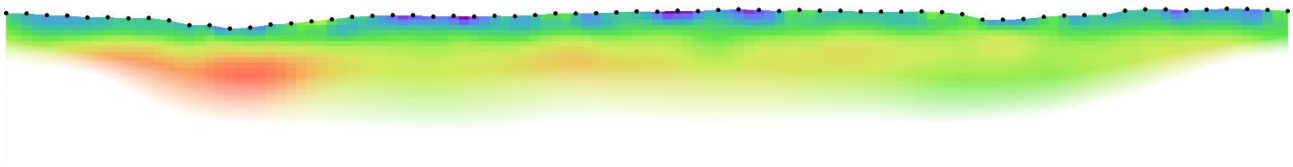
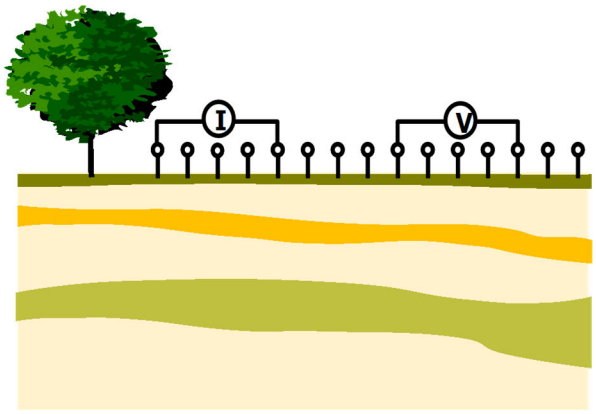
R2

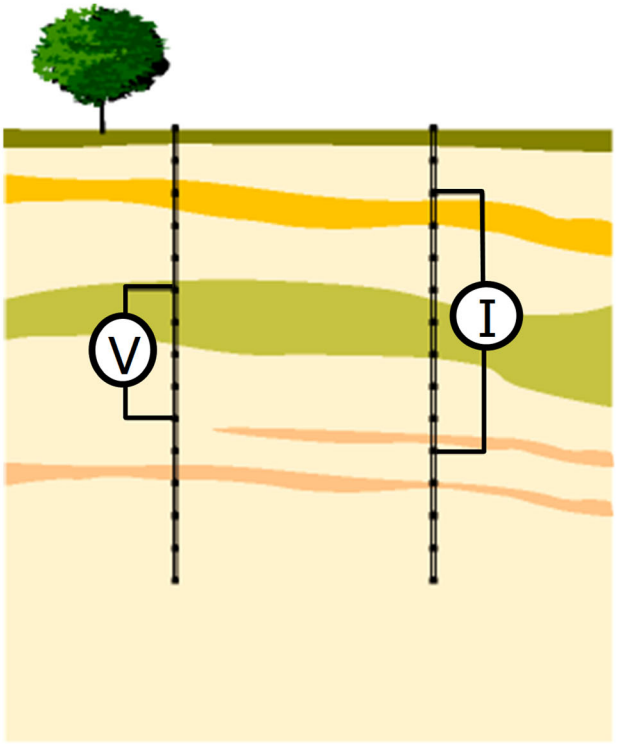
version 4.02 (July 2020)

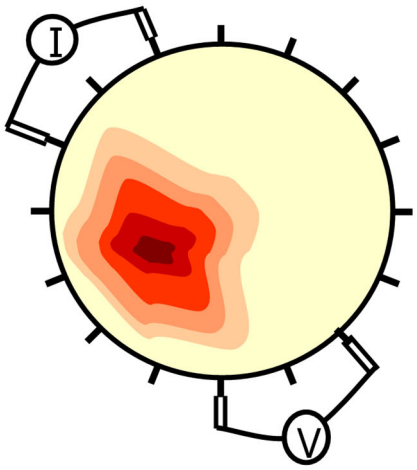
Andrew Binley
Lancaster University
July 2020

Lancaster University









Contents

1. Version history

Changes to *R2* from v4.01
If the starting model satisfies the data then output is generated (previously the files were left empty).
Changes to *R2* from v4.0
The calculation of resolution matrix has been improved. Checks for vtk output of negative values of resolution matrix made. Minor edits to output and documentation.
Changes to *R2* from v3.3
General quadrilateral mesh input now an option. When a triangular mesh is used the mesh.dat file now needs specification of the Dirichlet node in line 1. Bug fix: difference inversion file output now corrected. Error checking on presence of input files. Mesh error checks when mesh is read from file **mesh.dat**. Output of model roughness in main log file.
Changes to *R2* from v3.2
Bug fix: geometric factor calculations for triangular meshes. Bug fix: output of electrode codes in the _err.dat file (only has impact if electrode numbers are different to the order they are defined in **R2.in**. Bug fix: incorrect step length set when convergence reached in first alpha trial.
Changes to *R2* from v3.1
Bug fix: calculations with a triangular mesh when node 1 was close to (or at) an electrode (that resulted in high errors in forward model calculations for measurements involving such an electrode).
In a difference inversion the vtk output file now contains the percentage differences. Error checks on input of the data file (**protocol.dat**) are now made (to avoid incorrect electrode numbering). Output of a vtk file containing inversion results at each iteration is now included (previously this was only output to a ***.dat*** file). Some utility codes have been added to help with creating triangular meshes. Third party codes and scripts have been added. Documentation has been improved by adding more illustration of meshing and parameterisation.
Changes to *R2* from v3.0
Anisotropic regularisation is now implemented for a triangular mesh. In earlier versions anisotropic regularisation only worked for quadrilateral meshes. An example triangular mesh problem is included in the distribution and this document. In addition, the output of co-ordinates of the mesh has changed for a triangular mesh (the polarity of the vertical co-ordinate) to allow some consistency with quadrilateral mesh problems and some notes are added in this document about co-ordinate conventions.
Changes to *R2* from v2.7c
All main arrays are now set at run time and so there is no set limit to the size of the problem that can be solved. Because ***R2*** can now work with much larger problems the forward solver uses pardiso (e.g. http://www.pardiso-project.org/) as a linear equation solver, which allows more compact array storage. Users may find that smaller problems run a little slower than with earlier versions because of the overheads in using this solver. ***R2*** now also allows more control of the convergence by specifying a target decrease in misfit at each iteration (see changes to entry of Line 18 in **R2.in**).
Changes to *R2* from v2.7b
Option to output roughness matrix and jacobian.
Changes to *R2* from v2.7a
Bug fix for input of poly line specifying output region. Note that example files supplied for v2.7a and earlier will not output the region properly with 2.7b – these input files have been changed with this release. Also changed upper limits of problem size.
Changes to *R2* from v2.7
Bug fix for case when parameter number set to zero.
Changes to *R2* from v2.6
Version 2.7 now allows the user to specify the region of the mesh to be output. In addition, if a difference inversion is run then an additional output file is created giving the percentage change from the baseline model. This version also contains some bug fixes to vtk output. The input file is changed from earlier versions to allow specification of the output region.

2. **Computer requirements for *R2* v4.02**

In this release two versions have been compiled for the Windows environment. A 64bit version, **R2.exe**, is provided in the package, along with the 32bit version (**R2_w32.exe**). The 32bit version is only provided for continuity of ***R2*** and its use is not recommended and it has not been tested. Linux and Mac users should be able to run ***R2*** with the command "wine R2.exe" (thanks to Rodolphe Cattin for this tip).

NOTE 1: ***R2*** *is provided as a standalone executable. It does not need to be installed – the executable is put in the folder containing the input files and run from there. Output files will be created in the same folder.*

NOTE 2: *You will be able to run* ***R2*** *by double clicking the executable. However, if the program stops abruptly (for example, due to an error in the input file or if you are trying to run an executable compiled for a different processor architecture) then you will not see any error message on the screen since the window will disappear. Therefore, it is advisable to run* ***R2*** *from the Command Prompt in Windows (just run CMD from the Start Menu – you may need to move your working directory and run* ***R2*** *from there).*

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

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

3. **Introduction to *R2* v4.02**

R2 is a forward/inverse solution for 3D or 2D current flow in a quadrilateral or triangular mesh. The mesh is made up of a set of elements. Parameters (for the inverse solution) are made up of one or more elements. Electrodes are specified at node points. These are the corners of the elements. The boundary conditions along all four boundaries of the mesh are Neumann conditions (zero flux) and therefore if you are investigating a half space you must extend left, right and lower boundaries of the mesh to some distance away from the area of investigation (typically 5 to 10 times the distance – see later). One of the nodes must be set to a Dirichlet node (see later). The mesh can be made up of either quadrilateral elements or triangular elements.

The current version does not have upper limits set for the size of the problem that can be solved. However, it is important that the user has some appreciation of whether the problem they are trying to solve is realistic for their given hardware. Large problems in inverse mode can be memory hungry. As soon as the user's RAM is used then the computer will start using virtual memory (paging to disk) which can be very slow. To help the user asses memory needs ***R2*** will output an estimate of the memory needs early on in its execution. For large problems it is important that the user compares this with physical memory (RAM) that is available.

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

R2 is provided for non-commercial use. Any users wishing to use ***R2*** for commercial applications should contact the author.

4. **Designing meshes**

R2 uses a mesh of finite elements to compute the forward model (in both forward and inverse mode). As stated earlier, the mesh is based on a quadrilateral (*structured* or *unstructured*) or triangular (*unstructured*) mesh of finite elements. Each element is defined by node points on its perimeter (see Figure 4.1). ***R2*** computes the voltage at each node point given a dipole current source applied at a pair of nodes. Therefore, the minimum requirement of the mesh is that the electrodes must be sited at node points. The accuracy of the computed voltage field is strongly dependent on the discretisation of the nodes: nodes should be closely spaced near electrode sites as voltage gradients will be high.

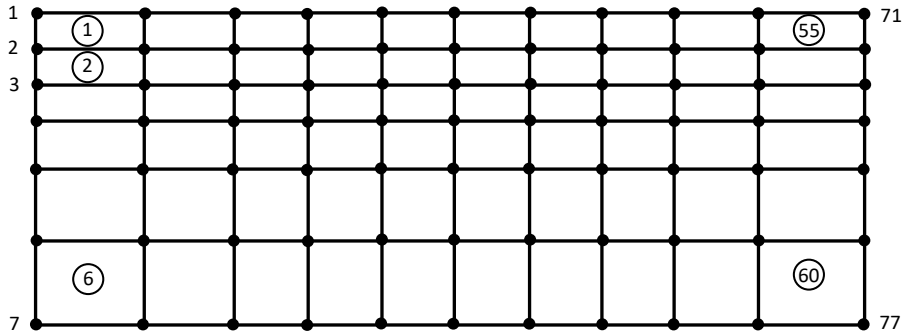


Figure 4.1: Example finite element showing ordering of nodes and elements (in circles) for a structured quadrilateral mesh. For unstructured meshes the user defines the element and node numbering.

The mesh should clearly cover the survey area laterally and the expected depth of investigation, however, given that injected current will transfer further horizontally and vertically, the mesh should extend sufficiently in order to account for this (mimicking infinite boundaries). There is no need to retain a fine discretisation in these 'infinite' boundary regions: it is good practice to let the elements gradually increase in size laterally and vertically outside the region of investigation. Figure 4.2 illustrates the zoning of discretisation. Note that this example is for a surface electrode array. For others, e.g. cross-borehole, similar concepts are applied.

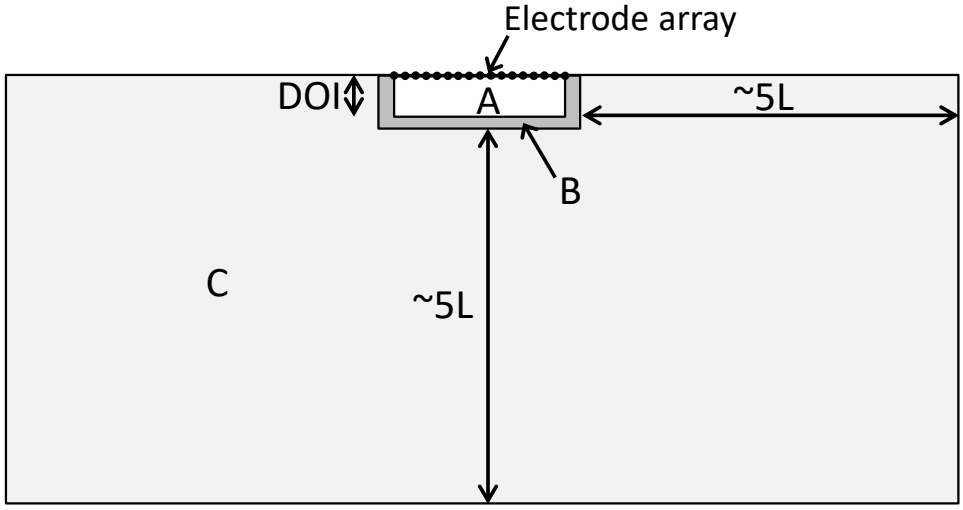


Figure 4.2: Extending the finite element mesh to account for 'infinite' boundaries. DOI is the depth of investigation. Zone A is the region of interest; elements should be finely discretised especially near the electrodes. Zone B has the same discretisation as A and is included to ensure good accuracy of the forward calculations; this zone typically extends two or three time the electrode spacing. Zone C typically extends ~5L where L is the length of the longest current dipole. Discretisation should gradually get coarser in Zone C, moving away from the region of investigation.

A basic (structured) quadrilateral mesh in ***R2*** is defined by a grid of numnp_z rows and numnp_x columns of nodes. This type of mesh can be set in three different ways (Figure 4.3). The simplest type is a rectangular grid (Figure 4.3a). In **R2.in** we define this as mesh_type = 4 (see later). In this case, the elevation of all nodes in a given row is constant. The mesh is defined by the x co-ordinates of each column, the z co-ordinates of the nodes in the top row (i.e. the topography) and the depth of each row relative to the top row. If the topography varies across the survey area then a mesh shown in Figure 4.3b results. Again this is mesh_type = 4. A more complex mesh structure (see Figure 4.3c) can also be created. In **R2.in** we define this as mesh_type = 5. In this case the x co-ordinates are still fixed for each column but the elevation of all nodes in each column must be input. In comparison to mesh_type = 5 more input defining the geometry is needed but the co-ordinates of every node is not needed.

The basic (structured) meshes described above are generated by ***R2*** following input of geometrical parameters in **R2.in**. A more sophisticated (unstructured) quadrilateral mesh (e.g. Figure 4.4) can also be used in ***R2*** but this must be read from a separate file **mesh.dat** (mesh_type = 6)**.** The file will contain the co-ordinates of all nodes, along with the element definitions (i.e. the four node numbers for each element). The specific format of **mesh.dat** is defined later.

For more complex geometry ***R2*** allows the use of a triangular mesh. Again, the co-ordinates for all nodes must be defined, along with the element definitions (i.e. the three node numbers for each element). In **R2.in** we define this as mesh_type = 3 and the mesh is read from the file **mesh.dat**.

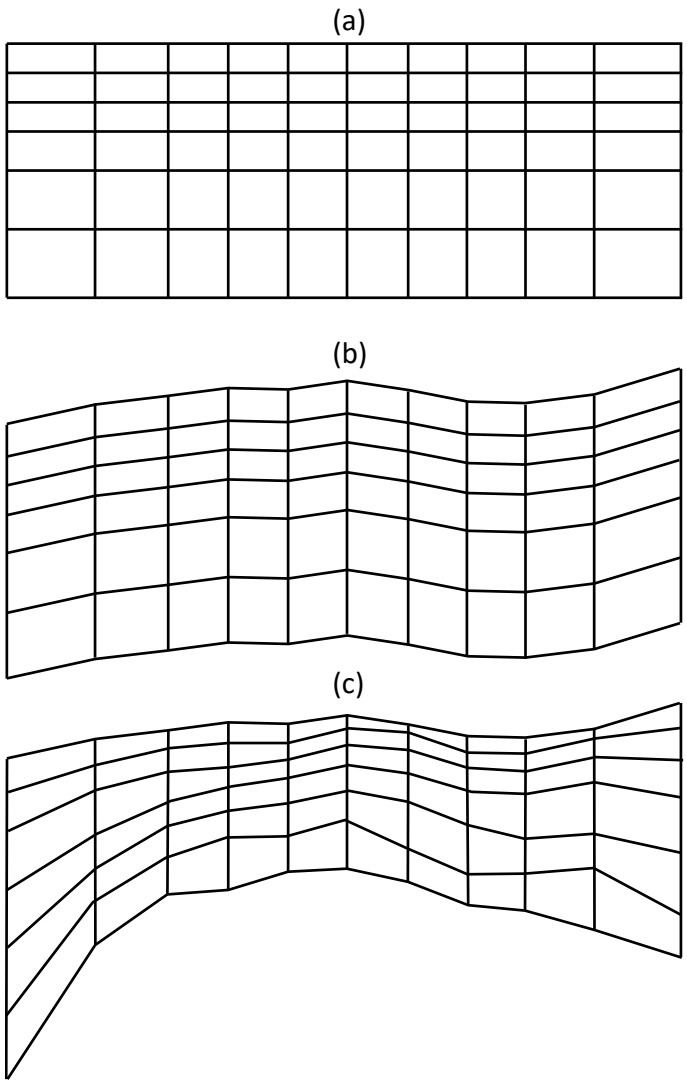


Figure 4.3: Basic (structured) quadrilateral mesh types. (a) and (b): mesh_type = 4; (c): mesh_type = 5.

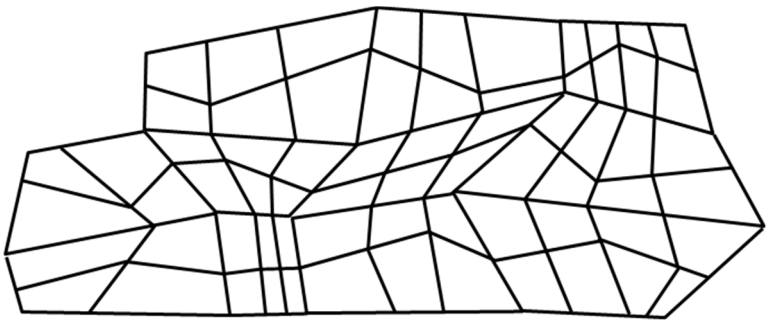


Figure 4.4: Unstructured quadrilateral mesh example (mesh_type = 6).

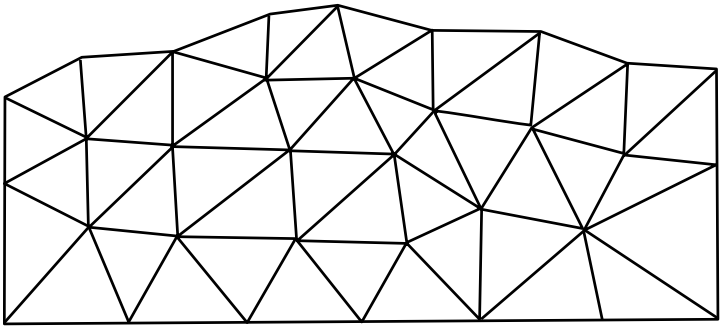


Figure 4.5: Triangular mesh example (mesh_type = 3).

If full flexibility in mapping the geometry of the problem is required then a triangular mesh is preferred. In some instances the use of a pre-defined quadrilateral mesh (mesh_type = 6) may be effective. It doesn't need to be unstructured. Figure 4.6 shows an example where rectangular geometry is used but since the mesh cannot be defined as one set of rows and columns, the mesh must be predefined and read from **mesh.dat**.

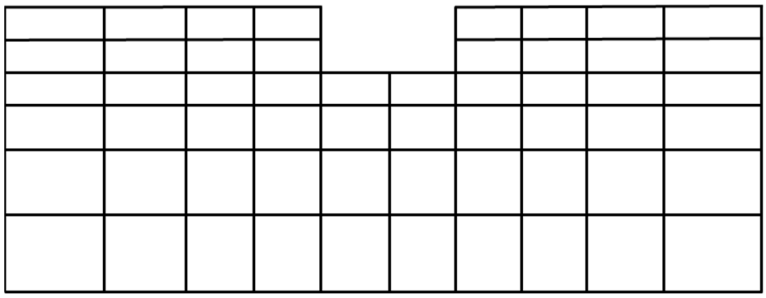


Figure 4.6: Structured quadrilateral mesh example (mesh_type = 6).

Triangular meshes not only allow full flexibility in defining geometry but also result in more computationally efficient meshes since the extremities of the mesh can be made to have coarser discretisation (see Figure 4.7). If you are working with a triangular mesh then you must create the mesh and store details of the geometry of the mesh in a file **mesh.dat**. There are a number of good meshing tools available. ***Gmsh*** (see http://www. gmsh.info) is a powerful 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***).

A key constraint of the structured meshes (mesh_type = 4 or 5) is that defining the geometry of parameter cells in an inverse solution is constrained (see section 5). If the mesh is prepared in **mesh.dat** (mesh_type = 3 or 6) the user has full control of parameter discretisation and zoning (see later).

Gmsh mesh utility codes for *R2*

The ***R2*** download package contains some simple codes for working with ***Gmsh***. These are located in the **Mesh_utilities/Binley** folder. ***GenGmshGeo2D*** creates a geometry file for ***Gmsh***. This can then be loaded in ***Gmsh*** and a 2D mesh easily created. The mesh in ***Gmsh*** is saved as a file **GenGmshGeo2D.msh** which can be transformed to the **mesh.dat** format for ***R2*** using the partner code ***GmshMsh2R2***. Note that Gmsh v3.0.6 was used in generating these scripts. More recent versions may not be compatible.

An important thing to note with mesh generation is that all finite elements should have node numbers in a counter-clockwise direction. I have noticed that ***Gmsh*** can produce mesh output with element nodes ordered in a clockwise direction. ***R2*** now includes a check on this and will fix ordering errors. ***R2*** will also check if duplicate nodes exist in the mesh (more than one node with the same coordinates). If these are found ***R2*** will report them and terminate – the user must then rectify the errors in the mesh file.

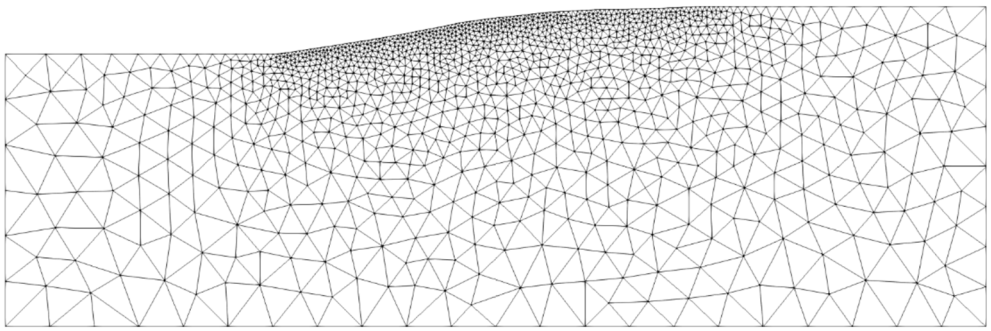


Figure 4.7: Example triangular mesh with variable discretisation.

Niels Claes (University of Wyoming) has provided a Matlab script (see folder **Mesh_utilities/Claes**) that will create a mesh file using ***Gmsh***.

Jimmy Boyd (British Geological Survey/ Lancaster University) has written a python script to convert Gmsh msh files to **mesh.dat** format for ***R2***. See **Mesh_utilities/Boyd**

Additionally, the graphical software ***ResIPy*** (https://pypi.org/project/resipy/) allows mesh generation within its interface of quadrilateral and triangular mesh.

Again, note that these scripts have been produced with specific versions of ***Gmsh*** and may not be compatible with more recent versions. Also note that the author has not tested these third party scripts.

5. **Parameterisation**

For a forward model calculation the resistivity for each element must be defined. For an inverse solution the starting resistivity model must be defined. This is normally a uniform model. In inverse mode the discretisation of parameters must be set. The simplest way to do this is to set each finite element as a parameter. In some cases a more complex parameterisation is required. In a structured quadrilateral mesh we can define 'patch' sizes for parameters. A patch is a group of elements, which is illustrated in Figure 4.8. The advantage of such patching of elements is the reduced computational time, since the number of parameters is reduced.

In some instances we may wish to fix the resistivity in part of the mesh, i.e. the parameters may only cover a subset of the finite element mesh (see, for example, Figure 4.9). ***R2*** allows this type of parameterisation by stating where the patching starts and ends in both x and y direction.

Greater control of parameterisation is allowed when a mesh is prepared *a priori*, for either a triangular mesh or a quadrilateral mesh (e.g. Figure 4.10) by defining a parameter number for each element in **mesh.dat**. Note that if a parameter number 0 is assigned for an element then the inverse solution fixes the resistivity of this element to the starting value.

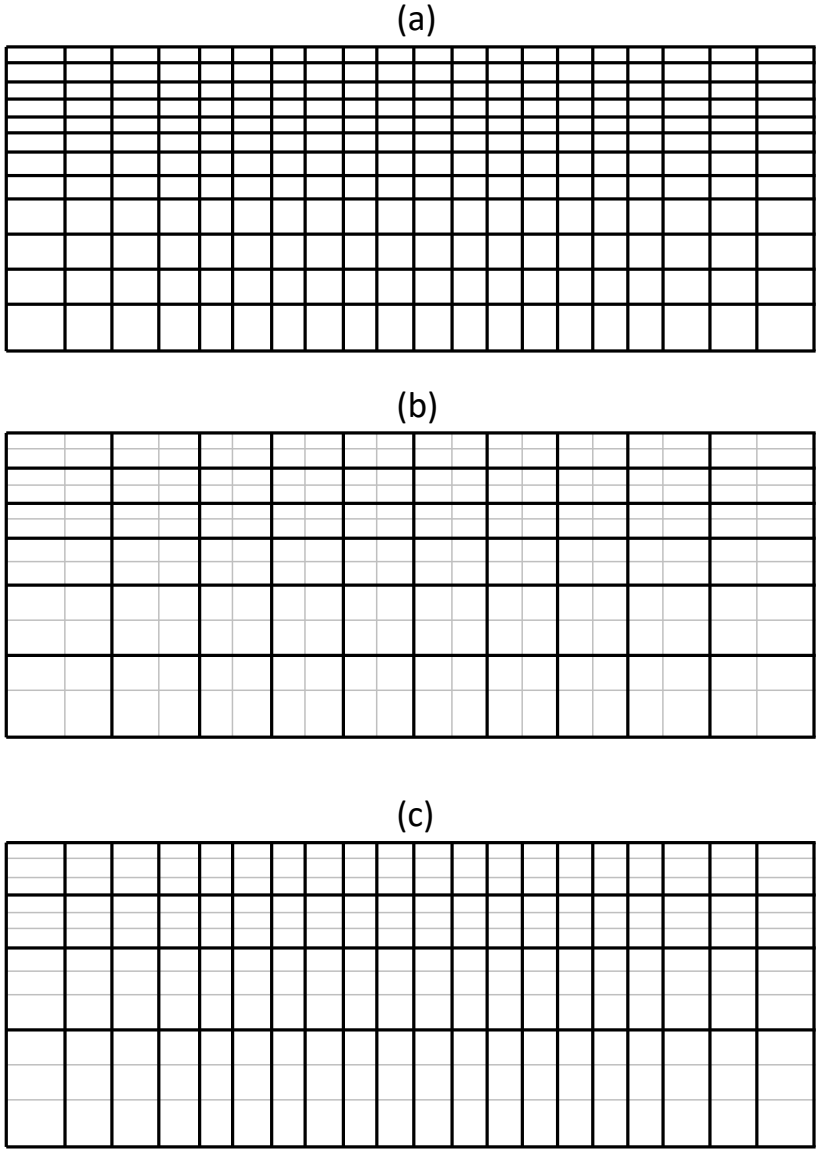


Figure 4.8: Defining parameter boundaries in a structured mesh. The grey lines show the finite element mesh, the black lines show the parameter zones. (a) Each element is a parameter. (b) Each parameter is a 2 x 2 patch of elements. (c) Each parameter is a 3 x 1 patch of elements.

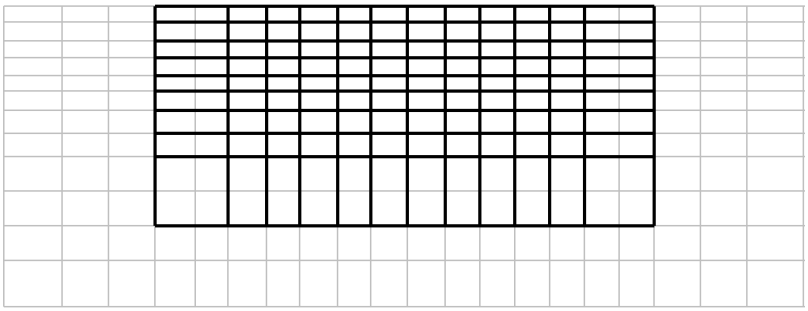


Figure 4.7: Parameterisation of a subset of the structured finite element mesh. The grey lines show the finite element mesh, the black lines show the parameter zones.

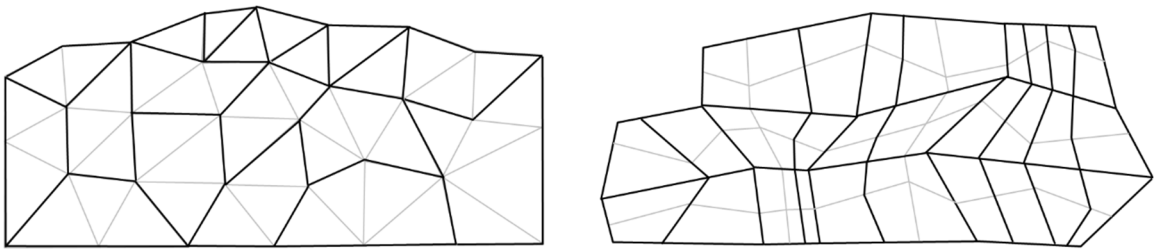


Figure 4.10: Parameterisation of an unstructured mesh defined in **mesh.dat**. The grey lines show the finite element mesh, the black lines show the parameter boundaries.

Further control of parameterisation is achieved through 'zoning' of parameters. In ***R2*** a zone is defined as a congruent collection of parameters (see Figure 4.11). Smoothing (in the inverse solution) does not occur across zones. This can be useful if *a priori* information allows the user to define sharp contrasts (e.g. at a water table boundary).

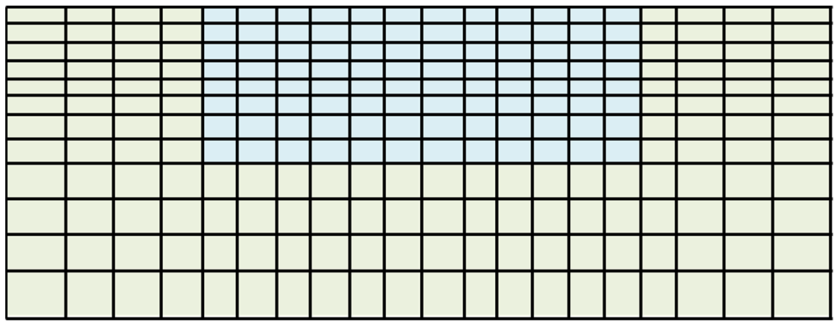


Figure 4.11: Zoning of parameterisation. The two colours represent different parameter zones in the mesh.

Again, full flexibility is possible with an unstructured mesh (see Figure 4.12). For each element the parameter and zone is defined in the **mesh.dat** file.

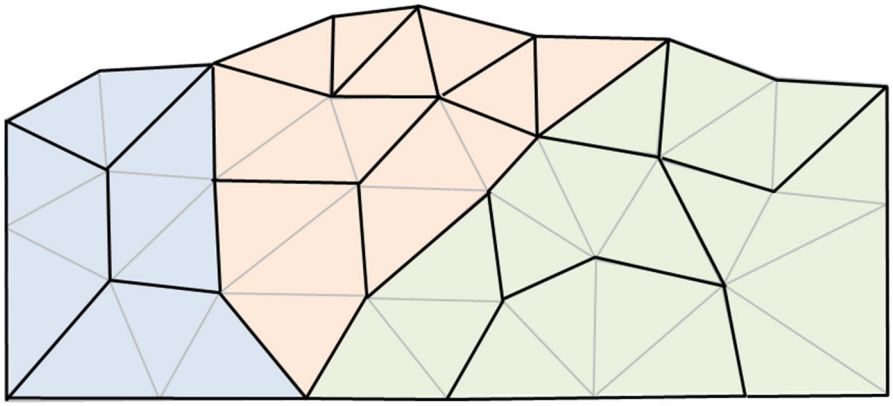
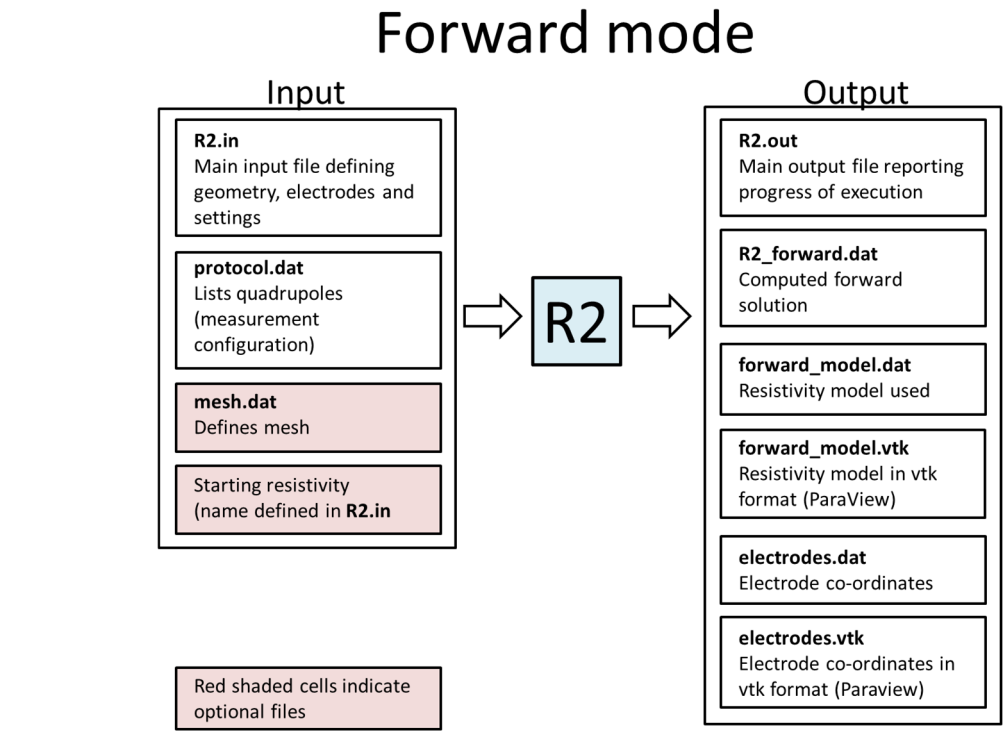


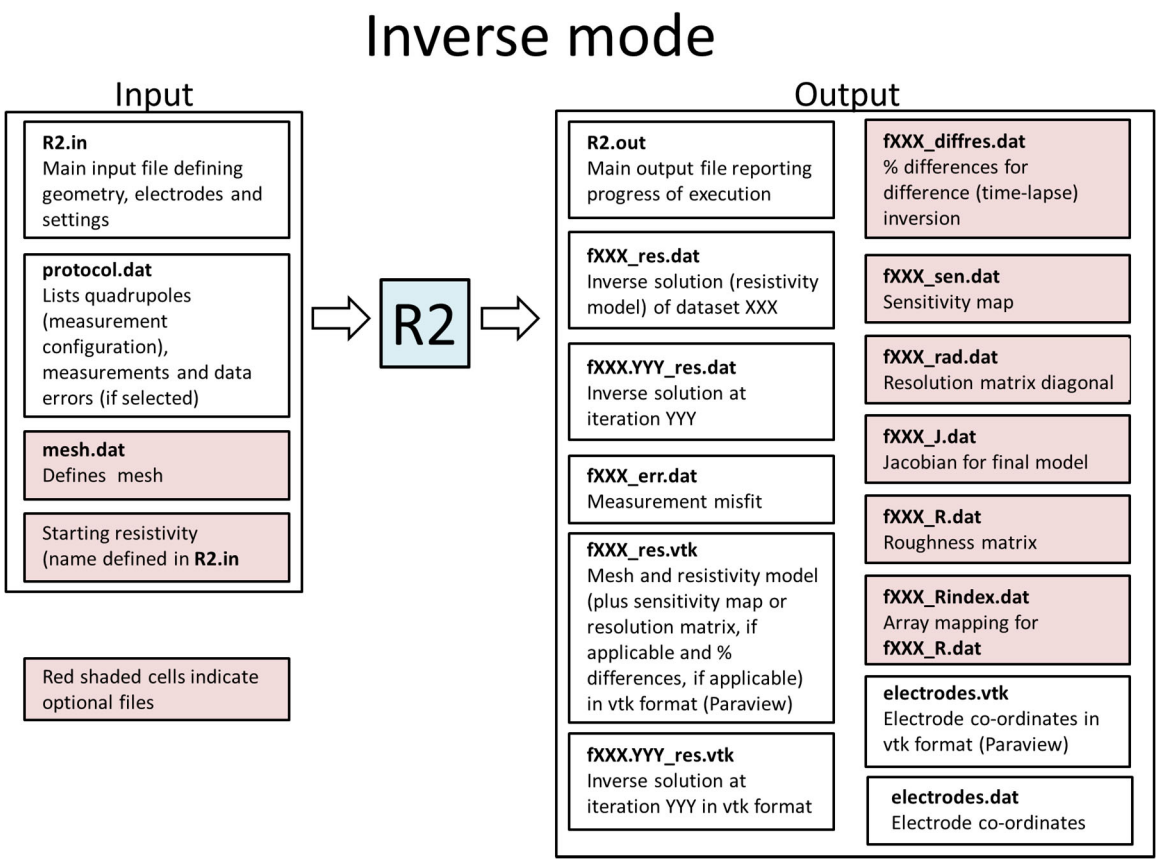
Figure 4.12: Zoning of parameterisation in an unstructured triangular mesh. The three colours represent different parameter zones.

In summary, when a mesh is defined in **mesh.dat**, each element is assigned (i) a parameter number (which is zero if it should remain fixed throughout the inversion); (ii) a zone number. For most cases the parameter number of each element is the element number and the zone is 1.

6. Input and output files

The input and output files used by ***R2*** are shown schematically below, and described in detail later.





R2 requires at least two input files: **R2.in** and **protocol.dat.** If a pre-defined mesh is used then an additional input file – **mesh.dat** – is required. **R2.in** contains information on the geometry of the problem to be solved. **protocol.dat** contains the four electrode configuration (and measurement for inverse mode). In inverse mode you can repeat datasets in **protocol.dat.**

R2 will output a number of files:

- **R2.out** which will contain main log of execution.
- **electrodes.dat** contains the coordinates of the electrodes.

If the problem to be run is a forward model then ***R2*** will output:

- **R2_forward.dat** which will contain the forward model for the electrode configuration in **protocol.dat**. The format of **R2_forward.dat** is the same as **protocol.dat** but with 2 extra columns: the first contains the calculated resistances and the second contains the calculated apparent resistivities. Note that the apparent resistivities are computed assuming infinite boundaries and a flat topography at z=0. If these do not apply then the computed geometric factors will be incorrect, and consequently the apparent resistivities will be incorrect. If ***R2*** computes the absolute value of the geometric factor to be less than 1e-10 then the apparent resistivity that is output is assigned to a value of -100000.00000 in the output file. This does not necessarily mean that the computed transfer resistances are poorly estimated – more likely that the assumptions of infinite boundaries and a flat topography at z=0 is not applicable.
- **forward_model.da**t which will contain the resistivity distribution for your forward model (i.e. what you specified in the input for ***R2***). Note that the format of these will be the same as described below for inverse mode.
- **forward_model.vtk** as above but vtk format (allowing plotting in ***ParaView***, for example).

If the problem to be run is an inverse model then ***R2*** will output:

- **f001_res.dat** which will contain the resistivity result of the inverse solution. **f001_res.dat** will contain four columns: x, z, resis, log10(resis), where x, z are coordinates at centroid of each element and resis is the resistivity in that element and log10(resis) is $\log_{10}$ of the resistivity value. The format is setup to work directly with contouring programs like Surfer.
- **f001_err.dat** will contain nine columns. The first four columns contain the electrode numbers. In the fifth column is the normalised data misfit, the next column contains the observed data recorded as an apparent resistivity, the next column contains the equivalent apparent resistivities for the computed model, the next column shows the original data weight (i.e. data standard deviation in same units as data), the next column is the final data weight, the last columns shows a "1" if any weights have been changed during the inversion, otherwise a "0" will appear. **NOTE:** the output format of **f001_err.dat** changed in ***R2*** v3.2.
- if you select resolution matrix calculation then **f001_rad.dat** will contain the diagonal of the resolution matrix for all elements. A value close to 1 indicates that the parameter for that element can be resolved perfectly, a value close to 0 indicates that the parameter cannot be resolved at all. The format is the same as **f001_res.dat**.
- if you select sensitivity map calculation then **f001_sen.dat** which 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 coverage. You will get a value for all elements. High values indicate high sensitivity to data, low values indicate poor sensitivity. Plot on a log scale. The format is the same as **f001_res.dat**.

- if you select sensitivity map calculation and output of the jacobian then three other files will be output: **f001_J.dat** (the jacobian); **f001_R.dat** (the roughness matrix); **f001_Rindex.dat** (the mapping of the roughness matrix, as the latter is stored in compressed form). All three files will have two integers in line 1. These are the array sizes. Note that the first subscript varies slowest, so if line 1 is "5 10" then an array x of size (5,10) is stored as: x(1,1), x(1,2), … , x(1,5), x(2,1), x(2,2), …. ,x(5,10). The jacobian is of size number of measurements x number of parameters. The roughness matrix is stored as number of parameters x N, where N will be either 5 or 13 depending on the roughness scheme used. The file **f001_Rindex.dat** shows the mapping of the compressed form of the roughness matrix in **f001_R.dat**, e.g. if line 2 (i.e. the line with the first row of the roughness matrix, given that the first line of the file contains the size of the array) of **f001_Rindex.dat** is "1 2 13 0 0" then in line 2 of **f001_R.dat** the roughness matrix entries R(1,1), R(1,2), R(1,13) are stored. Note that the zeros in **f001_Rindex.dat** mean than no entry exists because this parameter (parameter 1) is only connected to two other parameters (parameters 2 and 13).

- **f001_res.vtk** will contain resistivity, log10 resistivity, log10 sensitivity (if selected) and log10 resolution (if selected) in vtk format (allowing plotting in ***ParaView***, for example).

- If you have more than one dataset in **protocol.dat** (see later) then the files **f001_res.dat, f002_res.dat, f003_res.dat,** etc will be created. Similarly a set of **_err.dat, _rad.dat** and/or **_sen.dat** files will be output.

- The output **f001_res.dat** is made at convergence, however, sometimes it is useful to look at the resistivity image at various stages in the iterative process. For all iterations prior to the final iteration a file **f001.XXX_res.dat** will be output, where XXX is 001, 002, 003, etc for the first, second, third, etc iterations. The format of this file is the same as **f001_res.dat**. If multiple datasets appear in **protocol.dat** then corresponding output files will be created. For example, **f003.005_res.dat** will be the fifth iteration of the inversion of the third dataset in **protocol.dat**.

- **f001.XXX_res.vtk** will be output at each iteration and contain the resistivity model in vtk format at iteration XXX (001, 002, etc).

- If a difference inversion is run (reg_mode =2 in line 21 of **R2.in**) then an additional file will be created for each dataset. **f001_diffres.dat** will contain three columns: x, z, diffresis, where x,z are coordinates at centroid of each element and diffresis is the percentage change in resistivity from the baseline model. The format is setup to work directly with Surfer.

In addition ***R2*** will output:

- **electrodes.dat** , which contains the co-ordinates of the electrodes. The values are in three columns: x,z,y (where y is a dummy value – set to 0.0).

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

NOTE: If ***R2*** fails to converge in inverse mode, all output files except the sensitivity map/resolution matrix will still be output in order to allow the user to assess the source of the problem (the **fXXX.err** file is useful for this) but the user should check the **R2.out** file to ensure convergence is achieved before using any computed resistivity models.

Some comments on co-ordinate sign convention

When a quadrilateral mesh is created the user specifies the horizontal and vertical co-ordinates of the rows and columns forming the mesh. The convention here (see line 9 in **R2.in**) is for the vertical co-

ordinates to be specified as depth (not elevation). When these are output in a **.dat** file (e.g. **forward_model.dat**, for a forward model, or **f001.dat**, for an inversion) then the vertical co-ordinate sign is changed. This is so that programs like Surfer will show the section properly. The same switching of sign is changed in the vtk output (e.g. **forward_model.vtk** and **f001.vtk**). The electrode co-ordinates (output in **electrodes.dat** and **electrodes.vtk**) will also have a switched sign for the vertical co-ordinates.

When defining a mesh in **mesh.dat** meshes should be created with the vertical co-ordinate representing elevation, not depth. And so when such a mesh is used, the sign change is not made in any of the output files listed above.

7. Details of R2.in

Line1: (Char*80) header

where header is a title of up to 80 characters

Line 2: (2 Int, Real, 2 Int) job_type, mesh_type, flux_type, singular_type, res_matrix

where job_type is 0 for forward solution only or 1 for inverse solution; mesh_type is 3 for triangular mesh (see Figure 4.5), 4 for a regular quadrilateral mesh, 5 for a more generalised structured quadrilateral mesh (see Figure 4.3c), 6 for a general quadrilateral mesh (see Figure 4.4); flux_type is 2.0 for 2D current flow (i.e. line electrodes, which are infinitely long orthogonal to the section) or 3.0 (usual mode) for fully 3D current flow (i.e. point electrodes); singular_type is 1 if singularity removal (see Lowry et al., 1989) is to be applied (otherwise 0). Note that singularity removal can only be applied is (a) the boundaries are infinite and (b) the z=0 plane defines the upper boundary; (c) the upper boundary is flat (if any of these conditions do not apply then singularity removal cannot be used); res_matrix is 1 if a 'sensitivity' matrix is required for the converged solution. This matrix is not the Jacobian but is the diagonal of [$J^T W^T W J$] which gives an idea of the mesh sensitivity (see equation 5.20 of Binley and Kemna, 2005). One value is stored for each finite element in the mesh. High values indicate high sensitivity, low values indicate poor sensitivity. Plot on a log scale. Set res_matrix to 2 if the true resolution matrix is computed for a converged solution and the diagonal is stored (see equation 5.18 of Binley and Kemna, 2005), note that this calculation is more time consuming than the 'sensitivity matrix' option. Set res_matrix to 3 if the sensitivity map is required and an output of the jacobian matrix and roughness matrix. If neither sensitivity map or resolution matrix is required then set res_matrix to 0

If mesh_type is 3 or 6 then the file **mesh.dat** must be supplied which contains the mesh details including node coordinates and element indices (see details later).

If (mesh_type = 4) then a regular quadrilateral mesh is to be used and the following are read:

Line 3: (2 Int) numnp_x, numnp_z

where numnp_x is number of nodes in the x direction (horizontal) and numnp_z is the number of nodes in the z (vertical) direction

Line 4: (numnp_x Real) xx

where xx is an array containing x coordinates of each of numnp_x node columns

Line 5: (numnp_x Real) topog

where topog is an array containing elevations of each of numnp_x node columns. If the topography is flat then set topog to zero for all values.

Line 6: (numnp_y Real) zz

where zz is an array containing the depths of each of numnp_z node rows relative to the topog array. Set zz(1) to zero and the other values to a positive number (i.e. zz represents depth, not topography).

Else if (mesh_type = 5) then a more generalised structured quadrilateral mesh is to be used and the following are read:

Line 7: (2 Int) numnp_x, numnp_z

where numnp_x is number of nodes in the x direction (horizontal) and numnp_z is the number of nodes in the z (vertical) direction

Line 8: (numnp_x Real) xx

where xx is an array containing x co-ordinates of each of numnp_x node columns

Line 9: (numnp_y Real) zz

where zz is an array containing elevations (not depths) of each of numnp_z node rows for column 1 in the x direction. For each column of nodes zz(1) is the topography.

Repeat Line 9 for all numnnp_x columns.

End if

Note: It is wise to add a carriage returns to break up a long list of input values (in Line 4, 5, 6, 8 and 9, for example). Don't write more than 20 numbers on each line as the compiler may not like it.

If (mesh_type = 3) then read the following

Line 10: (Real) scale

where scale is a scaling factor for the mesh co-ordinates. This is usually 1.0 but if a standardised mesh is used, say for a unit circle, then this scaling factor is useful to adjust the mesh for a specific problem. Set scale=1 if you do not wish to change the coordinates of the mesh defined in **mesh.dat.**

End if

Line 11: (Int) num_regions

where num_regions is number of resistivity regions that will be specified either as starting condition for inverse solution or actual model for forward solution. The term "region" has no significance in the inversion – it is just a means of inputting a non-uniform resistivity as a starting model for inversion or for forward calculation.

If (num_regions = 0) then read the following

Line 12: (15*Char) file_name

where file_name is the name of the file containing the resistivitities from a previous inversion (the **_res.dat** file that had been produced). Note that the file_name must be no more than 15 characters and there should be no spaces before the file name and no characters in the line after the file name.

Else

Line 13: (2 Int, Real) elem_1, elem_2, value

where the resistivity value will be assigned for all elements from elem_1 to elem_2 (inclusive). Note that for a quadrilateral mesh the elements are numbered down columns first (top to bottom) then along rows (left to right).

Repeat Line 13 for all num_regions

End if

NOTE: *you must assign all elements a starting value. The number of elements in the mesh is (numnp_x-1) x (numnp_y-1) for a structured quadrilateral mesh. All these elements must be assigned a resistivity. Note also that if you assign an element a value, it will overwrite any previous assignment.*

If (job_type = 1. i.e. an inverse solution) then read the following

If (mesh_type = 4 or 5) then read the following

Line 14: (2 Int) patch_size_x, patch_size_z

where patch_size_x and patch_size_z are the parameter block sizes in the x and z direction, respectively. We differentiate between parameter size and element size to allow faster computation. The larger the patch size the few parameters and the faster the inversion, however, if we increase it too much we will reduce the flexibility to create variation of resistivity. If computational time is not a problem then use a patch size of 1 for x and z. Note that the number of elements in the x direction must be perfectly divisible by patch_size_x and the number of elements in the z direction must be perfectly divisible by patch_size_z otherwise set them both to zero. See examples in Figure 4.8. For Figure 4.8a patch_size_x =1 and patch_size_z = 1. For Figure 4.8b patch_size_x =2 and patch_size_z = 2. For Figure 4.8c patch_size_x =1 and patch_size_z = 3.

If (patch_size_x = 0) and (patch_size_z = 0) then read the following

Line 15: (2 Int) num_param_x, num_param_z

where num_param_x and num_param_z are the number of parameter blocks in the x and z directions

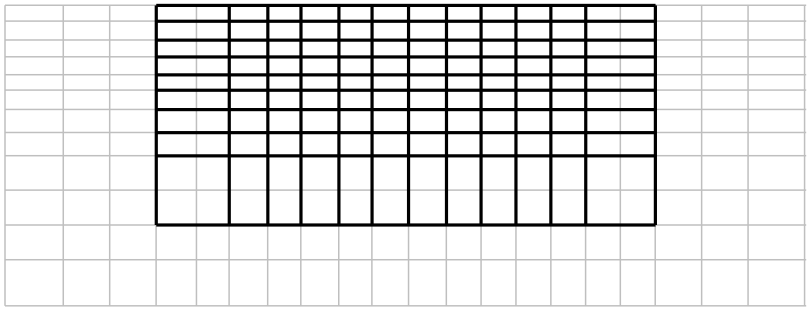
Line 16: (1+num_param_x Int) npxstart, npx(i), i=1,num_param_x

where npxstart is the column number in the mesh where the parameters start; npx specifies the number of elements in each parameter block in the x direction

Line 17: (1+num_param_y Int) npzstart, npz(i), i=1,num_param_z

where npzstart is the row number in the mesh where the parameters start; npz specifies the number of elements in each parameter block in the z direction

See example in Figure 4.9 (copied below). For this example we would set num_param_x = 12 and num_param_z = 9. Then we would set Line 16 as 3, 2,1,1,1,1,1,1,1,1,1,2 and Line 17 as 1, 1,1,1,1,1,1,1,1,2.



End if

End if

NOTE: *the following line input is different to v2.7c and older versions of* ***R2***

Line 18: (Int, Real) inverse_type, target_decrease

where inverse_type is 0 for pseudo-Marquardt solution or 1 for regularised solution with linear filter (usual mode) or 2 for regularised type with quadratic filter or 3 for qualitative solution or 4 for blocked linear regularised type (see also line 24). Note that the blocking defined here is only for a quadrilateral mesh – for blocking within a triangular mesh see the details for preparing **mesh.dat** later. target_decrease is a real number which allows the user to specify the relative reduction of misfit in each iteration. A value of 0.25 will mean that ***R2*** will aim to drop the misfit by 25% (and no more) of the value at the start of the iteration. This allows a slower progression of the inversion, which can often result in a better convergence. If you set target_decrease to 0.0 then ***R2*** will try to achieve the maximum reduction in misfit in the iteration.

if (inverse_type = 3) then

Line 19: (Int) qual_ratio

where qual_ratio is 0 for qualitative comparison with forward solution, i.e. only when one observed data set is available, or qual_ratio is 1 if the observed data in **protocol.dat** contains a ratio of two datasets. This option is a legacy feature of ***R2*** and is not normally adopted.

Line 20: (2 Real) rho_min, rho_max

where rho_min and rho_max are the minimum and maximum observed apparent resistivity to be used. rho_min can be negative. Note that if you are computing on an enclosed region (i.e. not an infinite half space), or if the topography is not flat and set to zero, then the apparent resistivities will not be valid and so extreme values should be used in Line 20, e.g. -1.0e10, 1.0e10

Else

NOTE: *the following line input is different to v2.4 and older versions of* ***R2***

Line 21: (2 Int) data_type, reg_mode

where data_type is 0 for true data based inversion or 1 for log data based. Note that the latter should improve convergence but may not work for internal electrodes (e.g. borehole type) where the polarity can change due to resistivity distributions; reg_mode is 0 for normal regularisation; or 1 if you want to include regularisation relative to your starting resistivity (this is whatever you have set in input lines 11 to 13); or 2 if you wish to regularise relative to a previous dataset using the "Difference inversion" of LaBrecque and Yang (2001). If you select reg_mode=1 then Line 22 will require a regularisation parameter alpha_s. If you select reg_mode=2 then **protocol.dat** must contain an extra column (see below) with the reference dataset. In addition, your starting model (see Line 12) should be the inverse model for this reference dataset (i.e. you need to invert the reference dataset before running the time-lapse inversion). Also note that if you select reg_mode=2 then data_type is automatically set to 0 irrespective of what was entered in Line 21.

NOTE: *the following line input is different to v2.4 and older versions of* ***R2***

if ((reg_mode = 0) or (reg_mode = 2)) then

Line 22: (Real, 2 Int, Real) tolerance, max_iterations, error_mod, alpha_aniso

Else

Line 22: (Real, 2 Int, 2 Real) tolerance, max_iterations, error_mod, alpha_aniso, alpha_s

End if

where tolerance is desired misfit (usually 1.0); max_iterations is the maximum number of iterations; error_mod is 0 if you wish to preserve the data weights, 2 if you wish the inversion to update the weights as the inversion progresses based on how good a fit each data point makes. error_mod=2 is recommended – this is a routine based on Morelli and LaBrecque (1996). Note that no weights will be increased. The smoothing factor used in the code (alpha) is searched for at each iteration. The search is done over a range of steps in alpha, the number of steps is 10. alpha_aniso is the anisotropy of the smoothing factor, set alpha_aniso > 1 for smoother horizontal models, alpha_aniso < 1 for smoother vertical models, or alpha_aniso=1 for normal (isotropic) regularisation. alpha_s is the regularisation to the starting model (if you set reg_mode = 1 in Line 21). Set alpha_s to a high value (e.g. 10) to highly penalise any departure from this starting model. Note that alpha_s will stay fixed – if you set it too high then ***R2*** may not converge. **R2.out** will report the value of alpha used to regularise smoothing within the image – the regularisation relative to a reference model is additional to this. The user may find setting alpha_s useful as a comparison of inversions from two runs with difference reference models allows an assessment of the depth of investigation following the approach of Oldenburg and Li (1999).

Line 23: (4 Real) a_wgt, b_wgt, rho_min, rho_max

where a_wgt and b_wgt are error model parameters describing the standard deviation of measurements following:

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

where R is the resistance measured (LaBrecque et al., 1996 equation 14). Note that it is the inverse of the standard deviation that is used as weight in the diagonal of the weight matrix **W**.

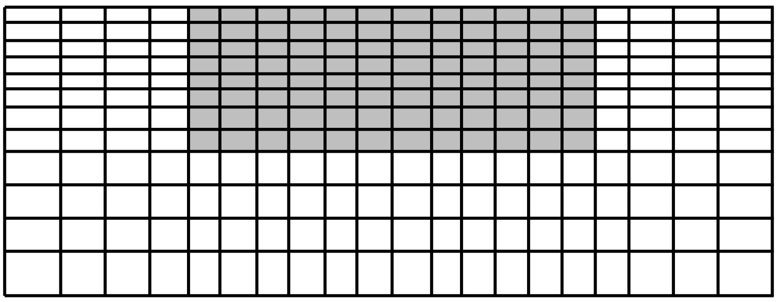
It is advisable to estimate a_wgt and b_wgt from error checks in the field data (ideally from reciprocal measurements - not measures of repeatability). Typically for surface data a_wgt will be about 0.01 Ohms and b_wgt will be about 0.02 (roughly equivalent to 2% error). Note that if you select data_type=1 in Line 21 then although the resistance data are transformed into log apparent conductivities the a_wgt and b_wgt parameters should still reflect the variance of the resistance; rho_min and rho_max are the minimum and maximum observed apparent resistivity to be used for inversion (use large extremes if you want all data to be used). If data are ignored by ***R2*** because of the apparent resistivity limits then these will be reported individually in **R2.out**. **NOTE:** that the apparent resistivity calculations assume that you have set the ground surface to z=0 and that the ground surface is flat. Note also that you can select to include individual errors for each measurement in the data input file **protocol.dat** – to do this a_wgt and b_wgt should be set to 0.0 – **protocol.dat** will then require an additional column (see later).

Line 24: (num_param_x Int) param_symbol

If you have specified zoning of parameters (inverse_type = 4 in line 18) so that each zone is disconnected from other zones then for a structured quadrilateral mesh (see Figure 4.11) the zones are specified by producing a simple plan of the parameter mesh. You must input for each row of parameters an integer representing the parameters. This is repeated for each row. Make sure that you put a space between each integer. As an example consider the zoned mesh in Figure 4.11 (copied below) with 20 elements in the x direction and 12 elements in the z direction. In this example we wish to zone the region shaded. As each element is a parameter then the patch size in x and z is 1, so in total there are 20 parameters (x) by 12 parameters (z). If we want to set the boundary of the shaded region so that there is no smoothing to the unshaded region then we would input:

```
1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Note that we have used 1s and 2s to define the regions. We could have used any other integer.



If for the problem above we had a patch size of 2 in x and z then Line 24 would be:

```
1 1 2 2 2 2 2 2 1 1
1 1 2 2 2 2 2 2 1 1
1 1 2 2 2 2 2 2 1 1
1 1 2 2 2 2 2 2 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
```

If we had defined the problem to have a patch size in x of 4 and a patch size in z of 2 then Line 24 would be:

```
1 2 2 2 1
1 2 2 2 1
1 2 2 2 1
1 2 2 2 1
1 1 1 1 1
1 1 1 1 1
```

Repeat line 24 for all num_param_z

End if

End if

NOTE: Lines 25 to 26 define the region to be output (note that this was new to version 2.7)

Line 25: (Integer) num_xz_poly
where num_xz_poly is the number of x,z co-ordinates that define a polyline bounding the output volume. If num_xz_poly is set to zero then no bounding is done in the x-z 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). **NOTE:** ***R2*** stores the vertical co-ordinates for nodes in a structured quadrilateral mesh with a convention positive upwards. For example, if the ground surface has an elevation of 0m and you wish to output to a depth of 8m then z=-8m must be used for the lower boundary of the polygon. Similarly, if the ground surface elevation is 100m and you wish to output to a depth of 8m then z=-92m must be used for the lower boundary of the polygon. This was not the convention for v2.7a and so any input files created for that version must be changed (this only applies to line 26). If a triangular mesh or quadrilateral mesh in **mesh.dat** is used then the co-ordinates specified in the mesh file are used and the above comments about sign convention do not apply.

Line 26: (2 Real) x_poly(1), z_poly(2)
where x_poly(1), z_poly(1) are the co-ordinates of the first point on the polyline.
Repeat line 26 for all num_xz_poly co-ordinates.

Line 27: (Int) num_electrodes

where num_electrodes is number of electrodes

If (mesh_type = 3 or 6) then

Line 28: (2 Int) j, node

where j is the electrode number and node is the node number in the finite element mesh

Else

Line 29: (3 Int) j, column, row

where j is the electrode number, column is the column index for the node the finite element mesh and row is the row index for the node in the finite element mesh. The column value must be in the range 1 to numnp_x and the row value must be in the range 1 to numnp_y. Both values must be integer values.

End If

Repeat Line 29 for all num_electrodes

END OF INPUT FOR **R2.in**

8. Details of protocol.dat

protocol.dat contains measurement schedule (and data for inverse if selected)

Line 1: (Int) num_ind_meas

where num_ind_meas is number of measurements to follow in file

If (job_type = 1) then

Line 2: (5 Int, 3 Real) j, elec(1,k), elec(2,k), elec(3,k), elec(4,k), v_i_ratio, v_i_ratio_0, data_sd

where j is not used (but usually is used as a measurement number); elec(1,k) is the electrode number for the (M) P+ electrode; elec(2,k) is the electrode number for the (N) P- electrode; elec(3,k) is the electrode number for the A (C+) electrode; elec(4,k) is the electrode number for the B (C-) electrode; v_i_ratio is measured resistance value (or ratio of two measured values if inverse_type=2 and qual_ratio=1 in R2.in); v_i_ratio_0 is the resistance data for background case (only read if reg_mode=2); data_sd is data standard deviation (must be positive and only read if a_wgt and b_wgt in line 23 of **R2.in** are both zero). **NOTE:** v_i_ratio should contain the polarity of the measurement – do not assign only absolute values.

Repeat Line 2 for all num_ind_meas

Else

Line 3: (5 Int) j, elec(1,k), elec(2,k), elec(3,k), elec(4,k)

where j is not used (but usually is used as a measurement number); elec(1,k) is the electrode number for the M (P+) electrode; elec(2,k) is the electrode number for the N (P-) electrode; elec(3,k) is the electrode number for the A (C+) electrode; elec(4,k) is the electrode number for the B (C-) electrode

Repeat Line 3 for all num_ind_meas

End if

You can add as many datasets to the file **protocol.dat**. Just concatenate the datasets into one file. ***R2*** will continue to read and process data using the settings defined in **R2.in**

END OF INPUT FOR **protocol.dat**

9. Details of mesh.dat

It is useful if your mesh generator permits 'materials' to be defined, allowing some zoning of the mesh (to permit blocking at interfaces). Also, you may find it beneficial (for computational efficiency) to create a coarse mesh to define the parameters and then refine this mesh (splitting a triangle element into more elements) to have more elements for the forward solution. The simplest mesh consists of an equal number of parameters and elements and one zone. More complex arrangements allow for grouping of elements into parameters and multiple zones. Regularisation is not applied at the interface of zones.

NOTE: Line 1 changed from version 3.3

Line 1: (3 Int) numel, numnp, ndirichlet

Where numel is the number of triangle elements, numnp is the number of nodes and ndirichlet is the node number of a specified dirichlet node, which should be a node far away from all electrode nodes. If ndirichlet is set to zero then the code will compute the node that is furthest away. However, for some geometries (e.g. a circular region) the node that is the furthest away based on average distance may be close to one of the nodes. For such problems it is advisable to set the node in the centre of the mesh as the dirichlet node to avoid biasing of the computed potential field.

Line 2: (6 Int) n, index(1,n), index(2,n), index(3,n), param(n), zone(n)

Where n is the element number; index(1,n), index(2,n) and index(3,n) are the node numbers of the element, numbered in a counter-clockwise direction (***R2*** will check if this is correct for a triangular or quadrilateral mesh that is read from **mesh.dat**); param(n) is the parameter number of the element (to make every element a parameter then make this value equal to the element number); zone(n) is the zone number for element n. To have one zone make zone(n) equal to 1 for all elements. Zones must be connected elements. Parameters cannot occupy more than one zone. NOTE also, to make an parameter fixed to the starting resistivity, set param(n) to zero but note that if this is done all elements with param(n) = 0 must be at the end of the block of elements (see Surface 8 example below). This will involve reordering elements and care must be taken to ensure that any associated files with the element mapping (e.g. a start resistivity file, if used) but follow the same new element numbering. Note that param(n) and zone(n) are not used in forward mode but these columns still need to be added to **mesh.dat**

Repeat line 2 for all numel elements.

Line 3: (Int, 2 Real) n, x(n), z(n)

Where n is the node number; x(n), z(n) are the coordinates of node n.

Repeat line 3 for all numnp nodes.

END OF INPUT FOR **mesh.dat**

10. **Examples**

The folder "Examples" contains a number of worked examples of ***R2*** to illustrate how to setup input files and work with model output.

Surface electrode array 1 – getting started

The subfolder "Examples/Surface_1/dpdp" contains an example synthetic model of a surface electrode array using a dipole-dipole measurement scheme. The example is taken from Binley and Kemna (2005). For this problem 25 electrodes are positioned at 2m spacing on a flat surface of a half space. The electrodes are numbered 1 to 25 from left to right. A forward model is setup to determine the measured transfer resistances for a dipole-dipole scheme with 117 measurements. The resistivity model is shown in Figure 10.1. A small target with resistivity 10 Ωm lies within a 100 Ωm half space: positioned vertically between depths 1m and 4m and horizontally between 14m and 16m.

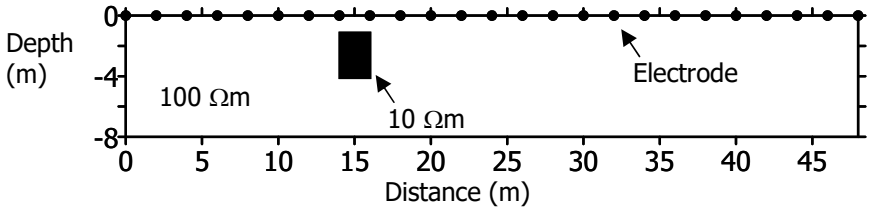


Figure 10.1: Definition of synthetic model for surface array 1 problem

The subfolder "Examples/Surface_1/dpdp/Forward" contains the **protocol.dat** file for the forward problem. Also contained in the folder is the file **R2.in** which defines the geometry and resistivity model. Since the model is a half space the finite element mesh must extend significantly away from the region of investigation (horizontally and vertically downwards). The mesh developed consists of 225 node columns and 49 node rows (i.e. 11,025 nodes, 10,752 elements). The file **R2.in** shows how the mesh is designed to get progressively coarser away from the region of study. Note that the co-ordinates of the mesh have been set so that electrode 1 is at (0,0) for this problem. In the mesh electrode 1 is located at node column 17 (i.e. there are 16 elements to the left of the electrode array to represent an infinite boundary condition to the left. For this example 8 elements are placed between electrodes and so node 2 is at node column 25, node 3 is at column 33, etc. Since the electrodes are located on the ground surface the row node for all electrodes is 1. All the electrode positions are assigned in **R2.in**. The file also assigns the resistivity for all elements. For this problem it is done by defining the resistivity of 9 congruent blocks of elements. First all elements in the mesh are set to 100 Ωm and then 8 columns of vertically adjacent elements are defined to set the 10 Ωm anomaly (remember that the elements are numbered vertically then horizontally).

When ***R2*** is run the output files are:

R2.out, which contains the main log of execution
electrodes.dat, which contains the electrode co-ordinates
R2_forward.dat, which contains the forward model, i.e. the 117 transfer resistances. Note that the apparent resistivity for each of the 117 measurements is also stored.
forward_model.dat, which contains the co-ordinates of the centroid of each finite element in the mesh, the resistivity of each finite element along with the logarithm (to base 10) of the resistivity. This file is useful for checking if the resistivities were defined correctly in **R2.in**

In Binley and Kemna (2005) the same forward model is presented in pseudo section format.

The subfolder "Surface_1/dpdp/Inverse" contains files for running the inversion of the transfer resistances determined above. For this a uniform starting resistivity of 100 Ωm is defined in the file **R2.in**. The 'data' to be inverted is stored in file protocol.dat: here the values are simply the transfer resistances that appeared in the **R2_forward.dat** file described earlier.

For the inverse problem we have used a patch_size of 4 in both x and y directions, i.e. each inverse parameter is a 4 by 4 block of finite elements.

When ***R2*** is run in this case the output files are:

R2.out, which contains the main log of execution;
electrodes.dat, which contains the electrode co-ordinates;
f001_res.dat, which contains the computed resistivity (and $\log_{10}$ resistivity) for each finite element (in the entire mesh – not just the region of interest);
f001_err.dat, which contains the misfit for each of the 117 measurements;
f001_sen.dat, which contains the sensitivity map computed using equation 5.20 in Binley and Kemna (2005);
f001.001_res.dat, which contains the computed resistivity (and $\log_{10}$ resistivity) for each finite element after the first iteration. Note that the inversion converged after 2 iterations for this problem and so this is the only intermediate solution.

Figure 10.2 shows the results of the inversion (compare with Fig 5.8 of Binley and Kemna(2005)). This is an image map of the results in f001_res.dat (x and y in columns 1 and 2 and logarithm of resistivity in column 4). Note that only the region within the electrode array and to a depth of 8m has been plotted.

In Figure 10.3 the sensitivity map is shown (res_matrix in line 2 of **R2.in** is set to 1). The values are computed with the equation 5.20 of Binley and Kemna (2005). High values indicate areas of high measurement sensitivity. It is often useful to use this map to mask the resistivity image (see example on cover page of this document). Showing values of the sensitivity map < 0.001 of the maximum value as opaque is a useful guide for this.

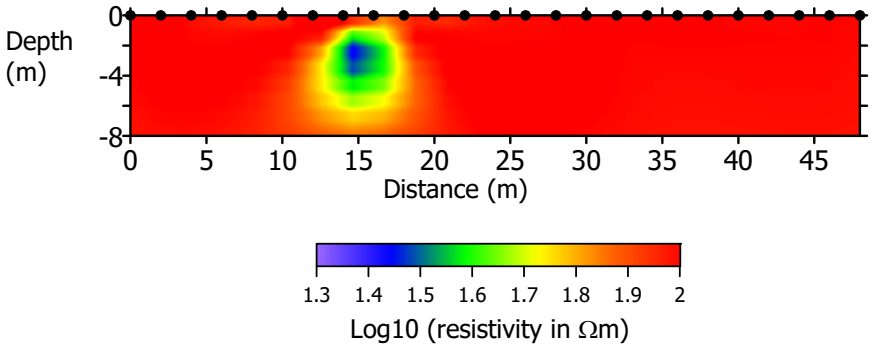


Figure 10.2: Inverse model for surface array 1 problem with dp-dp array

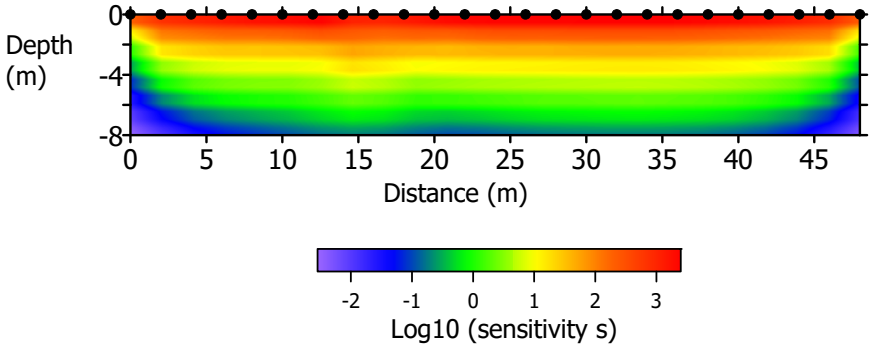


Figure 10.3: Sensitivity map for inverse model for surface array 1 problem with dp-dp array

Had the problem been run with res_matrix in line 2 of **R2.in** set to 2 then the diagonal of the resolution matrix would have been computed. This is useful for comparing models and measurement schemes. In Figure 10.4 the map of the resolution matrix diagonal is shown. Values should be ideally equal to 1 (logarithm equal to 0) – values less than this indicate the effect of smoothing on the parameter value (influence of adjacent parameter values). The map of the diagonal of the resolution matrix is very useful for determining a suitable filter for displaying results.

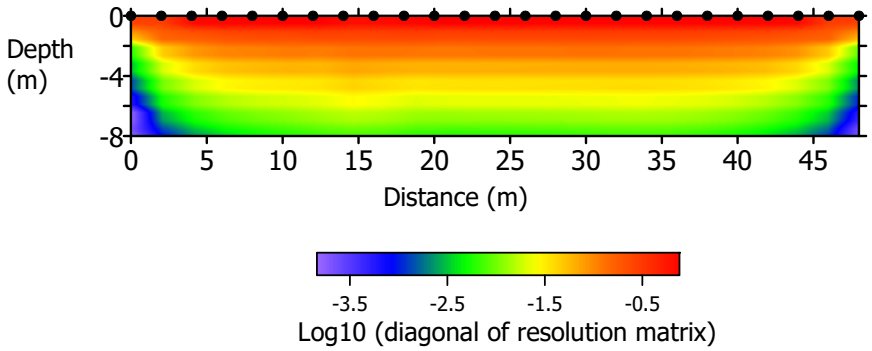


Figure 10.4: Diagonal of resolution matrix for inverse model for surface array 1 problem with dp-dp array

The subfolder "Surface_1/Wenner/Forward" contains files for running a forward model for the same problem but using a Wenner configuration (see figure 5.7 of Binley and Kemna(2005) for the pseudo section. The subfolder "Surface_1/Wenner/Inverse" contains the files for running the inverse model. Figure 10.5 shows the resulting model. Figure 10.6 shows the diagonal of the resolution matrix for this solution, illustrating a weaker resolution in comparison to the dipole-dipole array (c.f. Figure 10.4),

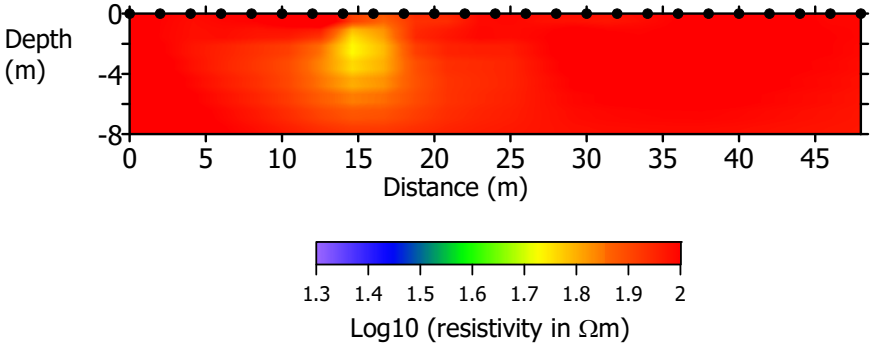


Figure 10.5: Inverse model for surface array 1 problem with Wenner array

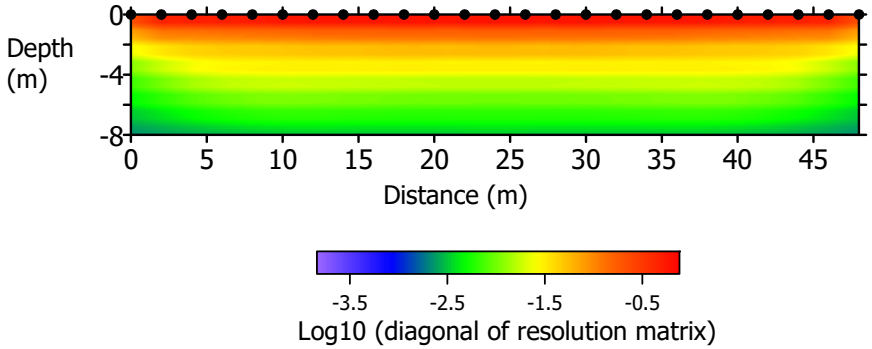


Figure 10.6: Diagonal of resolution matrix for inverse model for surface array 1 problem with Wenner array

Surface electrode array 2 – adding topography

The subfolder "Examples/Surface_2/dpdp" contains an example similar to the previous case but with varying surface topography. Here the ground surface slopes from 0m at electrode 1 to 4.8m at electrode 25 (see Figure 10.7). The file **R2.in** is now different for the forward and inverse model runs through the addition of topography data. Figure 10.8 shows the inverse solution for this case.

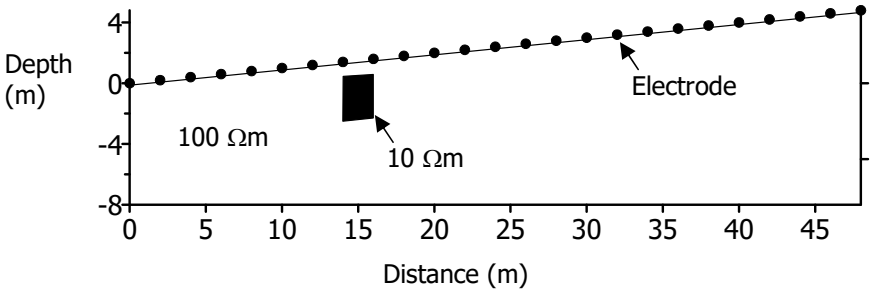


Figure 10.7: Definition of synthetic model for surface array 2 problem

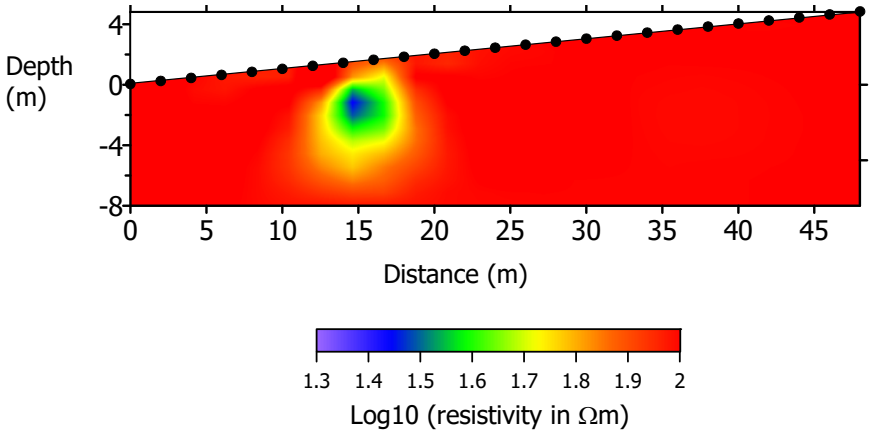


Figure 10.8: Inverse model for surface array 2 problem with dipole-dipole array

Surface electrode array 3 – fixing parameter values

Occasionally it is useful to fix resistivity values within the mesh. This is particularly useful for time lapse imaging where we wish to focus on changes within a particular part of the mesh. In ***R2*** this can be achieved with a quadrilateral mesh by defining left, right, upper and lower limits of the parameter zone (see Line 16 and line 17 definitions for **R2.in**). To illustrate this we invert data from a previous example but constrain the parameter zone to a smaller region of the mesh.

The subfolder "Examples/Surface_3/dpdp" contains an example similar to the Surface array 1 example but this time the inverse model is defined so that not all elements are parameters. The forward model used for generating the data file (protocol.dat) is that from Figure 10.1, i.e. in "Examples/Surface_1/dpdp/Forward".

For this example we use a patch_size of zero in the x and z directions and then define the location of the zone to be parameterised. In **R2.in** a patch of 4 elements per parameter is defined in the horizontal direction starting at element 1. All elements are grouped into parameters in the horizontal and thus there are 56 patches of 4 elements declared (a total of 224 elements). In the vertical we define the parameter zone to be from 1m to 6m depth and a patch size of 2 elements is used (10 parameters in total corresponding to the 20 elements). The starting element for the parameterisation in the vertical is 5 (since the first four elements cover the first 1m in this case). Note that the resistivity of any element that is not declared to contribute to a parameter remains unchanged from the starting value (in this case 100 Ωm).

Figure 10.9 shows the resultant inverse model. The sharp boundaries (in the vertical) at 1m and 6m are a result of the constrained parameter zone (there is no smoothing over the boundaries).

Note that if you restrict the parameter zone too much then convergence of the solution may be problematic (since you will be reducing the degrees of freedom of the inverse solution).

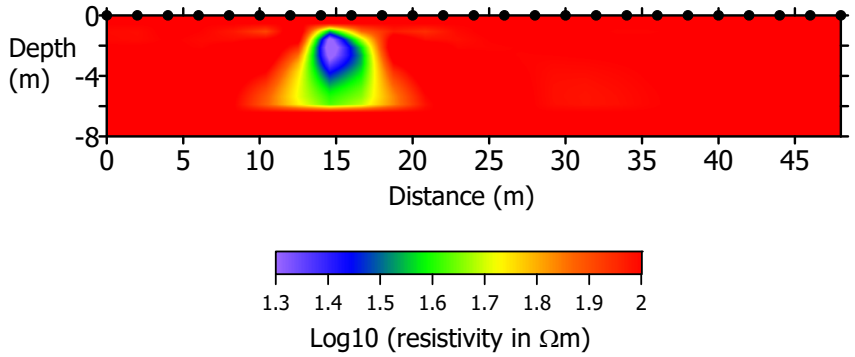


Figure 10.9: Inverse model for surface array 3 problem

Surface electrode array 4 – anisotropic smoothing

The subfolder "Examples/Surface_4/dpdp" contains an example similar to the Surface array 1 example but this time the smoothing is set to be anisotropic. For this case the smoothing is exaggerated in the vertical by setting alpha_aniso in Line 22 of **R2.in** to 0.1 (10 times more smoothing in the vertical). Figure 10.10 shows the resultant inversion (c.f. Figure 2 with isotropic smoothing).

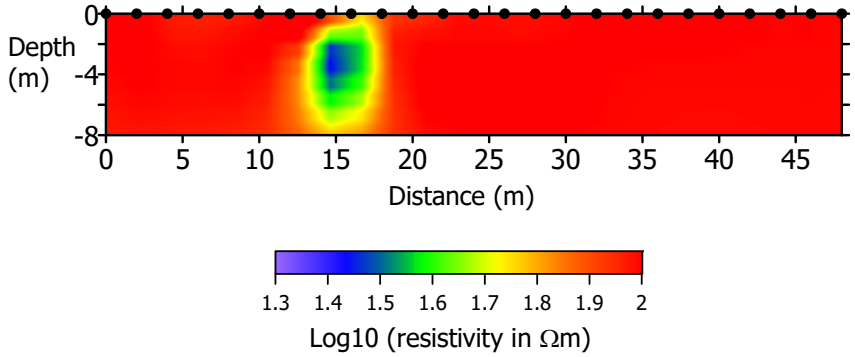


Figure 10.10: Inverse model for surface array 4 problem

Surface electrode array 5 – more complex meshing

The quadrilateral mesh examples so far have used a simple mesh definition. For the surface array 2 example the mesh was distorted by changing the topography of the upper surface of the mesh. In this example we illustrate how to change the mesh in a more flexible manner. By setting mesh_type to 5 in Line 2 of **R2.in** we can specify the row coordinates for every column of the mesh. This requires more input information than the previous examples but gives much greater flexibility.

The subfolder "Examples/Surface_5/dpdp/Forward" contains a forward modelling example similar to the Surface array 1 example but in this case a zone of low resistivity lies just below the ground surface and varies in thickness from 0.5m at electrode 1 to 1m at electrode 25. In addition, the electrodes are located in this example at the bottom of this conductive zone (see Figure 10.11). Such a model may be representative of electrical imaging using electrodes placed at the bed of a stream (the conductive zone representing the stream).

To setup this forward model the 49 row coordinates are defined for all 225 column positions. In addition, **R2.in** must also contain the definition of more groups of elements than before to represent the conductive zone (remember that the zones are defined as groups of congruent elements and since the elements are numbered in the vertical then we must define 224 such groups for this problem, in addition to the rest of the region, i.e. 225 groups in all).

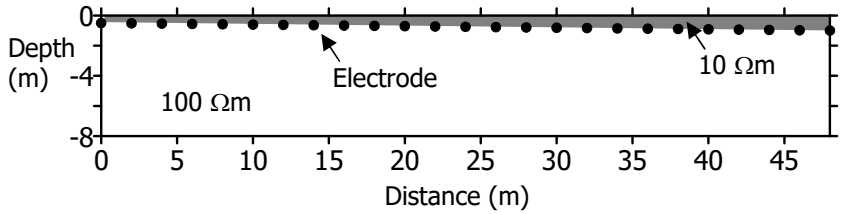


Figure 10.11: Forward model for surface array 5 problem

Surface electrode array 6 – depth of investigation

The subfolder "Examples/Surface_6" contains three examples illustrating the use of a reference resistivity model. The examples use the dipole-dipole forward model from Surface electrode array 1 but invert three difference cases with reg_mode set to 1 (see line 21 of **R2.in**).

In case_01 we set α_s to 10 with a resistivity background (starting model or reference model) equal to a uniform ρ_{back}=100Ωm. The results are shown in Figure 10.12a. The result is similar to that shown in Figure 10.2 (no regularisation relative to a reference model).

In case_02 we set α_s to 50 with a uniform ρ_{back}=100Ωm. The results are shown in Figure 10.12b. The target recovery is now weaker as the inversion applies more penalty to deviation from 100Ωm.

In case_03 we set α_s to 10 with a uniform ρ_{back}=50Ωm. The results are shown in Figure 10.12c. Recalling that the background resistivity in the forward model is 100Ωm, Figure 13c illustrates the zone over which the measurements have sensitivity – the lower left and right regions are clearly not influenced by the measurements in this case (which is consistent with the resolution matrix in Figure 10.4).

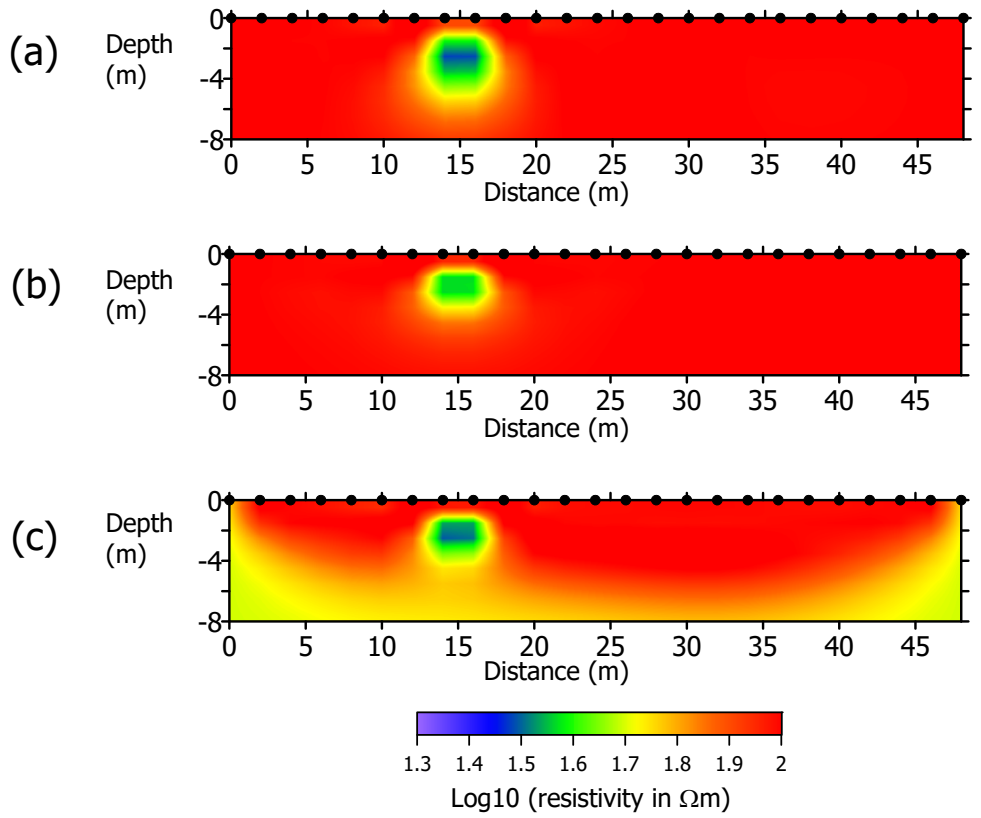


Figure 10.12: Surface array 6 – regularising relative to a reference resistivity model. (a) α_s=10, ρ_{back}=100Ωm. (b) α_s=50, ρ_{back}=100Ωm. (c) α_s=10, ρ_{back}=50Ωm.

We can use these results to assess the depth of investigation (DoI), following the method of Oldenburg and Li (1999). We can compute the value:

$$R(x,y) = \frac{m_1(x,y) - m_2(x,y)}{m_{1,r} - m_{2,r}}$$

Where m_1 are the inversion results in Figure 12a (in log units) using 100Ωm as a reference and m_2 are the inversion results in Figure 10.12c (in log units) using 50Ωm then, $m_{1,r} = \log_{10}(100)$ and $m_{2,r} = \log_{10}(50)$. Figure 10.13 shows the variation of R. Oldenburg and Li (1999) suggest a reasonable value of R = 0.1 or 0.2 as a suitable depth of investigation. Figure 10.13 shows a contour of R = 0.2 to illustrate this.

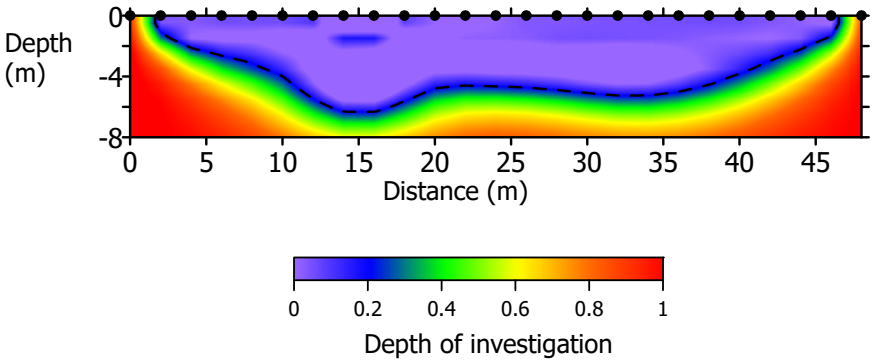


Figure 10.13: Depth of investigation for Surface array 6 problem.

Surface electrode array 7 – time-lapse (difference) inversion

The subfolder "Examples/Surface_7" contains an example illustrating the use of a difference inversion, which may be useful for time-lapse (monitoring) studies. Here we use two forward models as datasets representing changes in resistivity from time 0 to time 1 (see Figure 10.14).

"Examples\Surface_7\dpdp\Forward_t0" contains input files for running a forward model at time 0 and "Examples\Surface_7\dpdp\Forward_t1" contains input files for running a forward model at time 1. In each case ***R2*** produces the file **R2_forward.dat**, which contains the transfer resistances due to the resistivity structure defined. The two **R2_forward.dat** files will be used to create an input data file for a difference inversion.

If we select a difference inversion the data file for inversion (**protocol.dat**) contains two columns of data, as defined earlier in this document. The first column is the measured data (here, at time 1) and the second column is the reference dataset (here, at time 0). The folder "Examples\Surface_7\dpdp\Inverse_difference" contains input files for the difference inversion.

For a difference inversion the starting model for the inversion (which is often just a homogenous model for normal inversions) is the resistivity that is consistent with the reference dataset. For this synthetic example we need to determine this model by inverting the forward model at time 0. This is equivalent to Surface electrode array 1, but note that we must save the entire resistivity model, not just the region of interest. ***Start_resis.dat*** is resulting inversion of the forward model from time 0.

Figure 10.15 shows the results (in **f001_res.vtk**) plotted as log resistivity, using ParaView. The electrode locations (from file **electrodes.vtk**) are also shown.

When running a difference inversion an output file **f001_diffres.dat** is produced (in addition to the normal inverse output files). **f001_diffres.dat** provides values of percentage change (from the starting model) in resistivity. The file **f001_res.vtk** also contains this information and is illustrated in Figure 10.16.

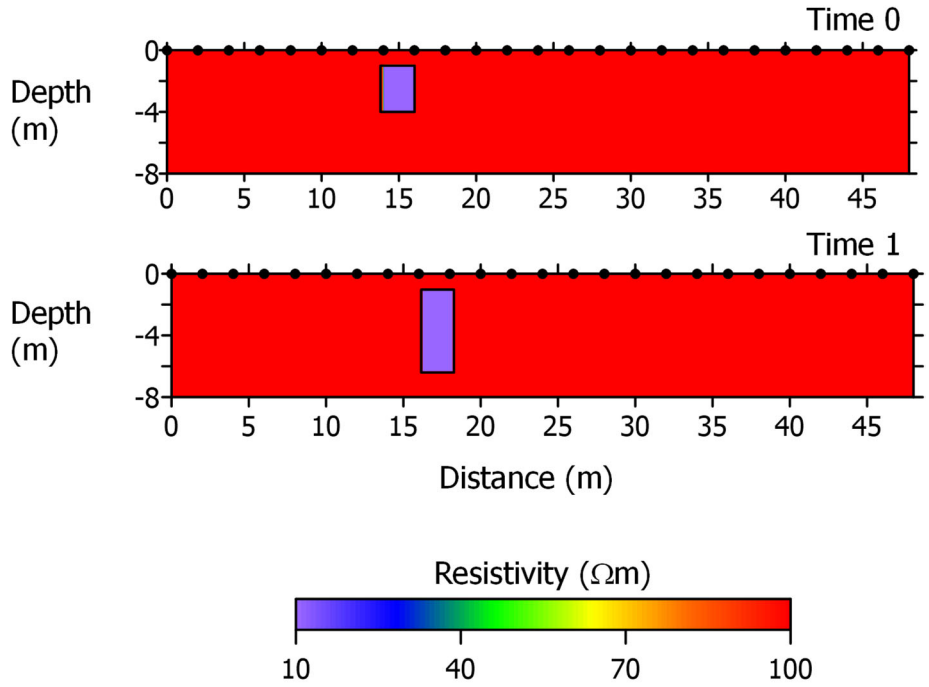


Figure 10.14. Forward model definitions for difference inversion example.

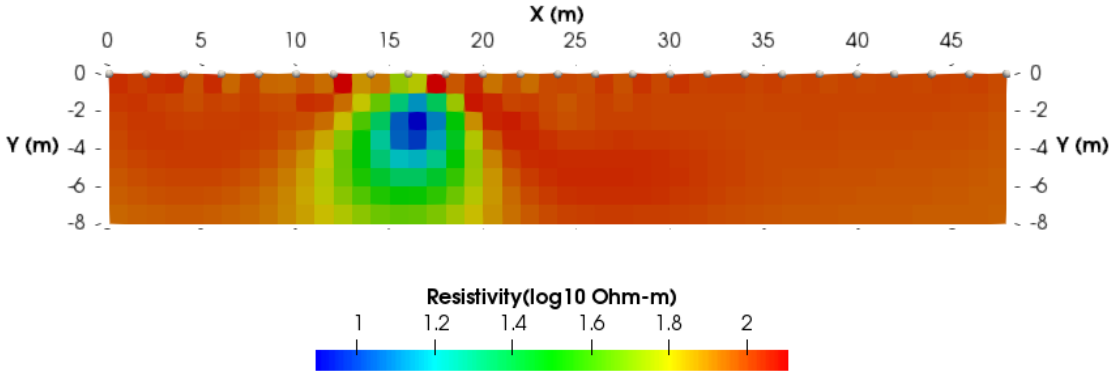


Figure 10.15. Resistivity model obtained from difference inversion (plotted in ***ParaView***).

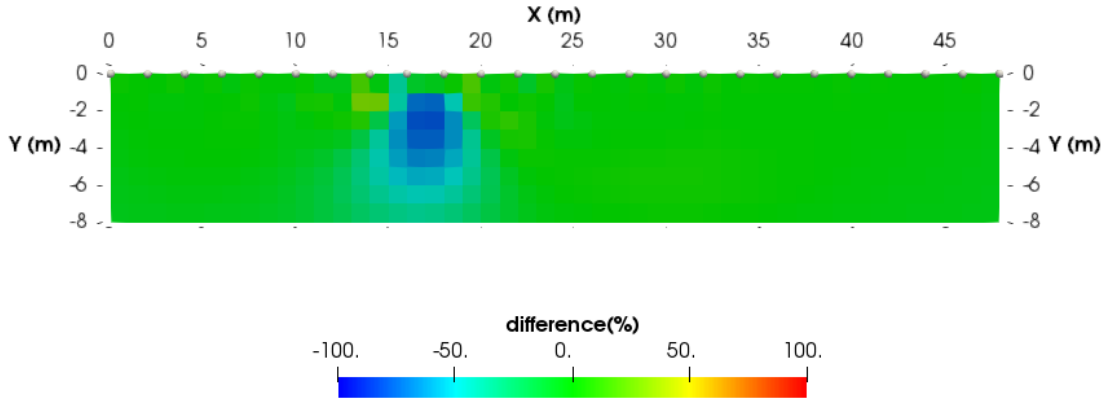


Figure 10.16. Change in resistivity from difference inversion (plotted in ParaView).

Surface electrode array 8 – triangular meshing

The folder Examples/Surface_8/ contains input files for running a forward and inverse problems for the dipole-dipole survey (from Surface electrode array 1) using a triangular mesh. The mesh is defined in **mesh.dat**. It contains 4,204 elements and 2,160 nodes. The region modelled extends approximately 200m to the left and right of the electrode array, and approximately 200m beyond the zone of investigation.

The folder Examples/Surface_8/Forward contains the input files for a forward model. In this mesh the first 40 elements represent the 10Ωm: in **R2.in** the two regions are defined. Figure 10.17 shows a plot of the forward model definition using ***ParaView***. Note that the region extracted for plotting is based on the position of the centroid of elements and consequently a 'jagged' boundary often exists for triangular mesh output.

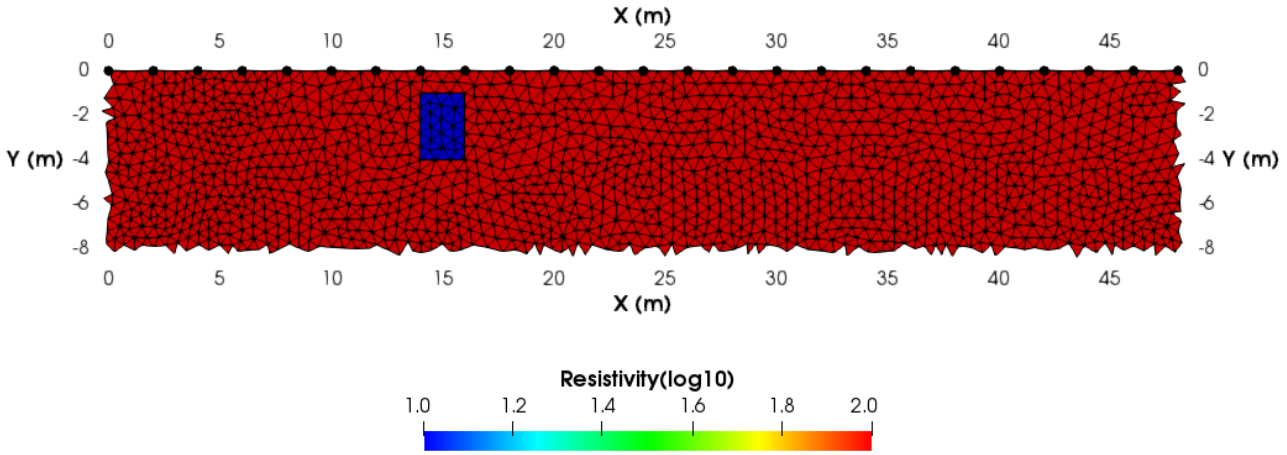


Figure 10.17: Definition of forward model using a triangular mesh

The folder Examples/Surface_8/Inverse_1 contains the input files for an inversion of the dipole-dipole data using a triangular mesh. Figure 10.18 shows the result, plotted ***in ParaView***.

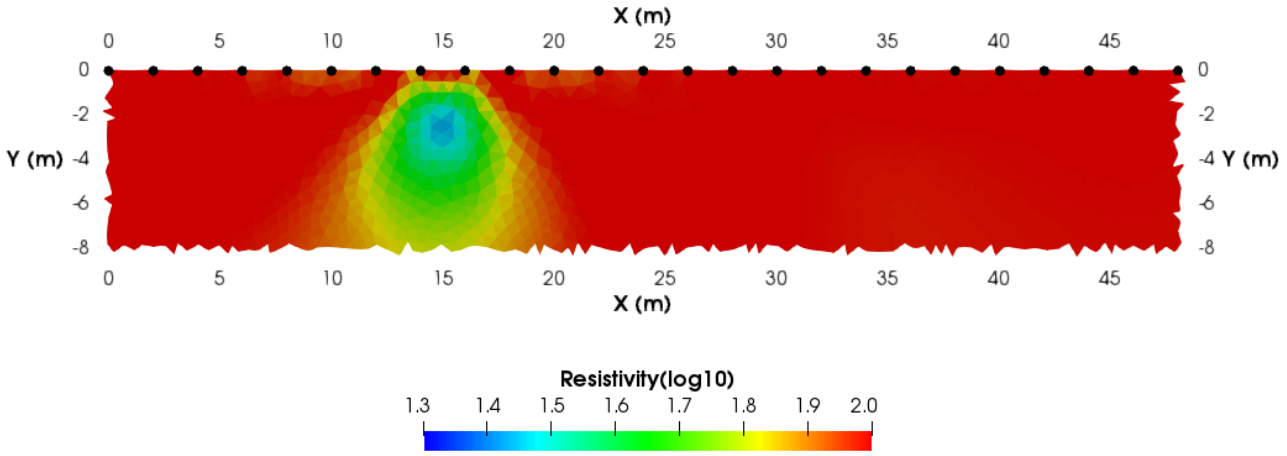


Figure 10.18: Inversion of dipole-dipole data.

The folder Examples/Surface_8/Inverse_2 contains the input files for an inversion of the same data using anisotropic regularisation to minimise lateral smoothing (see Figure 10.19).

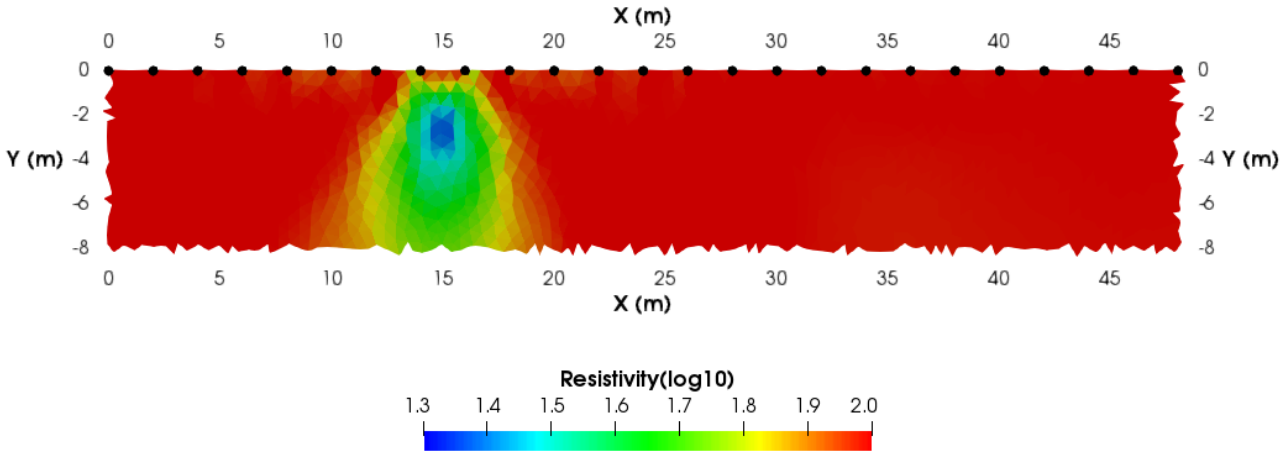


Figure 10.19: Inversion of dipole-dipole data with enhanced vertical smoothing.

The folder Examples/Surface_8/Inverse_3 contains the input files for an inversion of the same data as above but in this example the inverse region is blocked into two zones: one representing the low resistivity zone in Figure 10.17 and the other representing the remainder of the mesh. The inversion is show in Figure 10.20. Note that although there is variation within each zone (since the same number of parameters exists), by defining the boundaries a near-perfect result is achieved (although this is a somewhat artificial case).

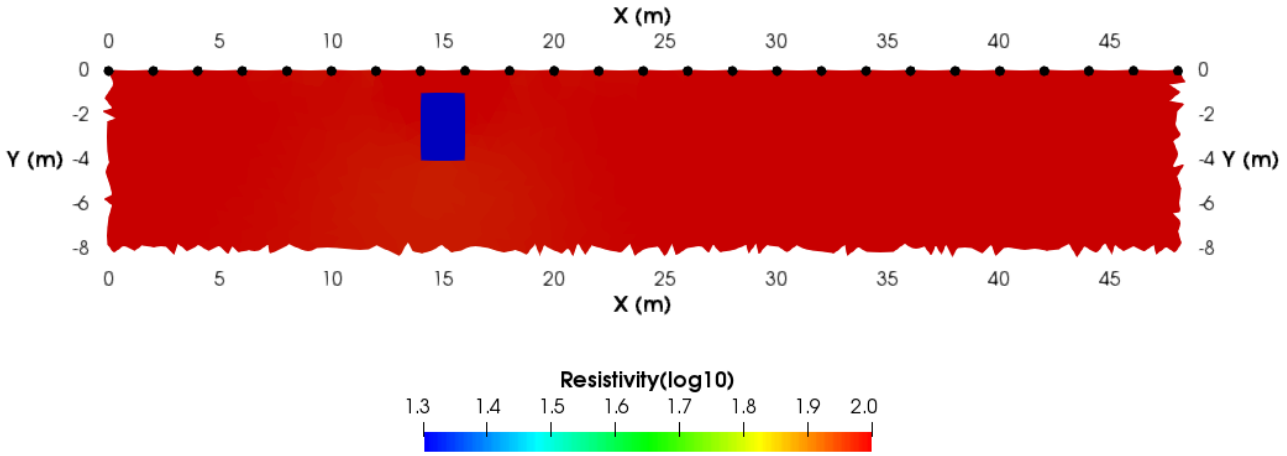


Figure 10.20: Inversion of dipole-dipole data with region blocking.

The folder Examples/Surface_8/Inverse_4 contains input files to illustrate the effect of fixing resistivity using a triangular mesh. In this case the starting resistivity is set to 20 Ωm and 40 elements that occupy the area where the low resistivity feature exists are removed from the parameterisation by setting their parameter number to zero in **mesh.dat**. Note that to do this we have to move this block of elements in **mesh.dat** to the end of the list of elements (compare Examples/Surface_8/Inverse_4/**mesh.dat** with Examples/Surface_8/Inverse_3/**mesh.dat**). Since we have assigned parameter number zero to these elements then the resistivity remains fixed to the starting model (20 Ωm) but all other elements can change in the inversion. Figure 10.21 shows the resulting inversion. Note that the true target is 10 Ωm and so by forcing the region to be 20 Ωm the adjacent elements are affected as the inversion compensates for the difference. Figure 10.22 shows the same inversion result but on a wider colour scale to illustrate other artefacts that result from forcing an incorrect value in the low resistivity zone.

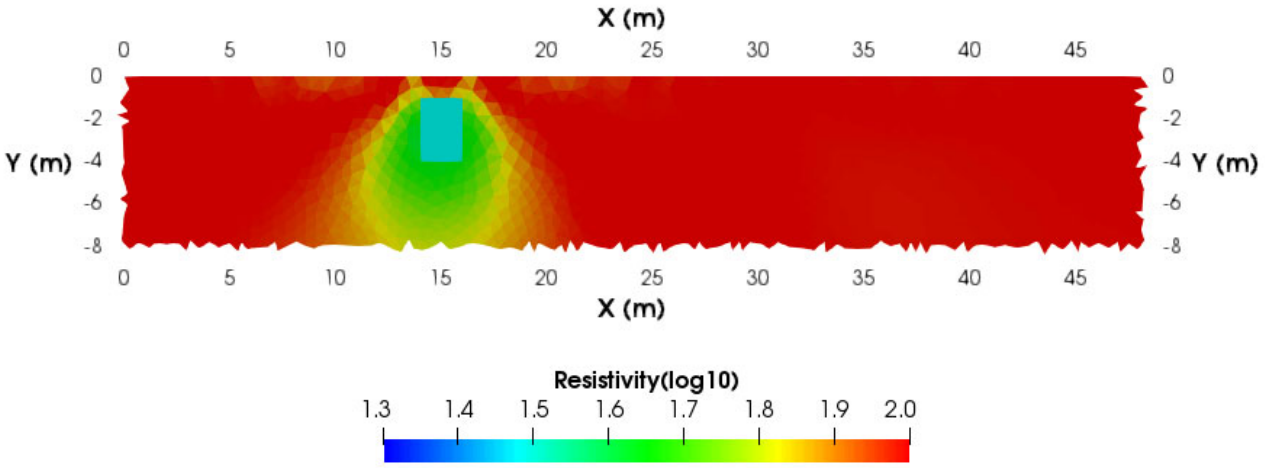


Figure 10.21: Inversion of dipole-dipole data with 40 elements (where the 'target' is located) set to 20 Ωm (the true value is 10 Ωm and so some smearing around the 'target zone' exists).

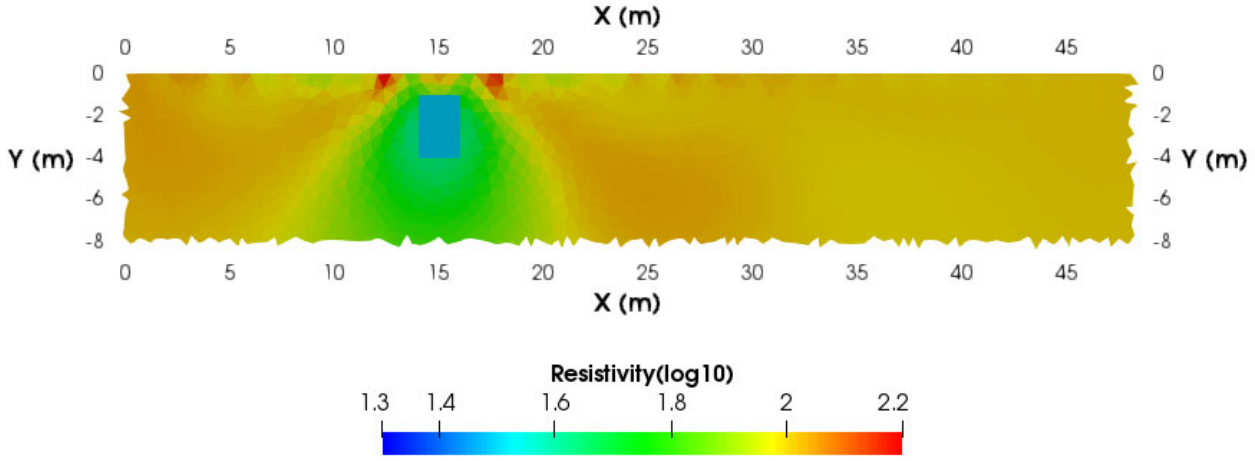


Figure 10.22: As Figure 10.21 but with extended colour scale to show artefacts resulting from forcing the resistivity in the 'target zone' to be different from the true value.

Surface electrode array 9 – quadrilateral mesh in mesh.dat

The folder Examples/Surface_9/ contains input files for running a forward and inverse problem when a quadrilateral mesh is read from **mesh.dat** (i.e. mesh type = 6). The example used here is the same setup as Surface electrode array 1 using a dipole-dipole configuration, but rather than defining a mesh in **R2.in**, it is read from the file **mesh.dat**. Figure 10.23 shows the inverse model. Clearly for this problem the use of a separate mesh file is unnecessary, but it allows the user to understand the way in which a pre-defined quadrilateral mesh can be used.

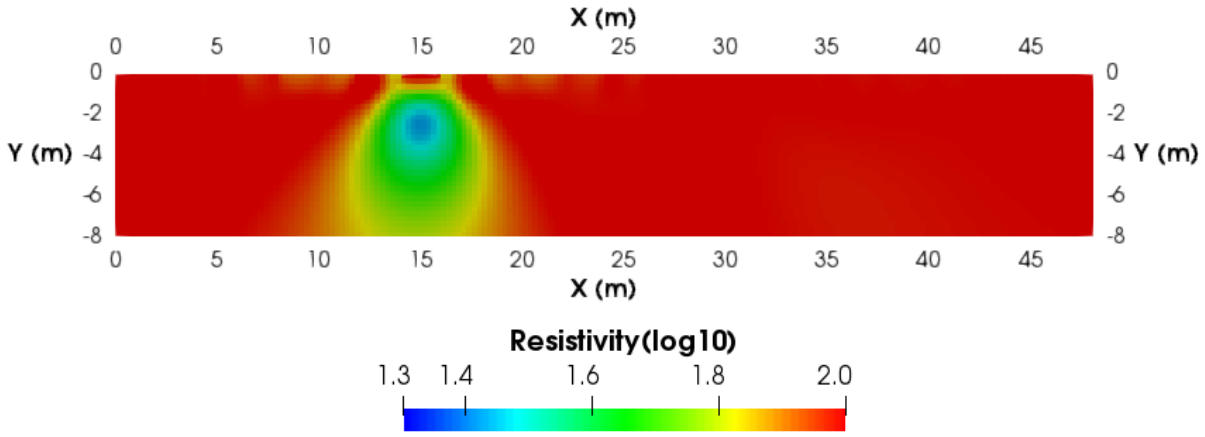


Figure 10.23: Inversion of dipole-dipole surface electrode data using quadrilateral mesh read from **mesh.dat**

Cross borehole array

The subfolder "Examples\Xbh" contains forward and inverse models for two cross borehole examples illustrated in Binley and Kemna (2005). The first case considered here is included in "Examples\Xbh\8m_skip7". In this case measurements are made between two boreholes 8m apart, as illustrated in Figure 10.24. As in previous examples a zone with resistivity 10 Ωm is embedded in the 100 Ωm half space. The measurement scheme used is a "skip 7": dipole – dipole with 7 electrode in between each current and potential electrode pair (see the **protocol.dat** file).

The forward model input files are included in "Examples\Xbh\8m_skip7\forward" and the inverse model files are in "Examples\Xbh\8m_skip7\inverse". Figure 10.25 shows the output of the inverse solution using the forward model as "data".

The second cross borehole case is for two boreholes 15m apart, as illustrated in Figure 10.26. The forward model input files are included in "Examples\Xbh\15m_skip7\forward" and the inverse model files are in "Examples\Xbh\15m_skip7\inverse". Figure 10.27 shows the output of the inverse solution using the forward model as "data". The effect of increased spacing of the boreholes on sensitivity of the measurements can be seen by comparing Figures 10.27 and 10.25.

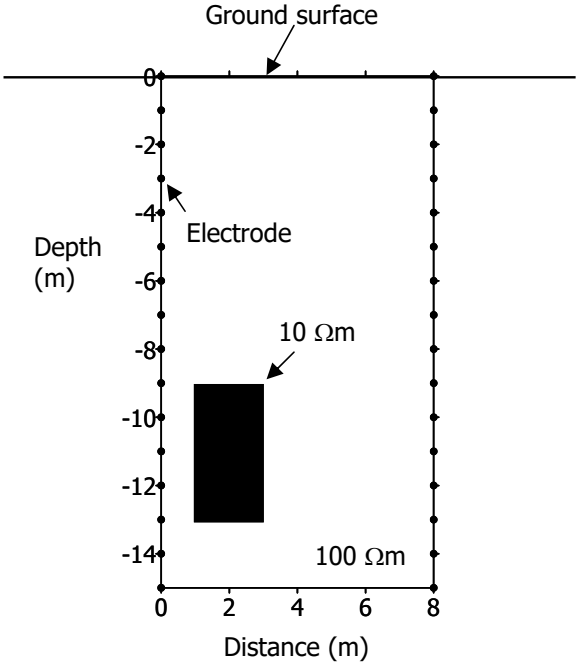


Figure 10.24: Forward model definition for cross borehole case 1

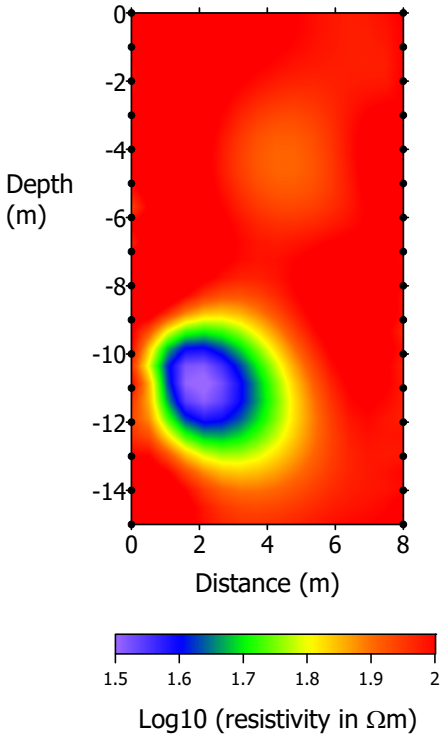


Figure 10.25: Inverse model for cross borehole case 1

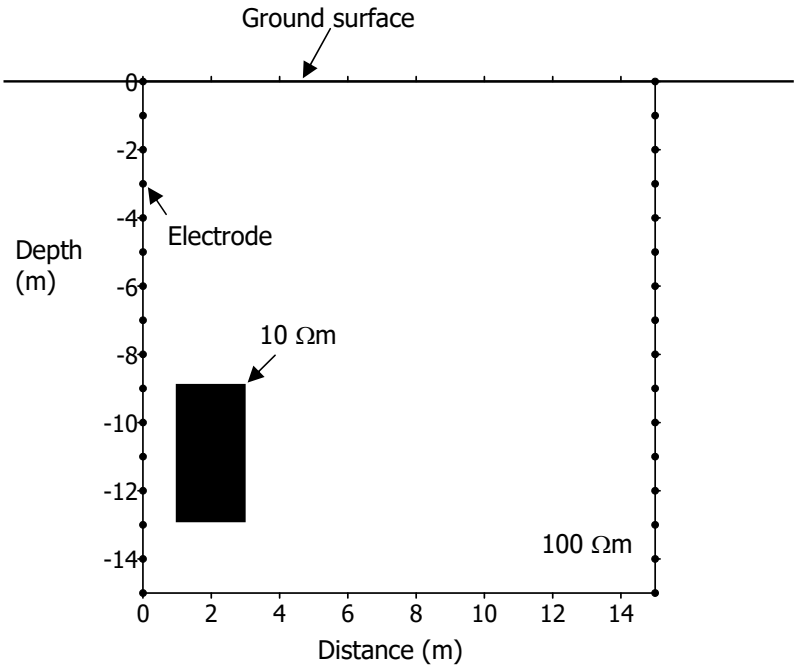


Figure 10.26: Forward model definition for cross borehole case 2

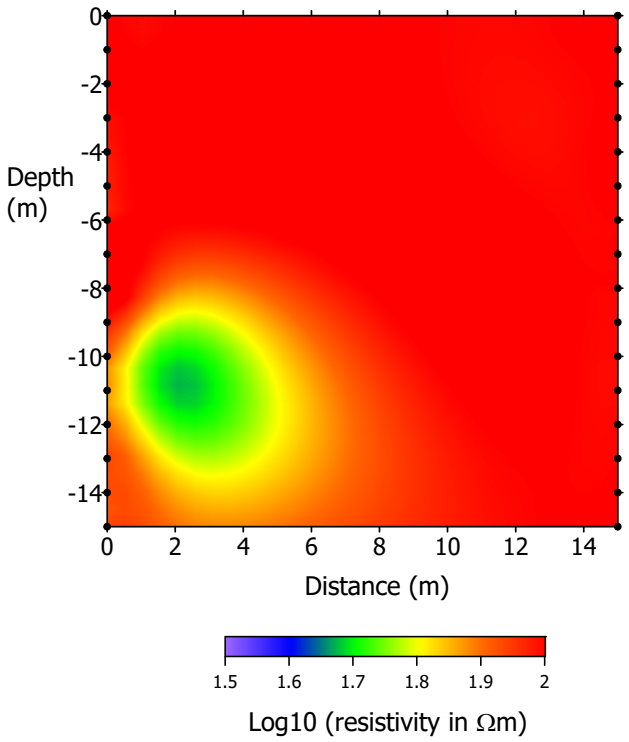


Figure 10.27: Inverse model for cross borehole case 2

Column model

The subfolder "Examples\Column" contains forward and inverse models for a simple circular arrangement of electrodes. In this example 16 electrodes are placed around the perimeter of a circular region (representing, for example, a soil column). In the forward model (see Figure 10.28) a 500Ωm circular target is placed in the 100Ωm vessel. The mesh in Figure 10.28 was used to compute the response of 103 measurements in a dipole-dipole configuration. The generated transfer resistance were then perturbed with noise with zero mean and a standard deviation related to the transfer resistance, R according to:

$$\sigma(R) = \sqrt{a^2 + b^2 R^2},$$

with $a = 0.001\Omega$ and $b = 0.02$.

These data were then used for the inversion with a_wgt set to 0.001 and b_wgt set to 0.02. Note that to avoid any bias in the modelling a different finite element mesh was used for the inversion, since the forward modelling mesh has a higher density of elements in the region of the anomaly in order to correctly place the anomaly in the mesh. In the inverse model mesh a more uniform triangular mesh is used. Figure 10.28 shows the inverse model for the problem.

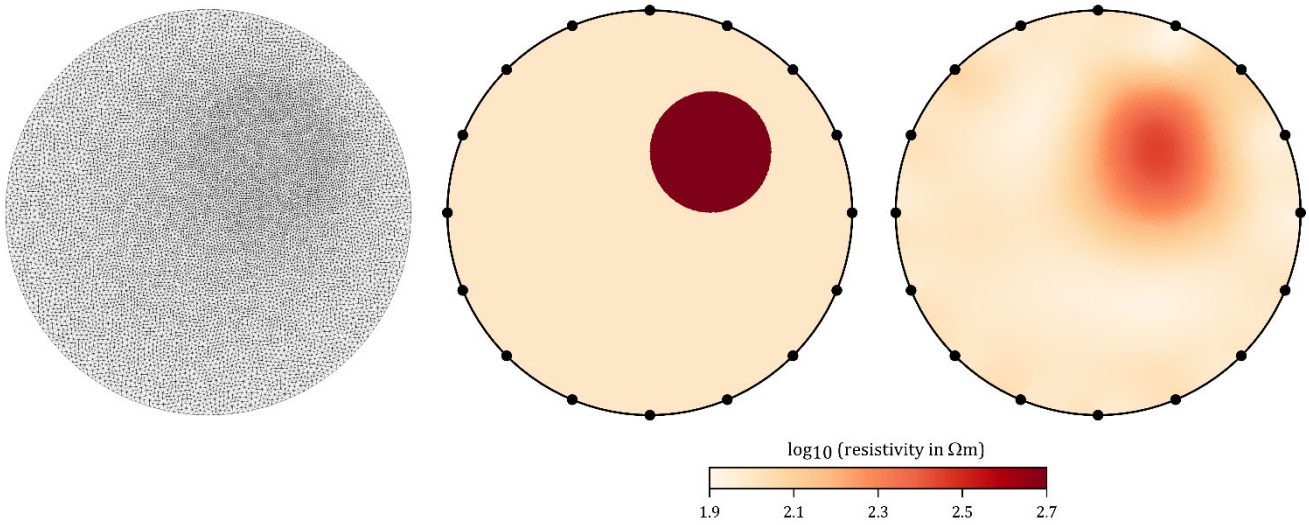


Figure 10.28: Column model example. Left hand image shows finite element mesh used for forward model. Centre image is the forward model. Right image is the inverse model.

11. **Additional codes and scripts**

The ***R2*** package contains a number of codes and scripts that may help the user in creating input files and visualising output from ***R2***. Note that I have not tested any of these except the first set of mesh utilities. Some of these scripts may not be compatible with current versions of ***R2*** and Gmsh.

Meshing

Folder **Mesh_Utilities/Binley**

GenGmshGeo2D.exe
Creates a Gmsh geo (geometry file) which may be used for creating a triangular mesh. Note that the geo file can be edited before meshing, e.g. to add topography. The geo file created is **GenGmshGeo2D.geo**

GmshMsh2R2.exe
Reads the **GenGmshGeo2D.geo** created (and, perhaps, edited) in Gmsh, which can be meshed, creating a msh (mesh file) **mesh.dat.** Note that the first line will contain the number of elements, which is needed in **R2.in** for defining the resistivity (starting model or forward model definition).

Gen_R2in.exe
*R*eads a **ProfileR.in** file (see www.es.lancs.ac.uk/people/amb/freeware/profiler/profiler.htm) and creates an **R2.in** file for an equivalent mesh. This is useful for users moving from the simpler inverse code ***ProfileR*** to ***R2***.

Folder **Mesh_Utilities/Boyd**

gmsh2R2msh
This python code and executable (written by Jimmy Boyd (British Geological Survey/Lancaster University) with convert a Gmsh msh file to an ***R2*** **mesh.dat** file.

Folder **Mesh_Utilities/Claes**

create_mesh
A Matlab script written by Niels Claes (Wyoming University) will create an ***R2*** **mesh.dat** file for a triangular mesh.

Plotting

Folder **Plot_Utilities/Claes**

plot_resistivity
This folder contains a Matlab code for plotting output from ***R2***, kindly provided by Niels Claes (University of Wyoming).

Folder **Plot_Utilities/Tso**

plot_resistivity
This folder contains a Matlab code for plotting output from ***R2***, kindly provided by Michael Tso (Lancaster University). The code will plot different levels of transparency depending on the sensitivity map.

Complete wrapper (files not included but can be downloaded from links)

pyres
Kevin Befus (University of Wyoming) has produced a python wrapper for ***R2*** – see Befus (2017) and https://github.com/kbefus/pyres.

ResIPy GUI
Guillaume Blanchy (Lancaster), Sina Saneiyan (Rutgers) , Jimmy Boyd (BGS/Lancaster), Paul McLachlan (Lancaster/BGS) have developed an open source python GUI for ***R2*** and sister codes. See https://gitlab.com/hkex/pyr2 which also includes links for standalone executables. More information is also available at https://www.researchgate.net/project/ResIPy-GUI-for-R2-family-codes

12. **Getting started with ParaView**

The vtk files created by ***R2*** have been structured to work in ***ParaView*** - an open source visualisation application that can be downloaded from https://www.paraview.org/ Users are advised to study online tutorials on ***ParaView***. Here are a few simple tips to help the user get going.

The **fXXX_res.vtk** file contains the inverted resistivity, log transformed resistivity, a sensitivity map or resolution matrix (if selected) and a difference inversion (if selected). The file also contains the finite element mesh structure.

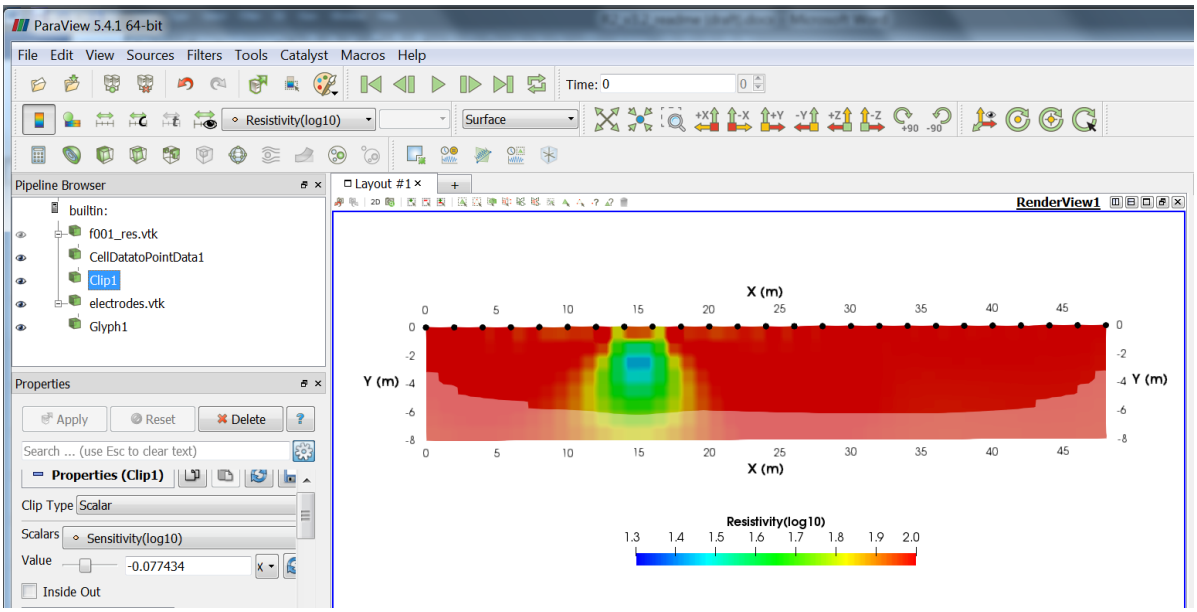
Open a **fXXX_res.vtk** file in ***ParaView***, click Apply under Properties and you will get a map of the resistivity. Under Coloring in Properties you can select one of other variables stored, e.g. log10resistivity.

The default Representation of the image is Surface. Change to Surface with edges to see the finite element mesh and the resistivity image.

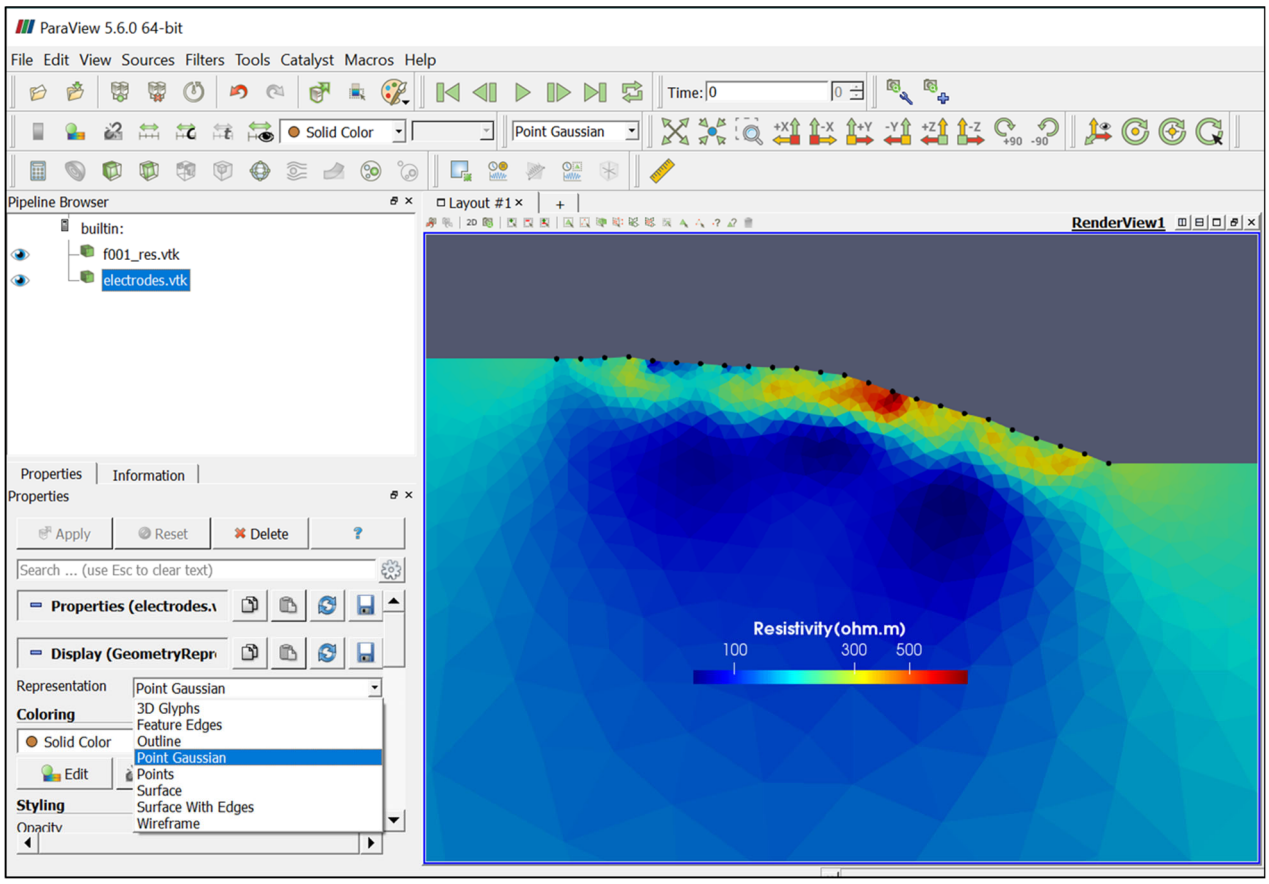
Axis labels and the colour legend are easily changed to suit the user.

To get an interpolated image (rather than one that shows element by element) then highlight the vtk file in the Pipeline Browser and select the Cell Data to Point Data Filter. Then select Apply in Properties.

If you want to show an image with thresholded values based on the sensitivity map then select an Opacity of 0.6 under Styling in Properties. Now select the Clip Filter. Select Clip Type as Scalar under Properties and select Sensitivity(log10) as Scalars. Select a mid-range value in the slider bar for Value. Under Coloring select Resistivity(log10). You should now see the thresholded region as a solid colour and the outer area as opaque (see figure below).



To show the electrodes, open the **electrodes.vtk** file and click Apply under Properties. Now select the Glyph filter and under Glyph type select Sphere. Make the radius of the sphere smaller than the default, say 0.05, and select Apply. You can change the colour of the glyph in the Properties box. If not all electrodes are displayed then select Masking under Properties and select Glyph mode to All Points. I have noticed in recent versions of Paraview that Glyph plotting sometimes does not show all the points. This appears to be a bug in recent Windows versions of Paraview. If this occurs then you can show electrodes positions by simply selecting the properties type of **electrodes.vtk** in ***Paraview*** to Point Gaussian (see example below).


ParaView 5.6.0 64-bit
File Edit View Sources Filters Tools Catalyst Macros Help
Time: 0
Solid Color
Point Gaussian
Pipeline Browser
builtin:
f001_res.vtk
electrodes.vtk
Layout #1
RenderView1
Properties
Information
Properties
Apply
Reset
Delete
Search ... (use Esc to clear text)
Properties (electrodes.
Display (GeometryRepr
Representation
Point Gaussian
Coloring
Solid Color
Edit
Styling
Opacity
3D Glyphs
Feature Edges
Outline
Point Gaussian
Points
Surface
Surface With Edges
Wireframe
Resistivity (ohm.m)
100
300
500

13. **Common User Errors**

Below is a list of some common user errors that I have encountered. This may be useful for new users.

A common mistake is for a new user to go straight into trying to run an inverse solution without getting a good feeling for the model that is being used. New (and old) users working on new problems should first try run a forward model for a uniform resistivity. This will help sort out any problems with the definition of the mesh, etc. It will also be useful in understanding the quality of the forward model and help judge this against the quality of the data.

If you can, run the code from the command line. You will need to run CMD in Windows, then move to the correct folder and then type ***R2.*** Doing it like this help see any errors if the program crashes unexpectedly because of incorrect input.

In the example input files provided there are comments at the end of most lines in the form "<< comment" . Note that these are always at the end of a line. You cannot have these appearing on their own in a line. If you do then ***R2 =***will try read this comment when it is expecting numerical input and simply crash.

The mesh is based on elements and nodes. In **R2.in** Lines 3 to 9 are based on nodes, whereas Line 13 is based on elements. It is important to understand the difference and not mix the two.

On Line 22 in **R2.in**, specifying a tolerance of 1.0 means that you are happy that you have estimated your errors correctly (Line 23). Don't just use the a_wgt and b_wgt values in the example files – spend time to understand the likely errors in your measurements and model.

Setting data_type to 1 in Line 22 of **R2.in** means that the data you input will be log transformed in ***R2***, not that you have to supply logged data. You must specify the polarity of the data, as always.

Setting the minimum and maximum apparent resistivity (Line 23 of **R2.in**) is only valid if you have a flat surface and an infinite half space problem, otherwise the geometric factors that ***R2*** will compute will be incorrect.

For a structured quadrilateral mesh the electrode positions are defined by their column and row positions in the mesh (Line 27 of **R2.in**). These are not the co-ordinates of the electrodes but their position in the mesh.

In the definition of the input files, each line has been defined in terms of the type of numbers that are required. For example, (Real, 2 Int, 2 Real) means one real number, followed by two integers, followed by two reals. You can substitute integers for reals but not the other way round. So if the code is expecting an integer and your line entry has 1.3, for example, then the code will crash.

Note that the data in **protocol.dat** should be provided in transfer resistances, NOT apparent resistivities. Also note that the polarity should be retained in the data. It is very wise to check the polarity of your measurements – you can do this by computing the geometric factor for your measurement configuration (provided topographic and non-infinite boundaries are not significant). If you don't know how to compute the geometric factors then you should run a forward model with ***R2*** for a uniform half space and compare the computed polarities with those in your data. For a surface electrode array your data should be the same polarity as the model, otherwise the measurements will not be included in the inversion. For electrodes not on the surface the polarity can change as the resistivity structure changes in the inversion.

Make sure you check that the solution has converged in inverse mode (see **R2.out**). Just because a resistivity model has been computed it does not mean that convergence has been reached. If the solution has not converged then go through the **fXXX_err.dat** file and look at see if any particular

measurements are problematic. Also check that you are confident with the a_wgt and b_wgt error settings you have applied (Line 23, **R2.in**). A common mistake is to set these too low. A good estimate of b_wgt, in particular, is important. Normally, you should be able to get convergence in less than 5 iterations. It is not wise to increase the maximum number of iterations to a large number. If you don't get convergence in 10 iterations then there is definitely some problem with the data, the assumptions or the input files.

14. **References**

Befus, K.M. , 2017, pyres: A Python Wrapper for Electrical Resistivity Modeling with R2, J. Geophys. Eng., doi: 10.1088/1742-2140/aa93ad

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

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

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

Boyd, J., Blanchy, G., Saneiyan, S., Binley, A., 2019. 3D geoelectrical problems with ResIPy, an open-source graphical user interface for geoeletrical data processing. FastTIMES 24.

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

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

Lowry, T., M.B. Allen, and P.N. Shive, 1989, Singularity removal: A refinement of resistivity modeling techniques, Geophysics, 54, 766–774.

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

Oldenburg, D. and Y. Li, 1999, Estimating depth of investigation in dc resistivity and IP surveys, Geophysics, 64(2), 403-416.

---

*If you make use of **R2** then please contact the author (a.binley@lancaster.ac.uk) so that you can be added to a mailing list for future updates, fixes, etc.*

For more information, including example files contact:

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

Lancaster University
