
R3t

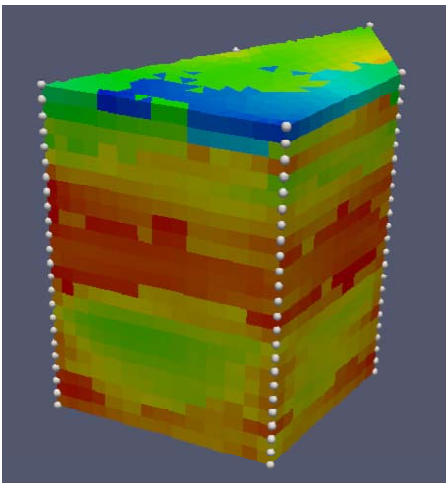
version 1.5a

Andrew Binley
Lancaster University
April, 2011



Summary

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

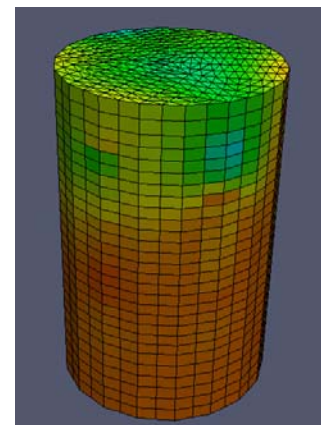


The user must define the mesh for **R3t** as a series of elements, each with 6 nodes. The user must also specify the position of the electrodes within the mesh. The electrodes can be located anywhere in the mesh, provided they fall on node points. Neumann (no current flow) boundary conditions are applied 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 "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).

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.



References

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

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)

Parts of the region can now be set to be fixed during inversion. No change in input unless this option is required (see details for mesh3d.dat).

Installation and execution

R3t is provided as a standalone executable that does not require any installation. The code must be run from a folder containing the input files (detailed below). The user can run the program by double clicking the **R3t** executable. Two executables are provided: R3t_win32.exe and R3t_x64.exe. The former is for Win32 systems and will be limited to problem size (because of the array size limits in 32bit Windows operating systems). R3t_x64.exe will run only on 64bit processors and does not suffer the problem size limitations of a 32bit system.

When running **R3t** will output to a cmd (command) window. All of this output will be echoed to a file R3t.out (see details below) and when **R3t** ends the cmd window will close. If the program terminated properly the last line in R3t.out should be ">> Program ended normally".

The user may find it useful to run the cmd as a separate window and then run **R3t** from there, since any output of warnings or errors will be displayed to a window which will remain open. A useful shortcut for starting the cmd window is to press the "Windows" key on the keyboard and

the "R" key at the same time. Then type "cmd" and a window should appear. You will need to move to the folder where you wish to run **R3t** from – to do this type "cd directory", where "directory" is the full path name of the folder you want to run the program from (e.g. "c:\users\me\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 a 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 x co-ordinate, the element y co-ordinate, the element z co-ordinate, the element resistivity and the element log₁₀ 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 (ie plot the fifth column). Note that values are only output for the zone specific by the user.

f001.err will contain ten 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 column shows a "1" if any weights have been changed during the inversion, otherwise a "0" will appear, the seventh to tenth columns show the electrode numbers.

f001.vtk will contain the resistivity, log₁₀ resistivity and log₁₀ 'sensitivity' 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.

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

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).

Details of mesh3d.dat

R3t models the voltage field and determines resistivity parameters based on a 3D mesh of 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.

The mesh3d.dat file defines the X,Y,Z coordinates of each node point and also the six node points forming each element. Other properties of each element are also defined for an inverse solution. For each element we assign a parameter number and a zone number. The parameter number allows us to set each element as a parameter or to set clusters of adjacent elements as a parameter (the smaller the number of parameters, the faster the computation of the inverse solution). A zone number allows us to create sharper contrasts in resistivity in an inverse solution – smoothing (regularisation) is switched off across boundaries of zone.

The resistivity does not vary within each element. 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 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 1 and 2). If this is done then each element is assigned to a parameter number which will be common to more than one finite element. If the user wishes to have the resistivity of some parameters unchanged during the inversion then the parameter number for these parameters (and hence the element or elements that forms the parameters) should be set to zero. The resistivity for these parameters will then remain unchanged from the starting resistivity. This can be useful for time lapse inversion if the user wishes to force the resistivity to stay fixed outside a region in which changes are expected to take place.

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. This zoning of parameters can also be useful in modelling cross-borehole data where a region representing a borehole can be 'disconnected' (in terms of regularisation) from the surrounding region.

R3t doesn't contain a mesh generator – the user needs his/her 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).

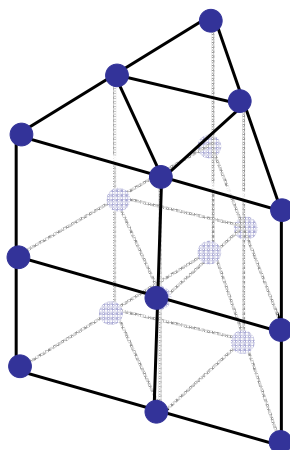


Figure 1. Example finite element and parameter discretisation in R3t. In this example the group of elements may be assigned to one parameter.

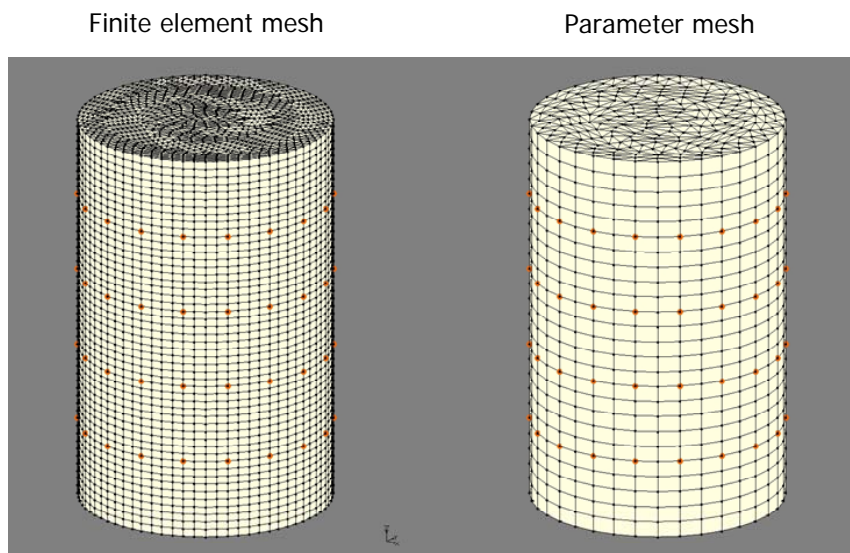


Figure 2. Example finite element and parameter discretisation in R3t

Each element contains 6 nodes. These nodes should be numbered 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.

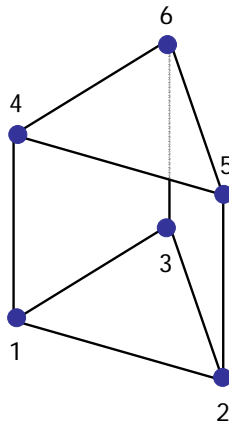


Figure 3. Node numbering of triangular prism finite element

Line 1: (3 Int, 1 Real) `numel`, `numnp`, `num_dirichlet`, `datum`

where `numel` is the number of elements in the mesh; `numnp` is the number of node points in the mesh and `num_dirichlet` is the number of Dirichlet (fixed potential) node. 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(1,i)`, `kx(2,i)`, `kx(3,i)`, `kx(4,i)`, `kx(5,i)`, `kx(6,i)`

where `i` is the element number and `kx(1,i)` to `kx(6,i)` are the element node numbers of element `i` (see Figure 3 for ordering).

Else

Line 2: (9 Int) `i`, `kx(1,i)`, `kx(2,i)`, `kx(3,i)`, `kx(4,i)`, `kx(5,i)`, `kx(6,i)`, `param_elem(i)`, `zone_elem(i)`

where `i` is the element number, `kx(1,i)` to `kx(6,i)` are the element node numbers of element `i`, (see Figure 3 for ordering), `param_elem(i)` is the parameter number of element `i`, `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. If `param_elem(i)` is set to zero then the resistivity for all elements associated with this parameter will stay fixed to the starting resistivity.

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

Repeat line 4 for all `num_dirichlet` nodes

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 had 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: (Int) `inverse_type`

where `inverse_type` is 0 for normal regularisation or 1 for difference regularisation. Note that this does not take into account difference in data but simply regularises on difference between start resistivity and current model.

Line 8: (2 Real, 2 Int) `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.

Line 8 changed in version 1.4

Line 9: (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 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 regularised 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 11: (Real) `min_step`

where `min_step` is the minimum step length for attempting to improve solution. This is usually set to 0.001

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 "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 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 "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 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 "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 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 "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

END OF INPUT FOR *protocol.dat*

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 4).

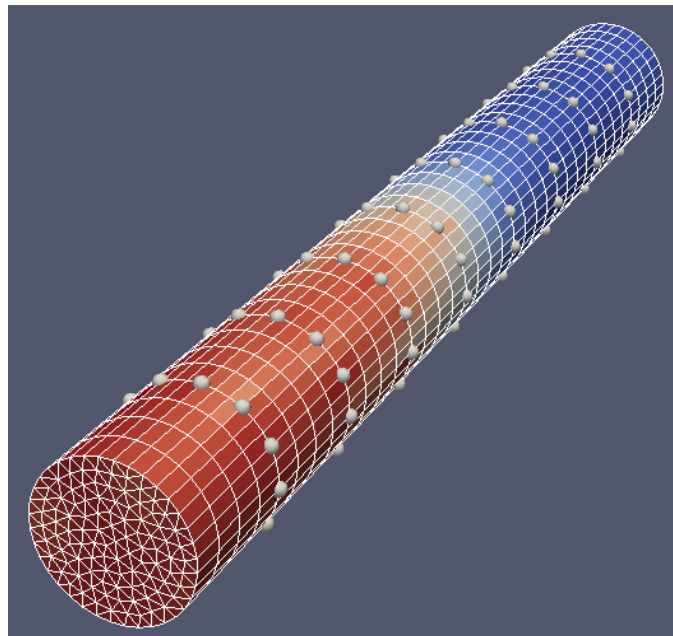


Figure 4. example Paraview display showing electrode locations

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: $z=5.5\text{cm}$, 11cm, 16, 22, 27.5, 33, 38.5, 44cm.

In Figure 5A the entire mesh is selected for output. In Figure 5B 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 5A 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 3B 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)

```

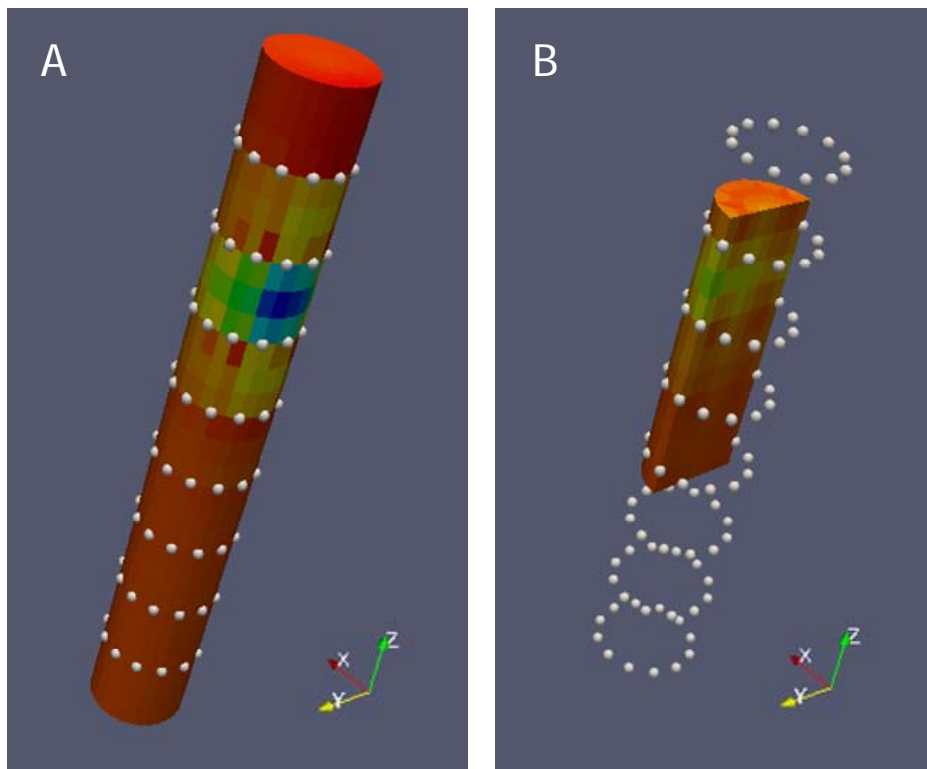


Figure 5. Example output zone selection (see text for details).

Example input files

Forward model

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 6. 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.

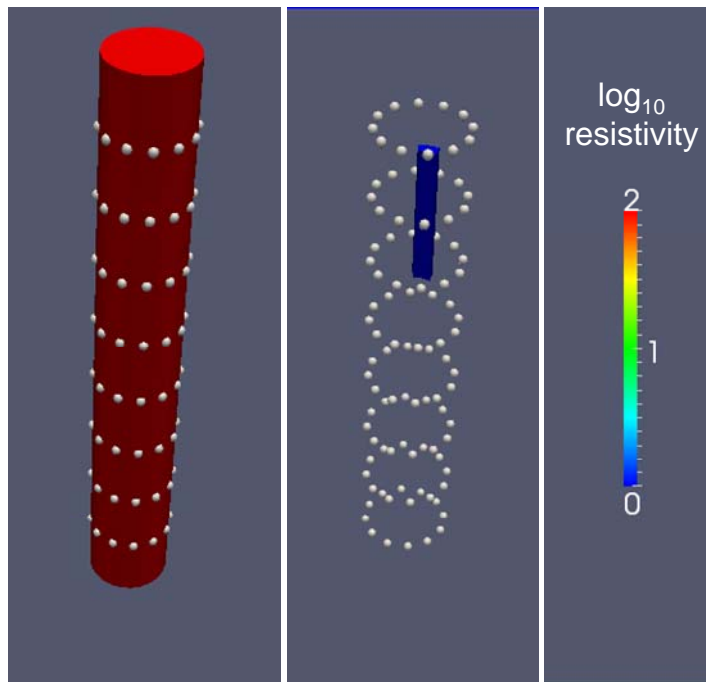


Figure 6. Resistivity model for Forward(1) problem

Inverse model

The folder "Inverse(1)" contains an example inversion dataset. The output from the forward model in Forward(1) is 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 7 shows the result from the inversion alongside the true anomaly.

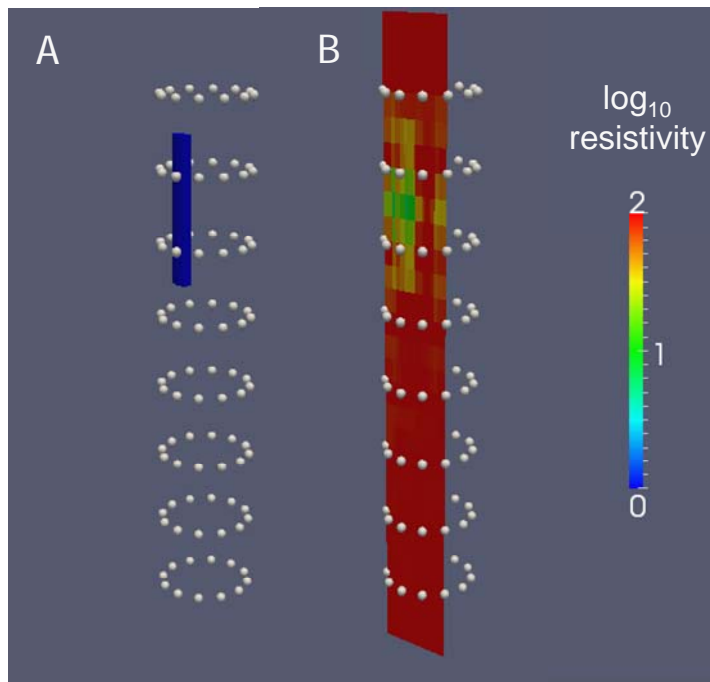


Figure 7. (A) Target and (B) resistivity model from Inverse(1) problem.

For more information, including example files contact:

Andrew Binley

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

Email: a.binley@lancaster.ac.uk