Full Newton Inversion of 3D Magnetotelluric Data

D. Varılsüha[1]

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

Introduction

The Gauss-Newton (GN) algorithm is often preferred for the inversion of the three-dimensional EM data due to its robustness and convergence speed when compared to inversion algorithms such as NLCG or LBFGS. Unlike these only gradient-based inversion algorithms, the GN approach requires the calculation of the Hessian matrix, either explicitly or its product with a vector implicitly.

The Hessian matrix also contains higher-order derivatives and these derivatives are omitted for simplicity for the GN approach. However, for the Newton approach, these high-order derivatives are calculated and included in the Hessian.

In this work, the comparison of the GN and Newton inversion algorithms is presented using a model with topography and its synthetic data which contains distortion and noise.

While performing these inversions, the sensitivity matrix is not explicitly formed, and thus the Hessian. Instead, the product of Hessian and a vector is calculated implicitly using an iterative solver (GMRES). Pratt et al. (1998) showed the Newton inversion for time-domain seismic problems, on the other hand, this could be the first Newton inversion applied to an EM problem.

Theory

The objective functional to be minimized is defined as,

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

Where **d** is data, **F** is the forward modeling operator, **C** is the covariance matrix, **m** is for conductivity parameters, **D** is the distortion parameters to be estimated and $\mathbf{D}_0$ holds the values for the non-distorted case. λ and κ are the trade-off parameters.

The gradient (**g**) of the objective functional is obtained as below after using the Taylor series and using the second term's derivative w.r.t **m** and **D**,

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

where **J** is the sensitivity matrix. The Hessian matrix, **H**, obtained as,

$$\mathbf{H} = 2\mathbf{J}^{*\mathrm{T}}\mathbf{J} + 2\begin{bmatrix}\lambda\mathbf{C}^{\mathrm{T}}\mathbf{C} & 0\\ 0 & \kappa\mathbf{I}\end{bmatrix} + \mathbf{R}\,. \quad (3)$$

For the GN inversion algorithm, the matrix **R** is omitted. For the forward modeling, a linear matrix system should be solved to obtain the vector **x**.

$$\mathbf{Ax}=\mathbf{b} \;\text{ or }\; \mathbf{x}=\mathbf{A}^{-1}\mathbf{b}, \quad (4)$$

then the impedances or other data types are calculated via,

$$\mathbf{F}=\mathbf{z}(\mathbf{x},\mathbf{D}). \quad (5)$$

The sensitivity matrix is obtained in the following fashion,

$$\mathbf{J}=\begin{bmatrix}\frac{\partial\mathbf{F}}{\partial\mathbf{m}} & \frac{\partial\mathbf{F}}{\partial\mathbf{D}}\end{bmatrix}=\begin{bmatrix}\frac{\partial\mathbf{z}(\mathbf{x},\mathbf{D})}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\mathbf{m}} & \frac{\partial\mathbf{z}(\mathbf{x},\mathbf{D})}{\partial\mathbf{D}}\end{bmatrix}. \quad (6)$$

The left part of the **J** is calculated using the chain rule where the derivative of **x** w.r.t. **m** is calculated the following way,

$$\frac{\partial(\mathbf{A}_\mathbf{m}\mathbf{x}_\mathbf{m}=\mathbf{b})}{\partial\mathbf{m}} \rightarrow \frac{\partial\mathbf{x}}{\partial\mathbf{m}} = \mathbf{A}_\mathbf{0}^{-1}\left(-\frac{\partial(\mathbf{A}_\mathbf{m}\mathbf{x}_0)}{\partial\mathbf{m}}\right) \quad (7)$$

Egbert and Kelbert (2012) explain the sensitivity matrix equations in detail and they use the following letters for the sparse matrices to define the Jacobian,

$$\mathbf{J}=[\mathbf{LA}^{-1}\mathbf{P}\quad\mathbf{Q}]\,. \quad (8)$$

In an inversion algorithm, the matrix **J** is usually never explicitly calculated but its product with a vector is calculated implicitly which tends to be computationally extremely cheaper.

For the Newton method, the matrix **R** must taken into account and it is defined as

$$\mathbf{R} = -2\begin{bmatrix}\frac{\partial\mathbf{J}^{\mathrm{T}}}{\partial\mathbf{m}} & \frac{\partial\mathbf{J}^{\mathrm{T}}}{\partial\mathbf{D}}\end{bmatrix}(\mathbf{d}-\mathbf{F})^{*} \quad (9)$$

which is a symmetric matrix. Before multiplying it with (**d-F**), it can be defined in a tensor form.

$$\begin{bmatrix}\frac{\partial\mathbf{J}^{\mathrm{T}}}{\partial\mathbf{m}} & \frac{\partial\mathbf{J}^{\mathrm{T}}}{\partial\mathbf{D}}\end{bmatrix} = \begin{bmatrix}\frac{\partial}{\partial\mathbf{m}}\left(\frac{\partial\mathbf{z}(\mathbf{x},\mathbf{D})}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\mathbf{m}}\right)^{\mathbf{T}} & \frac{\partial}{\partial\mathbf{D}}\left(\frac{\partial\mathbf{z}(\mathbf{x},\mathbf{D})}{\partial\mathbf{x}}\frac{\partial\mathbf{x}}{\partial\mathbf{m}}\right)^{\mathbf{T}}\\ \frac{\partial}{\partial\mathbf{m}}\left(\frac{\partial\mathbf{z}(\mathbf{x},\mathbf{D})}{\partial\mathbf{D}}\right)^{\mathbf{T}} & \frac{\partial}{\partial\mathbf{D}}\left(\frac{\partial\mathbf{z}(\mathbf{x},\mathbf{D})}{\partial\mathbf{D}}\right)^{\mathbf{T}}\end{bmatrix} \quad (10)$$

To obtain the 3D tensor for this term, the second derivative for the derivation chain rule must be applied. Even though the math shows a 3D tensor and its product with a vector, in its implementation, it is not necessary to form any 3D tensors, only the 2D matrices and their products with vectors are used so that the matrix **R** is never formed and only its product with a vector is calculated implicitly.

Algorithm and Software

There are many techniques to solve the GN equations using the Hessian (**H**) and the gradient (**g**). In Table 1, three of them are given and the last one is used for this work.

Table 1. Three different approaches to perform the solution of **Hp**=-**g**. **p** represents the search direction vector, λ represents the trade-off parameter and α is for the step-length. $\mathbf{dm}$ is the parameter update vector.

Aggressive	Standard	This work
• Form the Jacobian (**J**) explicitly • Form the Hessian (**H**) explicitly • solve $\mathbf{H}(\lambda)\mathbf{p}=-\mathbf{g}(\lambda)$ • repeat the process to find the best λ and α values to obtain: $\mathbf{dm}=\alpha\mathbf{p}$	• Keep the λ constant • Never form the **J** explicitly • Solve the **Hp**=-**g** iteratively till stagnation • Do a line search to obtain α • Obtain $\mathbf{dm}=\alpha\mathbf{p}$	• Keep the λ constant • Never form the **J** explicitly • Solve the **Hp**=-**g** iteratively until certain conditions are met (**i.e.** norm(**p**)>10; iter.no>30; $\epsilon < 10^{-2}$) • No line search, $\mathbf{dm}=\mathbf{p}$

The inversion software is developed by Varılsüha (2020) and further changes are made by Varılsüha (2022) to increase its efficiency. The details of the inversion tool and the system it is running on are given in Table 2.

Table 2. The capabilities of the software (DEVA3DMT) used in this work are on the left while the system that has been used is given on the right.

DEVA3DMT	The system
• Structured hexahedral mesh • The vector and scalar potentials formulation (an ungauged PDE) • Hybrid finite-difference — finite-elements numerical method • Mesh decoupling/moving footprint inversion • Triple Mesh Design • MATLAB and CUDA-C languages • Krylov subspace solvers (GPBiCG) to solve the linear equations • Runs only on GPUs to increase its efficiency	• 8-core Intel 10900K CPU • 8 x RTX 3090 Nvidia GPU in parallel • 128GB of main memory • MATLAB R2024a and CUDA Toolkit 12.6

Model and Data

In Figure 1, the two-mountain model can be seen. It is created by Usui (2015) using an unstructured tetrahedral mesh. In Table 3, the details of the data and some of the inversion settings are given.

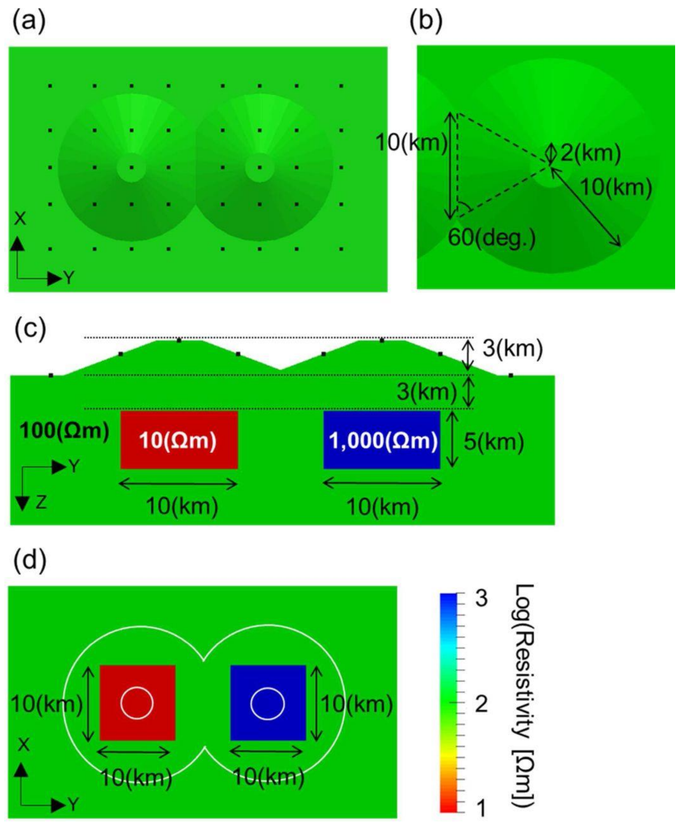


Figure 1. The two-mountain model is depicted by Usui (2015). a) and b) shows the surface and the mountains' dimensions. c) shows the vertical and d) shows the horizontal cross-sections where the conductive and resistive anomalies are present.

Table 3. The data characteristics and some of the inversion settings are presented.

Data and Inversion settings
• 16 frequencies ranging from 10Hz and 10^{-3}Hz. • 40 receiver locations. • Full impedance and MTF data are present. • There is noise and distortion in data. • 116K conductivity and 160 distortion parameters to be estimated in inversion. • $\epsilon = 10^{-9}$ is set for the termination of the forward and pseudo-forward solutions.

Inversion

In Figure 2, the results from Newton, Gauss-Newton, and LBFGS inversions are given. In Figure 3, the statistics for these there inversion algorithms are shown.

Figure 2. Inversion results of the two-mountain model data. Three different algorithms are used to show the difference between the Newton, Gauss-Newton, and LBFGS approaches.

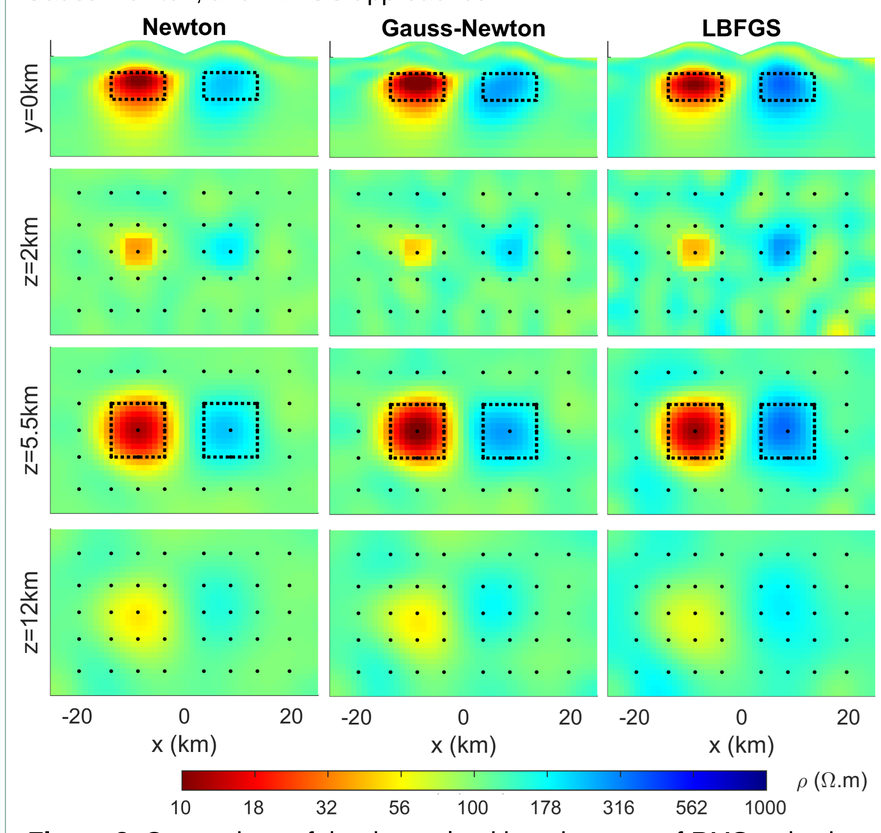
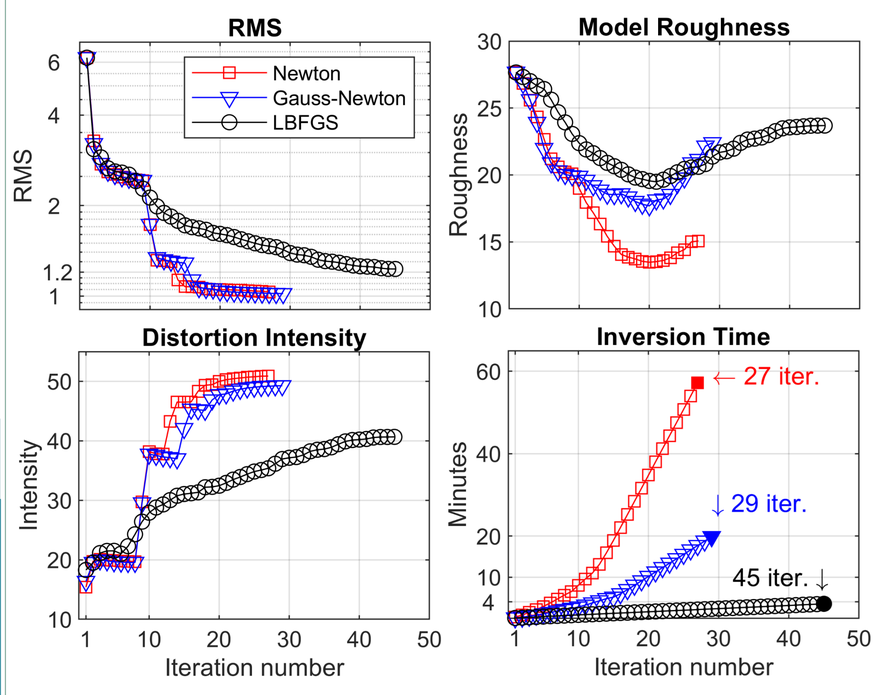


Figure 3. Comparison of the three algorithms in terms of RMS reduction, model roughness, distortion intensity, and inversion time.



Conclusions

- Each GMRES iteration to solve the **H(dm)**=-**g** equations using the GN algorithm requires 2 pseudo-forward solutions.
- For the Newton approach, it requires 5 pseudo forward solutions, this number reduces to 4 when the distortion term is omitted.
- Due to the iterative solvers employed to solve the equations, the Newton method takes 2.5X amount of time while the GN method requires X amount of time.
- When the factorization and backward-forward sweeping approach are used instead of iterative solvers, the time difference will reduce greatly.
- Gauss-Newton and Newton approaches are found to be more robust than the LBFGS.
- For this data set, Newton approach required fewer inversion steps.
- The Newton approach seems to produce smoother models when compared to the GN and LBFGS while using the same trade-off parameters and the same cooling strategy.

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.

Varılsüha, D (2022) Speeding up the inversion of the 3D MT problem, 25th EM Induction Workshop, Çeşme, Turkey, September 11-17, 2022.