R3t

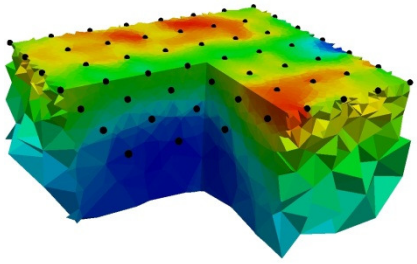
version 1.8

Andrew Binley
Lancaster University
March, 2013

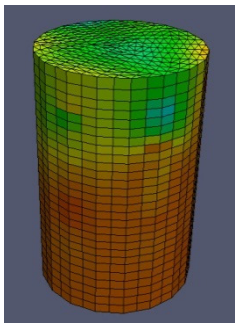
Summary

R3t is a forward/inverse solution for 3D current flow in 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).

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. 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.



R3t will output calculated parameters (resistivity) 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 "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.

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 8 bytes). **R3t** 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. **R3t** runs on a 64bit Intel compatible processor. A 32bit version is available from the author but it will be limited for small problems (due to array limits in 32bit computation).

References

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

Changes from earlier versions

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 (which can be downloaded from http://www.paraview.org/paraview/resources/software.html). See also http://www.vtk.org/Wiki/The_ParaView_Tutorial for a tutorial on ParaView.

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 previous 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 (1.4)
Dynamic memory allocation – no need for fixed array sizes. No change in input.

Changes from earlier version (1.5)
Tetrahedral mesh option 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 (1.7)
Minor output modifications. Some input error checking added.

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***

Summary of inversion approach

R3t determines the spatial distribution of resistivity for a given set of four electrode transfer resistance measurements. The forward solution in **R3t** is based on linear finite (tetrahedral or triangular prism) elements. The resistivity of each of these elements is determined in the inverse solution. As discussed later, the resistivity can be fixed in some elements and elements can be grouped into parameter blocks. **R3t** adopts a Gauss Newton procedure for the inverse solution using a weighted least squares objective function for the data misfit. The solution is stabilised through the use of regularisation – this forces the solution to be either smooth, or constrained by the starting model.

The user is directed to Binley and Kemna (2005) (and references therein) for more information. The basic procedure is as follows.

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:
$\mathbf{J}$ is the Jacobian, such that $J_{i,j}=\partial d_i/\partial m_j$,
$\mathbf{d}$ 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,
$\mathbf{R}$ 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 $\mathbf{m}$.

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) 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.

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.

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:

$$\Psi_{total} = \Psi_m + \alpha\Psi_m, \tag{6}$$

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. In one mode of operation **R3t** does a line search for α at each Gauss Newton iteration. In this mode the user selects the number of steps in this line search. **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 mode or operation 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 so that the smoothest model is found. To do this the user can select not to apply a line search and specify the starting α. If this is done, **R3t** keeps the same α at each iteration, reducing it only if the solution starts to diverge (data misfit increases in subsequent steps) or if the reduction in total objective function (equation (6)) between successive iterations is less than 5%. In either case, α is reduced by 50% for the following iteration. The user is recommended to experiment with either solution strategies.

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:

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

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 LaBreque 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:

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

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:

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

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*.

References

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

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

Installation and execution

R3t 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 (R3t.exe) is simply copied into a working folder where the input files are. The program can be run by double clicking on R3t.exe. A black "command window" will open and **R3t** will run, displaying some output to that window. When **R3t** 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 **R3t** in the C drive then you type cd c:\R3t then enter. You can now type R3t and enter to run **R3t**.

File specifications

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 restart an inverse solution or if you wish to compute a forward model for an inhomogeneous resistivity distribution.

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

R3t.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). The file will have

five columns: the element centroid x co-ordinate, the element centroid y co-ordinate, the element centroid z co-ordinate, the element resistivity and the element $\log_{10}$ resistivity.

f001.sen will contain the diagonal of the matrix [J^T W^T W J] (see Binley and Kemna, 2005) which gives an idea of the mesh sensitivity. You will get a value for all elements. High values indicate high sensitivity to data, low values indicate poor sensitivity. The format is the same as ***f001.dat***. Plot on a log scale (i.e. plot the fifth column). Note that values are only output for the zone specific by the user. Note also 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.err will contain fourteen 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 (ie 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 seventh to fourteenth columns show the electrode numbers. 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, $\log_{10}$ resistivity, $\log_{10}$ '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 specific by the user.

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).

In addition ***f001.001.dat***, ***f001.002.dat***, ***f001.003.dat***, etc will be created for iteration 1,2,3, etc These files will contain resistivities at the end of these iterations. Only the resistivity and log10 resistivity for each element is stored. The values are output in the same order as in ***f001.dat***, which is z,y,x order. Note that values are only output for the zone specific by the user.

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 **R3t** will output two files:

R3t.out which will contain main log of execution,

R3t.fwd will contain the forward model for the electrode configuration in ***protocol.dat*** The format of ***R3t.fwd*** 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).

Mesh generation and parameterisation

R3t models the voltage field and determines 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.

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.

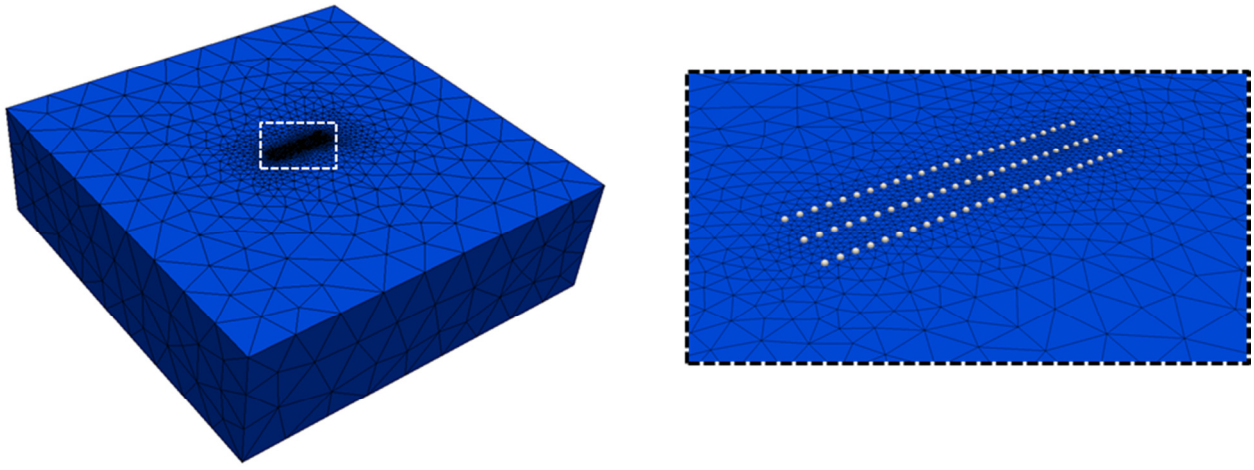


Figure 1. Example tetrahedral mesh with region extending some distance to represent infinite boundaries. The figure on the right shows a zoomed section, illustrating higher density of nodes near to electrodes (grey symbols).

The resistivity does not vary within each element in **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 Figure 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.

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

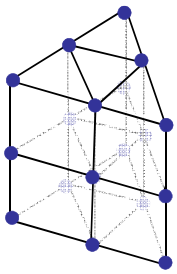
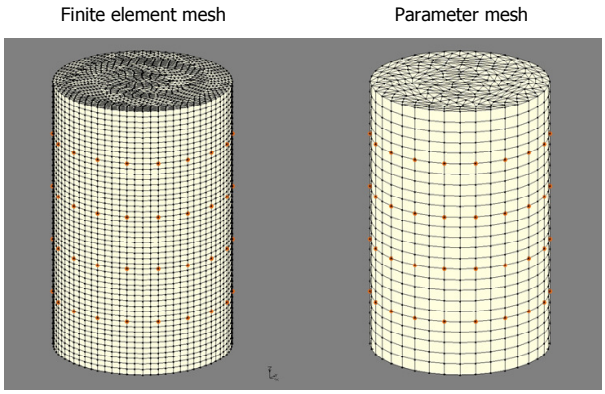


Figure 3. Example finite element and parameter discretisation for a triangular prism mesh in **R3t**



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.

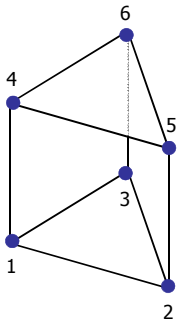


Figure 4: node numbering in a triangular prism element

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 not change in the inversion from the starting resistivity value 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.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 **R3t** 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 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*** format input.

Details of mesh3d.dat

Line 1 changed in version 1.6

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 ***R3t.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 ***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.

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

Line 5 is new to version 1.8

If (job_type (see ***R3t.in*** file) = 1) then

If (the number of 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***

Details of R3t.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.

Line 2 changed in version 1.4

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). 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.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 13 to 15) 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

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

Line 6: (Int) data_type
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.

Line 7 changed in version 1.6

Line 7: (Int) inverse_type
where inverse_type 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 (inverse_type = 0) then read the following

Line 8: (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. **NOTE** smoothing anisotropy is currently not configured for a tetrahedral mesh.

Else read the following

Line 8: (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 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 8 changed in version 1.4

Line 9: (Real, Int) cginv_tolerance, cginv_maxits
where cginv_tolerance is the tolerance (typically 0.0001) for the conjugate gradient solution of the inverse equations 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 50, to gain an approximate solution). A value of 500 for cginv_maxits will achieve a satisfactory solution for most problems, however, setting the value to 50 will lead to a faster execution.

Line 10: (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 requlalised 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. If you wish to specify a minimum value for the starting alpha then set alpha_max to a negative value of the minimum starting value. For example, setting alpha_max to -20 will mean that the starting value of alpha is at least 20. An alternative solution approach is to use one alpha at each of the inverse iterations, i.e. there is no line search for an optimum at each iteration. This can help result in a smoother final model. If num_alpha_steps is set to 1 then alpha_max is the starting value of alpha. This value is then used in subsequent iterations unless the total objective function does not drop by 5% in a subsequent iteration, or if the data misfit increases. If this happens then alpha is reduced by 50% in the following iteration. If this approach is taken then it is advisable to set max_iterations in Line 8 to at least 20.

Line 11: (Real) min_step
where min_step is the minimum step length for attempting to improve solution. This is usually set to 0.001 to 0.01.

Line 12: (2 Real) a_wgt, b_wgt
where a_wgt and b_wgt are error variance model parameters. a_wgt is an offset error in the same units as the resistance data and b_wgt is the relative error. If both are set to zero then the ***protocol.dat*** file must contain individual weights.

Lines 13 to 15 define the region to be output (note that this was new to version 1.3b)

Line 13: (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 14: (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 15: (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 15 for all num_xy_poly co-ordinates.

End if

Line 16: (Int) num_electrodes
where num_electrodes is number of electrodes.

Line 17: (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 17 for all num_electrodes

END OF INPUT FOR ***R3t.in***

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. 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, 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 with error resis_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
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

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 "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***

Output during execution of R3t

Figure 5 shows example output of **R3t** during execution of an inverse solution. Much of this information is stored in ***R3t.out*** for later investigation.

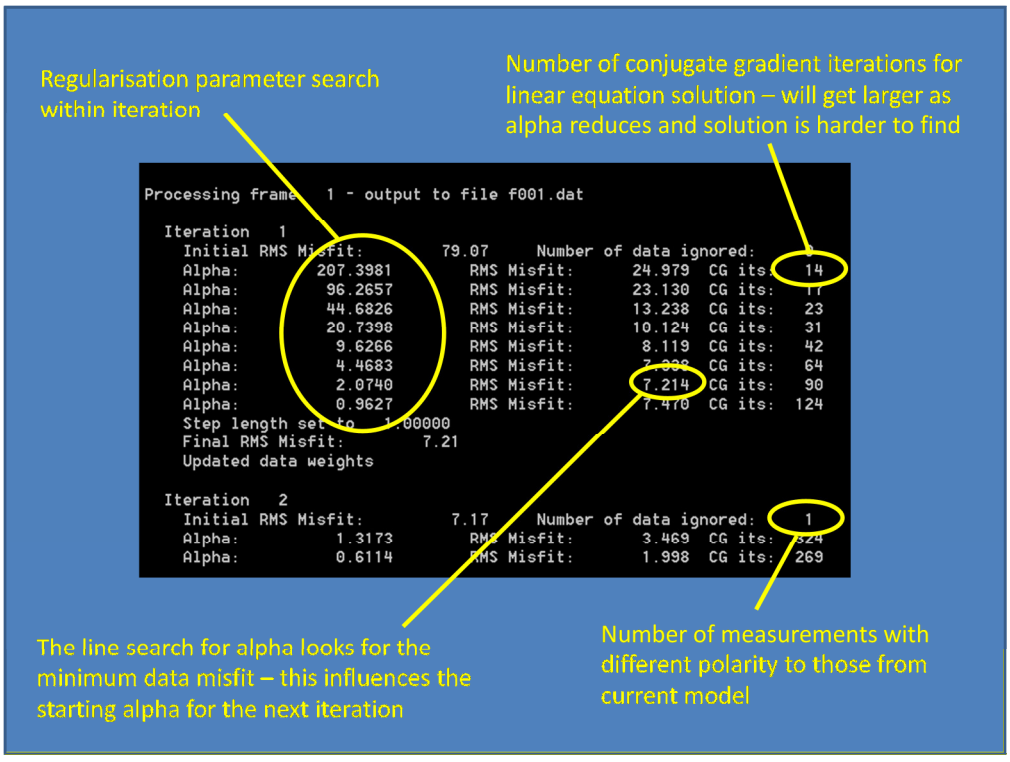


Figure 5. Example output from execution of **R3t** *in inverse mode.*

Viewing results files with Paraview

The output files for resistivity (e.g. ***f001.dat***) and sensitivity map (e.g. ***f001.sen***) 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/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 "Resistivity(log10)" or (if convergence is achieved) "Sensitivity_map(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 6 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 -3.4cm ≤ x ≤ 3.4cm; -3.4cm ≤ y ≤ 3.4cm; 0cm ≤ z ≤ 49.5cm; $x^2+y^2 \le 3.4^2 cm^2$. The electrodes are located at planes: z=5.5cm, 11cm, 16, 22, 27.5, 33, 38.5, 44cm.

In Figure 6A the entire mesh is selected for output. In Figure 6B the mesh in the region 0cm ≤ x ≤ 3.4cm; -3.4cm ≤ y ≤ 3.4cm; 20cm ≤ z ≤ 40cm is selected for output.

For the case in Figure 6A the input lines 13 to 15 in ***R3t.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 6B the input lines 13 to 15 in ***R3t.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)
```

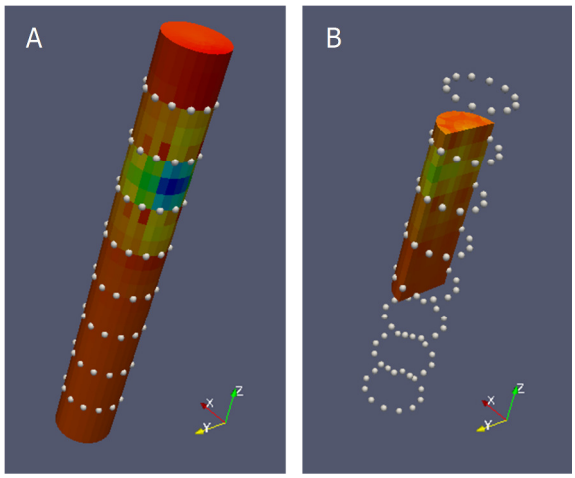


Figure 6. Example output zone selection (see text for details).

Example input files

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

Unstructured mesh example

The unstructured mesh example is based around the problem shown in Figure 7. 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. 3 lines of electrodes are shown in Figure 6, each with 25 electrodes, 2m spaced. The three lines are spaced 5m apart. The entire mesh is shown in Figure 1. 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.

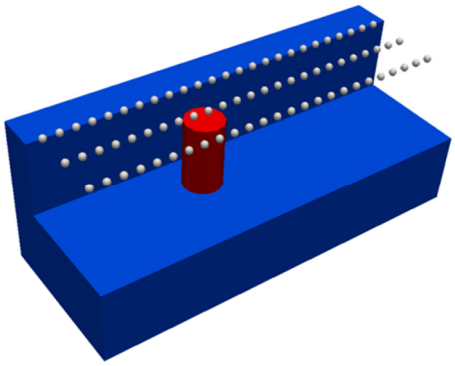


Figure 7. Resistivity model for cylinder problem. The cylinder (red) is 500Ωm the background (blue) is 100Ωm. Part of the background region has been cut out to show the cylinder. Note that this is a subregion of the mesh shown in Figure 1.

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.fwd***, containing the calculate 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.fwd***. Next, the calculated dataset was filtered to remove measurements with very large geometric factors. 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 starting model of 100Ωm throughout the entire mesh. We set error parameters a_wgt and b_wgt to be 0.0 and 0.02 (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 8 show results from the inversion.

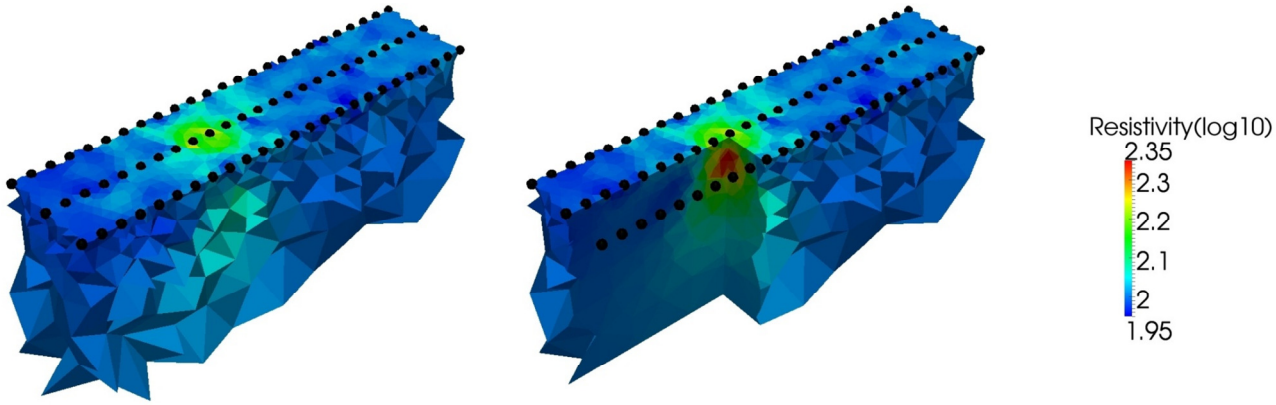


Figure 8. Inversion of resistivity model for unstructured mesh problem Inverse(1). The black symbols are the electrodes.

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 0.5 – which means that the weighting of penalty for changes from the starting model to the penalty for roughness is 1:2.

Figure 9 shows the result. When compared with the right hand side plot in Figure 8, it can be seen that the resistive anomaly is shrunk, 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.

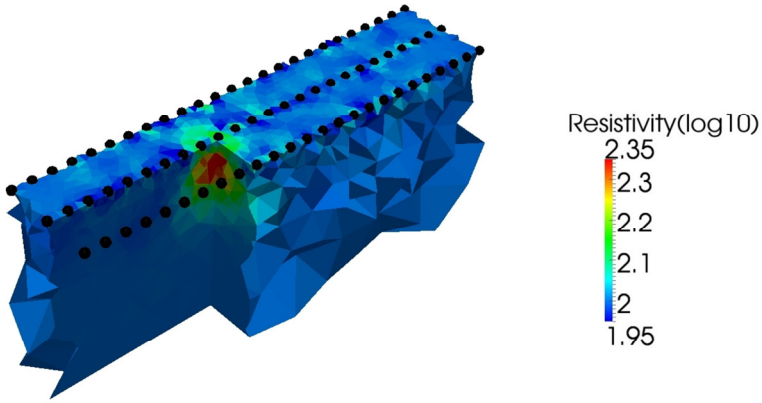


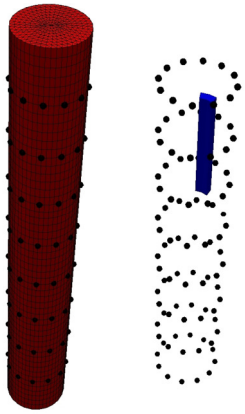
Figure 9. Inversion of resistivity model for unstructured mesh problem Inverse(2).

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). We also use this example to show the way in which parameter cells can be defined as clusters of elements.

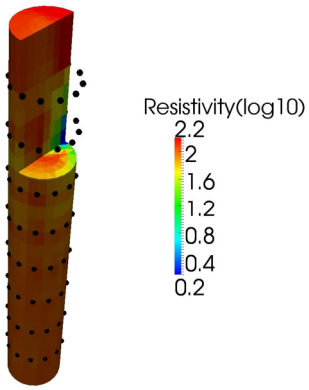
The folder "Models\Structured mesh example\Forward" contains an example mesh and input files for the computation of a forward model for the resistivity structure shown in Figure 10. For this example the file ***resis.dat*** contains the resistivity model. 894 four electrode measurements using the array of 96 electrodes are defined in the file ***protocol.dat***. The computed resistances are shown in the file ***R3t.fwd***. Note that the apparent resistivities computed should be ignored as the model is not an infinite half space.

Figure 10. Resistivity model for structured mesh forward problem. Left hand side figure shows mesh and electrode geometry. Right hand side figure shows 1Ωcm object within 100Ωcm cylinder.



The folder "Models\Structured mesh example\Inverse" contains an example inversion dataset. The output from the forward model above was perturbed with 2% Gaussian noise and inverted. The mesh file (***mesh3d.dat***) shows that the 25920 finite elements are grouped into 3240 parameters (8 elements per parameter). Figure 11 shows the result from the inversion.

Figure 11. Inverse model for structured mesh problem. Symbols show electrode geometry.



For more information, including example files contact: Andrew Binley, Lancaster Environment Centre, Lancaster University, LANCASTER, LA1 4YQ, UK. Email: a.binley@lancaster.ac.uk

---