Full Newton Inversion of 3D Magnetotelluric Data

D. Varılsüha[1]
[1]Istanbul Technical University, deniz.varilsuha@itu.edu.tr

SUMMARY

The three-dimensional (3D) electromagnetic (EM) inversion performed using the Gauss-Newton and the full Newton algorithms is compared to evaluate the performance and robustness. The inversion requires the calculation of the Hessian matrix, containing the second derivatives of the objective functional, and the gradient vector, composed of the first derivatives of the objective functional. The main purpose of this work is to investigate the benefits and computational aspects of the full Newton method which unlike the Gauss-Newton method includes the higher-order derivatives in the Hessian matrix. The findings of this study suggest that when only the conductivity parameters are considered in an inversion, the Newton method is computationally twice as expensive. At the same time, when the distortion parameters are considered to be affecting the data types and they are to be estimated in inversion, the computational cost increases to 2.5 times. On the other hand, the Newton method is found to be slightly smoother than the Gauss-Newton approach. The calculated higher derivatives in the Hessian matrix will be more significant when more non-linear data types, such as the phase and amplitude tensor, are to be used in inversion.

Keywords: Magnetotellurics, Newton Inversion, 3D, GPU

INTRODUCTION

The Gauss-Newton (GN) inversion is based on using the Taylor series on the objective functional to obtain the first (**g**) and the second derivatives (**H**) and solving the parameter update vector, $\Delta\mathbf{m}$, using those first and second derivatives. The second derivative term which is often called the Hessian matrix (**H**), contains the multiplication of the sensitivity matrices and also the second derivatives of the sensitivity matrix. The GN approach truncates the latter term, the high-order derivatives but it is also possible to calculate this term efficiently as it is suggested by Pratt et al. (1998) for the seismic inversion.
In that study, the sensitivity matrix and the high-order derivatives are calculated explicitly however it is also possible to solve the equations iteratively without explicitly forming the **H** matrix or the sensitivities thus only its multiplication with a vector can be calculated which makes the implementation of the full Newton method is much easier. In this study, the Newton method is explained for the inversion of frequency domain 3D MT problem.

METHODS

The objective functional can be defined for the 3D MT problem as,

$$\mathrm{U}(\mathbf{m},\mathbf{D}) = \|(\mathbf{d}-\mathbf{F})\|_2^2 + \lambda\|\mathbf{Cm}\|_2^2 + \kappa\|\mathbf{D}-\mathbf{D}_0\|_2^2, \quad (1)$$

where **d** is for data vector **F** is the forward solution vector of a given model conductivity **m**. **C** is the matrix that controls the model roughness. **D** and $\mathbf{D}_0$ are the vectors for the distortion parameters and the parameters for the non-distorted case. The data and forward solution vector should be multiplied by a data weighting matrix, such as $\mathbf{W}_\mathrm{d}$, however, it is omitted here for simplicity.
For the forward modeling, a linear system equation given in the following equation is solved for

$$\mathbf{Ax} = \mathbf{b} \text{ or } \mathbf{x} = \mathbf{A}^{-1}\mathbf{b}, \quad (2)$$

where **A** is the coefficient matrix and the vector **b** is populated by the non-zero boundary conditions for the 3D MT problem. Even though the problem is solved for two polarizations and the whole frequency range, I simplified the expression as Equation 2 suggests and I call the whole forward modeling routine as one forward solution. The term pseudo forward solution refers to the case of the right-hand side vector **b**, replaced with another vector, and the solution is obtained accordingly. The matrix **A** is complex-symmetric and non-Hermitian so its transpose won't be explicitly stated when it is necessary. The gradient of Equation 1 is stated as,

$$\mathbf{g} = -2\mathbf{J}^\mathrm{T}(\mathbf{d}-\mathbf{F})^* + 2\begin{bmatrix}\lambda\mathbf{C}^\mathrm{T}\mathbf{Cm}\\ \kappa(\mathbf{D}-\mathbf{D}_0)\end{bmatrix}, \quad (3)$$

where $*$ denotes the complex conjugate of a

vector/matrix since the data used in this study is complex-valued impedance and magnetic transfer function (MTF) and the sensitivity matrix **J** is also complex-valued.

The forward solution is obtained via a function ($\mathbf{F} = \mathbf{z}(\mathbf{x}, \mathbf{D})$), and its derivative to parameters will result in the Jacobian matrix using the chain rule in the derivation,

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{F}}{\partial \mathbf{m}} & \frac{\partial \mathbf{F}}{\partial \mathbf{D}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{m}} & \frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{D}} \end{bmatrix}. \quad (4)$$

The vector **x** and its derivative are found by taking the derivative of Equation 2,

$$\frac{\partial (\mathbf{A_m x_m = b})}{\partial \mathbf{m}} \rightarrow \mathbf{A'x} + \mathbf{Ax'} = \mathbf{0} \rightarrow \mathbf{x'} = \mathbf{A^{-1}}(-\mathbf{A'x}). \quad (5)$$

The prime ($'$) denotes the derivatives w.r.t the **m** for simplicity. For my implementation, the right-hand side vector (**b**) doesn't depend on **m** so that its derivative is zero. Egbert and Kelbert (2012) use certain letters for these derivatives and Equation 4 can be written in their notation the following way for simplicity,

$$\mathbf{J} = [\mathbf{LA^{-1}P} \quad \mathbf{Q}], \quad (6)$$

where **L**, **P,** and **Q** are all sparse matrices. One wishes to solve the equation given in $\mathbf{LA^{-1}P}$ to obtain the full Jacobian however it would be easier to calculate its product with a vector since the matrices **L** or **P** will reduce to a vector and only one pseudo-forward solution will be needed.

The Hessian matrix is defined in the following way if we take the derivative of the gradient w.r.t the parameters, **m** and **D**, again.

$$\mathbf{H} = 2\mathbf{J}^{*\mathrm{T}}\mathbf{J} + 2\begin{bmatrix} \lambda \mathbf{C}^{\mathrm{T}}\mathbf{C} & 0 \\ 0 & \kappa \mathbf{I} \end{bmatrix} + \mathbf{R}, \quad (7)$$

where **I** represent the identity matrix with the length of the distortion parameters The matrix **R** represents the higher order derivatives which is omitted naturally for the GN inversion. However, in the Newton inversion, we will include them in the Hessian. The higher-order derivatives are stated with,

$$\mathbf{R} = -2\begin{bmatrix} \frac{\partial \mathbf{J}^{\mathrm{T}}}{\partial \mathbf{m}} & \frac{\partial \mathbf{J}^{\mathrm{T}}}{\partial \mathbf{D}} \end{bmatrix} (\mathbf{d} - \mathbf{F})^*, \quad (8)$$

where **R** is a symmetric matrix and the second derivatives in big brackets are 3D tensors however when they are multiplied by $(\mathbf{d} - \mathbf{F})^*$, or $\Delta\mathbf{d}^*$for short, it will reduce to a matrix again. If we analyze this tensor in detail the following is obtained

$$\begin{bmatrix} \frac{\partial \mathbf{J}^{\mathrm{T}}}{\partial \mathbf{m}} & \frac{\partial \mathbf{J}^{\mathrm{T}}}{\partial \mathbf{D}} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial \mathbf{m}}\left(\frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{m}}\right)^{\mathbf{T}} & \frac{\partial}{\partial \mathbf{D}}\left(\frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{m}}\right)^{\mathbf{T}} \\ \frac{\partial}{\partial \mathbf{m}}\left(\frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{D}}\right)^{\mathbf{T}} & \frac{\partial}{\partial \mathbf{D}}\left(\frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{D}}\right)^{\mathbf{T}} \end{bmatrix} \quad (9)$$

The distortion tensor (**D**) acts on the impedance data linearly so that the lower right corner of this tensor is zero naturally. The off-diagonal parts of this tensor are the transpose of each other. The upper right corner term could be stated in the following way when it is multiplied with $\Delta\mathbf{d}^*$,

$$\mathbf{P}^{\mathbf{T}}\mathbf{A}^{-1}\left(\frac{\partial \mathbf{L}^{\mathbf{T}}}{\partial \mathbf{D}} \Delta\mathbf{d}^*\right) = \mathbf{P}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{L}_{\mathbf{D}}^{\mathbf{T}} \quad (10)$$

where $\mathbf{L_D}$ is a matrix defined by the term giving in parenthesis and it should be prepared beforehand. The lower left corner of the matrix **R** is the transpose of the Equation 10, $\mathbf{L_D}\mathbf{A}^{-1}\mathbf{P}$.

The upper left corner requires the chain rule for the second derivative and it can stated this way,

$$\frac{\partial}{\partial \mathbf{m}}\left(\frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{m}}\right)^{\mathbf{T}} \Delta\mathbf{d}^* = \frac{\partial^2 \mathbf{x}}{\partial \mathbf{m}^2}^{\mathbf{T}} \frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}}^{\mathbf{T}} \Delta\mathbf{d}^* + \frac{\partial \mathbf{x}}{\partial \mathbf{m}}^{\mathbf{T}} \left[\frac{\partial^2 \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}^2} \Delta\mathbf{d}^*\right] \frac{\partial \mathbf{x}}{\partial \mathbf{m}}. \quad (11)$$

The second term in brackets on the right side given in the equation above can be expressed more clearly in such form

$$\mathbf{G} = \frac{\partial^2 \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}^2} \Delta\mathbf{d}^* = \sum_{\mathrm{i}=1}^{\mathrm{N_d}} \begin{bmatrix} \frac{\partial^2 \mathbf{z}_\mathrm{i}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}_1^2} & \frac{\partial^2 \mathbf{z}_\mathrm{i}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}_1 \partial \mathbf{x}_2} \\ \frac{\partial^2 \mathbf{z}_\mathrm{i}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}_2 \partial \mathbf{x}_1} & \frac{\partial^2 \mathbf{z}_\mathrm{i}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}_2^2} \end{bmatrix} \Delta\mathbf{d}_i^*. \quad (12)$$

The first derivative of the function **z** w.r.t the vector **x** is represented by **L** (Equation 6). The expression above requires the second derivative. The vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ represent the solutions for two polarizations. $\mathrm{N_d}$ represents the number of data and matrix in big brackets are symmetric, the off-diagonal parts are transpose of each other. So, the matrix **G** is symmetric. Finally, the second term in Equation 11 can be written in such a way,

$$\frac{\partial \mathbf{x}}{\partial \mathbf{m}}^{\mathbf{T}} \left[\frac{\partial^2 \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}^2} \Delta\mathbf{d}^*\right] \frac{\partial \mathbf{x}}{\partial \mathbf{m}} = \mathbf{P}^{\mathbf{T}}\mathbf{A}^{-1}\mathbf{G}\mathbf{A}^{-1}\mathbf{P}\,. \quad (13)$$

The first part in Equation 11 requires the second derivative of **x** w.r.t the parameter vector **m**. This can be obtained by deriving the Equation 5 w.r.t. **m** again in such a way,

$$\frac{\partial (\mathbf{A'x + Ax' = 0})}{\partial \mathbf{m}} \rightarrow \mathbf{A''x} + \mathbf{A'x'} + (\mathbf{A'x'})^{\mathbf{T}} + \mathbf{Ax''} = \mathbf{0} \rightarrow \mathbf{x''} = \mathbf{A}^{-1}(-\mathbf{A''x} - \mathbf{A'x'} - (\mathbf{A'x'})^{\mathbf{T}}), \quad (14)$$

where the expressions in parenthesis are all 3D tensors that will be reduced to a matrix when multiplied with a vector. The whole term for the upper left corner of **R** can be expressed as,

$$\frac{\partial}{\partial \mathbf{m}}\left(\frac{\partial \mathbf{z}(\mathbf{x},\mathbf{D})}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \mathbf{m}}\right)^{\mathbf{T}} \Delta\mathbf{d}^* = (-\mathbf{A''x} - \mathbf{A'x'} - (\mathbf{A'x'})^{\mathrm{T}})^{\mathrm{T}}\mathbf{A}^{-1}\mathbf{L}^{\mathrm{T}}\Delta\mathbf{d}^*. \quad (15)$$

The vector term $\mathbf{A}^{-1}\mathbf{L}^{\mathrm{T}}\Delta\mathbf{d}^*$ is already calculated in Equation 3 for the gradients (**g**). The terms in transpose are tricky parts to compute. For that reason, I will refer to the technique first proposed by Egbert and Kelbert (2012) and further developed by Varılsüha (2020) for much more complex mesh types, partial differential equations, and numerical methods. The coefficient matrix **A** can be written linearly w.r.t. the parameter vector **m** as a summation of sparse matrices such as,

$$\mathbf{A} = \sum_\mathrm{i} \mathbf{Y}_\mathrm{i} \mathrm{diag}(\mathbf{W}_\mathrm{i} \mathrm{e}^{\mathbf{m}}) + \sum_\mathrm{j} \mathbf{M}_\mathrm{j}, \quad (16)$$

where **Y**'s and **M**'s are sparse matrices and hold the coefficients. **W**'s are mapping/averaging matrix acts on the conductivities which are defined in a logarithmic fashion ($\mathrm{e}^{\mathbf{m}}$). The derivative of **A** and its

multiplication with a vector **y** can be defined as,

$$\mathbf{A'y} = \sum_i \mathbf{Y_i}\,\mathrm{diag}(\mathbf{y})\mathbf{W_i}\,\mathrm{diag}(\mathbf{e^m}). \quad (17)$$

It is possible to further multiple $\mathbf{A'y}$ with a vector **q** on the left side, which will come in handy later on,

$$\mathbf{q^T A'y} = \mathbf{y^T}(\sum_i \mathrm{diag}(\mathbf{q^T Y_i})\mathbf{W_i})\mathrm{diag}(\mathbf{e^m}). \quad (18)$$

So if we refer to $\mathbf{A^{-1}L^T\Delta d^*}$ as $\mathbf{s_1}$ then,

$$-(\mathbf{A'x'})^{\mathrm{T}}\mathbf{s_1} = \mathrm{diag}(\mathbf{e^m})\left(\sum_i \mathrm{diag}\left(\mathbf{s_1^T Y_i}\right)\mathbf{W_i}\right)^{\mathrm{T}}\mathbf{A^{-1}P}. \quad (19)$$

Similarly, its non-transposed version can be written such as,

$$-(\mathbf{A'x'})\mathbf{s_1} = \mathbf{P^T A^{-1}}\left(\sum_i \mathrm{diag}\left(\mathbf{s_1^T Y_i}\right)\mathbf{W_i}\right)\mathrm{diag}(\mathbf{e^m}). \quad (20)$$

Both $(\mathbf{A'x'})^{\mathrm{T}}\mathbf{s_1}$ and $(\mathbf{A'x'})\mathbf{s_1}$ are the transpose of each other so that their summation will result in a symmetric matrix as it should.

The last term in Equation 15, $-\mathbf{A''x}$ can be expressed such as,

$$-\mathbf{s_1^T A''x} = -\mathrm{diag}\left(\mathbf{e^m}.\left(\mathbf{x^T}\left(\sum_i \mathrm{diag}\left(\mathbf{s_1^T Y_i}\right)\mathbf{W_i}\right)\right)^{\boldsymbol{T}}\right), \quad (21)$$

where '.' is the dot product between two vectors. The resulting matrix is again a symmetric diagonal matrix. During each step of GN inversion, the system of $\mathbf{H\Delta m} = -\mathbf{g}$ can be solved iteratively with an iterative solver without explicitly forming the Hessian. To evaluate the **g** at the beginning of the iterative process one pseudo-forward solution is required. During the iterative process, the Hessian must be multiplied with an arbitrary vector **v** once in each iteration to solve the GN equations. This ultimately means multiplying $\mathbf{J^{*T}J}$ by this vector **v**

$$\mathbf{v} = \begin{bmatrix}\mathbf{v_m}\\ \mathbf{v_D}\end{bmatrix} \quad (22)$$

where $\mathbf{v_m}$ has the size of parameter vector **m** and $\mathbf{v_D}$ has the size of distortion vector **D.**

Equation 6 states the structure of the Jacobian matrix so that each iteration for the GN method requires two pseudo-forward solutions. While multiplying and calculating $\mathbf{Jv}$, it is necessary to save $\mathbf{A^{-1}Pv_m}$ as $\mathbf{s}_2$ along with $\mathbf{s}_1$ because it will be used in Newton's inversion.

As for the Newton inversion, in addition to GN operations, Equations 10, 13, 19, 20, and 21 should be evaluated. Equation 10 requires one extra pseudo-forward solution when multiplied with $\mathbf{v_D}$ on the right. As for its transposed version for the lower left corner of matrix **R**, the vector $\mathbf{s}_2$ can be used to save time. We can use $\mathbf{s}_2$ again for Equation 13 when it is multiplied on the right side, however for its left side terms, we need one more pseudo-forward solution. $\mathbf{s}_2$ can be used in Equation 19 too and this expression doesn't require any additional computation. However, its transposed version given in Equation 20 requires an additional pseudo-forward solution as it is multiplied by $\mathbf{v_m}$ on the right side. Finally, Equation 21 doesn't require any matrix solutions.

RESULTS

To compare the inversion between the GN and Newton inversion algorithms, a synthetic data with distortion and noise is chosen. This data belongs to a model called the two-mountain model which is depicted in Figure 1. Its author (Usui, 2015) used unstructured tetrahedral mesh to generate the model response and, in this study, I'm inverting it using a hexahedral mesh and two of the inversion algorithms considered in this study.

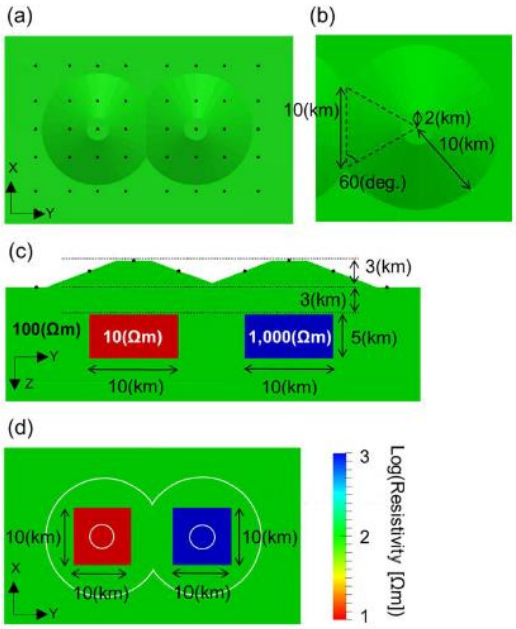

Figure 1. The two-mountain model is depicted by Usui (2015). a) and b) shows the surface and the mountains' dimensions. c) shows the vertical and d) shows the horizontal cross-sections where the conductive and resistive anomalies are present.

The data contains 16 frequencies ranging from 10Hz to 10^{-3} Hz. There are 40 receiver locations which are shown in Figure 1a. To invert this data, a parameter mesh with 116K conductivity parameters is created. Additionally, 160 distortion parameters are defined and estimated in inversion.

All the forward and pseudo-forward solutions are performed until the iterative solver reaches a relative residual norm of 10^{-9}. For both inversion algorithms, I started with a layered initial model obtained from the 1D inversion of data. The GN and Newton inversion took 28 and 27 inversion steps respectively and they lowered the nRMS values to 1.021 to 1.035 respectively from their starting point of 6. To perform the calculations, a workstation with 8 GPUs (RTX 3090s) is used and all the performance critical calculations are performed on these cards. Finally, The GN inversion took 22 minutes to complete while the Newton inversion took 61 minutes.

The results suggest that both inversions did a good job and estimated the conductive and resistive

structures without being affected by the distortion of data as shown in Figure 2. However, the Newton inversion has found the conductive anomalies values closer to actual values. The resistive structure which is estimated by the Gauss-Newton inversion is slightly out of its bounds, unlike the Newton inversion.

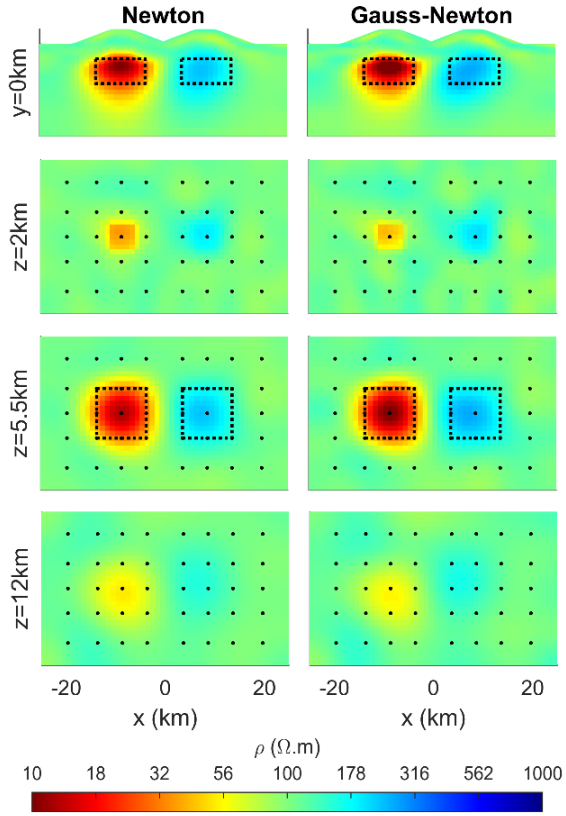

Figure 2. The model obtained by the Newton inversion and the left and the Gauss-Newton inversion on the right. Dots represent the receiver locations while the dashed lines indicate the true location of the anomalies. Red blacks indicate the resistivity values below 10 Ωm which is more apparent in the GN inversion for undershooting values.

Discussion

In this study, only the impedance tensor and MTF are considered as data in both inversions. Geophysical problems are ill-posed problems which means they are non-unique, non-linear, and unstable. For that reason, it would not be wise to use more non-linear data types other than the abovementioned data types. In theory, it would be unnecessary to include the higher derivatives if the problem acts more like a linear one. On the other hand, the 3D MT problem has a lot of data types that show strong non-linearity (i.e. the phase tensor, and amplitude tensor). One might wish to use these data types in inversion and use the benefit of the Newton inversion. I believe the most important aspect of the Newton inversion would be its distortion parameter relation. The objective functional in Equation 1 contains two trade-off parameters to balance the model roughness and distortion intensity. It is very easy to lose real structures in the distortion parameters. The off-diagonal parts of the higher derivative term given in Equation 9 define a cross-derivation between the conductivity and distortion parameters. It should be able to provide stability between different parameter types, which is the subject of further study.

Conclusions

The implementation details of the Newton method are shown for the 3D MT inversion and the results are compared with the GN approach. The GN approach requires two pseudo-forward solutions during the iterative process to solve the parameter update vector. If only the conductivity parameters are in interest, the Newton method doubles this amount and thus the computational time. Additionally, the distortion parameters are included in the objective functional an additional pseudo-forward solution is required for the higher derivative terms. The results suggest that the Newton method finds the conductive anomaly in the model used much closer to its actual conductivity value. The GN inversion undershoots the low conductivity values by a bigger margin. The resistive structure on the other hand is slightly out of its bounds when the GN algorithm is used. On the other hand, the Newton algorithm finds it within the given bounds for the model considered in this study. Overall the Newton inversion provided a much smoother model in inversion with the data used in this study.

Acknowledgments

I'd like to thank C.G. Farquharson for his mentorship. I also want to acknowledge M. Neukirch and A. Minakov who led me to investigate a more robust inversion algorithm.

References

Egbert GD, Kelbert A (2012) Computational recipes for electromagnetic inverse problems, Geophys. J. Int., 189, 251-267.

Pratt RG, Shin C, Hicks GJ (1998) Gauss-Newton and full Newton methods in frequency-space seismic waveform inversion. Geophys. J. Int., 133, 341-362.

Usui Y (2015) 3-D inversion of magnetotelluric data using unstructured tetrahedral elements: applicability to data affected by topography. Geophys. J. Int., 202, 828-849.

Varılsüha D (2020) 3D inversion of magnetotelluric data by using a hybrid forward-modeling approach and mesh decoupling, Geophysics, 85(5), E191-E205.