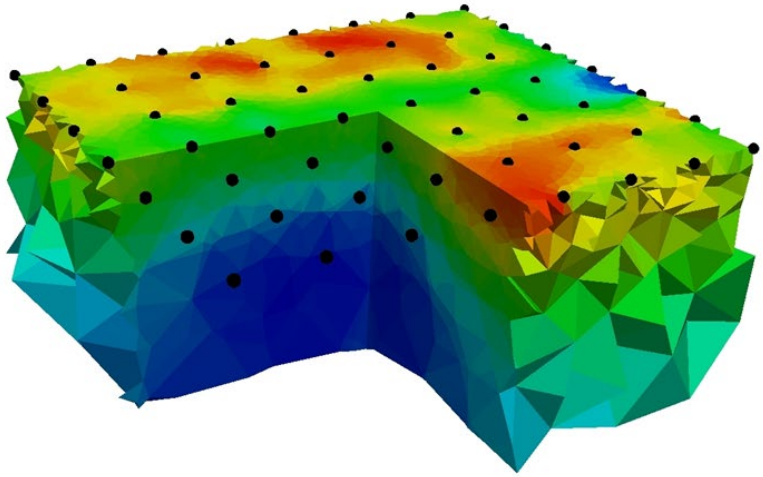
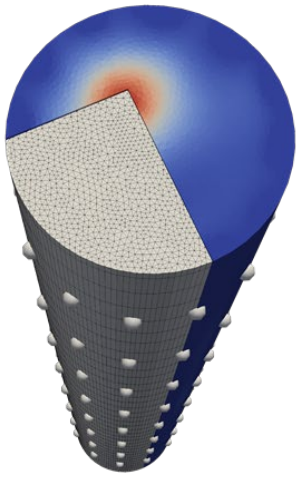
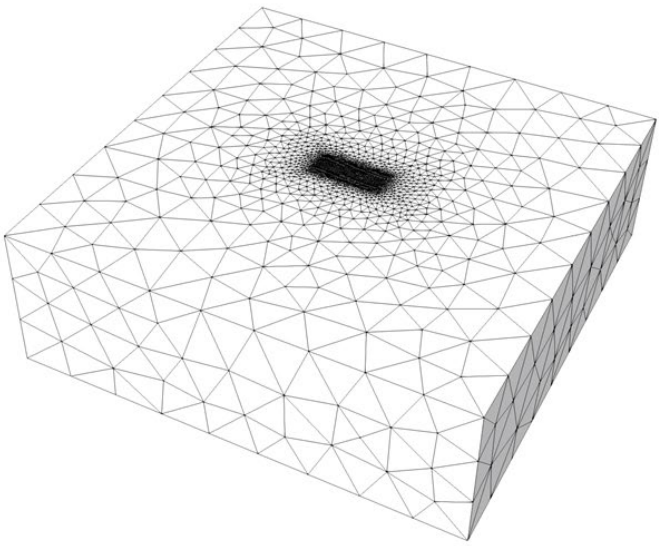
R3t

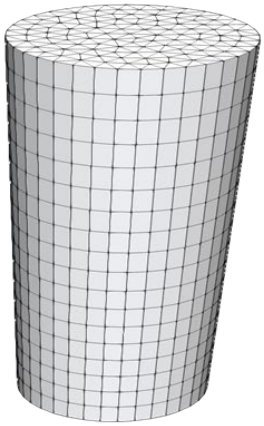
version 2.32

Andrew Binley
Lancaster University
February, 2024









1. Version history

Changes from earlier version (2.31)
If starting model satisfies convergence the model is output (previously a blank file was saved).
Changes from earlier version (2.3)
Corrected output of difference results for non-converged case.
Changes from earlier version (2.2)
Added output of percentage difference in resistivity in vtk file when difference regularisation is used.
Changes from earlier version (2.1)
Added option to read in a mesh file containing node and element connections to reduce computation time at execution.
Changes from earlier version (2.01)
Added resolution matrix calculations and option to store jacobian and roughness matrix.
Changes from earlier version (2.0)
Minor bug fixes and output of number of measurements in line 1 of forward model output (to be consistent with ***R2* v4.0**).
Changes from earlier version (1.9)
Major changes to input file ***R3t.in*** to be consistent with ***R2* v4.0**.
Added input file checks. Apparent resistivity checks on input. Electrode checking on input. Added misfit target decrease option as in ***R2***. Output format of **fxxx_err.dat** changed. Added roughness to main output log.
Changes from earlier version (1.8a)
Output file names changed to be consistent with ***R2* v3.1**
Changes from earlier version (1.8)
Added a check during input of measurements to ensure that zero resistances are not read.
Changes from earlier version (1.7a)
Added option for different regularisation smoothing in regularisation zones (see **mesh3d.dat**).
An alternative procedure has been added for the inverse steps, allowing the regularisation parameter alpha to be better controlled by the user. Vtk file now contains the parameter zones defined in ***mesh3d.dat***
Changes from earlier version (1.7)
Minor output modifications. Some input error checking added.
Changes from earlier version (1.6)
Changed regularisation options (see line 7 in ***R3t.in***). Input changed in **R3t.in**
Allows regions of the mesh to have fixed resistivity (see **mesh3d.dat**).
Changes from earlier version (1.5)
Tetrahedral mesh option added.
Changes from earlier version (1.4)
Dynamic memory allocation – no need for fixed array sizes. No change in input.
Changes from earlier version (1.3b)
Version 1.4 now includes the option to apply singularity removal in the computation of voltages and thus provides a more accurate forward model for both forward and inverse modes (see line 2 in ***R3t.in***). This version also allows anisotropic smoothing (see line 8 of ***R3t.in***). In addition, the linear solver in the forward calculations will now use out of core storage if insufficient in core (RAM) is available.
Changes from earlier version (1.3a)
Version 1.3b allows the user to specify the volume of output, which may be particularly useful for meshes setup for 'infinite' boundary conditions. (Thanks to Giorgio Cassiani for supplying a polygon bounding routine to help with this).

Changes from earlier version (1.3)

Version 1.3a outputs results (resistivity, log10 resistivity, sensitivity map, electrode co-ordinates) in vtk format, allowing easy visualisation with ParaView.

2. Computer requirements for *R3t*

In this release two versions have been compiled for the Windows environment. A 64bit version, **R3t.exe**, is provided in the package. Linux users should be able to run ***R3t*** with the command "wine R3t.exe" (thanks to Rodolphe Cattin for this tip).

NOTE 1: ***R3t*** *is provided as a standalone executable. It does not need to be installed – the executable is put in the folder containing the input files and run from there. Output files will be created in the same folder. Alternatively, you can create a shortcut to* **R3t.exe** and copy to the shortcut to the working folder.

NOTE 2: *You will be able to run* ***R3t*** *by double clicking the executable. However, if the program stops abruptly (for example, due to an error in the input file or if you are trying to run an executable compiled for a different processor architecture) then you will not see any error message on the screen since the window will disappear. Therefore, it is advisable to run* ***R3t*** *from the Command Prompt (just run CMD from the Start Menu – you may need to move your working directory and run* ***R3t*** *from there).*

NOTE 3: *All input files should be prepared with a text editor. [I prefer to use* ***TextPad*** *(www.textpad.com) because it allows much greater editing facilities although any text editor will work]. It is important that you do not include tabs in the files. These are often inserted if you copy and paste from Excel, for example. You should convert these tabs to spaces (****TextPad*** *will allow you to set this up to happen automatically).*

3. **Introduction to *R3t***

R3t is a forward/inverse solution for 3D current flow in a tetrahedral or triangular prism mesh. The mesh is made up of a set of elements. Parameters (for the inverse solution) are made up of one or more elements. The user must define the mesh for ***R3t*** as a series of elements, each with either 4 nodes (tetrahedron) or 6 nodes (triangular prism). The user must also specify the position of the electrodes within the mesh. The electrodes can be located anywhere in the mesh, provided they fall on node points. Electrodes are specified at node points. These are the corners of the elements. The boundary conditions along all boundaries of the mesh are Neumann conditions (zero flux) and therefore if you are investigating a half space you must extend left, right and lower boundaries of the mesh to some distance away from the area of investigation (typically 5 to 10 times the distance – see later). The mesh can be made up of either tetrahedral or triangular prism mesh elements.

R3t will output calculated parameters (resistivity) for the entire mesh (in inverse mode) and the user must extract results for the region they wish to study. The region is parameterised in terms of resistivity blocks by grouping patches of elements.

Measurements are defined in a separate file as a set of four electrode indices. Each electrode is defined as a "string" number and an "electrode" number (note that the "string" index is used simply to help group electrodes, e.g. in surface lines or boreholes). The "string" index can be the same for all electrodes if the user wishes not to use this labelling. Measurements are input as transfer resistances (not apparent resistivity). This is to allow more flexible geometries to be analysed (e.g. columns and tanks). Note that the polarity of the transfer resistance must be included in the measurement (since they can be positive or negative).

The current version does not have upper limits set for the size of the problem that can be solved. However, it is important that the user has some appreciation of whether the problem they are trying to solve is realistic for their given hardware. Large problems in inverse mode can be memory hungry. As soon as the user's RAM is used then the computer will start using virtual memory (paging to disk) which can be very slow. To help the user assess memory needs ***R3t*** will output an estimate of the memory needs early on in its execution. For large problems it is important that the user compares this with physical memory (RAM) that is available.

For information on solving DC resistivity forward and inverse problems see Binley and Slater (2020), Binley (2015) and Binley and Kemna (2005). Contact the author for a digital copy of Binley (2015).

R3t is provided for non-commercial use. Any users wishing to use ***R3t*** for commercial applications should contact the author.

4. **Mesh generation and parameterisation**

R3t models the voltage field and determines resistivity parameters based on a 3D mesh of tetrahedral or triangular prism elements (Figure 1). Electrodes must be defined at node points anywhere in the mesh. For field based applications the mesh should be extended out to a reasonable distance (laterally and vertically) to account for 'infinite' current flow. There is no need to retain a fine discretisation in these 'infinite' boundary regions: it is good practice to let the elements gradually increase in size laterally and vertically outside the region of investigation. It is good practice to define an "inner zone" and an "outer zone" in the mesh for semi-infinite problems. The inner zone will have fine discretisation, whereas the discretisation in the outer zone is coarser. A good rule of thumb is to place the 'infinite' boundaries 5L away from the electrode array, where L is the length of the longest current dipole.

Triangular prism meshes are effectively "structured" since the mesh is formed from a triangular mesh in the x-y plane that is projected in the z direction in layers. These layers (the element height) can have different thicknesses but each layer must be parallel to the others. This type of mesh may be convenient for fairly simple geometries but it is impossible to create complex x-y-z boundaries, for example, topography. Furthermore, a triangular prism mesh can be very inefficient (computationally) because, in a half space problem that would be encountered for a field study, as we extend the mesh away from the region of interest to represent infinite x-y-z boundaries by keeping the same element height we can end up with very thin but wide elements. A tetrahedral mesh overcomes this as it allows us full flexibility in the shape of elements. Thus we can have small elements in the area where highest potential gradients exist (where the survey is being carried out) but vary large elements close to 'infinite' boundaries (see examples later). Furthermore, the ability to incorporate complex topography permits the full range of geometries in the model.

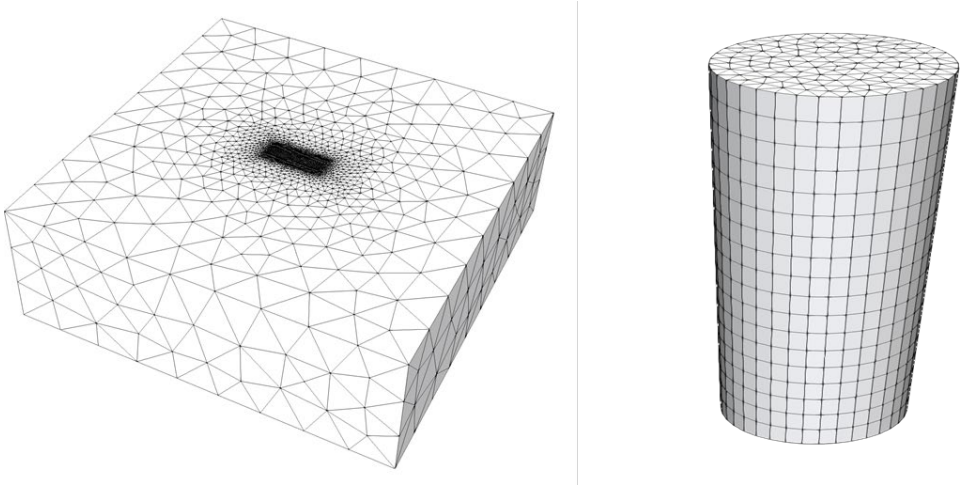


Figure 1. Example tetrahedral and triangular prism meshes.

The resistivity does not vary within each element in ***R3t***. The resistivity distribution is defined (for a forward model or starting condition for an inverse model) using the element mesh. For inversion, parameter boundaries must be defined. The finest (and simplest) discretisation is achieved by having the parameter boundaries equal to the element boundaries – in this case each parameter is assigned to a finite element that is unique to that parameter. For coarser parameter discretisation (and consequently faster execution of the code) parameters can be defined as collections of elements (see Figures 2 and 3). If this is done then each element is assigned to a parameter number which will be common to more than one finite element. The advantage of having a coarse parameter mesh and a fine finite element mesh is that more accurate voltages (for each forward modelling step) are computed on the fine element mesh, while resistivities are determined on a coarser parameter mesh allowing faster execution.

For a triangular prism mesh, each element contains 6 nodes. These nodes should be numbered (as in Figure 4) so that the lower triangle forming the prism contains nodes 1, 2 and 3 (numbered in a counter-clockwise manner) and the upper triangle contains nodes 4, 5 and 6.

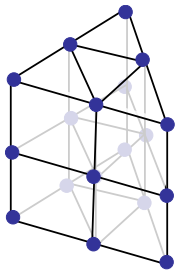


Figure 2: Grouping of triangular prism finite elements to form a single triangular prism parameter cell.

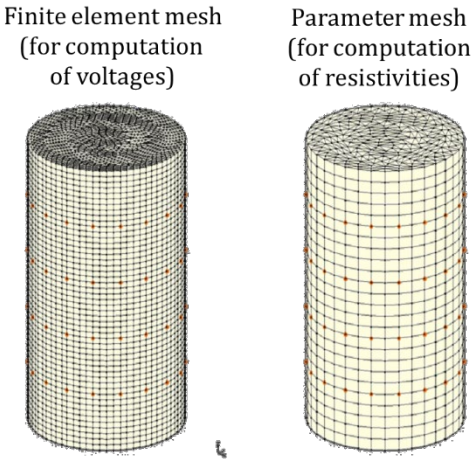


Figure 3. Example finite element and parameter discretisation for a triangular prism mesh in ***R3t***

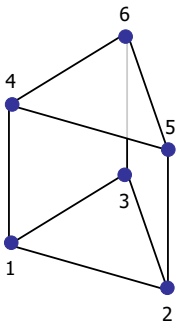


Figure 4: Node numbering in a triangular prism element

The parameter mesh can also be 'zoned' to permit sharp contrasts over boundaries that are known *a priori* (e.g. at a water table). To do this each parameter is assigned a zone number (see zone_elem below). If the zone number is the same for all parameters then the inversion will seek a smooth model based on the gradient of (log) resistivity across all parameter boundaries. If different zones are used then there will be zero smoothing imposed across the boundary between zones. Each zone can have different smoothing. For example, if a zone represents a borehole then the user may wish to have enhanced smoothing in this zone. In order to do this a scalar for each zone is given (see details below on input to **mesh3d.dat**).

In addition, the user may wish to keep some resistivities fixed throughout the inversion (e.g. in regions where the resistivity is known *a priori*). To do this, the user specifies the parameter number for a given element as "0". All elements that are designated with such a parameter number will remain fixed to the starting resistivity (defined in **R3t.in**).

R3t doesn't contain a mesh generator – the user needs their own software to do this. However, there are a number of good meshing tools available. ***Gmsh*** (see http://www. gmsh.info) is a powerful 3D

finite element mesh generator with a large user base with video tutorials available online. Alternatively, software for general finite element analysis (e.g. ***COMSOL***) contain mesh generators, as do software for specific applications. ***Gmsh*** can be used to create 3D tetrahedral meshes directly. A geo file is first created, which defines the geometry. Then the meshing is done and a .msh file is created, which contains the element geometry. This file then needs to be converted to a **mesh3d.dat** format. Included in one of the examples is a geo file for ***Gmsh***. Jimmy Boyd (BGS/Lancaster) has written a python script to convert a .msh file from ***Gmsh*** to a **mesh3d.dat** format file for ***R3t***. This can be found in /Mesh utilities/Jimmy Boyd. An executable and source code is provided in the folder.

Gmsh can create triangular prism meshes but it is a bit awkward to do. One approach is to create a 2D triangular element mesh and then build a triangular prism mesh (following the node number convention in Figure 4) from a user's bespoke program.

5. **Inverse modelling in *R3t***

In ***R3t*** an iterative process solves the following equations:

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{W}_{\mathrm{d}}^{\mathrm{T}}\mathbf{W}_{\mathrm{d}}\mathbf{J}+\alpha\mathbf{R}\right)\Delta\mathbf{m}=\mathbf{J}^{\mathrm{T}}\mathbf{W}_{\mathrm{d}}^{\mathrm{T}}\left(\mathbf{d}-\mathbf{f}(\mathbf{m}_i)\right)-\alpha\mathbf{R}\mathbf{m} \quad (1)$$

$$\mathbf{m}_{\mathrm{i+1}}=\mathbf{m}_{\mathrm{i}}+\Delta\mathbf{m}, \quad (2)$$

where:
J is the Jacobian, such that $J_{i,j}=\partial d_i/\partial m_j$,
d is the data vector,
$\mathbf{m}_i$ is the parameter vector at iteration *i*,
$\mathbf{W_d}$ is the data weight matrix, assumed to be diagonal, with diagonal values $W_{i,i}=1/\epsilon_i$, where ϵ_i is the standard deviation of measurement i,
α is the regularisation (or smoothing) parameter,
R is the roughness matrix, which describes the connectivity of parameter blocks,
$\Delta\mathbf{m}$ is update in parameter values at each iteration,
$\mathbf{f}(\mathbf{m})$ is the forward model for parameters **m**.

In ***R3t*** the parameters are the logarithm of the electrical conductivity in each element (or group of elements that form a parameter block). The data are either transfer resistances or, if log data selected (see detailed input instructions later for **R3t.in**) then the logarithm of transfer resistances are used. Note that this does not require only positive data, however, the forward model computed value should be the same polarity, otherwise the difference is not defined. If ***R3t*** encounters a difference in polarity between modelled and measured values during an inversion then these measurements are ignored.

Equations (1) and (2) are solved repeatedly until satisfactory convergence is achieved. In ***R3t*** this is defined by the data misfit reaching a required tolerance. If we express data misfit as a root mean square error, i.e.

$$\mathrm{RMS}=\sqrt{\frac{1}{N}\sum\left(\frac{d_i-f_i(\boldsymbol{m})}{\epsilon_i}\right)^2}, \quad (3)$$

where *N* is the number of measurements, then the target tolerance should be 1 (following a chi-squared distribution).

Equations (1) and (2) result from the minimisation of an objective function composed of a data misfit and a model misfit. The former describes the mismatch between the observations (**d**) and the forward model (f(**m**)) and can be expressed as:

$$\Psi_d=(\mathbf{d}-\mathbf{f}(\mathbf{m}))^T\mathbf{W}_{\mathrm{d}}^T\mathbf{W}_{\mathrm{d}}(\mathbf{d}-\mathbf{f}(\mathbf{m})). \quad (4)$$

The model misfit can be expressed as:

$$\Psi_m=\mathbf{m}^T\mathbf{R}\mathbf{m}. \quad (5)$$

In an Occam's inversion we seek to minimise:

$$\Psi_{total}=\Psi_d+\alpha\Psi_m, \quad (6)$$

for the largest α, i.e. we wish to obtain the smoothest distribution of resistivity that is consistent with the observed data. ***R3t*** achieves this through an iterative process in which equation (1) is solved and equation (2) applied. At each iteration α can change, keeping it as large as possible. ***R3t*** does a line search for α at each Gauss Newton iteration (using up to 10 values of α). ***R3t*** computes a reasonable starting value for α at the beginning of the process by assessing an equal balance of the terms in the

brackets of left hand side of equation (1). This often results in satisfactory convergence within a few iterations. However, this can lead to unsmooth models as the inversion process is attempting to find a solution too quickly. It can be beneficial to slow the process. To do this the user can select the maximum change in misfit during each iteration (using the target_decrease parameter – see **R3t.in** input). The user is recommended to experiment with either solution strategies.

During each iteration ***R3t*** will output values of Ψ_d (reported as a root mean square (RMS) error) and Ψ_m (reported as "roughness").

The model misfit in equation (5) describes the roughness of the variation in model parameter. In some cases we may wish to obtain a model that is smooth in terms of the difference between model parameter and some reference model. Such a situation is useful for time-lapse inversion (see later). In this case we can express a model misfit as:

$$\Psi_m = (\mathbf{m} - \mathbf{m}_0)^T \mathbf{R} (\mathbf{m} - \mathbf{m}_0), \tag{7}$$

where $\mathbf{m}_0$ is the reference parameter model.

For this approach equation (1) needs to be modified to:

$$\left(\mathbf{J}^{\mathrm{T}} \mathbf{W}_{\mathrm{d}}^{\mathrm{T}} \mathbf{W}_{\mathrm{d}} \mathbf{J} + \alpha \mathbf{R}\right) \Delta \mathbf{m} = \mathbf{J}^{\mathrm{T}} \mathbf{W}_{\mathrm{d}}^{\mathrm{T}} \left(\mathbf{d} - \mathbf{f}(\mathbf{m}_i)\right) - \alpha \mathbf{R} (\mathbf{m} - \mathbf{m_0}). \tag{8}$$

In ***R3t*** we refer to this as a *difference regularisation*. This can be used for any starting model specified by the user. We can also apply the approach of LaBrecque and Yang (2001) with this implementation. They propose the solution of equation (8) with the term:

$$\left(\mathbf{d} - \mathbf{f}(\mathbf{m}_i)\right) \tag{9}$$

defined as:

$$(\mathbf{d} - \mathbf{d}_0) - \left(\mathbf{f}(\mathbf{m}_i) - \mathbf{f}(\mathbf{m}_0)\right). \tag{10}$$

To implement this the user should compute the forward model with the reference dataset $\mathbf{d}_0$, this gives $\mathbf{f}(\mathbf{m}_0)$. Then, dataset $\mathbf{d}$ should be modified to:

$$\left(\mathbf{d} - \mathbf{d}_0 + \mathbf{f}(\mathbf{m}_0)\right). \tag{11}$$

By then selecting the difference inversion option and using $\mathbf{m}_0$ as the starting model, the method of LaBrecque and Yang (2001) will be implemented. Note that in ***R3t*** $\mathbf{m}_0$ is the logarithm of the conductivity, but for user input the resistivity is specified.

An alternative regularisation could penalise departure from a reference model, i.e. we could express model misfit as:

$$\Psi_m = (\mathbf{m} - \mathbf{m}_0)^T (\mathbf{m} - \mathbf{m}_0). \tag{12}$$

This would not lead to a smooth model, however.

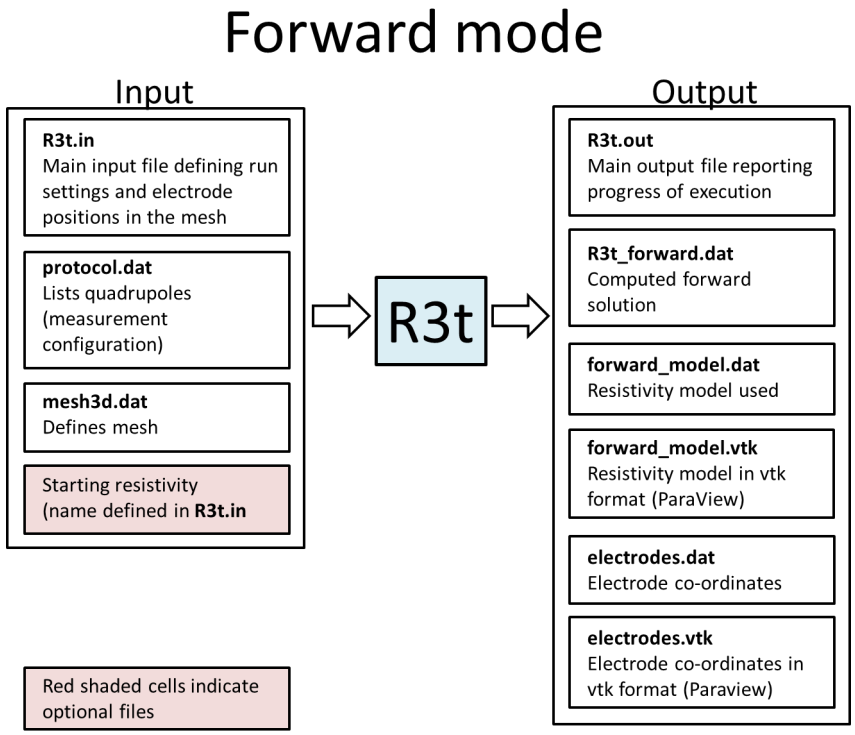
In order to accommodate such an approach in ***R3t***, the following model misfit option is implemented:

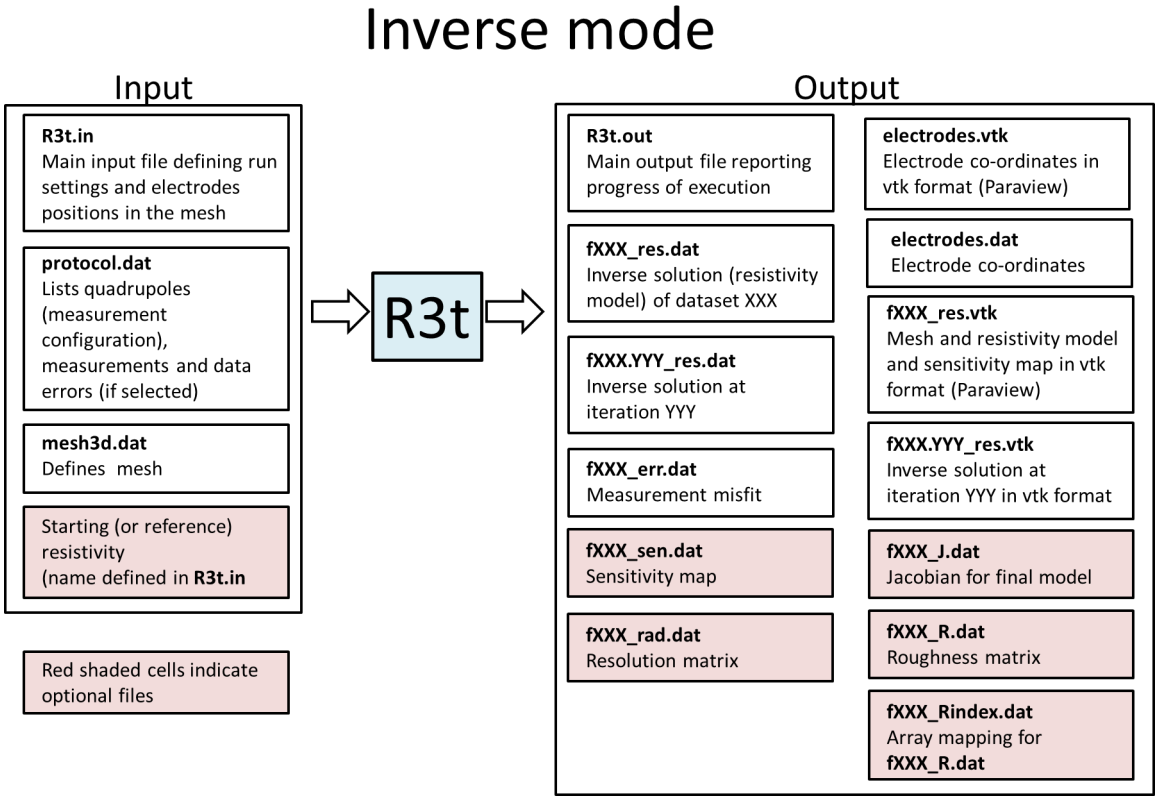
$$\Psi_m = (\mathbf{m} - \mathbf{m}_0)^T \mathbf{R} (\mathbf{m} - \mathbf{m}_0) + \alpha_s (\mathbf{m} - \mathbf{m}_0)^T (\mathbf{m} - \mathbf{m}_0), \tag{13}$$

where α_s is a weighting factor: a high value forces consistency with the reference model; a low value forces smoothing of the difference. In ***R3t*** this is referred to as a *background regularisation*. The three types of regularisation are specified by the user with reg_mode in **R3t.in** (see later).

6. Input and output files

R3t requires at least three data files: **R3t.in**, **protocol.dat** and **mesh3d.dat**. Note that an additional file is needed if you wish to compute a forward model for an inhomogeneous resistivity distribution or if you want to use an inhomogeneous resistivity distribution as a starting model or reference model for an inversion.





In *forward* mode ***R3t*** will output six files:

- **R3t.out** which will contain main log of execution.

- **R3t_forward.dat** will contain the forward model for the electrode configuration in **protocol.dat** The format of **R3t_forward.dat** is the same as **protocol.dat** but with 2 extra columns: the first contains the calculated resistances and the second contains the calculated apparent resistivities (note that apparent resistivities are computed assuming that the z=0 is the flat surface of a half space; if this is not the case (e.g. if the region is a bounded domain such as a cylindrical column) then the apparent resistivities should be ignored). If ***R3t*** computes the absolute value of the geometric factor to be less than 1e-10m then the apparent resistivity that is output is assigned to a value of -100000.00000 Ωm in the output file. This does not necessarily mean that the computed transfer resistances are poorly estimated – more likely that the assumptions of infinite boundaries or a flat topography at z=0 is not applicable.

- **forward_model.dat** which will contain the resistivity model. The file will contain a value for each finite element in the grid (within the zone specified by the user). The file will have five columns: the element centroid x co-ordinate (in m), the element centroid y co-ordinate (in m), the element centroid z co-ordinate (in m), the element resistivity (in Ωm) and the element $\log_{10}$ resistivity (in $\log_{10}$ Ωm).

- **forward_model.vtk** will contain the resistivity model in vtk format. Both resistivity and $\log_{10}$ resistivity are stored. This can be loaded in ***ParaView*** (see later in this document), allowing easy visualisation with the mesh outline. Note that values are only output for the region specified by the user.

- **electrodes.dat** contains the co-ordinates of the electrodes. The values are in three columns: x,y,z (in m).

- **electrodes.vtk** contains the co-ordinates of the electrodes in vtk format. The values are in three columns: x,y,z. Use this file if you are working with ***ParaView*** to look at the resistivity images. Once you have opened the **electrodes.vtk** file in ***ParaView*** you select "apply" then select "Representation" as "Point Gaussian" and change the radius of the symbols.

In *inverse* mode ***R3t*** will output several files:

- **R3t.out** which will contain main log of execution.

- **f001_res.dat** will contain the resistivity result of the inverse solution. **f001_res.dat** will contain a value for each finite element in the grid (within the zone specified by the user). The file will have five columns: the element centroid x co-ordinate (in m), the element centroid y co-ordinate (in m), the element centroid z co-ordinate (in m), the element resistivity (in Ωm) and the element $\log_{10}$ resistivity (in $\log_{10}$ Ωm).

- If selected, **f001_sen.dat** will contain the diagonal of the matrix $[\mathbf{J}^T \mathbf{W}^T \mathbf{W} \mathbf{J}]$ (see Binley, 20015; Binley and Kemna, 2005) which gives an idea of the measurement sensitivity. You will get a value for all elements in the region defined by the user. High values indicate high sensitivity to data, low values indicate poor sensitivity. The format is the same as **f001_res.dat**. Plot on a log scale (i.e. plot the fifth column). Note that if parts of the mesh are output where the user fixed the parameter values (see input to **mesh3d.dat**) then the then the sensitivity values will be output as 10^{-99}.

- If selected, **f001_rad.dat** will contain the diagonal of the matrix resolution matrix (see Binley, 20015; Binley and Kemna, 2005). You will get a value for all elements in the region defined by the user. High values indicate high sensitivity to data, low values indicate poor sensitivity. The format is the same as **f001_res.dat**. Plot on a log scale (i.e. plot the fifth column). Note that if parts of the mesh are output where the user fixed the parameter values (see input to **mesh3d.dat**) then the sensitivity values will be output as 10^{-99}. If computed resolution matrix diagonal values are negative a value of $\log_{10} = -9.999$ is output.

- If you select sensitivity map calculation and output of the jacobian then three other files will be output: **f001_J.dat** (the jacobian); **f001_R.dat** (the roughness matrix); **f001_Rindex.dat** (the mapping of the roughness matrix, as the latter is stored in compressed form). All three files will have two integers in line 1. These are the array sizes. Note that the first subscript varies slowest, so if line 1 is "3 6" then an array x of size (3,6) is stored as: x(1,1), x(1,2),x(1,3), x(2,1), x(2,2),x(2,3), … x(6,1), x(6,2),x(6,3). The jacobian is of size number of measurements × number of parameters. The roughness matrix is stored as number of parameters × N, where N will be either 5 (tetrahedral element) or 6 (triangular prism). The file **f001_Rindex.dat** shows the mapping of the compressed form of the roughness matrix in **f001_R.dat**, e.g. if line 2 (i.e. the line with the first row of the roughness matrix, given that the first line of the file contains the size of the array) of **f001_Rindex.dat** is "1 2 13 0 0" then in line 2 of **f001_R.dat** the roughness matrix entries R(1,1), R(1,2), R(1,13) are stored. Note that the zeros in **f001_Rindex.dat** mean than no entry exists because this parameter (parameter 1) is only connected to two other parameters (parameters 2 and 13).

- **f001.vtk** will contain the resistivity, $\log_{10}$ resistivity, $\log_{10}$ 'sensitivity' (or $\log_{10}$ resolution matrix) and parameter zones in vtk format. If reg_mode is 2 (see description of **R3t.in** later) then the percentage difference between the resistivity and the starting model is also output. This file can be loaded in ***ParaView*** (see later in this document), allowing easy visualisation with the mesh outline. Note that values are only output for the region specified by the user.

- **f001_err.dat** will contain fourteen columns. The first eight columns contain the quadrupole configuration (as in **protocol.dat**). The next column is the normalised data misfit. This is followed by the observed and modelled data recorded as apparent resistivity. The next two columns shows the original and final data weights (i.e. reciprocal of data standard deviation in same units as data). The last column shows a "1" if any weights have been changed during the inversion, otherwise a "0" will appear. If the inversion works successfully then the normalised data misfit values should follow a Gaussian distribution with zero mean and unit standard deviation (e.g. 99% of the values should lie between -3 and +3).

- **electrodes.dat** contains the co-ordinates (in m) of the electrodes as described above.

- **electrodes.vtk** contains the co-ordinates of the electrodes in vtk format, as described above.

In addition **f001.001_res.dat**, **f001.002_res.dat**, **f001.003_res.dat**, etc (and the ***.vtk** equivalents) will be created for iteration 1,2,3, etc. These files will contain resistivities at the end of these iterations. Only the resistivity and $\log_{10}$ resistivity for each element is stored. The values are output in the same order as in **f001_res.dat**. Note that values are only output for the zone specified by the user.

If you have more than one dataset in **protocol.dat** then the files **f001_res.dat, f002_res.dat, f003_res.dat**, etc. will be created. Similarly, a set of ***.vtk**, ***_sen.da**t and ***_err.dat** files will be output Note that in the files **f***_***.*** (e.g. **f001_res.dat**, **f001_res.vtk**, etc.) a value will be output for each finite element in the selected output region. However, the user may have grouped element values in parameters (see **mesh3d.dat** input) and so, in this case, more than one element will contribute to the overall volume of a parameter.

7. Details of mesh3d.dat

In explaining the input requirements for all files the required real and integer values are given for each input line. Whilst integer values can be used for real terms, the converse is not true.

The mesh consists of a number of node points and finite elements. Each element is defined by its node points (4 for tetrahedra; 6 for triangular prisms). **mesh3d.dat** contains the element nodes for each element and then the set of co-ordinates for each node. For an inverse model each finite element is assigned a parameter number and a zone number. Groups of adjacent finite elements can share a parameter number. Similarly, groups of adjacent parameters can share a zone number. Regularisation is applied between parameters, but is not applied across zone boundaries. This allows the user to enforce a sharp transition in resistivity across planes that are known *a priori*. The inversion output file **f***.vtk** will include the parameter zones, which can be useful for checking the assigned zonation (see unstructured mesh example later).

R3t will used the mesh information to build a matrix of node connections (a list of nodes connected to each node) and a matrix of element connections (a list of elements connected to each element). The former is used to build an index matrix for compressed storage of the *conductance* matrix used to solve the finite element equations; the latter is used to form the roughness matrix for regularisation in the inverse solution. Forming these two connectivity matrices takes some computational effort and the user has the option to avoid this by calculating the connectivity matrices and inserting this information in **mesh3d.dat**. This could be beneficial in the long run if the same mesh is to be used for several inversions. To use this option the advanced-flag is set to 1 in Line 1 (see below).

Line 1 changed in version 2.2

Line 1: (3 Int, 1 Real, 2 Int) numel, numnp, num_dirichlet, datum, npere, advanced-flag
where numel is the number of elements in the mesh; numnp is the number of node points in the mesh, num_dirichlet is the number of Dirichlet (fixed potential) node, npere is the number of nodes in each element (4 for a tetrahedral mesh, 6 for a triangular prism mesh). Normally num_dirichlet will be equal to 1. datum is the elevation at ground level (normally zero) or some nominal datum. The value of datum only affects apparent resistivity calculations and not the inverse solution. advanced-flag is 0 (normal mode) if basic mesh information is to be provided or 1 if the file also contains information about element and node connectivity (see above).

If (job_type (see **R3t.in** file input definitions in the next section) = 0, i.e. a forward solution) then

> Line 2: (1 Int, npere Int) i, (kx(j, i), j = 1,npere)
> where i is the element number and kx(j,i) to kx(npere,i) are the element node numbers of element i.

Else (i.e. for an inverse solution)

> If (advanced-flag =0) then
>
> > Line 2: (1 Int, npere Int, 2 Int) i, (kx(j, i), j = 1,npere), param_elem(i), zone_elem(i)
> > where i is the element number, kx(j,i) to kx(npere,i) are the element node numbers of element i; param_elem(i) is the parameter number of element i (set param_elem(i) to zero if you do not want the resistivity to change from the starting resistivity, defined in Lines 4 or 5 in **R3t.in**; zone_elem(i) is the parameter zone. If all parameters are connected then set zone_elem(i) equal to the same number for all i, otherwise use different numbers for different disconnected region. For N zones, zone_elem(i) should be 1,2,…,N.
>
> Else

Line 2: (1 Int, npere Int, 2 Int, nfaces Int) i, (kx(j, i), j = 1,npere), param_elem(i), zone_elem(i), (connected(j),j=1,nfaces)
where i is the element number, kx(j,i) to kx(npere,i) are the element node numbers of element i; param_elem(i) is the parameter number of element i (set param_elem(i) to zero if you do not want the resistivity to change from the starting resistivity, defined in Lines 4 or 5 in **R3t.in**; zone_elem(i) is the parameter zone. If all parameters are connected then set zone_elem(i) equal to the same number for all i, otherwise use different numbers for different disconnected region. For N zones, zone_elem(i) should be 1,2,…,N. connected(1) to connected(nfaces) is a list of elements connected to element i (nfaces = 4 for a tetrahedral mesh or 5 for a triangular prism mesh). Note that if there are less than nfaces elements connected (e.g. for a boundary element) then add zeros (there must be nfaces entries per element).

End if

End if

Repeat line 2 for all numel elements.

If (advanced-flag =0) then

Line 3: (Int, 3 Real) i, x(i), y(i), z(i)
where i is the node number; x(i), y(i) and z(i) are the node coordinates of node i (in m).

Else

Line 3: (Int, 3 Real, 60 Int) i, x(i), y(i), z(i), (connected(j),j=1,60)
where i is the node number; x(i), y(i) and z(i) are the node coordinates of node i (in m); connected(1) to connected(60) is a list of nodes connected to node i. The list should be in ascending order. There must be 60 entries – use 0 as an entry to pad non-existent nodes.

End if

Repeat line 3 for all numnp node points.

Line 4: (Int) dirichlet_node
where dirichlet_node is the node number of a Dirichlet node. Normally only one Dirichlet node is set. Note: avoid setting dirichlet_node to a node number that is used as an electrode site. Ideally dirichlet_node is a node far away from the electrodes.

Repeat line 4 for all num_dirichlet nodes.

Line 5 is new to version 1.8

If (job_type (see **R3t.in** file) = 1, i.e. an inverse solution) then

If (the number of parameter zones is > 1) then

Line 5: (Int, Real) i, alpha_scale(i)
where i is the zone number (see Line 2) and alpha_scale(i) is the scalar for smoothing in that zone.

Else

alpha_scale(1) is set to 1.0 (no input is needed to define this)

End if

End if

END OF INPUT FOR **mesh3d.dat**

8. **Details of R3t.in**

Line1: (Char*80) header
where header is a title of up to 80 characters

Line 2 has been changed from version 2.01

Line 2: (3 Int) job_type, singularity_type, res_matrix
where job_type is 0 for forward solution only or 1 for inverse solution; singularity_type is 0 if you do not want to use singularity removal in the forward model calculations or 1 if singularity removal is applied. **NOTE**: singularity removal will increase the forward model accuracy significantly but in order for this to be applied (i) the ground surface must be flat and at z=0; (ii) the problem must be an infinite half space. Without such constraints the analytical solution for a homogenous problem cannot be computed and this is necessary for the singularity removal. If either of the two conditions do not apply then singularity_type must be 0. res_matrix is 1 (normal mode) if a 'sensitivity' matrix is required for the converged solution. This matrix is not the Jacobian but is the diagonal of [$\mathbf{J}^T \mathbf{W}^T \mathbf{W} \mathbf{J}$] which gives an idea of the mesh sensitivity (see equation 5.20 of Binley and Kemna, 2005). One value is stored for each finite element in the mesh. High values indicate high sensitivity, low values indicate poor sensitivity. Plot on a log scale. Set res_matrix to 2 if the true resolution matrix is computed for a converged solution and the diagonal is stored (see equation 5.18 of Binley and Kemna, 2005); note that this calculation is more time consuming than the 'sensitivity matrix' option. Set res_matrix to 3 if the sensitivity map is required and an output of the jacobian matrix and roughness matrix.

Line 3: (Int) num_regions_flag
where num_regions_flag is zero if you wish to read in a file containing the starting resistivity model (for an inversion) or the forward model resistivity distribution. Set num_regions_flag to any other number for a uniform start condition in inverse mode.

If (num_regions_flag = 0) then read the following

> Line 4: (Character file_name)
> where file_name is the name (maximum 20 characters) of the file containing the starting model. Make sure that there are no spaces before the filename and no characters in the line after the filename. The file must contain just the resistivities for all elements in the mesh and these must be in element number order (as output in **f001_res.dat**, for example). Four values for each element are read: x, y, z, resistivity. The x,y,z values are not used and are designated so that an output from ***R3t*** in the **f001_res.dat** format can be used. The values should be separated by spaces or commas (tabs are not recommended – if you use this format then replace tabs with spaces). The file can contain more than four columns of numbers but only the first four in each row are read for each element. **NOTE**: if you output an inverse solution from a previous run and use the reduced region (see Lines 10 to 12) then you cannot use this file for a forward model run since there will not be the required number of entries. **NOTE**: there should be no blank line(s) between Line 3 and Line 4.

Else

> Line 5: (Real) resis
> where the resistivity resis will be assigned to all elements. The units will be Ohm-m if the measured resistances are in Ohms and the mesh geometry is defined in metres.

End if

Line 6 - 9 changed in version 2.0 to make the input follow the same notation and process as ***R2***.

If (job_type = 1, i.e. inverse mode) then read the following

Line 6: (Int, Real) inverse_type, target_decrease
where inverse_type is: 1 (note that this value is not actually used but has been retained to make the input file more consistent with that for ***R2***); target_decrease is a real number which allows the user to specify the relative reduction of misfit in each iteration. A value of 0.25 will mean that ***R3t*** will aim to drop the misfit by 25% (and no more) of the value at the start of the iteration. This allows a slower progression of the inversion, which can often result in a better convergence. If you set target_decrease to 0.0 (normal operation) then ***R3t*** will try to achieve the maximum reduction in misfit in the iteration.

Line 7: (2 Int) data_type, reg_mode
where data_type is 0 for untransformed data and 1 (recommended) for log-transformed data. **NOTE:** ***R3t*** converts the measurements to log values – the user does not need to do this. reg_mode is 0 for normal regularisation; 1 for *background regularisation* (see equation (13)); 2 for *difference regularisation* (see equation (7)). Note that option 2 is not a difference inversion (as per LaBrecque and Yang, 2001) but can be used for this purpose with a modification to the data in **protocol.dat** as shown in equation (11).

If (reg_mode = 0 or 2) then read the following

> Line 8: (Real, 2 Int, Real) tolerance, max_iterations, error_mod, alpha_aniso
> where tolerance is desired misfit (usually 1.0); max_iterations is the maximum number of iterations; error_mod is 0 if you wish to preserve the data weights, 2 (recommended) if you wish the inversion to update the weights as the inversion progresses based on how good a fit each data point makes - this is a routine based on Morelli and LaBrecque (1996) and sometimes referred to as "robust invesion". Note that no weights will be increased. alpha_aniso is the smoothing anisotropy: a value greater than 1 will lead to more smoothing in the horizontal than the vertical. A value less than 1 will lead to exaggerated vertical smoothing. **NOTE** smoothing anisotropy is currently not configured for a tetrahedral mesh.

Else read the following

> Line 8: (Real, 2 Int, 2 Real) tolerance, max_iterations, error_mod, alpha_aniso, alpha_s
> where tolerance,, max_iterations, error_mod, alpha_aniso are desired above and alpha_s is an additional penalty factor applied to the starting resistivity model (see equation (13)). If alpha_s is 1.0 then the regularisation applies the same weight to smoothing the model as to constraining to the background model. A smaller (no zero) value of alpha_s will retain some constraint to the background model.

End if

Line 9: (4 Real) a_wgt, b_wgt, rho_min, rho_max
where a_wgt and b_wgt are error model parameters are error model parameters describing the standard deviation of measurements following:

std(R) = sqrt((a_wgt*a_wgt) + (b_wgt*b_wgt) * (R*R))

where R is the resistance measured (LaBrecque et al., 1996 equation 14). Note that it is the inverse of the standard deviation that is used as weight in the diagonal of the weight matrix **W**. a_wgt is effectively an offset error in the same units as the resistance data and b_wgt is effectively the relative error. If both are set to zero then the **protocol.dat** file must contain individual weights. It is advisable to estimate a_wgt and b_wgt from error checks in the field data (ideally from reciprocal measurements - not measures of repeatability). Typically for surface data a_wgt will be about 0.01 Ohms and b_wgt will be about 0.02 (roughly equivalent to 2% error). Note that if you select data_type =1 in Line 7 then although the resistance data are transformed into log apparent conductivities the a_wgt and b_wgt parameters should still

reflect the variance of the resistance; rho_min and rho_max are the minimum and maximum observed apparent resistivity to be used for inversion (use large extremes if you want all data to be used). If data are ignored by ***R3t*** because of the apparent resistivity limits then these will be reported individually in **R3t.out**. NOTE: that the apparent resistivity calculations assume that you have set the ground surface to z=0, that the ground surface is flat and infinite boundaries exist. Note also that you can select to include individual errors for each measurement in the data input file **protocol.dat** – to do this a_wgt and b_wgt should be set to 0.0 – **protocol.dat** will then require an additional column (see later).

Line 10: (2 Real) z_min, z_max
where z_min and z_max define the minimum and maximum vertical co-ordinates of the volume to be output.

Line 11: (Int) num_xy_poly
where num_xy_poly is the number of x,y co-ordinates that define a polyline bounding the output volume. If num_xy_poly is set to zero then no bounding is done in the x-y plane. The co-ordinates of the bounding polyline follow in the next line. Note: the first and last pair of co-ordinates must be identical (to complete the polyline). So, for example, if you define a bounding square in x,y then you must have 5 co-ordinates on the polyline. The polyline must be defined as a series of co-ordinates in sequence, although the order can be clockwise or anti-clockwise (see examples later).

Line 12: (2 Real) x_poly(1), y_poly(1)
where x_poly(1), y_poly(1) are the co-ordinates of the first point on the polyline.

Repeat line 12 for all num_xy_poly co-ordinates.

End if

Line 13: (Int) num_electrodes
where num_electrodes is number of electrodes.

Line 14: (3 Int) j,k, node
where j is the "string" number of electrode; k is "electrode" number in that "string" and node is the node number in the finite element mesh. The "string" label is sometimes a useful secondary label, e.g. for multiple lines of electrodes.

Repeat Line 14 for all num_electrodes

END OF INPUT FOR **R3t.in**

9. Details of protocol.dat

protocol.dat contains the measurement schedule (and data for inverse if selected). NOTE: **R3t** reads resistance data not apparent resistivity data. If your instrument outputs apparent resistivity you should convert it back to a transfer resistance (measured voltage divided by injected current). Note also that the polarity should be included – transfer resistances can be negative and positive. ***R3t*** (in forward model mode) will output modelled transfer resistances and apparent resistivities to **R3t_forward.dat**.

Line 1: (Int) num_ind_meas
where num_ind_meas is number of measurements to follow in the file.

If (job_type = 1) then

If (a_wgt = 0 AND b_wgt = 0) then

Line 2: (9 Int, 2 Real) j, bh(1,k), elec(1,k), bh(2,k), elec(2,k), bh(3,k), elec(3,k), bh(4,k), elec(4,k), resis, resis_error
where j is not used (but usually is used as a measurement number); bh(1,k) and elec(1,k) is the "string" and electrode number for the P+ electrode; bh(2,k) and elec(2,k) is the "string" and electrode number for the P- electrode; bh(3,k) and elec(3,k) is the "string" and electrode number for the C+ electrode; bh(4,k) and elec(4,k) is the "string" and electrode number for the C- electrode; resis is the measured resistance (in Ω)with error resis_error (in Ω).

Repeat Line 2 for all num_ind_meas

Else

Line 3: (9 Int, Real) j, bh(1,k), elec(1,k), bh(2,k), elec(2,k), bh(3,k), elec(3,k), bh(4,k), elec(4,k), resis
where j is not used (but usually is used as a measurement number); bh(1,k) and elec(1,k) is the "string" and electrode number for the P+ electrode; bh(2,k) and elec(2,k) is the "string" and electrode number for the P- electrode; bh(3,k) and elec(3,k) is the "string" and electrode number for the C+ electrode; bh(4,k) and elec(4,k) is the "string" and electrode number for the C- electrode; resis is the measured resistance (in Ω).

Repeat Line 3 for all num_ind_meas.

End if

Else (for forward solution only)

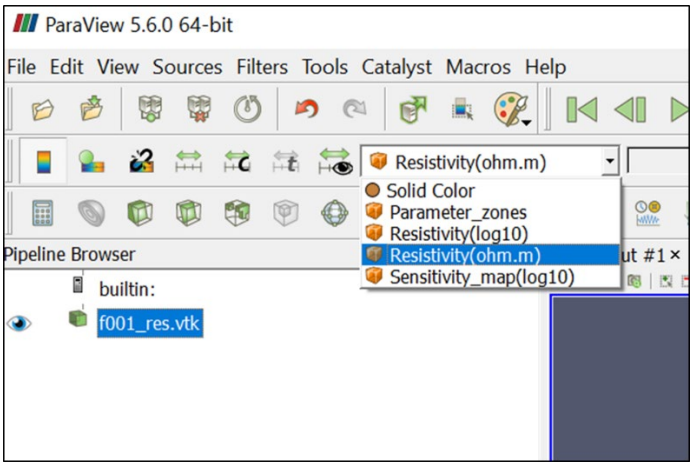
Line 4: (9 Int) j, bh(1,k), elec(1,k), bh(2,k), elec(2,k), bh(3,k), elec(3,k), bh(4,k), elec(4,k)
where j is not used (but usually is used as a measurement number); bh(1,k) and elec(1,k) is the "string" and electrode number for the P+ electrode; bh(2,k) and elec(2,k) is the "string" and electrode number for the P- electrode; bh(3,k) and elec(3,k) is the "string" and electrode number for the C+ electrode; bh(4,k) and elec(4,k) is the "string" and electrode number for the C- electrode.

Repeat Line 4 for all num_ind_meas.

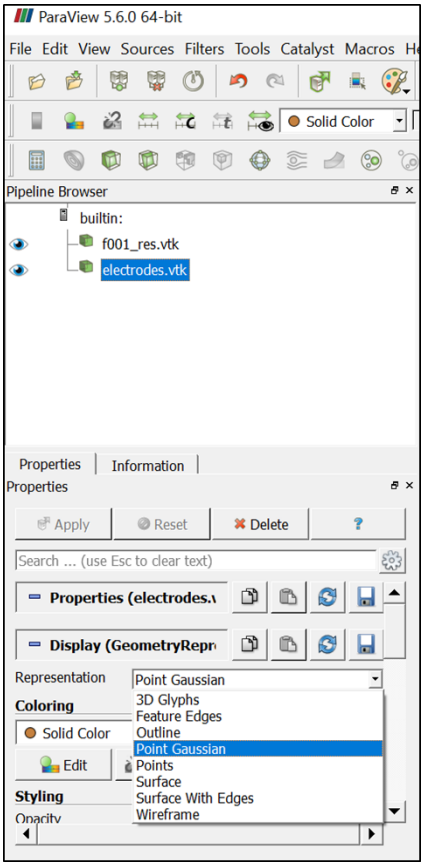
End if
END OF INPUT FOR **protocol.dat**.

10. Using ParaView to view results

The output files for resistivity (e.g. **f001_res.dat**) and sensitivity map (e.g. **f001_sen.dat**) or resolution matrix ((e.g. **f001_rad.dat**) are text files that can be plotted as 3D volumes using various software. ***R3t*** also outputs these results in vtk format, which allows visualisation using ***ParaView*** (which can be downloaded from http://www.paraview.org/download/). The output file **f001_res.vtk** can be opened directly with **ParaView**; all the user needs to do is select "apply" once the file is opened and a 3D image of resistivity will be shown. From the menu bar at the top of the ***Paraview*** screen (see figure below) the user can select "Resistivity(ohm.m)" , "Resistivity(log10)" or (if convergence is achieved) "Sensitivity_map(log10)" or "resolution(log10)" (depending on which option has been selected by the user). It is also possible to image the "Parameter_zones" (see zone_elem in **mesh3d.dat**), which can be useful if zonation of regularisation is applied.



The electrode co-ordinates are also stored in vtk format so that they can be plotted with the image from the inversion. To display the electrodes the user must first open the **electrodes.vtk** file in ***ParaView*** you select the Display (Geometry representation)" with "Representation"" as "3D Glyph". Under "Glyph Parameters" select "Sphere" for "Glyph type". Then select Apply. Recent versions of ***ParaView*** on Windows appear to have a bug in showing Glyph Parameters – occasionally not all electrodes are shown. If you experience this then you can simply select the **electrodes.vtk** file in ***ParaView*** and select the "Representation" as "Point Gaussian" (see figure to the right).



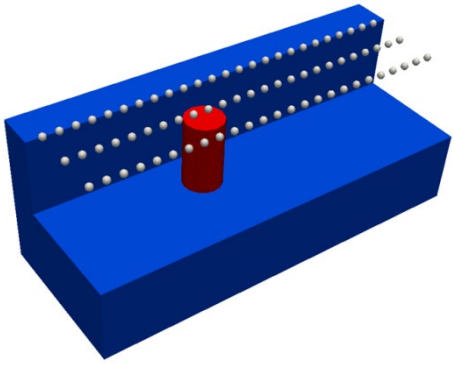
11. Examples

The folder **Models** contains example input files for the computation of forward and inverse problems. There are two subfolders: **Unstructured mesh example** and **Structured mesh example**.

Unstructured mesh example

The unstructured mesh example is based around the problem shown in Figure 5. The problem consists of a 5m diameter, 10m high, cylindrical object (resistivity: 500Ωm) placed with its top at a depth of 1m below ground level, embedded in a uniform 100Ωm background. Three lines of electrodes are shown in Figure 5, each with 25 electrodes, 2m spaced. The three lines are spaced 5m apart. The entire mesh is shown in Figure 6. Note that we are using this problem for the purpose of illustrating the content of input files for ***R3t*** – we are not optimising the survey for this particular problem. In fact, if the intention was to resolve the resistive cylinder in the subsurface then a much better electrode configuration would be used.

Figure 5. Resistivity model for cylinder problem. The cylinder (red) is 500Ωm the background (blue) is 100Ωm. Part of the background region has been cut out to show the cylinder. Note that this is a subregion of the mesh shown in Figure 6.



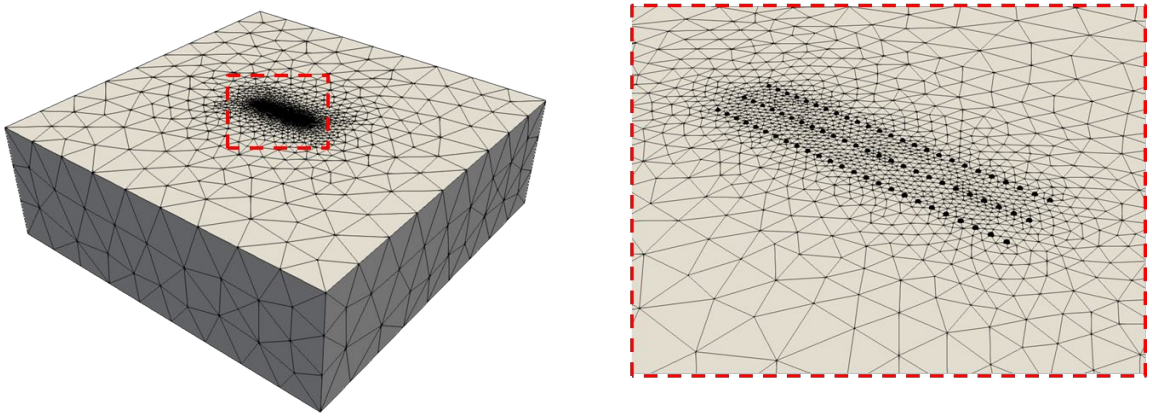


Figure 6. Tetrahedral mesh with region extending some distance to represent infinite boundaries. The figure on the right shows a zoomed section, illustrating higher density of nodes near to electrodes (black symbols).

The forward model is included in folder **\Models\Unstructured mesh example\Forward**. Within this folder a subfolder **\Models\Unstructured mesh example\Forward\gmsh files** contains the geometry file **mesh.geo** used to create the mesh, and the resultant ***Gmsh*** mesh file **mesh.msh**. This mesh file was then converted to **mesh3d.dat** format and the file **resistivity.dat** was created to contain the resistivity for each element. Note that the first three columns of **resistivity.dat** contain zeros, this is because they are not needed on input, ***R3t*** just reads the fourth column – the resistivities for each element in the mesh.

As can be seen from **R3t.in**, we have designated the three electrode lines as three "strings", each with 25 electrodes, thus electrode "1 25" is string 1, electrode 25; electrode "2 3" is string 2, electrode 3, etc.

A measurement sequence was created and stored in **protocol.dat**. ***R3t*** was then run and the file **R3t_forward.dat**, containing the calculated forward model, was created.

This forward model file is shown in the Excel spreadsheet **Forward_data (noisy).xlsx.** The first sheet ("forward") in this spreadsheet contains the output from **R3t_forward.dat** with an extra column in which the geometric factor has been calculated. Next, the calculated dataset was filtered to remove measurements with very large geometric factors (in this case > 5000m or less than -5000m). Although this isn't really necessary for this synthetic problem, it is useful to illustrate for normal practice. The filtered measurements are shown in sheet "forward_lowK".

Next, noise was added to the synthetic data. This is shown in the sheet "forward_with_noise". 2% Gaussian noise was added to resistances. These are then stored in the file **forward_noisy.dat.** This file then serves as input for an inversion.

The folder **\Models\Unstructured mesh example\Inverse(1)** contains input files for inversion of the noisy data created above. Note that a different mesh has been used for the inversion to avoid biasing with a cylindrical shape embedded in the mesh. The ***Gmsh*** files are included in **\Models\Unstructured mesh example\Inverse(1)\gmsh files**. Because a different mesh has been used for inversion, the node numbers for electrodes will be different to those used in the forward model, as can be seen from the entries in **R3t.in**.

For the inversion we use a starting model of 100 Ωm throughout the entire mesh. We set error parameters a_wgt and b_wgt to be 0.0 and 0.02, respectively (since these are the correct statistics of the error added earlier).

In **R3t.in** we have set the output region to be 0m ≤ x ≤ 48m; -5m ≤ y ≤ 5m; -20m ≤ z ≤ 0m. Figure 7 show results from the inversion.

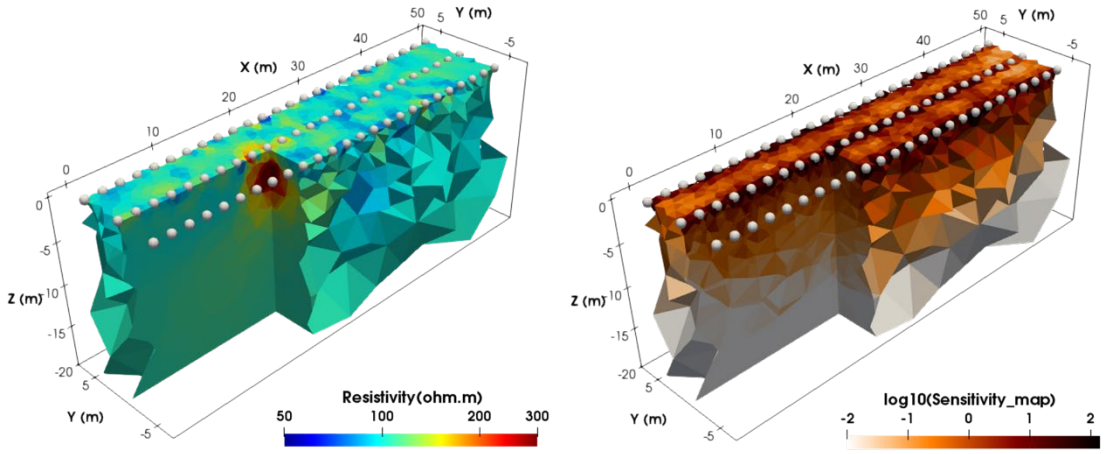


Figure 7. Inversion of resistivity model for unstructured mesh problem Inverse(1). The image on the right is the sensitivity map. The symbols shown mark the electrode positions.

The folder **\Models\Unstructured mesh example\Inverse(2)** contains input files for inversion of the same dataset as above, but in this case we use the alpha_s parameter to try and constrain the inversion by penalising departure from the starting model (in this case a uniform 100Ωm). In this example we set alpha_s to 1.0 – which means that the weighting of penalty for changes from the starting model to the penalty for roughness is 1:1.

Figure 8 shows the result. When compared with the plot in Figure 7, it can be seen that the resistive anomaly is shrunk slightly and the region below it (where there is less data coverage) is much closer to 100Ωm, i.e. the resistive feature is now more a result of the data rather than over-smoothing. Carrying out comparisons like this can be useful as it allows the user to see where in the region the data has an effect.

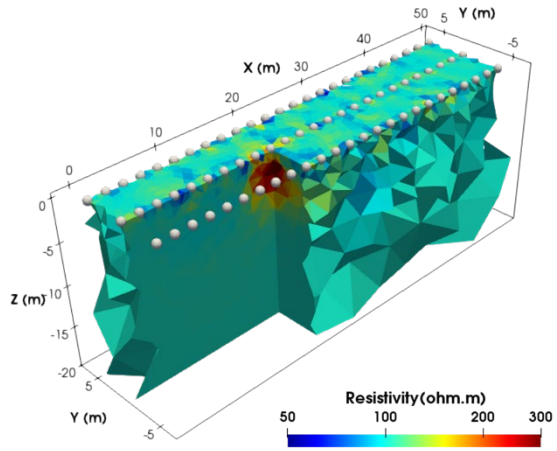


Figure 8. Inversion of resistivity model for unstructured mesh problem Inverse(2).

The folder **\Models\Unstructured mesh example\Inverse(3)** contains input files for inversion of the same dataset as above, but in this case we use the alpha_aniso to try to enhance smoothing (regularisation) in the vertical direction. The settings using in Inverse(1) were used but alpha_aniso was set to 0.01 (i.e. 100 times more smoothing in the vertical). Figure 9 shows the result – some improvement in the final model is evident.

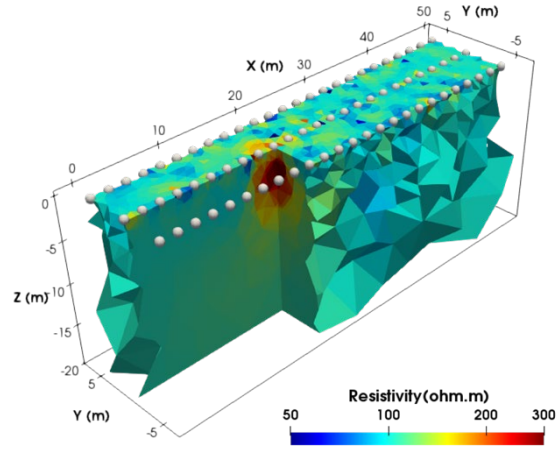


Figure 9. Inversion of resistivity model for unstructured mesh problem Inverse(3).

The folder **\Models\Unstructured mesh example\Inverse(4)** contains input files for inversion of the same dataset as above, but in this case we demonstrate the use of zoning of parameters. For this example the same mesh that was used for the forward problem is adopted (as this geometrically captures the cylinder within the region). Two zones are defined: zone 2 for elements within the cylinder and zone 1 for all other elements, which can be plotted in ***ParaView*** by selecting "parameter_zones" as the variable to plot (Figure 10a). The inversion was then run as normal. The result is shown in Figure 10b. Even though regularisation is applied within each zone, the inversion now captures the resistivity of the cylinder target. Clearly this is a somewhat artificial example but the effectiveness of setting known planes of contrast is demonstrated.

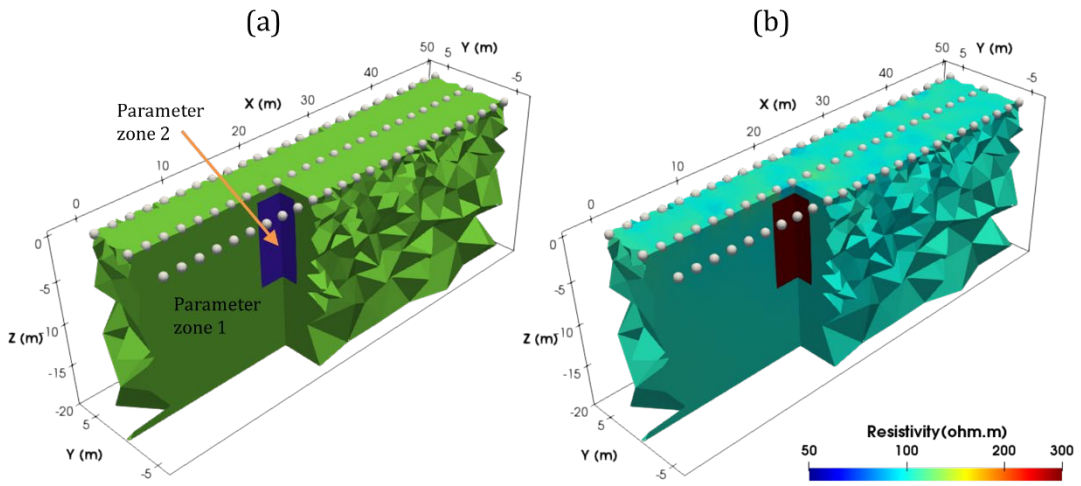


Figure 10. Unstructured mesh problem Inverse(4). (a) Parameter zonation. (b) Inverse model.

Advanced mesh example

The folder "**Models\Advanced mesh example\Inverse** contains the **mesh3d.dat**, **R3t.in** and **protocol.dat** files for an unstructured mesh that has node and element connectivity already computed and stored in **mesh3d.dat**. Note that the only advantage of the advanced mesh setting is the reduced computational time at execution time.

Figure 11 shows the mesh and electrode geometry and the inverted model (illustrated as two 2D vertical slices).

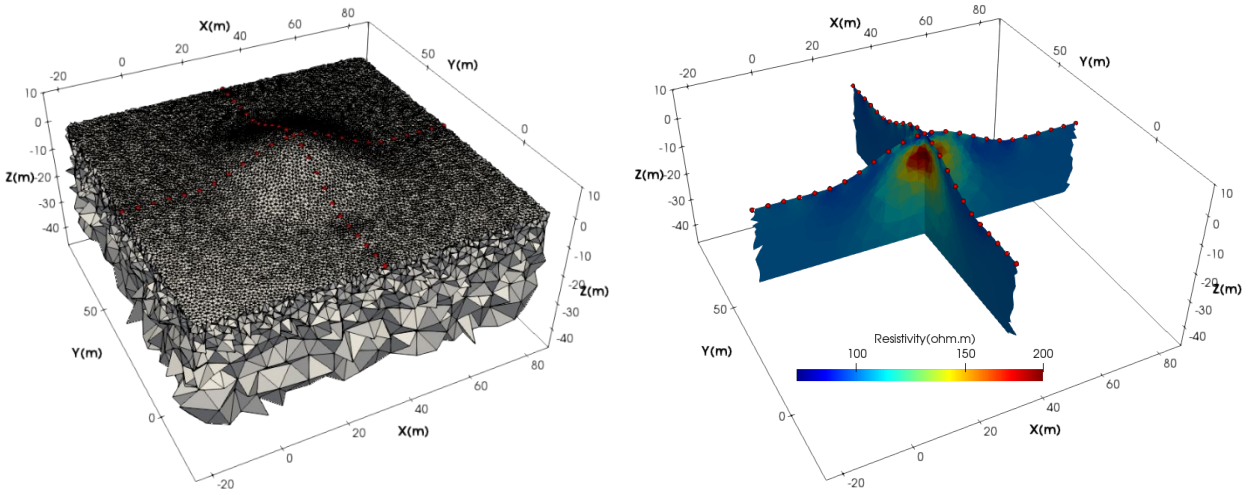


Figure 11. Unstructured mesh problem with advanced mesh setting allowing incorporation of element and node connection in **mesh3d.dat**. *The electrode positions are shown by the red symbols.*

Structured mesh example

We illustrate the use of a structured mesh (based on triangular prisms) for a cylindrical (core) geometry (although note that a structured mesh could be used for a wide range of problems).

A laboratory column was constructed with 8 planes of 12 electrodes, as shown in Figure 13a. the column was filled with water with conductivity 100 µS/cm (equivalent to 100 Ωm). In the upper part of the column a plastic pipe was partially inserted into the column (off-centre) as shown in Figure 13a.

Data were collected in a dipole-dipole configuration in horizontal planes. To model the data a mesh was constructed with 156,672 elements and 83,213 nodes (Figure 13b). The folder "**Models\Structured mesh example\Inverse** contains the **mesh3d.dat**, **R3t.in** and **protocol.dat** files. Figure 13c shows the result from the inversion.

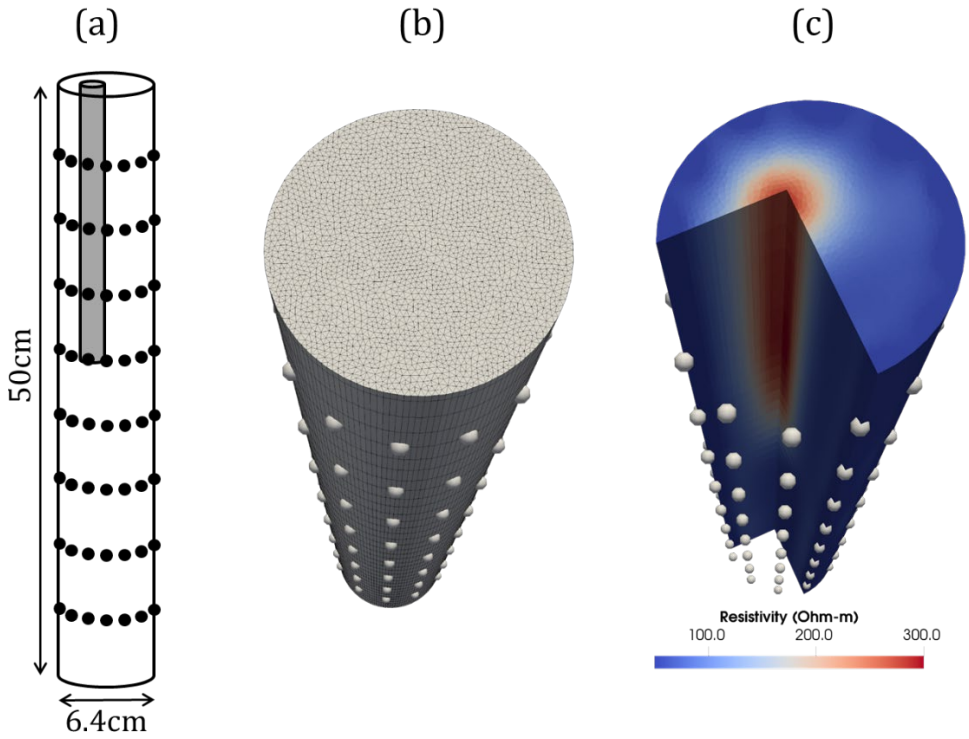


Figure 13. Structured mesh problem. (a) Column setup. (b) Finite element (and parameter) mesh. (b) Inverted resistivity model.

14. **References**

Binley, A., 2015, Tools and Techniques: DC Electrical Methods, In: Treatise on Geophysics, 2nd Edition, G Schubert (Ed.), Elsevier., Vol. 11, 233-259, doi:10.1016/B978-0-444-53802-4.00192-5. (available from the author on request).

Binley, A. and A. Kemna, 2005, Electrical Methods, In: Hydrogeophysics by Rubin and Hubbard (Eds.), 129-156, Springer

Binley, A. and L. Slater, 2020, *Resistivity and Induced Polarization. Theory and Applications to the Near-Surface Earth*, Cambridge University Press (see https://www.cambridge.org/binley)

LaBrecque, D.J. and X. Yang, 2001, Difference Inversion of ERT Data: a Fast Inversion Method for 3-D In Situ Monitoring, Journal of Environmental and Engineering Geophysics, 6(2), 83-89.

LaBrecque, D. J., M. Miletto, W. Daily, A. Ramirez and E. Owen, 1996, The effects of noise on Occam's inversion of resistivity tomography data, Geophysics, 61, 538-548.

Morelli, G. and D.J. LaBrecque, 1996, Advances in ERT modeling, European Journal of Environmental and Engineering Geophysics, 1, 171-186.

---

*If you make use of **R3t** then please contact the author (a.binley@lancaster.ac.uk) so that you can be added to a mailing list for future updates, fixes, etc.*

For more information, including example files contact:

Andrew Binley
Lancaster Environment Centre
Lancaster University
Lancaster LA1 4YQ, UK
Email: a.binley@lancaster.ac.uk

Lancaster University
