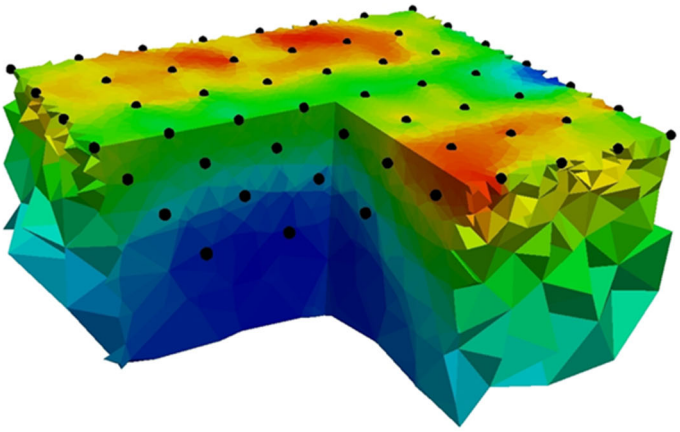
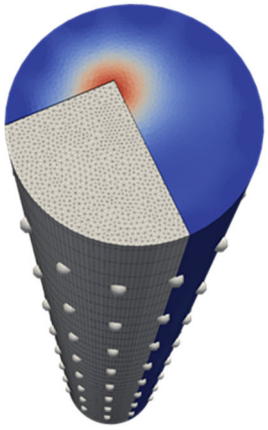
cR3t

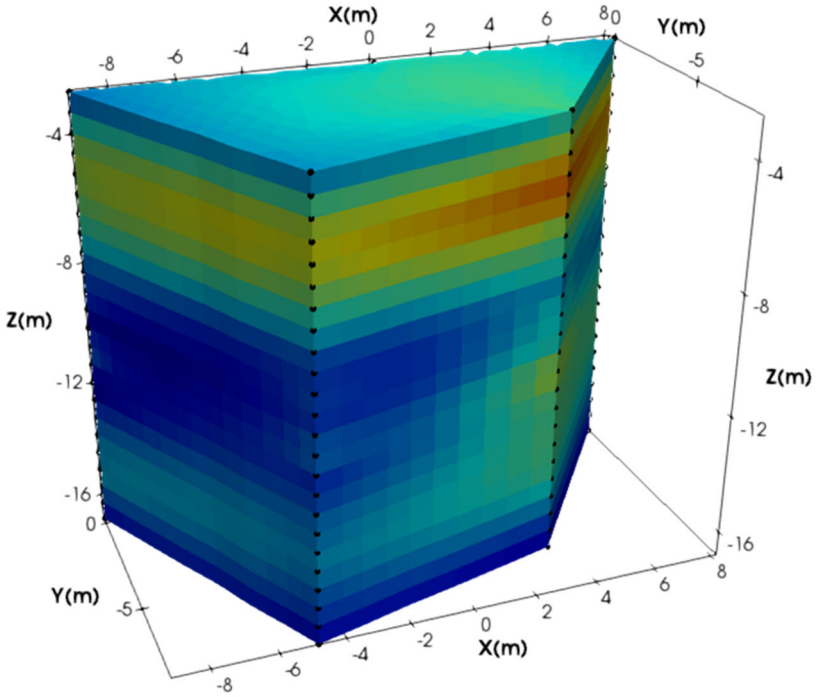
version 1.0

Andrew Binley
Lancaster University
May, 2020

Lancaster University







Contents

		Page
1.	Version history	1
2.	Computer requirements for ***cR3t*** v1.0	2
3.	Introduction to ***cR3t*** v1.0	3
4.	Mesh generation and parameterisation	5
5.	Inverse modelling in ***cR3t*** v1.0	8
6.	Input and output files	10
7.	Details of **mesh.dat**	13
8.	Details of **cR3t.in**	15
9.	Details of **protocol.dat**	18
10.	Using ***ParaView*** to view results	20
11.	Example	21
12.	References	23

1. Version history

Changes from earlier version (0.2b)

Major changes to input file ***cR3t.in*** to be consistent with ***R3t* v2.0**.
Added input file checks.
Apparent resistivity checks on input.
Electrode checking on input.
Added misfit target decrease option.
Output format of **fxxx_err.dat** changed.
Added roughness to main output log.
Output file names changed to be consistent with ***R3t* v2.0**
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***

2. **Computer requirements for c*R3t* v1.0**

In this release two versions have been compiled for the Windows environment. A 64bit version, **cR3t.exe**, is provided in the package. Linux users should be able to run ***cR3t*** with the command "wine R3t.exe" (thanks to Rodolphe Cattin for this tip).

NOTE 1: ***cR3t*** *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* **cR3t.exe** and copy to the shortcut to the working folder.

NOTE 2: *You will be able to run* ***cR3t*** *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* ***cR3t*** *from the Command Prompt (just run CMD from the Start Menu – you may need to move your working directory and run* ***cR3t*** *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).*

cR3t has been designed so that no other commercial software is required for pre- and post-processing. Simple structured meshes can be created directly in ***cR3t***, alternatively more complex geometry can be meshed using freely available codes, such as ***Gmsh*** (http://www. gmsh.info). ***cR3t*** does not produce graphical output but results compatible with the freely available ***ParaView*** (https://www.paraview.org) are produced. Other free codes, such as ***GMT*** (https://github.com/GenericMappingTools/gmt) can be used. Users wishing to use a graphical user interface for meshing, modelling and plotting may be interested in ***ResIPy*** - an open source python GUI for ***cR2*** and sister codes. See https://gitlab.com/hkex/pyr2 which also includes links for standalone executables. More information is also available at https://www.researchgate.net/project/ResIPy-GUI-for-R2-family-codes. See also Boyd et al.(2019) and Blanchy et al.(2020).

3. **Introduction to *cR3t* v2.0**

cR3t 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 ***cR3t*** 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.

cR3t will output calculated parameters (resistivity magnitude and phase angle) 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 ***cR3t*** 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 (2015) and Binley and Kemna (2005). Contact the author for a digital copy of the former.

cR3t is provided for non-commercial use. The program is offered 'as this' without any warranty of any kind. Any users wishing to use ***cR3t*** for commercial applications should contact the author.

It is strongly recommended that the user becomes familiar with inverting DC resistivity data with sister code ***R3t*** before working with ***cR3t***, and inverting 2D IP data with sister code ***cR2***. Many of the concepts about meshing, parameterisation are similar and the documentation for ***R3t*** and ***cR2*** covers examples that will help the user become familiar with ***cR3t***.

In ***cR3t*** complex resistivity is defined by a magnitude and phase angle. The magnitude is equivalent to a DC resistivity since the phase angles are likely to be small. In an inversion, ***cR3t*** also computes the real and imaginary conductivity since these are more useful for analysis of the conduction and polarization of the subsurface. Data for ***cR3t*** are supplied in terms of magnitude and phase angle of the measured impedance. A negative phase angle is a positive IP effect. It is important to understand that the magnitude is a positive number, unlike DC resistance, which can be positive or negative. For example, a surface DC resistivity dipole-dipole survey with four adjacent electrodes, 1,2,3 and 4 in AB-MN configuration 1,2 – 3,4 will give a negative resistance since the geometric factor is negative. Let's say that the value is -1.0Ω. If a complex resistivity survey is carried out with the same configuration then the magnitude would be 1.0Ω and the phase angle (for a non-polarizing subsurface) would not be zero but would be $-\pi$ radians, i.e. -3,141.6 mrad. Preparing data in this way

can be messy and lead to confusion and so measurements with a negative geometric factor it may be easier to express the measurement as the equivalent with a positive geometric factor. The values below are equivalent.

A	B	M	N	Magnitude (Ω)	Phase angle (mrad)
1	2	3	4	1.0	-3,141.6
1	2	4	3	1.0	0.0
2	1	3	4	1.0	0.0

IP data measured in the time domain will give a resistance and chargeability (M). The resistance is easily transformed to a magnitude taking care of the polarity issue above. The chargeability can be transformed to an equivalent phase angle using the method described in Kemna et al. (1997). The conversion is a function of the chargeability sampling and current injection frequency. Typically the phase angle (in mrad) is equivalent to ∼-1.3M (in mV/V). See Mwakanyamale et al.(2012) as an example of such a conversion for application of ***cR2*** (the 2D sister code of ***cR3t***) to time domain IP data. Binley et al. (2016) illustrate the use of ***cR3t*** for a cross-borehole problem.

4. **Mesh generation and parameterisation**

cR3t models the voltage field and determines complex 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 recommended that the user defines 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.

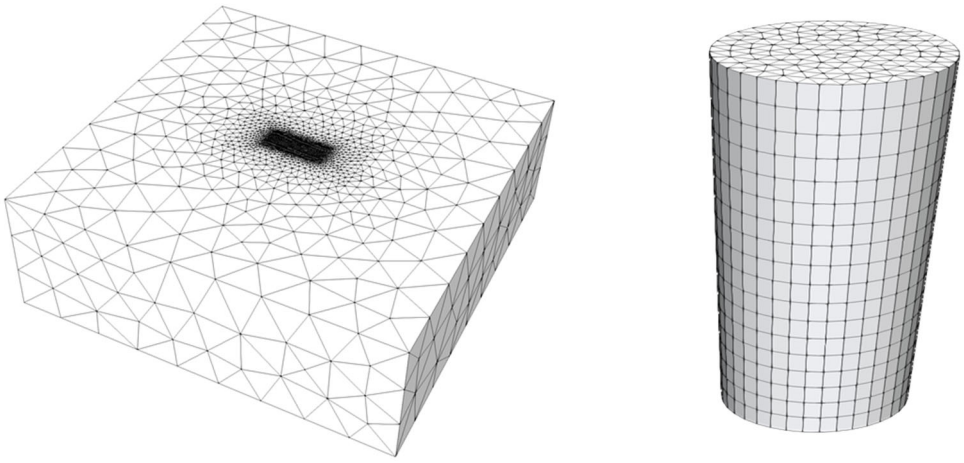


Figure 1. Example tetrahedral and triangular prism meshes.

The resistivity does not vary within each element in ***cR3t***. 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 adjacent 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.

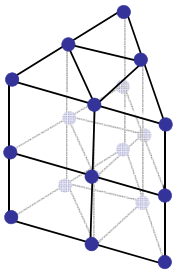


Figure 2: Grouping of triangular prism finite elements to form a single triangular prism parameter cell.

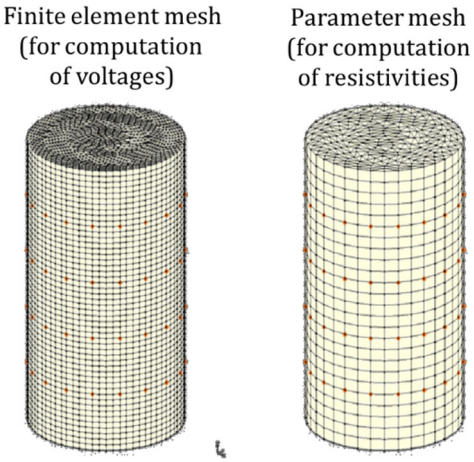


Figure 3. Example finite element and parameter discretisation for a triangular prism mesh in ***cR3t***

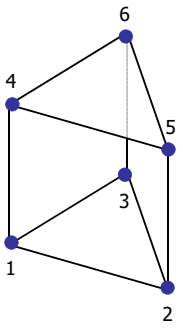


Figure 4: Node numbering in a triangular prism element

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 **cR3t.in**). NOTE: elements assigned a parameter number '0' must be listted at the end of element matrix in **mesh.dat** (see below).

cR3t 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 ***cR3t*** (it also works for 2D meshes in ***R2*** and ***cR2***). This can be found in /Mesh utilities/Jimmy Boyd. An executable (**gmsh2R2msh.exe**) and source code (**gmsh2R2msh.py**) 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 *cR3t* v1.0**

In ***cR3t*** 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 the complex algebra formulation (in ***cR3t***) **J**, **d**, **m**, **f**() are complex (contain real and imaginary components).

In ***cR3t*** the parameters are the logarithm of the complex electrical conductivity in each element (or group of elements that form a parameter block). The data are the logarithm of the transfer impedances supplied by the user (which are provided as a magnitude and phase angle).

Equations (1) and (2) are solved repeatedly until satisfactory convergence is achieved. In ***cR3t*** 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:

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

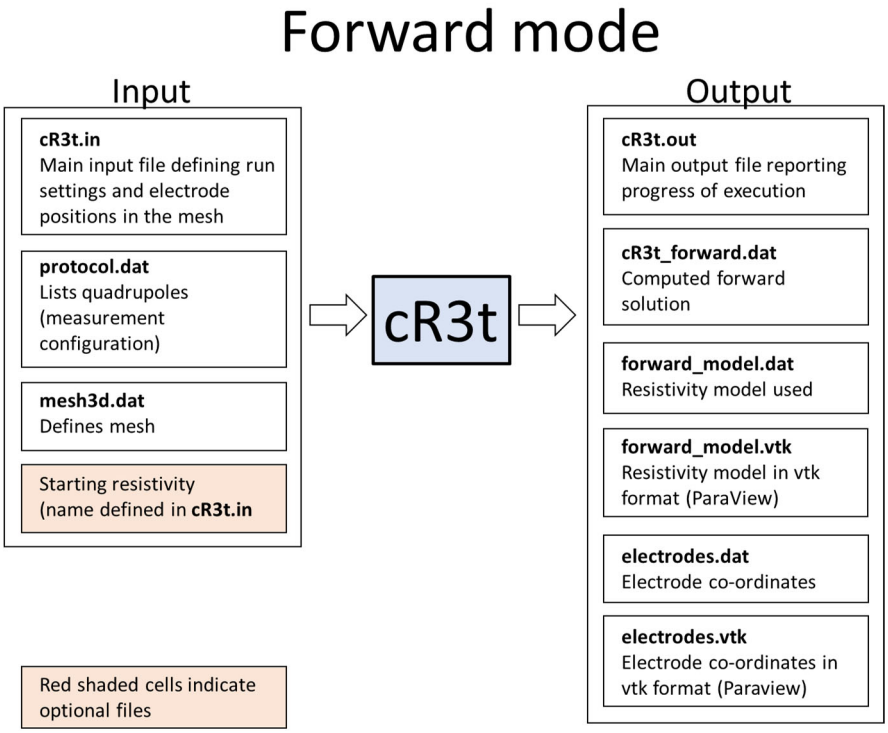
for the largest α, i.e. we wish to obtain the smoothest distribution of resistivity that is consistent with the observed data. ***cR3t*** achieves this through an iterative process in which equation (1) is solved and equation (2) applied. At each iteration α can change, keeping it as large as possible. ***cR3t*** does a line search for α at each Gauss Newton iteration (using up to 10 values of α) The purpose of the line search is to find the value of α that results in the lowest value of Ψ_d for the specific iteration step. ***cR3t*** computes a reasonable starting value for α at the beginning of the process by assessing an equal

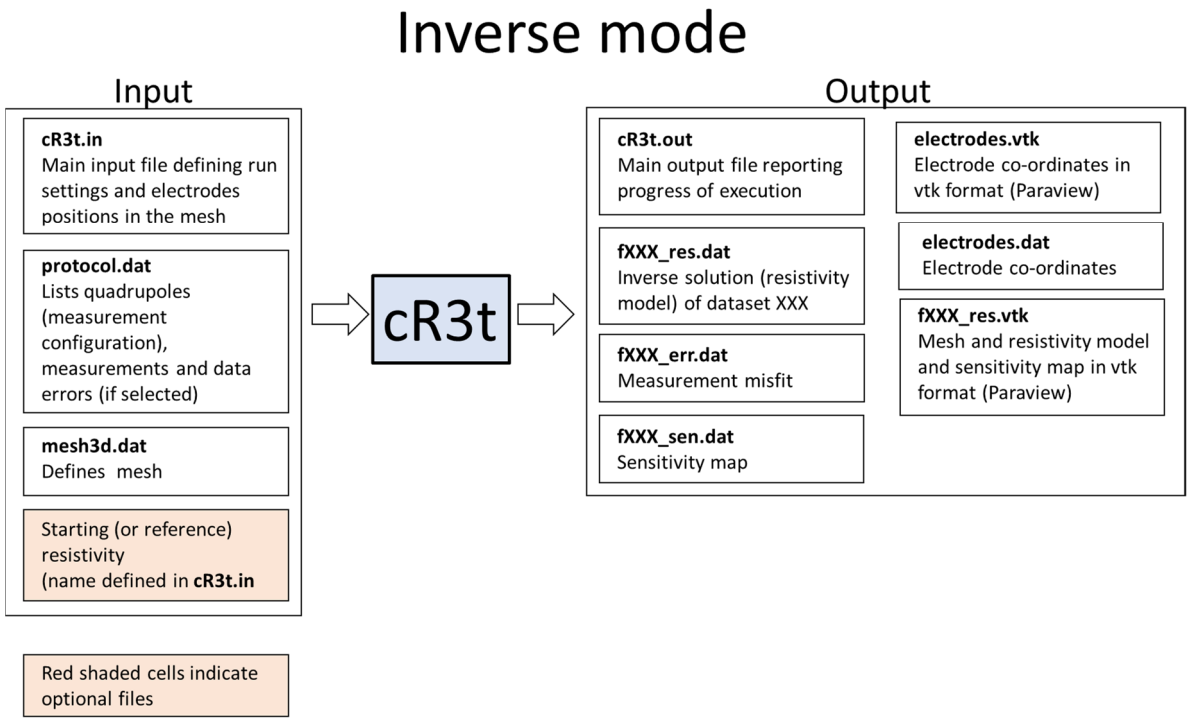
balance of the terms in the brackets of left hand side of equation (1). The iterative process often results in satisfactory convergence within a few iterations. However, this can lead to unsmooth models as the inversion process is attempting to find a solution too quickly. It can be beneficial to slow the process. To do this the user can select the maximum change in misfit during each iteration (using the target_decrease parameter – see **cR3t.in** input). The user is recommended to experiment with either solution strategies.

During each iteration ***cR3t*** will output values of Ψ_d (reported as a root mean square (RMS) error) and Ψ_m (reported as "roughness").

6. Input and output files

cR3t requires at least three data files: **cR3t.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 (magnitude and phase) distribution or if you want to use an inhomogeneous complex resistivity distribution as a starting model or reference model for an inversion.





In *forward* mode ***cR3t*** will output six files:

- **cR3t.out** which will contain main log of execution.

- **cR3t_forward.dat** will contain the forward model for the electrode configuration in **protocol.dat** The format of **cR3t_forward.dat** is the same as **protocol.dat** but with 3 extra columns: the first contains the calculated transfer impedance magnitude (in Ω), the second contains the calculated phase angle of the transfer impedance (in mrad), and the third contains the calculated apparent resistivities (in Ωm) (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 ***cR3t*** 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 eight 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 magnitude (in Ωm); the element resistivity phase angle (in mrad); the element $\log_{10}$ resistivity magnitude (in $\log_{10}$ Ωm); the element $\log_{10}$ real conductivity (in $\log_{10}$ S/m); the element $\log_{10}$ imaginary (quadrature) conductivity (in $\log_{10}$ S/m).

- **forward_model.vtk** will contain the complex resistivity model in vtk format. Resistivity magnitude, phase angle, $\log_{10}$ resistivity magnitude, $\log_{10}$ real conductivity and $\log_{10}$ imaginary conductivity 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 ***cR3t*** will output several files:

- **cR3t.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 eight 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 magnitude (in Ωm); the element resistivity phase angle (in mrad); the element $\log_{10}$ resistivity magnitude; the element $\log_{10}$ real conductivity (in $\log_{10}$ S/m); the element $\log_{10}$ imaginary conductivity (in $\log_{10}$ S/m). Note that values are only output for the region specified by the user. Note also that for any parameter cell if the phase angle computed is positive then $\log_{10}$ imaginary conductivity is undefined and a value of zero will be substituted.

- **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 sensitivity values will be output as 10^{-99}.

- **f001.vtk** will contain the resistivity, $\log_{10}$ resistivity, $\log_{10}$ real conductivity; $\log_{10}$ imaginary conductivity, $\log_{10}$ 'sensitivity' and parameter zones in vtk format. 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. Note also that for any parameter cell if the phase angle computed is positive then $\log_{10}$ imaginary conductivity is undefined and a value of zero will be substituted.

- **f001_err.dat** will contain sixteen 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 and as phase angle. 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 of the electrodes as described above.

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

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 – each element within a parameter block of elements will share the same value of complex resistivity.

7. Details of mesh3d.dat

NOTE: 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***_res.vtk** will include the parameter zones, which can be useful for checking the assigned zonation (see unstructured mesh example later).

Line 1: (3 Int, 1 Real, 1 Int) numel, numnp, num_dirichlet, datum, npere
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) nodes, 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 is only used for apparent resistivity calculations and not affect the inverse solution; it allows the code to compute a correct geometric factor provided the ground surface is flat.

If (job_type (see **cR3t.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)

> 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 **cR3t.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.

End if

Repeat line 2 for all numel elements.

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.

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 1.0

If (job_type (see **cR3t.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 cR3t.in**

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

Line 2: (2 Int) job_type, singularity_type
where job_type is 0 for forward solution only or 1 for inverse solution; singularity_type is 0 (normal mode) 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.

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 complex resistivities for all elements in the mesh and these must be in element number order (as output in **f001_res.dat**, for example). Five values for each element are read: x, y, z, resistivity magnitude, resistivity phase angle. The x,y,z values are not used and are designated so that an output from ***cR3t*** 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 five columns of numbers but only the first five 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 9 to 11) 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: (2 Real) resis, phase
> where the resistivity magnitude resis and resistivity phase angle phase will be assigned to all elements. The units of resis will be Ωm if the measured impedances are in Ω and the mesh geometry is defined in metres. The units of phase should be mrad.

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: 0 (note that this value is not actually used but has been retained to make the input file more consistent with that of sister codes and allow for future upgrades); 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 ***cR3t*** 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 ***cR3t*** will try to achieve the maximum reduction in misfit in the iteration.

Line 7: (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 inversion". 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.

Line 8: (4 Real) min_error, a_wgt, b_wgt, rho_min, rho_max
where min_error is the minimum magnitude error (this is to ensure that very low errors are not assigned and is only used if a_wgt and b_wgt are both zero), a_wgt and b_wgt are error variance model parameters: a_wgt is the relative error of magnitudes; b_wgt is absolute error of phase values (in mrad); rho_min and rho_max are the minimum and maximum observed apparent resistivity magnitude to be used for inversion (use large extremes if you want all data to be used). **NOTE** that if your mesh contains topography, or the surface elevation is not zero, or the left, right and lower extent of the mesh does not represent infinite boundaries then the geometric factor computed in the code will be incorrect and thus any comparison of apparent resistivities against upper and lower limits will be invalid. For such a case you should set rho_min and rho_max to be very low and very high values, e.g. -10e10 and 10e10, respectively. 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. If a_wgt and b_wgt are set to zero then **protocol.dat** must contain errors for the data weights (see next section). 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 0.02 (equivalent to 2% error), b_wgt will be typically 2mrad for good data, but could be much higher.

Line 9: (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 10: (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 11: (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 11 for all num_xy_poly co-ordinates.

End if

Line 12: (Int) num_electrodes
where num_electrodes is number of electrodes.

Line 13: (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 13 for all num_electrodes

END OF INPUT FOR **cR3t.in**

9. Details of protocol.dat

protocol.dat contains the measurement schedule (and data for inverse if selected). NOTE: ***cR3t*** reads impedance data not apparent resistivity data. Data are read as the magnitude of the transfer impedance and the phase angle (negative for a positive IP effect). If your instrument outputs apparent resistivity you should convert it back to a transfer impedance (the magnitude is the measured voltage divided by injected current). Note also that the magnitude should be positive. It is important to understand how to deal with measurements that have a DC resistivity equivalent of a negative transfer resistance – see comments at the end of Section 3. ***cR3t*** (in forward model mode) will output modelled transfer impedances and apparent resistivities to **cR3t_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, 4 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), mag, phase, mag_error, phase_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; measured impedance has magnitude mag (in Ω) and phase angle phase (in mrad), mag_error is absolute error in magnitude (in Ω), phase_error is the phase error (in mrad).

Repeat Line 2 for all num_ind_meas

Else

Line 3: (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), mag, phase
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; measured impedance has magnitude mag (in Ω) and phase angle phase (in mrad).

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.

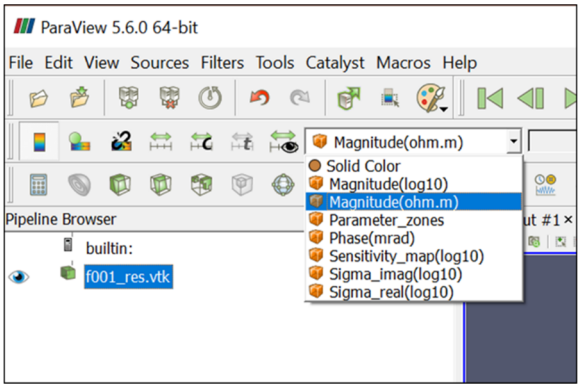
End if

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

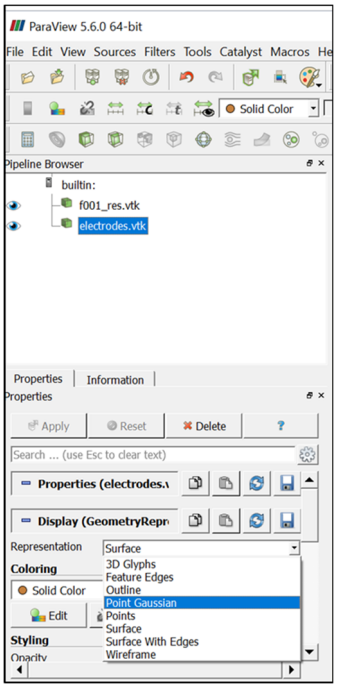
10. Using ParaView to view results

The output files for resistivity (e.g. **f001_res.dat**) and sensitivity map (e.g. **f001_sen.dat**) are text files that can be plotted as 3D volumes using various software. ***cR3t*** 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:

- "magnitude(log10)" ($\log_{10}$ resistivity magnitude in Ωm);
- "magnitude(ohm.m)" (resistivity magnitude in Ωm);
- "phase(mrad)" (resistivity phase angle in mrad);
- "Sigma_real(log10)" ($\log_{10}$ real conductivity in S/m);
- "Sigma_imag(log10)" ($\log_{10}$ imaginary conductivity in S/m);
- "Sensitivity_map(log10)" ($\log_{10}$ sensitivity map);
- "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).



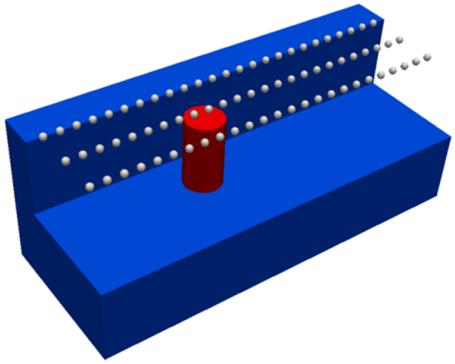
11. Example

The folder **Models** contains example input files for the computation of forward and inverse problems. There is one subfolder: **Unstructured mesh example**. If the user wishes to work with a structured mesh (triangular prisms) the user should consult the example provided with ***R3t.***

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 magnitude: 500Ωm, phase angle -100mrad) placed with its top at a depth of 1m below ground level, embedded in a uniform 100Ωm, -10mrad 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 ***cR3t*** – 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, -100mrad the background (blue) is 100Ωm, -10mrad. 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.



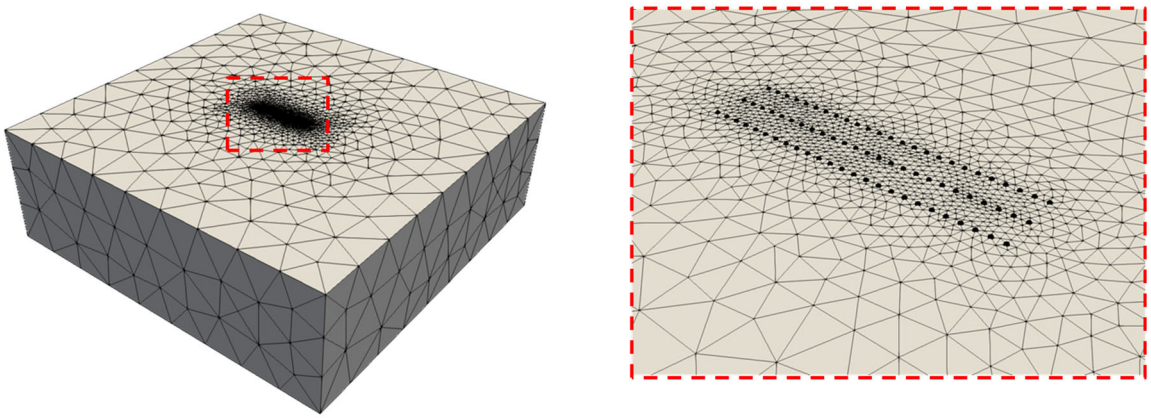


Figure 6. Tetrahedral mesh with region extending some distance to represent infinite boundaries. The figure on the right shows a zoomed section, illustrating higher density of nodes near to electrodes (black symbols).

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, ***cR3t*** just reads the fourth and fifth columns – the resistivity magnitude and phase angle for each element in the mesh.

As can be seen from **cR3t.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**. ***cR3t*** was then run and the file **cR3t_forward.dat**, containing the calculated forward model, was created. Note that the first 69 measurements have a computed transfer impedance with a phase angle of approximately π radians. This is because these measurements for a DC resistivity problem would give a negative transfer resistance (refer to the comments at the end of section 3).

This forward model file is shown in the Excel spreadsheet **Forward_data (noisy).xlsx.** Columns J and K are the computed transfer impedance values; Column L and M contain two sets of random numbers from a U[0,1] distribution. These are then used to perturb the measurements in columns N and O (with a 2% magnitude error and 1 mrad phase error). This then serves as input for an inversion.

The folder **\Models\Unstructured mesh example\Inverse** 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\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 **cR3t.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 0.02 and 1.0, respectively (since these are the correct statistics of the error added earlier).

In **cR3t.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.

Note that the phase angle recovered in the inversion is significantly weaker than the true value in the region of the anomaly. It is important that the user appreciates the challenge of resolving IP features in inverse modelling to avoid over-interpretation of results.

The download files for sister code ***R3t*** contains further examples to allow the user to understand how to zone parameter regions and how to work with structured meshes.

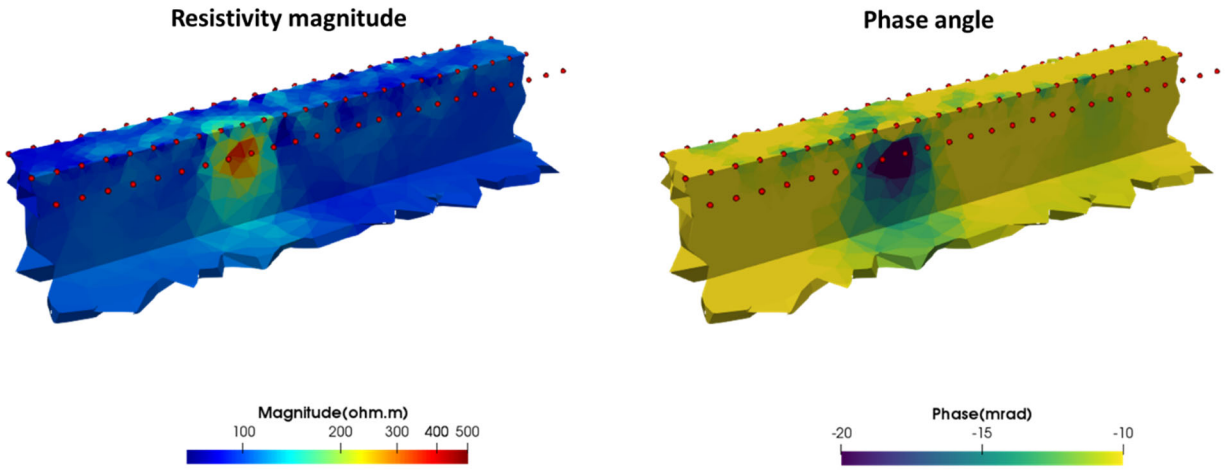


Figure 7. Inversion of resistivity model for unstructured mesh problem Inverse.. The red symbols are the electrodes.

12. **References**

Binley, A., J. Keery, L. Slater, W. Barrash and M. Cardiff, 2016, The hydrogeologic information in cross-borehole complex conductivity data from an unconsolidated conglomeratic sedimentary aquifer, Geophysics, 81(6), E409–E421, DOI: 10.1190/GEO2015-0608.1

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

Blanchy, G., S. Saneiyan, J. Boyd, P. McLachlan and A. Binley, 2020, ResIPy, an intuitive open source software for complex geoelectrical inversion/modeling in 2D space, Computer & Geosciences, 137, DOI: 10.1016/j.cageo.2020.104423

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.

Kemna, A., E. Räkers, and A. Binley, 1997, Application of complex resistivity tomography to field data from a kerosene-contaminated site: Environmental and Engineering Geophysics (EEGS) European Section, 151–154.

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

Mwakanyamale, K., L. Slater, A. Binley and D. Ntarlagiannis, 2012, Lithologic Imaging Using Induced Polarization: Lessons Learned from the Hanford 300 Area, Geophysics, 77, 397-409.

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.

---

*If you make use of **cR3t** 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