ResIPy, an intuitive open source software for complex geoelectrical inversion/modeling

Authors:
Guillaume Blanchy[1, *]
Sina Saneiyan[2]
James Boyd[1, 3]
Paul McLachlan[1]
Andrew Binley[1]

Affiliations:
[1]Lancaster Environment Centre, Lancaster University, Lancaster, UK
[2]Department of Earth and Environmental Sciences, Rutgers, The State University of New Jersey, Newark, NJ, United States
[3]British Geological Survey, Keyworth, Nottingham, UK

Corresponding author:
Guillaume Blanchy (g.blanchy@lancaster.ac.uk) Lancaster Environment Centre, Lancaster University, Lancaster, LA1 4YQ, UK

Authorship statement:
GB and SS contributed to the GUI
GB, SS, JB and PM contributed to the API
GB specifically contributed to the R2 and Survey classes
JB specifically contributed to the mesh generation and handling
SS specifically contributed to the IP part of the API and GUI
PM specifically contributed to the sequence generation of the API and the testing of the GUI
GB, SS, JB and PM wrote the paper
AB wrote all the Fortran executables and provided feedback on the manuscript

Code availability
The open-source code (GPL license) is available on GitLab: https://gitlab.com/hkex/pyr2.

Highlights

- Geophysics is more frequently used in interdisciplinary projects by non-specialists.
- ResIPy is a simple to use, intuitive, open source graphical user interface and API.
- ResIPy is a good teaching tool to learn how to invert and model geoelectrical data.
- Data filtering and error modeling of resistivity and IP data improve inversion.
- Field applications and survey design with ResIPy is demonstrated.

Declaration of interest
None

Abstract

Electrical resistivity tomography (ERT) and induced polarization (IP) methods are now widely used in many interdisciplinary projects. Although field surveys using these methods are

relatively straightforward, ERT and IP data require the application of inverse methods prior to any interpretation. Several established non-commercial inversion codes exist, but they typically require advanced knowledge to use effectively. ResIPy was developed to provide a more intuitive, user-friendly, approach to inversion of geoelectrical data, using an open source graphical user interface (GUI) and a Python application programming interface (API). ResIPy utilizes the mature R2/cR2 inversion codes for ERT and IP, respectively. The ResIPy GUI facilitates data importing, data filtering, error modeling, mesh generation, data inversion and plotting of inverse models. Furthermore, the easy to use design of ResIPy and the help provided inside makes it an effective educational tool. This paper highlights the rationale and structure behind the interface, before demonstrating its capabilities in a range of environmental problems. Specifically, we demonstrate the ease at which ResIPy deals with topography, advanced data processing, the ability to fix and constrain regions of known geoelectrical properties, time-lapse analysis and the capability for forward modeling and survey design.

Keywords

Geophysics, inversion, data filtering, electrical resistivity tomography, induced polarization, R2/cR2

1 Introduction

Geoelectrical methods are powerful and well-established tools for non-intrusive characterization of subsurface geoelectrical properties. These methods were developed in the early 1900s for mineral resource exploration (e.g., Schlumberger, 1920). However, electrical resistivity tomography (ERT) and induced polarization (IP) are now extensively used in a wide range of environmental studies. Applications include monitoring landslides (Uhlemann et al., 2018), precision agriculture (Vanella et al., 2018), assessing permafrost degradation (Mewes et al., 2017), determining hydraulic properties (Benoit et al., 2018), imaging of landfill sites (Ntarlagiannis et al., 2016), monitoring groundwater-surface water interactions (McLachlan et al., 2017) and monitoring of bio-mediated soil stabilization (Saneiyan et al., 2019). As geoelectrical methods become embedded in cross-disciplinary studies there is a need for relatively easy to use data inversion tools, which retain levels of complexity required for modeling of more sophisticated applications.

The translation of geoelectrical measurements to geoelectrical properties requires the use of inverse methods. These methods aim to find the best distribution of geoelectrical parameters that is consistent with observed measurements. This involves minimizing the misfit between the set of four electrode measurements and the predicted response from a geoelectrical model. Because of the non-linear nature of the problem, the inversion proceeds in an iterative manner until the misfit between the predicted response and the measurements are within a given tolerance. Forward modeling can also be used to generate synthetic data given a synthetic geoelectrical model (workflow shown with red arrows in Figure 1). Typically, the measurements are composed of a set of transfer resistances (or apparent resistivities) from different four electrode configurations (quadrupoles). If the induced polarization (IP) method is used, the chargeability (in time-domain IP surveys) or phase angle (in frequency domain IP surveys) is

85 also recorded in addition to the transfer resistance. At low frequencies (below 10Hz, i.e. the
86 usual operation frequencies of resistivity/IP instruments) chargeability and phase angle have a
87 linear relationship and the complex transfer impedance can be derived from time domain IP
88 measurements. Therefore, the inversion seeks to find the resistivity (or complex resistivity – in
89 the case of an IP survey) distribution that can explain the measurements. For more details
90 about the inverse methods used here, see Binley (2015) and Binley and Kemna (2005).

91 Several established tools exist for inverting geoelectrical data (e.g. Pidlisecky and Knight, 2008).
92 Some codes are specialized for inverting monitoring (time-lapse) measurements (e.g. Karaoulis
93 et al., 2013) or for including hydrological or other geophysical information in the inversion (e.g.
94 Johnson et al., 2017; Nath et al., 2000). Most non-commercial tools are built around command-
95 line software implementations that require significant experience to operate effectively, which
96 can be challenging for new users, and limits use in an educational environment. There is a
97 growing interest in open source codes within the scientific community, as they provide both
98 users and developers access to comment and advance codes, allowing contributions from
99 multiple developers. More significantly, perhaps, is the increasing demand for the sharing of
100 tools for reproducible science. An open source approach allows users to tailor a given code to
101 suit their needs. Successful examples of open source codes in geophysics include pyGIMLI
102 (Rücker et al., 2017) and SIMPEG (Cockett et al., 2015) both providing a Python application
103 programming interface (API).

104 In the spirit of open source provision, we developed ResIPy (formerly named pyR2) to facilitate
105 processing, modeling and inversion of geoelectrical data. ResIPy is written in Python and is
106 open source (source code is available on a GitLab repository: https://gitlab.com/hkex/pyr2). The
107 software handles importing, filtering, error modeling of geoelectrical data and makes use of the
108 freely available R2, cR2 and R3t codes
109 (http://www.es.lancs.ac.uk/people/amb/Freeware/Freeware.htm) for modeling/inversion of data.

R2, cR2 and R3t are mature codes for resistivity and IP problems but lack any graphical user interface. Befus (2018) recently documented a Python wrapper for R2. In contrast, ResIPy offers full IP capability and data quality control features, and has been developed to suit educational/training needs. ResIPy also has 3D capabilities (Boyd et al. 2019) but these will not be detailed in this 2D-focused manuscript. R2 and cR2 are finite element based, allowing the incorporation of complex topography and modeling of bounded regions. They allow full flexibility of electrode assignment; accommodating, for example, surface electrode and borehole electrode based surveys. Inverse modeling in the codes is conducted using a weighted least squares objective function coupled with a range of regularization options, including time-lapse data analysis (e.g. Binley, 2015).

R2 was developed for solving DC resistivity problems. cR2, in contrast, is tailored for IP problems by formulating the problem in terms of complex resistivity (e.g. Binley and Kemna, 2005). Both codes require specifically formatted text files for data input, specification of forward or inverse model settings, and mesh construction. ResIPy removes the need for such text input in a graphic user interface (GUI), whilst assisting the user in pre- and post-processing stages. Use is made of the freely available meshing code Gmsh (Geuzaine and Remacle, 2009) for complex mesh construction. The underlying philosophy of ResIPy is to retain the necessary sophistication of geoelectrical inversion whilst enhancing the accessibility to a wider range of users. Moreover, ResIPy provides an environment for training that may be refined and customized to meet user needs. Hence, ResIPy is particularly well suited for educational purposes. Its intuitive interface, open source nature and wide capabilities allow new users to explore, at their pace, geoelectrical data analysis. Figure 1 shows the main capabilities of ResIPy.

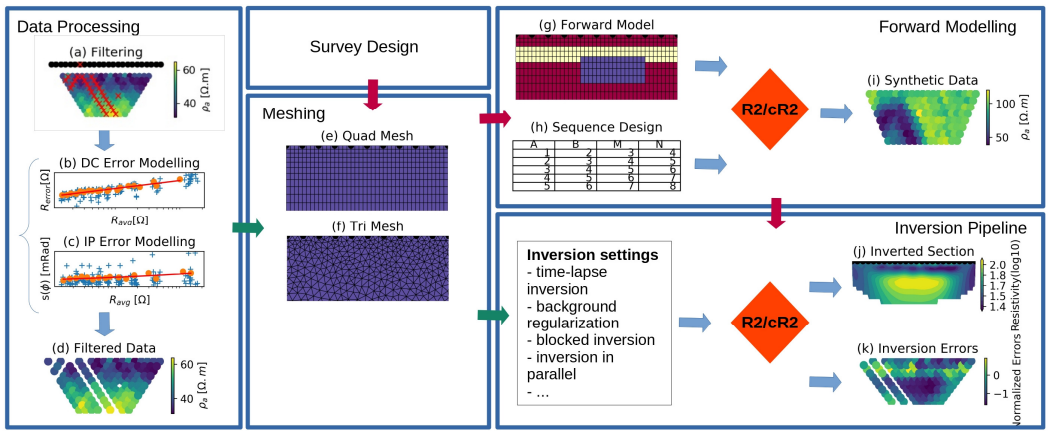

Figure 1: Diagram of the capabilities of ResIPy. Inversion workflow (green arrows): data can be imported and bad measurements or electrodes can be filtered out (a). If reciprocal measurements are present an error model can be fitted for DC resistivity (b) and for IP (c). A quadrilateral (e) or triangular (f) mesh is then generated. The mesh and the filtered data (d) are sent to the inversion pipeline. Different inversion settings can be defined such as blocking regions of the mesh or time-lapse settings. The resulting inverted section is then produced with R2/cR2 (j) along with diagnostic pseudo section of the normalized error of the inversion (k). Modeling workflow (red arrows): based on a hypothesis, a mesh is created and a synthetic model designed (d). After creating a sequence (e) the forward response can be computed (f) using R2/cR2. Those synthetic data can then be sent to the inversion pipeline to be inverted.

We first describe the general design of the code with the API and GUI. Then, data processing and mesh generation options are explained. Finally, different aspects of ResIPy are illustrated through different environmental field and synthetic cases.

2 Structure of the code

2.1 Software design

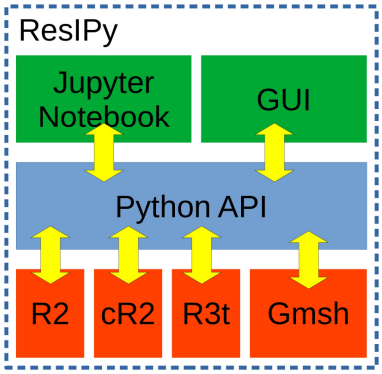

Figure 2: ResIPy internal working with three main layers. On top, the visualization layer. In the middle the Python API that is in charge of all calling the executable. At the base are the compiled executable R2, cR2 and Gmsh.

ResIPy is made of three layers (Figure 2). The bottom layer is composed of the compiled inversion codes R2 (and R3t) and cR2 that are called during inversion or forward modeling for DC resistivity and complex resistivity, respectively. This layer also contains the software Gmsh (http://gmsh.info) that is used to generate triangular meshes. The middle layer is composed of the Python API. This interface contains a set of functions that acts as a wrapper around the executables, facilitating the writing of their input files (R2.in, cR2.in, mesh.geo) and the reading of their outputs. The Python API also contains specific processing routines such as for filtering the data or performing advanced error modeling of DC and IP data. A detailed list of the API functions can be found in Appendix 1. The top layer is composed of visualization tools that provide a graphical environment to the user.

The Python API is object-oriented and has several classes. The main class is called R2 *(R2.py)* which manages the data processing and inversion. The GUI initiates an R2 object each time a new inversion/modeling problem is started. Next is the `Survey` class (*Survey.py*) that handles one dataset for one survey. Multiple surveys (e.g. from a time-lapse experiment), can be handled inside the same R2 object using the `R2.surveys` attribute. Finally, the `Mesh` class (*meshTools.py*) handles the tasks associated with the construction of the finite element mesh (e.g. mesh generation, mesh refinement, electrode positioning, etc.). Each `R2` object contains an instance of the `Mesh` class in `R2.mesh`. More details about the mesh as well as a full overview of the classes and their respective methods are provided in Appendix 1.

The Python API is documented within the code according to scipy/numpy docstring guidelines (https://docs.scipy.org/doc/numpy-1.15.0/docs/howto_document.html). The advantage of this approach is that html documentation can be easily compiled and updated using the Python documentation generator Sphinx (https://hkex.gitlab.io/pyr2). The GUI also provides help through the interface (tool tips), which allows the user to learn more about different aspect of the inversion and error modeling.

2.2 Standalone graphical user interface

The standalone GUI is written in PyQt5, making it easy to modify and therefore allows for future development. Moreover, graphs are plotted using matplotlib (Hunter, 2007) and can be exported at every step. The GUI uses a series of tabs (Figure 3) that allows a non-linear workflow and takes the user through the necessary stages of importing and filtering data (or creating synthetic data for forward modeling), generating a mesh and inverting data. The import tab is used to load geoelectrical and topographical data. Geoelectrical data can be imported directly using a number of standard formats (e.g. IRIS Instruments Syscal files, Res2DInv files, and the

standard R2 and cR2 input files) or manually imported using the “Custom Parser” tab. Additionally, topographical data can be entered manually or loaded from a comma separated value (csv) file at “Electrode (XYZ/Topo)” tab. After importing data, the user can continue through the workflow, as outlined in the following sections, or move directly to inversion using default settings with the “Invert” button in the “Importing” tab. Using default settings allows the user to generate reliable images in most cases, which may be a useful for novice users or for fast assessment of data (e.g. in the field). It is important to note that all inversion parameters available to R2 and cR2 can be accessed and modified under the “Inversion settings” tab. For instance, the user can change the regularization type, whether the inversion converts data to logarithmic values, data error estimates, smoothing anisotropy and the maximum number of iterations. Help is provided for each parameter, with further details available in the R2 and cR2 manuals (http://www.es.lancs.ac.uk/people/amb/Freeware/Freeware.htm). Furthermore, under advanced settings the user has the option to do batch inversions in parallel on multicore machines.

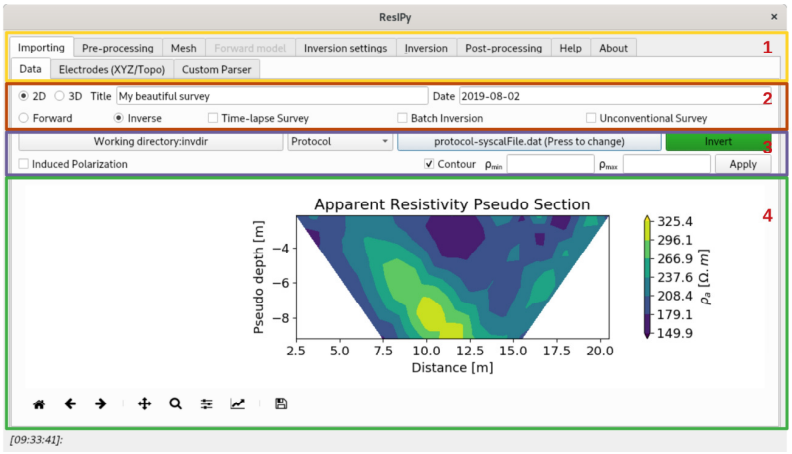

Figure 3: General layout of the standalone graphical user interface with (1) different tabs for each processing step, (2) Options for type of survey and inverse/forward modeling , (3) Data import and IP check, (4) pseudo-section plot of the imported data.

2.3 Data quality control

ResIPy is capable of rigorous data cleaning and quality control, this can either be done automatically or with user control. Both approaches take into account whether reciprocal measurements are present in the dataset or not. In the GUI, data quality control options are available under the “Pre-processing” tab.

2.3.1 Automatic data cleaning/filtering

The first step of data cleaning in ResIPy is the `basicFilter()` method, which removes the following measurements: (1) infinity or NaN values, (2) duplicates, (3) invalid measurements (e.g. quadrupoles were current electrodes are also potential electrodes – A or B at same position as M or N). If there are reciprocal measurements in the input file, ResIPy automatically calls `reciprocal()` and calculates reciprocal errors. The number of measurements with a relative reciprocal error above 20% are also notified to the user (using the API), but are not discarded by default. The above mentioned methods are also called when a dataset is manually added using `addData()` (e.g. when a reciprocal dataset is added separately).

2.3.2 User-controlled quality control methods

In addition to automatic data cleaning step, ResIPy has several user-controlled quality control methods implemented in the code API as well as the GUI. These methods are divided into two categories: (1) data cleaning/filtering and (2) data error analysis.

2.3.2.1 Data cleaning

User-controlled data cleaning/filtering is carried out in multiple separable steps. All the processing is available in the GUI under “Pre-processing” tab. If reciprocal measurements are present, the following methods can be used to clean up dataset: (1) `filterRecip(percent)`, where ‘percent’ is a desired percentage value to remove measurements with high error (2) `removeUnpaired()` to remove quadrupoles that do not have a reciprocal pair. In the GUI, these methods can be found in “Reciprocal Filtering” tab under “Pre-processing” tab. The error probability distribution histogram is also provided to help visualization of dataset quality (Figure 4c). Additionally, the user can select and remove unwanted measurements (regardless of reciprocity) by using `manualFiltering()` method (also available in the GUI under “Manual/Reciprocal Filtering” tab in “Pre-processing”). This interactive method allows the user to manually pick and remove data points within the GUI. Furthermore, the user can eliminate all measurements carried out by a specific electrode (Figure 4a and b).

Further user-controlled data cleaning/filtering is limited to filtering datasets with chargeability/phase values (“Phase Filtering” tab in “Pre-processing”). Quality control is particularly important for IP applications given the smaller signal to noise ratio, compared to DC resistivity problems (Slater and Lesmes, 2002; Zarif et al., 2017).

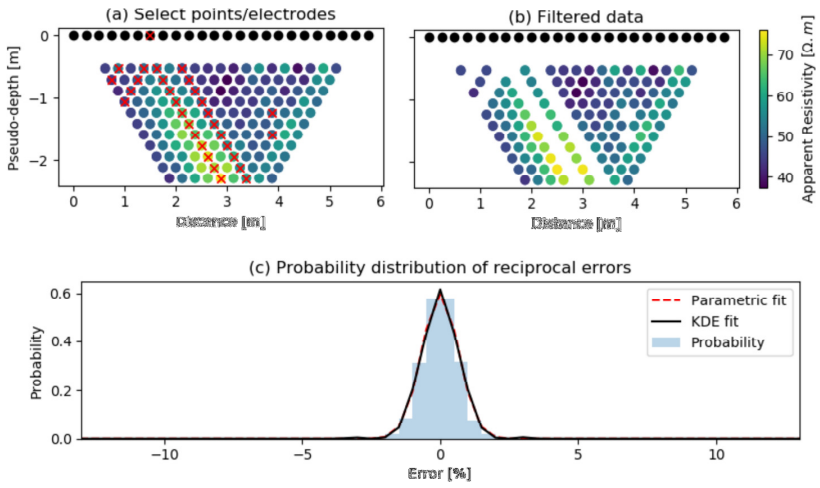

Figure 4: Interactive manual filtering. (a) Pseudo section with selected unwanted data points (crossed out in red), (b) Pseudo section with removed data points (user must hit "Apply" button to remove the crossed out data points). And (c) probability distribution of the reciprocal error with parametric and non parametric fit (Kernel Density Estimate = KDE).

To give the user full control of the IP data cleaning/filtering, different methods are implemented in the code. In the GUI, the user can apply the available filtering methods and see the results in an interactive Raw versus Filtered graph (Figure 5). All the phase angle filters can be used separately and are reversible at this stage. In the GUI, the user can select the "Reset all phase filters" button to reset back to the state after manual/reciprocal filtering.

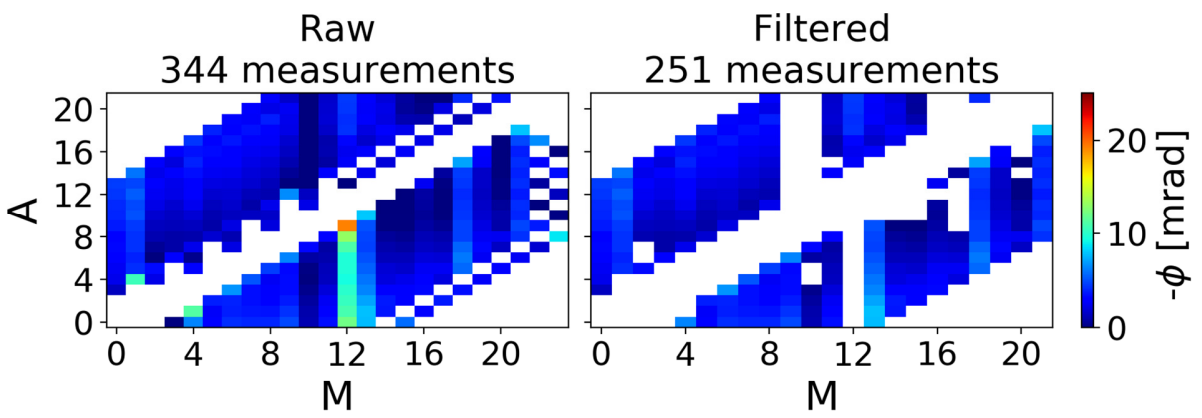

Figure 5: Interactive phase angle (ϕ) filtering diagrams. (left) Raw measurements (no filters). (right) Filtered dataset (including both automatic and user-controlled filtering). Each measurement is represented by a colored pixel where the y coordinate is position number of the first current electrode (A) and x coordinate is position number of first potential electrode (M) for a 4 electrode (A-B/current pair, M-N/potential pair) quadrupole (Flores Orozco et al., 2013). White pixels represent no measurement at that location.

2.3.2.2 Data error analysis

In addition to the data cleaning, ResIPy is capable of data error modeling for DC resistivity and/or IP data. Data error analysis tabs in the GUI ("Resistance Error Model" and "Phase Error Model") are only available when there are reciprocal measurements within the input dataset(s).

Resistance error model:

Observed errors are based on individual measurement reciprocal errors according to:

$$R_{error} = \left|R_{normal} - R_{reciprocal}\right|. \quad (1)$$

To calculate an error model (linear or power-law), ResIPy uses multi-bin analysis (for more details of the method, see Koestel et al. (2008) and Mwakanyamale et al. (2012)) where errors (equation 1) are binned into 20 bins of equal count and sorted based on average resistance error R_{avg} [Ω], given by

$$R_{avg} = \frac{|R_{normal} + R_{reciprocal}|}{2} . \quad (2)$$

239

240 *Phase error model*:

241 Observed errors are based on phase angle discrepancies between normal and reciprocal
242 measurements (*s(ϕ)* [mrad])

$$s(\phi) = |\phi_{normal} - \phi_{reciprocal}| \quad (3)$$

243 and are plotted versus individual normal measurement resistances (R_{normal} [Ω]). Phase error
244 models (power-law and parabolic) are calculated using multi-bin analysis (Mwakanyamale et al.,
245 2012; Flores Orozco et al., 2012); where phase angle discrepancies have been binned into 20
246 equal count bins and sorted based on R_{normal} [Ω]. The final error model fit formula is written on
247 top of the graph with the coefficient of determination (R^2) (Figure 6). For more details about all
248 the methods used in this section, see Table 1.

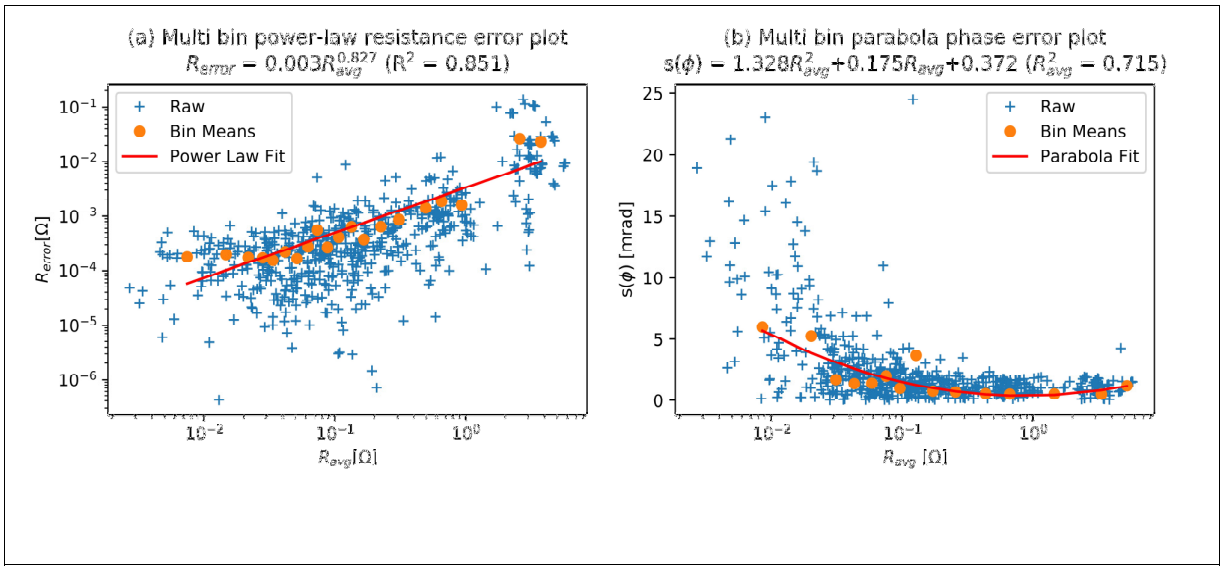

Figure 6: Multi-bin error models. (a) Resistance error model (linear), (b) Phase angle error model

(parabola). Other options are also available to choose within the GUI.

2.4 Meshing

In ResIPy, two types of 2D finite element meshes can be used: structured quadrilateral (see section 2.4.1) or unstructured triangular (see section 2.4.2). Regardless of elemental shape, the mesh elements tend to be finer near the electrodes and get coarser at greater distances from the electrodes. This is to address the need for greater discretization in areas of high potential gradient. The mesh is composed of a finer mesh defined by the electrode locations which is encompassed in a coarser mesh with a larger lateral and depth extent (for semi-infinite boundary problems). This is because the mesh boundaries are non-flux (Neumann). In a normal field setting, current from the electrodes will propagate beyond the survey bounds; R2 and cR2 model electrical current flow for the entire mesh assigned to the problem. Hence, in order to reliably model current flow, the mesh boundaries need to be sufficiently far away from the electrode positions. Note there are exceptions where such infinite boundaries are not appropriate (e.g. a non-infinite boundary would exist if conducting electrical surveys near cliff faces, or in laboratory tank experiments). For those specific cases a customized mesh can be imported in to the ResIPy workflow.

The lateral extent of the fine mesh region is dependent on the X (horizontal) coordinates of electrodes (which are represented as nodes in the mesh). The fine mesh region extends to the following depth estimated using

$$Z_{min} - \frac{2X_{max}}{3}. \quad (4)$$

Where Z_{min} is the lowest elevation of electrodes in the surface or borehole array, and X_{max} is the distance between the longest quadrupole in the survey. Note that this is not a depth of investigation, for example as computed by the method of Oldenburg and Li (1999), but rather a conservative estimate of it to facilitate meshing.

2.4.1 Quadrilateral mesh

ResIPy defines a quadrilateral mesh as an array of X and Z coordinates (i.e. a structured grid), and an array of elevation values with the same length as the X array. The mesh is composed of a fine region defined by the survey geometry with a coarser surrounding region (because of the infinite boundaries). Only the finer mesh region is displayed in the GUI. The number of nodes between the electrodes can be adjusted in the GUI (Figure 8). In the API, the mesh growth factors in the Z direction can be adjusted with `zf` and `zgf` attributes for the fine and the coarse region respectively. In the X direction, a growth factor for the coarse region can also be set in the API (`xgf`). In the case of buried electrodes (e.g. cross-borehole surveys), the X and Z coordinates of the electrodes are inserted into the quadrilateral mesh after the main mesh generation scheme.

2.4.2 Triangular mesh

Triangular meshes allow application to more complicated geometry (e.g. topography and geometrical features within the region of study). In ResIPy, the `trian_mesh()` function generates the mesh by calling Gmsh.exe to perform the meshing process. The `trian_mesh()` function provides an input file for gmsh (.geo) and parses the output (.msh).

Similar to the quadrilateral mesh, it is possible to control the mesh refinement by specifying a characteristic length associated with each electrode node. Smaller characteristic lengths will

result in a finer mesh. Similar to the quadrilateral mesh, the user can specify a growth factor that controls the increase in element size with depth. With both quadrilateral and triangular meshes it is advisable to avoid fine elements in areas with low sensitivity, as they will not add anything to the interpretation of the inverted model but will increase computation time. These two parameters can be set in the GUI using slider or in the API using the and `cl_factor` attributes of the `R2.createMesh()` method.

Both quadrilateral and triangular mesh options are available in ResIPy to encompass the capabilities of the R2/cR2 codes. A quadrilateral mesh is generated faster than a triangular mesh in ResIPy and output from a structured mesh (e.g. the array of resistivities following inversion) can be easier to work with (e.g. to extract vertical or horizontal resistivity profiles). However, triangular meshes are more versatile, can account for complex topography and are computationally more efficient. Consequently, triangular meshing is recommended in ResIPy.

2.4.2.1 Whole space problems

In some cases, it might be appropriate to assume the electrodes are buried at such an extensive depth that current flow does not interact with the surface or any other boundaries. In such cases, ResIPy offers a scheme whereby electrode coordinates are inserted into a fine triangular mesh region with a larger surrounding region (Figure 7).

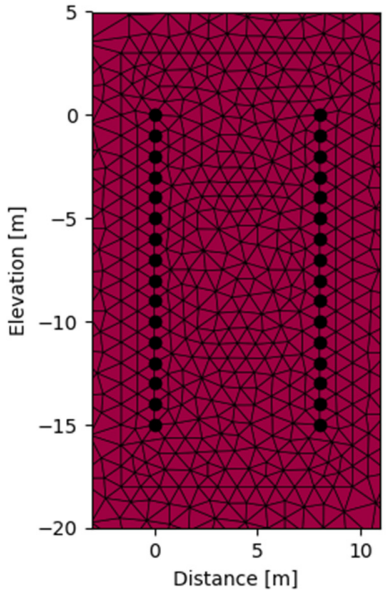

Figure 7: Example of a pair of borehole arrays in a whole space problem. Note that the view is cropped and that the real mesh extends much further away from in all directions. Also note that the mesh shown in coarsely discretized for illustration purposes.

2.4.3 Region definition

For generating a forward model for survey design, or for inverse modeling of a survey with known subsurface boundaries, ResIPy allows the user to define different regions within the mesh. These regions can be assigned a specific resistivity and phase angle values. Regions can be selected in the GUI using an interactive plot picker and table system (Figure 8). In some cases, the user may wish to prevent regularization in the inversion across certain boundaries, for example if there is a known geological boundary. To do this the user can specify that these regions are different zones. In this paper, we make a clear distinction between the term 'region' which is a spatial group of elements, and the term 'zone' which is a special case of a region where the regularization is suppressed along its boundaries. The example in section 3.3

considers a river with a fixed river resistivity, and the example in section 3.5 considers how to generate and invert synthetic data using the forward modeling capabilities.

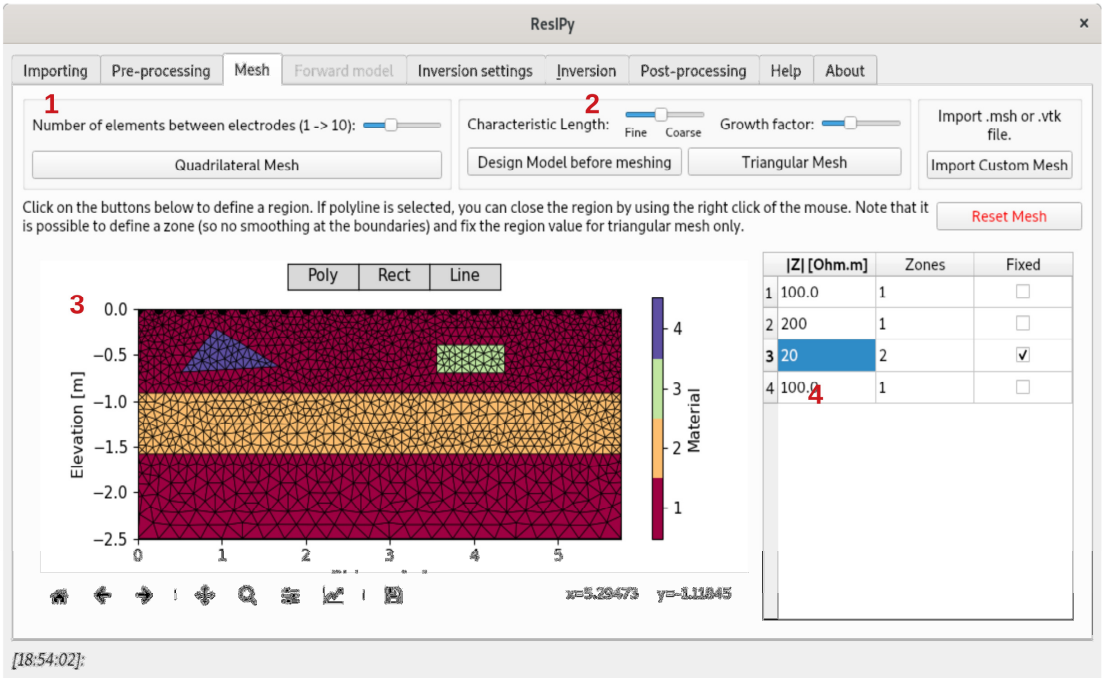

Figure 8: The interface allows for both quadrilateral (1) and triangular (2) mesh generation. The interactive mesh display allows to draw regions of different shapes (3) and specify their properties using the panel on the right panel(4).

3 Applications

The following examples demonstrate the capabilities of ResIPy. Each of the examples aims to expose particular aspects of ResIPy relevant for the case study. For each example the steps to reproduce the results in the GUI along with the lines of code in the API that does the same are provided. This aims to make the link between the GUI and the Python API more obvious.

Further examples are available in the GitLab repository (https://gitlab.com/hkex/pyr2/tree/master/examples).

3.1 Survey design

Knowing the measurement response for a given model is a powerful tool to assess method limitations. This is particularly useful when trying to optimize the survey design for an intended target, or for determining if detecting a parameter of interest is realistic or not. Forward modeling can be done in the ResIPy API using the `R2.forward()` method, or in the GUI by selecting "Forward" check box in the main importing tab. ResIPy offers four types of sequences: dipole-dipole, Wenner, Schlumberger, multiple-gradient. The user has also the possibility to import and generate their own custom sequence. Note that R2/cR2 are capable of modeling any quadrupole sequence or combination of sequences.

The sensitivity of the array to a certain target will depend on quadrupole configuration; hence, for survey design this is an important consideration. For example, Wenner arrays tend to favor sensitivity to horizontal features rather than vertical ones (Binley, 2015). Additionally, the electrode spacing of the survey will dictate the ability of the array to resolve a given target, as the array spacing controls spatial resolution and depth of investigation. Arrays with smaller electrode spacing have a shallower depth of investigation than larger arrays but have higher spatial resolution. Therefore, in the case of surveys with a known target but unknown location, arrays with different electrode spacing and quadrupole configurations can be trialed through forward modeling to find a setup that is best suited to the problem.

The following example compares the sensitivity of a Dipole-Dipole and a Wenner sequence to resolve a shallow target. The target, a rectangular feature buried at 1 m depth with dimensions of 3 m by 1 m (Figure 9a), can be defined in the "Mesh tab" using the interactive plot or using

the API method `R2.addRegion()`. The resistivity of the target is set to 10 Ohm.m whilst the background resistivity is set to 100 Ohm.m. The sequence is chosen in the “Forward Model” tab or using the `k.createSequence()` method from the API. Given a starting model (Figure 9Figure 10a) and a sequence, the forward model can be run. The measurements produced are displayed as a pseudo-section (Figure 9Figure 10b and c). In this case 5% noise is added to the measurements to simulate a more realistic scenario. The synthetic data are then inverted to see how much information can be recovered from them (Figure 9Figure 10d and e). Figure 9 shows that a dipole-dipole array is better suited to this kind of problem compared to a Wenner array. In Figure 9d (Wenner array), a low resistivity region can be observed but its location is widespread. Figure 9e (dipole-dipole array) more closely resembles the input resistivity model, and the low resistivity region is better collocated with the placement of the target.

```
k = R2(typ='R2')
k.setElec(np.c_[np.linspace(0, 24, 24), np.zeros((24, 2))])
k.createMesh(typ='quad')
target = np.array([[7,-2.2],[12,-2.2],[12,-5],[7,-5]])
k.addRegion(target, 10, -3) # target definition
k.createSequence(params=[('wenner_alpha',1),
                         ('wenner_alpha',2),
                         ('wenner_alpha',3),
                         ('wenner_alpha',4),
                         ('wenner_alpha',5),
                         ('wenner_alpha',6),
                         ('wenner_alpha',7),
                         ('wenner_alpha',8),
                         ('wenner_alpha',9),
                         ('wenner_alpha',10)])

k.forward(iplot=True, noise=0.05) # add 5 % noise
k.invert(iplot=True)
k.showResults(index=0, attr='Resistivity(Ohm-m)', sens=False)
k.showResults(index=1, attr='Resistivity(Ohm-m)', sens=False)

# now for the dipole dipole
k.createSequence([('dpdp1', 1, 8)])
k.forward(iplot=True, noise=0.05)
k.invert(iplot=True)
```

```
k.showResults(index=1, attr='Resistivity(Ohm-m)', sens=False)
```

362

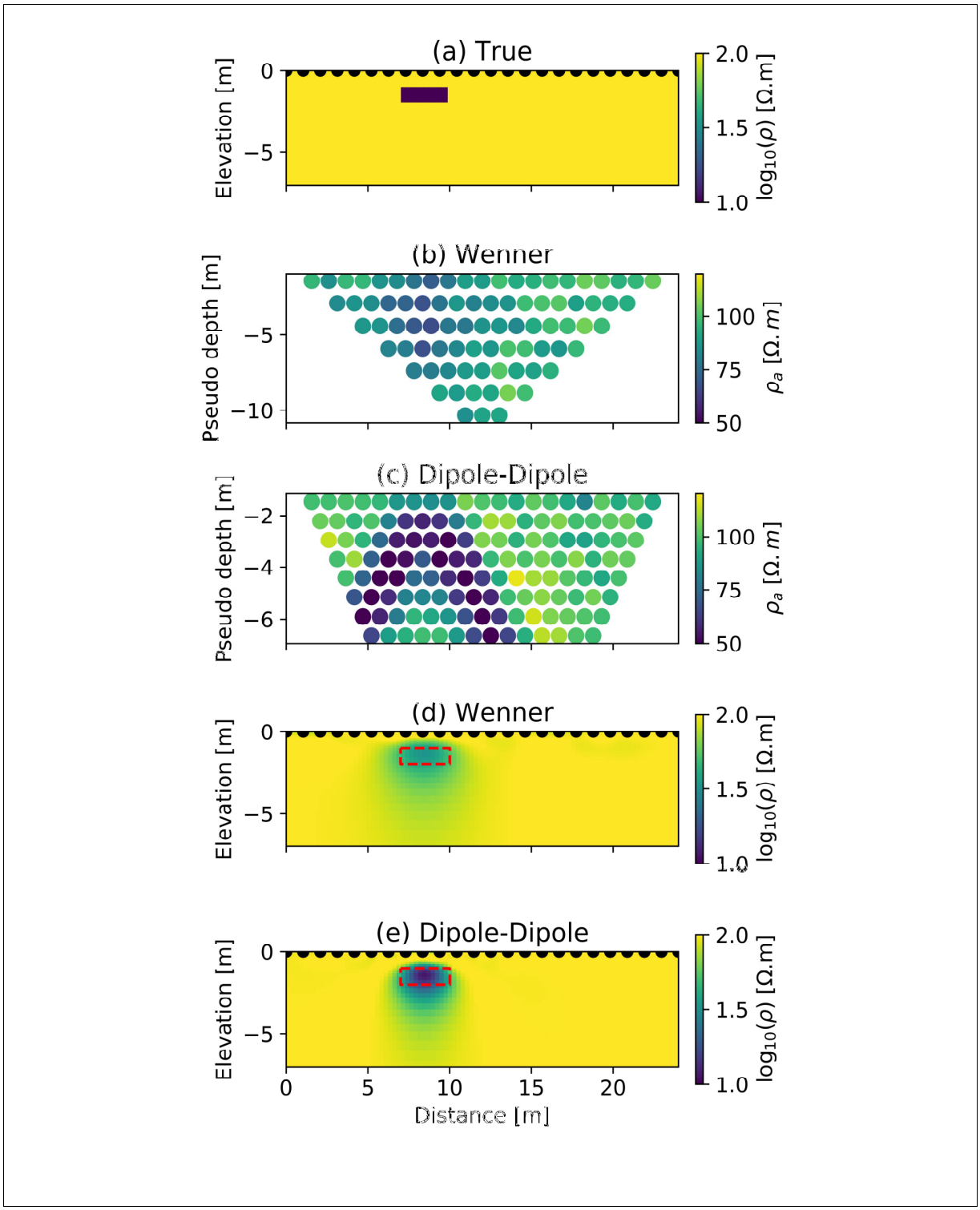

Figure 9: Forward modeling in ResIPy. (a) The original resistivity model for which measurements are computed. (b) and (d) the pseudo and inverted section of apparent resistivities for a Wenner array respectively, (c) and (e) the pseudo and inverted section for a Dipole-Dipole array. The red dashed line in (d) and (e) shows the true position of the target.

363

364 This example uses synthetically generated data to optimize the design of the survey. Once this
365 step is done, the survey is carried out and field/lab measurements are collected. The following
366 examples demonstrate how those measurements are processed with ResIPy.

367 3.2 2D resistivity with topography

368 Castle Hill in Lancaster (UK) is the site of a first century Roman fort (Wood, 2017). At the site
369 there are no remains of the Roman fort walls above the ground but targeted archaeological
370 investigations have found traces of the walls foundations. The aim here is to map the extent of
371 walls around the site using several ERT cross transects. Only one of those transects is used
372 here. In this example, the steep topography of the hill strongly impacts the inversion results, i.e.
373 if topography is not included in the mesh, the inversion outputs unrealistic results containing
374 artifacts. The results are displayed in Figure 10 where the high resistivity anomaly on the top of
375 the slope corresponds to the walls foundations. All transects together help to define the
376 positions of the walls and hence the extent of the Roman fort.

377 GUI:

378 1. Importing data: exact electrode locations can be added in the “Electrodes (XYZ/Topo)”
379 tab
380 2. (optional) choose mesh type: we use triangular mesh
381 3. Inversion

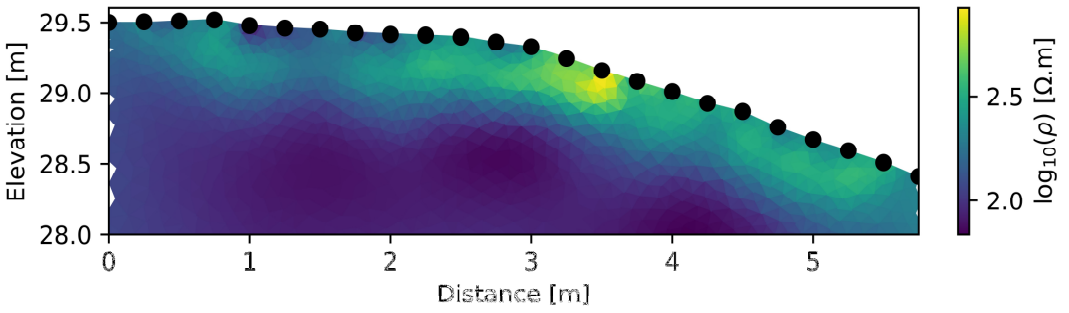

Figure 10: Inverted section of one of the ERT transects crossing over the wall. A zone of higher resistivity approximately 3.5 m along the transect agrees well with other excavations nearby, and probably represents the remains of wall foundations.

The same can be achieved using the API:

```
k = R2() # initiate an R2 instance
k.createSurvey('syscalFileTopo.csv', ftype='Syscal') # import data
k.importElec('elecTopo.csv') # importing the electrodes positions
k.fitErrorPwl() # fit a power law
k.err = 'True' # tells the inversion to use the error model we've fitted
(done automatically in the GUI) this will set a_wgt and b_wgt at 0
k.createMesh(typ='trian') # create quadrilateral mesh
k.invert() # run the inversion
k.showResults() # show the inverted section
```

3.3 2D IP

Recently, it has been shown that IP is a capable tool for monitoring soil strengthening involving calcite precipitation in both lab and field scale (Saneiyan et al., 2019, 2018). Here we use data reported by Saneiyan et al. (2019) to show how the IP filtering options available in ResIPy can enhance inversion quality. To illustrate the processing capabilities of ResIPy we first invert a dataset where the raw IP measurements are used directly without data filtering. Second, we show how data filtering can enhance the final inversion. Note that for IP problems the inverse model can be displayed as an image of resistivity magnitude and phase angle, or as an image

of real conductivity and imaginary conductivity. The resistivity magnitude and phase angles are parameters directly derived from the measured impedances. The real and imaginary conductivity are derived from the magnitude and the phase angle. The advantage of imaginary conductivity over phase angle is that it provides an unbiased estimate of the polarization of the medium.

3.3.1 Inversion without data cleaning

Similar to the previous section, we can approach the problem with either using GUI or straight from API:

GUI:

1. Importing data: code automatically detect “IP” values, if a known file type is chosen (e.g. Syscal)
2. (Optional) choose mesh type: we use triangular mesh
3. Inversion.

API:

```
k = R2(typ='cR2') # initiate an R2 instance (considering there is IP data
in the input data)
k.createSurvey('IP_MICP_ALL.csv', ftype='Syscal') # import data
k.createMesh(typ='trian') # create triangular mesh
k.invert() # run the inversion (and write cR2.in and protocol.dat
automatically)
k.showResults(attr='Phase(mrad)') # show the inverted section
```

For this case, without data cleaning, the inversion of the phase angle did not converge within 10 iterations (observed by consistent unrealistic and very high RMS misfit values per iteration) and

409 the inversion results did not show meaningful subsurface structures. In order to apply data
410 quality control, we then follow the steps reported in Saneiyan et al. (2019).

411 3.3.2 Inversion with data cleaning and error analysis

412 The steps are similar to previous but here we include data quality control routines, filtering and
413 error analysis.

414 GUI:

415 1. Import data: IP_MICP_ALL.csv.
416 2. Reciprocal filtering: removing data points with > 5% reciprocal error
417 3. Phase filtering ("Phase Filtering" tab in "Pre-processing"):
418 a. Removing nested measurements (measurements where M or N are in between A
419 and B)
420 b. Phase range filtering: setting 0 < -ϕ < 20
421 4. Error modeling:
422 a. Resistance error model: power law
423 b. Phase error model: power law
424 5. (Optional) choose mesh type: we use triangular mesh
425 6. Inversion

426 API:

```
k = R2(typ='cR2') # initiate an R2 instance (considering there is IP data in
the input data)
k.createSurvey('IP_MICP_all.csv', ftype='Syscal') # import data
k.filterRecip(percent=5) # removing datapoints with > 5% reciprocal error
k.filterNested() # removing nested measurements
k.filterRangeIP(0,20) # setting phase shift range to 0 < -ϕ < 20
k.fitErrorPwl() # adding resistance power-law error model to data
k.fitErrorPwlIP() # adding phase power-law error model to data
```

```
k.err = 'True' # using error models (DC and IP) - automatically done in the
GUI when fitting the error model
k.createMesh(typ='trian') # create triangular mesh
k.param['a_wgt'] = 0 # "a_wgt" = 0 when there is individual resistance error
k.param['b_wgt'] = 0 # "b_wgt" = 0 when there is individual phase error
k.param['tolerance'] = 1.14 # based on data, field site and experience
k.param['min_error'] = 0.001 # based on data, field site and experience
k.invert() # run the inversion (and write cR2.in and protocol.dat
automatically)
k.showResults(attr='Magnitude(Ohm.m)') # show the inverted real conductivity
section
k.showResults(attr='Phase(mrad)') # show the inverted phase shift section
```

This time the data was successfully inverted (resistivity RMS misfit = 1.47 and phase RMS misfit = 1.11 in 3 iterations). Figure 11 shows the final inversion plots.

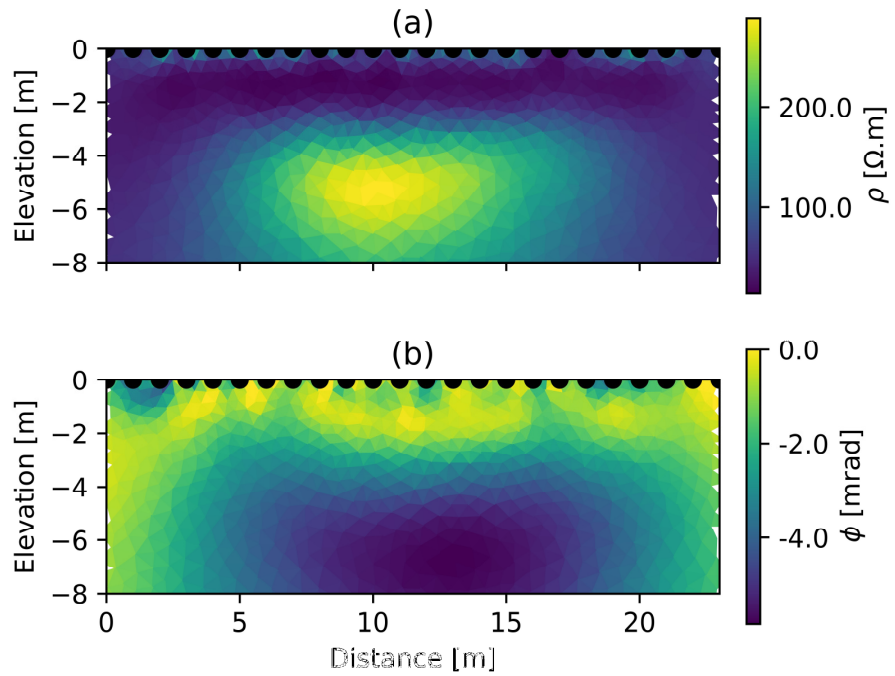

Figure 11: Inverted IP plots. (a) resistivity plot, (b) phase angle plot.

According to Saneiyan et al. (2019), the phase angle anomaly below -3.5 m is the area impacted by microbial induced carbonate precipitation (MICP) processes and ResIPy successfully shows this in the inversion plots. Saneiyan et al. (2019) show that a consistent increase in the phase angle below -3.5 m is observed during a 15-days experiment, confirming the impacted area by MICP has been detected by the IP survey successfully.

3.4 River: blocky resistivity inversion

ERT has been used in a number of studies for characterizing riverbeds, lakebeds and canals using waterborne and fixed arrays for both static and time-lapse investigations (e.g. Ball et al., 2006; Crook et al., 2008; Ward et al., 2013). In this example, we demonstrate how ResIPy allows the user to create a blocky region corresponding to the river and therefore better resolve the subsurface. In this case ResIPy allows the resistivity of elements of the mesh representing the river water column to be fixed, and regularization at the boundary between the river and surrounding region to be suppressed (i.e. using zones for regularization). The survey used here was collected using a transect that spanned the chalk fed river Lambourn (UK) and part of an adjacent riparian wetland. The inverted section is shown in Figure 12.

GUI:

1. Importing the data using the ‘Protocol’ file type
    a. Inputting the topography file for the electrodes
    b. Burying the substream electrodes
    c. Adding additional topography points to define where the river intersects the river bank
2. Meshing: triangular meshing is selected

a. Use the interactive plot to select a region corresponding to the river, define it as a separate (and fixed) zone and assign it a starting resistivity of 25 Ohm.m (value independently measured in the river)

3. Invert

API:

```
k = R2()
k.createSurvey('river-protocol.dat', ftype='Protocol')
# following lines will add electrode position, surface points and specify
if electrodes are buried or not. Similar steps are done in the GUI in (a),
(b), (c)
x = np.genfromtxt('river-elec.csv', delimiter=',')
k.setElec(x[:,:2]) # electrode positions
surface = np.array([[0.7, 92.30],[10.3, 92.30]]) # additional surface point
for the river level
buried = x[:,2].astype(bool) # specify which electrodes are buried (in the
river here)
k.filterElec([21, 23, 22, 2, 3]) # filter out problematic electrodes 21 and
2
k.createMesh(typ='trian', buried=buried, surface=surface, cl=0.2,
cl_factor=10)
xy = k.elec[1:21,[0,2]] # adding river water level using 2 topo points
k.addRegion(xy, res0=25, blocky=True, fixed=True) # fixed river resistivity
to 25 Ohm.m
k.param['b_wgt'] = 0.05 # setting up higher noise level
k.invert()
k.showResults(sens=False, vmin=1.2, vmax=2.2, zlim=[88, 93])
```

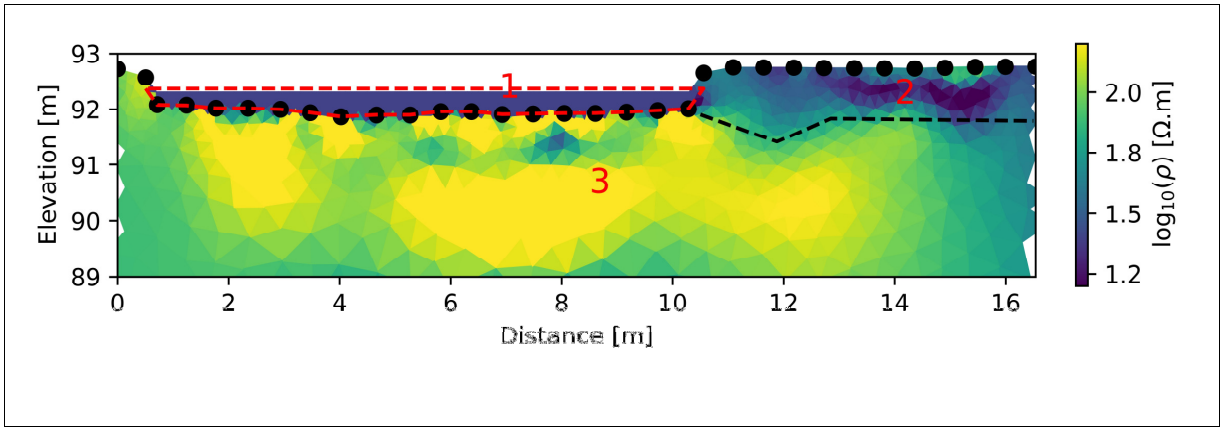

Figure 12: Inverted section showing (1) the river water corresponding to the fixed region surrounded by red dashed lines, (2) the peat layer, more conductive and (3) the gravels beneath more resistive. The block dashed line is an interpretation of the interface between the peat and the gravels.

3.5 Time-lapse monitoring of soil drying due to root water uptake

ResIPy allows users to perform inversion of time-lapse resistivity and IP surveys as well as batch surveys. These options need to be selected in the GUI before importing data. In both cases (time-lapse or batch survey) the user must select a directory containing the datasets rather than single data files (ResIPy automatically will ask for an import directory). Note that all files are imported in alphabetical order. For difference inversion, all surveys are automatically matched to keep only the quadrupoles common to all surveys.

A specific option to run the inversions in parallel is available in “Advanced” tab under “Inversion Settings” tab. In this case multiple inversions will be run on different logical processors, which will significantly speed up the total inversion process (if a multi core machine is used). Note that this consumes more memory for large meshes.

The dataset used in this example is a series of ERT surveys made between March to May 2017 at a wheat field maintained by Rothamsted Research at Woburn, UK. The aim of this study is to monitor the root water uptake of different wheat varieties for the purpose of selecting resilient lines (Whalley et al., 2017). ERT arrays were installed under different wheat varieties and left in place during the season. Regular ERT measurements were collected and converted to soil moisture content to observe the depth of the soil moisture depletion due to root water uptake. In this example, four ERT surveys of one variety are inverted using a time-lapse routine inversion

(difference inversion) that specially invert for change in resistivity (LaBrecque and Yang, 2001). All changes in resistivity are expressed as percentage difference compared to the background survey (15th March 2017). The inverted sections illuminate the drying pattern of the variety throughout the growing season (Figure 13).

In the GUI:

1. Importing data (for time-lapse: checking the 'Time-lapse' survey check box)
2. Fitting a power-law error model (applied on all data points for all time steps, the same global error model will then be used for computing error for each survey)
3. Create a triangular mesh
4. Inversion settings: in the advanced setting tab, we checked parallel inversion (i.e. multiple instances of the executable are run at the same time to speed up the inversion).
5. Inversion

API:

```
k = R2() # initiate an R2 instance
k.createTimeLapseSurvey('timeLapse/', ftype='Syscal') # import directory with
the data
k.fitErrorPwl() # fit a power-law
k.err = 'True' # tells the inversion to use the error model
k.createMesh(typ='trian', cl=0.5) # create a triangular mesh with a
characteristic length of 0.5
k.invert(parallel=True) # run the inversion (and write R2.in and protocol.dat
automatically), uses multiple cores if parallel is True
k.showResults(index=0) # show the first inverted section
k.showResults(index=1) # show the second inverted section
k.showResults(index=1, attr='difference(percent)') # show the differences
between the first and second survey
```

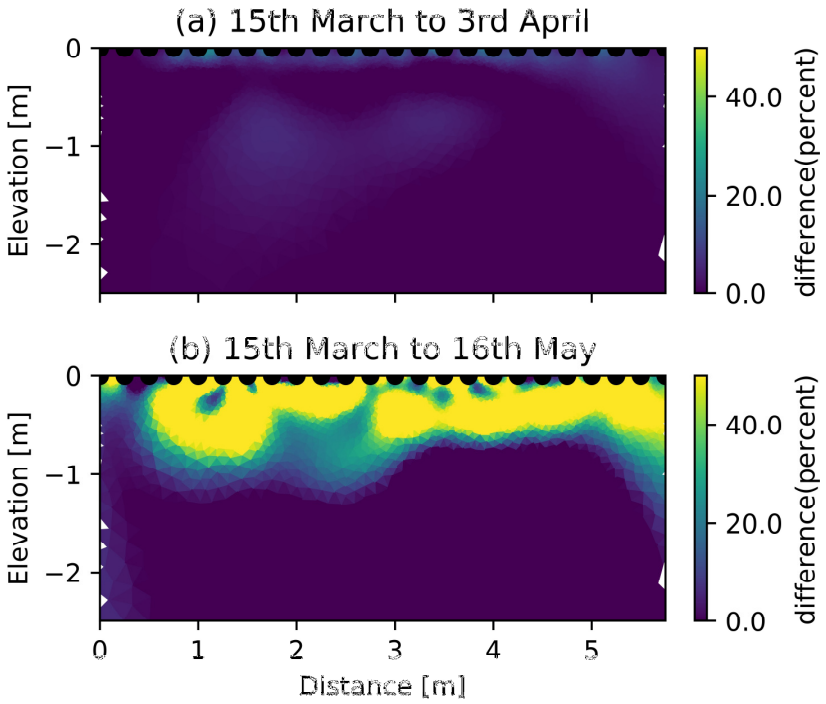

Figure 13: Time-lapse inverted section showing the differences from the background (15th March 2017) to (a) 3rd April and (b) 16th May 2017. There is an increasing resistivity in the subsurface, interpreted as an increasing drying due to root water uptake by the wheat. The change in resistivity reveals the depth of the drying which varies for different wheat varieties (Whalley et al., 2017).

4 Conclusion

ResIPy is a geophysical data analysis, modeling and inversion tool that simplifies the problem and allows users to have full control over sophisticated modeling/inversion parameters in an intuitive graphical user interface. ResIPy provides a platform for multi-disciplinary projects in which reliable results are produced in an easy to follow nonlinear user interface. ResIPy allows modeling and inversion of 2D and 3D resistivity and IP data, and is ideally suited for educational purposes. While most available inversion codes/software are capable of basic data filtering, ResIPy provides a thorough data cleaning routine. We have illustrated some of the key features

of ResIPy, showing, for example, how data filtering and error modeling can enhance data inversion, especially for IP surveys. ResIPy has been successfully used in multiple field and modeling situations using both the GUI and the API.

We believe this open source project will not only increase the usability of the mature R2/cR2 inversion/modeling codes, but also improve the accessibility of geophysics in interdisciplinary projects while also providing a powerful open source tool for teaching purposes.

5 Acknowledgements

We would like to thank the members of Lancaster – Rutgers Hydrogeophysical Knowledge Exchange group (HKEx) for their valuable feedback throughout this project.

6 Computer Code Availability

The data used in the examples and compiled standalone executables of the software are all available on the GitLab repository: https://gitlab.com/hkex/pyr2. Documentation of the API along with examples can be found at https://hkex.gitlab.io/pyr2.

7 Appendix 1

Below can be found a table summarising the main methods and functions available in ResIPy API. The list of arguments (signature) of the methods/functions are not displayed for the sake of simplicity but detailed help can be found in the documentation online (https://hkex.gitlab.io/pyr2/api.html).

521 *Table 1: API methods in ResIPy*

class	methods/attributes	what it does
R2 (R2.py)	createSurvey()	Import single survey dataset from file
	createTimeLapseSurvey()	Import time-lapse datasets from directory
	createBatchSurvey()	Import batch datasets from directory
	setElec()	Set the electrodes
	importElec()	Import electrodes position from file
	manualFiltering()	Manually select outliers point on the pseudo-section
	filterDip()	Filter dipole
	filterData()	Filter data (used in the outlier removal)
	fitErrorPwl()	Fit a power law error model resistivity (inherited from Survey)
	fitErrorLin()	Fit a linear error model resistivity (inherited from Survey)
	fitErrorPwlIP()	Fit a power law error model IP (inherited from Survey)
	fitErrorLinIP()	Fit a linear error model IP (inherited from Survey)
	fitErrorParabolaIP()	Fit a hyperbola error model to IP (inherited from Survey)
	createMesh()	Create a mesh (quadrilateral or triangular)
	showMesh()	Display the mesh
	addRegion()	Add a region of specific resistivity to the mesh
	createModel()	Interactive region definition
	createSequence()	Create sequence for forward modeling
	importSequence()	Import sequence for forward modeling
	forward()	Run the forward model

	write2in()	Write the .in file with all inversion settings
	write2protocol()	Write the protocol.dat file with the measurements
	invert()	Run the inversion
	showResults()	Show the inverted section
Mesh (meshTools.py)	show()	Display the mesh with the default attribute
	add_e_nodes()	Add electrode node indexes to mesh
	summary()	Prints summary information about the mesh.
	assign_zone()	Assigns 2D zones to the mesh.
	computeElmDepth()	Calculate depth to datum for each element
	write_vtk()	Writes a .vtk file
	write_attr()	Writes a tabbed file with element attributes
	paraview()	Show mesh in Paraview application if available
Functions in meshTools.py	import_vtk()	Import a .vtk file and returns an instance of Mesh
	quad_mesh()	Create a quadrilateral mesh (called by R2)
	trian_mesh()	Create a triangular mesh (called by R2)
	custom_mesh_import()	Import a .msh, .vtk, .dat mesh format and return mesh instance.
	systemCheck()	Returns and prints information about the user's system; Operating system, number of logical CPU cores detected and memory available.

8 References

Ball, L.B., Kress, W.H., Steele, G.V., Cannia, J.C., Andersen, M.J., 2006. Determination of canal leakage potential using continuous resistivity profiling techniques, Interstate and Tri-

State Canals, western Nebraska and eastern Wyoming, 2004 (USGS Numbered Series No. 2006–5032), Scientific Investigations Report.
Befus, K.M., 2018. *pyres*: a Python wrapper for electrical resistivity modeling with R2. J. Geophys. Eng. 15, 338–346. https://doi.org/10.1088/1742-2140/aa93ad
Benoit, S., Ghysels, G., Gommers, K., Hermans, T., Nguyen, F., Huysmans, M., 2018. Characterization of spatially variable riverbed hydraulic conductivity using electrical resistivity tomography and induced polarization. Hydrogeol. J. https://doi.org/10.1007/s10040-018-1862-7
Binley, A., 2015. 11.08 - Tools and Techniques: Electrical Methods, in: Schubert, G. (Ed.), Treatise on Geophysics (Second Edition). Elsevier, Oxford, pp. 233–259. https://doi.org/10.1016/B978-0-444-53802-4.00192-5
Binley, A., Hubbard, S.S., Huisman, J.A., Revil, A., Robinson, D.A., Singha, K., Slater, L.D., 2015. The emergence of hydrogeophysics for improved understanding of subsurface processes over multiple scales: The Emergence of Hydrogeophysics. Water Resour. Res. 51, 3837–3866. https://doi.org/10.1002/2015WR017016
Binley, A., Kemna, A., 2005. DC resistivity and induced polarization methods, in: Hydrogeophysics. Springer, pp. 129–156.
Boyd, J., Blanchy, G., Saneiyan, S., Binley, A., 2019. 3D geoelectrical problems with ResIPy, an open-source graphical user interface for geoeletrical data processing. FastTIMES 24.
Cockett, R., Kang, S., Heagy, L.J., Pidlisecky, A., Oldenburg, D.W., 2015. SimPEG: An open source framework for simulation and gradient based parameter estimation in geophysical applications. Comput. Geosci. 85, 142–154. https://doi.org/10.1016/j.cageo.2015.09.015
Crook, N., Binley, A., Knight, R., Robinson, D.A., Zarnetske, J., Haggerty, R., 2008. Electrical resistivity imaging of the architecture of substream sediments. Water Resour. Res. 44. https://doi.org/10.1029/2008WR006968
Flores Orozco, A., Williams, K.H., Kemna, A., 2013. Time-lapse spectral induced polarization imaging of stimulated uranium bioremediation. Surf. Geophys. 11, 531–544. https://doi.org/10.3997/1873-0604.2013020
Geuzaine, C., Remacle, J.-F., 2009. Gmsh: A 3-D finite element mesh generator with built-in pre- and post-processing facilities. Int. J. Numer. Methods Eng. 79, 1309–1331. https://doi.org/10.1002/nme.2579
Hunter, J.D., 2007. Matplotlib: A 2D Graphics Environment. Comput. Sci. Eng. 9, 90–95. https://doi.org/10.1109/MCSE.2007.55
Johnson, T.C., Hammond, G.E., Chen, X., 2017. PFLOTRAN-E4D: A parallel open source PFLOTRAN module for simulating time-lapse electrical resistivity data. Comput. Geosci. 99, 72–80. https://doi.org/10.1016/j.cageo.2016.09.006
Karaoulis, M., Revil, A., Tsourlos, P., Werkema, D.D., Minsley, B.J., 2013. IP4DI: A software for time-lapse 2D/3D DC-resistivity and induced polarization tomography. Comput. Geosci. 54, 164–170. https://doi.org/10.1016/j.cageo.2013.01.008
Koestel, J., Kemna, A., Javaux, M., Binley, A., Vereecken, H., 2008. Quantitative imaging of solute transport in an unsaturated and undisturbed soil monolith with 3-D ERT and TDR. Water Resour. Res. 44, n/a–n/a. https://doi.org/10.1029/2007WR006755
LaBrecque, D.J., Yang, X., 2001. Difference inversion of ERT data: A fast inversion method for 3-D in situ monitoring. J. Environ. Eng. Geophys. 6, 83–89.
McLachlan, P.J., Chambers, J.E., Uhlemann, S.S., Binley, A., 2017. Geophysical characterisation of the groundwater–surface water interface. Adv. Water Resour. 109, 302–319. https://doi.org/10.1016/j.advwatres.2017.09.016
Mewes, B., Hilbich, C., Delaloye, R., Hauck, C., 2017. Resolution capacity of geophysical monitoring regarding permafrost degradation induced by hydrological processes. The Cryosphere 11, 2957–2974. https://doi.org/10.5194/tc-11-2957-2017

Mwakanyamale, K., Slater, L., Binley, A., Ntarlagiannis, D., 2012. Lithologic imaging using complex conductivity: Lessons learned from the Hanford 300 Area. GEOPHYSICS 77, E397–E409. https://doi.org/10.1190/geo2011-0407.1

Nath, S.K., Shahid, S., Dewangan, P., 2000. SEISRES—a visual C++ program for the sequential inversion of seismic refraction and geoelectric data. Comput. Geosci. 26, 177–200.

Ntarlagiannis, D., Robinson, J., Soupios, P., Slater, L., 2016. Field-scale electrical geophysics over an olive oil mill waste deposition site: Evaluating the information content of resistivity versus induced polarization (IP) images for delineating the spatial extent of organic contamination 135, 418–426. https://doi.org/10.1016/j.jappgeo.2016.01.017

Oldenburg, D.W., Li, Y., 1999. Estimating depth of investigation in dc resistivity and IP surveys. Geophysics 64, 403–416.

Orozco, A.F., Kemna, A., Zimmermann, E., 2012. Data error quantification in spectral induced polarization imaging 77, E227–E237. https://doi.org/10.1190/geo2010-0194.1

Pidlisecky, A., Knight, R., 2008. FW2_5D: A MATLAB 2.5-D electrical resistivity modeling code. Comput. Geosci. 34, 1645–1654.

Rücker, C., Günther, T., Wagner, F.M., 2017. pyGIMLi: An open-source library for modelling and inversion in geophysics. Comput. Geosci. https://doi.org/10.1016/j.cageo.2017.07.011

Saneiyan, S., Ntarlagiannis, D., Ohan, J., Lee, J., Colwell, F., Burns, S., 2019. Induced polarization as a monitoring tool for in-situ microbial induced carbonate precipitation (MICP) processes. Ecol. Eng. 127, 36–47. https://doi.org/10.1016/j.ecoleng.2018.11.010

Saneiyan, S., Ntarlagiannis, D., Werkema, D.D., Ustra, A., 2018. Geophysical methods for monitoring soil stabilization processes. J. Appl. Geophys. 148, 234–244. https://doi.org/10.1016/j.jappgeo.2017.12.008

Schlumberger, C., 1920. Etude sur la prospection electrique du sous-sol. Gauthier-Villars.

Slater, L.D., Lesmes, D., 2002. IP interpretation in environmental investigations. GEOPHYSICS 67, 77–88. https://doi.org/10.1190/1.1451353

Uhlemann, S., Wilkinson, P.B., Maurer, H., Wagner, F.M., Johnson, T.C., Chambers, J.E., 2018. Optimized survey design for Electrical Resistivity Tomography: combined optimization of measurement configuration and electrode placement 36.

Vanella, D., Cassiani, G., Busato, L., Boaga, J., Barbagallo, S., Binley, A., Consoli, S., 2018. Use of small scale electrical resistivity tomography to identify soil-root interactions during deficit irrigation. J. Hydrol. 556, 310–324. https://doi.org/10.1016/j.jhydrol.2017.11.025

Ward, A.S., Gooseff, M.N., Singha, K., 2013. How Does Subsurface Characterization Affect Simulations of Hyporheic Exchange? Groundwater 51, 14–28. https://doi.org/10.1111/j.1745-6584.2012.00911.x

Whalley, W.R., Binley, A., Watts, C.W., Shanahan, P., Dodd, I.C., Ober, E.S., Ashton, R.W., Webster, C.P., White, R.P., Hawkesford, M.J., 2017. Methods to estimate changes in soil water for phenotyping root activity in the field. Plant Soil 415, 407–422. https://doi.org/10.1007/s11104-016-3161-1

Wood, J., 2017. Roman Lancaster: The Archaeology of Castle Hill. Br. Archaeol. 38–45.

Zarif, F., Kessouri, P., Slater, L., 2017. Recommendations for Field-Scale Induced Polarization (IP) Data Acquisition and Interpretation. J. Environ. Eng. Geophys. 16.

524

525

526

527

528

529