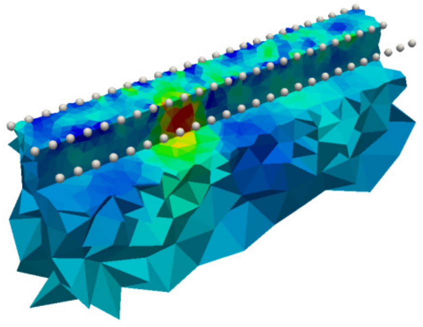
cR3t

version 0.2a

Andrew Binley
Lancaster University
February, 2014

Summary

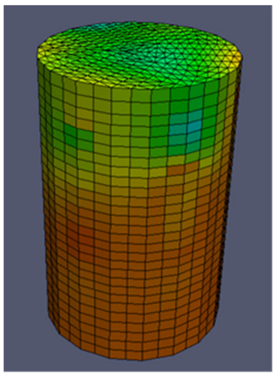
cR3t is a forward/inverse solution for 3D complex resistivity tomography based on a tetrahedral or triangular prism mesh. The inverse solution is based on a regularised objective function combined with weighted least squares (an 'Occams' type solution) as defined in Binley and Kemna (2005) and Kemna et al.(2004).



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. Neumann (no current flow) boundary conditions are applied at the boundary of the mesh and so if a half-space is modelled then the mesh should be extended away from the region of interest to account for 'infinite' boundaries.

cR3t will output calculated parameters (resistivity magnitude and phase angle) for the entire mesh 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 "borehole" number and an "electrode" number (note that the "borehole" index is used simply to help group electrode strings and does not have to refer to a "borehole"; note also that the "borehole" index can be the same for all electrodes if the user wishes not to use this labelling). Data are defined as measured impedance magnitude and phase angle.



The current version will work with any size problem but the user should be careful about not setting up a problem that is too large for the memory (RAM) available. This is particularly relevant for inverse problems. Keep in mind that the size of the Jacobian (sensitivity) matrix, for example, is the number of measurements x the number of parameters (and each value will be stored as 16 bytes, since it is a complex quantity). **cR3t** will output the required memory at the start of execution of the code, but it is advisable to think about memory demands before the mesh is setup.

Changes from earlier version (0.2)

Bug fix in vtk output. No changes to input files.

Changes from earlier version (0.1)

Version 0.1 was based on R3t v1.5 and considered only a triangular prism mesh. Version 0.2 now permits a tetrahedral mesh and is more aligned with R3t v1.7.

Installation and execution

cR3t has been compiled as a standalone program with static libraries. This means that the user does not need to run an installation script – the program (cR3t.exe) is simply copied into a working folder where the input files are. The program can be run by double clicking on cR3t.exe. A black "command window" will open and cR3t will run, displaying some output to that window. When cR3t finishes this window will close. All output files will then be stored in the working folder. If the folder doesn't contain all the input files, or if there are errors in the input files then the command window may open briefly, report an error, then close. If this happens it is beneficial to open a command window (use the Run command in the Windows Start Menu and run CMD). If you don't see the Run command in the Windows Start Menu then you need to install it – simply search for "run command" in "help and support" on the Start Menu and you will see instructions how to do this. When you run CMD using the Run command you will see a black "command window" with the flashing cursor at the end of a line showing the current folder. You need to change this location to your working folder. If you working folder is cR3t in the C drive then you type cd c:\cR3t then enter. You can now type cR3t and enter to run cR3t.

File specifications

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 restart an inverse solution or if you wish to compute a forward model for an inhomogeneous resistivity distribution. ***cR3t.in*** defines the main inversion settings and also the electrode locations within the mesh; ***protocol.dat*** defines the measurement sequence and (for an inversion) the data (note that if you have multiple sets of measurements you can just add them to this file in the same format; ***mesh3d.dat*** defines the finite elements mesh and (for an inversion) the parameter blocking and zoning.

In *inverse* mode **cR3t** will output several files:

cR3t.out which will contain main log of execution.

f001.dat which will contain the resistivity result of the inverse solution. ***f001.dat*** will contain a value for each finite element in the grid (within the zone specified by the user – see later). The file will have six columns: the element centroid x co-ordinate; the element centroid y co-ordinate; the element centroid z co-ordinate; the element resistivity magnitude; the element resistivity phase angle; the element $\log_{10}$ resistivity magnitude.

f001.err will contain 14 columns. In the first column is the normalised data misfit, the second column contains the observed data recorded as an apparent resistivity, the third column contains the equivalent apparent resistivities for the computed model, the fourth column shows the original data weight (i.e. data standard deviation in same units as data), the fifth column is the final data weight, the sixth columns shows a "1" if any weights have been changed during the inversion, otherwise a "0" will appear, the remaining 8 columns show the electrode numbers for the given measurement (each electrode is defined by 2 numbers, as described later). 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).

f001.vtk will contain the resistivity magnitude, resistivity phase angle and $\log_{10}$ resistivity magnitude in vtk format. This can be loaded in *ParaView*, allowing easy visualisation with the mesh outline. Note that values are only output for the zone specific by the user (see later).

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

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 you select the "Glyph" icon; this allows you to plot the electrodes as small spheres (or other objects).

If you have more than one dataset in protocol.dat then the files ***f001.dat, f002.dat, f003.dat***, etc will be created. Similarly, a set of ***.err*** files will be output

Note that in the files ***f***.****** (e.g. ***f001.dat***, ***f001.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.

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

cR3t.out which will contain main log of execution,

cR3t.fwd will contain the forward model for the electrode configuration in ***protocol.dat*** The format of ***cR3t.fwd*** is the same as ***protocol.dat*** but with 3 extra columns: the first contains the calculated impedance magnitude and phase angles along with 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).

Details of mesh3d.dat

cR3t models the complex voltage field and determines complex resistivity parameters based on a 3D mesh of tetrahedral or triangular prism elements. 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 (see Figure 1).

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. Furthermore, the ability to incorporate complex topography permits the full range of geometries in the model.

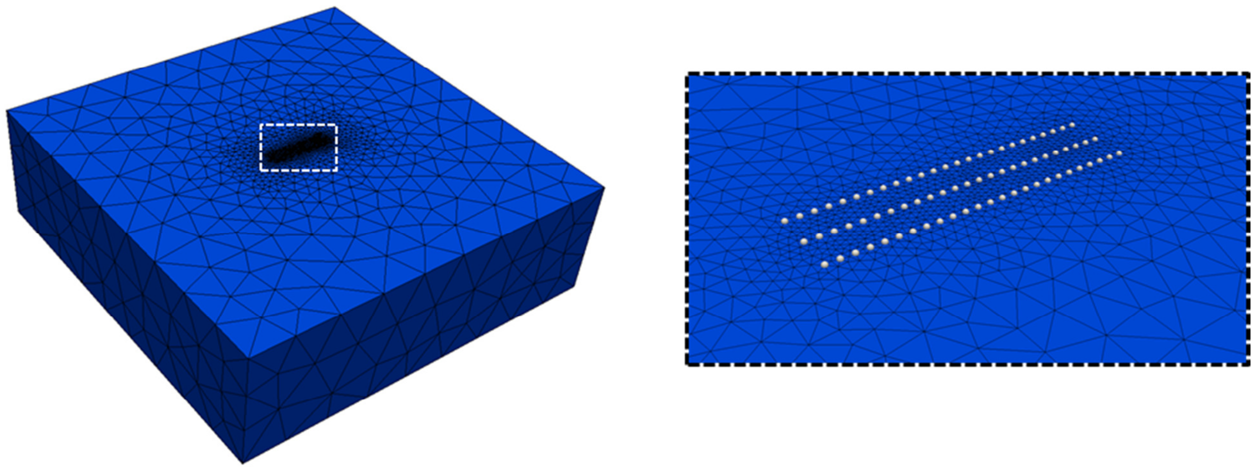


Figure 1. Example tetrahedral mesh with region extending some distance to represent infinite boundaries. The figure on the right shows a zoomed section, illustrating higher density of nodes near to electrodes (grey symbols).

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 elements (see Figure 2). If this is done then each element is assigned to a parameter number which will be common to more than one finite element.

For a triangular prism mesh, each element contains 6 nodes. These nodes should be numbered to 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.

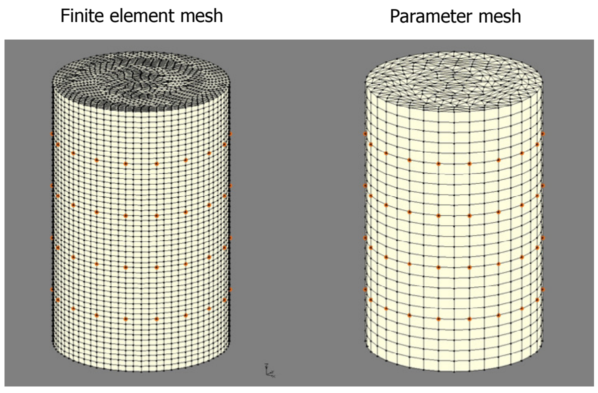


*Figure 2. Example finite element and parameter discretisation for a triangular prism mesh in **cR3t***

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. 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.

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 not change in the inversion from the starting resistivity value remain fixed to the starting resistivity (defined in ***cR3t.in***).

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.geuz.org/gmsh/) 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 (e.g. groundwater code environments like GMS).

The ***cR3t*** download package contains a number of scripts and documents written by other users to help with mesh generation. These are contained in the folder "Mesh generation docs and scripts". These were written for the DC equivalent code ***R3t***, however, the mesh format for ***cR3t*** is identical to that for ***R3t***. These user documents are located in subfolders under the name of the author.

Judy Robinson (Rutgers University) has produced a tutorial for generating structured meshes (for triangular prims) with Gmsh (gmsh R3t tutorial (tri prism).pdf). Judy has also kindly provided Matlab scripts for working with Gmsh and ***R3t***, along with an example.

James Ramm has provided a guide for ***R3t***, which includes some Matlab scripts for generating tetrahedral meshes with topography.

Florian Wagner (University of Potsdam) has produced a Python script for converting gmsh output (tetrahedral or triangular prism meshes) to ***R3t*** and ***cR3t*** format input.

Line 1: (3 Int, 1 Real) 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) 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). This last value only affects apparent resistivity calculations.

If (job_type (see ***cR3t.in*** file) = 0) then

Line 2: (9 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

Line 2: (9 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.

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

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.

Repeat line 4 for all num_dirichlet nodes

END OF INPUT FOR ***mesh3d.dat***

Details of cR3t.in

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

Line 2: (Int) job_type, singularity_type
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. **NOTE**: singularity removal should not be used in v0.2 as this has not been fully tested.

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.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 **R3t** in the ***f001.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 12 to 14) 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 will be Ohm-m (resis) and mrad (phase) if the measured impedances are in Ohms and mrad, and the mesh geometry is defined in metres.

End if

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

Line 6: (Int) inverse_type
where inverse_type is: 0 for normal regularisation or 1 for regularisation of difference between current resistivity and start resistivity in addition to an extra weight applied to the start resistivity.

If (inverse_type = 0) then read the following

Line 7: (2 Real, 2 Int, Real) tolerance, no_improve, max_iterations, error_mod, alpha_aniso
where tolerance is desired misfit (usually 1.0); no_improve is termination criteria such that if during two iterations the misfit doesn't change by no_improve (%) then the inverse solution is stopped; max_iterations is the maximum number of iterations; error_mod is 1 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. 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.

Else read the following

Line 7: (2 Real, 2 Int, 2 Real) tolerance, no_improve, max_iterations, error_mod, alpha_aniso, alpha_s
where tolerance, no_improve, max_iterations, error_mod, alpha_aniso are described above and alpha_s is an additional penalty factor applied to the starting resistivity model. 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 8: (Real, Int) cginv_tolerance, cginv_maxits
where cginv_tolerance is the tolerance for the conjugate gradient solution of the inverse problem (typically 0.0001) and cginv_maxits is the maximum number of iterations for the conjugate gradient solution of the inverse problem (this could be set to a high value < number of parameters or could be set low, say 20, to gain an approximate solution).

Line 9: (Real, Int) alpha_max, num_alpha_steps
The regularisation (or smoothing) parameter, alpha, is optimised each iteration by carrying out a line search. alpha_max is maximum starting value of reqularised scalar and num_alpha_steps (usually 10) is the number of alpha values used each iteration to search for the optimum alpha. Set alpha_max to a large number (say 10e10) if you do not wish to limit the maximum value of alpha.

Line 10: (Real) min_step
where min_step is the minimum step length for attempting to improve solution. This is usually set to 0.001

Line 11: (2 Real) a_wgt, b_wgt, c_wgt, rho_min, rho_max
where a_wgt, b_wgt, c_wgt are error parameters. b_wgt is the relative error of resistivity magnitude; c_wgt is the absolute error in measured phase angle (in mrad). If both are set to zero then the ***protocol.dat*** file must contain individual weights and a_wgt is used to specify the minimum magnitude error. a_wgt is not used if b_wgt and c_wgt are used to specify errors and so its value can be set to any real number. rho_min and rho_max are user specified minimum and maximum apparent resistivities. **NOTE**: the specification of minimum and maximum apparent resistivities will only be applicable if the modelled region is an infinite half space with a flat surface at z=0. If this does not apply then the computation of geometric factors (and hence conversion from data magnitudes to apparent resistivity) will not be correct. For such cases the user should specify, respectively, -10e10 and 10e10 for the minimum and maximum apparent resistivity values.

Line 12: (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 13: (Integer) 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 14: (2 Real) x_poly(1), y_poly(2)
where x_poly(1), y_poly(1) are the co-ordinates of the first point on the polyline.

Repeat line 14 for all num_xy_poly co-ordinates.

End if

Line 15: (Int) num_electrodes
where num_electrodes is number of electrodes.

Line 16: (3 Int) j,k, node
where j is the "borehole" number of electrode; k is "electrode" number in that "borehole" and node is the node number in the finite element mesh. The "borehole" label is sometimes a useful secondary label, e.g. for multiple lines of electrodes.

Repeat Line 16 for all num_electrodes
END OF INPUT FOR ***cR3t.in***

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. If your instrument outputs apparent resistivity you should convert it back to a transfer impedance magnitude (measured voltage divided by injected current). Note also that transfer resistances can be negative and positive but the impedance must be positive since it is an absolute value. If you have a negative impedance magnitude the you can add (or subtract) pi radians from the phase angle, or you can assign the positive value and change the electrode numbers to create the polarity shift. If your IP instrument records chargeability then you need to covert these to phase angles. An approximate to this conversion is phase angle (mrad) = -chargeability (mV/V). Kemna et al.(1997) provide a theoretical solution for such a conversion. The user may also be interested in Mwakanyamale et al.(2012). Note that phase angles are normally negative, positive values can be included although these represent a negative IP effect and the user is advised to check if such an effect indicates measurement problems.

cR3t (in forward model mode) will output modelled transfer impedances and apparent resistivities. The latter are output just for information (and will have no significance if you are not using a half-space geometry with flat topography).

Line 1: (Int) num_ind_meas
where num_ind_meas is number of measurements to follow in 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), resis, phase, resis_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 "borehole" and electrode number for the P+ electrode; bh(2,k) and elec(2,k) is the "borehole" and electrode number for the P- electrode; bh(3,k) and elec(3,k) is the "borehole" and electrode number for the C+ electrode; bh(4,k) and elec(4,k) is the "borehole" and electrode number for the C- electrode; resis is the measured impedance magnitude with error resis_error; phase is the measured impedance phase angle with error phase_error

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, phase
where j is not used (but usually is used as a measurement number); bh(1,k) and elec(1,k) is the "borehole" and electrode number for the P+ electrode; bh(2,k) and elec(2,k) is the "borehole" and electrode number for the P- electrode; bh(3,k) and elec(3,k) is the "borehole" and electrode number for the C+ electrode; bh(4,k) and elec(4,k) is the "borehole" and electrode number for the C- electrode; resis is the measured impedance magnitude; phase is the measured impedance phase angle.

Repeat Line 3 for all num_ind_meas

End if

Else (for forward solution only)

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

End if

If the user wishes to invert other datasets using the same mesh and input settings (as defined in ***cR3t.in***) then additional datasets can be appended to ***protocol.dat*** in the same format as above (with the number of measurements preceding the data, as above).

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

Viewing results files with Paraview

The output files for resistivity (e.g. ***f001.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/paraview/resources/software.html). The output file ***f001.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 the user can select "Magnituide", "Phase (mrad)" or "Magnitude(log10)".

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 "apply" then select the "Glyph" icon. The default display will be arrows, this needs to be changed to something more appropriate, e.g. spheres (as shown in Figure 1).

Bounding region for output

The two examples in Figure 3 show output of a cylindrical problem to illustrate the selection of output zones for the resistivity model. The 6.8cm cylinder is defined in the mesh with $-3.4\text{cm} \leq x \leq 3.4\text{cm}$; $-3.4\text{cm} \leq y \leq 3.4\text{cm}$; $0\text{cm} \leq z \leq 49.5\text{cm}$; $x^2+y^2 \leq 3.4^2\text{cm}^2$. The electrodes are located at planes: z=5.5cm, 11cm, 16, 22, 27.5, 33, 38.5, 44cm.

In Figure 3A the entire mesh is selected for output. In Figure 3B the mesh in the region $0\text{cm} \leq x \leq 3.4\text{cm}$; $-3.4\text{cm} \leq y \leq 3.4\text{cm}$; $20\text{cm} \leq z \leq 40\text{cm}$ is selected for output.

For the case in Figure 3A the input lines 12 to 14 in ***cR3t.in*** are:

```
0.0   50.0 << z_min_z_max
5    << num_xy_poly
-3.4    -3.4    << x_poly(1), y_poly(1)
-3.4     3.4    << x_poly(2), y_poly(2)
 3.4     3.4    << x_poly(3), y_poly(3)
 3.4    -3.4    << x_poly(4), y_poly(4)
-3.4    -3.4    << x_poly(5), y_poly(5)
```

For the case in Figure 3B the input lines 12 to 14 in ***cR3t.in*** are:

```
20.0   40.0 << z_min_z_max
5    << num_xy_poly
0.0    -3.4     << x_poly(1), y_poly(1)
0.0     3.4     << x_poly(2), y_poly(2)
3.4     3.4     << x_poly(3), y_poly(3)
3.4    -3.4     << x_poly(4), y_poly(4)
0.0    -3.4     << x_poly(5), y_poly(5)
```

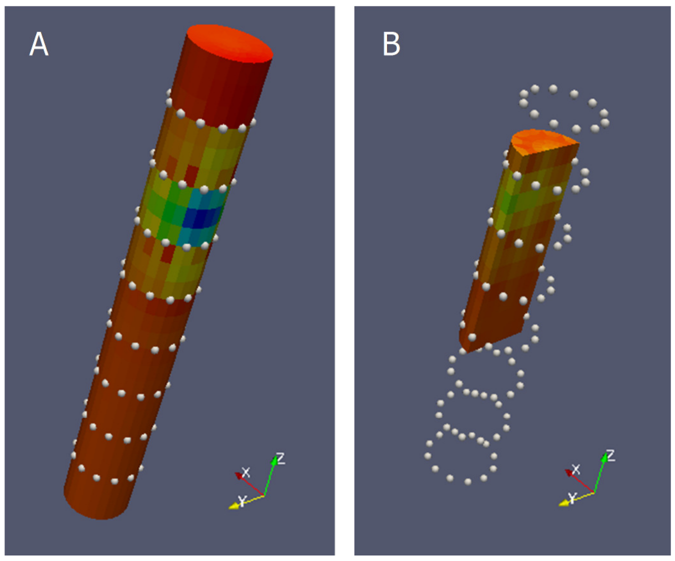


Figure 3. Example output zone selection (see text for details).

Example input files

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

Unstructured example

The unstructured example is based around the problem shown in Figure 4. The problem consists of a 5m diameter, 10m high, cylindrical object (resistivity: 500Ωm, -50mrad) placed with its top at a depth of 1m below ground level, embedded in a uniform 100Ωm, -10mrad background. 3 lines of electrodes are shown in Figure 4, each with 25 electrodes, 2m spaced. The three lines are spaced 5m apart. The entire mesh is shown in Figure 1.

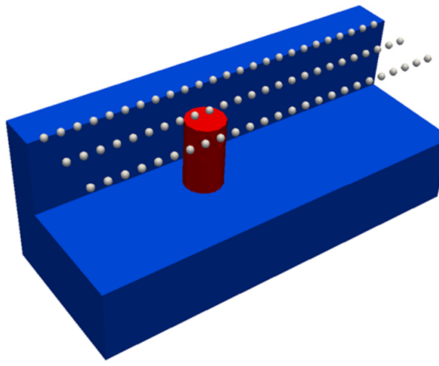


Figure 4. Resistivity model for cylinder problem. The cylinder (red) is 500Ωm, -50mrad, the background (blue) is 100Ωm, -10mrad. Part of the background region has been cut out to show the cylinder. Note that this is a subregion of the mesh shown in Figure 1.

The forward model is included in folder "\Models\Unstructured example\Forward". Within this folder a subfolder "\Models\Unstructured 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 complex resistivity for each element.

A measurement sequence was created and stored in ***protocol.dat***. ***cR3t*** was then run and the file ***cR3t.fwd***, containing the calculate forward model, was created.

Next, the calculated dataset was filtered to remove measurements with very large geometric factors and noise was added to the synthetic data. This is all included in the Excel spreadsheet ***Forward_data (noisy).xlsx.*** 2% Gaussian noise was added to magnitudes and 1mrad Gaussian noise was added to the computed phase angles. These are stored in the file ***forward_noisy.dat.*** This file then serves as input for an inversion.

The folder "\Models\Unstructured 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 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 ***cR3t.in***. Figure 5 show results from the inversion.

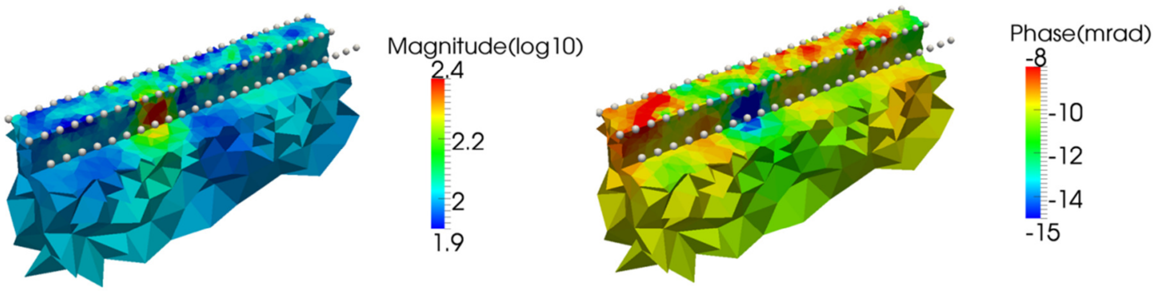


Figure 5. Inversion of resistivity model for cylinder problem.

1. Forward model for cylinder mesh based on triangular prisms

The folder "Forward(1)" contains an example mesh and input files for the computation of a forward model for the resistivity structure shown in Figure 4. For this example the file resis.dat contains the resistivity model. 894 four electrode measurements using the array of 96 electrodes are defined in the file protocol.dat. The computed resistances are shown in the file ***cR3t.fwd***. Note that the apparent resistivities computed should be ignored as the model is not an infinite half space.

References

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

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.

Kemna, A., A. Binley and L. Slater, 2004, Cross-borehole IP imaging for engineering and environmental applications, Geophysics, 69(1), 97-105.

Mwakanyamale, K., L. Slater, A. Binley and D. Ntarlagiannis, 2012, Lithologic imaging using complex conductivity: Lessons learned from the Hanford 300 Area, Geophysics, 77(6), E397–E409

For more information, including example files contact:

Andrew Binley

Lancaster Environment Centre,
Lancaster University,
LANCASTER, LA1 4YQ, UK.

Email: a.binley@lancaster.ac.uk