Earth Science Research; Vol. 5, No. 2; 2016
ISSN 1927-0542 E-ISSN 1927-0550
Published by Canadian Center of Science and Education

Improving Levenberg-Marquardt Algorithm Inversion Result Using Singular Value Decomposition

Widodo[1] & Durra Handri Saputera[1]

[1] Institute of Technology Bandung, West Java, Indonesia
Correspondence: Widodo, Faculty of Mining and Petroleum Engineering, Institute of Technology Bandung, Bandung, Indonesia. Tel: 6281-1861-7095. E-mail: widodo@gf.itb.ac.id

Received: March 2, 2016 Accepted: March 13, 2016 Online Published: April 12, 2016
doi:10.5539/esr.v5n2p20 URL: http://dx.doi.org/10.5539/esr.v5n2p20

Abstract
Inversion is a process to determine model parameters from data. In geophysics this process is very important because subsurface image is obtained from this process. There are many inversion algorithms that have been introduced and applied in geophysics problems; one of them is Levenberg-Marquardt (LM) algorithm. In this paper we will present one of LM algorithm application in one-dimensional magnetotelluric (MT) case. The LM algorithm used in this study is improved version of LM algorithm using singular value decomposition (SVD). The result from this algorithm is then compared with the algorithm without SVD in order to understand how much it has been improved. To simplify the comparison, simple synthetic model is used in this study. From this study, the new algorithm can improve the result of the original LM algorithm. In addition, SVD is allowing more parameter analysis to be done in its process. The algorithm created from this study is then used in our modeling program, called MAT1DMT.
Keywords: inversion, Levenberg-Marquardt, magnetotelluric

1. Introduction

Subsurface imaging is the main focus in geophysical exploration. The image of earth surface below can be obtained by measuring various physical properties of the earth. One of the examples is by measuring earth's electromagnetic properties in frequency domain. This method is called magnetotelluric method. Throughout the years, the development of magnetotelluric modeling and inversion algorithm has shown numerous successes. Although three-dimensional (3D) modeling and inversion algorithms are available, 1D modeling and inversion are still being performed, especially to obtain preliminary results before doing higher dimensional modeling.

Before structure and lithology analysis can be made, data obtained from the field needs to go through modeling process. This process is called backward modeling or often called inversion process. Aside from the challenge to be able to create the inversion solution model that synchronized with the geology model, the inversion process using a computation of certain algorithm is still facing one major problem, which is the result stability issue. This problem arises because geophysical inverse problems are ill-posed problems where small changes in the data can cause large changes in the solution (Jupp & Vozoff, 1975). That is why understanding process and the development of geophysical inversion algorithm are important since the solution of inversion process will affect the quality of data interpretation.

One of the common solutions that has been used in the geophysical inverse problems is the modification of linearized approach of the inversion problem. There have been many successful algorithms created to solve MT inversion problems, some of the examples are Levenberg-Marquardt algorithm (Marquardt 1963), which utilize damping parameter in its algorithm and Occam algorithm (Constable et al, 1987) which apply the smoothness constrain in the algorithm.

In this paper we will present an application of LM algorithm in 1D MT case. In addition of the original algorithm results, we will also show the result from the modified algorithm using singular value decomposition (Jupp and Vozoff, 1975). The purpose of this study is to understand inversion result improvement of this modified algorithm compared to the original LM algorithm without SVD. To validate original and modified algorithm created in this study, result from established software will be used as comparison. The algorithm created from this study will be used in our own created program called MAT1DMT. The script that used in this study is available at **appendix A**.

2. Methodology

This section will briefly explain the basic theory of the algorithm and the test of the algorithms used in this study. The basic theory explained will be mainly focused on inversion theory, which consists of the basic theory and the explanation about the original LM algorithm and the modification of the algorithm.

2.1 Inversion

MT forward modeling problems can generally be written with

$$\mathbf{d} = \mathrm{g}(\mathbf{m}) \tag{1}$$

With **d** is data, which can be apparent resistivity values (ρ_a) or phase (Φ), and g (**m**) is the forward operator that relates model parameters (**m**) with its response. In 1D case, models can be defined the by the layer resistivity and thickness of the subsurface. Jupp and Vozoff (1975) suggested the value of a model in logarithmic to avoid a negative model value in the inversion result.

Inversion process is essentially a matching process of the calculated model response with the observational data. The implementation of this matching process is performed automatically using mathematical and statistical algorithms. In general, the inversion solution which can be applied to non-linear cases, including MT, is the Gauss Newton solution.

$$\boldsymbol{m} = \boldsymbol{m_0} + \Delta \boldsymbol{m} \tag{2}$$

$$\Delta \boldsymbol{m} = [\boldsymbol{J^T W J}]^{-1} \boldsymbol{J^T W}\big(\boldsymbol{d} - g(\boldsymbol{m_0})\big) \tag{3}$$

With **m** is the solution models, Δm is the model parameters update, $\boldsymbol{m_0}$ is the initial model, and J is the Jacobian matrix (J_{ij}) or models sensitivity matrix.

$$J_{ij} = \frac{\partial g_i}{\partial m_j} \tag{4}$$

2.2 Levenberg-Marquardt Algorithm

Levenberg-Marquardt Algorithm is an algorithm that applies the minimization of model perturbation to the Gauss Newton solution. It can be done by minimizing the objective function F

$$F = \big(\boldsymbol{d} - g(\boldsymbol{m_0} + \Delta \boldsymbol{m})\big) + \lambda \big|\big|\Delta m\big|\big|^2 \tag{5}$$

The solution now becomes

$$\Delta \boldsymbol{m} = [\boldsymbol{J^T W J} + \lambda \boldsymbol{I}]^{-1} \boldsymbol{J^T W}\big(\boldsymbol{d} - g(\boldsymbol{m_0})\big) \tag{6}$$

With the parameter λ is showing of the effect of model perturbation. Small λ value will make the equation (6) becomes equal to Gauss – Newton solution in equation (3). The parameter λ is initialized with a large initial value. If the solution after iteration has larger misfit than the previous iteration λ value is increased, otherwise it is reduced. To ensure the calculation stability the equation (6) is modified to equation (7).

$$\Delta \boldsymbol{m} = [\boldsymbol{J^T W J} + \lambda diag(\boldsymbol{J^T W J})]^{-1} \boldsymbol{J^T W}(\boldsymbol{d} - g(\boldsymbol{m})) \tag{7}$$

Jupp and Vozoff (1975) are suggesting another solution to improve the stability of the results. This solution is obtained by modifying the algorithm using Singular Value Decomposition (Lanczos, 1958). Singular Value Decomposition (SVD) is a method to divide a M x N sized matrix into three arbitrary matrices that has the relation :

$$\mathbf{J} = \mathbf{U S V^T} \tag{8}$$

With **U** is M x N matrix which is eigenvector from data, **V** is N x N matrix eigenvector from parameter space, and **S** is N x N diagonal matrix which value is the square root from eigenvalue of **J.**

In that study, Jupp and Vozoff (1975) defining the damping parameter in LM algorithm using a N x N diagonal matrix **P** that has component value defined in equation (9).

$$\mathrm{p_i} = \frac{\mathrm{k_i^{2H}}}{\mathrm{k_i^{2H}} + \mu^{2\mathrm{H}}} i = 1,2,3,\ldots,N \tag{9}$$

With k_i is ratio between the diagonal values from **S** matrix (s_i) with its first value (s_1), H is an integer which is defined H = 2 in the algorithm and μ is error level which is defined by the user and is determined with a high value and then decreased as the error of the solution model is reduced in every iteration, the determination of this value will be further explained in the discussion section. Using this damping matrix, the modified solution of LM algorithm using SVD is then become equation (9)

$$\Delta \mathbf{m} = \mathbf{V P S^{+} U^T}\big(\mathbf{d} - \mathrm{g}(\mathbf{m})\big) \tag{10}$$

With S^+ is N x N diagonal matrix that has $1/s_i$ (i = 1,2,3, …, N) as its component values.

2.3 Scaling Factor

In our inversion algorithm, we are adding one more extra parameter to the calculation which is called the scaling factor. The idea of the scaling factor parameter is to scale the solution model into a new model which has the same average calculated response as the observed data. The scale factor is a scalar value that is obtained by dividing the calculated apparent resistivity of the solution model in an iteration and then multiply the factor to the solution model which can be expressed mathematically as equation (11) and equation (12).

$$\mathrm{SF} = \frac{\text{average calculated apparent resistivity}}{\text{average observed apparent resistivity}} \tag{11}$$

$$\mathbf{m_s} = \mathbf{m} * \mathrm{SF} \tag{12}$$

With $\mathbf{m_s}$ is the scaled model and **m** is the solution model in an iteration.

In the case of 1D MT, the observed apparent resistivity is greatly reflecting the earth subsurface resistivity model of the measurement point. Therefore, it is only natural to choose the solution resistivity model which has an average value that match the average observed apparent resistivity value and scaling factor is used to fulfill this condition. However, in order to explore more solutions, the scaling factor multiplication is not applied to every solution model in the inversion process; but instead, an extra constraint is added in regard of this matter. The constraint included in the algorithm is the phase fitting condition, which means that only solution model, where the calculated value is in a certain range of phase misfit, is getting multiplied by the scaling factor. The phase fit criterion was chosen because phase data is reflecting the changes of resistivity in between layer boundaries of the earth layers.

In this study, the phase misfit criterion value is set at minimum to make sure that the scaling process is only executed when the solution model has shown the correct pattern of resistivity. However, this value is increased when a certain number of error level increments do not produce the desired result. The purpose of this treatment is to fix the solution model so the desired result can be achieved, but this treatment may not be always give a good result since increased tolerance of phase misfit means that more model is scaled regardless of its resistivity pattern.

2.4 Algorithm Test

To simplify the validation in comparing the algorithms, we will show inversion using synthetic data. First example is an inversion form synthetic data obtained from forward modeling calculation of homogeneous layer earth and synthetic data obtained from forward modeling calculation of simple three layered earth. We will also use varying starting models to understand the effect of starting model on inversion result. In addition to that, we will display the quality of the result using variety number of error level parameter and our proposed determination of error level in our algorithm.

The quality of the solution model is measured by the misfit value which is the root mean square (RMS) error value between the calculated and observed values of apparent resistivity and phase, that are normalized to the observed data values and then it will be presented in the percentage value.

3. Results and Discussion

In this study, we will present three examples that will show the quality of both inversion algorithms. The first example is an inversion from synthetic data produced from a homogenous layer with resistivity value of 200Ωm using both original and modified Levenberg Marquardt algorithm. The starting model used in this process is a homogenous layer with resistivity value of 100Ωm. The next example is an inversion case of 3 layered earth models with resistivities (ρ) and thicknesses (d) as follows: ρ_1 = 100 Ωm , $\rho_2 = 1000\,\Omega m$, $\rho_3 = 100\,\Omega m$, $d_1 = 1000\,m$, $d_2 = 2000\,m$. Starting model used for all inversion is 5 layered earth model with same resistivity value of 100 Ωm. These two cases of inversion can be called as a good case, where the model results only show a small ambiguity effect in the model. Another example, which is shown on figure 3, is an example of a bad case, where a solution model with significant difference from the synthetic model can produce the almost similar responses as the synthetic model responses. The starting model used in the next example is 1000 Ωm. These examples can be seen on figure 1 to figure 3.

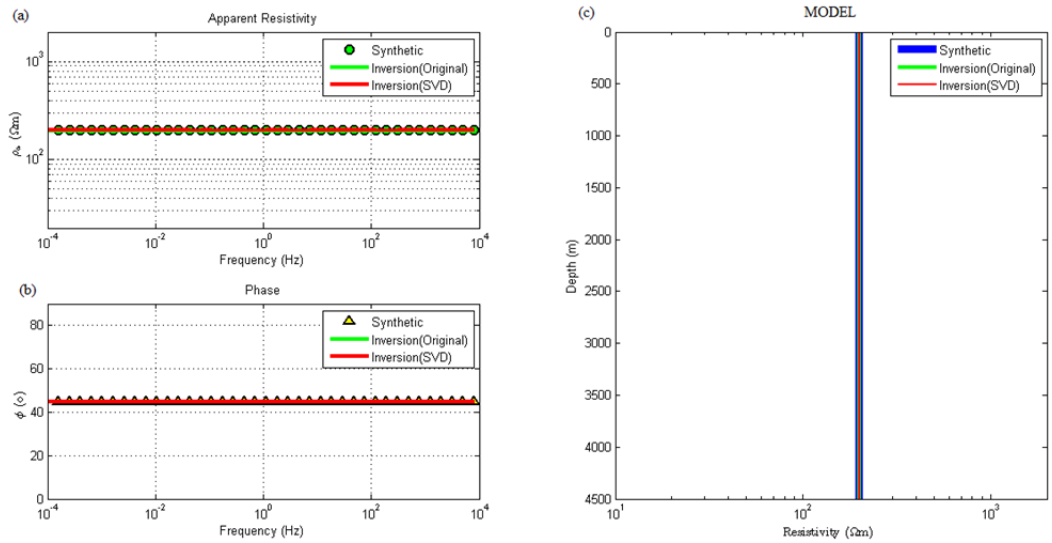

Figure 1. Inversion results from the original and modified Levenberg Marquardt algorithm using 100 Ωm homogenous resistivity layer as starting model. Part (a) shows the apparent resistivity curve, (b) shows the phase curve, and (c) shows the 1-D resistivity models obtained. Synthetic model is indicated by blue line and its responses are indicated by green dots for apparent resistivity and yellow triangle for phase. Models and responses obtained from inversion using the original algorithm are indicated by green line and the result of the modified algorithm is indicated by red line

The inversion model results from both inversion algorithms, which are displayed on figure 1, are forming the same model as the synthetic model. These results are indicating that the iterations involved in the algorithm calculations are leading the starting model to a model solution that has the same apparent resistivity and phase response as the data, showing that the algorithm used is giving the desired result.

As it can be seen on figure 2 that both algorithm inversions are producing a good result, where ~~the~~ both algorithms are providing a solution that has small error in response relatively to the data. In addition to that, the original LM algorithm is almost forming the exact synthetic model used in the forward modeling process, but the modified algorithm inversion model is not forming the exact synthetic model. However this may not be an issue, since the deviation from the true resistivity is not significant, where the layers may still be interpreted as the same lithology.

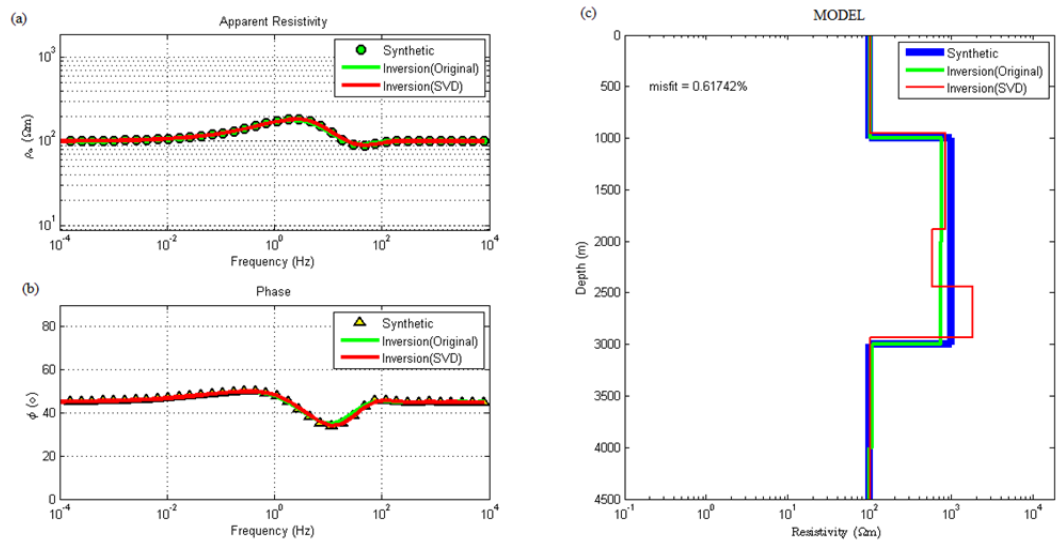

Figure 2. Inversion results from the original and modified Levenberg Marquardt algorithm using 100 Ωm uniform resistivity layers as starting model. Part (a) shows the apparent resistivity curve, (b) shows the phase curve, and (c) shows the 1-D resistivity models obtained. Synthetic model is indicated by blue line and its responses are indicated by green dots for apparent resistivity and yellow triangle for phase. Models and

responses obtained from inversion using the original algorithm is indicated by green line and the result of the modified algorithm is indicated by red line

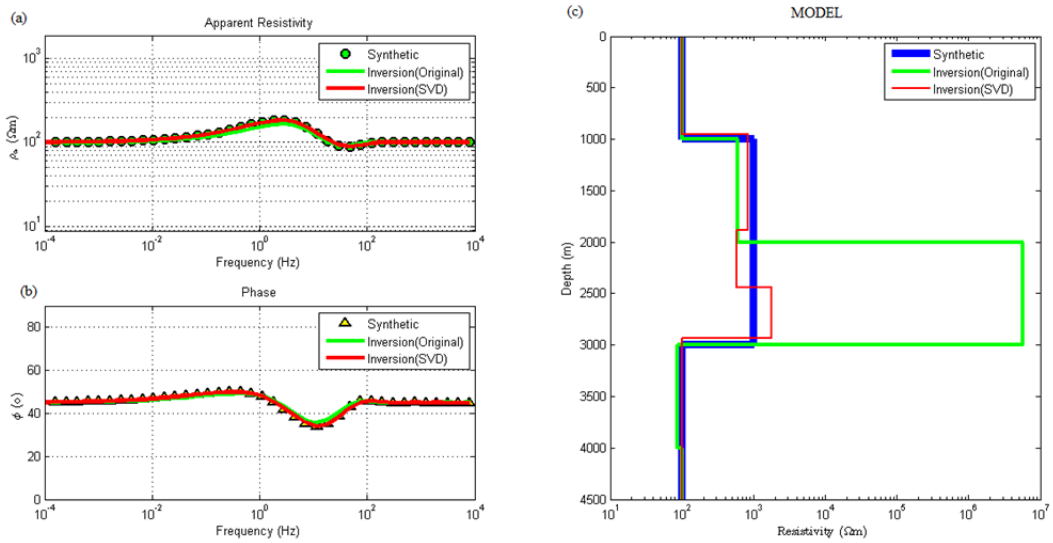

Figure 3. Inversion results from the original and modified Levenberg Marquardt algorithm using 1000 Ωm uniform resistivity layers as starting model. Part (a) shows the apparent resistivity curve, (b) shows the phase curve, and (c) shows the 1-D resistivity models obtained. Synthetic model is indicated by blue line and its responses are indicated by green dots for apparent resistivity and yellow triangle for phase. Models and responses obtained from inversion using the original algorithm is indicated by green line and the result of the modified algorithm is indicated by red line

From the result shown in figure 3, it can be seen that the inversion process is producing various results using different starting model. Unlike the results produced from inversion using 100 Ωm uniform layers, the result from original algorithm has a significant difference, where a part of the solution model is becoming a very resistive layer. This 'bad' result is caused by the instability in inversion process, where a small change in data leads to large change in the solutions (Jupp and Vozoff, 1975). However, this problem is solved in the inversion result using modified algorithm with SVD. It can be seen in figure 3 that the model result from inversion using the modified algorithm is showing a consistent pattern of resistivity compared to the result before, when 100 Ωm uniform layers is used as starting model. Therefore, it can be said that the modified LM algorithm using SVD can produce a more stable results than the original LM algorithm, regardless of the starting model used.

Even though the modified algorithm has been able to create the solution that has less dependency to its starting model, the stability problem still persists. One of the causes is the determination error level parameter in the inversion process. **Figure 4** is illustrating the effect of various fixed error level values used in the inversion process of the synthetic data shown at **Figure 3** with various starting model.

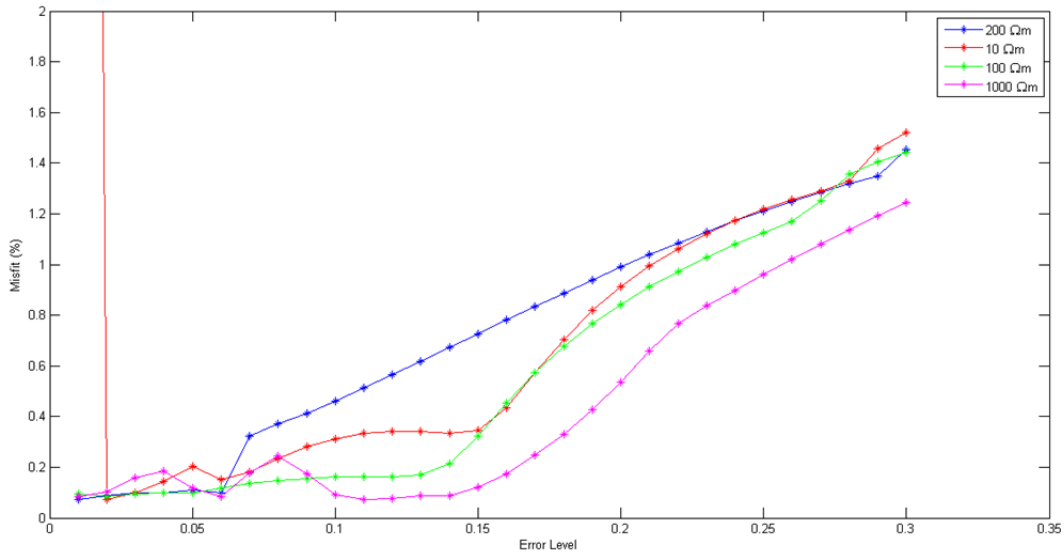

Figure 4. The misfit value of the inversion at the convergence condition. Blue, red, green, and magenta dotted line are indicating the results from 200 Ωm, 10 Ωm, 100 Ωm, and 1000 Ωm starting model. Each dot in the line is representing 0.01 interval of error level value

As it can be seen on the **Figure 4**, the misfit has a tendency to significantly increased as the error level value is increased, especially when the value is above 0.15 (15% error value) and the minimum value of misfit is located there. However, there is also a lower limit of error level value when the inversion is not giving the desired result and instead in the case with starting model of 10 Ωm (red dotted line in the figure), the misfit value is increased at rhe lowest value of error level.Therefore the suggested determination of error value by Jupp and Vozoff (1975), which is determined to be a high value and then lowered if the misfit were reduced or increased if the misfit were increased, has been the right decision because it will ensure the right error level value, which is not too high or too low. In addition to that, we suggest that the initial value of error level to be in the range of 0.10 – 0.20 because it is ensuring that the error level value not become too low or too high. Another inversion scheme available is to use various fixed error level value for the entire inversion and take the result with smallest misfit value as the solution model like the result illustrated on the **Figure 4**, although this scheme may not be recommended because it will consume more computation time.

4. Conclusion

Based on all results discussed in the previous section, it can be concluded that the original Levenberg Marquardt inversion result has a high dependence on starting model used in inversion. In some cases, this algorithm can produces good results, but in other cases it can also produces bad results, which can lead to misinterpretation due to instability that may happen in the inversion process. However, this problem can be solved by the modified algorithm using SVD. This algorithm is providing a more consistent results than the original ones, which is can be said that it is more reliable to be used in the inversion process. In addition to improve the result further we suggest the determination of error level initial value in the inversion to be in the range of 0.10 to 0.20 and if the inversion still does not give the desired result, a change of starting model may be required as its response can be too far from the solution model and the inversion solution is stuck on the local optima of the misfit. Also the use of multiple error level values in one inversion program can become one of the inversion scheme alternatives to see which error level value is giving the best result.

References

Cagniard, L. (1953). Basic theory of magnetotelluric method of geophysical prospecting. *Geophysics, 18*, 605-635. http://dx.doi.org/10.1190/1.1437915

Constable, S. C., Parker, R. L., & Constable, C. G. (1987). Occam's inversion: A practical algorithm for generating smooth models from electromagnetic sounding data. *Geophysics, 52*, 289-300. http://dx.doi.org/10.1190/1.1442303

Jupp, D. L. B., &Vozoff, K. (1975). Stable Iterative Methods for the Inversion of Geophysical Data.*Geophysical Journal of the Royal Astronomical Society, 42*(3), 957-976. http://dx.doi.org/10.1111/j.1365-246x.1975.tb06461.x

Lanczos, C. (1958). Linear Systems in self-adjoint form, Am. math. *Monthly, 65*, 665-679. http://dx.doi.org/10.2307/2308707

Marquardt, D. W. (1963). An algorithm for least-squares estimation of nonlinear parameters.*Journal of the Society for Industrial and Applied Mathematics, 11*(2), 431-441. http://dx.doi.org/10.1137/0111030.

Schmuker, U., & Wiedelt, P. (1975). Electromagnetic Induction in The Earth, Aarhus Lecture Note, 182pp. http://www.complete-mt-solutions.com/mtnet/papers/ClassicPapers/Schmucker+Weidelt_1975_AarhusLectureNotes.pdf

Zhdanov, M. S. (2002). Geophysical Inverse Theory and Regularization Problems, Methods in Geochemistry and Geophysics, Elsevier, Amsterdam, New York, Tokyo.

APPENDIX A. Algorithm's Script

```
% Forward Modelling MT
function [apparentResistivity, phase] = fmodl(resistivities, thicknesses,frequency)
mu = 4*pi*1E-7; %Magnetic Permeability (H/m)
w = 2 * pi * frequency; %Angular Frequency (Radians);
n=length(resistivities); %number of layers
Cs = zeros(1,n);
for i = 1 : n
    conductivities(i) = 1/resistivities(i);
end
%Calculating basement transfer function
Cn = 1/(sqrt(sqrt(-1)*w*mu*conductivities(n)));
Cs(n) = Cn;
if n > 1
%Calculating transfer function for the upper layer
for j = n-1:-1:1
    conductivity = conductivities(j);
    thickness = thicknesses(j);
%calculating k
    k = sqrt(sqrt(-1)*w*mu*conductivities(j));
%Calculating the next layer transfer function
    atas = k*Cs(j+1)+ tanh(k*thickness);
    bawah = 1 + k*Cs(j+1)*tanh(k*thickness);
    Cs(j) = atas/(k*bawah);
end
end
% Pick the impedance value of the top layer
Z = sqrt(-1)*w*mu*Cs(1);
absZ = abs(Z) ;
apparentResistivity = (absZ * absZ)/(mu * w);
phase = rad2deg(atan2(imag(Z),real(Z)));
% Original Marquardt inversion algorithm
function [marq_f] = marquardtinv(handles)
%Input :
```

```
% - Handles = handles that contain data and starting model
%Output :
% marq_f = inversion output
%initializing arbitrary parameter and get starting model
selesai = 100;
damping_oke = 0;
damping = 1000;
bubar = 0;
[m0,d_inv] = ambilmodelinv2(handles);
n_layers = length(m0);
frekuensi = handles.data(:,1);
app_resistivity = handles.data(:,2)';
fasa = handles.data(:,3)';
%% inversion section
for iterasi = 1 : 31
if selesai <= 1 %inversion break condition
break
end
%creating jacobian matrix using finite difference
J = zeros(length(frekuensi),length(m0));
for i = 1 : length(frekuensi)
      [app_inv1(i),fasa_inv1(i)] = ForwardModel(m0,d_inv,frekuensi(i));
end
misfit = 100*sqrt(mean((([app_resistivity fasa] - ...
                        [app_inv1 fasa_inv1])./[app_resistivity fasa]).^2));
%creating jacobian matrix with model perturbation 1%
h = 1/100;
m1 = m0 + m0*h;
m2 = m0 - m0*h;
m3 = m0;
for i = 1 : length(m0)
      m3(i) = m1(i);
for j = 1 : length(frekuensi)
      [app_inv2(j),fasa_inv2(j)] = ForwardModel(m3,d_inv,frekuensi(j));
end
      m3 = m0;
m3(i) = m2(i);
for j = 1 : length(frekuensi)
      [app_inv3(j),fasa_inv3(j)] = ForwardModel(m3,d_inv,frekuensi(j));
end
      m3 = m0;
J(:,i) = ([app_inv2']-[app_inv3'])/(2*m0(i)*h);
    J2(:,i) = (fasa_inv2' - fasa_inv3')/(2*m0(i)*h);
```

```
end
J = [J;J2];
selesai = misfit;
if iterasi == 1 % iteration 1 = starting model
      app_invs = app_inv1;
      fasa_invs = fasa_inv1;
else
while damping_oke == 0
hlm = svd_inv(J'*J + damping*diag(diag(J'*J)))*J'*...
            ([app_resistivity';fasa'] - [app_inv1';fasa_inv1']);
%update parameter
      m = m0 + hlm;
for i = 1 : length(frekuensi)
[app_invs(i),fasa_invs(i)] = ForwardModel(m,d_inv,frekuensi(i));
end
      misfit2 = 100*sqrt(mean((([app_resistivity fasa] - ...
            [app_invs fasa_invs])./[app_resistivity fasa]).^2));
%checking if the misfit reduced
if misfit2 < misfit
              damping_oke = 1; %accept model and damping parameter
damping = damping/2; %reducing damping parameter
            m0 = abs(m);
            misfit = misfit2;
            selesai = misfit;
else
            damping = damping*2; %increasing damping parameter
            bubar = bubar + 1;
if bubar >= 30
break
                  bubar = 9999;
end
 end
end
 end
if bubar >= 1000
break
else
damping_oke = 0;
%saving result
marq_f(iterasi) = struct('rhoapp',app_invs,'fasa',fasa_invs,'model',...
                  m0,'d',d_inv,'misfit',selesai,'J',J);
end
end
```

```
% Modified Marquardt Inversion using SVD
function [marq_f] = marquardtinv_svd(handles)
%Input :
% - Handles = handles contain data and starting model
%Output :
% marq_f = inversion output
%initializing arbitrary parameter and get starting model
selesai = 100;
damping_oke = 0;
damping = 1000;
bubar = 0;
[m0,d_inv] = ambilmodelinv2(handles);
n_layers = length(m0);
m0 = [log10(m0);log10(d_inv)];
frekuensi = handles.data(:,1);
app_resistivity = handles.data(:,2)';
fasa = handles.data(:,3)';
%% Inversion section
for i = 1 : length(frekuensi)
  [app_inv1(i),fasa_inv1(i)] = ForwardModel(10.^m0(1:n_layers)...
            ,10.^m0(n_layers:length(m0)),frekuensi(i));
end
misfit = 100*sqrt(mean((([app_resistivity fasa] - ...
                    [app_inv1 fasa_inv1])./[app_resistivity fasa]).^2));
for iterasi = 1 : 30
if selesai <= 1
break
end
%creating jacobian matrix using finite difference
J = zeros(length(frekuensi),length(m0));
%creating jacobian matrix with perturbation model 1%
h = 1/100;
m1 = m0 + abs(m0*h);
m2 = m0 - abs(m0*h);
m3 = m0;
for i = 1 : length(m0)
      m3(i) = m1(i);
for j = 1 : length(frekuensi)
      [app_inv2(j),fasa_inv2(j)] = ForwardModel(10.^m3(1:n_layers)...
            ,10.^m3(n_layers+1:length(m0)),frekuensi(j));
end
      m3 = m0;
m3(i) = m2(i);
```

```
%         m3(i) = m0(i);
for j = 1 : length(frekuensi)
      [app_inv3(j),fasa_inv3(j)] = ForwardModel(10.^m3(1:n_layers)...
           ,10.^m3(n_layers+1:length(m0)),frekuensi(j));
end
      m3 = m0;
    J(:,i) = (log10(app_inv2')-log10(app_inv3'))/(2*m0(i)*h);
    J2(:,i) = (fasa_inv2' - fasa_inv3')/(2*m0(i)*h);
end
J = [J;J2];
%% SVD
%iteration 1 = starting model (for analysis)
if iterasi > 1
E =([log10(app_resistivity) fasa] - [log10(app_inv1) fasa_inv1]);
err_level = 0.1;
[J_M J_N] = size(J);
if J_M >= J_N
[Ju,Js,Jv] = svd(J,0);
else
      [Jv,Js,Ju] = svd(J,'econ');
end
Jsd = diag(Js);
% make k = si/s1 dan S+ = 1/si
for i = 1 : length(Jsd)
if Jsd(i) == 0
      Splus(i) = 0;
else
      Splus(i) = 1/Jsd(i);
end
      J_k(i) = Jsd(i)/Jsd(1);
end
Splus = diag(Splus);
%creating damping matrix
for i = 1 : J_N
if J_k(i) >= err_level^2
      T(i) = J_k(i)^(4)/(J_k(i)^(4)+err_level^4);
else
            T(i) = 0;
end
end
T = diag(T);
Bplus = Jv*T*Splus*Ju';
if J_M >= J_N
```

```
dm = Bplus*E';
else
    dm = Bplus'*E;
end
m0 = m0 + dm;
%calculate update
for i = 1 : length(frekuensi)
    [app_inv1(i),fasa_inv1(i)] = ForwardModel(10.^m0(1:n_layers)...
        ,10.^m0(n_layers+1:length(m0)),frekuensi(i));
%calculate ratio between calculated and observed apparent resistivity
    SF = app_resistivity/app_inv1;
end
phsmis1 = 100*sqrt(mean((([fasa] - [fasa_inv1])./[fasa]).^2));
% scaled misfit
for i = 1 : length(frekuensi)
    [app_inv999(i),fasa_inv999(i)] = ForwardModel(10.^m0(1:n_layers)*SF...
        ,10.^m0(n_layers+1:length(m0)),frekuensi(i));
end
misfit = 100*sqrt(mean((([app_resistivity fasa] - ...
                [app_inv1 fasa_inv1])./[app_resistivity fasa]).^2));
misfit999 = 100*sqrt(mean((([app_resistivity fasa] - ...
                [app_inv999 fasa_inv999])./[app_resistivity fasa]).^2));
%calibrate apparent resistivity when the phase is fit
if phsmis1 < 50
    m0(1:n_layers) = log10(10.^m0(1:n_layers)*SF);
    app_inv1 = app_inv999;
    fasa_inv1 = fasa_inv999;
    misfit = misfit999;
    d_inv = 10.^m0(n_layers+1:length(m0));
end
end
selesai = misfit;
clear T
clear Splus
%saving result
marq_f(iterasi) = struct('rhoapp',app_inv1,'fasa',fasa_inv1,'model',...
            10.^m0(1:n_layers),'d',d_inv,'misfit',selesai,'J',J);
end
end
```