also be used for visualise whole data blocks. As an example, Figure 20 shows images of data from the NM (TD)n survey, also demonstrating data transformation. The scaling factor S for $\tilde{\mathbf{d}} = arcsinh(\mathbf{d}/S)$ was set to a value of 500.

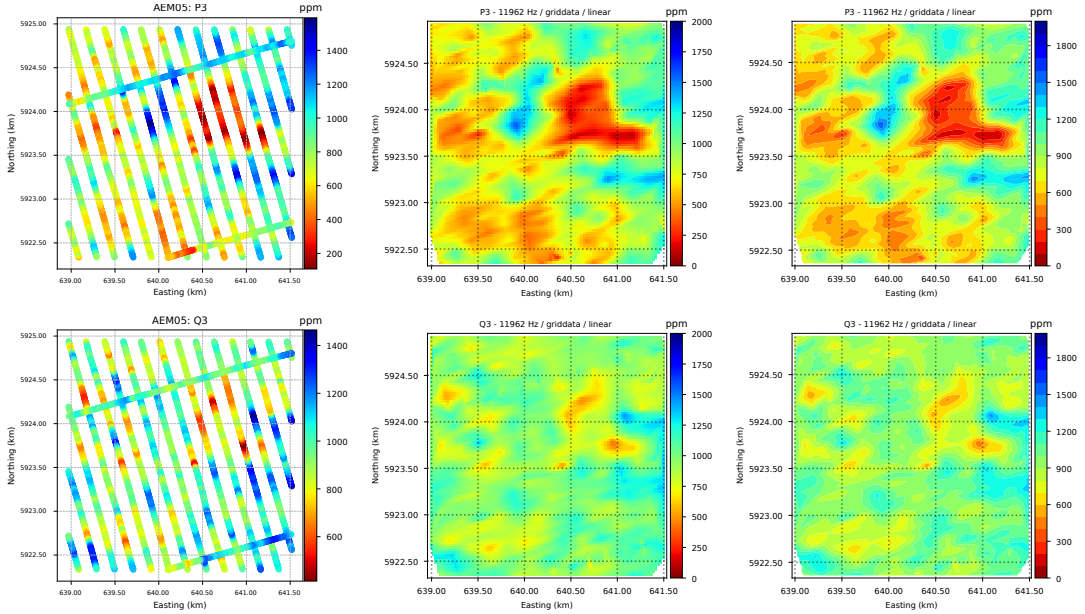

Figure 18: Figure shows the observations for a frequency of 11962 Hz within a rectangular area near St. Gorman's Well (Dublin Basin). Left: Scatter plot. Center: Pseudocolour image. Right: Contour plot. For the latter two, linear interpolation using `scipy.interpolate.griddata` was used.

Another useful script in this early phase is also `Tutorial1_PRE_kmz.py` which will mark the site positions, marking start and end points with the profile names, and may include links to flight line plots as shown in Figure 21. It is easily generalised to include other plots (daya fits, models) as well.

5.2 Processing field data

`Tutorial2_PRE_process_data.py` gives examples for further processing of field data. As the other scripts starting from the data set generated from `PRE_data.py` the first step is to create a list of input files. This can currently be done by directly setting it in the script, reading from a file, or searching within a directory. The most often used variant is searching, an is given as an example:

```
InpDatDir = AEMPYX_DATA+"MyProject/"
SearchStrng = "*.npz"
dat_files = util.get_filelist(searchstr=[SearchStrng], searchpath=InpDatDir)
dat_files = sorted(dat_files)
ns = numpy.size(dat_files)
if ns ==0:
    error("No files with searchstring <"+SearchStrng+"> found!. Exit.")
```

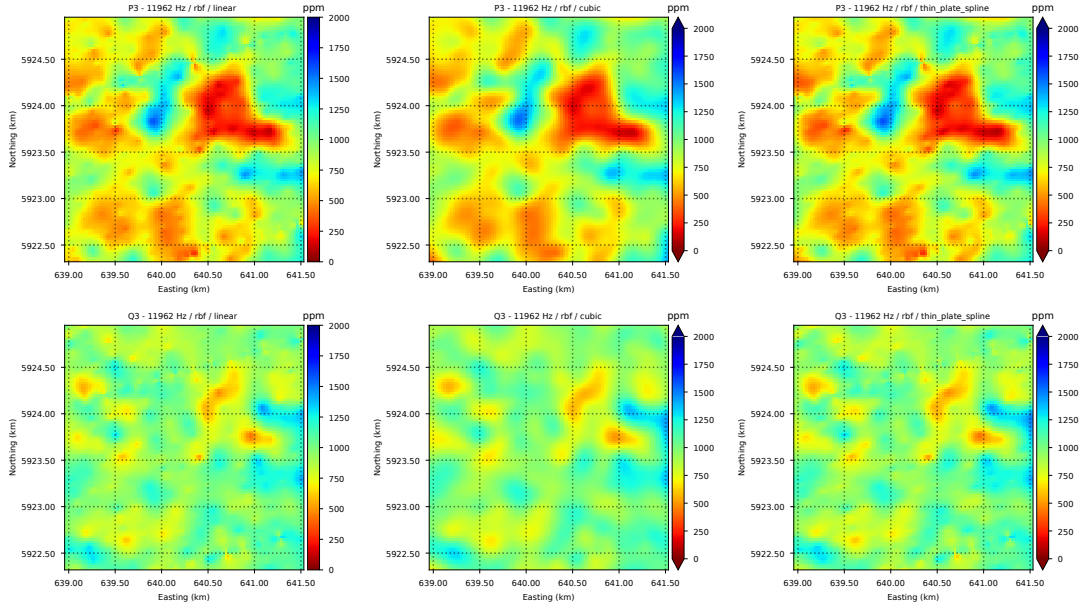

Figure 19: Figure shows the same data as fig. 18, interpolated to a regular grid (≈10 m× 10 m) applying radial basis functions with linear, cubic, and thin plate spline (minimum curvature constraint) kernels [155].

Unix-style wildcards are allowed here. Given this file list, the files can now be used as

```
for f in dat_files:
    Data, Header, _ = aesys.read_aempy(File=f, System=AEM_system, OutInfo=False)
    #### do something ###
    aesys.write_aempy(File=f_new, Data=Data_new,
                      System=AEM_system, Header=Head_new, OutInfo=OutInfo)
```

Within the datasets extracted, several types of errors can be found, which should not be taken into account for any inversion. To deal with this problem, several procedures are defined which replace the “bad” values by certain flags, which may be numeric (e.g.,`-99999`) or NaN (by default). The most common bad values are the sites which are marked by the “power line indicator”, high-fly zones, and areas which show spurious signals. These could result from human installations, naturally magnetic bodies, or other structures, which are not interpretable by any (even 3-D) resistivity distribution. Once found, the bad values have to be processed, depending on the further use. If the algorithms to be applied to the data do not require quasi-continuous data, we think the preferred procedure is to simply delete these sites. However, if the gaps are small, an interpolation may be useful. Within a given work flow the processing may consist of a sequence of commands identifying bad values/sites, and the appropriate treatment of the gaps. An example for this is given below. Here we replace the gaps with a (possibly smoothed) cubic spline interpolation.

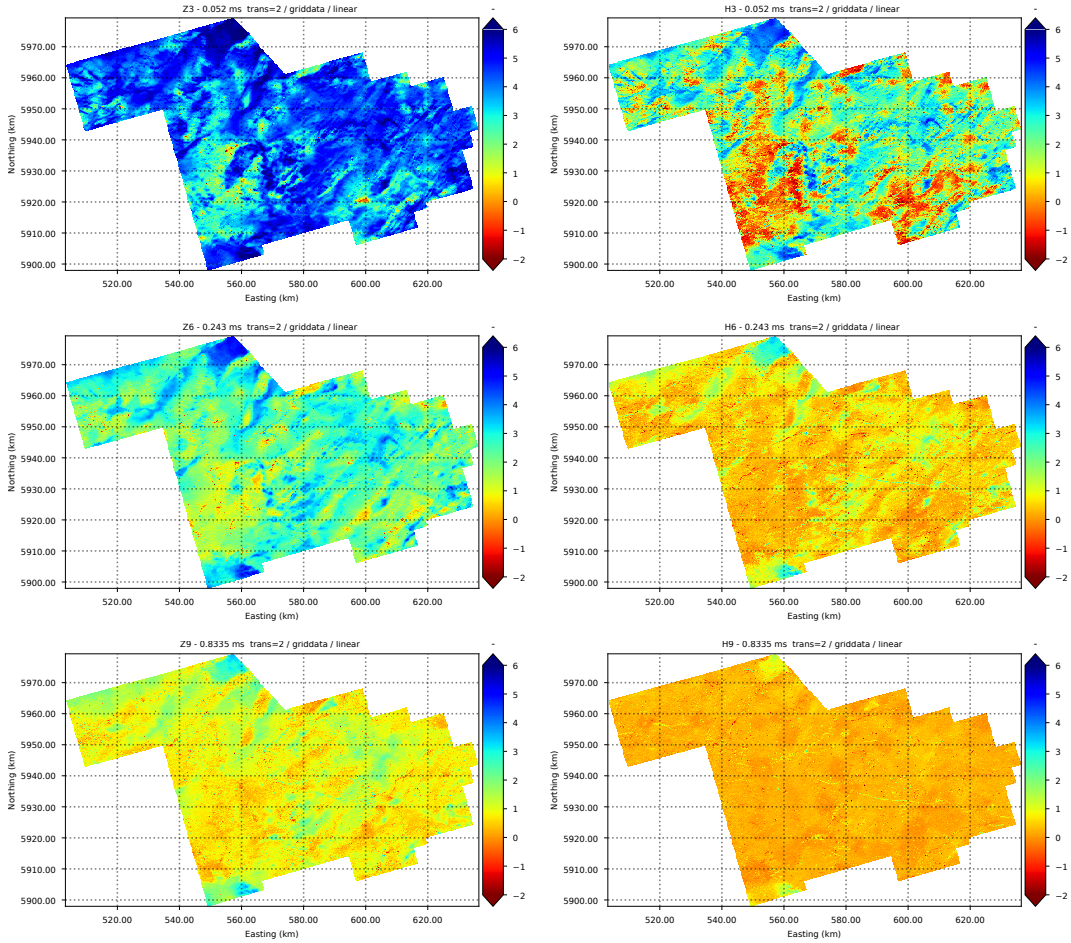

Figure 20: Figure shows data from Block NM, interpolated to a regular grid (≈100 m× 100 m) a linear, interpolation.

```
D = data_obs
# possible impute values: "delete", "median", "average","noise", "value", "spline", "nspl"
impute=["spline", pars]
# possible criteria: "gre", "les"
criterion = "greater than"
threshval=70.
Header = aesys.grow_header(Header, "ALT, threshold = " + str(threshval) + " " + impute[0])
columns=[4,4]       # column 4 is altitude
D, nanindex = prep.insert_flag(D, criterion,threshval,columns)
columns=[4,12]      # column 4 to 12 are interpolated
D, nanindex = prep.handle_gaps(D, columns, impute)
```

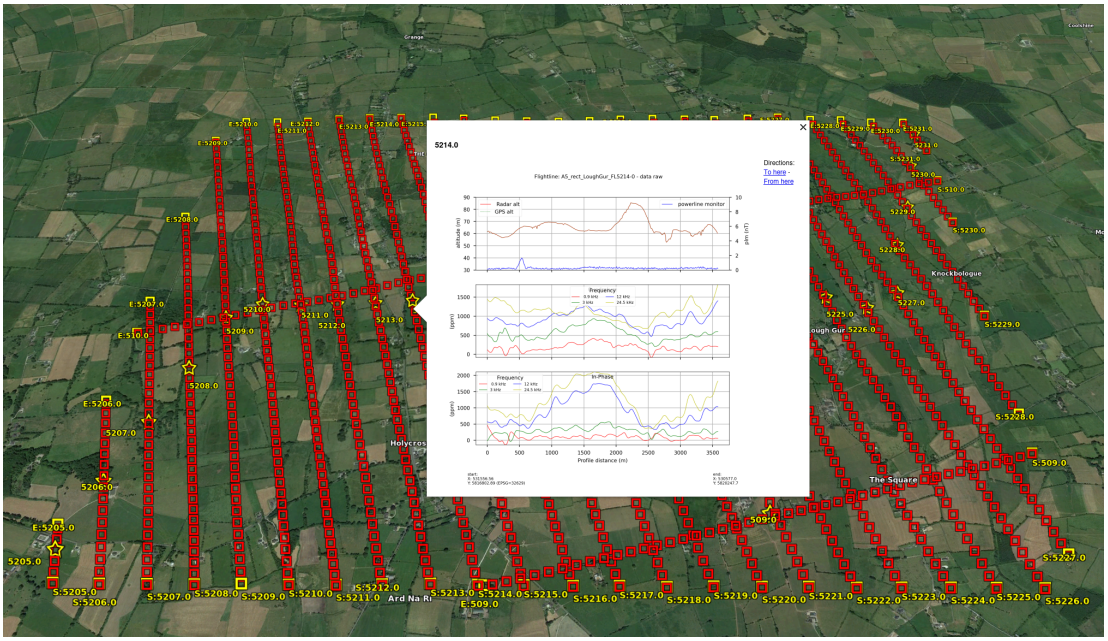

Figure 21: Figure shows data from Block A5 (near Lough Gur), with linked flight line plot. The plots will be activated by klicking on the start/centre/end point of each profile, marked wit a yellow symbol.

As shown by Beamish and Levaniemi [11] radar altitude is sensitive to differences in land use, in particular canopy. This can be mitigated by running a digital filter on this channel. Thus another processing option was added, making the comprehensive library `scipy.signal` available. Currently only Butterworth filtering [132, 107] is implemented, however it is simple to add other options, if required.

```
columns = [4, 5]
Header = aesys.grow_header(Header, "ALT, LPF, IIR n=8")
D, comment = prep.filter_column(D, columns, method=["butter", 4, 1.0 / 20.0])
```

Here, a n^{th} order recursive Butterworth filter is run in both directions (to inhibit phase-shifts). The same procedure is useful for the PLM channel, which sometimes shows non-smooth behaviour, making it difficult to use as a rejection criterion. Thus some low-pass filter should be run before the rejection step.

```
action = " plm lowpass filter"
columns = [14, 14]
Header = aesys.grow_header(Header, "PLM, LPF, IIR n=8")
D, comment = prep.filter_column(D, columns, method=["butter", 4, 1.0 / 20.0])

action = "plm threshold "
```

```
threshval = plmthresh
columns = [14, 14]
Header = aesys.grow_header(
Header, "PLM, threshold = " + str(threshval) + " " + impute[0])
D, nanindex = prep.insert_flag(D, action, threshval, columns,
System=AEM_system)
```

A procedure of particular interest is the decomposition in principal components (PCA), which can be used to reduce noise contamination, and to impose regularity (consistence) on the data. It has been used in AEM before [119, 97]. The PCA approximation is based on the singular value decomposition (SVD) [81, 52] of the matrix of an observation, $\mathbf{D}$, which has n_{data} rows and n_{site} columns. After removing the average of the rows, this matrix can be decomposed into an orthonormal set of basis functions using the SVD as

$$\mathbf{D} = \mathbf{USV}^T \quad , \tag{36}$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, and $\mathbf{S}$ is diagonal, and contains the singular values in decreasing sequence. Choosing the k largest values and truncating the matrices correspondingly, an approximate matrix $\tilde{\mathbf{D}}$ is obtained, which contains only the coherent components of $\mathbf{D}$. It can be shown that this procedure is equivalent to using the eigenvalues of the covariance matrix $\mathbf{DD}^T$. This procedure is implemented here as `prep.svd_decomp`.

```
k = 3
columns=[6,13]
Data_k = prep.svd_decomp(D, k, columns)
```

The results for field data from the hitherto available AEM05 surveys and the test lines [102] suggest that the first 1-3 singular values present the spatially coherent part of the observations. In Section 5.3 we will usually present inversions using only the first 1 or 2 singular components, if not indicated otherwise.

The PCA processing also represents a technique which is useful in enhancing data coherency beyond flight lines. Figure presents the spatial interpolation of the full dataset as shown in Figures18 and 19 compared to low k fields. The results are shown in Figure 23. The advantage of this approach is that it allows us to constrain our inversions without necessarily following profiles. The degree of realism of such 3D-constrained inversions needs to be checked more thoroughly.

Given the high redundancy of the flight line sampling, it is often advantageous to decimate the data. There is a new option for the processing now, which will do this, allowing for several methods. This can be a classical decimation (i.e., low-pass filter and sample every kth) point, or a moving average/median. In addition, also a split of flight lines into several interlaced data sets, which may also be useful for uncertainty evaluation. These options are currently implemented within a separate script, `Tutorial2_PRE_reduce_flightline.py`, but could easily be integrated in the processing or inversion script.

```
action = "decimate"
if "dec" in action:
```

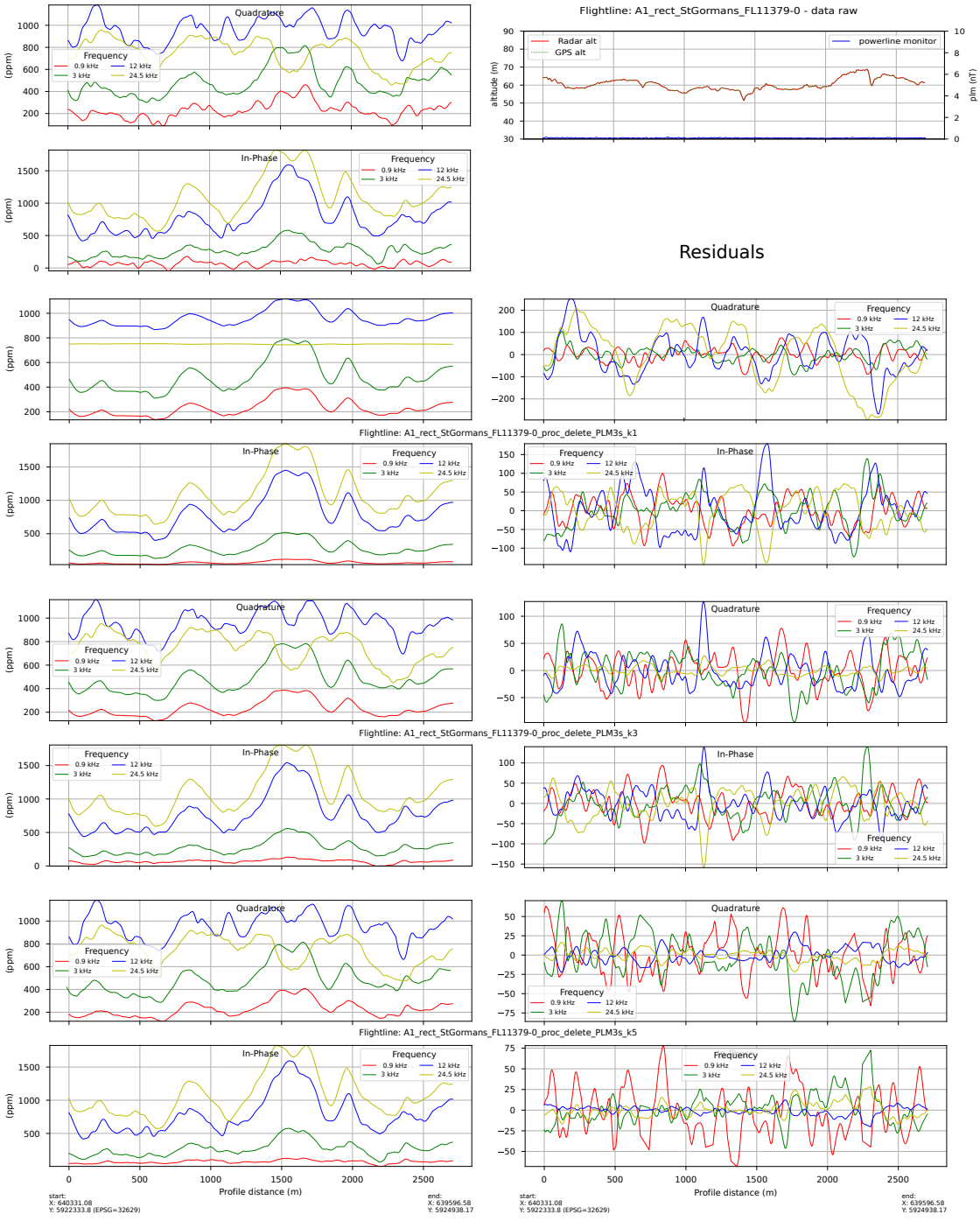

Figure 22: SVD decomposition of a data set from the A1 block. This figure shows from top to bottom original data, and approximate data for the first 1, 3, and 5 singular values. Left: Approximate data. Right: Corresponding residuals. Note the different vertical scales.

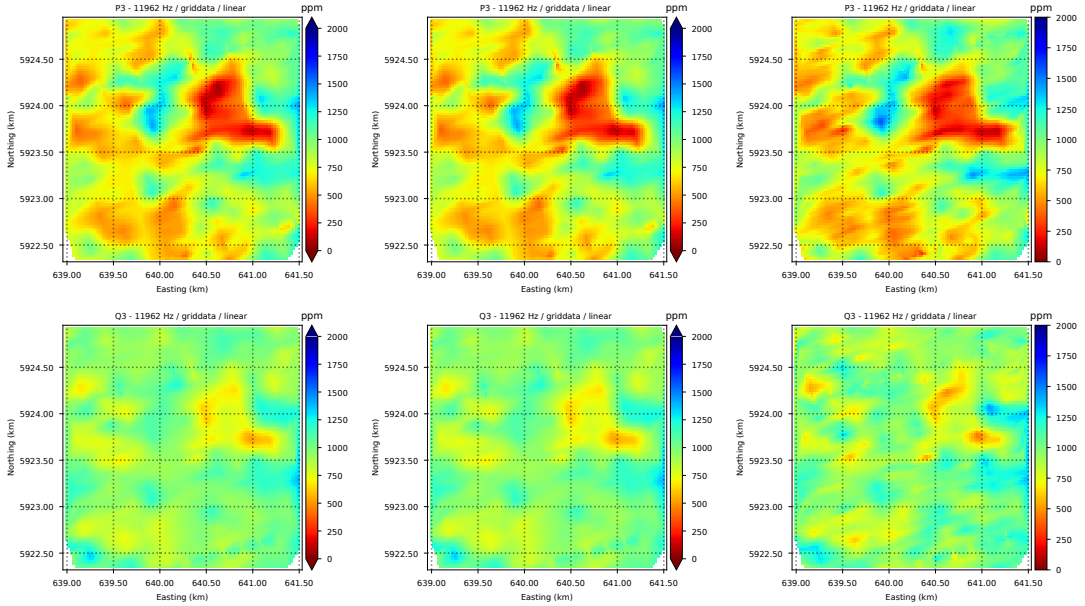

Figure 23: Figure shows the same data as Figures 18 and 19 as pseudocolour images, interpolated to a grid of cell size of 25×25 m. Left: k=1 singular vector. Center: k=3 singular vectors. Right: original data.

```
    Window = 3
    Method = ["mean", Window] # ["median", Window]
    D, blkhead = prep.reduce_data(D, Method=Method, System = AEM_system)
```

or, for splitting the data into `Step` sets:

```
action = "split"
if "spl" in action:
    Step = 3
    Method = ["Step", Step]
    if Step > 1:
        for start in numpy.arange(Step):
            Dsplit=D[start:-1:Step]
```

The split data sets can be written to a file, or used directly within the script.
Finally, another useful option is cutting out an intervals from a single flight line:

```
        action = "cut"
        if "cut" in action:
        Interval = [1000., 2000]
        r_prof = pre.get_profile()
```

```
        Dsplit=D[start:-1:Step]

```

5.3 Flight line inversion

Here we will guide the user through a full flight line inversion with the script `Tutorial3_INV_data.py`. This is a much more complex task than the processing, and a general flow chart can be found in Figure 9. We assume that appropriate data sets have bee produced in the earlier steps. To start an inversion, several steps are required:

1. Definition of the general inversion type, which includes choosing of the regularization operator, and weighting parameters.
2. Setup of the initial and forward model.
3. Choice and optionally additional processing of data.

The control variables, data-, and model-related information is passed to the inversion algorithms (modules `alg.py` or `alg_par.py`) through 3 python dictionaries, `Ctrl`, `Data`, and `Model`, while results are stored in `Results`. We have chosen this method, as this gives us maximal flexibility in what to pass, without changing the calls to different algorithms. This means that the content of these dictionaries will differ depending on the chosen approach. This method is also used for the uncertainty-related algorithms, which often are based on inversion methods (e. g., RTO, Jackknife), and can be introduced by adding the necessary parameters to `Ctrl`. The general call is (here for Tikhonov inversion with optimal choice of the regularisation parameter τ):

```
Results = alg.run_tikh_opt(Ctrl=Ctrl, Model=Model, Data=Data,
OutInfo=OutInfo)
```

The only system-specific settings can be found in the blocks below, such that no special versions for the currently relevant systems (AEM05 and GENESIS) need to be kept.

```
AEM_system = "aem05"  # "genesis"

if "aem05" in AEM_system.lower():
    FwdCall,NN, _, _, _, = aesys.get_system_params(System=AEM_system)
    nL = NN[0]
    ParaTrans = 1
    DataTrans = 0
    DatErr_add =  50.
    DatErr_mult = 0.03
    data_active = numpy.ones(NN[2], dtype="int8")

if "genes" in AEM_system.lower():
    FwdCall, NN, _, _, _, = aesys.get_system_params(System=AEM_system)
    nL = NN[0]
    ParaTrans = 1
```

```
    DataTrans = 0
    DatErr_add = 100.
    DatErr_mult = 0.01
    data_active = numpy.ones(NN[2], dtype="int8")
    data_active[0:11]=0  # only vertical component
    # data_active[10:11]=0  # Vertical + 'good' horizontals'
```

Data are read from the NPZ file produced by the processing scripts. As already described above, a list of flight line data file names is created, and a loop is executed over this list. Sites are inverted independent of each other, so that the data sets do not need to be flight lines, but can also be area-covering files.

```
InpDatDir = AEMPYX_DATA+"MyProject/"
SearchStrng = "*.npz"
dat_files = util.get_filelist(searchstr=[SearchStrng], searchpath=InpDatDir)
dat_files = sorted(dat_files)
if numpy.size(dat_files)==0:
    error("No files with searchstring <"+SearchStrng+"> found!. Exit.")
for f in dat_files:
    Data, Header, _ = aesys.read_aempy(File=f, System=AEM_system, OutInfo=False)
    ...
    for s in numpy.arange(numpy.shape(Data)[0]):
        ...
```

For each of these data sets, there will be a deeper loop over the sites. While the content of `Ctrl` remains constant for a full inversion run, `Data`, and `Model` are site-specific, and need to be set (or at least modified) within the site loop.

5.3 `Ctrl`: Inversion type and regularisation control

This describes the setup of `Ctrl`. The parameters stored into this dictionary need to be defined already. Note that some elements within depend on the model and data, such as the setup of regularisation operators. This means that the sequence in the actual script will not be the one in our description in the following. The content of `Ctrl` will vary with the inversion type. Here we concentrate on the well-developed `TikhOpt` , while the other implemented methods are very similar.

The structure of `Ctrl` is given by:

```
Ctrl = dict([
("system", [AEM_system, FwdCall]),
("inversion", [RunType, RegFun, Tau0, Tau1, Maxiter,Thresh,
               LinPars, SetPrior, Delta, RegShift]),
("covar", [L0, Cm0, L1, Cm1]),
("transform", [DataTrans, ParaTrans]),
("uncert", Uncert)])

```

The first option to define is `RunType`, and the parameters determining regularisation, in particular the weighting between data fit and constraining operators. As pointed out in Section 3, this weighting is crucial for a successful inversion. In the block below, we opt for a Tikhonov-type inversion, with τ_0 small, and optimisation of the parameter τ_1. This means that the inversion is only weakly constrained by the prior model $\mathbf{m}_a$, but the optimally flat model is sought. To achieve this the generalised cross validation function GCV [151] is minimized within a predefined set of in most cased logarithmically spaced values. GCV appears to be the most robust method implemented. Other options are the L-curve method [58], UPRE [150], or MLE [38].

```
RunType = "TikhOpt"    # "TikhOcc", "MAP_ParSpace", "MAP_DatSpace"
RegFun = "gcv" # "fix", "lcc", "gcv", "mle", "upr", "ufc"

RegVal0 = 1.e-5
NTau0 = 1
Tau0min = numpy.log10(RegVal0)
Tau0max = numpy.log10(RegVal0)
Tau0 = numpy.logspace(Tau0min, Tau0max, NTau0)

RegVal1Min = 0.1
RegVal1Max = 1000.
NTau1 =65
Tau1min = numpy.log10(RegVal1Min)
Tau1max = numpy.log10(RegVal1Max)
Tau1 = numpy.logspace(Tau1min, Tau1max, NTau1)

```

We also allow a shift from the optimum within the list of weights, `RegShift`, which should in general be 0. However, it can be used to nudge the inversion to be more conservative (>0) or daring (<0), with the latter allowing more but possibly noise-contaminated structure in the model.

The iteration process itself is controlled by `inversion`. Checks for convergence use

```
Maxiter = 10
Thresh = [0.9, 1.0e-2, 1.0e-2]
```

where a check on all four numbers, `Maxiter` and `Thresh`, will end the iteration. Apart from the maximum number of iterations, the process stops if the data fit is good enough, it's change is small, or the model change is small.

The measure for the goodness of fit which is commonly used in large-scale inversion is the normalized RMS defined as

$$NRMS = \sqrt{(N_d - 1)^{-1} \sum_{j=1}^{N_d} \left(\frac{d_{obs,j} - d_{cal,j}}{\sigma_j} \right)^2} \quad . \tag{37}$$

If all assumptions are fulfilled, particularly if the error is specified correctly, this parameter should be near 1. In this case, very small values indicate over-fitting. As the underlying statistical theory

requires large N_d, in AEM, a different and intuitive quantity is often used, namely the (possibly weighted) Symmetric mean absolute percentage error SMAPE

$$SMAPE = \frac{100}{N_d} \sum_{j=1}^{N_d} \frac{w_j |d_{obs,j} - d_{cal,j}|}{(|d_{obs,j}| + |d_{cal,j}|)} \quad , \tag{38}$$

which is given in %. This is the number also given in the output of CSIRO's AirBeo code. It has the advantage of being independent of the assumed data error, but is less interpretable in statistical terms.

To fill `covar`, the regularisation operators need to be constructed, which here is the identity and a first order differential operator, which red to be restricted to the active parameters defined in Section 5.3.2.

```
D0 = inverse.diffops(dz, der=False, mtype="sparse", otype="L0")
L = [D0 for D in range(7)]
L0 = scipy.sparse.block_diag(L)
Cm0 = inverse.extract_cov(L0.T@L0, mod_act)

D1 = inverse.diffops(dz, der=False, mtype="sparse", otype="L1")
L = [D1 for D in range(7)]
L1 = scipy.sparse.block_diag(L)
Cm1 = inverse.extract_cov(L1.T@L1, mod_act)
```

As pointed out in Section 3, the weighting of these operators defined by one of the schemes mentioned above is a crucial part of a successful inversion.

The following block defines parameters for the simplified line search for each iteration. If `LinPars = []`, no line search will be applied. The line search was newly added to `AEMpyX`, as this often improves the convergence considerably with a lower cost than increasing the number of iterations.

```
Maxreduce = 5
Rfact = 0.66
LinPars = [Maxreduce, Rfact]
```

The last parameters to choose are related to transformations of data and parameter. Electrical resistivity of the subsurface spans many decades, and `ParaTrans = 1` transforms the resistivities to their natural logarithms. In the case of inversion for IP parameters (m, τ, c), there are several possible choices, as pointed out in Fiandaca et al. [45], some of which have been implemented but not tested. In the case of FD data, usually no transformations are necessary (`DataTrans=0`), unless negative values are supposed to be inverted. If TD data are strictly positive, they can be used as they are, or transformed to their natural logarithms `DataTrans = 1`, but the transformation proposed in [128] and [129] are advised (`DataTrans = 2`). Note that the transformation also neet to be taken into account for the corresponding errors.

```
DataTrans, ParaTrans = 0, 1
```

The remaining parameters are of minor importance. `Delta` is the perturbation used for Jacobian calculations, and a value of 1.e-5 is a safe choice.

5.3 `Model`: model setup

Finally, the model-related parameters need to be set and stored into the `Model` dictionary. It contains the indicator array for parameter activity `mod_act`, the prior and initial models `mod_apr` and `mod_ini`, respectively, the model error `mod_err`, and the model limits `mod_bnd` (currently not yet activated).

```
Model = dict([
             ("m_act", mod_act),
             ("m_apr", mod_apr),
             ("m_var", mod_var),
             ("m_bnd", mod_bnd),
             ("m_ini", mod_ini)
             ])
```

In order to define all these parameters, we fist need the number of layers in the model, as everything following depends on it. In `AEMpyX`, seven parameters per layer are possible as a shown in Figure 7. The structure of the model is defined by the following code block by calling `inverse.initialize_1dmod(Nlyr)`, where `Nlyr` is the number of layers including the lower half-space. Taking into account the number of physical parameters per layer, this will return the necessary arrays, which then have to be activated and pre-set with appropriate values. Experience with a large number of flight lines from the different Tellus frequency-domain surveys suggest a robust choice of 20-40 layers with logarithmically increasing thicknesses. Note that in order to make the smoothing regularization as chosen in the example effective, the domain of calculation should be larger than the depth of penetration for the lowest frequency (which depends on the electrical resistivity), reducing the importance of the underlying half-space in the inversion.

```
        Nlyr = 25
        mod_act, mod_apr, mod_var, mod_bounds, m_state = inverse.init_1dmod(Nlyr)

```

All parameters are initialised as inactive (`m_act = 0`), and the other fields are filled with appropriate defaults. This is necessary, as the forward calculations need values even for the inactive parameters. Subsequently, the actual inversion is set up by activating the parameters to be inverted for, and the layer thicknesses. Finally, limits for resistivity parameter are set. Though a logarithmic barrier function [21] has been prepared for implementing these limits, it is not yet activated in the toolbox.

```
        mod_act[0*Nlyr:1*Nlyr] = 1

        dzstart, dzend  = 3., 10.
        dz = numpy.logspace(numpy.log10(dzstart), numpy.log10(dzend), Nlyr)

        Guess_r = 100.0  # initial guess for resistivity in mod_apr
        Guess_s = 10.    # mod_std defines standard deviation of mod_apr
        mod_apr[0*Nlyr:1*Nlyr] = Guess_r
        mod_var[0*Nlyr:1*Nlyr] = Guess_s**2
        mod_apr[6*Nlyr:7*Nlyr-1] = dz[0:Nlyr - 1]
        mod_var[6*Nlyr:7*Nlyr-1] = 1.
        mod_bnd[:,0] = 1.e-30
        mod_bnd[:,1] = 1.e+30

```

The first line activates the electrical resistivities for inversion, the priors and their variances are set accordingly below.

The `Model` dictionary is site-specific, as the prior may be so, for instance when using earlier, low-resolution inversion results for the prior. It thus needs to be set or modified in the site loop.

5.3 `Data`: data and error setup

Data are passed to the inversion algorithm through the dictionary `Data` with the following structure.

```
Data = dict([
            ("d_act", dat_act[ii,:]),
            ("d_obs", dat_obs[ii,:]),
            ("d_err", dat_err[ii,:]),
            ("alt", site_alt[ii])
            ])
```

While the observations including altitude are taken directly from the dataset read, and the activation has already ben defined earlier (though this could also be done here in a data-driven manner), the errors still have to be defined. Useful data errors are usually not provided by the contractors. In Section 6 we have already discussed this problem. In the inversion script, currently we use a combination of additive and multiplicative errors, which have been set to fixed values in the system-specific blocks. Note that the errors are defined in the physical domain. The required transformations in case `DataTrans!=0` are done within the inversion algorithm proper. In contrast to claims in the literature , we do not find big differences in the results when data are positive (as should be for pure resistivity-driven responses), though the use of transformed data is necessary when visualising them. A comparison of raw (`DataTrans=0`) and arcsinh-transformed (`DataTrans=2`) is shown in Figure **??**.

```
dat_act = numpy.tile(data_active,(nsite,1))
dat_err = numpy.zeros_like(dat_obs)
dat_err[ii, :], _ = inverse.set_errors(dat_obs[ii, :],
                                       daterr_add=DatErr_add,
                                       daterr_mult=DatErr_mult)
```

5.3 `Results`: Output from inversion

All inversion procedure return a dictionary called `Results`, which, however, may differ in content, and thus the corresponding output NPZ file will differ.

Default outputs for the traditional inversion methods (`Tikh[Opt, Occ]`, `Map-[D, M]`) stored in `Results` are:

```
results = \
    dict([
    ("model", [modl, merr, sens, m_state]),                    # model-related
    ("data", [d_obs, d_cal, d_err, d_state, d_act]),           # data-related
    ("log", [niter, conv_status, nrms_iter,                    # iteration log
            dnorm_iter, mnorm_iter, rvals_iter, dfits_iter]),
        ])
```

If `Uncert = True`, some additional are generated, and merged into the dictionary:

```
uncpars = \
    dict([
        ("jacd", Jd),               # jacobian
        ("cpost", C),               # cov a-post
        ("merr", E),                # model error
        ("mresm", [Rm, Sm, Nm]),    # model resolution
        ("dresm", [Rd, Sd, Nd]),    # data resolution
        ("gi", G),                  # generalized inverse
    ])
results.update(uncpars)
```

Noting that the scripts are called site-wise, it is still to decide how much of the information, even when the uncertainty option is not used, is stored into the output files. As an example, we give the current storage in the following. It is easy to change the list according to particular preferences. The `Ctrl` dictionary is saved for every run (flight line) in a separate NPZ file.

```
numpy.savez_compressed(
    file=Fileout,
    fl_data=file,
    fl_name=fl_name,
```

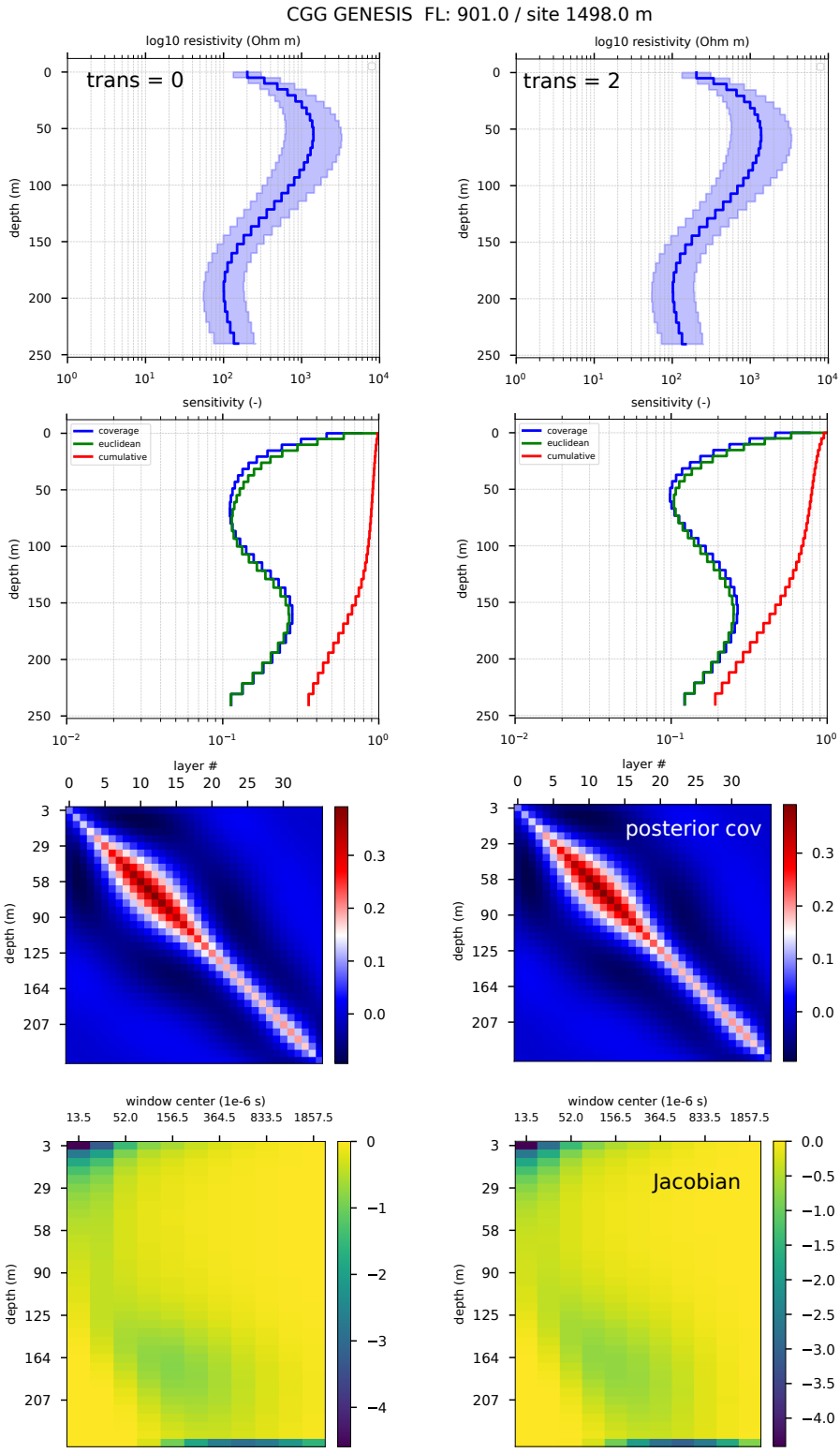

Figure 24: Comparison of results from raw and transformed data from Bundoran testline 901. The left column results from raw data, the right from transformed data, where the scale value for the arcsinh transform is set to 1000. Though there are small differences, the results are basically very similar. Note the behaviour of the last row, which is the effect of the halfspace resistivity. It will disappear when the parametrization is extended to greater depth.

```
    header=header,
    site_log =site_log,
    mod_ref=mod_apr,
    mod_act=mod_act,
    dat_act=dat_act,
    site_modl=site_modl,
    site_sens=site_sens,
    site_merr=site_merr,
    site_dobs=site_dobs,
    site_dcal=site_dcal,
    site_derr=site_derr,
    site_nrms=site_nrms,
    site_num=site_num,
    site_y=site_y,
    site_x=site_x,
    site_gps=site_gps,
    site_alt=site_alt,
    site_dem=site_dem)
```

If `Uncert` is activated, currently only the flattened Jacobian $\tilde{\mathbf{J}}$ and the posterior covariance $\hat{\mathbf{C}}_m^p$ are added, meaning that the two lines below are added. Most traditional uncertainty and resolution measures can be derived from this information.

```
     site_jac=site_jac,
     site_pcov=site_postcov,
```

It is easy to add the corresponding fields to the file.

Currently several scripts exist for visualisation of the results. `Tutorial3_INV_plot_datafit_flightline.py` will produce depth slices for flight lines, optionally together with the data fit, and the site nRMS. Here, `PlotType` will determine how the figure will look like.

```
        """
        PlotType will determine what will be plotted in the figure:
        0:      model, data fit, rms
        1:      model + rms
        2:      model
        3:      datafit
        """
        PlotType = 0

```

5.3 Choice of inversion setup

In this section we show some visualisation examples illustrating the different choices for the inversion set up. For comparison, we have arbitrarily chosen flight line 11379-0 from the A1 block (St. Gormans rectangle).The aim of this exercise is to give an idea of what the various components

will influence. An successful inversion will always be a trade-off between the resolution and reliability required by the user's scientific aims, and the computational effort. It can be reduced by decreasing the numbers of data used for the inversion, the complexity of the underlying model, nd by the choice of the inversion algorithm. In the following we will show examples for several of the mentioned choices.

The number of data in AEM is considerable, but fortunately there is a high redundancy along the flight lines, where the sampling interval is less then 10 m for FD, and less than 20 m for TD. Obviously this can be used decrease computational times, in particular for the case of FD data. in Figure 25 we show inversion results for data decimated by a factor of 3, 5, and 7. Depending on the use of the results, on can obviously reduce the data set considerably by appropriate decimation levels. As a side effect, the implicit smoothing may also improve convergence. However, experience shows that sophisticated decimation schemes (as are available in `AEMpyX`) are not even necessary for many tasks. It may be enough to invert just every n^{th} site. Figure 26 shows inversion results for a split with n=5

However, experience shows that sophisticated decimation schemes (as are available in `AEMpyX`) are not even necessary for many tasks. It may be enough to invert just every n^{th} site. Figure 26 shows inversion results for a split with n=5, together with the full data set, and averages over 5 adjacent sites.

The influence of the number of singular components in the PCA of flight line data on the inversion results is illustrated in Figure **??**, which shows the output from a Tikhonov-type inversion (25 layers). The inverted data are the same as shown in Figure 22.

The complexity of the underlying model in the case of 1-D inversion is, apart from the possibility of inverting for different data types, the number of layers. We have concentrated on the approach of highly-parametrized (tomography-like) models, with an explicit application of regularising constraints. Keeping in mind that we only have few data values per site, we have tested the cases of 60, 50, 40, 30, 20, and 10 layers. The results are shown in Figure 27, and suggest that numbers between 20 and 30 seem to be viable trade-off.

Furthermore, choosing an appropriate algorithm will play a major role. Noticing that the more sophisticated methods of regularisation parameter choice as `gcv` increase the computing time considerably. For the typical choice for the number of regularisation parameters tested in each iteration, here 64), the factor is about 5. It might be suggested that it is enough to run it with fixed parameters after some testing. This is of course dangerous. A τ_1 too small will lead to a diverging inversion, and/or to "noisy" models. This is illustrated in Figure 28.

Summarizing, for most purposes it recommended to use only 2 or 3 singular values for data de-noising, and decimate the data by a factor of 3 or 5 along the flight line. As we generally choose a very small value for zeroth-order regularisation ($\tau_0 = 1.e-5$), the choice of the prior model is not excessively important for the well-determined features. As we can see in Section 5.4, this can be used to define a depth of investigation following the approach of Oldenburg and Li [105]. It is only rarely successful using a fixed first-order regularisation parameter τ_1. We suggest to use the `gcv` or `mle` methods for reliable and consistent results. As can be derived from results presented in Section 5.4, for the case of layer resistivities, only 1-3 parameters (AEM05) or 2-4 (GENESIS) can be resonably well determined from the data.

5.4 Uncertainty quantification

In this section, we will describe the use of `AEMpyX` for the evaluation of time- and frequency-domain datasets. Keeping in mind that all uncertainty quantification methods will necessarily require

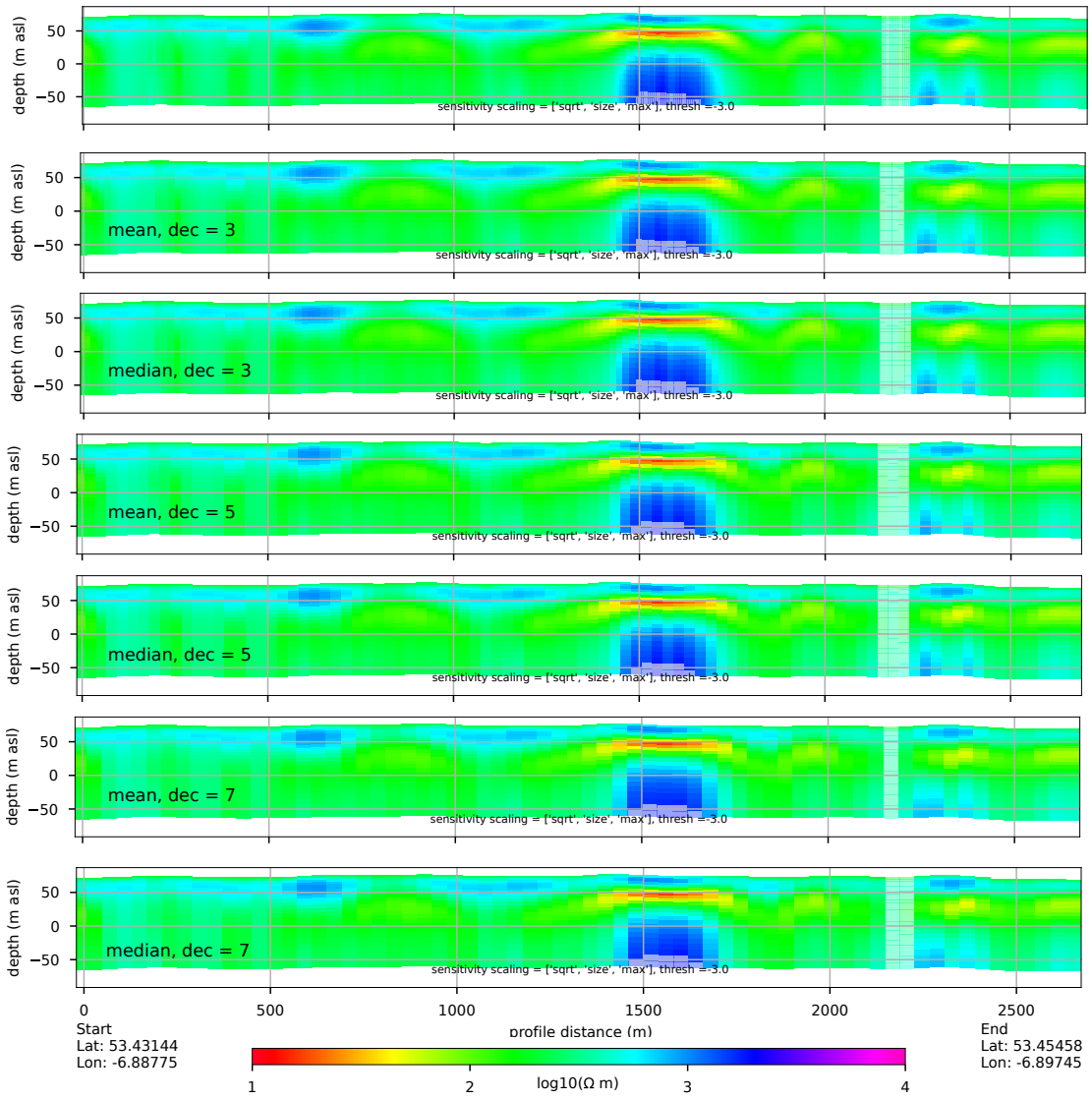

Figure 25: Inversion results for different decimation levels. The top panel is inverted from the full flight line data set, while below you find results for levels of 3, 5, and 7. For comparison, decimation by mean and median are included.

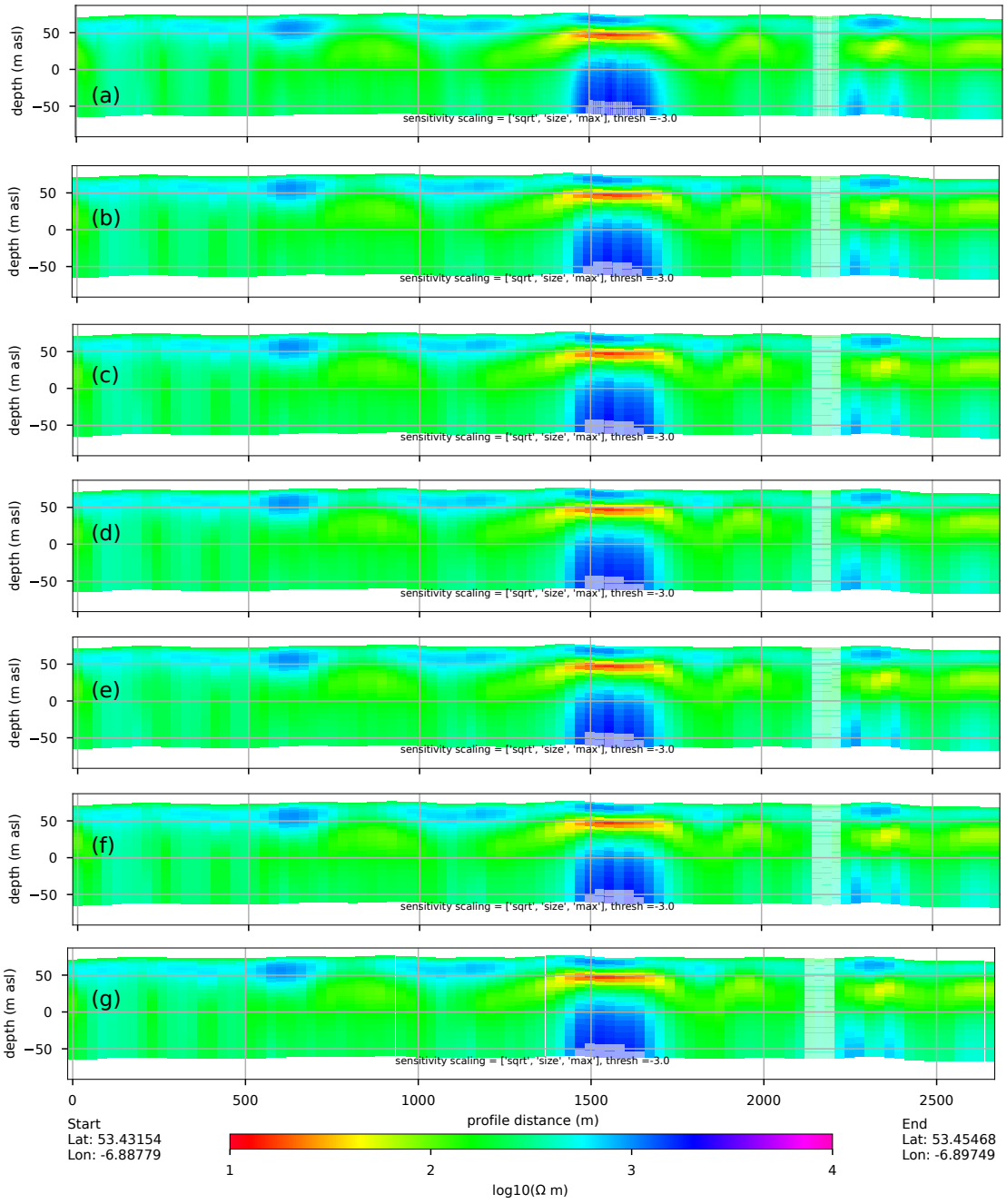

Figure 26: Inversion results for different data reductions. Inversion results for: (a) full flight line; (b) averages over 5 sites; (c)–(g) all 5 split data sets.

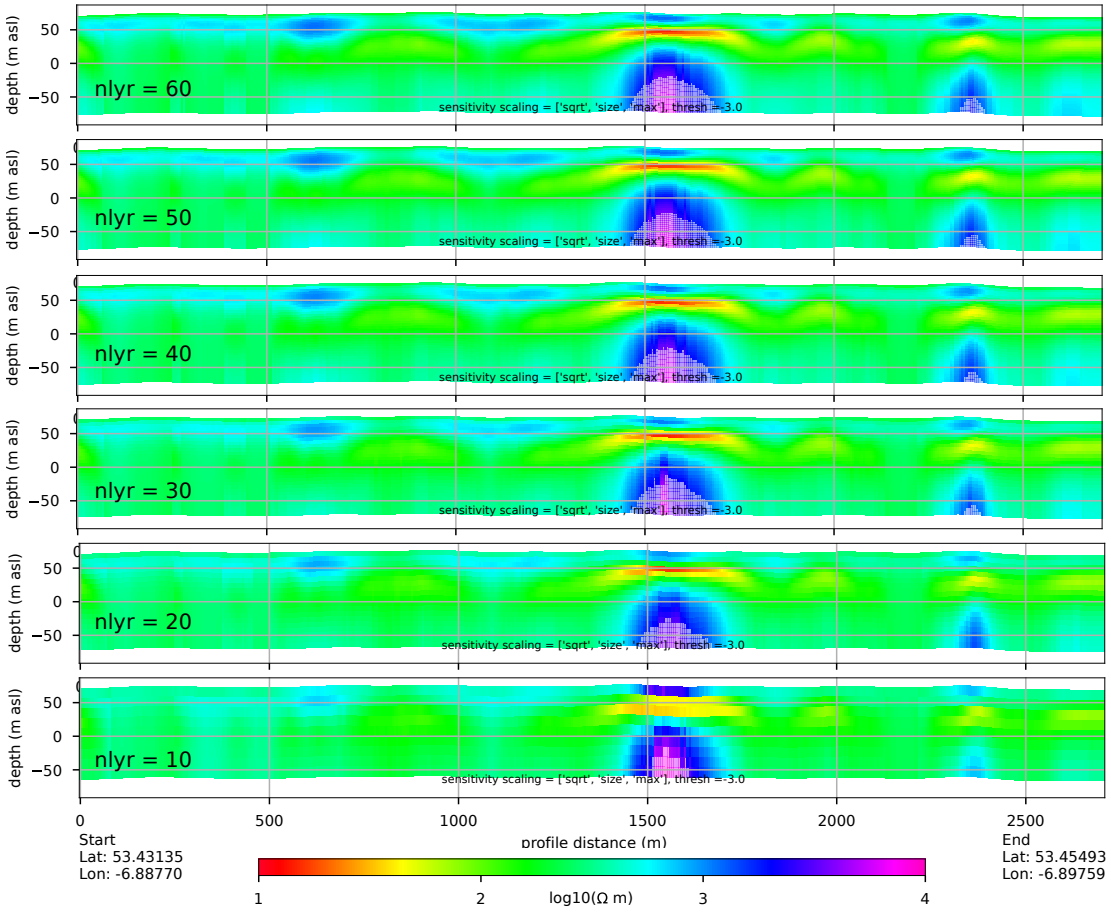

Figure 27: Inversion results for different choices for number of layers for the inversion. From top to bottom models are shown for 60, 50, 40, 30, 20, and 10 layers, employing a Tikhonov approach with optimal regularisation via GCV. The initial model for all inversions was a 100 Ωm half space. The choice of 20-30 layers seems to be a trade-of between resolution and computational costs.The limit for sensitivity-based shading is 10^{-3} for the square-root of sensitivity.

considerably more computer resources, it is obvious that most of the methods discussed in the literature can not be used for each single site in the enormous data sets of Tellus surveys, even if data are decimated as suggested in Section 5.2. The typical scale of such an effort is probably some local survey with a more or less well defined target. Form the methods mentioned here, the traditional ones, which are a by-product of the normal inversions, require least effort, though their graphical presentation for a full data set is not always practicable. For this reason we provide some of this information within the normal plotting routines, (e.g., errors, different flavours of sensitivity, DoI) as shading/blanking of model areas. While the necessary parameters for these traditional methods are always available from the inversion proper, further analysis, as the graphical presentation and ancillary calculations, will usually be done on smaller subsets, as regions of interest, typical structures, or simply some (possibly random) sampling. For this purpose, given a particular data set, we offer different options, some of them are implemented in

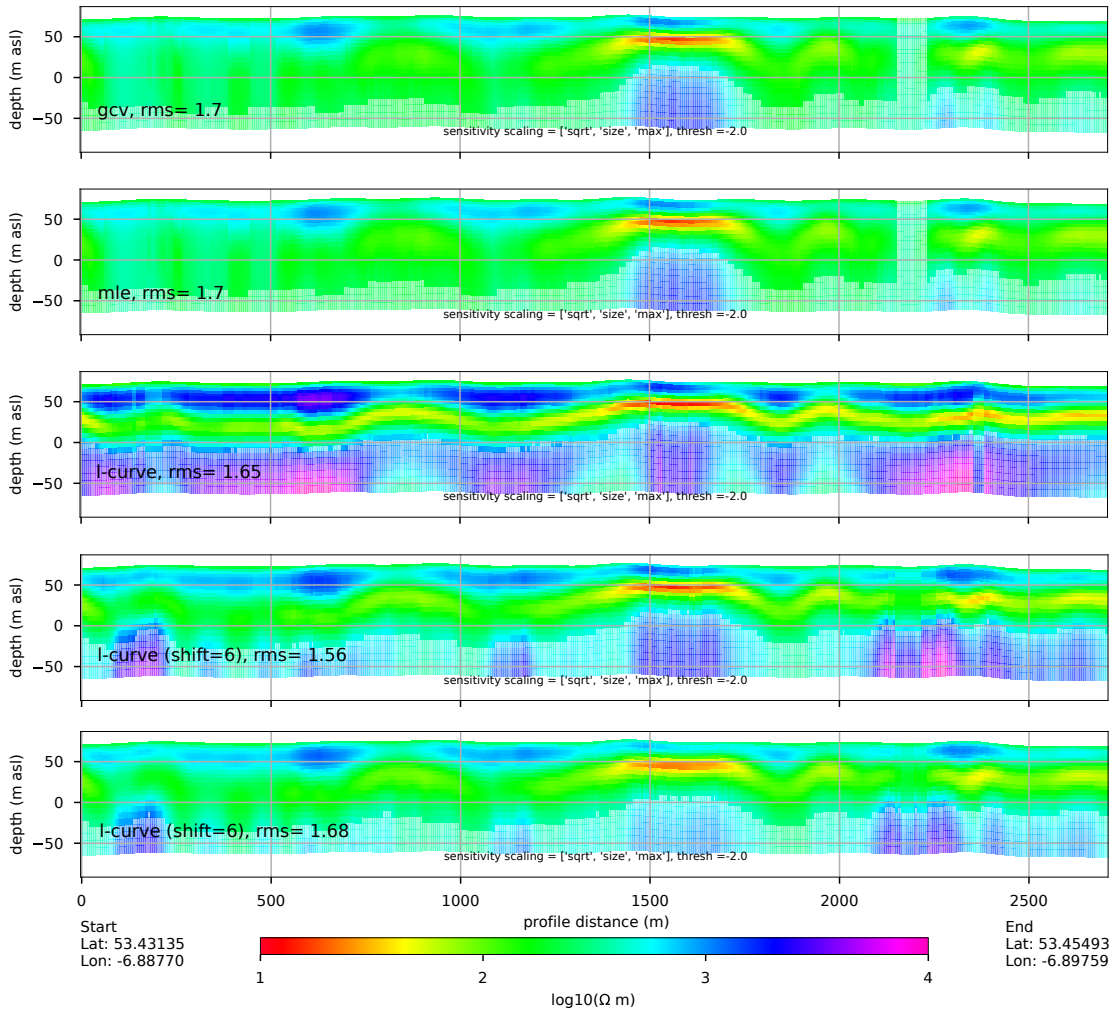

Figure 28: Inversion results for different choices of regularisation. Our experience shows that the statistically based methods as `gcv` or `mle` lead to more conservative model structures, while the L-curve approach `lcc` produces stronger contrasts. The regularisation parameter τ_1 is considerable smaller in this case, with the corresponding decrease in nRMS.Very similar results can be obtained if τ_1 is shifted about half a decade to higher values. Note that for this figure the sensitivity-based shading is 10^{-2}, corresponding to a decrease of sensitivity by a factor of 10^4.

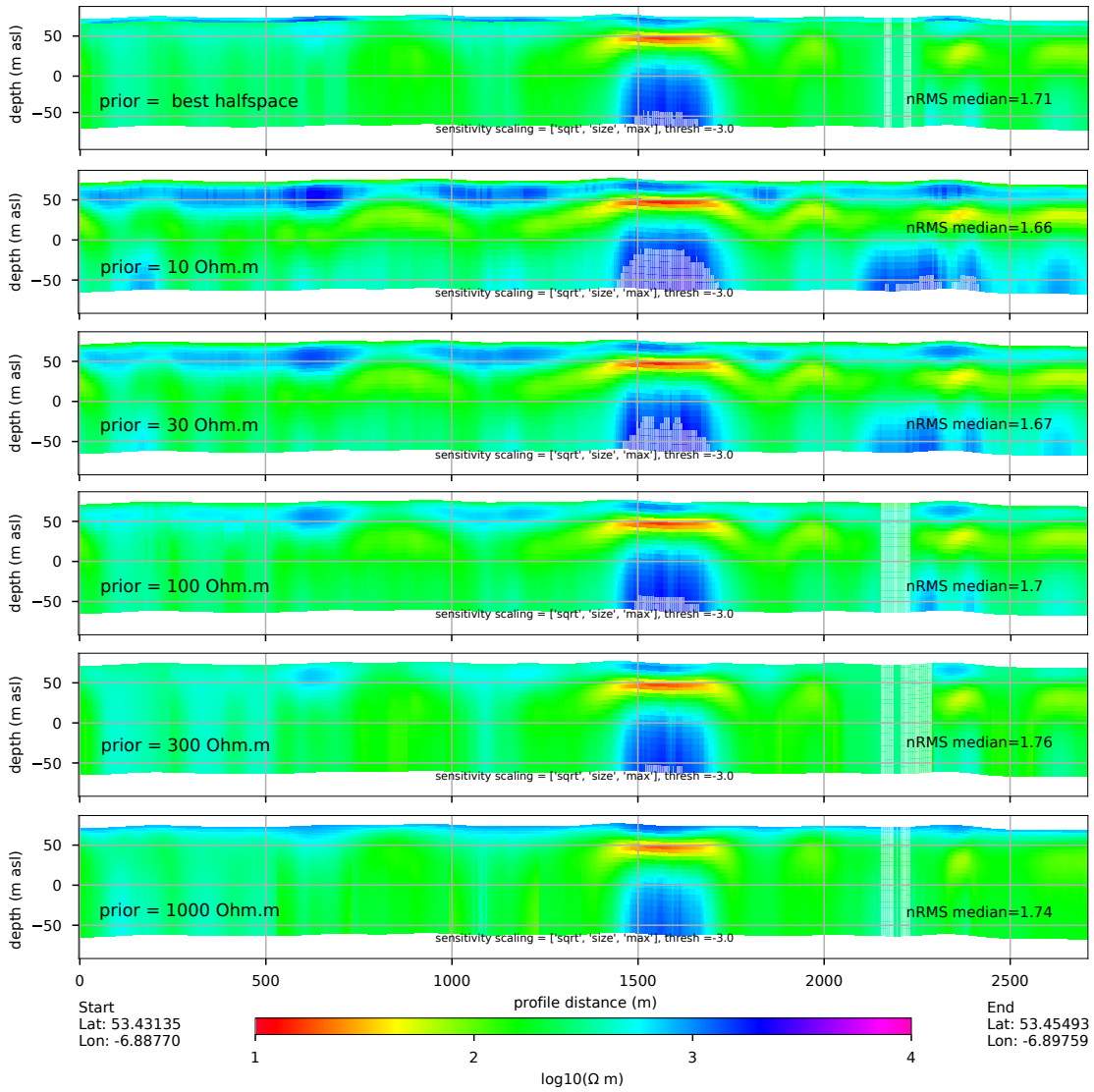

Figure 29: Inversion results for different choices of the prior. From top to bottom we show the results for a "best halfspace fit". The resistivity values of this best half-space fall between 100 Ωm and 1000 Ωm. In order to see any effect, the zero-order regularisation parameter τ_0 needed to be increased to a value of $10^{-1)}$.

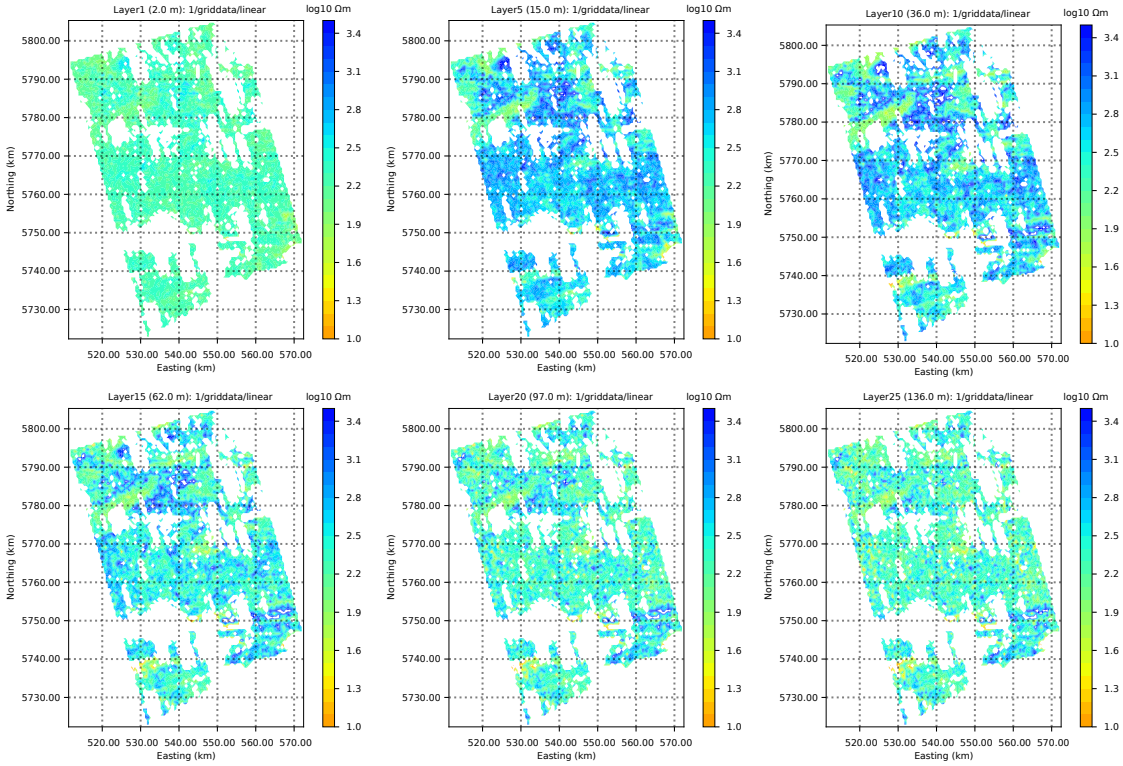

Figure 30: Inversion results

the scripts.

5.4 Traditional Methods

All mentioned inversion algorithms produce parameters which can be useful for further analysis and evaluation of the results. As already visible in the respective Jacobians (Figure 38), the Time domain results reach potentially much deeper than the FD ones.

5.4 Ad-hoc methods

Being aware of the very high

5.4 Ensemble methods

Randomise-then-optimize (RTO)

True McMC methods We use synthetic data, which were computed as responses of a simple 3-layered earth model, and were inverted using the Metropolis-Hastings algorithm (see Section **??**). As an example, we first explain the time-domain forward modelling script which is supposed to generate a synthetic dataset for a single-site Bayesian simulation, which are shown in the following section. The first block sets system-related parameters, where typical values

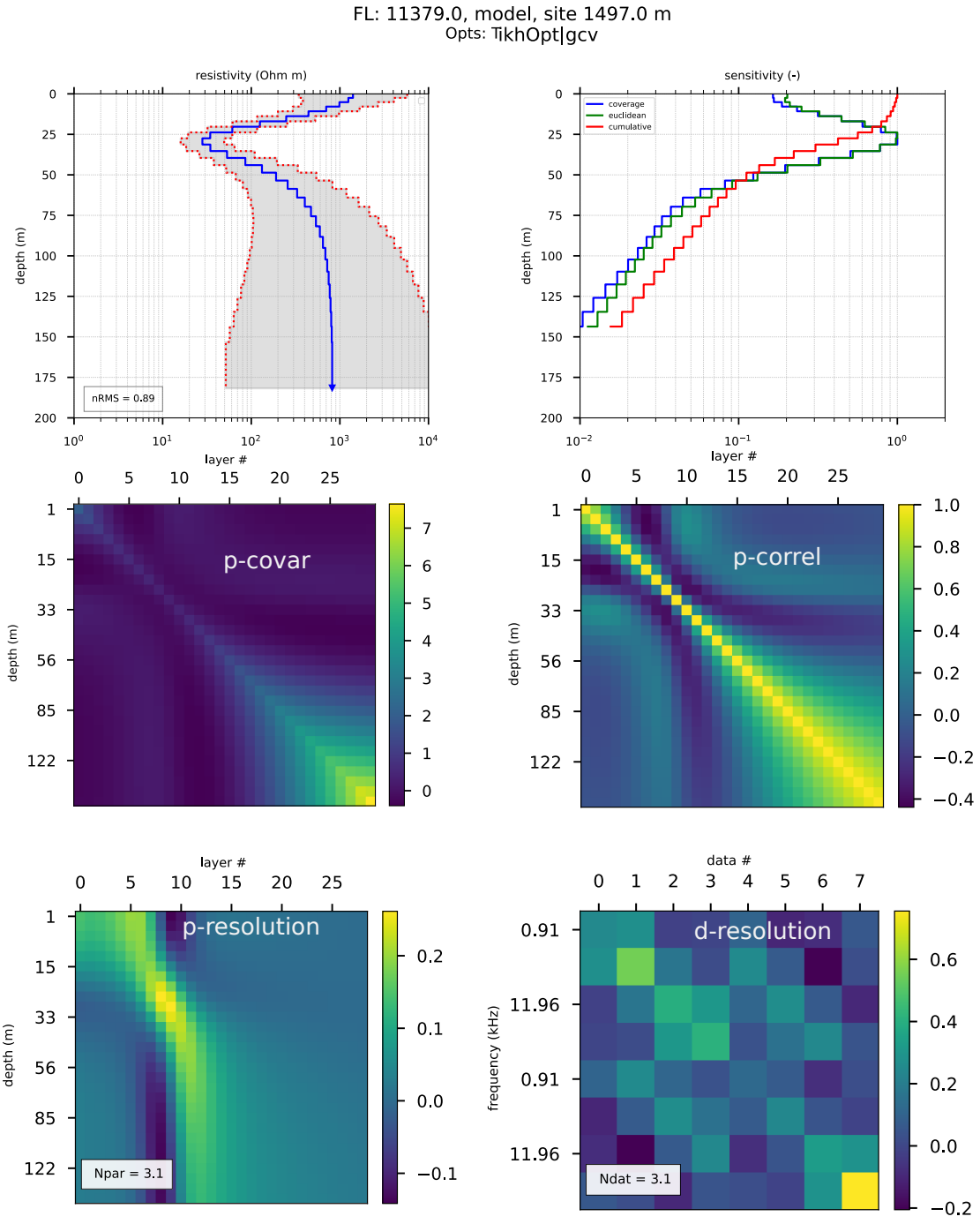

Figure 31: Top right: Different forms of sensitivity as described in the text. Center left: posterior covariance matrix. Centre right: posterior correlation matrix. Bottom: Parameter and data resolution matrices, respectively.

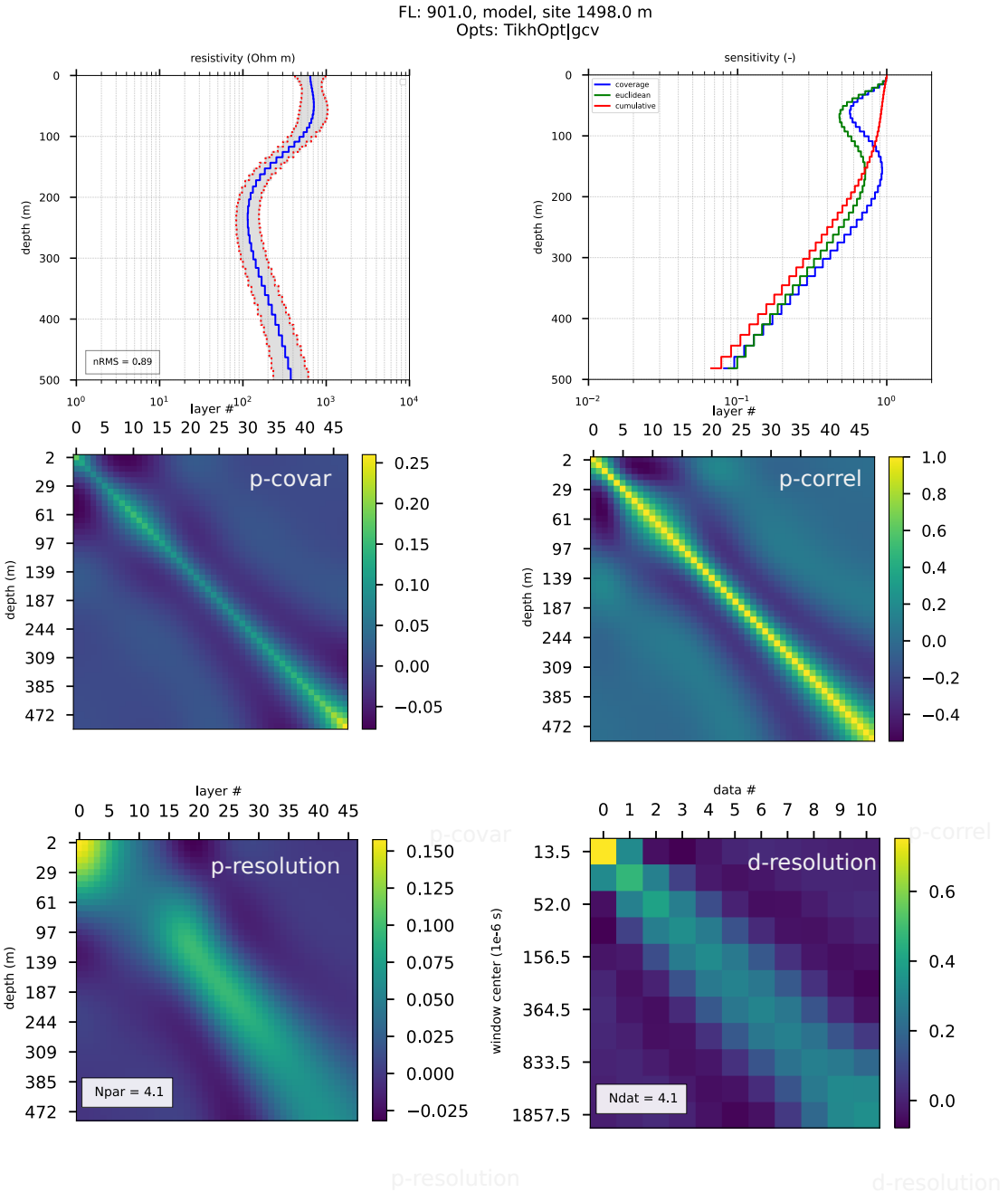

Figure 32: See Figure 31

for altitude (*alt*) and data errors (*data_err*) are chosen. The error adapted corresponds to an estimated mean error for a TEMPEST system, as found in comparable simulations [21]. We assume that the error for GENESIS system should be similar.

```
aem_system = 'genesis'
alt=100. # flight altitude
data_err = 0.01 # assumed data error
runstrng='_'+aem_system+'_3LRES'

nlyr = 3 # number of layer
true_res = ([100.0, 5.0, 100.0]) # resistivity values of the true model
true_res=np.log10(true_res)
true_mu = ([1.0, 1.0, 1.0])
true_ep = ([1.0, 1.0, 1.0])
true_m = ([0.0, 0.0, 0.0])
true_t = ([0.0, 0.0, 0.0])
true_c = ([1.0, 1.0, 1.0])
true_thk = ([50.0, 25.0]) # layer thicknesses
true_depth = np.append( 0., np.cumsum(true_thk))
```

6 Data

In this section, we will concentrate on our recent study of the Tellus data, in particular on the problem of defining reasonable models for the error present in the input data. A few steps for the FD data have already been taken in [75, 76]. The inclusion of the much more problematic TD data made it necessary to investigate further.

6.1 Tellus AEM systems

As of today, the Tellus Programme includes the Airborne EM surveys given in Table 5. One of the important features of this huge data set is the use of two systems, or veven three if the Northern Ireland part is included.The North Midlands survey comprises acquisition of time-domain EM (TDEM) data, which was carried out by CGG Airborne Survey (Pty) Ltd. with the GENESIS system, while the other surveys include acquisition of frequency-domain EM (FDEM) data, which were carried out by Sanders Geophysics Ltd. with the GTK airborne system (AEM-95), and the Joint Airborne Geoscience Capability (JAC) airborne system (AEM-05). This situation is one of the motivations for this research project.

6.2 Frequency-Domain EM Systems - AEM-05 and AEM-95

The Northern Ireland part of the Tellus Programme was carried out with GTK AEM-95 system which had two frequencies 3125 and 14368 Hz), and the vertical, co-planar coils had a wing tip separation of 21.4 m. The other FD surveys were completed using the improved AEM-05 system which was developed as Joint Airborne Geoscience Capability (JAC) by a partnership between the Finnish and British Geological Surveys. The AEM-05 system operates at four frequencies, 912 Hz, 3005 Hz, 11962 Hz, and 24510 Hz, and subsequently operated by Sanders Geophysics Ltd,

Table 5: State of the Tellus AEM survey as of January 2022. Several blocks were reprocessed and delivered to SFI by SGL several times.

Block	System	Report	#Sites	Remark
NI	AEM-05 AEM-95			< 2006, Tellus Northern Ireland
CV	AEM-05			2006, Joint Airborne Geoscience Capability (JAC) pilot studies (Cavan-Monaghan, Silvermines and Castleisland)
TB	AEM-05			2013, Tellus Border (Donegal, Sligo, Leitrim, Cavan, Monaghan, Louth)
NM	CGG GENESIS			2015 North Midlands (Roscommon, Longford, Westmeath)
A1	AEM-05			2016, A1 block, (Meath, Kildare, Offaly, rural Dublin and parts of Laois, Wicklow)
WF	AEM-05			2016, Waterford (Waterford, parts of southern Tipperary and Kilkenny)
A2	AEM-05			2017, A2 block (Galway, including parts of Mayo, Tipperary and Offaly
A3	AEM-05			2018, A3 block (Mayo)
A4	AEM-05			2018 A4 block (Donegal
A5	AEM-05			2019 A5 block (Limerick)
A6	AEM-05			2019 A6 block (West Cork)
A7	AEM-05			2019 A7 block (SE Ireland)
A8/A9	AEM-05			2020/21 A8/A9 blocks (Cork and neighbouring parts of Limerick, Waterford and Tipperary)

(SGL). Their DHC-6 Twin Otter is configured with a four-frequency, wing-tip mounted FDEM system. The transmitter-receiver coil pairs are mounted in a vertical-coplanar orientation which reduces noise by minimizing coupling with the wing tip surface. Additionally, the coils in any one set (transmitter or receiver) are axially offset and are kept adequately separated from each other. The system also comes equipped with a 50/60 Hz power line monitor which becomes particularly useful in identifying cultural interference when surveying in urban settings. The system has a 40 Hz sampling rate which is later decimated to 10 Hz in the processing. Survey acquisition parameters for the FDEM surveys are summarised in Table 6.

Figure 33: Airplane operated by SGL for the FDEM surveys.

Table 6: Summary of FD survey data acquisition parameters.

Flight Line Spacing:	200 m
Flight Line Direction:	SE-NW (≈345°)
Tie Line Spacing:	2000 m
Nominal flying height:	60 m above ground level (rural), up to 240 m (urban)
Data sample intervals:	<10 m (10 Hz)

Apart from the data proper, the latest deliveries provided by SGL include a new Power Line Monitor (PLM). This magnetometer power line monitor data channel included is derived from the 160 Hz magnetic data. These measurements are fed through a frequency-domain band pass filter centred on 3 samples (i.e., 0.01875 s). This step extracts the 50 Hz power line signal that is observed in the magnetometer while suppressing all other signal. The absolute value is taken from the output of that band pass filter and is subsequently passed through a median slope time-domain filter with a window of 8 samples (i.e.,0.05 s), effectively estimating the noise envelope. The magnetometer power line monitor channel is not as susceptible to interference from spurious sources such as radio transmitters and is also able to detect power lines with less current. The magnetometer power line monitor data channel is included with the frequency-domain electromagnetic data. This PLM is supposed to be better than the one provided by SGL previously. Any indicators of potential noisy data is highly desirable, given the dense net of electrical installations in Ireland. We used threshold values between 2 and 10 for this parameter when qualifying the data as “bad”, and potentially excluding them from the inversion process. We find, however, that the use of this parameter has disadvantages, which are connected to the choice of threshold. In may cases it is difficult to find a level which at the same time leads to reasonably short exclusion intervals, and connected gaps. This is due to the sometimes complicated structure of the signatures, and the presence of trends, often at the beginning of flight lines (see Figure **??**). In many case we were not able to see observational effects in zones with high PLM or vice versa - which may be due to the underlying physics (geometry of noise

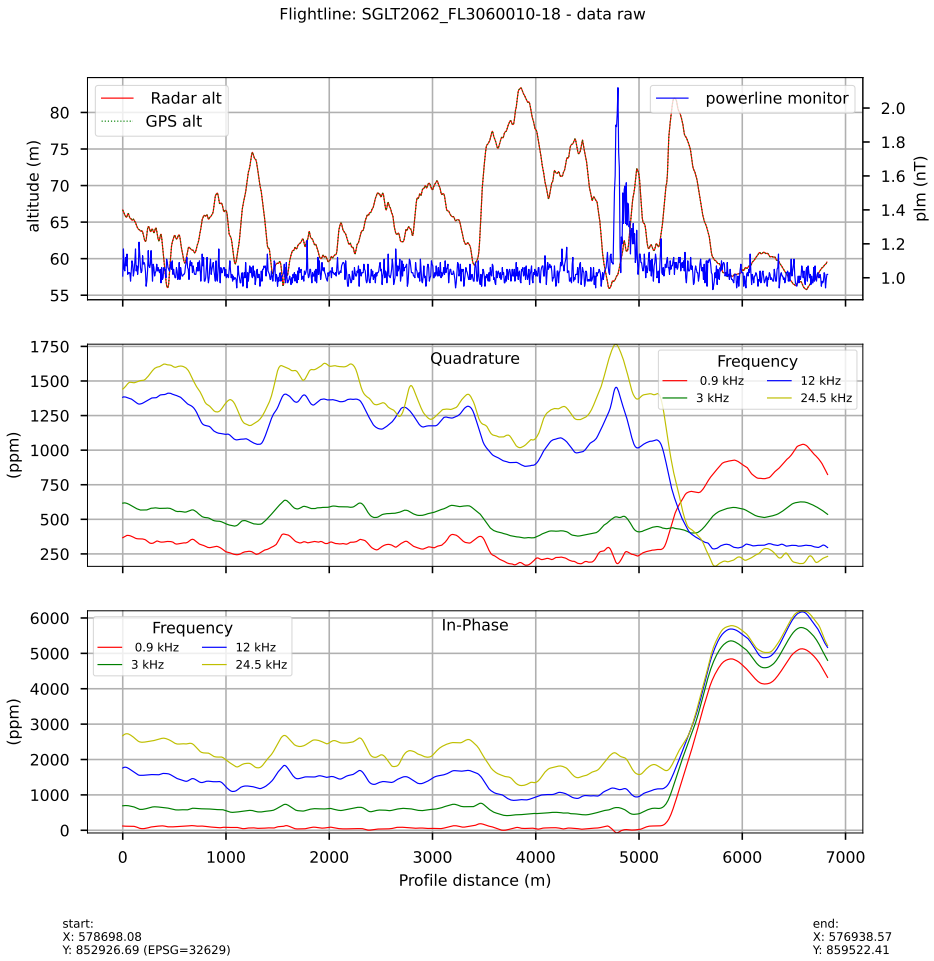

Figure 34: Field data originating from the SGL AEM05 system, from the Bundoran test lines, including the new magnetometric PLM. In this case it is not easy to decide whether the PLM correctly indicates intervals of disturbed data, which need to be left out for inversion.

source, and their activity). This has also been shown in the recent GSI report on data variability [102], which gives a comprehensive study on potential error sources based on nearly co-located test lines. A further study on the behaviour of this data quality indicator on synthetic and field cases would be strongly indicated.

6.3 Time-Domain EM System - CGG GENESIS

For the Northern Midlands (NM) Survey of the Tellus Programme, the AEM data were acquired by CGG Airborne Survey Ltd. between September 2014 and June 2015 using the GENESIS system described by CGG (2015). It is essentially a TEMPEST system, adapted to shallow surveys. The GENESIS 3-axis towed bird assembly provides accurate low noise sampling of the X (horizontal

in-line), Y (horizontal transverse) and Z (vertical) components of the electromagnetic field. However, only the vertical and in-line components were processed and delivered. While this makes sense with respect to modelling, it impedes the application of corrections for potentially important errors related to bird movement, as described for the TEMPEST system [134, 133, 21, 22].

The receiver coils measure the time derivative of the magnetic field, which is integrated during the processing, such that the outcome is in fT. Signals from each axis are transferred to the aircraft through a tow cable specifically designed for its electrical and mechanical properties. The EM transmitter (TX) was mounted above the aircraft with a loop running between the wings and tail mount. The EM receiver (RX) was deployed on a bird 45 m below the aircraft. Survey acquisition parameters are summarised in Table 7.

Figure 35: Airplane operated by CGG for the TDEM survey in the Northern Midlands.

When working with the GENESIS data, there are considerable problems obtaining correct technical information from CGG, who, moreover, have sold their full airborne business (including staff) to a new company, Xcalibur Multiphysics (`https://www.xcaliburmp.com`). The information in report CGG [28], literature [39, 133], and the AirBeo input files received from CGG do not agree. We believe, however, that we have found the correct setup, reproducing the original AirBeo modelling provided by CGG (Figure 36). Furthermore, careful re-checks of the TD Jacobian calculations (including the asinh transform suggested in the literature) show stable behaviour for a wide range of perturbation parameters. Thus we conclude that from the technical viewpoint our implementation of the GENESIS system is correct.

The first observation with the GENESIS data is the extraordinary amount of negative values in the data, particularly in the in-line horizontal component, as seen in Figure 37. A simple counting statistics for Block NM is presented in Table 8. Negative values in TDEM may result from a variety of reasons. They may occur when conditions are not 1-D, which will produce negatives mainly in the X component, as seen in Table 8. For this reason we concentrate on the vertical component (Z) for the 1-D inversions of field data. One-dimensional layered, pure conductivity models can not produce sign-reversals in the transients.

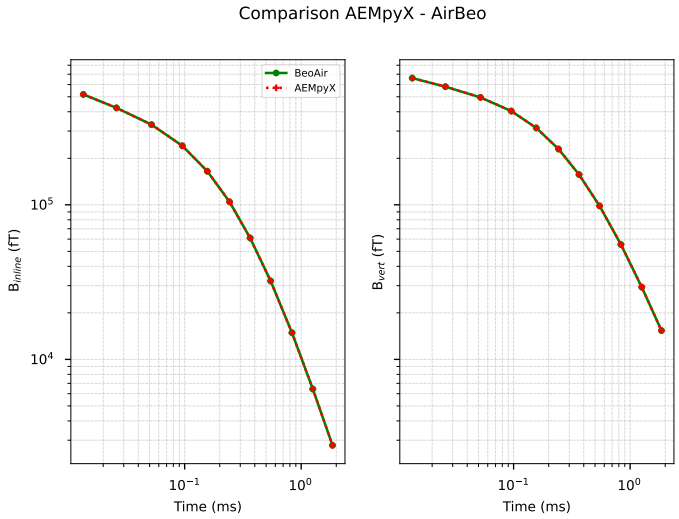

Figure 36: Comparison of original AirBeo forward modelling with AEMpyX core module results. The results are reproduced to full accuracy (14 digits).

Table 7: Summary of TD survey data acquisition parameters.

Flight Line Spacing:	200 m
Flight Line Direction:	SE-NW (≈345°)
Tie Line Spacing:	2000 m
Nominal flying height:	90 m above ground level (rural), 240 m (urban)
Data sample intervals:	≈12 m (5 Hz)

Noise effects apply mainly to the early and late times of the transients, but particularly when negative values in the transient occur a late times, noise may be the reason. If no strong perturbances (as power lines) are present, there may exist effects of induced polarisation (IP), producing negatives all over the time windows. These effects can be used for further characterisation of the subsurface, and are important in ore exploration, or wherever clay minerals are present. In Ireland, black shales are common [see 121]. In principle, the corresponding parameters can be inverted for, though this increases the number of parameters considerably (three more per layer for the Cole-Cole parametrisation). Though their effects have been known for some time [26, 84, 85], super-paramagnetic (SPM) rocks/soils have only lately come into the view of AEM [124, 91, 92]. As with IP, they could be used for further characterisation of the shallow subsurface.

In the case shown in Figure 37 some observations can be made: In both, the vertical (Z) and the in-line (X) component, the last two channels are already noisy, though they in principle characterize deeper structures and should therefore show a smoother character. As the in-line (X) component is considerably smaller than the vertical(Z), the negative values occur mainly in X component. This is consistent with the correlation of the negative data with the flight altitude,

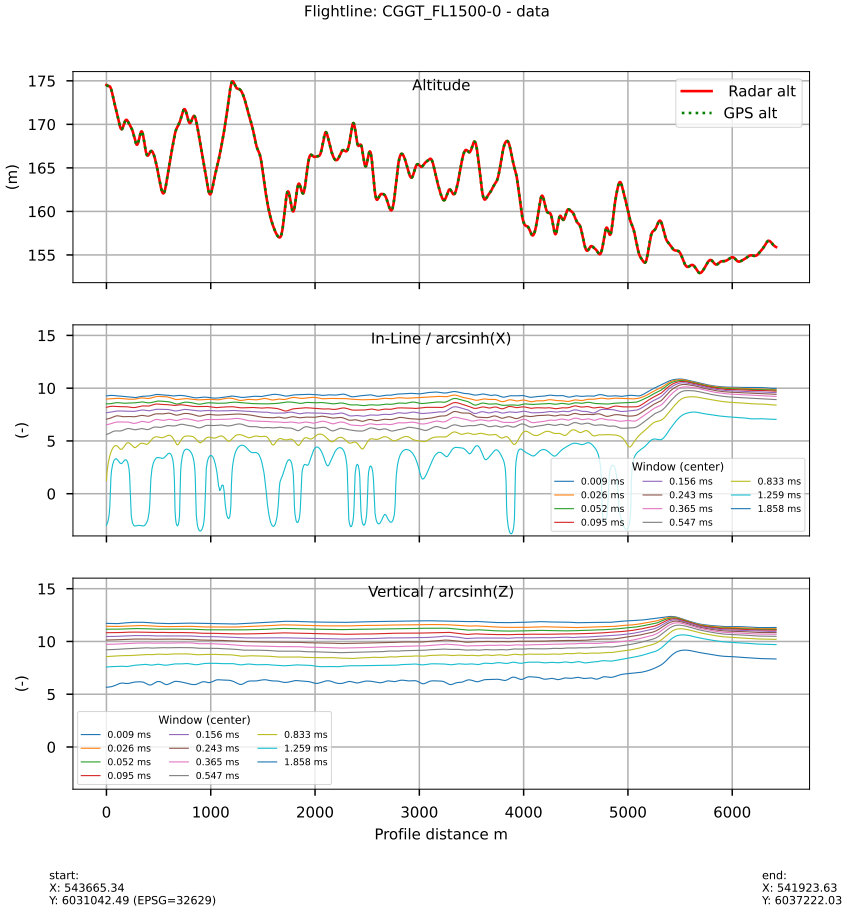

Figure 37: This is an example for field data originating from the CGG GENESIS system. Note that because of the large bandwidth of the data, and the presence of negative values in the in-Line (X) component, the data were transformed using the *asinh* function [129]. Data were taken from the Bonduran test site.

resulting from the exponential decrease of subsurface signal with system flight altitude. While it seems to be possible to work with the Z component (with an appropriate channel-dependent error model), the last two X channels are not usable in most cases. It can also be observed that the flight altitude changes considerably (in all test lines), which could be taken as an indication of problematic atmospheric conditions (see Section 6.4).

It comes clear from the basic physics of the two systems, that their potential depth of investigation and resolving powers are largely different. Figure 38 shows the Jacobian matrices for both systems, which display large difference in structure and values.

Table 8: Summary of GENESIS TD survey data: negative values. The total values are 28.% for X (in-line component), 2.9% for Z (vertical component), and 15.5% for all data.

Window (ms)	X (%)	Z (%)
0.026	7.26	0.96
0.052	15.2	0.13
0.095	5.1	0.075
0.156	12.0	0.28
0.243	19.6	0.47
0.365	31.7	1.12
0.547	43.2	1.89
0.833	41.8	2.56
1.259	47.9	4.92
1.858	84.0	19.8

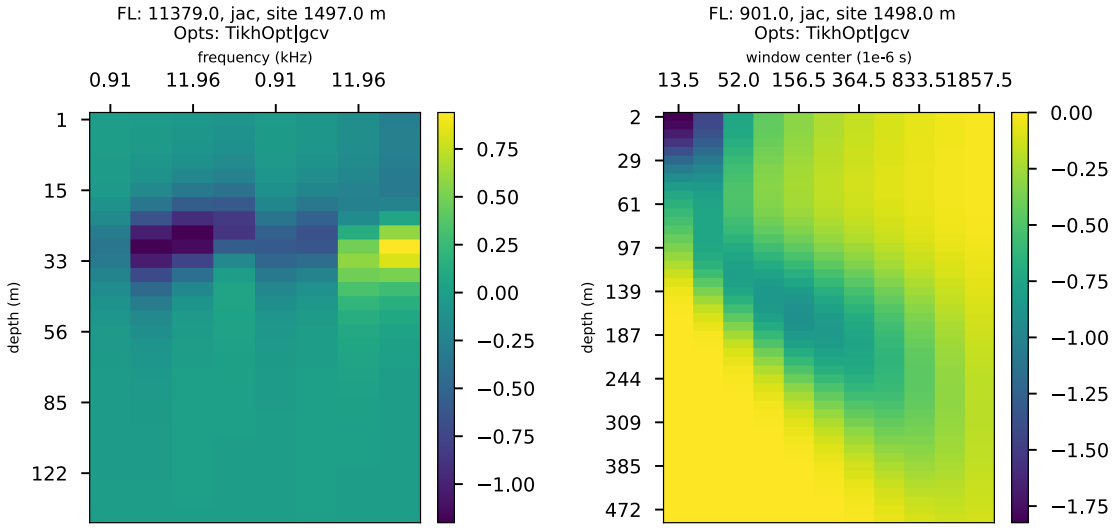

Figure 38: This figure shows an exemplary comparison of Jacobian matrices for TD- and FD systems. Data were taken from the Bonduran test site (TD), and the A1 block (FD).

6.4 Noise in field surveys

The success of Bayesian and other inverse approaches depends strongly on our ability to obtain a realistic noise model. As shown later in Section 3, this choice influences not only the weighting of data-fit and regularisation terms in the objective function (Equation **??**), but also the relative importance within the data set. As the data span several magnitudes, particularly in time-domain methods, a simple additive error will not suffice, and even a combined additive and multiplicative model, as implemented in `AEMpyX` may not be enough. It has to be kept in mind that any error model used must be seen as a hypothesis, which will require revisions whenever new information is available, as also corroborated by the recent variability study [102].

[76] has shown that the errors estimated from the Bundoran high-fly experiment are spatially (and thus also temporally) correlated [76]. The correlation length for these perturbation is in the order of a few 100 m, corresponding to some seconds in time. No real frequency dependency could be detected, and for the real inversions we originally assumed a constant additive error of 30 ppm to 70 ppm, which always led to reasonable convergence an a good data fit. In some cased this was complemented with a 3% to 5% multiplicative error, in order to inhibit over-fitting the largest values. In view of the more recent study by Muller [102], this may be slightly low, with about 100 ppm more realistic within this framework.

In the Tellus project, error estimation was attempted by flying test lines for both systems in the area of Bundoran (Co Sligo), and later for the FD systems in the Waterford area. Kiyan and Rath [75] and Kiyan et al. [76] present an analysis of the so-called high-fly experiment for the AEM05 system at Bundoran. In contrary to a commonly made assumption [70, 62], we think that the noise is not purely random, but is the result of neglected, unconsidered, or even unknown physical processes. Here, it is plausible to think that properties of the atmosphere are involved. Heterogeneities of atmospheric or wind properties are the probably best candidates, which may in turn will be influenced by surface conditions (e. g., canopy). As already suggested by Muller [102], a detailed surface field study may be useful.

The very procedure of the high-fly experiment [54] has some inherent weaknesses. It depends on the assumption that noise is independent of the flight altitude, and can thus be separated from the signal part originating from the subsurface. As we think the “noise” is related to physical processes, this is certainly not true. The altitude range between 50 m and 300 m covers the lower part of the atmospheric boundary layer (ABL), is well-known to be highly heterogeneous in space and time [e.g., 137, 136, 139, 86] . While a direct influence on the measurements is not easy to conceive, the heterogeneities in atmospheric conditions including turbulence may produce perturbations in movements of the plane (i.e., pitch, roll, yaw) for both systems, and the relative motion of the bird for GENESIS. These parameters unfortunately are not available in the Tellus database. If our hypothesis is correct, they might be used for a correction during the pre-inversion processing. As far as we know, only the plane movement has been corrected for the GENESIS system [28]. The problem is aggravated by the choice of the test line area, which is in a land/sea breeze zone, which is particularly prone to changes in ABL structure and turbulences [149, 5]. This can be nicely observed in the GENESIS data shown in Figure **??**, where the difficulty of keeping a constant altitude is obvious. The spatial nd temporal correlation found for the FD observations [76] is not far from the scales of turbulence in the lower ABL [1]. No local weather station are available for both test line zones. For these reasons, a repeatability approach [70, 62] may be more adequate for determining a “realistic” noise model, as proposed by Green and Lane [54] for the TEMPEST system. High-fly estimates of additive noise seem possible for both systems, and a number of repeats for determining the multiplicative errors are available, though their number is low.

For the frequency-domain the setup is similar:

To generate a synthetic frequency-domain dataset, a three-layered resistivity model was used. The model comprises a first layer with resistivity 100 Ωm (15 m), a second layer with resistivity 5 Ωm (25 m), and a lower half-space with resistivity 100 Ωm.

```
1         aem_system = 'aem05'
2         alt = 61. # flight altitude
3         data_err = 30 # assumed data error
```

```
runstrng='_'+aem_system+'_3LRES'

nlyr = 3 # number of layers
true_res = ([100.0, 5.0, 100.0]) # resistivity values of the true model
true_res=np.log10(true_res)
true_mu = ([1.0, 1.0, 1.0])
true_ep = ([1.0, 1.0, 1.0])
true_m = ([0.0, 0.0, 0.0])
true_t = ([0.0, 0.0, 0.0])
true_c = ([1.0, 1.0, 1.0])
true_thk = ([15.0, 25.0]) # layer thicknesses
true_depth = np.append( 0., np.cumsum(true_thk))
```

The outputs of these scripts are used as the "true model" in the following.

The python scripts associated with this section are `Tutorial4_FWD-td.py`, `Tutorial4_MH-td.py`, and corresponding plotting routines.

The setup is very similar to the flight line inversion described in the previous section. We start with the control parameters for the algorithm, followed by the data configuration and the prior and initial model. The MH algorithm needs the specification of a step size, which should be adapted to obtain a reasonable acceptance rate. It is done here by decreasing the step size if the acceptance rate is less than 30%, and by increasing the step size if the rate is greater than 60%.

```
# Metropolis-Hastings with 1000000 samples
nsample = 1000000
nburnin = 1000

# define stepsize of MCMC
accp_check = 1000
accp_max = 60.
accp_min = 30.
step0 = .05
stepfac = 0.6666

# import true data
aem_system = 'genesis'
Datafile= 'FWDModel_'+aem_system+'_3LRES5.npz'
tmp= np.load(Datafile)
data_true = tmp['data_true']
data_err = tmp['data_err'] # 0.01 assumed in this example

# Gaussian random noise added
# for this example taken from input

data_obs = data_obs + data_err*np.random.randn(1, sizedat[0])

# setting up mesh and prior model
nlyr = 20 # enter number of layers
mactive, prior_avg, prior_std, m_upper, m_lower = inverse.initialize_1dmod(nlyr)
mactive[0*nlyr:1*nlyr] = ionlyr[0:nlyr] # resistivity activated
```

```
28
29          dzstart = .7 # setting up layer thicknesses
30          dzend = .7
31          dz = np.logspace(dzstart,dzend,nlyr-1)+0.5*np.random.randn(nlyr-1)
32          z = np.append( 0., np.cumsum(dz))
33
34          guess_r = 100. # initial guess for resistivity in prior_avg
35          guess_mu = 1.0 # relative magnetic permeability
36          guess_ep = 1.0 # relative dielectric constant
37          guess_m = 0.0 # chargeability
38          guess_t = 0.0 # time constant
39          guess_c = 1.0 # freq c constant
40
41          prior_avg[0*nlyr:1*nlyr] = np.log10(guess_r*onlyr[0:nlyr])
42          prior_avg[1*nlyr:2*nlyr] = guess_mu*onlyr[0:nlyr]
43          prior_avg[2*nlyr:3*nlyr] = guess_ep*onlyr[0:nlyr]
44          prior_avg[3*nlyr:4*nlyr] = guess_m*onlyr[0:nlyr]
45          prior_avg[4*nlyr:5*nlyr] = guess_t*onlyr[0:nlyr]
46          prior_avg[5*nlyr:6*nlyr] = guess_c*onlyr[0:nlyr]
47          prior_avg[6*nlyr:7*nlyr-1] = dz[0:nlyr-1]
48
49          # prior_std defines standard deviation of prior_avg
50          guess_sr=0.2 # for resistivity
51          res_std=guess_sr*onlyr[0:nlyr];
52          prior_std[0*nlyr:1*nlyr] = res_std;
```

Coherency is imposed by a (potentially three-dimensional) Markovian covariance, which is used to generate the prior model, from which the samples are chosen. The correlation length in Equation **??** is L=30 m. Results from the MCMC simulations are presented in Figure 40, which shows the posterior distribution of the models.

The python scripts associated with this section are `Tutorial4_FWD-fd.py`, `Tutorial4_MH-fd.py`, and corresponding plotting routines.

For frequency-domain, we have chosen a different setup. In contrast to the time-domain example, which is a highly-parameterised model with fixed layer thicknesses, here we simulate both layer thicknesses and resistivities, however only with 3 layers.

```
1           # Metropolis-Hastings with 1000000 samples
2           nsample = 1000000
3           nburnin = 100         # define stepsize of MCMC
4           accp_check = 1000
5           accp_max = 60.
6           accp_min = 30.
7           step0 =1.
8           stepfac=0.6666
9           mode=0
10          onlyactive = False
11
12          # import true data
13          aem_system = 'aem05'
```

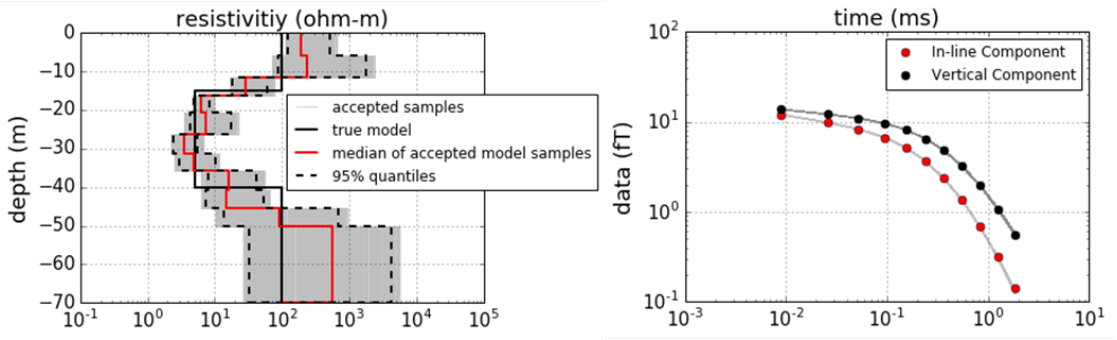

Figure 39: Figure shows a summary of the output from the Metropolis-Hastings (MH) simulation. The MH algorithm was run for 100,000 model samples and ≈40% of those samples were accepted. The left panel shows every 50th of accepted sample models (light grey lines). Superimposed are the true model (black line), bounds that contain 95% of the accepted models black dashed lines), and the median resistivity values (red line). The right panel illustrates the data fit in which the predicted data are represented with light and dark grey lines, and the observed data are represented with solid circles.

```
        Datafile= 'FWDModel_'+aem_system+'_3LRES5.npz'
        data_true = tmp['data_true']
        data_err = tmp['data_err'] # 30 ppm assumed for this example
        data_obs = np.ndarray.flatten(data_true)
        sizedat = np.shape(data_obs);

        # Gaussian random noise added to data true
        data_obs = data_obs + data_err*np.random.randn(1, sizedat[0])

        # setting up prior model
        nlyr = 3
        mactive, prior_avg, prior_std, m_upper, m_lower = inverse.initialize_1dmod(nlyr)
        mactive[0*nlyr:1*nlyr] = ionlyr[0:nlyr] # resistivity is activated
        mactive[6*nlyr:7*nlyr-1] = ionlyr[0:nlyr-1] # layer thickness is activated
        dzstart = 1.7
        dzend = 1.7
        dz = np.logspace(dzstart,dzend,nlyr-1)+0.5*np.random.randn(nlyr-1)
        z = np.append( 0., np.cumsum(dz))

        guess_r = 100. # initial guess for resistivity in prior_avg
        guess_mu = 1.0 # relative magnetic permeability
        guess_ep = 1.0 # relative dielectric constant
        guess_m = 0.0 # chargeability
        guess_t = 0.0 # time constant
        guess_c = 1.0 # freq c constant

        prior_avg[0*nlyr:1*nlyr] = np.log10(guess_r*onlyr[0:nlyr])
        prior_avg[1*nlyr:2*nlyr] = guess_mu*onlyr[0:nlyr]
        prior_avg[2*nlyr:3*nlyr] = guess_ep*onlyr[0:nlyr]
        prior_avg[3*nlyr:4*nlyr] = guess_m*onlyr[0:nlyr]
```

```
        prior_avg[4*nlyr:5*nlyr] = guess_t*onlyr[0:nlyr]
        prior_avg[5*nlyr:6*nlyr] = guess_c*onlyr[0:nlyr]
        prior_avg[6*nlyr:7*nlyr-1] = dz[0:nlyr-1]

        #prior_std defines standard deviation of prior_avg
        guess_sr = 0.2 # for resistivity
        res_std = guess_sr*onlyr[0:nlyr];
        res_var = res_std*res_std

        guess_sz = 5. # for layer thickness
        dz_std = guess_sz*onlyr[0:nlyr-1];
        dz_var = dz_std*dz_std
```

Results from the MCMC simulations can be found in Figure 40, which shows the posterior distribution of the models.

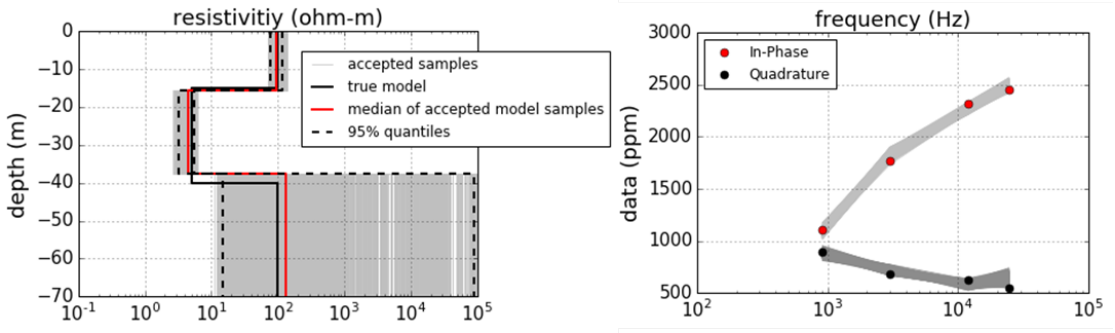

Figure 40: Figure shows a summary of the output from the Metropolis-Hastings (MH) simulation. The MH algorithm was run for 1,000,000 model samples and ≈40% of those samples were accepted. The left panel shows every 500th of accepted sample models (light grey lines). Superimposed are the true model (black line), bounds that contain 95% of the accepted models black dashed lines), and the median resistivity values (red line). The right panel illustrates the data fit in which the predicted data are represented with light and dark grey lines, and the observed data are represented with solid circles.

For practical use with data we suggest to use one of the more sophisticated and efficient methods as already mentioned in the Section 1 [80, 47, 96, 93]. Here we will present a worked out script using the DRAM methodology [57], as implemented by Miles [96]. We have chosen this one, because it was used by one of the authors already extensively for paleoclimatic inversions [79, 78, 116].

Acknowledgments

This work was funded by the Geological Survey of Ireland GSI under grants *sc2015-004* and *sc2015-004*. We would like to acknowledge the GSI Tellus team, who were most helpful and co-operative. Australia's CSIRO and the Amira International consortium is thanked for making their P223 modelling suite open access. We are especially indebted to David O'Leary and Eve Daly from NUIG Galway, and our colleagues from DIAS who helped with the ERT fieldwork.

Bibliography

[1] P. Alekseychik, G. Katul, I. Korpela, and S. Launiainen. Eddies in motion: visualizing boundary-layer turbulence above an open boreal peatland using uas thermal videos. *Atmospheric Measurement Techniques*, 14(5):3501–3521, 2021. doi: 10.5194/amt-14-3501-2021. 15, 73

[2] M. An. On resolution matrices. *Pure and Applied Geophysics*, 2022. 25

[3] W. Anderson. Calculation of transient soundings for a coincident loop system (program tcoloop). Technical report, USGS, 1982. 18

[4] W. L. Anderson. Improved digital filters for evaluating fourier and hankel transform integrals. Open file report, USGS, Denver, CO, 1975. 18

[5] J. A. Arrillaga-Mitxelena. *Thermally-driven Mesoscale Flows and their Interaction with Atmospheric Boundary Layer Turbulence*. PhD thesis, Universidad Complutense de Madrid, Spain, 2020. 15, 73

[6] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter Estimation and Inverse Problems*. Elsevier, Amsterdam, Netherlands, 2013. 5

[7] R. C. Aster, B. Borchers, and C. H. Thurber. *Parameter Estimation and Inverse Problems*. Elsevier, Amsterdam, Netherlands, 3rd edition, 2019. 15, 21, 25

[8] J. M. Bardsley, A. Solonen, H. Haario, and M. Laine. Randomize-then-optimize: a method for sampling from posterior distributions in nonlinear inverse problems. *SIAM J. Sci. Comp.*, 36(4):A1895–A1910, 2014. doi: http://dx.doi.org/10.1137/140964023. 5, 26

[9] J. M. Bardsley, A. Seppänen, A. Solonen, H. Haario, and J. Kaipio. Randomize-then-optimize for sampling and uncertainty quantification in electrical impedance tomography. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):1136–1158, 2015. doi: 10.1137/140978272. 5, 26

[10] S. Bayat. Resolution analyses for the tellus airborne electromagnetic data compared with co-located electrical resistivity tomography data. Master's thesis, School of Natural Sciences, National University of Ireland − Galway, Ireland, 2023. 26

[11] D. Beamish and H. Levaniemi. The canopy effect in aem revisited: investigations using laser and radar altimetry. *Near Surface Geophysics*, 8(2):103–115, 2010. 43

[12] P. A. Bedrosian, C. Schamper, and E. Auken. A comparison of helicopter-borne electromagnetic systems for hydrogeologic studies. *Geophysical Prospecting*, pages n/a–n/a, 2015. doi: 10.1111/1365-2478.12262. URL `http://dx.doi.org/10.1111/1365-2478.12262`. 26

[13] J. Bertin and J. Loeb. *Experimental and Theoretical Aspects of Induced Polarization, Vol. 1 and 2*. Borntraeger, Berlin, 1976. 16

[14] A. Binley and L. Slater. *Resistivity and Induced Polarization: Theory and Applications to the Near-Surface Earth*. Cambridge University Press, 2020. 16, 23

[15] G. Blanchy, S. Saneiyan, J. Boyd, P. McLachlan, and A. Binley. ResIPy, an intuitive open source software for complex geoelectrical inversion/modeling. *Computers & Geosciences*, page 104423, 2020. doi: 10.1016/j.cageo.2020.104423. 23

[16] D. Blatter, M. Morzfeld, K. Key, and S. Constable. Uncertainty quantification for regularized inversion of electromagnetic geophysical data - part ii: application in 1-d and 2-d problems. *Geophysical Journal International*, 231(2):1–21, 2022. doi: 10.1093/gji/ggac242. 5, 26

[17] D. Blatter, M. Morzfeld, K. Key, and S. Constable. Uncertainty quantification for regularized inversion of electromagnetic geophysical data—Part I: motivation and theory. *Geophysical Journal International*, 231(2):1057–1074, 2022. doi: 10.1093/gji/ggac241. 5, 26

[18] C. Bobe, E. Van De Vijver, J. Keller, D. Hanssens, M. Meirvenne, and P. De Smedt. Probabilistic 1-d inversion of frequency-domain electromagnetic data using a kalman ensemble generator. *IEEE Transactions on Geoscience and Remote Sensing*, PP, 2019. doi: 10.1109/TGRS.2019.2953004. 5

[19] C. Bobe, D. Hanssens, T. Hermans, and E. Van De Vijver. Efficient probabilistic joint inversion of direct current resistivity and small-loop electromagnetic data. *Algorithms*, 13 (6), 2020. doi: 10.3390/a13060144. 5

[20] C. S. Bobe. *Efficient probabilistic processing of frequency-domain electromagnetic data for subsurface modelling*. PhD thesis, Ghent University, Faculty of Bioscience Engineering, 2020. 5

[21] R. Brodie. GALEISBSTDEM: a deterministic algorithm for 1D sample by sample inversion of time-domain AEM data — theoretical details. Technical report, Geoscience Australia, 2015. 9, 19, 51, 65, 69

[22] R. Brodie. User manual for geoscience australia's airborne electromagnetic inversion software. Technical report, Geoscience Australia, 2016. 9, 69

[23] R. C. Brodie. *Holistic Inversion of Airborne Electromagnetic Data*. PhD thesis, Research School of Earth Sciences, Australian National University, 2010. 18

[24] S. Brooks, A. Gelman, G. Jones, and X. Meng. Handbook of Markov chain Monte Carlo. *CRC Press*, 2011. 28

[25] M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, Cambridge MA, USA, 2003. 39

[26] G. Buselli. The effect of near-surface superparamagnetic material on electromagnetic measurements. *Geophysics*, 47(09):1315–1324, 1982. 12, 70

[27] CGG. Geoscience Australia TEMPEST Geophysical Survey - Capricorn Regional Survey, Western Australia. Logistics and Processing Report 2446, CGG Aviation (Australia) Pty Ltd, 2014. 19

[28] CGG. Logistics and processing report. genesis airborne electromagnetic, magnetic and radiometric geophysical survey for Geological Survey of Ireland, project number: AIRJ262. Technical report, CGG, 2015. 10, 15, 69, 73

[29] N. Chada and X. Tong. Convergence acceleration of ensemble Kalman inversion in nonlinear settings. *Mathematics of Computation*, 91:1247–1280, 2022. doi: 10.1090/mcom/3709. 28

[30] N. K. Chada, M. A. Iglesias, L. Roininen, and A. M. Stuart. Parameterizations for ensemble kalman inversion. *Inverse Problems*, 34:055009, 2018. doi: 10.1088/1361-6420/aab6d9. 28

[31] N. K. Chada, A. M. Stuart, and X. T. Tong. Tikhonov regularization within ensemble Kalman inversion. *SIAM Journal on Numerical Analysis*, 58(2):1263–1294, 2020. 28

[32] N. B. Christensen. Optimized fast Hankel transform filters. *Geophysical Prospecting*, 38: 545–568, 1990. 18

[33] A. V. Christiansen, E. Auken, Y. Ley-Cooper, and K. A.en. Quantifying the effect of primary field modelling on TEMPEST data - the importance of uncertainty. In *ASEG-PESA-AIG 2016. 25th Geophysical Conference & exhibition. August 21–24, 2016, Adelaide, Australia*, 2016. 19

[34] E. Cleary, A. Garbuno-Inigo, S. Lan, T. Schneider, and A. M. Stuart. Calibrate, emulate, sample. *Journal of Computational Physics*, 424:109716, 2021. doi: 10.1016/j.jcp.2020.109716. 5

[35] K. Cole and R. Cole. Dispersion and absorption in dielectrics - I alternating current characteristics. *J. Chem. Phys.*, 9:341–352, 1941. doi: 10.1063/1.1750906. 16

[36] R. Delhaye. *Magnetotelluric Research of the Rathlin Basin*. PhD thesis, Faculty of Science, National University of Ireland, Galway, 2018. 23

[37] J. E. Doherty, R. J. Hunt, and M. J. Tonkin. Approaches to highly parameterized inversion: A guide to using pest for model-parameter and predictive-uncertainty analysis. Scientific Investigations Report 2010?5211, USGS, Reston, VA, 2010. URL `http://pubs.er.usgs.gov/publication/sir20105168`. 5

[38] A. Doicu, T. Trautmann, and F. Schreier. *Numerical regularization for atmospheric inverse problems*. Springer, Berlin, DE, 2010. 15, 49

[39] D. Eberle. Flying the time domain electromagnetic GENESIS system over the traditional Welkom gold mining district, Free State, South Africa, to identify contaminant seepage and acid mine drainage. In *13th SAGA Biennial Conference and Exhibition*, 2010. 10, 69

[40] B. Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, Philadelphia, 1982. 5, 23, 27

[41] B. Efron and T. Hastie. *Computer Age Statistical Inference. Algorithms, Evidence, and Data Science Stanford University*. Cambridge University Press, 2016. 5, 27

[42] B. Efron and C. Stein. The jackknife estimate of variance. *Annals Stat.*, 9:586–596, 1981. 5, 27

[43] G. Evensen, F. C. Vossepoel, and P. J. van Leeuwen. *Data Assimilation Fundamentals. A Unified Formulation of the State and Parameter Estimation Problem*. Spinger, Cham CH, 2022. 28

[44] C. G. Farquharson and D. W. Oldenburg. Non-linear inversion using general measures of data misfit and model structure. *Geophysical Journal International*, 134(1):213–227, 1998. doi: 10.1046/j.1365-246x.1998.00555.x. 20

[45] G. Fiandaca, L. M. Madsen, and P. K. Maurya. Re-parameterisations of the Cole-Cole model for improved spectral inversion of induced polarization data. *Near Surface Geophysics*, 16(4):385–399, 2018. doi: 10.3997/1873-0604.2017065. 6, 16, 50

[46] A. Fichtner, A. Zunino, L. Gebraad, and C. Boehm. Autotuning hamiltonian monte carlo for efficient generalized nullspace exploration. *Geophys J Int*, 227(2):941–968, 2021. doi: 10.1093/gji/ggab270. URL `https://doi.org/10.1093/gji/ggab270`. 5

[47] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman. emcee: The mcmc hammer. *PASP*, 125:306–312, 2013. doi: 10.1086/670067. 5, 29, 77

[48] S. Friedel. Resolution, stability and efficiency of resistivity tomography estimated from a generalized inverse approach. *Geophysical Journal International*, 153(2):305–316, 2003. doi: 10.1046/j.1365-246X.2003.01890.x. 23, 25

[49] A. Garbuno-Inigo, F. Hoffmann, W. Li, and A. M. Stuart. Interacting Langevin diffusions: Gradient structure and Ensemble Kalman Sampler. *Siam J. Applied Dynamical Systems*, 19(1):412–441, 2020. doi: 10.1137/19M1251655. 5

[50] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. 28

[51] R. Ghanem, D. Higdon, and H. Owhadi. *Handbook of Uncertainty Quantification*. Springer, Cham, Switzerland, 2017. ISBN 978-3-319-12384-4. doi: 10.1007/978-3-319-12385-1. URL `http://link.springer.com/10.1007/978-3-319-12385-1`. 28

[52] G. Golub and C. F. van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, 3rd ed. edition, 1996. ISBN 0-8018-5414-8. 44

[53] F. S. Grant and G. F. West. *Interpretation Theory in Applied Geophysics*. McGraw-Hill, New York, USA, 1965. 16

[54] A. Green and R. Lane. Estimating noise levels in aem data. In *Proceedings, ASEG 16th Geophysical Conference and Exhibition, February 2003, Adelaide.*, pages 1139–1141. Soc. of Expl. Geophys., 2003. 14, 15, 73

[55] T. Günther. *Inversion Methods and Resolution Analysis for the 2D/3D Reconstruction of Resistivity Structures from DC Measurements*. PhD thesis, Fakultät für Geowissenschaften, Geotechnik und Bergbau, Technische Universitä Bergakademie Freiberg, 2004. 23

[56] T. Günther, S. Friedel, and K. Spitzer. Estimation of information content and efficiency for different data sets and inversion schemes using the generalized singular value decomposition. In *20. Kolloquium Elektromagnetische Tiefenforschung, Königstein, 29.09.-3.10.2003*, 2003. 23

[57] H. Haario, M. Laine, A. Mira, and E. Saksman. Dram: Efficient adaptive mcmc. *Statistics and Computing*, 16:339–354, 2006. URL `http://dx.doi.org/10.1007/s11222-006-9438-0`. 5, 77

[58] P. C. Hansen. *Rank Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia PA, USA, 1998. 6, 15, 49

[59] P. C. Hansen. *Discrete inverse problems: Insight and algorithms*, volume 7 of *Fundamentals of Algorithms*. Society for Industrial and Applied Mathematics (SIAM), 2010. ISBN 978-0-898716-96-2. 15

[60] P. C. Hansen and D. P. O'Leary. The use of the L-curve in the regularization of discrete ill-posed problems. *SIAM J. Sci. Comput.*, 14:1487–1503, 1993. 6

[61] A. Haroon. *Development of Novel Time-domain Electromagnetic Methods for Offshore Groundwater Studies: A Data Application from Bat Yam, Israel*. PhD thesis, Mathematisch-Naturwissenschaftliche Fakultät der Universität zu Köln, 2016. 6

[62] A. Hegarty, G. Stanley, E. Kashdan, J. Hodgson, and A. C. Parnell. Repeatability analysis of airborne electromagnetic surveys. *Mathematics-in-Industry Case Studies*, 7(1):6, 2016. doi: 10.1186/s40929-016-0008-1. 14, 15, 73

[63] D. W. Hohmann. Numerical modeling for electromagnetic methods in geophysics. In M. N. Nabighian, editor, *Electromagnetic Methods in Applied Geophysics. Volume 1, Theory*, pages 314–362. SEG, Tulsa OK, USA, 1987. 16

[64] P. J. Huber. *Robust Statistics*. J. Wiley, New York, 1981. 20

[65] M. Iglesias and Y. Yang. Adaptive regularisation for ensemble kalman inversion. *Inverse Problems*, 37(2):025008, 2021. doi: 10.1088/1361-6420/abd29b. 28

[66] M. A. Iglesias. A regularizing iterative ensemble kalman method for pde-constrained inverse problems. *Inverse Problems*, 32(2):025002, 2016. 28

[67] M. A. Iglesias, K. J. H. Law, and A. M. Stuart. Ensemble kalman methods for inverse problems. *Inverse Problems*, 29(4):045001, 2013. 5, 28

[68] S. R. James, N. L. Foks, and B. J. Minsley. GSPy: A new toolbox and data standard for geophysical datasets. *Frontieers in Earth Sciences*, 10:907614, 2022. doi: 10.3389/feart.2022.907614. 39

[69] M. Karamanis, F. Beutler, and J. A. Peacock. zeus: A python implementation of ensemble slice sampling for efficient bayesian parameter inference. *arXiv preprint arXiv:2105.03468*, 2021. 29

[70] E. Kashdan, A. Hegarty, A. Parnell, D. Cellai, S. Gazzola, A. Gloster, A. Faqeeh, A. S. N.en, and K. O'Sullivan. Analysis and interpretation of tellus border electromagnetic data. In *102nd European Study Group with Industry, University College Dublin, Ireland, 30 June – 4 July 2014*, 2014. 14, 15, 73

[71] A. A. Kaufman, D. Alekseev, and M. Oristaglio. *Principles of Electromagnetic Methods in Surface Geophysics*. Elsevier, Amsterdam, NL, 2014. 5, 16, 18

[72] G. V. Keller and F. C. Frischknecht. *Electrical methods in geophysical prospecting*, volume Keller,. Pergamon Press, Boulder CO, USA, 1966. 16, 18

[73] K. Key. Is the fast Hankel transform faster than quadrature? *Geophysics*, 77(3):F21–F30, 2012. doi: 10.1190/geo2011-0237.1. 18

[74] H. Kim, D. Sanz-Alonso, and A. Strang. Hierarchical ensemble Kalman methods with sparsity-promoting generalized gamma hyperpriors. *arXiv preprint arXiv:2205.09322*, 2022. 28

[75] D. Kiyan and V. Rath. *Inverse Methods for Airborne Electromagnetic Data from the Tellus Surveys. The* `aempy` *Toolbox*. Dublin Institute for Advanced Studies (DIAS), 2017. 7, 14, 65, 73

[76] D. Kiyan, V. Rath, M. R. Muller, M. D. Ture, and J. Hodgson. 1-D inversion of frequency-domain airborne electromagnetic data using the open-source `aempy` toolbox. *Journal of Applied Geophysics*, 198:104562, 2022. doi: 10.1016/j.jappgeo.2022.104562. 7, 13, 14, 15, 65, 73

[77] D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo methods*. Wiley, Hoboken, 2011. doi: 10.1002/9781118014967. URL `http://dx.doi.org/10.1002/9781118014967`. 28

[78] I. Kukkonen, V. Rath, and A. Korpisalo. Paleoclimatic inversion of ground surface temperature history from geothermal data on the Olkiluoto drill hole OL-KR56. Working report 2015-49, Posiva Oy, Olkiluoto, Eurajoki, Finland, 2015. 77

[79] I. T. Kukkonen, V. Rath, L. Kivekäs, J. Šafanda, and V. Cermak. Geothermal studies of the outokumpu deep drill hole, finland: vertical variation in heat flow and paleoclimatic implications. *Phys. Earth Planet. Inter.*, 188:9–25, 2011. doi: 10.1016/j.pepi.2011.06.002. 77

[80] E. Laloy and J. A. Vrugt. High-dimensional posterior exploration of hydrologic models using multiple-try DREAM(ZS) and high-performance computing. *Water Resour. Res.*, 48: W01526, 2012. doi: 10.1029/2011WR010608. 5, 77

[81] C. Lanczos. *Linear Differential Operators*. D. Van Nostrand Co. Ltd, 1961. 44

[82] R. Lane, A. Green, C. Golding, M. Owers, P. Pik, C. Plunkett, D. Sattel, and B. Thorn. An example of 3D conductivity mapping using the TEMPEST airborne electromagenetic system. *Exploration Geophysics*, 31(1/2):162–172, 2000. 18, 19

[83] B. J. Last and K. Kubik. Compact gravity inversion. *Geophysics*, 48, 1983. doi: 10.1190/1.1441501. 19

[84] T. Lee. The effect of a superparamagnetic layer on the transient electromagnetic response of a ground. *Geophys. Prosp.*, 32(03):480–496, 1984. 12, 70

[85] T. Lee. The transient electromagnetic response of a magnetic or superparamagnetic ground. *Geophysics*, 49(7):854–860, 1984. doi: 10.1190/1.1441731. 12, 70

[86] X. Lee. *Fundamentals of Boundary-Layer Meteorology*. Springer, Cham, Switzerland, 2018. 15, 73

[87] Y. Lee. l_p regularization for ensemble Kalman inversion. *SIAM Journal on Scientific Computing*, 43(5):A3417–A3437, 2021. 28

[88] J. M. Legault. Airborne electromagnetic systems - state of the art and future directions. *CSEG RECORDER*, June:38–47, 2015. 18

[89] H. Levaniemi, D. Beamish, H. Hautaniemi, M. Kurimo, I. Suppala, J. Vironmki, R. Cuss, M. Lahti, and E. Tartaras. The JAC airborne EM system: Aem-05. *Journal of Applied Geophysics*, 67(3):219–233, 2009. doi: 10.1016/j.jappgeo.2007.10.001. 5

[90] L. R. Lines and S. Treitel. Tutorial - A review of least-squares inversion and its application to geophysical problems. *Geophys. Prosp.*, 32(02):159–186, 1984. Discussion in GRP-38-1-101-103 with reply by authors. 15

[91] J. Macnae. Quantitative estimation of intrinsic induced polarization and superparamagnetic parameters from airborne electromagnetic data. *Geophysics*, 81(6):E433–E446, 2016. doi: 10.1190/geo2016-0110.1. 12, 70

[92] J. Macnae. Fitting superparamagnetic and distributed cole-cole parameters to airborne electromagnetic data: A case history from quebec. *Geophysics*, 81(6):B211–B220, 2016. doi: 10.1190/geo2016-0119.1. 12, 70

[93] O. A. Martin, R. Kumar, and J. Lao. *Bayesian Modeling and Computation in Python*. CRC Press, Boca Raton FL, USA, 2021. 5, 29, 77

[94] W. Menke. *Geophysical Data Analysis: Discrete Inverse Theory*. Academic Press, 4th edition, 2018. 25

[95] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21, 1953. doi: 10.1063/1.1699114. 28

[96] P. Miles. pymcmcstat: A python package for bayesian inference using delayed rejection adaptive metropolis. *Journal of Open Source Software*, 4(38):1417, 2019. doi: 10.21105/joss.01417. 5, 77

[97] B. J. Minsley, B. D. Smith, R. Hammack, J. I. Sams, and G. Veloski. Calibration and filtering strategies for frequency domain electromagnetic data. *Journal of Applied Geophysics*, 80:56–66, 2012. 6, 30, 44

[98] K. Mosegaard and M. Sambridge. Monte Carlo analysis of inverse problems. *Inverse Problems*, 18:29–54, 2002. 5, 15

[99] K. Mosegaard and A. Tarantola. Monte-Carlo sampling of solutions to inverse problems. *J. Geophys. Res.*, 100(B7):12431–12447, 1995. 5, 15

[100] K. Mosegaard and A. Tarantola. Probabilistic approach to inverse problems. In *International Handbook of Earthquake and Engineering Seismology, Part A*, pages 237–265. Academic Press, San Diego, 2002. 5

[101] J. B. Muir and V. C. Tsai. Geometric and Level Set Tomography using Ensemble Kalman Inversion. *Geophysical Journal International*, 2019. doi: 10.1093/gji/ggz472. URL `https://doi.org/10.1093/gji/ggz472`. ggz472. 28

[102] M. R. Muller. Variability in the Tellus Waterford test-line FEM data. Technical report, Geological Survey Ireland, 2022. 9, 13, 14, 44, 68, 72, 73

[103] G. Nolet, R. Montelli, and J. Virieux. Explicit, approximate expressions for the resolution and a posteriori covariance of massive tomographic systems. *Geophysical Journal International*, 138(1):36–44, 1999. doi: 10.1046/j.1365-246x.1999.00858.x. 27

[104] W. Nowak. Best unbiased ensemble linearization and the quasi-linear kalman ensemble generator. *Water Resources Research*, 45(4), 2009. doi: 10.1029/2008WR007328. 5

[105] D. W. Oldenburg and Y. Li. Estimating depth of investigation in dc resistivity and ip surveys. *Geophysics*, 64(2):403–614, 1999. 23, 56

[106] D. S. Oliver, A. C. Reynolds, and N. Liu. *Inverse theory for petroleum reservoir characterization and history matching*. Cambrigde University Press, Cambridge UK, 2008. 5, 26

[107] A. V. Oppenheim and R. W. Schafer. *Discrete-time Signal Processing*. Pearson, Englewood Cliffs, NJ, 2014. 43

[108] J. Ory and R. G. Pratt. Are our parameter estimators biased? The significance of finite-difference regularization operators. *Inverse Problems*, 11:397–424, 1995. 25

[109] R. L. Parker. Understanding inverse theory. *Annual Review of Earth and Planetary Sciences*, 2:35–64, 1977. 15

[110] R. L. Parker. *Geophysical Inverse Theory*. Princeton University Press, Princeton, New Jersey, 1994. 15

[111] W. H. Pelton, S. H. Ward, P. G. Hallof, W. R. Sill, and P. H. Nelson. Mineral discrimination and removal of inductive coupling with multifrequency induced-polarization. *Geophysics*, 43(03):588–609, 1978. Discussion in GEO-49-09-1556-1557; Reply in GEO-49-09-1557-1558. 16

[112] O. Portniaguine and M. S. Zhdanov. Focusing geophysical inversion images. *Geophysics*, 64, 1999. doi: 10.1190/1.1444596. 5, 19

[113] M. H. Quenouille. Notes on bias in estimation. *Biometrika*, 43:353–360, 1956. 27

[114] A. Raiche. Modelling the time-domain response of AeM systems. *Expl. Geophys.*, 29(1/2): 103–106, 1998. 18

[115] A. P. Raiche. A flow through Hankel transform technique for rapid accurate Green's function computation. *Radio Science*, 34:549, 1999. 18

[116] V. Rath, J. Sundberg, J.-O. Näslund, and L. C. Liljedahl. Paleoclimatic inversion of temperature profiles from deep boreholes at forsmark and laxemar. Technical Report TR-18-06, Svensk Kärnbränslehantering AB, Stockholm (SKB), 2019. URL `https://www.skb.com/publication/2493035/`. 77

[117] N. Rawlinson, A. Fichtner, M. Sambridge, and M. K. Young. Seismic tomography and the assessment of uncertainty. *Advances in Geophysics*, 2014. 5, 27

[118] Z. Ren and T. Kalscheuer. Uncertainty and resolution analysis of 2d and 3d inversion models computed from geophysical electromagnetic data. *Surveys in Geophysics*, pages 1–66, 2019. doi: 10.1007/s10712-019-09567-3. 23

[119] P.-A. Reninger, G. Martelet, J. Deparis, J. Perrin, and Y. Chen. Singular value decomposition as a denoising tool for airborne time domain electromagnetic data. *Journal of Applied Geophysics*, 75(2):264 – 276, 2011. doi: 10.1016/j.jappgeo.2011.06.034. 44

[120] C. D. Rodgers. *Inverse Methods for atmospheric sounding*. World Scientific, Singapore, 2000. 15, 20, 22

[121] L. Römhild, M. Sonntag, D. Kiyan, R. Rogers, V. Rath, and J. H. Börner. Anisotropic complex electrical conductivity of black shale and mudstone from the Moffat Shale Group (Ireland). *Near Surface Geophysics*, 17:675–690, 2019. doi: 10.1002/nsg.12073. 12, 70

[122] G. K. Rosenkjaer, E. Gasperikova, G. A. Newman, K. Arnason, and N. J. Lindsey. Comparison of 3D MT inversions for geothermal exploration: Case studies for Krafla and Hengill geothermal systems in Iceland. *Geothermics*, 57:258–274, 2015. 27

[123] M. Sambridge and K. Mosegaard. Monte Carlo methods in geophysical inverse problems. *Reviews of Geophysics*, 40(3), 2002. doi: 10.1029/2000RG000089. 5

[124] D. Sattel and P. Mutton. Modelling the superparamagnetic response of AEM data. *Exploration Geophysics*, 46(1):118–129, 2015. doi: 10.1071/eg14005. 12, 70

[125] C. Scheidt, L. Li, and J. Caers. *Quantifying Uncertainty in Subsurface Systems*. AGU-Wiley, Hoboken NJ, USA, 2018. 23

[126] S. Schnaidt. *Improving Uncertainty Estimation in Geophysical Inversion Modelling*. PhD thesis, University of Adelaide, 2015. 27

[127] S. Schnaidt and G. Heinson. Bootstrap resampling as a tool for uncertainty analysis in 2-d magnetotelluric inversion modelling. *Geophysical Journal International*, 203(1):92–106, 2015. doi: 10.1093/gji/ggv264. URL `http://dx.doi.org/10.1093/gji/ggv264`. 27

[128] C. Scholl. Die Periodizität von Sendesignalen bei Long-Offset Transient Electromagnetics. Master's thesis, Institut für Geophysik und Meteorologie der Universität zu Köln, 2001. 6, 50

[129] C. Scholl and R. N. Edwards. Marine downhole to seafloor dipole-dipole electromagnetic methods and the resolution of resistive targets. *Geophysics*, 72(2):WA39–WA49, 2007. 6, 13, 50, 71

[130] K. Schwalenberg and V. Rath. Magnetotellurische Sensitivitäten im 2D-Fall. In K. Bahr and A. Junge, editors, *16. Kolloquium Elektromagnetische Tiefenforschung: Neustadt/Weinstr.*, page n/a, 1998. 23, 24

[131] K. Schwalenberg, V. Rath, and V. Haak. Sensitivity studies applied to a two-dimensional resistivity model from the Central Andes. *Geophysical Journal International*, 150, 2002. doi: 10.1046/j.1365-246X.2002.01734.x. 23, 24

[132] B. A. Shenoi. *Introduction to Digital Signal Processing and Filter Design*. Wiley, Englewood Cliffs, NJ, 2006. 43

[133] A. Smiarowski and D. Sattel. Modelling using receiver waveform and the importance of system geometry. In *ASEG Extended Abstracts, 2015:1*, 2015. doi: 10.1071/ASEG2015ab310. 9, 10, 69

[134] R. Smith. On removing the primary field from fixed-wing time-domain airborne electromagnetic data: some consequences for quantitative modelling, estimating bird position and detecting perfect conductors. *Geophys. Prosp.*, 49(04):405–416, 2001. 9, 69

[135] R. R. Strauss, S. Bishnu, and M. R. Petersen. Comparing the performance of julia on cpus versus gpus and julia-mpi versus fortran-mpi: a case study with mpas-ocean (version 7.1). *EGUsphere*, 2023:1–22, 2023. doi: 10.5194/egusphere-2023-57. URL `https://egusphere.copernicus.org/preprints/egusphere-2023-57/`. 5

[136] R. Stull. *Practical Meteorology. An Algebra-based Survey of Atmospheric Science*. University of British Columbia, 2017. 15, 73

[137] R. B. Stull. *An introduction to boundary layer meteorology*. Kluwier academic publishers, 1988. 15, 73

[138] J. S. Sumner. *Principles of Induced Polarization for Geophysical Exploration*. Elsevier, Ansterdam, NL, 1976. 16

[139] F. Tampieri. *Turbulence and Dispersion in the Planetary Boundary Layer*. Springer, Cham, CH, 2017. 15, 73

[140] A. Tarantola. *Inverse Problem Theory*. Elsevier, Amsterdam, The Netherlands, 1987. ISBN 0-444-42765-1. 22

[141] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. SIAM, Philadelphia PA, USA, 2005. 5, 6, 15, 22

[142] A. Tarantola and B. Valette. Inverse problem = quest for information. *J. Geophysics*, 50: 159–170, 1982. 5, 15, 20, 21

[143] A. Tarantola and B. Valette. Generalized nonlinear inverse problems solved using the least squares criterion. *Rev. Geophys. Space Phys.*, 20:219–232, 1982. 5, 15, 20, 21

[144] V. S. Thorkildsen and L.-J. Gelius. Electromagnetic resolution - A CSEM study based on the Wisting oil field. *Geophysical Journal International*, 2023. doi: 10.1093/gji/ggad046. URL `https://doi.org/10.1093/gji/ggad046`. ggad046. 25, 26

[145] B. W. Tichelaar and L. J. Ruff. How good are our best models? jackknifing, bootstrapping, and earthquake depth. *EOS*, 70(20):593, 605–606, 1989. 5, 23, 27

[146] C. M. Tso, M. Iglesias, P. Wilkinson, O. Kuras, J. Chambers, and A. Binley. Efficient multiscale imaging of subsurface resistivity with uncertainty quantification using ensemble Kalman inversion. *Geophysical Journal International*, 225(2):887–905, 2021. doi: 10.1093/gji/ggab013. 28

[147] W. Tukey. Bias and confidence in not quite large samples. *Annals of Mathematical Statistics*, 29:614, 1958. 27

[148] A. Vest Christiansen and E. Auken. A global measure for depth of investigation. *Geophysics*, 77(4):WB171–WB177, 2012. doi: 10.1190/geo2011-0393.1. 23, 24

[149] J. Vilà-Guerau de Arellano, C. C. van Heerwaardenand Bart J. H. van Stratum, and K. van den Dries. *Atmospheric Boundary Layer. Integrating Air Chemistry and Land Interactions*. Cambridge University Press, Cambridge, UK, 2015. 15, 73

[150] C. Vogel. *Computational methods for inverse problem*. SIAM, Philadelphia PA, USA, 2002. 49

[151] G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1990. 49

[152] J. R. Wait. *Electromagnetic Waves in Stratified Media*. Pergamon, Boulder CO, USA, 1962. 16, 18

[153] J. R. Wait. *Geo-electromagnetism*. Academic Press, New York, 1982. 16

[154] S. H. Ward and D. W. Hohmann. Electromagnetic theory for geophysical applications. In M. N. Nabighian, editor, *Electromagnetic Methods in Applied Geophysics. Volume 1, Theory*, pages 131–311. SEG, Tulsa OK, USA, 1987. 16

[155] H. Wendland. *Scattered data approximation*. Cambridge UNiversity Press, Campridge MA, USA, 2005. 39, 41

[156] Q. Xu. Representations of inverse covariances by differential operators. *Advances in Atmospheric Sciences*, 22(2):181–198, 2004. doi: 10.1007/BF02918508. URL `https://doi.org/10.1007/BF02918508`. 22

[157] X. Zhang, Michelén-Ströfer, and H. Xiao. Regularized ensemble Kalman methods for inverse problems. arXiv, 2020. 5, 19

[158] M. S. Zhdanov. *Inverse theory and applications in geophysics*. Elsevier, Amsterdam, Netherlands, 2015. 5, 19