Rapid parallel computation of optimised arrays for electrical imaging surveys

M.H.Loke, Geotomo Software

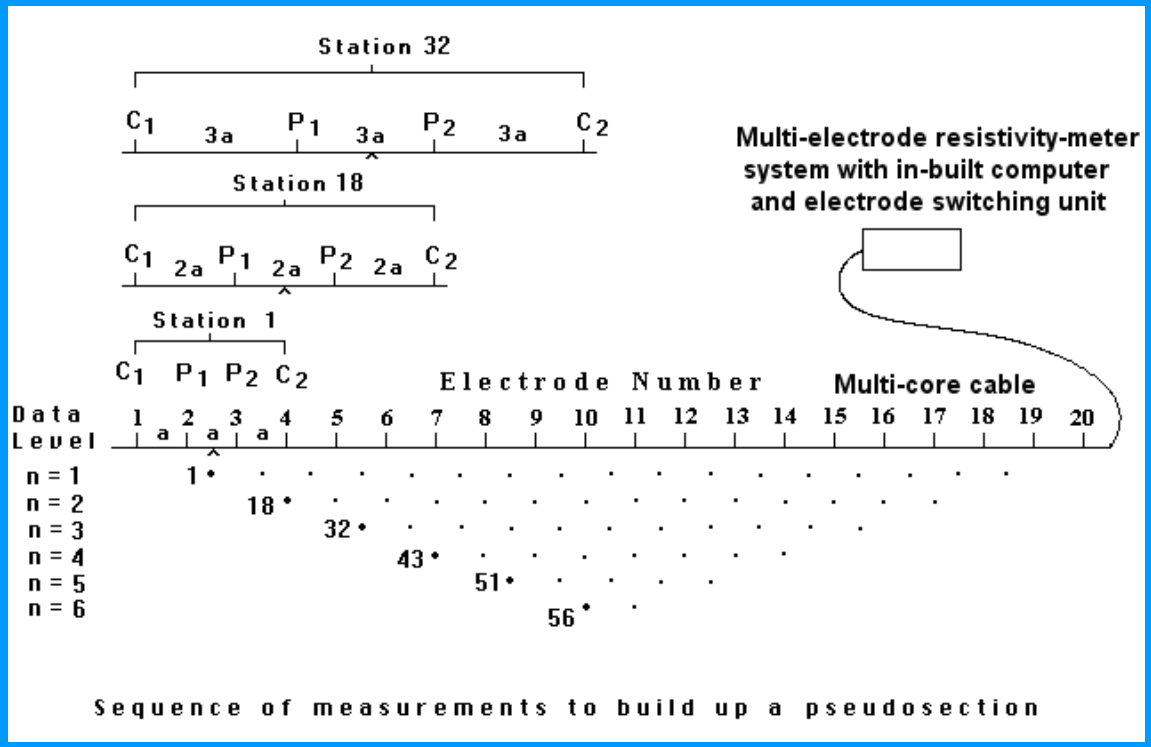
& P. Wilkinson, British Geological Survey

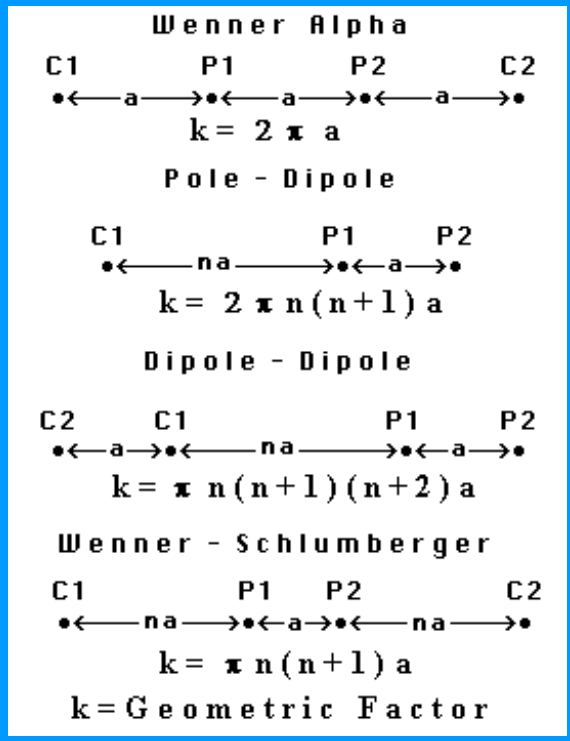
Email : drmhloke@yahoo.com

Paper presented at Near Surface 2009, 7-9 Sept. 2009, Dublin, Ireland.

2-D electrical imaging surveys

2-D electrical imaging surveys are now widely used. Many systems have computerised automatic switching units that allow any array configuration to be used. Most common arrays are the Wenner, dipole-dipole, Wenner-Schlumberger, pole-pole, and multiple-gradient arrays. All surveys have a limit on the number of readings that can be made due to economic and time constraints.





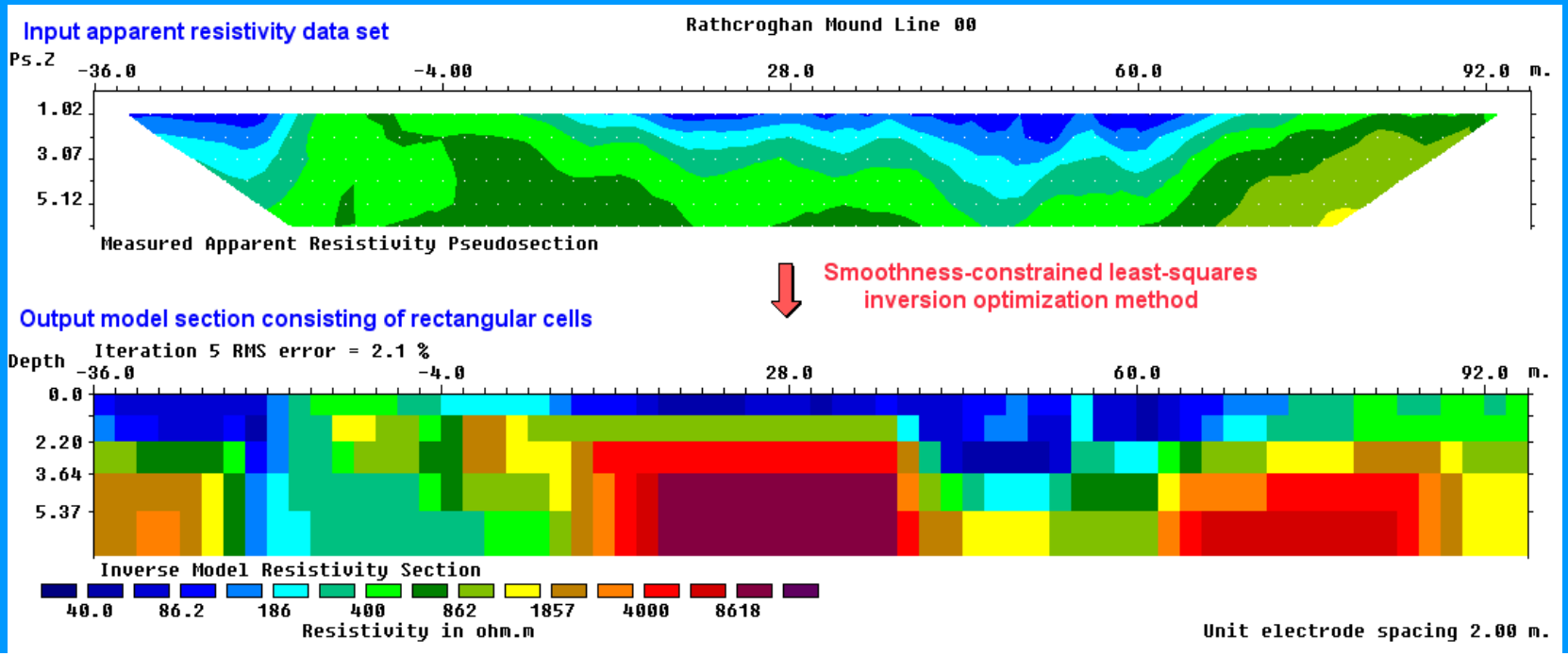
What is the best array to get the maximum amount of information?

Data inversion and the least-squares method

The smoothness-constrained least-squares method commonly used for the inversion for resistivity data, whose equation is given by

$$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}+\lambda\mathbf{F}\right)\Delta\mathbf{q}_{\mathrm{k}}=\mathbf{J}^{\mathrm{T}}\mathbf{g}-\lambda\mathbf{F}\mathbf{q}_{\mathrm{k}}$$

F contains the roughness filters and other constraints, λ is a damping factor to reduce the change in model resistivity values (Δq), and g is the data misfit. J is the Jacobian matrix of partial derivatives.



Array selection – subsurface resolution

Purpose of survey is to detect or resolve structures below. One approach is to select arrays that will maximize the subsurface resolution – an optimisation solution to the array design problem.

How do we quantify the 'resolution'? The model resolution equation is related to the least-squares equation commonly used for the inversion for resistivity data. $$\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}+\lambda\mathbf{F}\right)\Delta\mathbf{q}_{\mathrm{k}}=\mathbf{J}^{\mathrm{T}}\mathbf{g}-\lambda\mathbf{F}\mathbf{q}_{\mathrm{k}}$$

The relationship between the calculated model resistivity and the true resistivity is approximately given by

$$\mathbf{q}_{\mathrm{Model}}\approx\mathbf{R}\,\mathbf{q}_{\mathrm{Actual}},\quad \mathbf{R}=\left(\mathbf{J}^{\mathrm{T}}\mathbf{J}+\lambda\mathbf{F}\right)^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{J}$$

The **R** matrix is called the resolution matrix. It can be considered as a 'filter' or 'distorting lens' through which we see the subsurface.

To simplify the calculations, we first use the damped least-squares equation for the model resolution calculations (**F=I**).

Model resolution – simple example

Consider as simple model with only 4 cells.

The relationship between the calculated resistivity value for each cell and the true cell resistivity value is given by

$$\mathbf{q}_{\text{Model}} \approx \mathbf{R}\,\mathbf{q}_{\text{Actual}}, \quad \mathbf{R} = \left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{F}\right)^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{J}$$

Model with 4 cells

q_1	q_2
q_3	q_4

This can be written in matrix form as

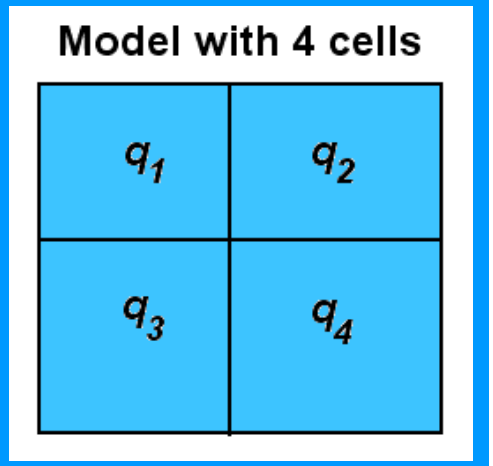
$$\begin{pmatrix} q_{M1} \\ q_{M2} \\ q_{M3} \\ q_{M4} \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & R_{13} & R_{14} \\ R_{21} & R_{22} & R_{23} & R_{24} \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & R_{42} & R_{43} & R_{44} \end{pmatrix} \begin{pmatrix} q_{A1} \\ q_{A2} \\ q_{A3} \\ q_{A4} \end{pmatrix}$$

$$\mathbf{q}_{\text{Model}} = \mathbf{R}\,\mathbf{q}_{\text{Actual}}$$

Simple examples of model resolution matrix

If the cells are perfectly resolved, the model resolution matrix will have diagonal elements of 1.0 and elsewhere it is 0.0. The calculated value for each cell depends only on the true value.

$$\begin{pmatrix} q_{M1} \\ q_{M2} \\ q_{M3} \\ q_{M4} \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \end{pmatrix} \begin{pmatrix} q_{A1} \\ q_{A2} \\ q_{A3} \\ q_{A4} \end{pmatrix}$$



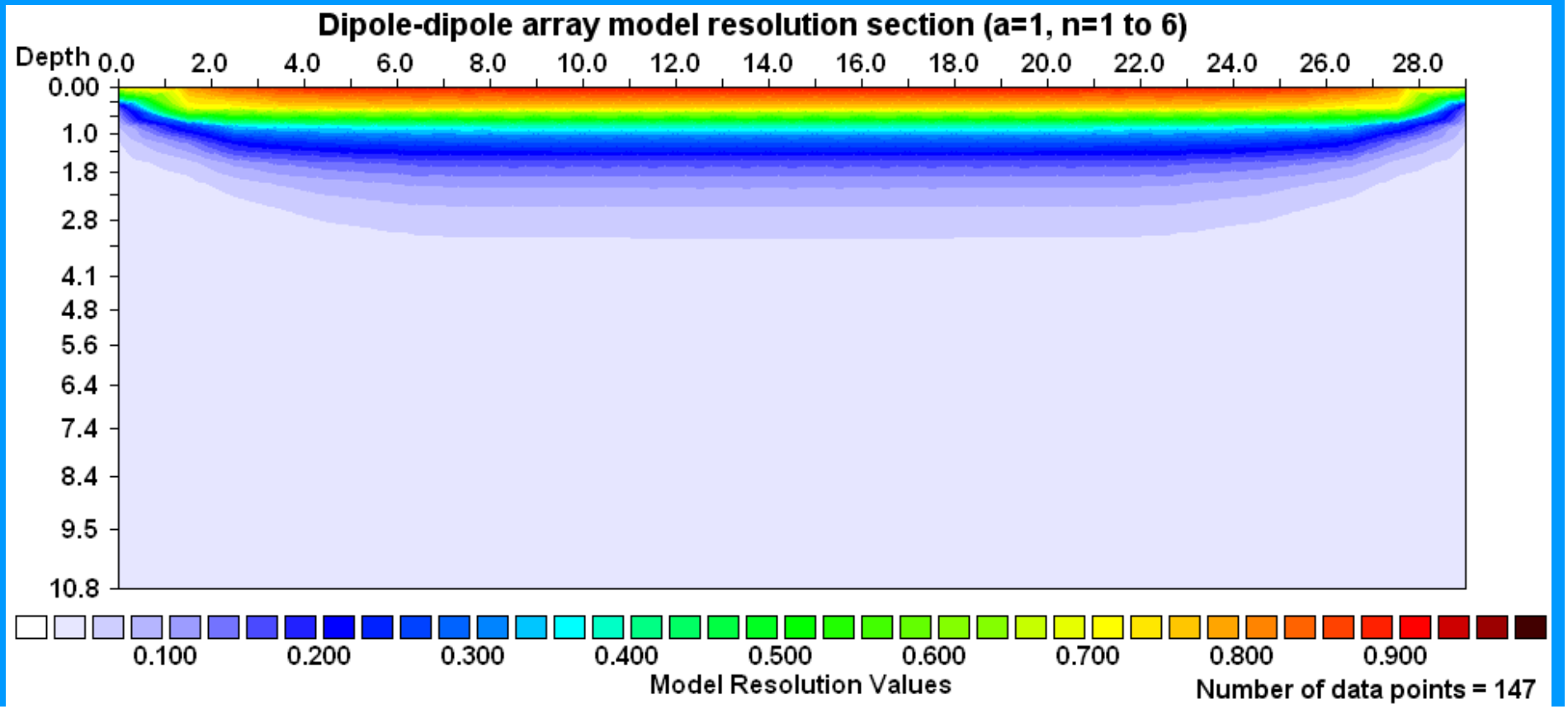
For imperfect resolution the diagonal elements are less than 1.0, and off-diagonal elements are non-zero. The diagonal elements give the 'degree' of resolution, while the off-diagonal elements give the degree of 'contamination' or cross-correlation with the neighboring model cells.

$$\begin{pmatrix} q_{M1} \\ q_{M2} \\ q_{M3} \\ q_{M4} \end{pmatrix} = \begin{pmatrix} 0.7 & 0.1 & 0.1 & 0.1 \\ 0.1 & 0.7 & 0.1 & 0.1 \\ 0.1 & 0.1 & 0.5 & 0.3 \\ 0.1 & 0.1 & 0.3 & 0.5 \end{pmatrix} \begin{pmatrix} q_{A1} \\ q_{A2} \\ q_{A3} \\ q_{A4} \end{pmatrix}$$

$$\mathbf{q}_{\text{Model}} = \mathbf{R}\,\mathbf{q}_{\text{Actual}}$$

Model resolution – 2D section plot

A plot of the R matrix diagonal elements values shows the degree at which the calculated model value depends on the true value. A value of 0.05 (5%) is usually chosen as the 'cutoff' value. Below is the resolution plot for a dipole-dipole survey with with 'a=1', and 'n=1 to 6' for a line with 30 electrodes with 1.0 meter spacing. The resolution is not significant below a depth of about 3.5 meters.



Maximum possible array configurations

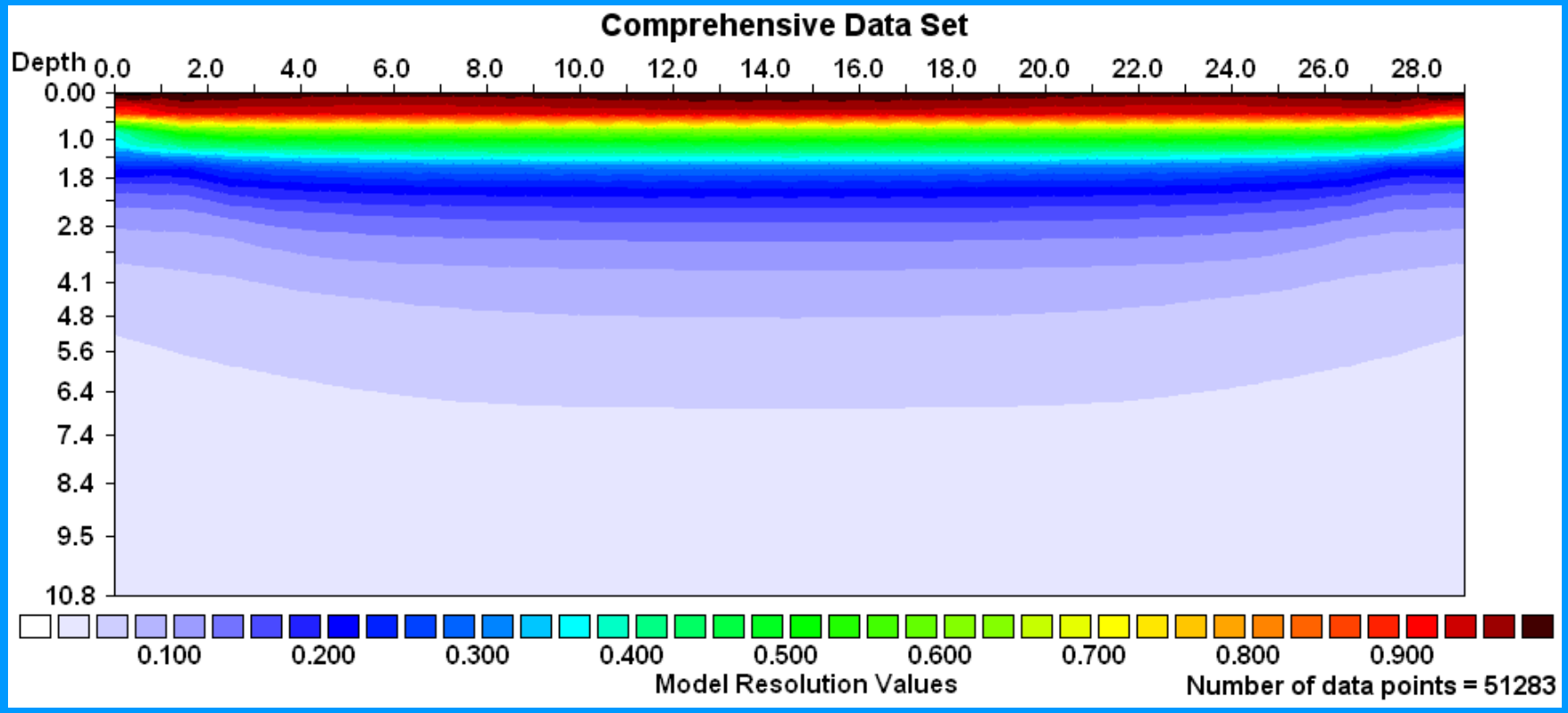
The model resolution section gives an insight on the section of the subsurface we can resolve. Is there a way to automatically select the arrays that will give the maximum resolution for a given number of electrodes?

The number of possible 4 electrodes array combinations (*M*) for a line with *N* electrodes is : $M=N(N-1)(N-2)(N-3)/8$

For a system with 30 electrodes, there are 82215 possible configurations. We can reduce the number by rejecting all configurations with the Gamma arrangement (interleaved current and potential electrodes), and those with very high geometric factors (i.e. low potentials) greater than the dipole-dipole with a=1 and n=6. This reduces the total number of possible array configurations to 51283.

The maximum possible resolution

A system with 30 electrodes has 51283 possible 'viable' array configurations (the 'comprehensive' data set). Measurements made with all these possible array configuration should give the best possible resolution, as shown below.



The region with significant resolution values extends to about 6.5 meters, more than the simple dipole-dipole (3.5 meters) array.

An automatic optimised arrays generator

Taking all the 51283 possible readings to give the maximum possible resolution is not practical. We want to take a small number of readings that gives almost the same resolution. A possible algorithm.

1. Start with a small set, the 'base' data set, such as the dipole-dipole readings with a=1 and n=1 to 6 (147 readings).

2. Calculate the increase in the model resolution as each new array from the 'comprehensive' data set is added to the 'base' set. Add the arrays that give the highest increase to the model resolution (and are sufficiently independent of each other) to the base set. In each iteration, increase the base set by about a fixed percentage (usually 3% to 9%).

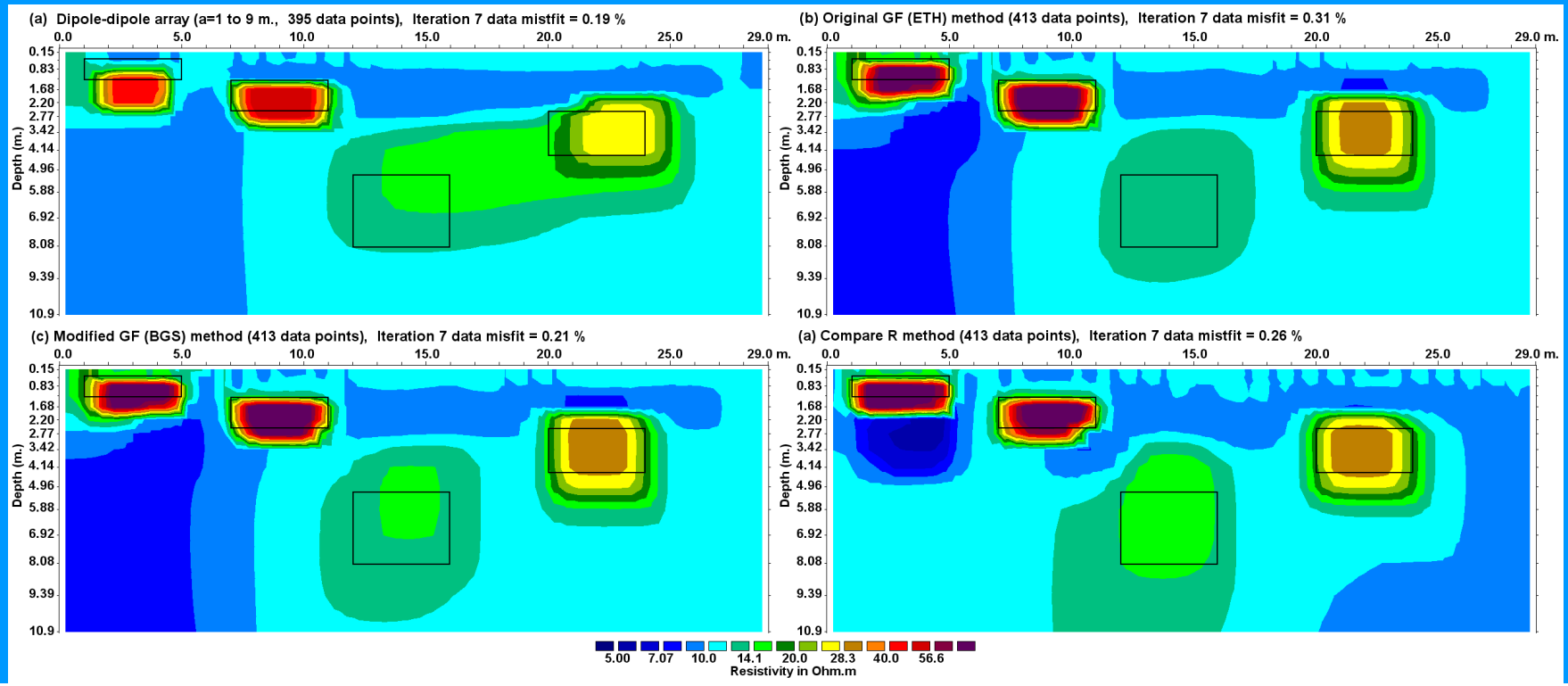
3. Use the new set (base plus new arrays) as the next base set, and repeat the procedure. Stop when the desired maximum number of arrays (eg. 5000) is reached.

Reason for using the 'Compare R' method

Two types of automatic array generators.

First type : Recalculate the change in the model resolution for each new array added. Slowest but gives the most accurate results.

Second type : Use linear estimates of the model resolution change. Faster but results are poorer. Example for synthetic model with four 100 ohm.m rectangular blocks in a 10 ohm.m background. GF methods use linear estimates. Compare R uses direct recalculation.



How to calculate the change in model resolution?

The Sherman-Morrison Rank-1 update is used to calculate the main diagonal elements of the model resolution matrix of the base set plus the test configuration. We first simplify the model resolution matrix ($\mathbf{R_b}$) equation into the following form.

$$\mathbf{R_b} = \mathbf{B_b A_b}, \textit{where } \mathbf{A} = \mathbf{J^T J} \textit{ and } \mathbf{B} = \left(\mathbf{J^T J} + \lambda \mathbf{F}\right)^{-1}$$

The following set of updating formulae is used

$$\mathbf{A_{b+1}} = \mathbf{A_b} + \mathbf{g} \otimes \mathbf{g}, \quad \mathbf{B_{b+1}} = \mathbf{B_b} - \frac{\mathbf{z} \otimes \mathbf{z}}{1+\mu}, \quad \mathbf{R_{b+1}} = \mathbf{B_{b+1}} \mathbf{A_{b+1}}$$

where g is the vector containing Jacobian values of the test configuration, z=Bg, μ=g.z and g⊗g denotes the matrix multiplication of g with $\mathbf{g^T}$.

$\mathbf{R_{b+1}}$ is the resolution matrix of the base set plus one new array.

Repeat this calculation about 51000 times for all the 'viable' array configurations!

The original 'Compare R' method

This method directly uses the Sherman-Morrison updating formula.

$$\mathbf{A}_{b+1} = \mathbf{A}_b + \mathbf{g} \otimes \mathbf{g}, \quad \mathbf{B}_{b+1} = \mathbf{B}_b - \frac{\mathbf{z} \otimes \mathbf{z}}{1+\mu}, \quad \mathbf{R}_{b+1} = \mathbf{B}_{b+1}\mathbf{A}_{b+1}$$

Three steps needed (1) calculate $\mathbf{A}_{b+1}$ (2) calculate $\mathbf{B}_{b+1}$ (3) calculate $\mathbf{R}_{b+1}$ from $\mathbf{A}_{b+1}$ and $\mathbf{B}_{b+1}$.

Problem : Too slow. A direct implementation of the updating formula took 6 hours to calculate the optimised arrays for a survey line with 30 electrodes!

Cause : Need to store updated temporary matrices $\mathbf{A}_{b+1}$ and $\mathbf{B}_{b+1}$ into computer main memory, then later retrieve to calculate $\mathbf{R}_{b+1}$. Memory access is much slower than CPU calculations.

Solution : Need a faster algorithm. Involves a deeper look at fast matrix techniques and optimising use of the computer hardware.

The Matrix-Vector method

Original 'Compare R' method uses 3 steps.

1. $\mathbf{A}_{b+1} = \mathbf{A}_b + \mathbf{g} \otimes \mathbf{g}$, 2. $\mathbf{B}_{b+1} = \mathbf{B}_b - \dfrac{\mathbf{z} \otimes \mathbf{z}}{1+\mu}$, 3. $\mathbf{R}_{b+1} = \mathbf{B}_{b+1}\mathbf{A}_{b+1}$

Rewrite all into a single equation.

$$\mathbf{R}_{b+1} = \mathbf{R}_b + \Delta\mathbf{R}_b, \text{ where } \Delta\mathbf{R}_b = -\frac{\mathbf{z} \otimes \mathbf{z}}{1+\mu}\mathbf{A}_b + \mathbf{B}_b\mathbf{g} \otimes \mathbf{g} - \frac{\mathbf{z} \otimes \mathbf{z}}{1+\mu}\mathbf{g} \otimes \mathbf{g}$$

Only need to calculate the diagonal elements of $\Delta\mathbf{R}_{b+1}$ (the change in the model resolution). No need to calculate $\mathbf{A}_{b+1}$ and $\mathbf{B}_{b+1}$, and then recalculate $\mathbf{R}_{b+1}$.

Reduce memory access by carrying out all calculations for $\Delta\mathbf{R}_{b+1}$ within the CPU registers. Also make use of CPU (2.66GHz i7) parallel processing features (SIMD and multi-core).

Result : Computer time reduced from 6 hours to 202 seconds.

Speedup factor : 106

The Matrix-Matrix method

The matrix-vector method uses the equation.

$$\Delta \mathbf{R}_b = -\frac{\mathbf{z} \otimes \mathbf{z}}{1+\mu}\mathbf{A}_b + \mathbf{B}_b \mathbf{g} \otimes \mathbf{g} - \frac{\mathbf{z} \otimes \mathbf{z}}{1+\mu}\mathbf{g} \otimes \mathbf{g}$$

Most of the time is spent calculating the matrix-vector products z=Bg and y=Az.

A single matrix-matrix multiplication is much faster than many matrix-vector products. Calculate the change in the resolution matrix elements for many add-on arrays at the same time using

$$\mathbf{Z} = \mathbf{BJ}, \mathbf{Y} = \mathbf{AZ} \quad \text{where } \mathbf{Z} = [\mathbf{z}_1 \mathbf{z}_2 .... \mathbf{z}_k], \mathbf{J} = [\mathbf{g}_1 \mathbf{g}_2 .... \mathbf{g}_k]$$

Best value for *k* is 28 for 64-bit Intel CPUs. Reduce the number of times matrics A and B have to be moved from DRAM to CPU.

Result : Computer time reduced to 87 seconds.

Speedup factor : 248

The GPU Matrix-Matrix method

The matrix-matrix method calculates change in the resolution matrix for many arrays at the same time using the equations.

$$\mathbf{Z = BJ}, \mathbf{Y = AZ} \quad \text{where} \quad \mathbf{Z} = [z_1 z_2 .... z_k], \mathbf{J} = [g_1 g_2 .... g_k]$$

The Intel i7 CPU has 4 physical cores. A GPU (Graphics Processor Unit) is a CPU with hundreds of cores but limited to simple operations, such as matrix-matrix multiplications. Calculate 512 arrays at one time with the GPU!

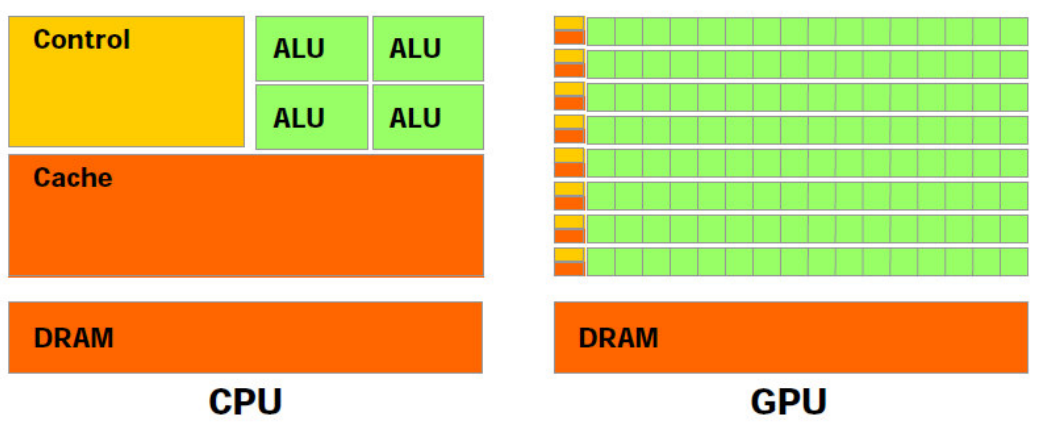
Nvidia 285GTX GPU

1024MB RAM, 1063 GFlops

30 Multi-processors

Multi-processor = 8 processors

Total 240 processors!



The GPU Devotes More Transistors to Data Processing

Source : Nivida Corp., 2008

Result : Computer time reduced to 50 seconds.

Speedup factor : 432

The Single-Precision GPU Matrix-Matrix method

Main limitation for GPU matrix-matrix method is data transfer from CPU memory to GPU memory.

CPU version : Matrix-matrix product takes 76% of total time

GPU version : Matrix-matrix product takes 0.2%, but data transfer takes 60% of total time.

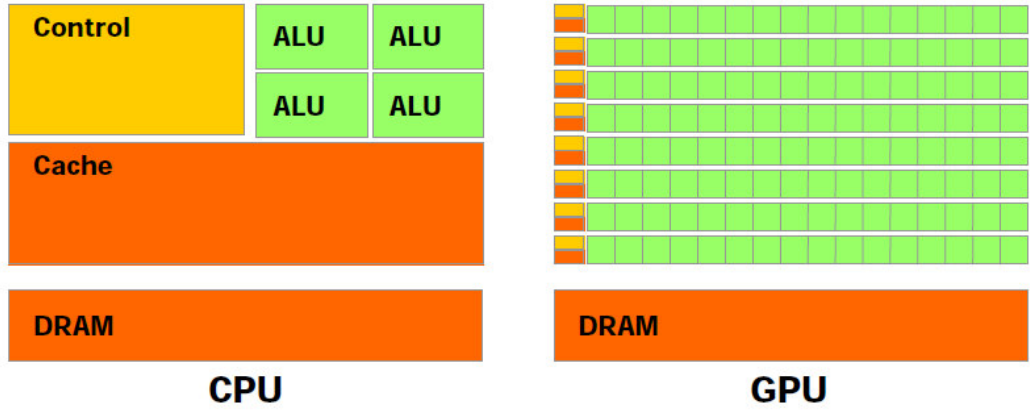
Next step : Run calculations in single-precision instead of double precision.

30 electrodes test : Average relative model resolution after 40 iterations.

Double Precision : 0.958

Single Precision : 0.955

Time reduced from 50 to 30 s.



The GPU Devotes More Transistors to Data Processing
Source : Nivida Corp., 2008

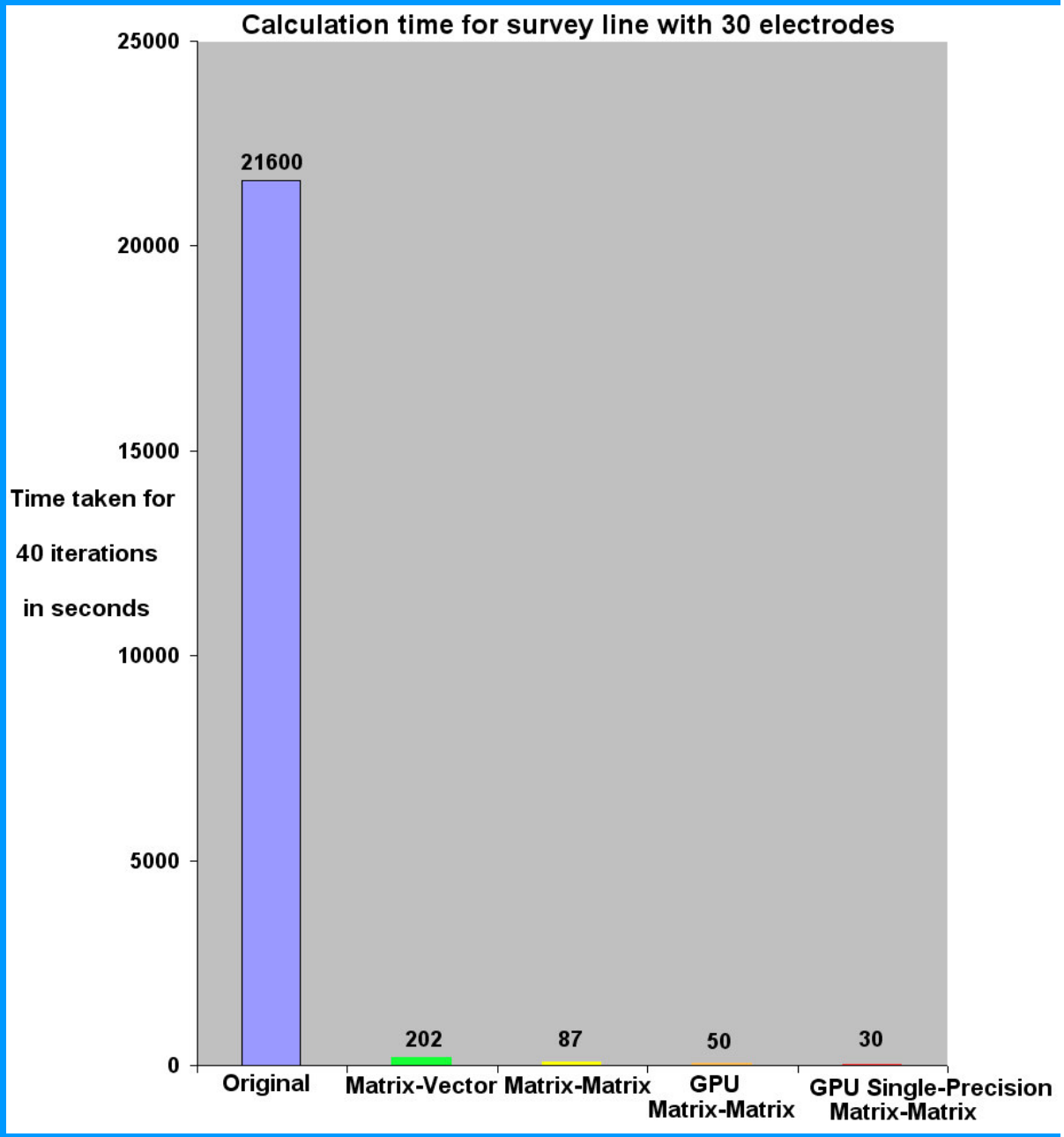
Result : Computer time reduced to 30 seconds.

Speedup factor : 720

Comparison of all ‘Compare R’ methods

Large reduction in computer time achieved by maximising the use of the computer CPU and GPU resources.

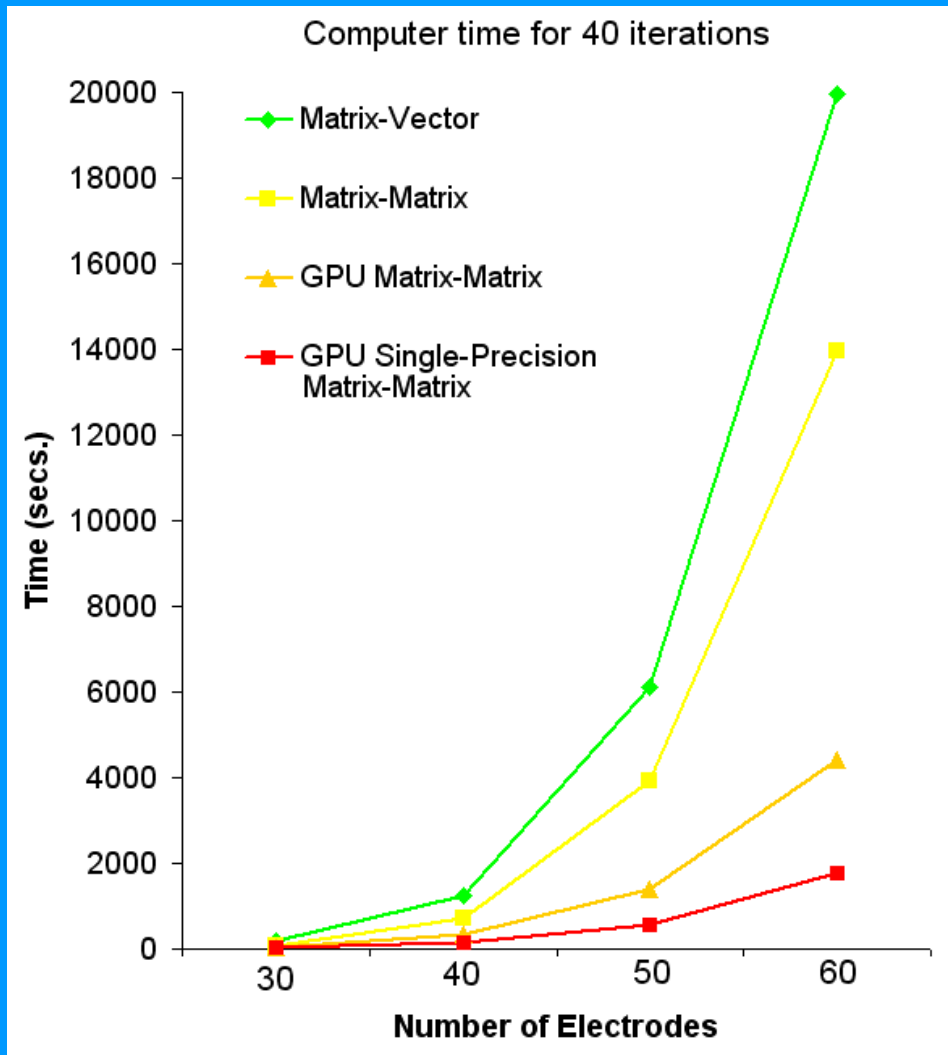
How can we use the computing power of the new algorithms?



Optimisation of longer survey lines

The fast algorithms enables the use of the 'Compare R' method for much longer survey lines. The speed advantage of the GPU methods over the CPU based methods increases exponentially with the number of electrodes.

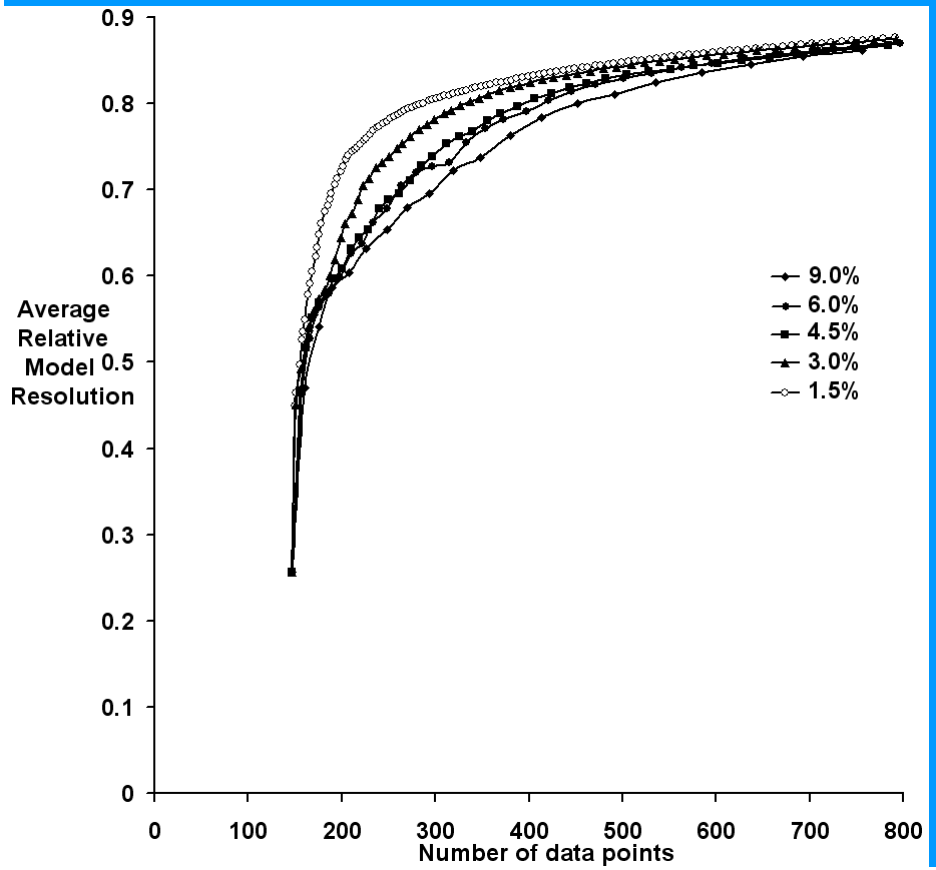
This makes it possible to use the array optimisation method for long 2D surveys lines, and 3D surveys as well.

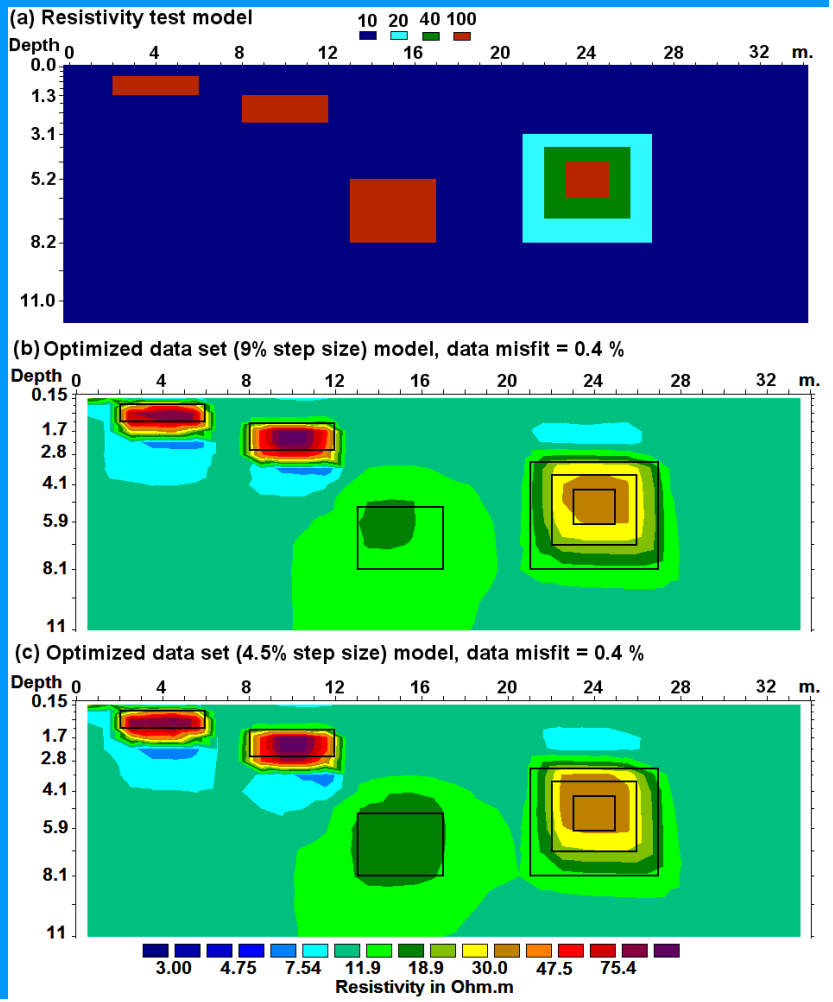


Use smaller step sizes

Use a smaller step size instead of increasing the number of arrays by 9% after each iteration. For small optimised data sets, better resolution can be obtained by using a smaller step size. Requires more iterations, but not a problem with the fast numerical methods.

Example : Synthetic model with optimised arrays with 599 data points.





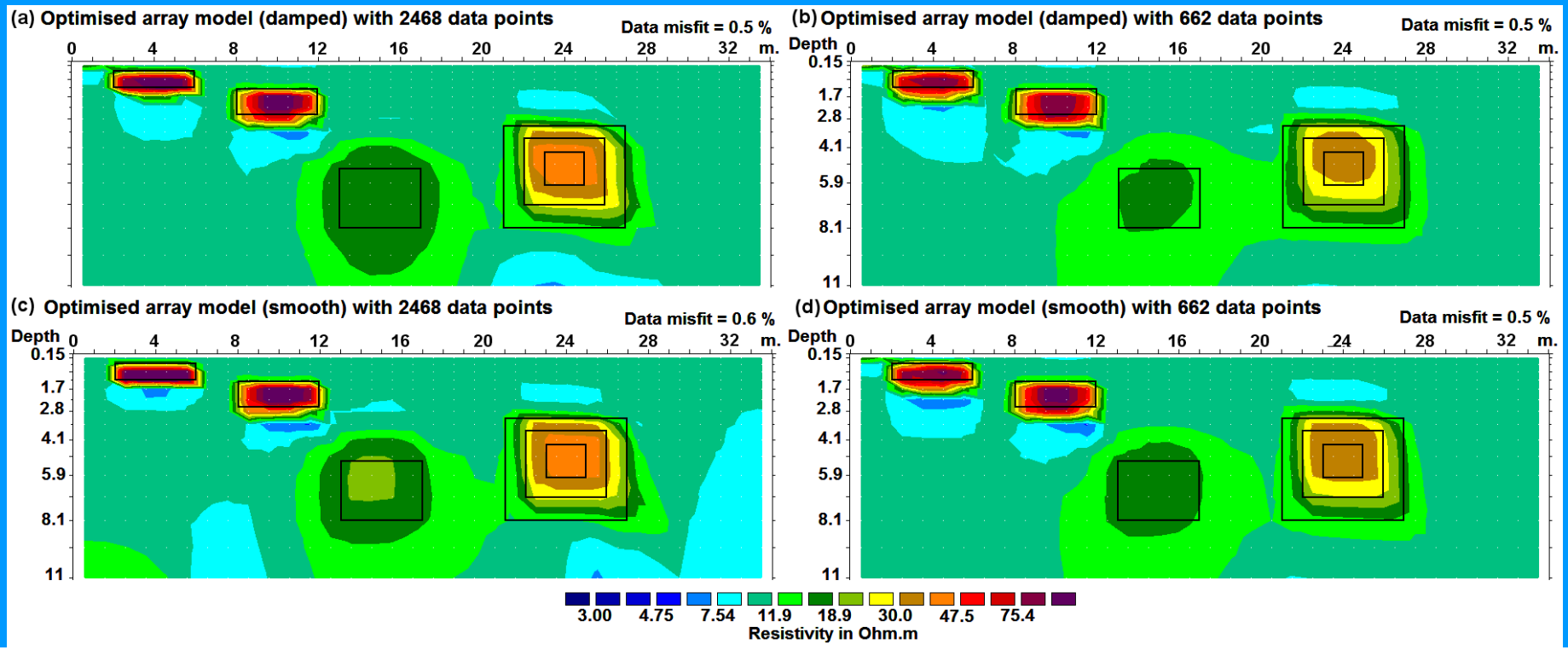
Using the L2 norm

Damped least-squares equation. $\mathbf{R} = \left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{I}\right)^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{J}$

Some results with the L2 norm (smoothness-constrained) least-squares equation. W is a first-order difference matrix.

$$\mathbf{R} = \left(\mathbf{J}^{\mathrm{T}}\mathbf{J} + \lambda\mathbf{W}^{\mathrm{T}}\mathbf{W}\right)^{-1}\mathbf{J}^{\mathrm{T}}\mathbf{J}$$

Slight better results for extended structures with L2 norm.



Conclusions

Optimised arrays generated by maximising the model resolution have significantly better resolution and greater depth of investigation than conventional arrays for 2D resistivity surveys.

Computer time required to generate the optimised data sets is greatly reduced by using matrix-matrix multiplication algorithms and the parallel processing capabilities of modern CPUs.

Computer time is further reduced by using the graphics card GPU as a highly parallel mathematical coprocessor.

The fast methods makes it practical to calculate optimised arrays for longer 2-D lines on common PCs.

The resolution for small data sets can be significantly improved by using a smaller step size for adding the new configurations in the iterative algorithm used to generate the optimised data sets.