

Versamedium

Versamedium Web Services: White Paper

Joel E. Rodríguez-Ramírez.

joel@versamedium.com

<http://www.versamedium.com/>
1630 Big Dipper Way, San Ysidro CA. 92173. U.S.A.

January, 2010.

Versamedium Web Services: White Paper.

Joel E. Rodríguez-Ramírez
Versamedium Web Services
1630 Big Dipper Way. San Ysidro CA. 92173. U.S.A.
E-mail: joel@versamedium.com

Abstract

A Internet web service is presented that solves *ill-posed* problems. The service is capable of solving complex linear and nonlinear problems in 1,2 or 3 dimensions. In order to find a solution, the service uses a variation of the well known regularized least squares method implemented as a dynamical system based in Tank & Hopfield [20]. The initial condition of the dynamics is represented by a model's arbitrary or "recycled" configuration states. After the dynamics has taken place, the final configuration states represents the solution of the optimization problem. Solutions through time are regularized dynamically in the evolving model by means of *local averaging* together with a β *weighting parameter*. Detailed instructions for the correct use of the web service, and two examples from the geophysical sciences both in 1-D (nonlinear) and 2-D (linear) are presented as well.

I. INTRODUCTION:

The Versamedium Web Services solves linear algebraic systems such as the one shown below:

$$\begin{bmatrix} d_1 \\ d_2 \\ \cdot \\ \cdot \\ d_M \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1(N-1)} & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2(N-1)} & a_{2N} \\ & & \cdot & \cdot & \cdot \\ & & \cdot & \cdot & \cdot \\ a_{M1} & a_{M2} & \cdots & a_{M(N-1)} & a_{MN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdot \\ \cdot \\ \cdot \\ x_{(N-1)} \\ x_N \end{bmatrix}. \quad (1)$$

If we rewrite the system in vector form, we have:

$$\vec{d} = \vec{A} \cdot \vec{x}, \quad (2)$$

basically, a remote user can solve a problem by sending the matrix \vec{A} and data vector \vec{d} through the Internet to our servers. After some iterations, the user receives a vector solution \vec{x}^* .

But the kind of problems that can be solved by the web services is constrained. Namely, to those in which there are more unknowns in column vector \vec{x} than data points in column vector \vec{d} . Therefore matrix \vec{A} has more columns than rows (*i.e.* $N > M$).

Note that, due to the fact that we have more unknowns than data in $\vec{d} = \vec{A} \cdot \vec{x}$, the web services can not directly compute \vec{A}^{-1} (inverse of \vec{A}) and this way have our system solved for \vec{x} as in:

$$\vec{A}^{-1} \cdot \vec{d} = \vec{x}, \quad (3)$$

Therefore some considerations must be taken into account for such a rather “unusual” system to deliver a solution. Our web services uses a variation of what is known as *regularized least-squares* method, to find an approximate solution out of the endless number of possible solutions.

A more detailed explanation of the computational process is found in section II (Understanding better the file structure). But basically the servers receive a set of 16 parameters, the matrix elements of \vec{A} , the data points \vec{d} and initial condition \vec{x}_0 . Then the web service goes through a series of computing *iterations* based in the 16 parameter values, the web service returns partial solutions ($\vec{x}_{iteration}^*$ *iteration* = 1, 2, 3, ..) at each iteration. Some of these solutions will generate “responses” $\hat{\vec{d}}_{iteration}^*$ who has best fitting errors with respect to the original data \vec{d} .

In the process, a model solution \vec{x}^* is obtained with the best response $\hat{\vec{d}}^*$ *i.e.* that best resembles the original data \vec{d} .

Observe that for the above considerations that describes the algebraic problem it is not possible to send a randomly generated matrix system to the web service and expect a solution. This is why some **essential** considerations must be taken into account.

A. Some definitions:

Consider the following example in which we “brute force” a solution. Namely that in which we generate random models \vec{x} , then calculate the forward *response* $\hat{\vec{d}}$ and obtain its misfit *rms error* with respect to the original data \vec{d} :

$$rms \ error \ (percent \ \%) = \sqrt{\frac{\sum_{i=1}^M \left(\frac{d_i - \hat{d}_i}{d_i}\right)^2}{M}} \cdot 100.0, \quad (4)$$

we then proceed by storing the models \vec{x} with the small rms error.

Eventually (with some patience), we will end up with a “good” solution in terms of having $\vec{d} - \hat{\vec{d}} \approx \vec{0}$. Despite that the above brute force solution might not have any physical meaning, it shares common ground with our “desired” solutions, namely that of having the sum of all differences approach zero, either from below zero or from above it. This is, we want *the residuals* approach zero as in:

$$\sum_{i=1}^M (d_i - \hat{d}_i) \approx 0. \quad (5)$$

In order to account for both negative or positive sum of differences, we might re-state our problem as that of having the following *target function* C approach zero.

$$C = \frac{1}{2} \sum_{i=1}^M (d_i - \hat{d}_i)^2, \quad (6)$$

that will be our *least-squares* goal. Re-writing the above equation in vectorial notation we have:

$$C = \frac{1}{2} (\vec{\mathbf{d}} - \vec{\mathbf{A}} \cdot \vec{\mathbf{x}})^T \cdot (\vec{\mathbf{d}} - \vec{\mathbf{A}} \cdot \vec{\mathbf{x}}), \quad (7)$$

B. Least Squares:

From *Calculus* concepts of *maxima and minima*. We can take the derivative the above function with respect the model $\vec{\mathbf{x}}$ and by equating to zero, then we have our *minimal* value of C for values of $\vec{\mathbf{x}}$ (see Appendix A : Least Squares).

$$\vec{\mathbf{x}} = (\vec{\mathbf{A}}^T \cdot \vec{\mathbf{A}})^{-1} \cdot \vec{\mathbf{A}}^T \cdot \vec{\mathbf{d}}, \quad (8)$$

in which we note that the existence of the solution $\vec{\mathbf{x}}$ for $M > N$ depends upon the existence of $(\vec{\mathbf{A}}^T \cdot \vec{\mathbf{A}})^{-1}$.

C. Regularized Least Squares:

Strictly speaking (from the algebraic point of view), the case for which $N > M$, for the linear system in equation (1), might not have a unique solution because $\vec{\mathbf{A}}^T \cdot \vec{\mathbf{A}}$ being singular, therefore lacking a computational inverse.

One way for solving these kind of *ill-posed* problems, in which no unique solution exists, because in effect, there is not enough information specified in the problem, was developed by A. Tikhonov [21]. The mathematical technique that Tikhonov developed for this, is known as *regularization*. It incorporates extra information (or assumptions, possibly from the physical world) that makes the algebraic system “less singular”.

So, by incorporating such “extra information” from physical matter properties in equation (1). We could expect to obtain meaningful solutions for the algebraic problem (see Appendix B: A physics example). This way, we might have an *inverse* algebraic solution which at the same time is, physically meaningful.

It is a common practice to introduce regularization as a extra term in the target function in equation (6) known as the Tikhonov matrix $\vec{\mathbf{\Gamma}}$. Considering the new term we have:

$$C = \frac{1}{2} \sum_{i=1}^M (d_i - \hat{d}_i)^2 + \|\vec{\Gamma} \cdot \vec{x}\|^2, \quad (9)$$

where $\|\cdot\|$ stands for the Euclidean norm.

Rewriting (9) in vectorial notation:

$$C = \frac{1}{2} \|\vec{d} - \vec{A} \cdot \vec{x}\|^2 + \|\vec{\Gamma} \cdot \vec{x}\|^2, \quad (10)$$

taking the derivative of the above function with respect to model \vec{x} and equating the result to zero, we can achieve the solution (see Appendix C: Regularized Least Squares).

$$\vec{x} = (\vec{A}^T \cdot \vec{A} + \vec{\Gamma}^T \cdot \vec{\Gamma})^{-1} \cdot \vec{A}^T \cdot \vec{d}. \quad (11)$$

In many cases the matrix $\vec{\Gamma}$, is chosen as the identity matrix $\vec{\Gamma} = \beta \vec{I}$, giving preference to solutions with smaller norms.

$$\vec{x} = (\vec{A}^T \cdot \vec{A} + \beta \cdot \vec{I})^{-1} \cdot \vec{A}^T \cdot \vec{d}. \quad (12)$$

In other cases, high-pass operators (e.g., a difference operator or a weighted Fourier operator) may be used to enforce smoothness if the underlying \vec{x} vector which is believed to be mostly continuous. This regularization improves the *conditioning* of the problem, thus enabling a numerical solution.

In equation (12) we note that for $\beta > 0$ the solution will exist provided that $(\vec{A}^T \cdot \vec{A} + \beta \cdot \vec{I})$ is non-singular. The effect of regularization may be varied via the scale of matrix $\beta \vec{I}$. Therefore the inverse solution to the problem might exist for a suitable regularizing value of β . For $\beta = 0$ this reduces to the unregularized least squares solution provided that $(\vec{A}^T \cdot \vec{A})^{-1}$ exist.

D. The Web Service File Structure:

When data is transmitted to our servers via the client program¹. It does so by sending the file with a properly ordered data structure. Our servers will only understand files with this structure. Failing to provide the correct structure will end the service by closing the communication channel. The file structure is composed of:

- 16 Header parameters.
- Matrix \vec{A} stored in row vector order.
- Data \vec{d} vector.
- Initial model \vec{x}_0 vector.

¹see <http://www.versamedium.com/geoinverse/start.html>.

We describe briefly the file structure values here. For a detailed explanation see (Section II. Understanding better the file structure).

NOMROWS : The matrix (integer) number of rows.

NUMCOLS : The matrix (integer) number of columns.

MODELMIN : Lower bound (floating point) number that describes where solutions are to be found. There will be no solution below this limit.

MODELMAX : Upper bound (floating point) number that describes where solutions are to be found. There will be no solution above this limit.

SCHEME : Value (integer) that specifies the dimensionality of the problem.

DISCX : Value (integer) that describes discretization in X (Section II. Understanding better the file structure).

DISCY : Value (integer) that describes discretization in Y (Section II. Understanding better the file structure).

DISCZ : Value (integer) that describes discretization in Z (Section II. Understanding better the file structure).

FILTX : Value (integer) that set's the local extension in X, of the regularizing window filter.

FILTY : Value (integer) that set's the local extension in Y, of the regularizing window filter.

FILTZ : Value (integer) that set's the local extension in Z, of the regularizing window filter.

BETA : Regularization (floating point) parameter that is used to ponderate the effect of the regularization window filter (defined above). This value can take values between 0.0 (without regularization) and 1.0 (full regularization).

AMPLITUDE : number of bits for amplitude (integer), used by our computational algorithm when the model values in \mathbf{x} are transformed into a space composed of binary values in the new dynamical system's domain. The **AMPLITUDE** number specifies the binary amplitude $2^{+\text{AMPLITUDE}}$ or "extent" of possible values. Computational time scales directly with this value.

PRECISION : number of bits for precision (integer), used by our computational algorithm when the model values in \mathbf{x} are transformed into a space composed of binary values in the new dynamical system's domain. The **PRECISION** number specifies the binary resolution $2^{-\text{PRECISION}}$ or "granularity" of the possible values. Computational time scales directly with this value.

ITERATIONS : Value (integer) that specifies the recursive number of linear solving minimization steps to be performed.

CODE : Reserved parameter (integer).

Following the 16 header parameters are the matrix \vec{A} the data \vec{d} and the initial condition \vec{x}_0 , which values are composed of single precision floating point² numbers stored in a single column.

The elements of matrix \vec{A} are stored in row order. This is, starting with the first element of the first row, the whole first row of matrix \vec{A} is stored in the file. Then after the last value, we start storing the first element of the second row, and so on.

This way for a matrix with 28 rows and 100 columns. The file will be composed of the 16 header parameters followed by 2928 floating point values. The immediate 2800 floating points (following the 16 header parameters) correspond to the matrix \vec{A} stored in row order. Then there will be 28 data points for \vec{d} , followed by 100 initial model \vec{x}_0 values. Therefore our example file structure will have 2944 rows.

II. UNDERSTANDING BETTER THE 16 HEADER PARAMETERS IN THE FILE STRUCTURE:

The 16 header parameters in the file structure, do instruct the web services how to solve the problem. By notifying the matrix dimensions, the regularizing scheme to be performed on model \vec{x}_0 , how it was discretized. These parameters are described into more detail in the following sub-sections:

A. *The **NOMROWS**, **NUMCOLS**, **MODELMIN** and **MODELMAX** header parameters.*

The first four parameters in the file inform the web service with the matrix dimensions and valid domain for the solutions:

NOMROWS: The matrix (integer) number of rows, It is also the number of data points.

NUMCOLS : The matrix (integer) number of columns. It is also the number of model \vec{x} unknowns.

MODELMIN : Lower bound (floating point) number that describes where solutions are to be found. There will be no solution below this limit. As mentioned before there is an “analog” to digital transformation of the model \vec{x} values and this parameter could be used to narrow the space of solutions in which the attention of the dynamical system gets focused.

MODELMAX : Upper bound (floating point) number that describes where solutions are to be found. There will be no solution above this limit. As mentioned before there is an “analog” to digital transformation of the model \vec{x} values and this parameter could be used to narrow the space of solutions in which the attention of the dynamical system gets focused.

B. *The **SCHEME**, **DISCX**, **DISCY** and **DISCZ** header parameters.*

Equation (1) despite its simplicity can handle higher orders in dimensionality and nonlinearities (indirectly) as well.

The **SCHEME** parameter describes the regularization scheme to be used, currently only 3 schemes are available, those are for 1-D, 2-D and 3-D.

For **SCHEME** = 1 we have that 1-D dynamical regularization is used, and the way discretization of model takes place is unique **DISCX** = **NOMCOLS**. Assigning all the model data values in \vec{x}

²IEEE 754 Standard Floating-Point Arithmetic.

sequentially to a 1-D virtual string array to be regularized by local averages (see next subsection) in the X-dimension only. The values $\text{DISCY} = \text{DISCZ} = 0$ (should be zero).

For $\text{SCHEME} = 2$ we have that 2-D dynamical regularization is used, and the way discretization of the model takes place can vary with possible combinations of DISCX and DISCY such that $\text{DISCX} \cdot \text{DISCY} = \text{NOMCOLS}$. Which are the values in which your 2-D model was discretized. Under this SCHEME The model data values in \vec{x} are assigned sequentially to a 2-D virtual mesh array in order to be regularized by a local *rectangular shaped averaging window* among near neighbors (see next subsection), this happens in the X-Y plane. The value $\text{DISCZ} = 0$ (should be zero).

For $\text{SCHEME} = 3$ we have that 3-D dynamical regularization is used, and the way discretization of model takes place can vary by any possible combinations of DISCX , DISCY and DISCZ such that $\text{DISCX} \cdot \text{DISCY} \cdot \text{DISCZ} = \text{NOMCOLS}$. Which are the values in which your 3-D model was discretized. Under this SCHEME The model data values in \vec{x} are assigned sequentially to a 3-D virtual cell array in order to be regularized by a local *cuboid shaped averaging body* among near neighbors (see next subsection), this happens in the X-Y-Z space.

C. The FILTX , FILTY , FILTZ and β regularizing header parameters .

As barely described in the previous sections, the web services relies in a regularizing scheme which locally averages values among near neighbors, then this value is ponderated by value β which will give a specific influence to the window average filter. We will go into more detail in the following subsections:

1) **True averages:** It has long been recognized that computing spatial averages of physical properties is a powerful tool for addressing the non-uniqueness character of inverse problems. In general, the aim of obtaining the distribution of the property itself remains an elusive target. This is because the space of models always have a higher dimension than the space of data. Asking for spatial averages instead of the property itself, is a more realistic goal that leads to unique recoveries. We use the averaging approach of Backus and Gilbert [1], for solving the problem within the approximation. Averages solve the non-uniqueness problem because they are unique themselves, and because all possible models that fit the data, all have the same computed averages. They can be called true averages because they are spatial averages of any possible solution, including the true or real.

2) **Simulated averages:** Our inversion method is based in this averaging notion. And the specifics of the method could be found elsewhere (see for example Rodriguez J. *et al.* [17]). At the same time it is customary to say that our method accounts for averages in higher dimensions in a natural way. The computed solutions models are everywhere averages of themselves within a given local neighborhood, regardless of dimensionality.

In order to get insightful view of the inner-workings of the inversion method, lets consider the following basic 1-D *binary* example in which model values are composed of 0 or 1 only.

Regularization is performed via the filter acting in a local neighborhood of the model j value. More specifically, we have original value $x_j^{(t)}$ at certain time t substituted by its local average as in:

$$x_j^{(t)} = \sum_{l=-\text{FILTX}}^{\text{FILTX}} w_l x_{(j+l)}^{(t)}. \quad (13)$$

Where the filter has $2 \text{FILTX} + 1$ constant weights³ $w_l = \frac{1}{2 \text{FILTX} + 1}$.

Then a new value $x_i^{(t+1)}$ is computed. Say by means of the following updating rule, that in fact is a dynamical system based in Tank & Hopfield [20], that we use for demonstration purposes:

$$x_i^{(t+1)} = \frac{1}{2} + \frac{1}{2} \text{sgn} \left\{ \sum_{j \neq i=1}^{\text{NUMCOLS}} A_{ij} \left[\sum_{l=-\text{FILTX}}^{\text{FILTX}} w_l x_{(j+l)}^{(t)} \right] + \text{constant} \right\}. \quad (14)$$

The updated values $x_i^{(t+1)}$ are obtained from averages of the original values $x_j^{(t)}$. Although it is not possible to speak about a smoother solution in the above dynamics, as the new model values are binary, the effect of having a wider filter will be to smooth the solution model. See for example Figures 1 and 2 in which we have applied a variation of the above regularizing averaging scheme in non-binary models. We can appreciate the effect of using three different filter widths, acting on some geophysical model.

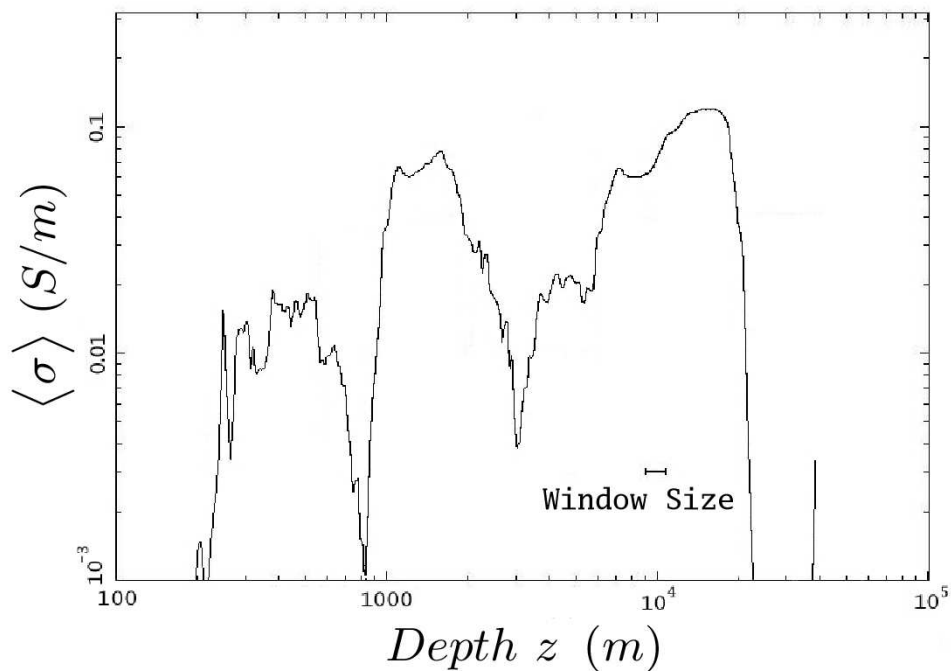


Fig. 1. Obtained geophysical model solution, when using 500 values ($\text{NUMCOLS} = 500$), and the described averaging technique with a) $\text{FILTX} = 1$ (two neighbors and the value itself).

³The longitudinal window size value remain constant during the dynamics. In the future we will implement more sophisticated *schemes* in which FILTX can vary. The theoretical scope of these new schemes is beyond the context of this paper.

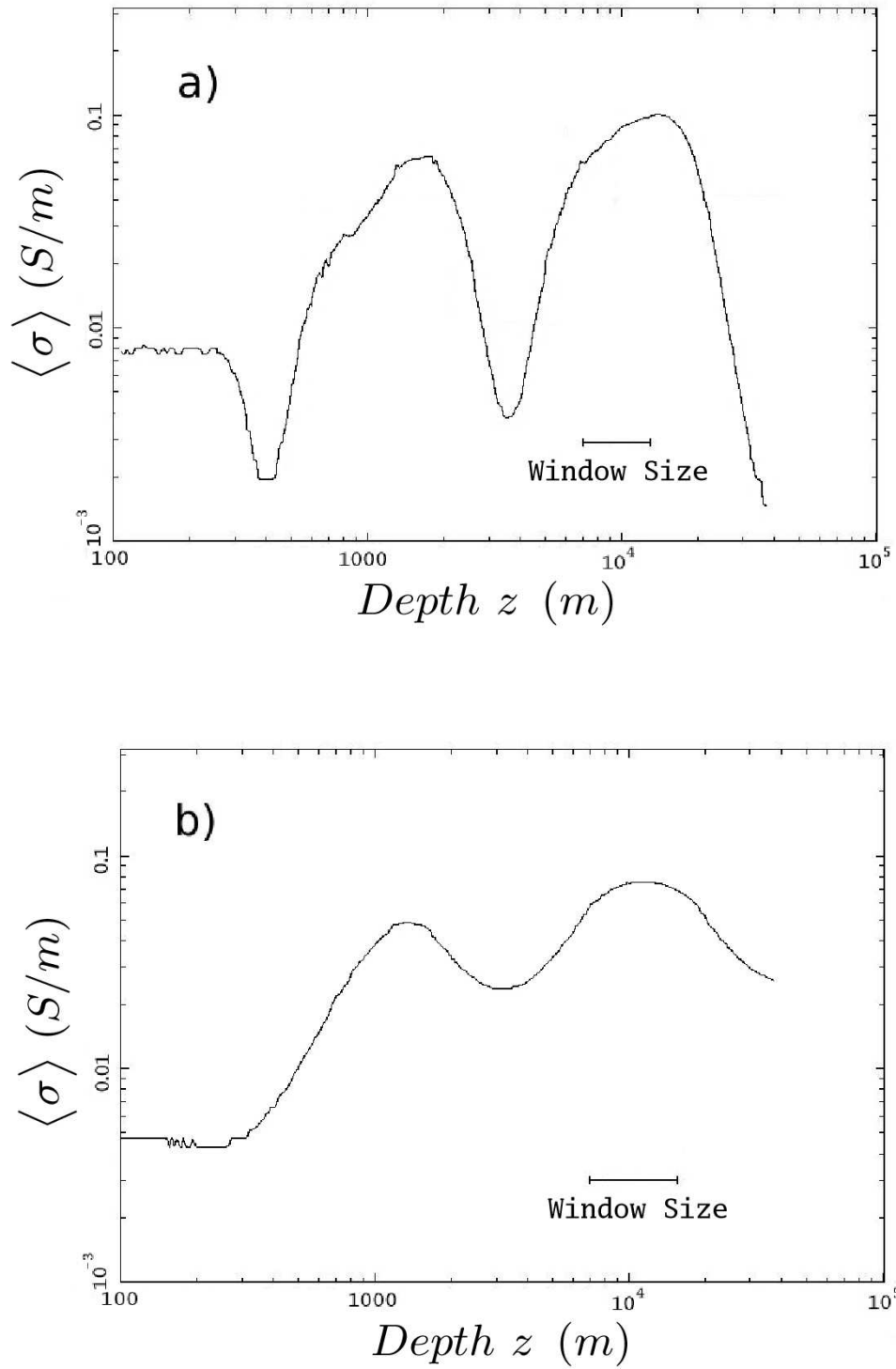


Fig. 2. Obtained geophysical model solution, when using 500 values (`NUMCOLS` = 500), and the described averaging technique with a) `FILTX` = 12 (twenty-two neighbors and the value itself). b) `FILTX` = 23 (forty-six neighbors and the value itself).

Although our method does not introduces a Tikhonov regularizing matrix $\vec{\Gamma}$, we do account for the “effectiveness” of the applied local filter (regularizing window), by introducing the β parameter. This parameter enable us to “tweak” or “stress-test” how singular $\vec{A}^T \cdot \vec{A}$ becomes with the new parameters. At the same time the β parameter accommodates the trade-off between two contrasting features in the physical sense. We then rewrite our new *binary* demonstration dynamical system as:

$$x_i^{(t+1)} = \frac{1}{2} + \frac{1}{2} \text{sgn} \left\{ \sum_{j \neq i=1}^{\text{NUMCOLS}} \left[(1 - \beta) A_{ij} x_j^{(t)} + \beta \sum_{l=-\text{FILTX}}^{\text{FILTX}} A_{ij} w_l x_{(j+l)}^{(t)} \right] + \text{constant} \right\}. \quad (15)$$

When $\beta = 1.0$ we get the full filtered solution. On the other hand, for $\beta = 0.0$ we return to the original unfiltered solution⁴.

The above oversimplified version, can be extended into more complex models [20][23][24]. This is, we can extend representation of the model values to arbitrary digital values. This is accomplished by the use of more bits in the model values representation, both in amplitude and precision. That will be the subject of the following subsection.

D. The **AMPLITUDE**, **PRECISION**, **ITERATIONS** header parameters.

As shown in the previous section. In order to find an \vec{x} solution. The initial model (with NUMCOLS unknowns) values in \vec{x}_0 are **linearly transformed** into configuration states in a dynamical system. But those new values will be limited both in amplitude and precision. The new unknown model values will be limited by $2^{\text{AMPLITUDE}}$ and $2^{-\text{PRECISION}}$, in amplitude and precision respectively. This way, if we have **AMPLITUDE** = 9 and **PRECISION** = 9 the new domain will be a $\Delta x = 0.00195$ discretized domain (2^{-9}) restricted to $[-512.00195 : +512.00195]$.

The **ITERATIONS** parameter is used to limit the number of *epochs* in which our dynamical system updates all the unknown states. Note, that in each iteration a solution is obtained. And every time you perform an iteration the web service returns a model and an *rms error* for that particular iteration, to the remote running **versa-client** program (see <http://www.versamedium.com/geoinverse/start.html>).

E. The **CODE** header parameter.

The **CODE** parameter is used by our web servers to keep track of the solutions and to limit the Internet bandwidth. We have released 2 codes for demonstration purposes:

The code **1234567890**, has been released to tackle problems in 1-D. And has implied restrictions. Currently: **SCHEME** = 1, **NUMCOLS** = 200, **AMPLITUDE** = 9, **PRECISION** = 9 and **ITERATIONS** = 5. This code is shown to solve complex 1-D nonlinear problems⁵ (see section Example 1 : The Vertical Electric Sounding (nonlinear) problem).

The code **1231231231**, has been released to tackle problems in 2-D. And has implied restrictions. Currently: **SCHEME** = 2, **NUMCOLS** = 2520, **AMPLITUDE** = 9, **PRECISION** = 9 and **ITERATIONS** = 5. This code is shown to solve complex 2-D Linear problems⁶ (see section Example 2. : The Magnetotelluric Electromagnetic Sounding Linear problem).

⁴At the same time we can extend the dynamical averaging regularization to 2-D or 3-D.

⁵also see <http://www.versamedium.com/geoinverse/versa-ves.html>.

⁶also see <http://www.versamedium.com/geoinverse/versa-mt.html>.

III. EXAMPLE 1 SIMULATED AVERAGES IN 1-D: THE VERTICAL ELECTRIC SOUNDING (NONLINEAR) PROBLEM.

Vertical electric soundings constitutes a valuable tool in determining the electric properties of the subsoil. Electric currents are injected in the subsoil through punctual electrodes and then, voltage is measured usually in the center of the electrode array as shown in Figure 3 (Schlumberger electrode configuration).

Surface resistivity measurements using direct current (DC) have been used traditionally to investigate changes of formation resistivity with depth, from typical depths of a few meters to several kilometers [10]. The interpretation or inverse problem consists of recovering a one-dimensional (1D) depth resistivity distribution from surface voltage measurements taken at different current electrode separations. Several methods have been developed in which the resistivity distribution is handled as a continuous or practically continuous function, so that the resulting variations automatically define zones of high and low resistivity that follow the vertical resistivity profile of a sedimentary basin. In this case the layer boundaries are guessed on the basis of the smooth, continuous profile.

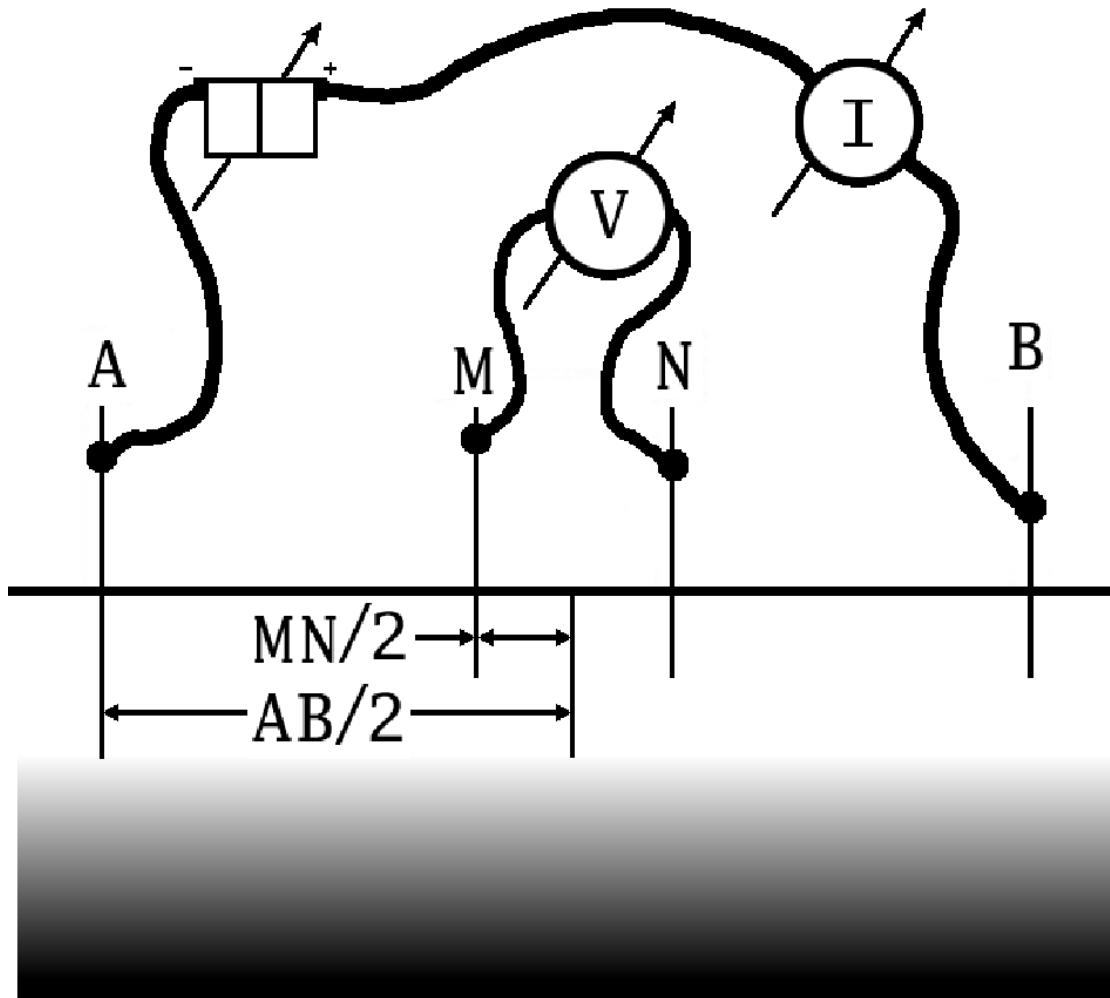


Fig. 3. Electric current is injected into the subsoil through the I current electrodes and then differences in electric potential V are measured through electrodes in the center of the array (Schlumberger electrode configuration).

Once the measurements have been obtained for different values of current electrodes separation ($AB/2$), we then calculate what is called an *apparent resistivity curve*, see Figure 4.

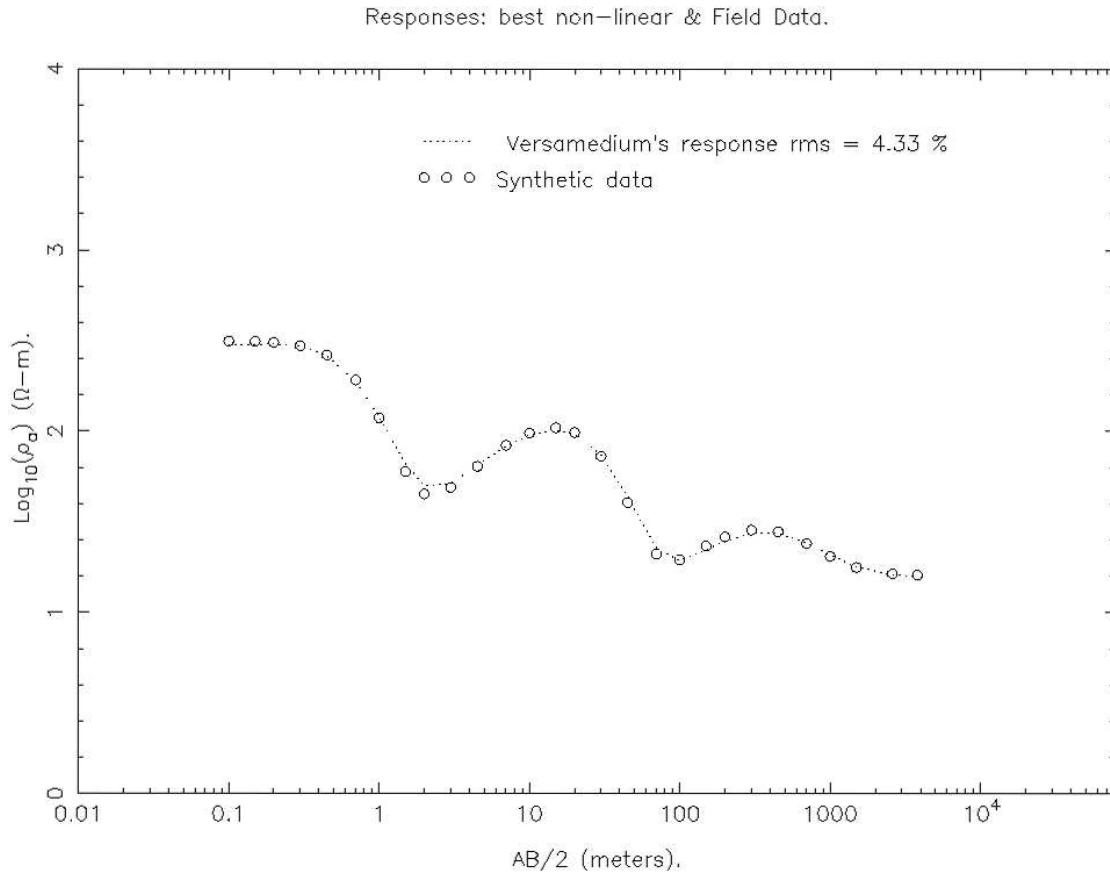


Fig. 4. Apparent resistive curve generated by the measure of Voltage V and known injected current I (depicted by circles). This figure also shows a calculated models response (dashed line).

For a detailed explanation of the theory behind the linearization technique see Appendix D (The nonlinear problem for Vertical Electrical Soundings (VES)). We now turn to a demonstration program in which the Versamedium Web Services solves this VES inverse problem, see <http://www.versamedium.com/geoinverse/versa-ves.html>.

The Internet enabled demonstration program **versa-ves**, step-linearize the problem. This way we pursue the nonlinear solution through linear approximations. For every “approaching step”, a linear system (equation 1) is generated and sent to our web servers via the **versa-client** program. In return the client user receives a set of partial solutions which should have decreasing *linear rms error* with each linear iteration. Although this does not implies that those solutions are the ones with the minimum *nonlinear rms error*. Therefore another *nonlinear rms error* calculation is performed with a nonlinear functional (direct computation) in the side of the remote user (client). By calculating these responses from the returned linear partial model solutions (**versamedium_partial_sol_#.dat**), the model with the minimum *nonlinear error* is then stored and used to recalculate a derivative matrix (See Appendix D), which is then sent via the **versa-client** program (again) to the web services. This is, the process described above is recursively repeated a no-linear number of iteration times. Each nonlinear iteration is composed of a number of linear **ITERATIONS**.

After running the program **versa-ves**, with the Schlumberger array data, a matrix system (equation 1) is generated and stored in say for example the **test-matrix.dat** file⁷.

We now show an extract of the output log, when running the **versa-ves** program, which should be self explanatory:

```
Type the name of VES data file: (default = demo.dat):
Data file Name is demofile: demo.dat
  AB/2 (meters)          Apparent Resistivity (Ohm-meters)
1.000000e-01           3.153649e+02
1.500000e-01           3.133988e+02
2.000000e-01           3.097840e+02
3.000000e-01           2.967249e+02
4.500000e-01           2.635422e+02
7.000000e-01           1.918084e+02
1.000000e+00           1.189949e+02
1.500000e+00           5.991498e+01
2.000000e+00           4.506209e+01
3.000000e+00           4.910503e+01
4.500000e+00           6.419183e+01
7.000000e+00           8.381851e+01
1.000000e+01           9.786901e+01
1.500000e+01           1.045534e+02
2.000000e+01           9.839312e+01
3.000000e+01           7.279412e+01
4.500000e+01           4.046773e+01
7.000000e+01           2.102559e+01
1.000000e+02           1.953480e+01
1.500000e+02           2.327704e+01
2.000000e+02           2.609296e+01
3.000000e+02           2.853105e+01
4.500000e+02           2.792675e+01
7.000000e+02           2.405656e+01
1.000000e+03           2.044544e+01
1.500000e+03           1.774854e+01
2.600000e+03           1.638006e+01
3.800000e+03           1.608359e+01
How many layers do you want in model (minimum 100)? : (default = 100):
Your number of layers are: 100
1-D regularization window size?: (default = 2):
1-D regularization window: 2
Regularization beta parameter? ( 0.0 <= beta <= 1.0 )?: (default = 1.000000):
Your beta parameter is: 1.000000
Solutions will have maximum=800.000000
Solutions will have minimum=0.100000
Type the number of nonlinear iterations? : (default = 5):
Number of nonlinear iterations: 5
Type the number of linear iterations? : (default = 10):
Number of linear iterations: 10
Would you like to recycle a previous solution (1 = Yes)? : (default = 0 No):
Recycle: 0
Graphics device/type (? to see list, default /NULL): /PS
```

We describe here one of “those” **linear** approximation steps. The file that has been generated includes a matrix system that has 28 rows and 100 columns. So there will be 16 header parameters followed by 2928 floating point values. As explained in the previous section, the immediate 2800 floating points

⁷both the versa-ves source code and the test-matrix.dat file are included in <http://www.versamedium.com/geoinverse/distro-versa-ves-0.1.tar.gz>

following the 16 header parameters, correspond to the matrix stored in row order (first row vector followed by the second row vector, and so on), following then, are 28 data points followed by 100 initial model values. At the end we will have a file which has 2944 rows. We now show the header parameters generated by the **versa-ves** program:

```
file: test-matrix.dat

28          # NOMROWS: Matrix number of rows or data points.
100         # NOMCOLS: Matrix number of columns or number of model values.
0.1        # MODELMIN: Minimum allowed value for the x model values.
800.0      # MODELMAX: Maximum allowed value for the x model values.
1          # SCHEME: 1-D.
100        # DISCX: Discretization in dimension-X.
0          # DISCY:
0          # DISCZ:
2          # FILTX: Nearest neighbors to consider in average dimension-X.
0          # FILTY: Nearest neighbors to consider in average dimension-Y.
0          # FILTZ:
1.0        # BETA: Regularization weighting parameter.
9          # AMPLITUDE:
9          # PRECISION:
5          # ITERATIONS:
1234567890 # CODE :
6.095135e-02 # Matrix value
1.559382e-02 # Matrix value
1.924179e-02 # Matrix value
2.355296e-02 # Matrix value
2.854579e-02 # Matrix value
3.418030e-02 # Matrix value
.          # Matrix value
.          # Matrix value
.          # Matrix value
```

Once the linear system file has been generated, the **versa-ves** program invokes the **versa-client** program (in the “background”) in order to send the test-matrix.dat to our servers and to receive the desired solutions as well. The process can be described by the following command:

```
./versa-client www.versamedium.com 8888 0 test-matrix.dat
```

Where **www.versamedium.com** is the Internet address , **8888** is the port, the **0** value stands for a new solution calculation (not recycled) and the **test-matrix.dat** is the name of the file with the matrix system data.

The **versa-ves** program hinders the above instruction from the user, but this process is performed in a automatized recursive way, through all the linear approximation-steps described above in order to obtain the nonlinear solution. Then after some computations the web services deliver a model solution to the remote user via the **versa-client** program. The resulting solution model when inverting the data presented in Figure 4 is shown graphically in Figure 5.

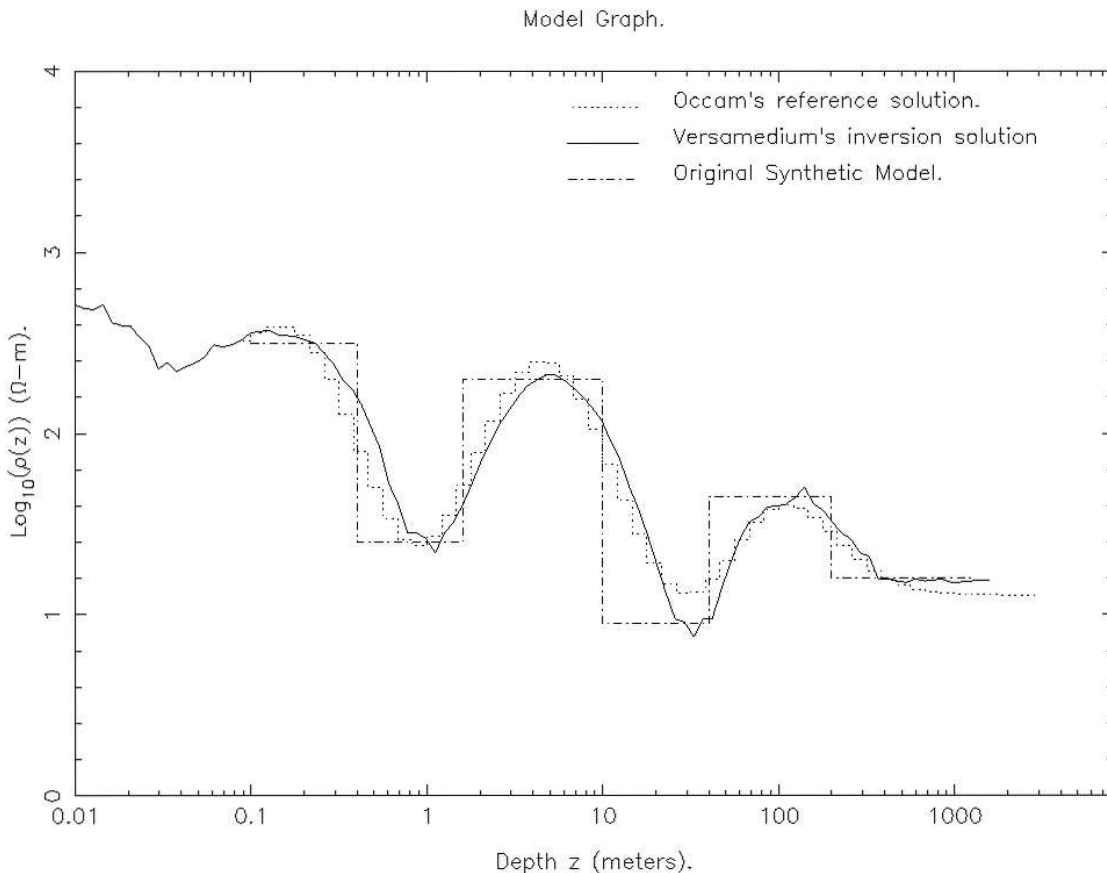


Fig. 5. This figure shows a model solution obtained by the Versamedium web services together with the synthetic model that was used to generate the data points in Figure 4. A reference Occam's solution model [4] is also presented for comparison purposes.

Then by diminish the window sizes and keeping $\beta = 1.0$ we re-calculate a new set of 5 **nonlinear** iterations. This time we only send the header values. We encourage the reader to test by himself (see <http://www.versamedium.com/geoinverse/versa-ves.html>).

IV. EXAMPLE 2. SIMULATED AVERAGES IN 2-D: THE MAGNETOTELLURIC ELECTROMAGNETIC SOUNDING LINEAR PROBLEM.

The Versamedium Web Service can also solve problems in 2 or 3 dimensions. Consider a real case, in which a complex nonlinear problem in 2-D has been linearized and represented in a matrix system similar to equation (1).

In the magnetotelluric method, surface measurements of natural time-varying electric and magnetic fields are readily converted to four complex impedance values per given angular frequency ω . In turn, these values are usually normalized to obtain apparent resistivities or, equivalently, apparent conductivities, by referring the actual impedances to those of a homogeneous half-space [3]. Here we use apparent conductivity σ_a as derived form the magnitude of a complex impedance Z , which for the moment represents any of the four elements of the impedance tensor. The formula for apparent conductivity is simply given as:

$$\sigma_a(x, \omega) = \omega \mu_0 |Z(x, \omega)|^{-2}, \quad (16)$$

where μ_0 stands for the magnetic permeability of free-space, and x represents horizontal distance in a x - z coordinate system whose z axis represents depth. $\sigma_a(x, \omega)$ represents the data at a given distance in a 2-D model with a flat topography and for a given angular frequency ω . The data are usually presented as individual sounding curves as function of period $T = \frac{2\pi}{\omega}$ for different distances x , or in a pseudo-section

format contouring values of σ_a as plotted over $x - T$ coordinates. $\sigma_a(x, \omega)$ represents what is available; what is required is $\sigma(x, z)$, the subsurface conductivity distribution.

The underlying **nonlinear** problem has been linearized **in just one “approximation” step!**, therefore there will be no need to recursively recalculate a derivative matrix as in the previous example. The details of the method can be found in Rodriguez J. *et al.* [17][18].

We perform a 2-D regularization averaging scheme, by using rectangular filter windows **FILTX**, **FILTY** and β . The data corresponds to the well known COPROD2 data set [9]. And our proposed solution has a discretization in space as a mesh array composed of 36 horizontal rows by 70 vertical columns or 2520 model cells. The data is composed of the apparent conductivity measured in several stations for every magnetotelluric T periods⁸. As not all the periods per station were considered. The data set was composed of 730 data points (See chapter V, for further important considerations on the discretization scheme).

Once the problem has been linearized a file with the matrix system is generated. The 16 header parameters used in this 2-D example are the following:

```
versamedium_coprod_2D-P.dat

730      # NOMROWS: Matrix number of rows or data points.
2520     # NOMCOLS: Matrix number of columns or number of model values.
0.0001  # MODELMIN: Minimum allowed value for the x model values.
0.90    # MODELMAX: Maximum allowed value for the x model values.
2       # SCHEME: 2-D.
70      # DISCX: Discretization in dimension-X.
36      # DISCY: Discretization in dimension-Y.
0       # DISCZ:
2       # FILTX: Nearest neighbors to consider in average dimension-X.
2       # FILTY: Nearest neighbors to consider in average dimension-Y.
0       # FILTZ:
0.5     # BETA: Regularization weighting parameter.
9       # AMPLITUDE:
9       # PRECISION:
5       # ITERATIONS:
1231231231 # CODE :
```

We will apply the web service algorithm to *invert* the field data. More instructions to replicate these results can be found in <http://www.versamedium.com/geoinverse/versa-mt.html>.

Figures 6(a) to 6(e) show the results of inverting the data. Decreasing values of β from 1.0 shows to improve the rms misfit and the resolution improve for smaller window size. We can see that when using the smallest window, allows to identify the two main anomalies found in other works. The strongest conductive anomaly in Figure 6(d) corresponds to the North American Central Plains (NACP) conductivity anomaly, which has been well discussed in the literature [9]. The smaller lobular anomalies towards the right end of the profile corresponds in turn to the known Thomson Nickel Belt (TOBE) anomaly.

⁸Actually it is more complicated than this, please refer to [17] for details.

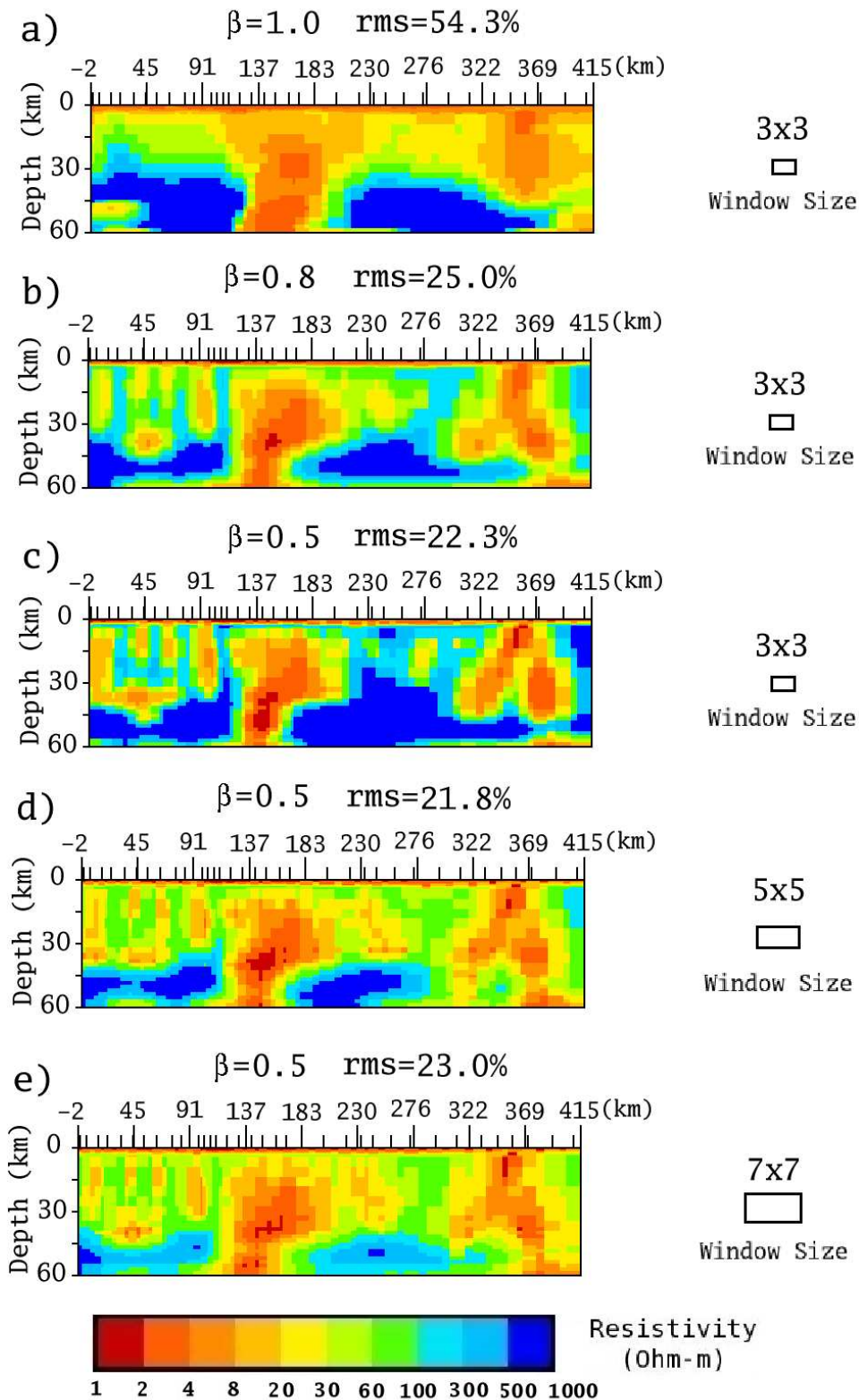


Fig. 6. Results obtained by applying the Versamedium web service to a 2-D real data set COPROD2 (Jones, 1993a) window size and β values where varied. a) For **FILTX=1**, **FILTY=1** (window size 3x3) and $\beta = 1.0$. b) For **FILTX=1**, **FILTY=1** (window size 3x3) and $\beta = 0.8$. c) For **FILTX=1**, **FILTY=1** (window size 3x3) and $\beta = 0.5$. d) For **FILTX=2**, **FILTY=2** (window size 5x5) and $\beta = 0.5$. e) For **FILTX=3**, **FILTY=3** (window size 7x7) and $\beta = 0.5$.

Note the number of values to be found in model \vec{x} are $36 \cdot 70 = 2520$. Which is the number of rows in the matrix. This is, for this example $\text{DISCX} \cdot \text{DISCY} = \text{NOMROWS}$.

We also have that **FILTX** and **FILTY** along with β are now required.

A good strategy is to start with “big” values for the regularizing window sizes, say **FILTX** = 4, **FILTY** = 4 and $\beta = 1.0$ then send the file to our servers and calculate a set of 5 linear **ITERATIONS** by means of:

```
./versa-client www.versamedium.com 8888 0 versamedium_coprod_2D-P.dat
```

then by diminish the window sizes and keeping $\beta = 1.0$ we re-calculate a new set of 5 iterations. This time we only send the header values. To do this we set the value 1 before the matrix system file name and use the number of the last best linear solution in order to “recycle” the initial condition (in this case `versamedium_sol_627.dat`).

```
./versa-client www.versamedium.com 8888 1 versamedium_coprod_2D-P.dat 627
```

Repeat the above process as needed in order to obtain a satisfactory result with minimum rms error. The graphics in Figures 6(a) to 6(e), were obtained in a process of roughly 25 linear iterations, in sets of 5 linear iterations as outlined above, with **AMPLITUDE** and **PRECISION** fixed to 9 bits. Every graph was obtained in a lapse of 30 seconds for a set of 5 **ITERATIONS**. In other words, all the graphics in Figures 6(a) to 6(e) were obtained in less than 3 minutes. Once the 24.9 Megabytes of the file containing the headers, matrix, data and initial model, were already in our servers.

V. IMPORTANT CONSIDERATIONS.

The matrix in the linear system (equation 1) that relates both domains *apparent conductivity* and subsoil *real conductivity* in the MT (same consideration for VES) examples **does not amplifies**, it is verifiable by noting that the addition of any row elements in matrix \vec{A} equates 1. Of course any problem could previously be arranged in order to meet this “restriction”.

As you might have noted, Versamedium web services do regularized by taking in account nearest neighbors in a discretized model *i.e.* by using **DISCX** and **DISCY**, but the web services do not know anything about the real model nor it has any idea of the physical kernel underlying in the linear system. This is, for the web service, everything occurs without any consideration on the units involved.

In the side of the remote client user although, the model is actually discretized with real physical variables. For example, in order to generate a physical meaningful kernel from section IV above (Example 2). The model was discretized with the following values for dimension-X (surface) and dimension-Y (depth- z^9) in meters.

```
X = {-1683.0, 1693.0, 8446.0, 11823.0, 17636.0, 20073.0, 26223.0, 29936.0, 36916.0,
40183.0, 46566.0, 49683.0, 56370.0, 59940.0, 67710.0, 71910.0, 78736.0, 81363.0,
88723.0, 93456.0, 99643.0, 101096.0, 104316.0, 106083.0, 109266.0, 110683.0,
115166.0, 118233.0, 124600.0, 127900.0, 134803.0, 138406.0, 145023.0, 148036.0,
154830.0, 158610.0, 165276.0, 168163.0, 175216.0, 179383.0, 189470.0, 195390.0,
207783.0, 214256.0, 224713.0, 228696.0, 235946.0, 239213.0, 247560.0, 252640.0,
262933.0, 268146.0, 280390.0, 287420.0, 300350.0, 306256.0, 318766.0, 325373.0,
```

⁹Our web services are mean to be generic friendly with other non-geophysical applications of course.

336653.0, 341326.0, 350063.0, 354126.0, 362373.0, 366556.0, 377590.0, 384440.0, 397233.0, 403176.0, 415063.0, 425000.0}

$Y = \{0.0, 150.0, 300.0, 450.0, 600.0, 750.0, 900.0, 1050.0, 1400.0, 1800.0, 2200.0, 2600.0, 3000.0, 3400.0, 4444.4, 8888.8, 13333.3, 17777.8, 22222.2, 26666.7, 31111.1, 33555.6, 36555.6, 38000.0, 40000.0, 41666.6, 43333.3, 45000.0, 46666.6, 48333.3, 50000.0, 51666.6, 53333.3, 55000.0, 56666.6, 58333.3\}$

Therefore when we apply the 3×3 regularizing window, **we are really not applying a square symmetrical filter** (*i.e.* 4 equal sides). Hence, the real meaning of the window size depicted to the right of the models in Figure 6, do really mean how many nearest neighbors have been considered in averaging.

With regard to the averaging window filter itself does not amplifies either. As shown in Figure 7. Even when at the border of the model, only the nearest neighbors are taken into account, therefore only 4 values are averaged, even if the filter is composed of 9 values.

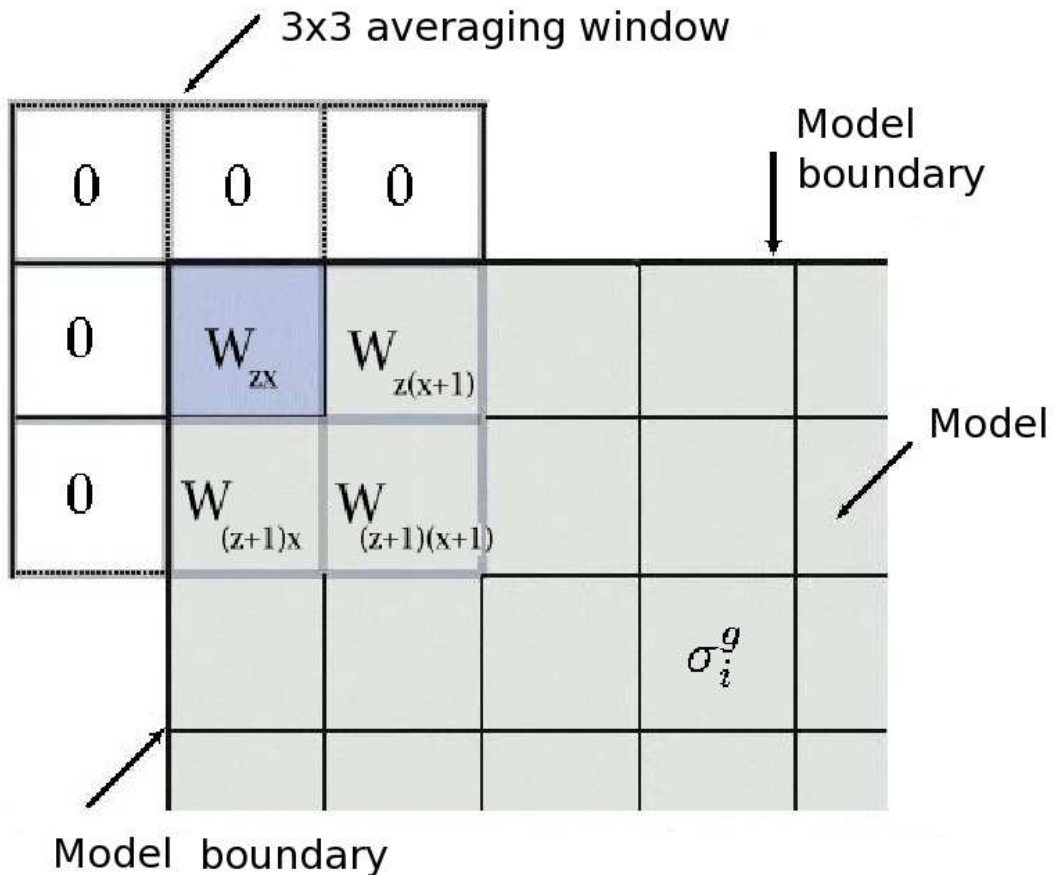


Fig. 7. This figure shows a 2-D regularizing window acting on a model, for **FILTX=1**, **FILTY=1** (window size 3x3).

The periods used to compute impedance tensor where:

$T = \{0.1333E + 01, 0.1778E + 01, 0.2667E + 01, 0.3556E + 01, 0.5333E + 01, 0.7111E + 01, 0.1067E + 02, 0.1422E + 02, 0.2133E + 02, 0.2844E + 02, 0.4267E + 02, 0.5689E + 02, 0.8533E + 02, 0.1138E + 03, 0.1707E + 03,$

0.2276E + 03, 0.3413E + 03, 0.4551E + 03, 0.6827E + 03, 0.9102E + 03,
0.1365E + 04, 0.1820E + 04, }

We did used almost all the above described periods from every station.

As a marginal comment, there exists other ways to regularize the model values during the dynamics (not implemented by the web services) as specified by Yuanchou [24]. It is possible to use a “*low pass filter*” based in *binomial coefficients* introduced in the dynamics, as in:

$$x_i^{(t+1)} = \frac{1}{2} + \frac{1}{2} \text{sgn} \left\{ \sum_{j \neq i=1}^{\text{NUMCOLS}} T_{ij} w_0 x_j^{(t)} + \left[\sum_{j \neq i=1}^{\text{NUMCOLS}} T_{ij} \sum_{k=0=-\text{FILTX}}^{\text{FILTX}} w_k x_{(j-k)}^{(t)} \right] + \text{constant} \right\}, \quad (17)$$

where $\{w_k\}_{-\text{FILTX}}^{\text{FILTX}}$ are the weights of a symmetric (and hence zero phase) low pass filter with gain $\sum_{k=-\text{FILTX}}^{\text{FILTX}} w_k$ unity. This is, the coefficients wont have amplifying effect in the filtered values if $w_0 < 1$ and the w_k are defined by

$$w_k = \left(\frac{2\text{FILTX}}{k + \text{FILTX}} \right) 2^{-2\text{FILTX}}. \quad (18)$$

We end up this considerations section by stating that a good strategy to achieve desired minimal solutions, is to start with a $\beta = 1.0$ value and a large values for the regularizing window sizes, say $\text{FILTX} = 4$ and $\text{FILTY} = 4$ for discretizations like those in example 2. Although a larger value for FILTX and FILTY do really depend on how many discretization cells in model where generated. Then we proceed by diminish the window sizes, an finally the beta value.

VI. CONCLUSIONS

The Versamedium Web Services have shown to provide acceptable good solutions to complex (linear or nonlinear) problems. As shown by the two examples in sections III and IV.

In order to solve the inverse problem, our method relies heavily in the concept of local spatial averages. Besides being a powerful tool for approaching the non-uniqueness problem, it is a natural and intuitive way to provide the extra information needed in under-constrained problems. Choosing windows at will is an appealing didactic feature for viewing a model that is not readily available to us because we never have perfect and complete data.

We showed that the 1-D and 2-D geoelectric inverse problems are solved using a robust variation of Tikhonov’s regularization and the dynamical system in which the web service is based. In the case of field data, our model showed good agreement with previous interpretations.

In the future we hope that Versamedium’s web service will scale and evolve in concert with new linearization methods and with the ever evolving computational processing power. Provided that faster floating point processing units are available. We have also choose the Internet as the perfect efficient way to deliver optimal and cost efficient solutions to remote deployed geophysicists.

VII. ACKNOWLEDGMENTS

Most of the geophysical theory was developed in conjunction with my dear colleagues Francisco Esparza-Hernández and Enrique Gómez-Treviño at Geofísica Aplicada/CICESE (México). I'm deeply thankful to them for such a great collaborative experience.

REFERENCES

- [1] Backus, G.E. & Gilbert, J.F., 1970. Uniqueness in the inversion of inaccurate gross earth data, *Phil. Trans. Roy. Soc. Lond.*, **A266**, 123-192.
- [2] Bailey, R.C., 1970. Inversion of the geomagnetic induction problem, *Proc. R. Soc. Lond.*, **A315**, 185-194.
- [3] Cagniard, L., 1953. Basic theory of the magneto-telluric method of geophysical prospecting, *Geophysics*, **18**, 605-635.
- [4] Constable S.C., Parker R.L. and Constable C.G. 1987. "Occam's inversion: A practical algorithm for generating smooth models from electromagnetic sounding data". *Geophysics*, **52**, 289-300.
- [5] Ghosh, D. P., 1971. Inverse filter coefficients for the computation of apparent resistivity standard curves for a horizontally layered earth: *Geophys. Prosp.*, **19**, 769-775.
- [6] Gómez-Treviño, E., 1987a. Nonlinear integral equations for electromagnetic inverse problems, *Geophysics*, **52**, 1297-1302.
- [7] Hopfield J.J. 1982. "Neural Networks and physical systems with emergent collective computational abilities". Proceedings of the National Academy of Sciences, U.S.A., **79**, 2554-2558.
- [8] Johansen, H. K., 1975. An interactive computer/graphic-display-terminal system for interpretation of resistivity soundings: *Geophys. Prosp.*, **23**, 449-458.
- [9] Jones, A. G., 1993a. The COPROD2 dataset: Tectonic setting, recorded MT data, and comparison of models, *J. Geomag. Geoelectr.*, **45**, 933-955.
- [10] Koefed, O., 1970. "A fast method for determining the layered distribution from the raised kernel function: *Geophys. Prosp.*, **18**, 564-570.
- [11] Niblett, E.R., Sayn-Wittgenstein, C., 1960. Variation of electrical conductivity with depth by the magnetotelluric method. *Geophysics*, **25**, 998-1008.
- [12] Oldenburg D.W., 1978. "The Interpretation of direct current resistivity measurements", *Geophysics*, **43**, No 3, 610-625.
- [13] Oldenburg, D.W., 1979. One-dimensional inversion of natural source magnetotelluric observations, *Geophysics*, **44**, 1218-1244.
- [14] Orellana, E. 1982. Prospección Geoelectrica en corriente continua, *Segunda Edición*, Biblioteca Técnica Philips : 578 pp.
- [15] Parker, R.L., 1977. "Understanding Inverse Theory", *Ann. Rev. Earth Planet. Sci.*, **5**, 35-64.
- [16] Parker, R.L., 1983. The magnetotelluric inverse problem, *Geophys. Surv.*, **6**, 5-25.
- [17] Rodríguez J., Esparza, F.J. & Gómez-Treviño, E. 2010. , "2-D Niblett-Bostick magnetotelluric inversion.", *Geologica Acta*, **8**, No 1, 15-31. (<http://www.geologica-acta.com/pdf/vol0801a03.pdf>)
- [18] Rodríguez J. 2005. "Inversion bidimensional de datos magnetoteluricos y de sondeos electricos verticales mediante redes neuronales tipo Hopfield", CICESE, Ph.D Thesis. (http://www.versamedium.com/geoinverse/joelr_phd_january_2005.pdf in Spanish)
- [19] Smith, J.T. & Booker, J.R., 1988. Magnetotelluric inversion for minimum structure, *Geophysics*, **53**, 1565-1576.
- [20] Tank D.W. & Hopfield J.J., 1986. Simple neural optimization networks: An A/D converter, signal decision circuit, and linear programming circuit. *IEEE Transactions on Circuits and Systems* **33**, 535-541.
- [21] Tikhonov, A. N., & Arsenin, V. Y., 1977. *Solution of ill-posed problems*, V.H. Winston and Sons.
- [22] Weidelt, P., 1972. The inverse problem of geomagnetic induction, *Z. Geophys.*, **38**, 257-289.
- [23] Yuanchou Zhang , 1997. "Wavelet Transforms, Neural networks and Migration Applied to Magnetotellurics", University of Saskatchewan, Ph.D Thesis. (<http://library2.usask.ca/theses/available/etd-10212004-000103/unrestricted/nq23897.pdf>)
- [24] Yuanchou Zhang and K.V. Paulson, 1997. "Magnetotelluric inversion using regularized Hopfield neural networks", *Geophysical Prospecting*, **45**, 725-743.

APPENDIX A : LEAST SQUARES.

A known solution of the proposed problem can be found by taking the derivative of the cost function C with respect to the model and then equating to zero.

$$C = \frac{1}{2}(\vec{d} - \vec{A} \cdot \vec{x})^T (\vec{d} - \vec{A} \cdot \vec{x}), \quad (19)$$

rewriting in vectorial form,

$$C = \frac{1}{2}(\vec{d}^T - \vec{x}^T \cdot \vec{A}^T) \cdot (\vec{d} - \vec{A} \cdot \vec{x}), \quad (20)$$

expanding terms,

$$C = \frac{1}{2}(\vec{d}^T \cdot \vec{d} - \vec{d}^T \cdot \vec{A} \cdot \vec{x} - \vec{x}^T \cdot \vec{A}^T \cdot \vec{d} + \vec{x}^T \cdot \vec{A}^T \cdot \vec{A} \cdot \vec{x}), \quad (21)$$

taking the derivative with respect to the model \vec{x}

$$\frac{\partial C}{\partial \vec{x}} = 0 \rightarrow (-\vec{d}^T \cdot \vec{A} + \vec{x}^T \cdot \vec{A}^T \cdot \vec{A}) = \mathbf{0}, \quad (22)$$

therefore,

$$(\vec{x}^T \cdot \vec{A}^T \cdot \vec{A}) = \vec{d}^T \cdot \vec{A}, \quad (23)$$

taking the transpose in both sides:

$$(\vec{x}^T \cdot \vec{A}^T \cdot \vec{A})^T = (\vec{d}^T \cdot \vec{A})^T, \quad (24)$$

hence,

$$(\vec{A}^T \cdot \vec{A}) \cdot \vec{x} = \vec{A}^T \cdot \vec{d}, \quad (25)$$

then we finally find that the minimum value of C , happens for:

$$\vec{x} = (\vec{A}^T \cdot \vec{A})^{-1} \cdot \vec{A}^T \cdot \vec{d}. \quad (26)$$

The solution will exist for the case in which the number of data points, is equal or greater than the number of unknowns. Moreover it is also necessary for $(\vec{A}^T \cdot \vec{A})$ not to be singular, which will guaranty that $(\vec{A}^T \cdot \vec{A})^{-1}$ exists.

APPENDIX B : A PHYSICS EXAMPLE.

Consider the problem from physics in Figure 8. In which we ought to calculate the electric field \vec{E}_1 at point P, due to the electrostatic charge Q_1 .

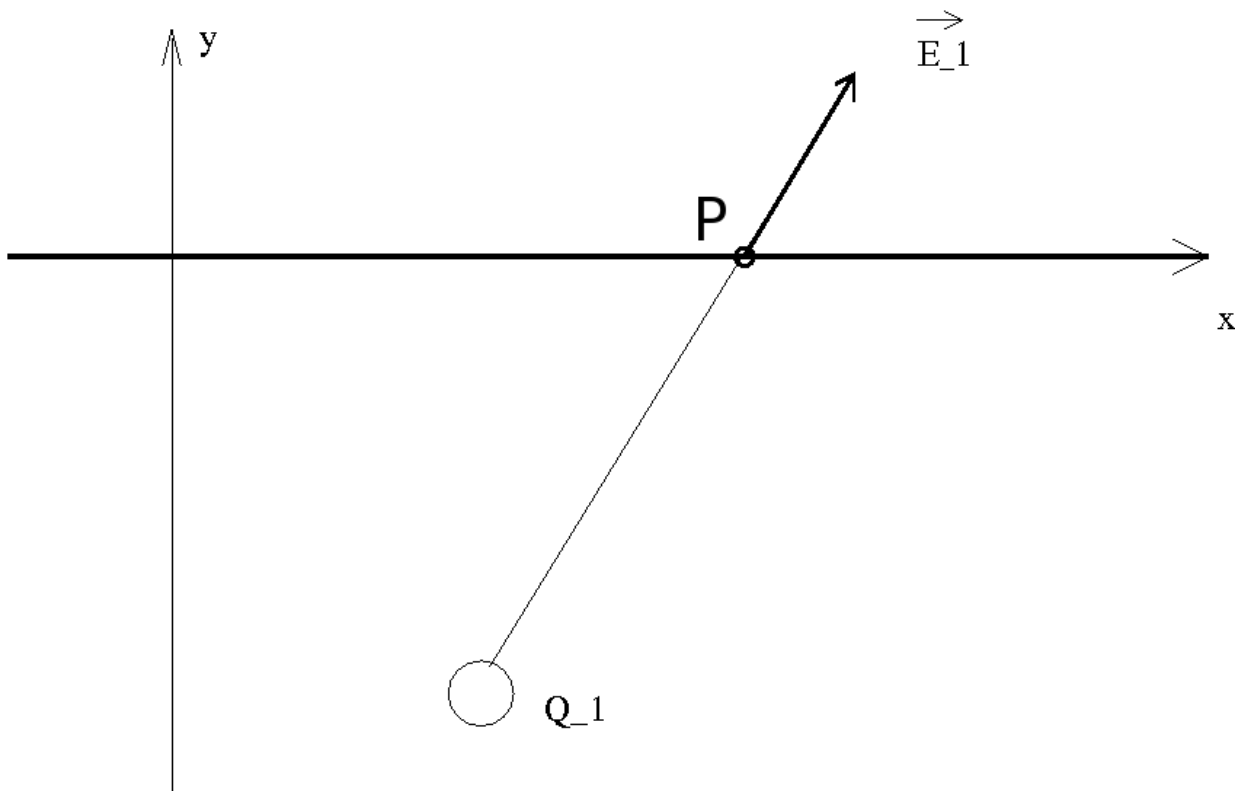


Fig. 8. Direct problem in which we calculate the electric field \vec{E}_1 at point P due to an electrostatic charge Q_1 .

The solution to this problem exists, and it is expressed as a nonlinear functional relationship between both the electric field and the electrostatic charge domains, expressed as:

$$\vec{E}_1 = \frac{1}{4\pi\epsilon_0} \frac{Q_1}{|\vec{R}_1|^2} \hat{R}_1. \quad (27)$$

So far, there is nothing unusual in the above academic problem. Until we re-state the problem to that in which we know just one electric field \vec{E}_{TOTAL} as in Figure 9, and we want to know the distribution of 2 electrostatic charges that could give rise to such $\vec{E}_{\text{TOTAL}} = \vec{E}_1 + \vec{E}_2$ total electric field at point P. For such a problem there are infinite possible solutions for electrostatic charges that could change in value and in position.

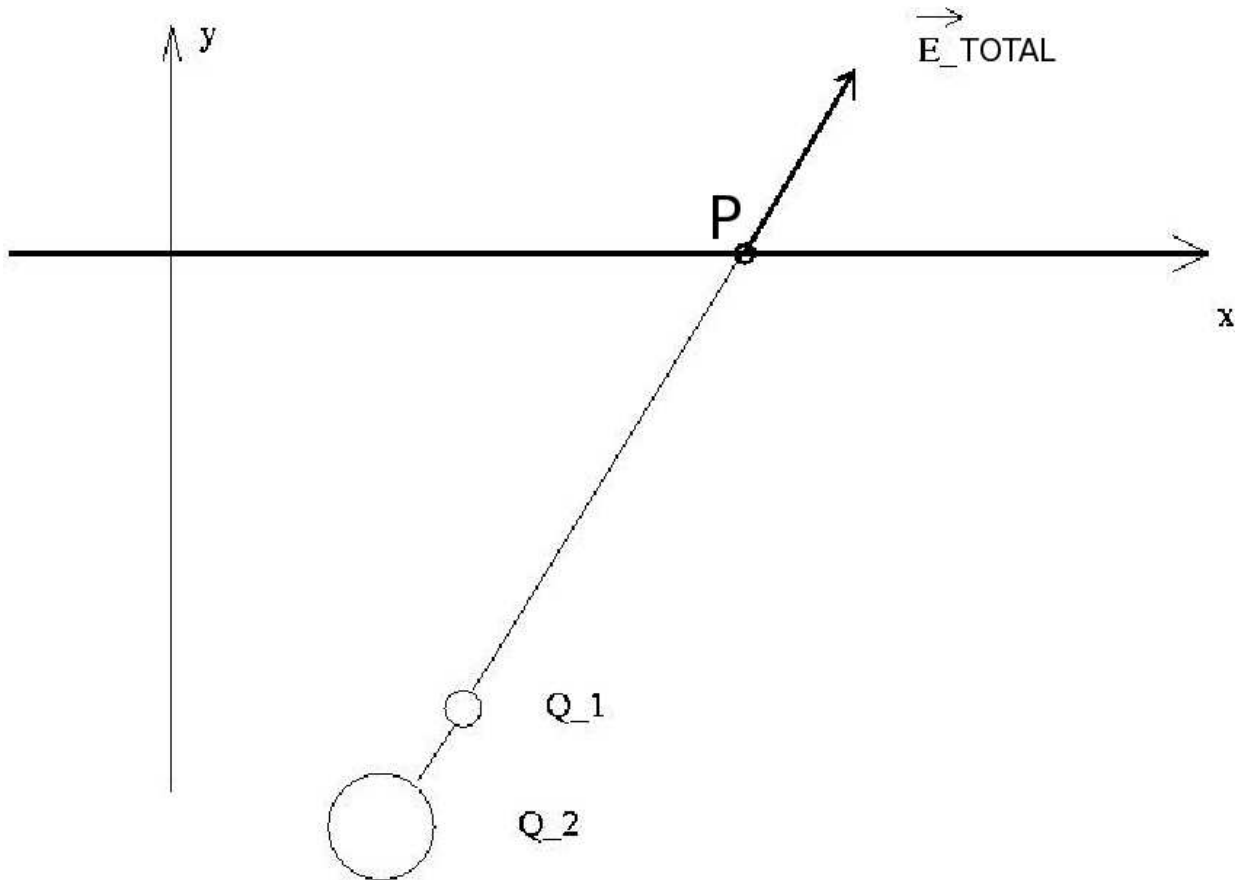


Fig. 9. Inverse problem in which we know the electric field $\vec{E}_{\text{TOTAL}} = \vec{E}_1 + \vec{E}_2$ at point P and we want to solve for the two electrostatic charges Q_1 and Q_2 that gives rise to such electric field.

But why stop there, when we can consider a spatial distribution of 100 Coulombian electrostatic charges spreaded in space. It is rather easy to *forward compute* the resulting electric field in say 10 spatial positions. But, what if instead, we have the 10 spatial electric fields (initially) and we are trying to find the distribution of 100 electrostatic charges that would give rise to such electric fields? (*inverse problem*).

Alternatively consider a *continuous* distribution of charges such as those in Figure 10. For which we ought to find the segments charge densities ρ_1 and ρ_2 when just one electric field \vec{E}_1 measurement is known.

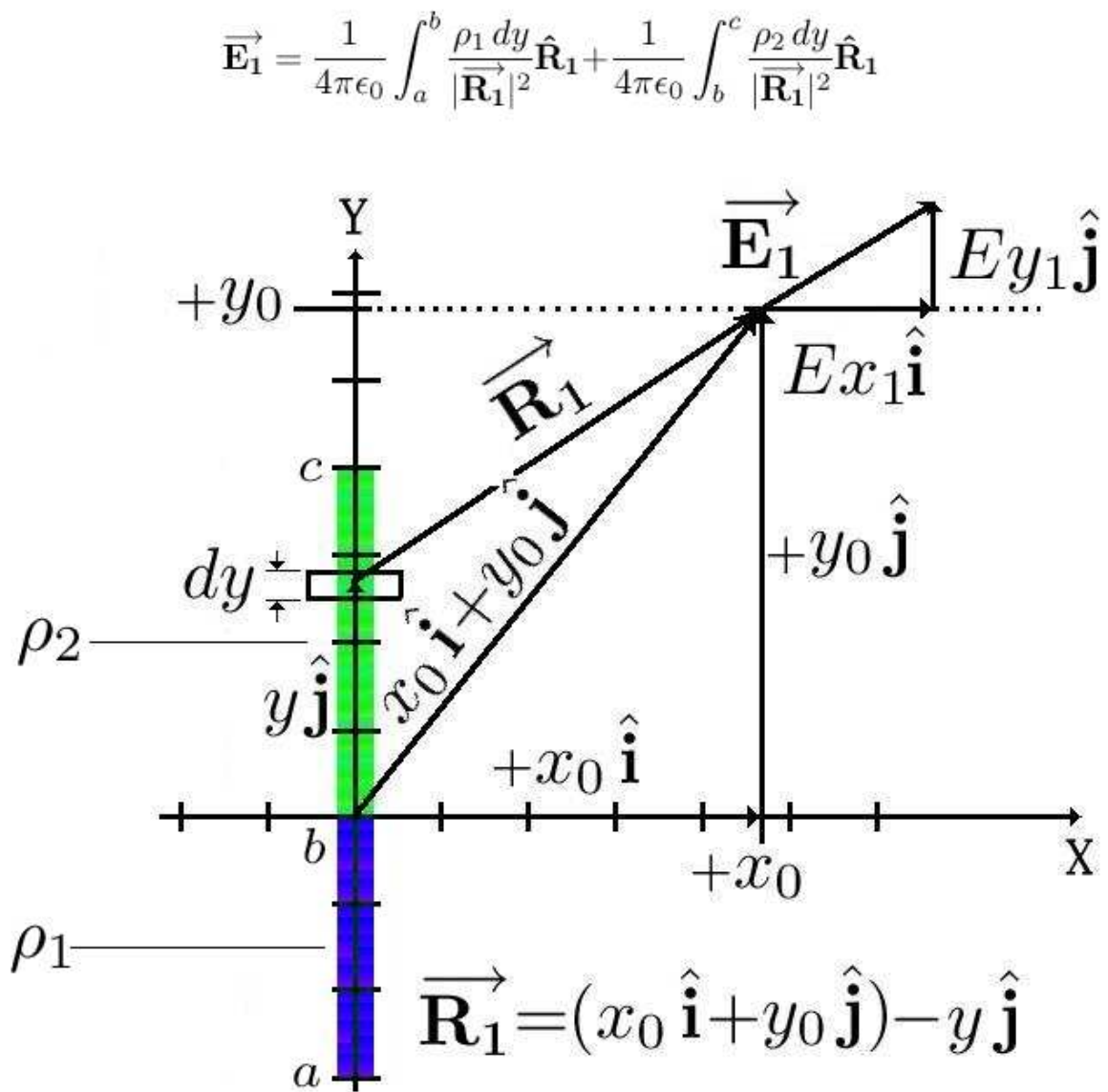


Fig. 10. Inverse problem in which we know the electric field $\vec{\mathbf{E}}_1$ and we want to solve for the two electrostatic charge densities ρ_1 and ρ_2 that gives rise to such electric field.

Furthermore lets consider a *continuous* distribution of charges such as those in Figure 11. For which we ought to find the linear electrostatic charge densities $\rho_1 \dots \rho_7$ and we only know just a few electric field measurements. This is a problem that is well suitable to be linearized and then, to be re-stated as a linear system such as the one represented by equation (1).

$$\begin{aligned} \vec{E}_1 = & \frac{1}{4\pi\epsilon_0} \int_a^b \frac{\rho_1 dy}{|\vec{R}_1|^2} \hat{R}_1 + \frac{1}{4\pi\epsilon_0} \int_b^c \frac{\rho_2 dy}{|\vec{R}_1|^2} \hat{R}_1 + \frac{1}{4\pi\epsilon_0} \int_c^d \frac{\rho_3 dy}{|\vec{R}_1|^2} \hat{R}_1 \\ & + \frac{1}{4\pi\epsilon_0} \int_d^e \frac{\rho_4 dy}{|\vec{R}_1|^2} \hat{R}_1 + \frac{1}{4\pi\epsilon_0} \int_e^f \frac{\rho_5 dy}{|\vec{R}_1|^2} \hat{R}_1 + \frac{1}{4\pi\epsilon_0} \int_f^g \frac{\rho_6 dy}{|\vec{R}_1|^2} \hat{R}_1 + \frac{1}{4\pi\epsilon_0} \int_g^h \frac{\rho_7 dy}{|\vec{R}_1|^2} \hat{R}_1 \end{aligned}$$

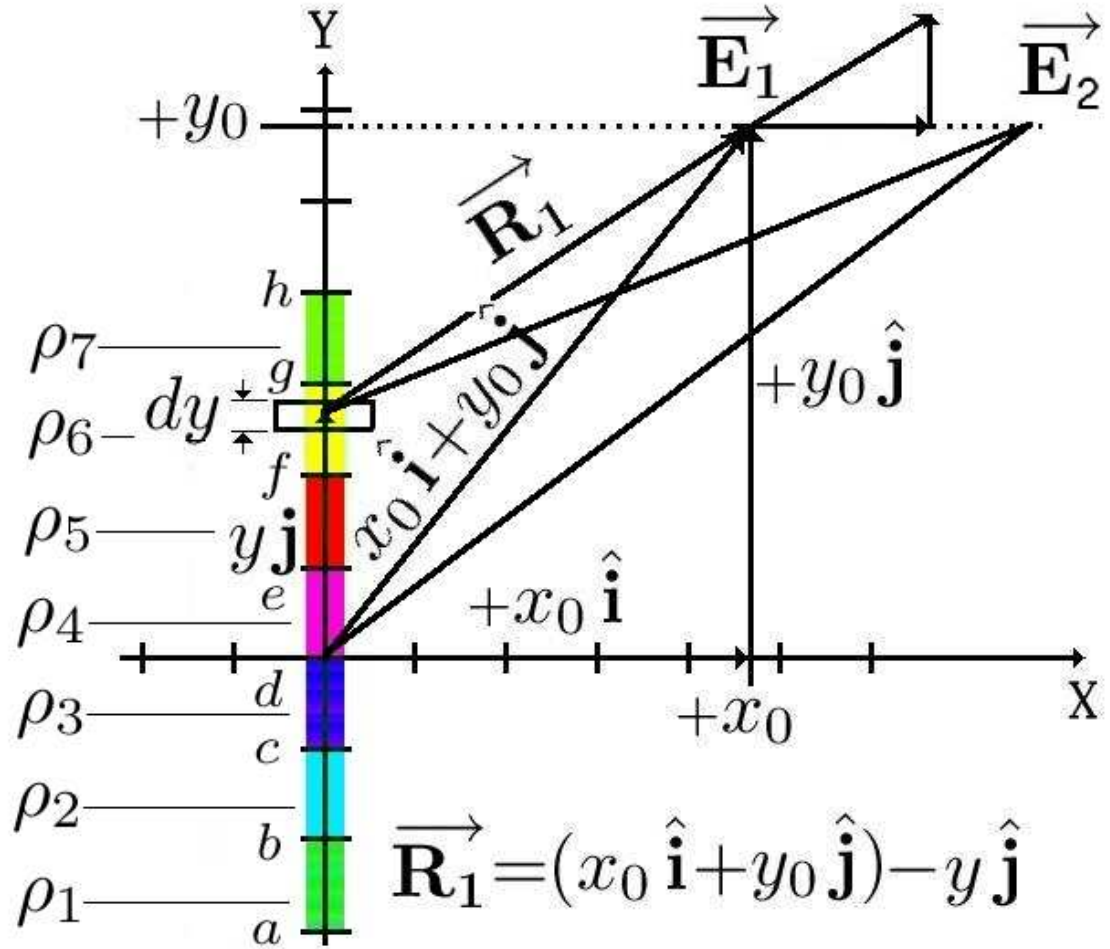


Fig. 11. Inverse problem in which we know the electric field \vec{E}_1 , \vec{E}_2 and we want to solve for the seven electrostatic charge densities $\rho_1 \dots \rho_7$ that gives rise to such electric field.

Of course there are an endless number of such *inverse* possible solutions. And once the nonlinear functional relationship has been linearized, and represented by a linear system¹⁰ as in equation (1). Then we can use the web services to obtain a solution.

The general idea is there. In this case we have nonlinear functional relationships between domains (i.e. electrostatic distribution and electric field) which could be in 1,2,3,.. or N dimensions, and the possibility of endless configurations of the physical properties and its distribution. Those solutions are provided by what is properly named *Inverse Theory* [15].

¹⁰Note: Versamedium Web Services, does not linearize any functional. Furthermore, it does not make any assumption on the variable quantities involved. The user is solely responsible for the discretization, linearization and representation of the problem in a suitable linear system, as described by equation (1).

For problems in the geophysical sciences we have our nonlinear relationship, which is a mapping functional between the subsoil properties such as resistivity \vec{x} and the apparent resistivity measurements at the earths surface or data \vec{d} . Then we can make an n-dimensional linear approximation version of the nonlinear functional (nonlinear inverse problems are often recast as local linear approximations).

In the two examples included in this work we will consider that the model $x(z)$ and the theoretical data \hat{d} are related through a functional relationship $\hat{d} = \mathbf{F}(x(z))$ where F could be related to a Fredholm integral of the first kind:

$$\hat{d}(x) = \int_{\Delta} A(x, z)x(z)dz, \quad (28)$$

where Δ is some real interval and $A(x, z)$ is a kernel derivable from theory. The corresponding discrete formulation can be stated as:

$$\hat{d}_k = \sum_{i=1}^N A_{ki}x_i, \quad k = 1, 2, 3, \dots, M. \quad (29)$$

Typical geophysical linear inverse problems can be stated as an Least Square Problem (LSP) objective function minimization problem, the objective function measures the misfit between the data and the corresponding modeling result. A cost function C (objective function) may be defined in terms of the sum of squares of the differences between the measured and the theoretical data, *i.e.*:

$$C = \frac{1}{2} \sum_{k=1}^M (d_k - \hat{d}_k)^2 = \frac{1}{2} \sum_{k=1}^M \left[d_k - \sum_{i=1}^N A_{ki}x_i \right]^2, \quad (30)$$

where d_k represents the measured data. The solution of the linear inversion problem may then be defined as the set $\{x_i^*\}_1^N$ which minimizes C .

APPENDIX C: REGULARIZED LEAST SQUARES.

As stated in equation (1), \vec{A} may be ill-conditioned or singular yielding a non-unique solution. In order to give preference to a particular solution with desirable properties. A regularization term $\vec{\Gamma}$ is included in this minimization, rewriting in vectorial form:

$$C = \frac{1}{2} (\vec{d} - \vec{A} \cdot \vec{x})^T \cdot (\vec{d} - \vec{A} \cdot \vec{x}) + (\vec{\Gamma} \cdot \vec{x})^T \cdot (\vec{\Gamma} \cdot \vec{x}), \quad (31)$$

$$C = \frac{1}{2} (\vec{d}^T - \vec{x}^T \cdot \vec{A}^T) \cdot (\vec{d} - \vec{A} \cdot \vec{x}) + (\vec{x}^T \cdot \vec{\Gamma}^T) \cdot (\vec{\Gamma} \cdot \vec{x}), \quad (32)$$

$$C = \frac{1}{2} (\vec{d}^T \cdot \vec{d} - \vec{d}^T \cdot \vec{A} \cdot \vec{x} - \vec{x}^T \cdot \vec{A}^T \cdot \vec{d} + \vec{x}^T \cdot \vec{A}^T \cdot \vec{A} \cdot \vec{x}) + (\vec{x}^T \cdot \vec{\Gamma}^T \cdot \vec{\Gamma} \cdot \vec{x}), \quad (33)$$

taking the derivative with respect to the model \vec{x} and equating to zero:

$$\frac{\partial C}{\partial \vec{x}} = 0 \rightarrow (-\vec{d}^T \cdot \vec{A} + \vec{x}^T \cdot \vec{A}^T \cdot \vec{A} + \vec{x}^T \cdot \vec{\Gamma}^T \cdot \vec{\Gamma}) = 0, \quad (34)$$

then we find that the minimum value of C , happens for:

$$\vec{x}^T \cdot (\vec{A}^T \cdot \vec{A} + \vec{\Gamma}^T \cdot \vec{\Gamma}) = (\vec{d}^T \cdot \vec{A}), \quad (35)$$

taking the transpose in both sides:

$$[\vec{x}^T \cdot (\vec{A}^T \cdot \vec{A} + \vec{\Gamma}^T \cdot \vec{\Gamma})]^T = (\vec{d}^T \cdot \vec{A})^T, \quad (36)$$

therefore,

$$(\vec{A}^T \cdot \vec{A} + \vec{\Gamma}^T \cdot \vec{\Gamma})^T \cdot \vec{x} = \vec{A}^T \cdot \vec{d}, \quad (37)$$

hence,

$$(\vec{A}^T \cdot \vec{A} + \vec{\Gamma}^T \cdot \vec{\Gamma}) \cdot \vec{x} = \vec{A}^T \cdot \vec{d}, \quad (38)$$

finally we obtain the solution:

$$\vec{x} = (\vec{A}^T \cdot \vec{A} + \vec{\Gamma}^T \cdot \vec{\Gamma})^{-1} \cdot \vec{A}^T \cdot \vec{d}. \quad (39)$$

APPENDIX D : THE VERTICAL ELECTRIC SOUNDINGS (VES) INVERSE PROBLEM.

In general inverse problems in geophysics are nonlinear. Relatively simple cases such as vertical electric soundings (VES) in 1-D, require special treatment in order to fully handle the nonlinearities [12][2][22][16]. In this chapter we describe the way in which we tackle the nonlinearities problem. It should be noted that methods based in linearization can be applied to solve the problem locally and iteratively. This way we solve gradually the nonlinear nature of the problem [13][19].

A. Linearizing the problem for Vertical Electrical Soundings (VES).

In this problem, the objective is to use the vertical electric soundings “curve” to infer the resistivity properties of the subsoil. It is known that such resistivity distribution constitutes a problem that lacks of a unique solution. This way the existing methods not only should fit the data well but also should impose some continuity condition in the resistivity of the subsoil. Oldenburg [12], used spectral expansion and considered the subsoil resistivity as a continuous function of depth. Constable *et al.* [4], used the derivative of the resistivity, considering that the subsoil properties could be approximated by using a great number of thin layers.

The electric field determination in a variable resistivity media can be made through the Maxwell equations applied to the case in which the electric and magnetic fields does not depend on time. When

we consider a media which resistivity varies only with depth, the problem oversimplifies as there exists vertical axis symmetry. Also the electric field has a vertical and radial component, therefore the determination of it in any point of the subsoil is relatively simple.

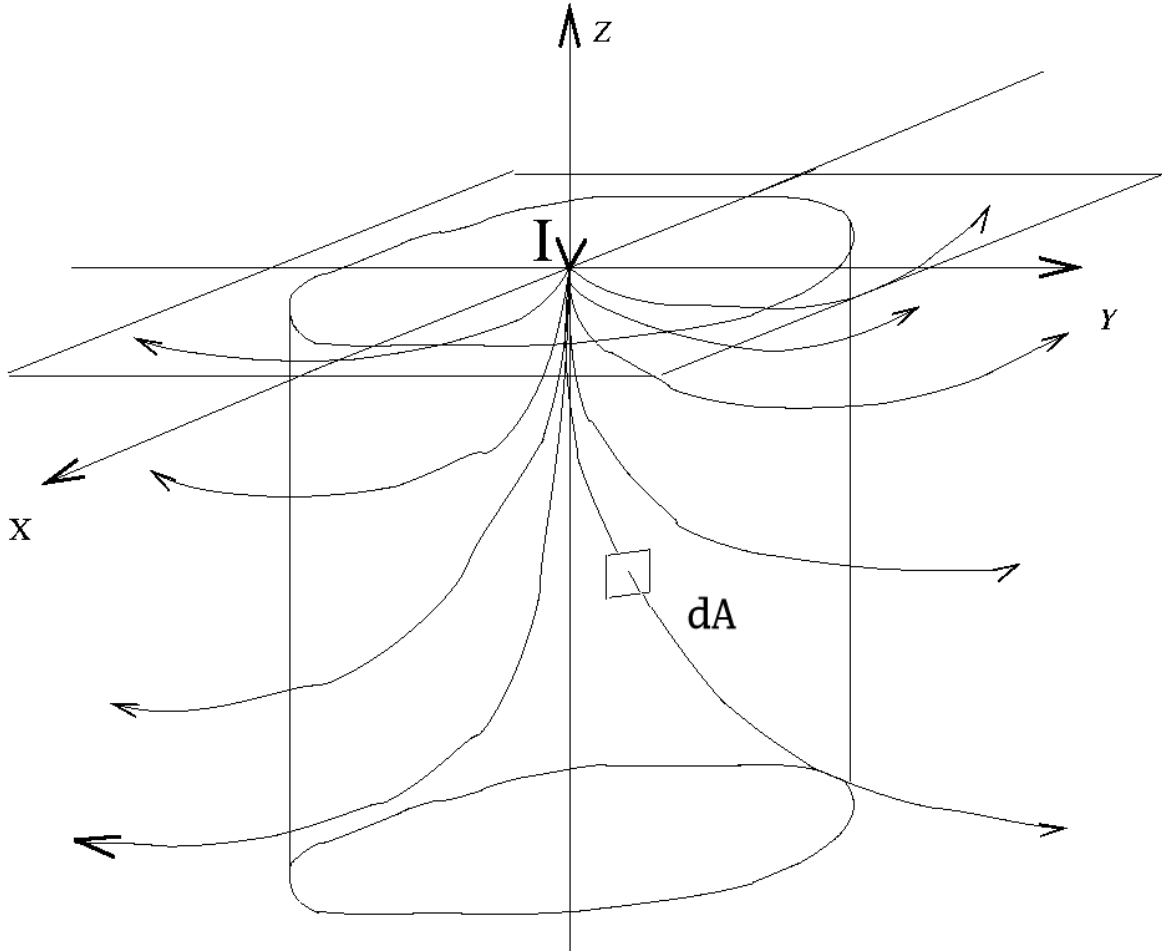


Fig. 12. Coordinate system used in determining the electric potential. The source is an punctual electrode that coincides with the origin of the coordinate system, the electric resistivity for $z > 0$ is infinite and for $z \leq 0$ varies only with the depth, but can be any function of depth.

It is consider that the current source is an punctual electrode localized in $z = 0$. Using cylindrical coordinates whose origin coincides with the location of the electrode (see Figure 12), the equation to solve out of the source is:

$$\nabla \cdot \left(\frac{\nabla V(r, z)}{\rho(z)} \right) = 0 \quad z > 0, \quad r > 0, \quad (40)$$

where $V(r, z)$ is the electric potential and $\rho(z)$ is the subsoil resistivity. The source will be included later, by means of the determination of the associated constants to the solution of the differential equation (40) (see Rodriguez [18] for an extended description on the differential equations.):

$$V(r, z) = \int_0^{\infty} A(\lambda)h(z, \lambda)J_0(\lambda r)d\lambda, \quad (41)$$

where $A(\lambda)$ is a function to be determined and the function $h(z, \lambda)$ satisfies:

$$h''(z, \lambda) - \frac{\rho'(z)}{\rho(z)}h'(z, \lambda) - \lambda^2h(z, \lambda) = 0. \quad (42)$$

The components of the electric field are:

$$E_r = -\frac{\partial V}{\partial r}, \quad E_z = -\frac{\partial V}{\partial z}, \quad (43)$$

therefore a border condition of the differential equation (42) is $h' \rightarrow 0$ if $z \rightarrow \infty$. From the equation (41) it can be seen that $V(r, z)$ is the Hankel transform of order zero of $\frac{A(\lambda)h(z, \lambda)}{\lambda}$. The border condition in the interface ground air is:

$$J_n = -\sigma(0)\frac{\partial V(r, z)}{\partial z} = I\delta(x)\delta(y) \text{ in } z = 0, \quad (44)$$

using the relationship between $V(r, z)$ and $\frac{A(\lambda)h(z, \lambda)}{\lambda}$, the equation (44) is transformed in:

$$-\sigma(0)\frac{A(\lambda)}{\lambda}h'(0, \lambda) = \frac{I}{2\pi}, \quad (45)$$

isolating $A(\lambda)$ from this equation and by substitute it in equation (41), we find that:

$$V(r, z) = -\frac{I\rho(0)}{2\pi} \int_0^{\infty} \lambda \frac{h(z, \lambda)}{h'(0, \lambda)} J_0(\lambda r) d\lambda, \quad (46)$$

by the superposition principle, the potential in the center of the punctual electrodes will be

$$V(r, 0) = -\frac{I\rho(0)}{\pi} \int_0^{\infty} \lambda \frac{h(0, \lambda)}{h'(0, \lambda)} J_0(\lambda r) d\lambda, \quad (47)$$

now r is the distance to the center of either of the electric punctual electrodes. The apparent resistivity for the Schlumberger array will be given by

$$\rho_a(r) = \pi r^2 \frac{E_r}{I}, \quad (48)$$

using the equation (43) and the expression for the electric potential given in equation (47), the apparent resistivity is

$$\rho_a(r) = -\rho(0)r^2 \int_0^\infty \lambda^2 \frac{h(0, \lambda)}{h'(0, \lambda)} J_1(\lambda r) d\lambda, \quad (49)$$

this equation gives the apparent resistivity for the Schlumberger array, considering to the resistivity of the subsoil a continuous function of depth. Nonetheless, can not be used for the case in which the layers are homogeneous, because the derivative of $\rho(z)$ it is not defined in the interface of two layers with different resistivities. In this case it is desirable [14] to solve the differential equation (42) in every homogeneous layer and apply the border conditions in each interface *i.e.* continuity of the potential and of the vertical component of the current density. The apparent resistivity for the Schlumberger array is finally defined by

$$\rho_a(r) = r^2 \int_0^\infty T_1(\lambda) J_1(\lambda r) \lambda d\lambda, \quad (50)$$

$T_1(\lambda)$ is named the resistivity transform [10], and it can be calculated with the following recursive formula

$$T_i = \frac{T_{i+1} + \rho_i \tanh(\lambda t_i)}{1 + T_{i+1} \tanh(\lambda t_i) / \rho_i}, \quad (51)$$

t_i is the thick of the i -th layer. For a media with N layers, the resistivity transform in the last layer is $T_N = \rho_N$. This way $T_1(\lambda)$ can be calculated with equation (51). The Hankel transform of order one in the equation for the apparent resistivity can be calculated numerically by means of digital filters (see Ghosh [5]). We choose to use the filters of Johansen [8], because these filters are relatively large than the former and therefore have a better global precision. Figure 13 shows the proposed layered model, in which the relationship between the apparent resistivity in the surface ρ_a and the real resistivity of a layered model in the subsoil ρ_z , is established by means of local linearization in every layer by using the continuity conditions of the potential in the interfaces in every layer.

$$\frac{\partial \rho_a}{\partial \rho_j} = \sum_k \frac{\partial T_1(\lambda_k)}{\partial \rho_j} f_k. \quad (53)$$

Seamingly, in order to evaluate $\frac{\partial T_i}{\partial \rho_j}$, we write

$$\frac{\partial T_1(\lambda_k)}{\partial \rho_j} = \frac{\partial T_1}{\partial T_2} \frac{\partial T_2}{\partial T_3} \frac{\partial T_3}{\partial T_4} \dots \frac{\partial T_{j-1}}{\partial T_j} \frac{\partial T_j}{\partial \rho_{j,j}}. \quad (54)$$

By taking derivatives in equation (51) to obtain expressions for $\frac{\partial T_i}{\partial T_{i+1}}$ and $\frac{\partial T_j}{\partial \rho_j}$:

$$\frac{\partial T_i}{\partial T_{i+1}} = \frac{\left[1 - \tanh^2(t_i \lambda_k)\right]}{c_i}, \quad (55)$$

and

$$\frac{\partial T_i}{\partial \rho_i} = \tanh(t_i \lambda_k) \frac{\left[1 + T_{i+1}^2 / \rho_i^2 + 2T_{i+1} \tanh^2(t_i \lambda_k) / \rho_i\right]}{c_i}, \quad (56)$$

where

$$c_i = \left[1 + \tanh^2(t_i \lambda_k) T_{i+1} / \rho_i\right]^2. \quad (57)$$

The recursive process can be started then, noting that

$$\frac{\partial T_N}{\partial \rho_N} = 1. \quad (58)$$

Once the derivatives are calculated it is possible to represent the problem by means of a linear relationship [6] expressed in the following form:

$$\vec{\rho}_a = \vec{\mathbf{A}} \cdot \vec{\rho}_z, \quad (59)$$

In which the matrix $\vec{\mathbf{A}}$ of derivatives is calculated recursively as it evolves towards the minimum of the cost function defined by the objective function, note that it also depends as well on the obtained model during each iteration, nonetheless this approach enable us to find a solution to the nonlinear problem, by means of local linearization expressed in equation (59). Once the problem has been posed, it can be solved using the regularized least squares Versamedium's method, which is explained with greater detail in section III, Example 1 Simulated averages in 1-D: The Vertical Electric Sounding (nonlinear) problem.