Resolving bottlenecks of 3D controlled-source electromagnetic Gauss-Newton inversion.

Anna Avdeeva[1], Rune Mittet[1] and Ole Martin Pedersen[1]
[1]Allton AS

Summary

We propose a new approach to Gauss-Newton optimization to solve the three-dimensional VTI-anisotropic controlled-source electromagnetic inverse problem. The main two bottlenecks of the Gauss-Newton implementation are computation and storage of the large Jacobian matrix, and large memory and time required for the solution to the system of normal equations. To overcome the first issue, the simulation and optimization meshes are decoupled with the use of node-based basis functions. This significantly reduces the number of optimization parameters and hence the memory requirements. The second issue is solved by using a new preconditioner in Conjugate-Gradient solver. Our preconditioner is based on a limited-memory quasi-Newton approximation to the inverse of the Hessian matrix and reduces number of Conjugate-Gradient iterations. This preconditioner is much more efficient than the commonly used Jacobi preconditioner. To further reduce the computational time the code is parallelized in a hybrid manner using MPI and openMP. The method is validated with an off-shore controlled-source electromagnetic dataset acquired at the slow spreading Mohns ridge located east of Greenland and southwest of Svalbard.

Keywords: CSEM, 3D, inversion

Introduction

The solution to a geophysical inverse problem is commonly sought through minimization of an objective function. To find such a minimum in the large-scale 3D case, the gradient-based or Gauss-Newton (GN) optimization methods are employed. While gradient-based methods are less demanding in terms of computer resources, Nguyen et al. (2016) showed that GN method can lead to a much better inversion results.

The GN method relies on the computation of the Jacobian matrix that contains information about the derivatives of the simulated data with respect to conductivity. Storage of the Jacobian matrix requires a huge amount of memory, especially in the 3D case. Moreover, due to the dense structure of the Jacobian matrix, the matrix-vector multiplications needed for solving the GN system of normal equations can become very expensive.

Many researchers proposed strategies to make 3D GN inversions feasible (see, for example, Li et al., 2011; Grayver et al., 2013; Amaya et al., 2016; Mittet & Avdeeva, 2023). Some straightforward solutions are to use a subset of the measured data or coarse inversion grids, but the choice of the subset or the coarsening is subjective and relies on the geophysicist's experience, and may lead to a loss of important information. A more advanced approach by Amaya et al. (2016) combines several source positions for simultaneous-source simulations. The resulting Hessian matrix can be described as a low-rank approximation to the GN Hessian.

To reduce the memory usage and the computation time while preserving the quality of the inversion results, Li et al. (2011) proposes a compressed implicit Jacobian scheme. It is therefore not necessary to store the large Jacobian matrix in memory, but this scheme comes at the price of some computational overhead. In this scheme the Jacobian matrix multiplication with a vector is converted to a matrix-vector operation of the field matrices with this vector. To further mitigate the computational overhead they apply the adaptive cross approximation method to compress one of the field matrices.

We propose an alternative solution to the problem. To reduce memory required by the Jacobian we use node-based basis functions to decouple simulation and optimization meshes. This strategy is discussed in detail in Mittet & Avdeeva (2023), where it is successfuly verified for a 2.5D case. In this abstract we focus on bottlenecks of the numerical implementation and give recepies on how to overcome these bottlenecks. We present a solution that can be applied to a large 3D data sets, with relatively modest computational resources. In addition, to speed up the solution of the system of normal equations a new preconditioner for Conjugate-Gradient (CG) solver is suggested. We validate the inversion on both 2.5D and 3D marine controlled-source electromagnetic (CSEM) datasets.

3D CSEM GN inversion methodology

The goal of a VTI-anisotropic CSEM inversion is to find the distribution of the model parameters $\mathbf{m}$, for example the electrical conductivity $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_v)$, in the volume of interest V from observations of electric and magnetic fields excited by sources $\mathbf{x}_s$ at receivers $\mathbf{x}_r$. This is sought through minimization of an objective functional:

$$\varphi = \varphi_d + \lambda \varphi_s \to \min_{\lambda, \mathbf{m}}, \tag{1}$$

where φ_d and φ_s are the data misfit and regularization terms. The data misfit is a weighted difference betweed predicted and observed data

$$\varphi_d(\mathbf{m}) = \frac{1}{2} \|\mathbf{f}(\mathbf{m}) - \mathbf{d}^{obs}\|^2_{\mathbf{W}_d^T \mathbf{W}_d}, \tag{2}$$

and the regularization term is a standard stabilizing functional based on the gradients of the model, that steer the solution towards a smooth model. Matrix $\mathbf{W}_d$ is a diagonal matrix with the data weights.

After second-order Taylor expansion of equation 1, we obtain the GN system of normal equations

$$\left[\Re\left\{\mathbf{J}_\mathbf{m}^H \mathbf{W}_d^T \mathbf{W}_d \mathbf{J}_\mathbf{m}\right\} + \lambda \nabla^2_\mathbf{m} \varphi_s(\mathbf{m}) + \alpha \mathbf{I}\right] \mathbf{p} = -\Re\left\{\mathbf{J}_\mathbf{m}^H \mathbf{W}_d^T \mathbf{W}_d \left[\mathbf{f}(\mathbf{m}) - \mathbf{d}^{obs}\right]\right\} - \lambda \nabla_\mathbf{m} \varphi_s(\mathbf{m}), \tag{3}$$

here $\mathbf{p}$ is the search direction vector, and $\mathbf{J}_\mathbf{m} \in \mathbb{C}_{N_d \times N_m}$ is the Jacobian matrix, consisting of partial derivatives of the data with respect to model parameters $\mathbf{m}$. N_d and N_m are the number of data and model parameters, respectively. The sought after model update vector $\delta\mathbf{m}$ is found by a line search along the direction $\mathbf{p}$. The vector $\boldsymbol{\sigma}$ is obtained from $\mathbf{m}$ by a set of parameter transformations.

To improve the condition number of the system the matrix on the left of the eq. 3, a small damping term $\alpha\mathbf{I}$ is added. In the beginning of the optimization process, $\alpha = 0.01 * \operatorname{diag}\left[\Re\left\{\mathbf{J}_\mathbf{m}^H \mathbf{W}_d^T \mathbf{W}_d \mathbf{J}_\mathbf{m}\right\} + \lambda \nabla^2_\mathbf{m} \varphi_s(\mathbf{m})\right]$.

For the realistic size VTI 3D inverse problem and in case $\mathbf{m} = \boldsymbol{\sigma}$, the Jacobian matrix $\mathbf{J}_{\boldsymbol{\sigma}}$ can be in the order of hundreds of TB (see Table 2 for a few examples). Such matrices are difficult to store and manipulate even on modern computer architectures. To reduce the memory requirements we propose to decouple the simulation and the optimization domains by a set of parameter transformations which are discussed in the next section. After the transformations the memory requirements for one Jacobian could be reduced from hundreds of TB to tens of TB, or even less depending on the user's choice. This memory requirements can still be large and therefore we propose a hybrid MPI/OpenMP implementation to split the problem between compute nodes and cores of a cluster in a scallable manner. After Jacobian is computed and can be stored in memory, the system of normal eqs. 3 is solved with the Conjugate-Gradient (CG) solver. To improve the convergence of the solver, a new preconditioner is suggested below.

Overcoming bottlenecks

Parameter transformations

In this section we introduce a set of parameter transformations that significantly reduces memory requirements needed for computation and storage of the Jacobian matrix. In total we perform three transformations of parameters:

- from conductivity $\boldsymbol{\sigma} \in \mathbb{R}_{2 \times n_x \times n_y \times n_z}$ to unbounded parameters $\boldsymbol{\chi} \in \mathbb{R}_{2 \times n_x \times n_y \times n_z}$, here n_x, n_y, n_z are number of nodes in simulation mesh along three orthogonal directions;
- from $\boldsymbol{\chi} \in \mathbb{R}_{2 \times n_x \times n_y \times n_z}$ to flat seabed parameters $\tilde{\boldsymbol{\chi}} \in \mathbb{R}_{2 \times n_x \times n_y \times m_z}$. Note that the number of the vertical nodes $m_z < n_z$, and the size of parameter vector is reduced, mostly due to the fact that the water is not anymore part of the domain;
- from $\tilde{\boldsymbol{\chi}} \in \mathbb{R}_{2 \times n_x \times n_y \times m_z}$ to a coarser inversion parameterization $\mathbf{m} \in \mathbb{R}_{2 \times n_x^o \times n_y^o \times n_z^o}$. This transformation is based on node-based basis functions. Superscript o denotes optimization mesh.

These transformations are discussed in detail in Mittet & Avdeeva (2023), here we only focus on the last one, since straightforward implementation of this transform can lead to large memory requirements and significant computational time.

To make the explanations simple we resort to a 1D case for now. As mentioned in Mittet & Avdeeva (2023), the parameter $\tilde{\chi}(x)$ can be considered continuous, if on the interval $[0, x_{max}]$

$$\tilde{\chi}(x) = \sum_n \tilde{\chi}_n \phi_n(x), \tag{4}$$

where $\tilde{\chi}_n$ and $\phi_n(x)$ are node values and sinc basis funtions centered at nodes x_n, respectively. The sampling interval is the same as in a simulation grid.

Alternative representation of $\tilde{\chi}(x)$ with a new set of basis functions ψ_ν and node values m_ν is

$$\tilde{\chi}(x) = \sum_\nu m_\nu \psi_\nu(x). \tag{5}$$

In this representation the sampling interval is 2 to 5 times coarser than in the simulation domain. The coefficients m_ν are the unknowns in the optimization problem.

In the 1D case, to perform the transformations from $\tilde{\boldsymbol{\chi}}$ to $\mathbf{m}$ and back, we form three transformation matrices

$$\begin{aligned} A^{(x)}_{ln} &= \int_0^{x_{max}} \phi_n(x)\phi_l(x)dx, \\ B^{(x)}_{\gamma n} &= \int_0^{x_{max}} \psi_\gamma(x)\phi_n(x)dx, \\ C^{(x)}_{\nu\gamma} &= \int_0^{x_{max}} \psi_\nu(x)\psi_\gamma(x)dx. \end{aligned} \tag{6}$$

For the 3D case, much larger matrices $\mathbf{C}$, $\mathbf{B}$ and $\mathbf{A}$ are computed. For example,

$$C_{\Gamma\Lambda} = C^{(x)}_{\nu\gamma} C^{(y)}_{\alpha\beta} C^{(z)}_{\lambda\kappa}. \tag{7}$$

Matrices $\mathbf{A}$, $\mathbf{B}$ are formed similarly.

With these matrices the parameter transforms are given as

$$\mathbf{A}\tilde{\boldsymbol{\chi}} = \mathbf{B}^T\mathbf{m}, \tag{8a}$$

$$\mathbf{C}\mathbf{m} = \mathbf{B}\tilde{\boldsymbol{\chi}}. \tag{8b}$$

The sizes of the transform matrices depend on the number of elements in the parameter vectors $\tilde{\boldsymbol{\chi}}$ and $\mathbf{m}$ and can become large. However, due to the use of nodal basis functions $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are sparse. Examples of the memory requirements for these matrices are shown in Table 1.

In the 1D and 2.5D cases, it is possible to perform and store their factorizations once, before the initial inversion iteration, and use direct solvers, such as MUMPS, to perform the necessary transformations. In the 3D case, the matrices are large, especially matrix $\mathbf{A}$, and factorization and direct solvers are not anymore a suitable choice. In the 3D case, a CG solver with the Jacobi preconditioner is used. Since both matrices $\mathbf{A}$ and $\mathbf{C}$ are diagonal-dominant the Jacobi preconditioner gives good convergency, normally with less than 10 iterations to achieve relative residual error of 10^{-5}. We employ a CG solver from the Eigen C++ library (see Guennebaud et al., 2010).

During the inversion we need to compute the Jacobian with respect to the parameters $\mathbf{m}$,

$$\mathbf{J_m} = \mathbf{J}_{\tilde{\boldsymbol{\chi}}}\mathbf{A}^{-1}\mathbf{B}^T. \tag{9}$$

Due to the large size of the matrix $\mathbf{A}$ and the fact that its inverse is not sparse, instead we use the following approximation

$$\mathbf{J_m} \approx \mathbf{J}_{\tilde{\boldsymbol{\chi}}}\mathrm{diag}^{-1}(\mathbf{A})\,\mathbf{B}^T. \tag{10}$$

Our tests showed that this approximation is sufficiently good.

Note that we do not need to compute the whole matrix $\mathbf{J}_{\tilde{\boldsymbol{\chi}}}$, but depending on the available memory we can compute only a small number of rows, that we transform using equation 10 to form a part of matrix $\mathbf{J_m}$, and then continue with the next portion of rows. We will discuss this below in section on the hybrid MPI and openMP implemetation.

Preconditioning CG solver

After the $\mathbf{J_m}$ is computed, the system of normal equations 3 can be solved with CG solver. We do not compute or store the system matrix of eq. 3, instead the action of $\mathbf{J_m}$ and $\mathbf{J_m}^H$ on a vector are implemented. To speed up the computations we keep both matrices $\mathbf{J_m}$ and $\mathbf{J_m}^H$ in memory.

It is well known that the number of CG iterations could be greatly reduced by finding a good preconditioner. Normally in 3D CSEM inversion the diagonal Jacobi preconditioner is used.

We propose a preconditioner based on a limited-memory quasi-Newton approximation to the inverse of the Hessian matrix. At each GN iteration k, we use the L-BFGS two loop recursion algorithm from Nocedal & Wright (1999), to construct the preconditioner. In the two loop recursion, we choose the diagonal matrix $\mathbf{H}_k^0$ to contain the inverse of the diagonal of system matrix on its diagonal. The computation of the diagonal of the system matrix is not very expensive and is computed anyway, to decide on the damping factor α (see eq. 3).

To make sure that this preconditioner is successful we propose a damped version of L-BFGS, in which vectors $\mathbf{s}_k = \mathbf{m}_{k+1} - \mathbf{m}_k$ are modified as follows:

$$\tilde{\mathbf{s}}_k = \theta_k\mathbf{s}_k + (1-\theta_k)\mathbf{H}_k\mathbf{y}_k, \tag{11}$$

here the scalar θ_k is defined as

$$\theta_k = \begin{cases} 1, & \text{if } \tau_k \geq 0.2 \\ 0.8/\left(1-\tau_k\right), & \text{otherwise} \end{cases}, \tag{12}$$

with $\tau_k = \frac{\mathbf{s}_k^T \mathbf{y}_k}{\mathbf{y}_k^T \mathbf{H}_k \mathbf{y}_k}$, $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$, $\mathbf{g}_k = \left(\frac{\partial \varphi}{\partial \mathbf{m}}\right)_k$ and $\mathbf{H}_k$ is L-BFGS approximation to the inverse of the Hessian matrix.

Similar approach was proposed in Nocedal & Wright (1999), but there it was formulated for the Hessian approximation $\mathbf{B}_{k+1}$, while we are interested in the update to the inverse of the Hessian $\mathbf{H}_{k+1}$.

The change to vectors $\mathbf{s}_k$ ensures positive definiteness of our preconditioner $\mathbf{H}_{k+1}$. Note that $\theta_k = 0$ gives $\mathbf{H}_{k+1} = \mathbf{H}_k$, and $\theta_k = 1$ gives the $\mathbf{H}_{k+1}$ matrix according to the unmodified L-BFGS method, so the values of θ_k between 0 and 1 gives a matrix that interpolates between these two cases. In Figure 1 we show the comparison of three 2.5D GN inversion runs in terms of the number of CG iterations per GN iteration. From the figure we see that solving the system without a preconditioner is very inefficient (black curve). The new preconditioner (red curve) for the most of the GN iterations performs at least 2 times better than the Jacobi preconditioner (blue curve). This approach was also tested on inversion of various 3D datasets and most of the time the number of CG iterations stays below 300.

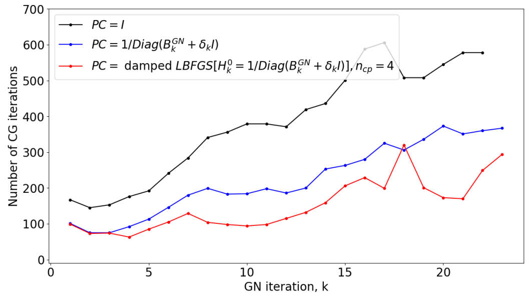


Figure 1: Effect of CG preconditioner on a number of CG iterations. Number of CG iterations is plotted for each GN iteration. Here Atlab3 dataset along central profile is inverted with 2.5D inversion.

Parallel implementation

To further speed up the computations we parallelize our solution. For the forward simulations the parallel implementation is straightforward, the responses from various sources are computed simalteneously on different cores. The parallel implemetation for Jacobian computation and CG solution to the system of normal equations is a bit more complicated.

The Jacobian is distributed by rows to various compute nodes. Depending on the available memory only a small number of rows of $\mathbf{J}_{\boldsymbol{\sigma}}$ is computed on the nodes at the same time. Then all three parameter transformations are performed, which fills some portion of rows of a distributed matrix $\mathbf{J}_{\mathbf{m}}$. This process is repeated until the whole matrix $\mathbf{J}_{\mathbf{m}}$ is filled. We use the PETSc library (Balay et al., 2023) to distribute and manipulate the matrix $\mathbf{J}_{\mathbf{m}}$. This library uses the message-passing model for parallel programming and employs MPI for all interprocessor communication.

The transformations of the Jacobian computed on each node are also parallelized by the use of openMP. The first two transformations of the Jacobian are straightforward. For the last one, every node has to have access to matrix $\mathbf{B_J} = \text{diag}^{-1}(\mathbf{A})\mathbf{B}^T$ (see eq. 10). To reduce the memory requirements and also to speed up the sparse-dense matrix multiplications $\mathbf{J}_{\tilde{\boldsymbol{\chi}}}\mathbf{B_J}$, in the matrix $\mathbf{B}$ we set elements B_{ij} to zero when $B_{ij} < \varepsilon * \max_{ij}(B_{ij})$. While for the parameter transformations 8 we use $\varepsilon = 10^{-5}$, for the Jacobian transformation 10 the coefficient $\varepsilon = 10^{-2}$. Multiple synthetic tests demonstrate that there is no quality degradation when using these values of ε.

We use openMP to speed-up all matrix-vector multiplications, required at each CG iteration. To solve the system of eq. 3 we utilize the CG solver provided within the PETSc library.

GN Inversion Applications

Application to Atlab-3 CSEM data

In this section we apply our inversion scheme to a dataset acquired at Mohns ridge by the ATLAB consortium. The consortium was formed by NTNU in 2016 with the aim to utilize a wide variety of geophysical data, in order to investigate the nature, dynamics, diversities and resources at mid-ocean ridges and oceanic plates. Since 2016 data was collected at the Mohns and Knipovich ridges, including CSEM, MT, seismic, chemical and biological data.

Johansen et al. (2019) presented the first CSEM inversion results from the Mohns ridge. They invert Atlab-1 data (green dots on Figure 2). Mittet & Avdeeva (2023) improved resolution and expanded the 2.5D CSEM inversion results, by inverting both Atlab-1 and Atlab-3 (blue and red dots on Figure 2) datasets. In this abstract we invert Atlab-3(2) data acquired in 2022 at the Mohns ridge (red dots). The data are measured along three profiles perpendicular to the Atlab-1 line, which facilitates 3D interpretation.

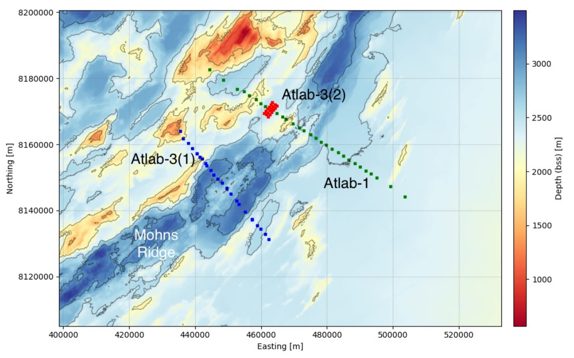


Figure 2: Locations of Atlab-1 and Atlab-3 datasets.

Each profile consists of 80 transmitters and 6 receivers separated by approximately 100 m and 800 m, respectively. The data at 1.6, 4.8, 8 and 11.2 Hz are inverted. The water depth in the area is approximately 2450 m and the bathymetry is relatively flat. The information on mesh sizes and memory requirements for the transformation and Jacobian matrices are shown in the first and third rows of Tables 1 and 2 for 2.5D and 3D inversions, respectively. The minimum total memory required for the inversion runs is 0.8 GB and 1.4 TB for 2.5D and 3D inversions, respectively.

Figure 3 compares vertical resistivity images obtained by the 3D and 2.5D inversions along the central Atlab-3(2) profile. The results are very similar and fit the data to an RMS of 1.1. Both images compare well with the overlayed seismic cross-section. The resistive layer obtained by 3D inversion has slightly more structure and the change in depth to the top of this layer below the 4th receiver is more obvious.

Conclusions

We propose a new algorithm for 3D VTI-anisotropic CSEM inversion, based on the Gauss-Newton optimization. To reduce the memory requirements of the Gauss-Newton method we suggest to decouple the simulation and optimization domains, using node-based basis functions. We show that, in the 3D case, the memory required by the Jacobian can be reduced from hundreds to tens of TB. To speed up the CG solution of the GN system of normal equations the new preconditioner is suggested. This preconditioner is at least two times more efficient than the commonly used Jacobi preconditioner. With this new GN scheme a large datasets can be inverted in 3D with modest computer resources. The algorithm is succesfully validated on 2.5D and 3D marine CSEM datasets acquired at Mohns ridge.

Acknowledgments

We thank the ATLAB consortium for allowing us to present the results of inversion of Atlab-3 dataset. ATLAB consortium has received funding from Norwegian governmental institutions and industry. The reprocessed seismic data was provided by ShearWater.

REFERENCES

Amaya, M., Morten, J. P., & Boman, L. (2016). A low-rank approximation for large-scale 3D controlled-source electromagnetic Gauss-Newton inversion. *Geophysics*, *81*(3), E211–E225.

Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., et al. (2023). *PETSc/TAO users manual* (Tech. Rep. No. ANL-21/39 - Revision 3.20). Argonne National Laboratory.

Grayver, A. V., Streich, R., & Ritter, O. (2013). Three-dimensional parallel distributed inversion of CSEM data using a direct forward solver. *Geophysical Journal International*, *193*, 1432–1446.

Guennebaud, G., Jacob, B., et al. (2010). *Eigen v3.* http://eigen.tuxfamily.org.

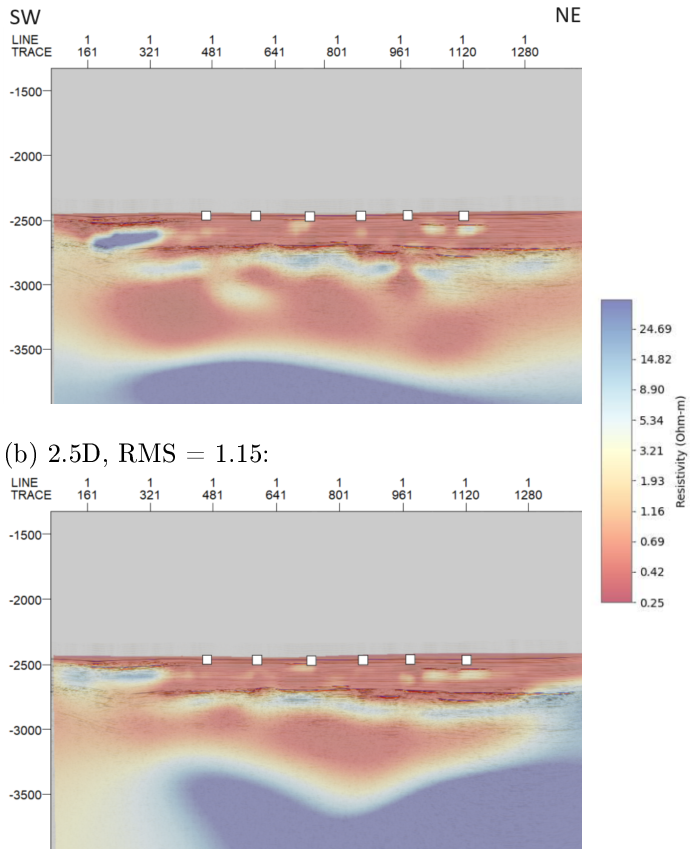
Johansen, S. E., Panzner, M., Mittet, R., Amundsen, H. E. F., Lim, E. V., Landrø, M., et al. (2019). Deep electrical imaging of the ultraslow-spreading Mohns Ridge. *Nature*, *567*, 379–383.

Li, M., Abubakar, A., Liu, J., Pan, G., & Habashy, T. M. (2011). A compressed implicit Jacobian scheme for 3D electromagnetic data inversion. *Geophysics*, *76*(3), F173–F183.

Mittet, R., & Avdeeva, A. (2023). Gauss-Newton inversion with node-based basis functions: Application to imaging of seabed minerals in an area with rough bathymetry. *Geophysics*, accepted.

Nguyen, A. K., Nordskag, J. I., Wiik, T., Bjørke, A. K., Boman, L., Pedersen, O. M., et al. (2016). Comparing large-scale 3D Gauss-Newton and BFGS CSEM inversions. In *Seg expanded abstracts 2016* (pp. 872–877). SEG.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization.* New York: Springer-Verlag.



(a) 3D, RMS = 1.12:

(b) 2.5D, RMS = 1.15:

Figure 3: Comparison of 3D (top) and 2.5D (bottom) inversion results. The vertical resistivity images along central Atlab-3(2) profile are shown. The seismic cross-section is plotted on top of the resistivity images for comparison.

problem size	$n_x \times n_y \times n_z$	$n_x^o \times n_y^o \times n_z^o$	$\mathbf{A}$ (GB)	$\mathbf{B}$ (GB)	$\mathbf{C}$ (GB)	$\mathbf{B_J}$ (GB)
2.5D, Atlab-3(2)	$415 \times 1 \times 167$	$157 \times 1 \times 43$	0.75	0.32	0.77	0.003
3D, small	$94 \times 169 \times 77$	$26 \times 53 \times 23$	5.2	6.5	0.007	0.06
3D, medium, Atlab-3(2)	$228 \times 214 \times 167$	$104 \times 97 \times 66$	40.6	45.7	0.15	0.57
3D, large	$622 \times 514 \times 101$	$199 \times 161 \times 18$	313	233	0.13	1.1

Table 1: Examples of memory requirements of the transformation matrices.

problem size	N_s	N_r	N_{tr}	N_{fr}	$n_x \times n_y \times n_z$	$\mathbf{J}_{\boldsymbol{\sigma}}$ (GB)	$n_x^o \times n_y^o \times n_z^o$	$\mathbf{J_m}$ (GB)
2.5D, Atlab-3(2)	80	6	480	4	$415 \times 1 \times 167$	3.98	$157 \times 1 \times 43$	0.39
3D, small	57	12	467	3	$94 \times 169 \times 77$	102	$26 \times 53 \times 23$	2.6
3D, medium, Atlab-3(2)	240	18	4320	4	$228 \times 214 \times 167$	8392	$104 \times 97 \times 66$	685
3D, large	700	139	54813	5	$622 \times 514 \times 101$	527483	$199 \times 161 \times 18$	9421

Table 2: Examples of reductions of memory required by the Jacobian matrix. Here we assume that we invert fields $(\mathbf{E}_x, \mathbf{E}_y, \mathbf{H}_x, \mathbf{H}_y)$ in 3D case and $(\mathbf{E}_x, \mathbf{H}_y)$ in 2.5D case. N_s, N_r, N_{tr}, N_{fr} denote the number of sources, receivers, active source-receiver pairs and frequencies, respectively.