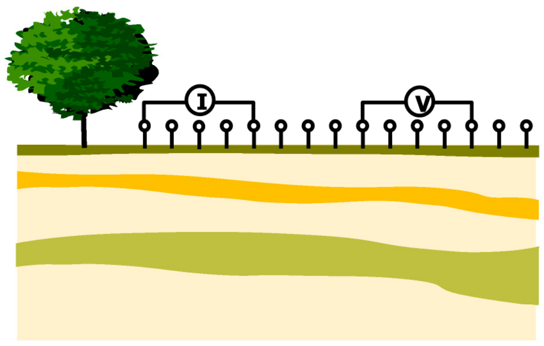
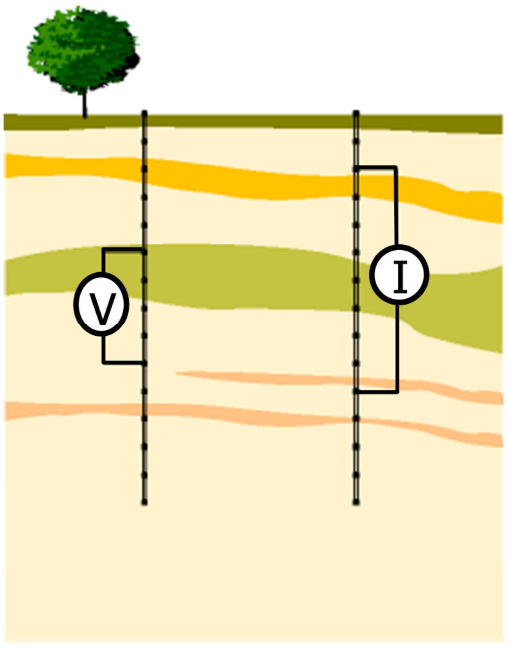
R2

version 3.1 (June 2016)

Andrew Binley
Lancaster University
June, 2016





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.

Changes to *R2* from v2.5

Now outputs results (resistivity, log10 resistivity, sensitivity map, electrode co-ordinates) in vtk format, allowing easy visualisation with ParaView (which can be downloaded from http://www.paraview.org/paraview/resources/software.html). See also http://www.vtk.org/Wiki/The_ParaView_Tutorial for a tutorial on ParaView.

This readme now contains a list of common user errors, which may be helpful for new users.

Computer requirements for *R2* v3.1

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. Linux users should be able to run *R2* with the command "wine R2.exe" (thanks to Rodolphe Cattin for this tip). I have not tested this, however. Users requiring a version compiled for other processors should contact the author.

Introduction to *R2* v3.1

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 (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 is a forward/inverse solution for 3D or 2D current flow in a quadrilateral or triangular mesh. ***R2*** requires at least two data files: **R2.in** and **protocol.dat.** If a triangular 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 measurement

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 10 to 20 times the distance). 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.

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.

Triangular meshes are usually created with the vertical co-ordinate representing elevation, not depth. And so when a triangular mesh is used, the sign change is not made in any of the output files listed above.

Input specification for *R2* v3.1

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.

- **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, y, resis, log10(resis), where x,y 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 Surfer.

- **f001_err.dat** will contain nine columns. In the first column is the normalised data misfit, the second column contains the observed data recorded as an apparent resistivity, the third column contains the equivalent apparent resistivities for the computed model, the fourth column shows the original data weight (i.e. data standard deviation in same units as data), the fifth column is the final data weight, the fifth columns shows a "1" if any weights have been changed during the inversion, otherwise a "0" will appear, the sixth to ninth columns show the electrode numbers.

- 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] 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.

- **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.

- 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, y, diffresis, where x,y 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,y.

- **electrodes.vtk** contains the co-ordinates of the electrodes in vtk format. The values are in three columns: x,y,z (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 you select the "Glyph" icon; this allows you to plot the electrodes as small spheres (or other objects).

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 or 4 for a regular quadrilateral mesh or 5 for a more generalised quadrilateral mesh; flux_type is 2.0 for 2D current flow (i.e. line electrodes) or 3.0 (usual mode) for fully 3D current flow; singular_type is 1 if singularity removal is to be applied (otherwise 0). Note that singularity removal can only be applied is (a) the boundaries are infinite and (b) the y=0 plane defines the upper boundary; 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 then a triangular mesh is to be used. This allows much greater flexibility of defining geometry but requires creation of a finite element mesh. 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_y

where numnp_x is number of <u>nodes</u> in the x direction (horizontal) and numnp_y is the number of <u>nodes</u> in the y (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) yy

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

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

Line 7: (2 Int) numnp_x, numnp_y
where numnp_x is number of nodes in the x direction (horizontal) and numnp_y is the number of nodes in the y (vertical) direction

Line 8: (numnp_x Real) xx

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

Line 9: (numnp_y Real) yy

where yy is an array containing depths of each of numnp_y node rows for column 1 in the x direction. Set yy(1) to zero and the other values to a positive number. (i.e. yy represents depth, not 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 doesn't 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 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_y

where patch_size_x and patch_size_y are the parameter block sizes in the x and y direction, respectively. 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 y direction must be perfectly divisible by patch_size_y otherwise set them both to zero.

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

Line 15: (2 Int) num_param_x, num_param_y

where num_param_x and num_param_y are the number of parameter blocks in the x and y 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) npystart, npy(i), i=1,num_param_y

where npystart is the row number in the mesh where the parameters start; npy specifies the number of elements in each parameter block in the y direction

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

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

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 (2000). 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, 1 or 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 num_alpha_steps. 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 variance model parameters following:
    var(R) = (a_wgt*a_wgt) + (b_wgt*b_wgt) * (R*R)
where R is the resistance measured.

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.log. **NOTE:** that the apparent resistivity calculations assume that you have set the ground surface to Y=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 blocking of parameters (inverse_type = 4 in line 18) so that each block type is disconnected from other blocks then the blocks 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 say a mesh with 10 elements in the y direction and 12 elements in the x direction is set to have a parameter patch size of 2, so in total there are 5 parameters (x) by 6 parameters (y). If we want to set the bottom two rows of parameters (4 elements) not to be smoothed along with the top four rows (8 elements) then the following could be input:

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

Repeat line 24 for all num_param_y

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_xy_poly

where num_xy_poly is the number of x,y co-ordinates that define a polyline bounding the output volume. If num_xy_poly is set to zero then no bounding is done in the x-y plane. The co-ordinates of the bounding polyline follow in the next line. **NOTE:** the first and last pair of co-ordinates must be identical (to complete the polyline). So, for example, if you define a bounding square in x,y then you must have 5 co-ordinates on the polyline. The polyline must be defined as a series of co-ordinates in sequence, although the order can be clockwise or anti-clockwise (see examples later). **NOTE:** ***R2*** stores the vertical co-ordinates for nodes in a 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 y=-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 y=-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 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), y_poly(2)
where x_poly(1), y_poly(1) are the co-ordinates of the first point on the polyline.
Repeat line 26 for all num_xy_poly co-ordinates.

Line 27: (Int) num_electrodes

where num_electrodes is number of electrodes

If (inverse_type = 3) 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**

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 P+ electrode; elec(2,k) is the electrode number for the P- electrode; elec(3,k) is the electrode number for the C+ electrode; elec(4,k) is the electrode number for the 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 (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 P+ electrode; elec(2,k) is the electrode number for the P- electrode; elec(3,k) is the electrode number for the C+ electrode; elec(4,k) is the electrode number for the C- electrode

Repeat Line 3 for all num_ind_meas

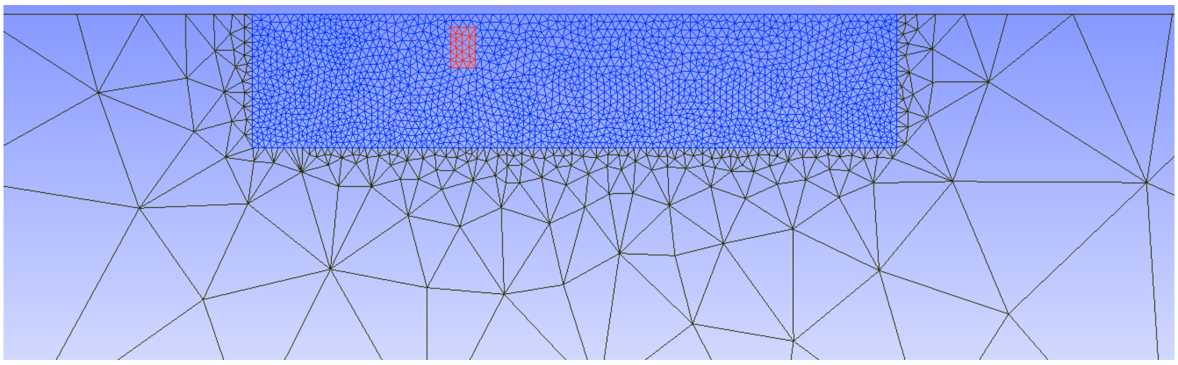
End if

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**

Details of mesh.dat

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.geuz.org/gmsh/) 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).



The ***R2*** download package contains an ***R2*** Gmsh tutorial (gmsh R2 tutorial.pdf) that was written but Judy Robinson at Rutgers University. Judy has also kindly provided two Matlab scripts for working with Gmsh and ***R2***, along with an example (based on the Surface1 example below).

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 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.

Line 1: (2 Int) numel, numnp

Where numel is the number of triangle elements and numnp is the number of nodes.

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; 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.

Repeat line 2 for all numel elements.

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

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

Repeat line 3 for all numnp nodes.

END OF INPUT FOR **mesh.dat**

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

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 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.

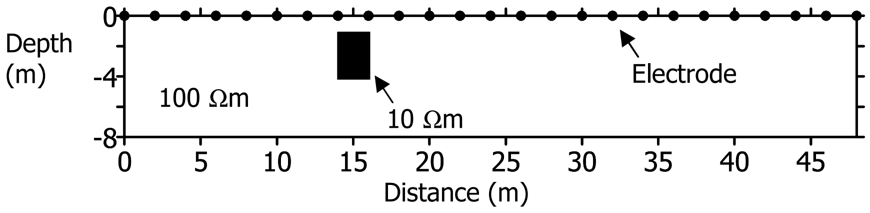


Figure 1: Definition of synthetic model for surface array 1 problem

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. 11025 nodes, 10752 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 log10 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 log10 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 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 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.

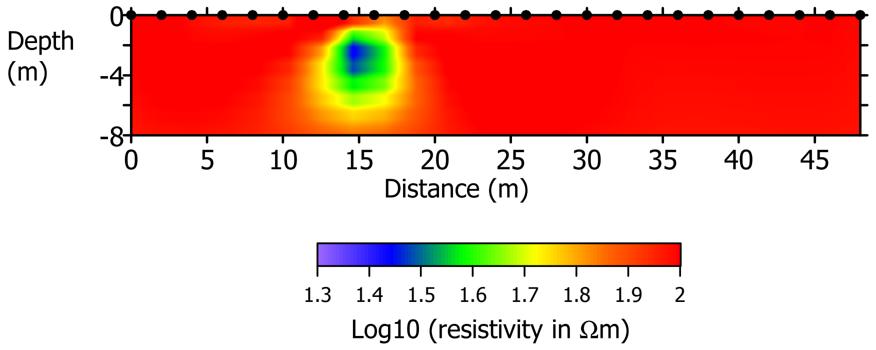


Figure 2: Inverse model for surface array 1 problem with dp-dp array

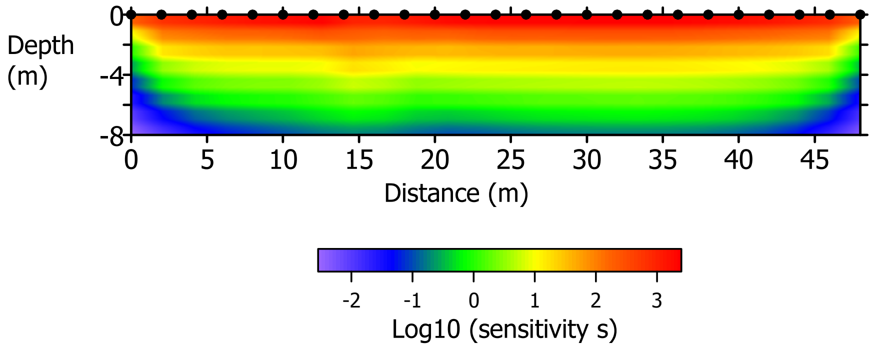


Figure 3: Sensitivity map for inverse model for surface array 1 problem with dp-dp array

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 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.

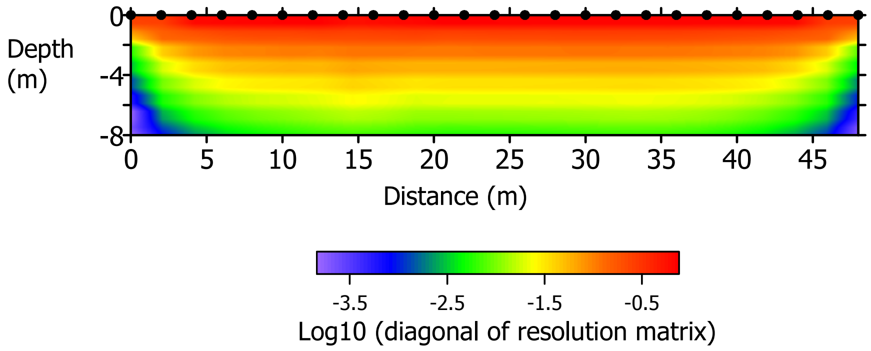


Figure 4: Diagonal of resolution matrix for inverse model for surface array 1 problem with dp-dp array

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 5 shows the resulting model. Figure 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 4),

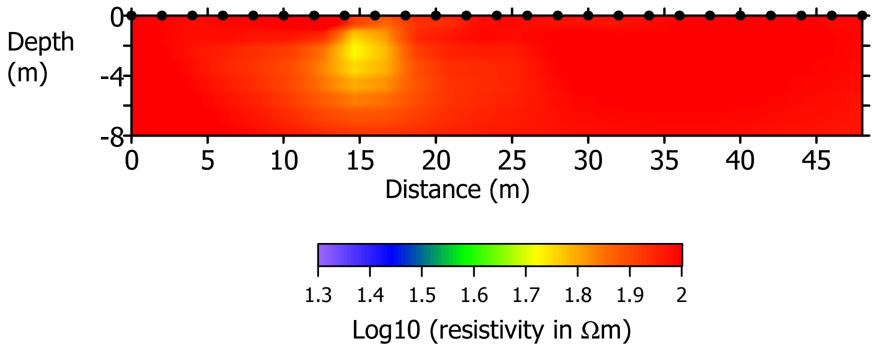


Figure 5: Inverse model for surface array 1 problem with Wenner array

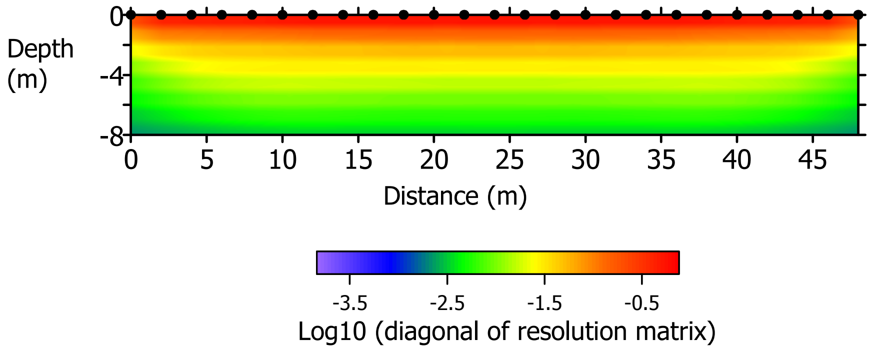


Figure 6: Diagonal of resolution matrix for inverse model for surface array 1 problem with Wenner array

Surface electrode array 2

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 7). The file **R2.in** is now different for the forward and inverse model runs through the addition of topography data. Figure 8 shows the inverse solution for this case.

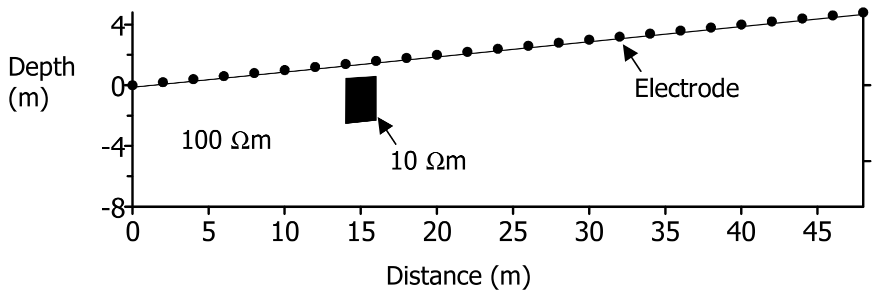


Figure 7: Definition of synthetic model for surface array 2 problem

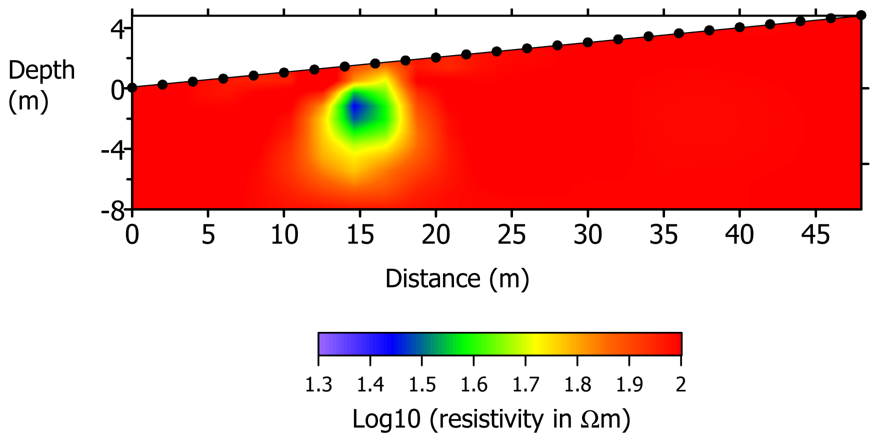


Figure 8: Inverse model for surface array 2 problem with dipole-dipole array

Surface electrode array 3

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 1, i.e. in "Examples/Surface_1/dpdp/Forward".

For this example we use a patch_size of zero in the x and y 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 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).

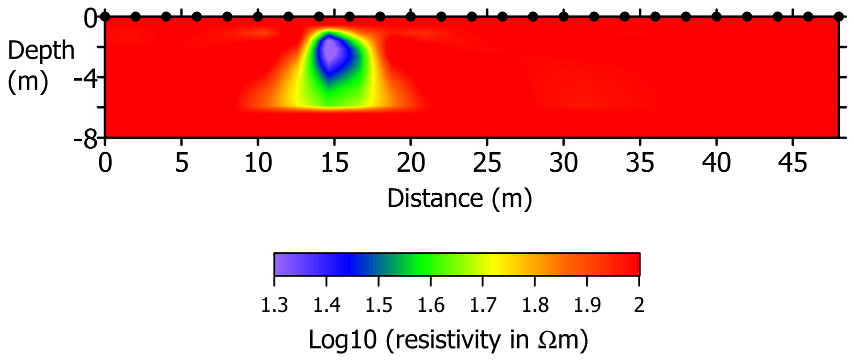


Figure 9: Inverse model for surface array 3 problem

Surface electrode array 4

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 shows the resultant inversion (c.f. Figure 2 with isotropic smoothing).

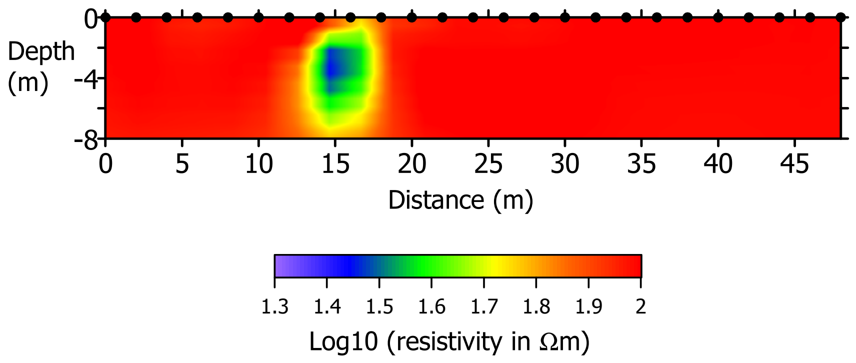


Figure 10: Inverse model for surface array 4 problem

Surface electrode array 5

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 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).

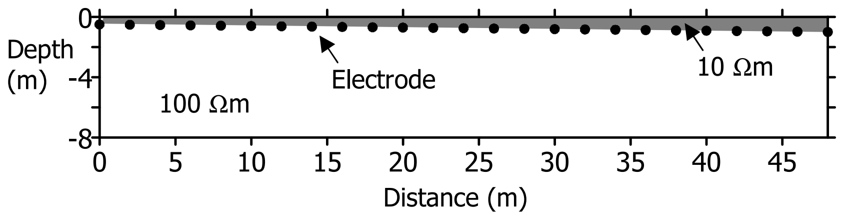


Figure 11: Forward model for surface array 5 problem

Surface electrode array 6

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 12a. The result is similar to that shown in Figure 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 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 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 4).

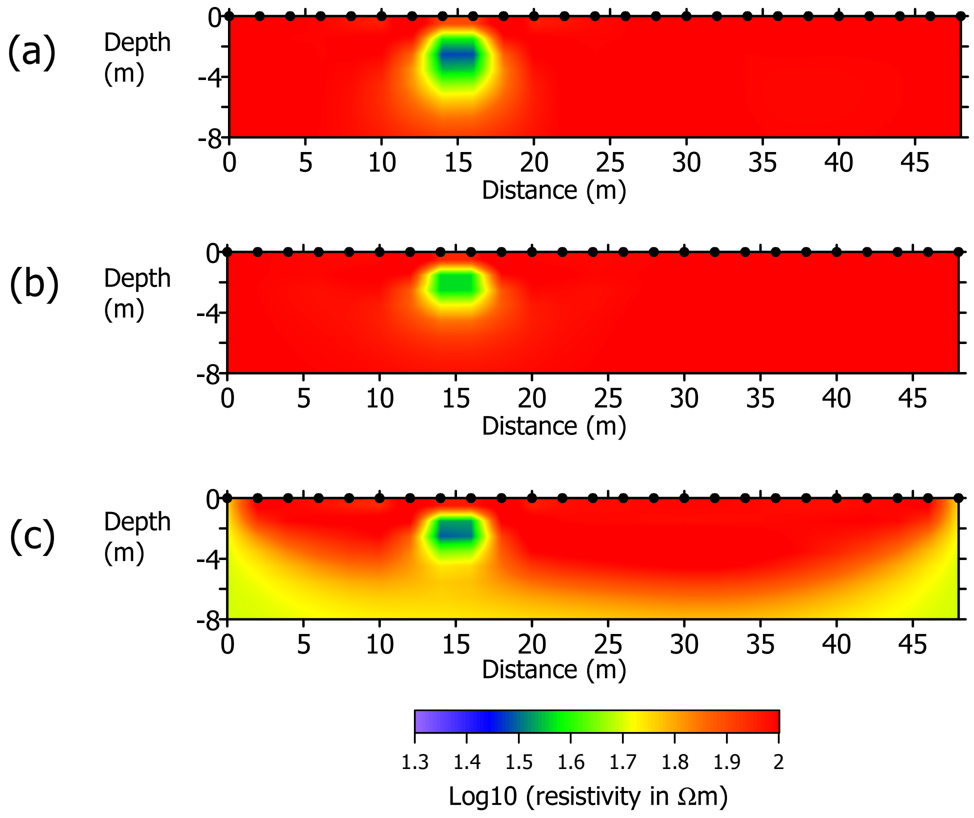


Figure 12: Surface array 6 – regularising relative to a reference resistivity model. (a) α_s=10, ρ_{back}=100Ωm. (b) α_s=50, ρ_{back}=100Ωm. (c) α_s=10, ρ_{back}=50Ωm.

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 12c (in log units) using 50Ωm then, $m_{1,r} = \log_{10}(100)$ and $m_{2,r} = \log_{10}(50)$. Figure 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 13 shows a contour of $R = 0.2$ to illustrate this.

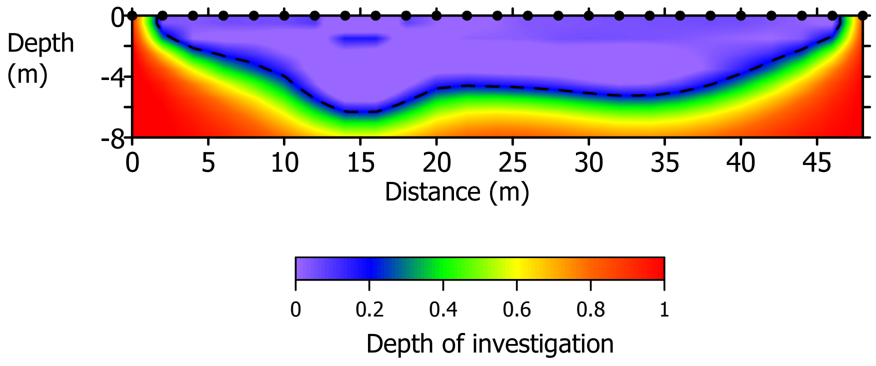


Figure 13: Depth of investigation for Surface array 6 problem.

Surface electrode array 7

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 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.

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. Figure 15 shows the results for the case considered here.

For illustration, Figure 16 shows the results (in **f001_res.vtk**) plotted as log resistivity, using ParaView. The electrode locations (from file **electrodes.vtk**) are also shown.

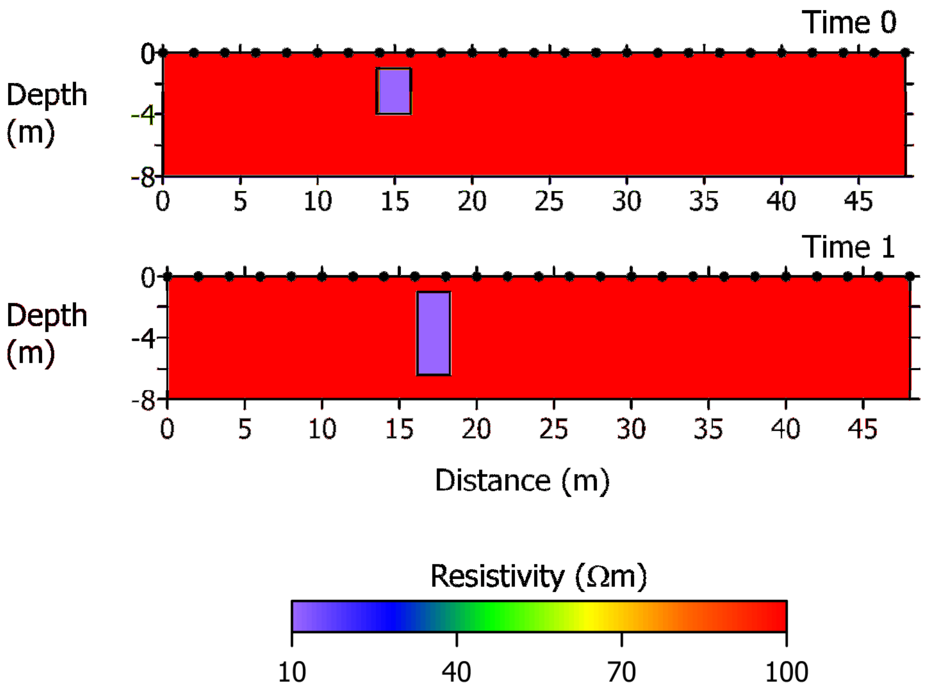


Figure 14. Forward model definitions for difference inversion example.

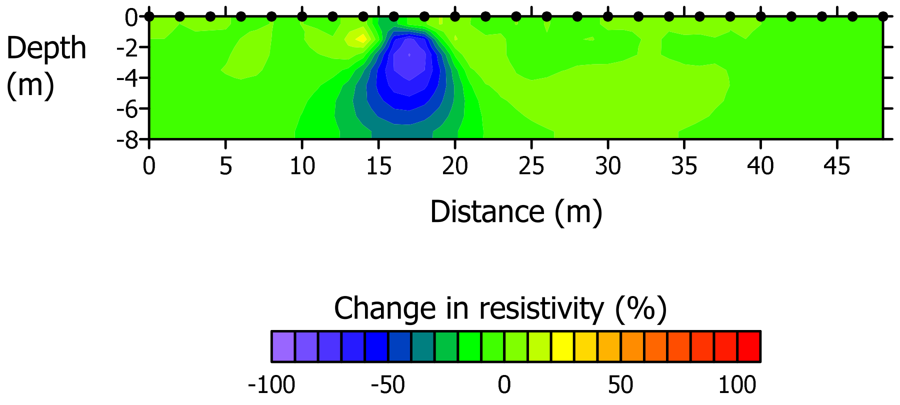


Figure 15. Change in resistivity from difference inversion.

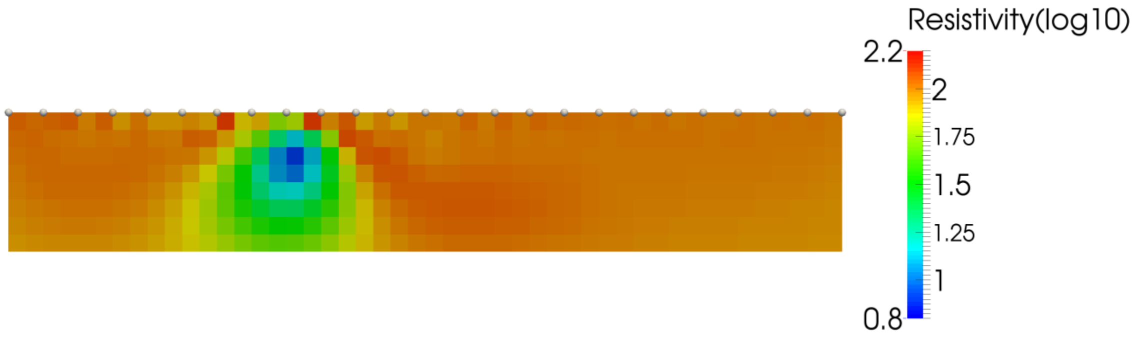


Figure 16. Resistivity model obtained from difference inversion (plotted in ParaView).

Surface electrode array 8

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 17 shows a plot of the forward model definitio 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.

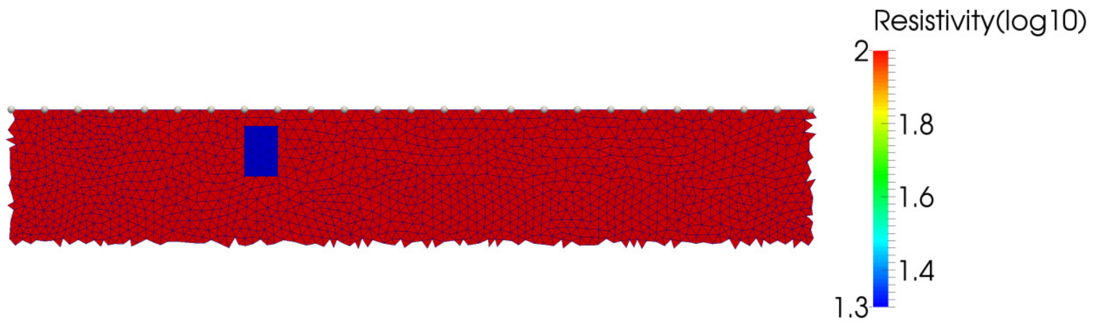


Figure 17: Definition of forward model using a triangular mesh

The folder Examples/Surface_8/Inverse_1 contains the input files for an inversion of the dipole-dipole data using a triangular mesh. Figure 18 shows the result, plotted in ParaView.

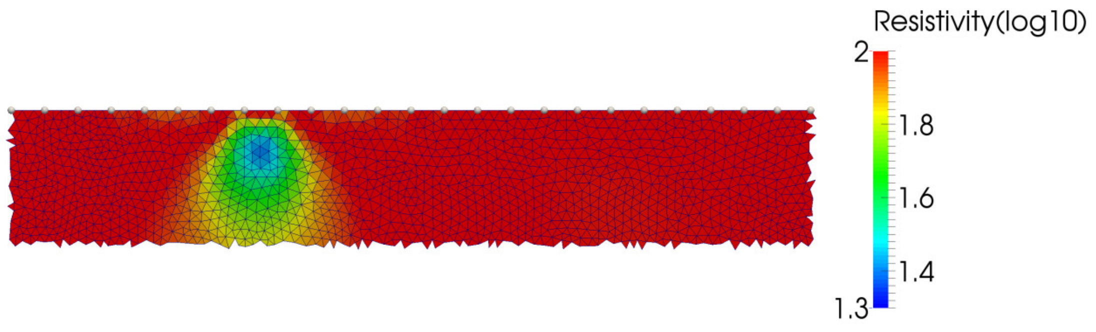


Figure 18: Inversion of dipole-dipole data.

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 19).

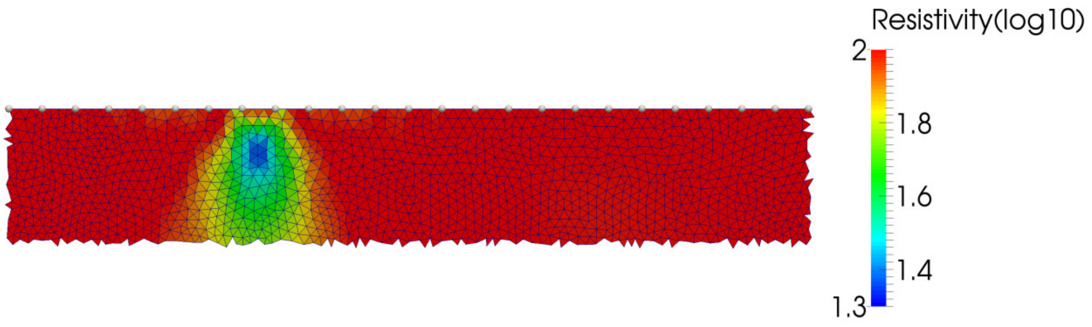


Figure 19: Inversion of dipole-dipole data with enhanced vertical smoothing.

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 17 and the other representing the remainder of the mesh. The inversion is show in Figure 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).

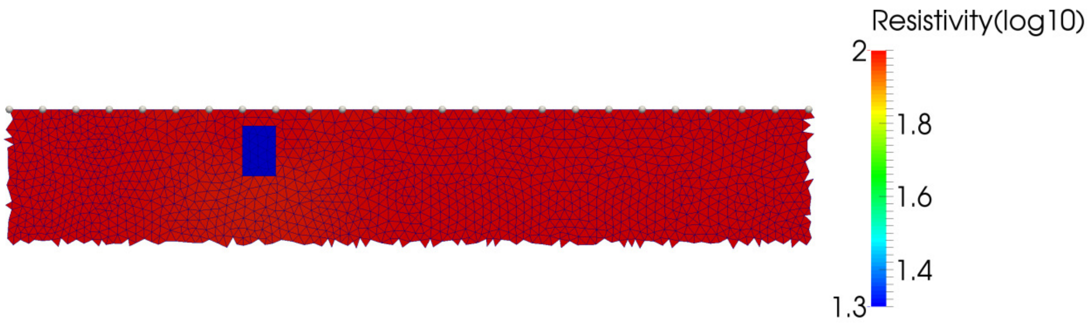


Figure 20: Inversion of dipole-dipole data with region blocking.

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 21. 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 22 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 23. 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 24 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 24 and 22.

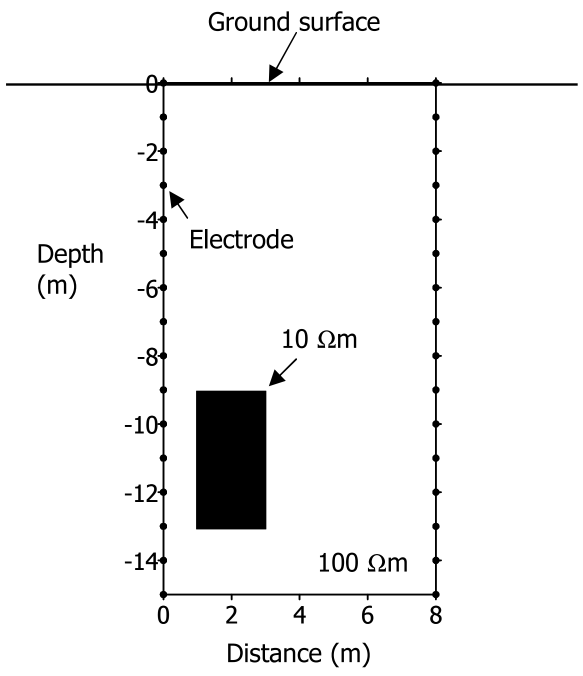


Figure 21: Forward model definition for cross borehole case 1

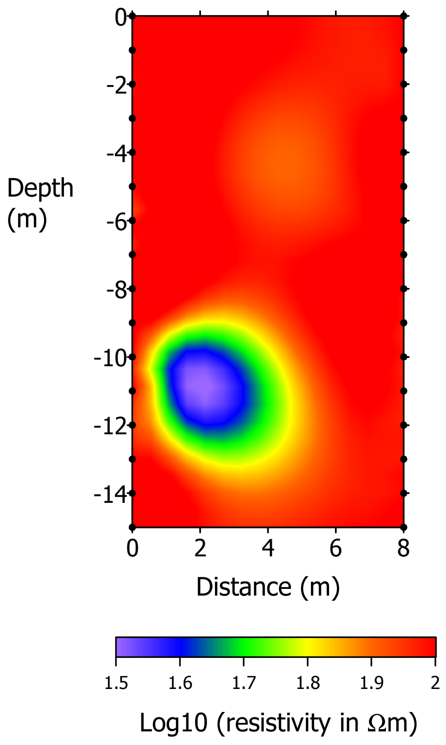


Figure 22: Inverse model for cross borehole case 1

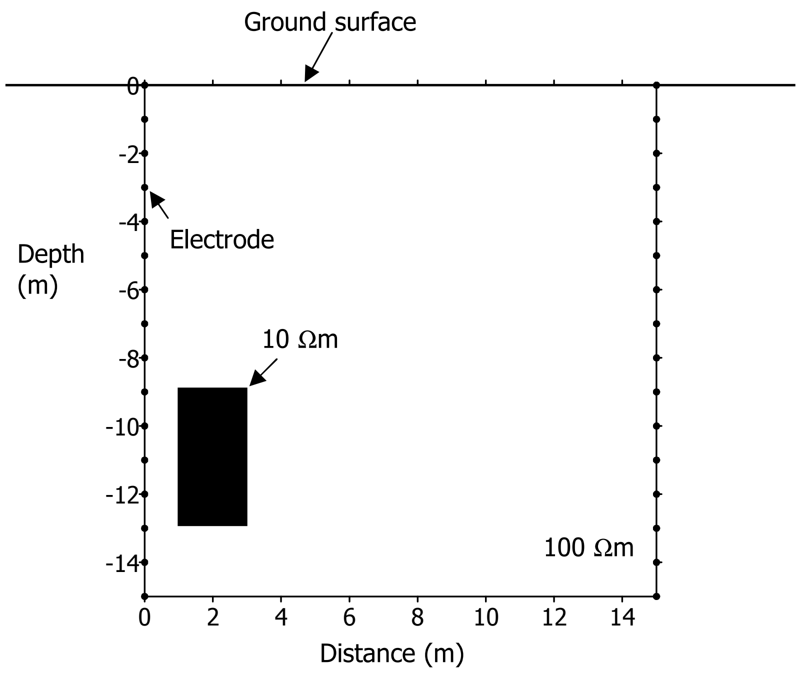


Figure 23: Forward model definition for cross borehole case 2

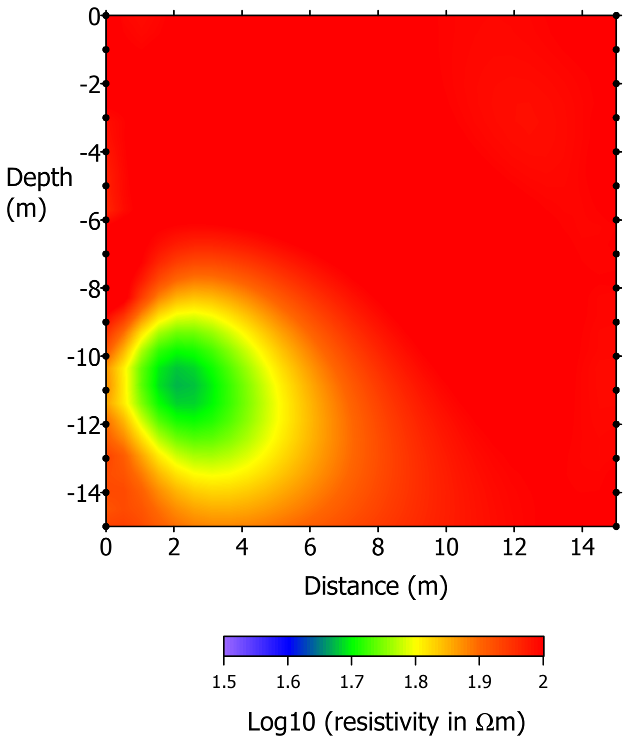


Figure 24: Inverse model for cross borehole case 2

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 and create a DOS window, 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. Note that R2(x64).exe will not run in CMD mode by typing R2(x64) because of the ()'s – the user needs to rename the file and remove the ()'s.

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. 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, 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 means that the data you input will be log transformed, 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) 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 quadrilateral mesh the electrode positions are defined by their column and row positions in the mesh (Line 27). 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 bit 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.

References

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

LaBrecque, D.J. and X. Yang, 2000, Difference inversion of ERT data: A fast inversion method for 3-D in-situ monitoring, Proc. SAGEEP 2000, 907-914.

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

W. 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
