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

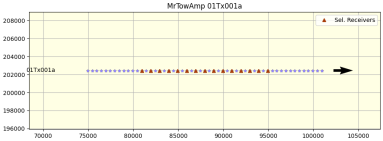
Anna Avdeeva, Rune Mittet, Ole Martin Pedersen

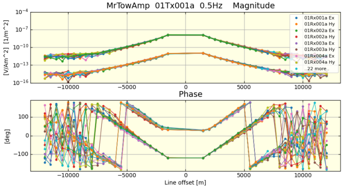
November 14, 2023

1. Introduction to GN CSEM inversion
2. Resolving GN inversion bottlenecks
   - Parameter Transformations
   - Memory requirements
   - Solving system of Normal Equations
3. Verification on Real Data
   - Atlab-1
   - Atlab-3(2)

Schematics of CSEM inversion

data: electric and magnetic fields

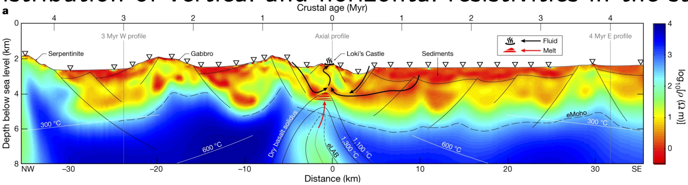




from Allton's software, CSEM_QC

↓ inversion

model: distribution of vertical and horizontal resistivities in the subsurface



From Johansen et al. (2019)

allton

Optimization problem

Objective function

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

Data misfit

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

System of normal equations

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

- $\mathbf{m}$ vector of model parameters
- $\mathbf{p}$ search direction, model update $\Delta\mathbf{m}$ is found with line search along $\mathbf{p}$
- $\mathbf{W}_d$ Data weights
- $\mathbf{J}_\mathbf{m}$ Jacobian matrix

Jacobian matrix memory requirements examples

$\mathbf{J_m} \in \mathbb{C}_{N_d \times N_m}$, $N_d = N_{tr} \times N_{fr} \times N_{comp}$, $N_m = n_x \times n_y \times n_z \times 2$

Consider $\mathbf{m}$ is conductivity $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_v)$

2.5D $N_s = 231, N_r = 30, N_{tr} = 4245, N_{fr} = 4, N_{comp} = 2,$
$N_m/2 = n_x \times n_z = 1076 \times 176$
$\rightarrow$ Jacobian memory 100 GB

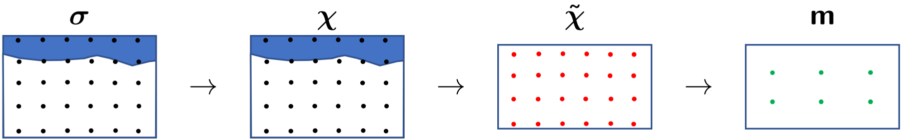
3D $N_s = 700, N_r = 139, N_{tr} = 54813, N_{fr} = 5, N_{comp} = 4,$
$N_m/2 = n_x \times n_y \times n_z = 622 \times 514 \times 101$
$\rightarrow$ Jacobian memory 520 TB

Jacobian can be very large $\rightarrow$
Smart solutions and big computational resources are required

Ideas for practical solutions in 3D case

- Reduce number of optimization parameters (separate simulation and optimization grids);
- Split Jacobian among various machines, using MPI;
- Substitute direct solvers (for example, MUMPS) with iterative solvers (such as CG, for example).

Inversion Parametrization



- $\boldsymbol{\sigma} = (\boldsymbol{\sigma}_h, \boldsymbol{\sigma}_v)$: conductivity at nodes of the simulation grid $(n_x \times n_y \times n_z \times 2)$
- $\boldsymbol{\chi}$: parameter with lower and upper bounds on the conductivity, no reduction in number of parameters
- $\tilde{\boldsymbol{\chi}}$: parameter after transformation from arbitrary seabed to flat seabed $(n_x \times n_y \times m_z \times 2, m_z < n_z)$
- $\mathbf{m}$: parameter after transformation to optimization domain, significant reduction in number of parameters $\left(n_x^o \times n_y^o \times n_z^o \times 2\right)$

$\boldsymbol{\sigma} \to \boldsymbol{\chi}$

Standard transformation:

$$\chi(x) = \ln\left(\frac{\sigma(x) - \sigma_0(x)}{\sigma_\infty(x) - \sigma(x)}\right) \tag{4}$$

$$\sigma(x) = \sigma_0(x) + \frac{1}{1 + e^{-\chi(x)}}\left[\sigma_\infty(x) - \sigma_0(x)\right] \tag{5}$$

Jacobian transformation is also straightforward:

$$\mathbf{J}_{\boldsymbol{\chi}} = \mathbf{J}_{\boldsymbol{\sigma}} \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\chi}} \tag{6}$$

allton

$\chi \rightarrow \tilde{\chi}$: transformation to flat seabed

- coordinate transform

$$\mathbf{x} = (x, y, z) = (\mathbf{h}, z) \rightarrow \mathbf{r} = (x, y, \xi) = (\mathbf{h}, \xi)$$

$$\xi(z) = \xi_{\max} \left[\frac{z - z_B(\mathbf{h})}{z_{\max} - z_B(\mathbf{h})} \right], z \geq z_B \tag{7}$$

$$z(\xi) = z_B(\mathbf{h}) + \frac{\xi}{\xi_{\max}} \left[z_{\max} - z_B(\mathbf{h}) \right] \tag{8}$$

z_B water depth
$z_{\max}$ maximum depth of the model
$\xi_{\max}$ distance from shallowest water depth to $z_{\max}$

- interpolation to regular grid
  We use spline interpolation both for parameter and Jacobian (only forward) transforms
- the original water volume is added after backward transform and interpolation

$\tilde{\chi} \rightarrow \mathbf{m}$: parameter representations

1D case:

$$\tilde{\chi}(x) = \sum_n \tilde{\chi}_n \phi_n(x), \tag{9}$$

$$\tilde{\chi}(x) = \sum_\nu m_\nu \psi_\nu(x). \tag{10}$$

$\tilde{\chi}_n$, $\phi_n(x)$ node values and basis functions centred at nodes x_n.

m_ν, $\psi_\nu(x)$ node values and basis functions centred at nodes x_ν.

Nodes x_ν are sampled 2 to 5 times coarser than nodes x_n

allton

$\tilde{\chi} \to \mathbf{m}$: transformation matrices

1D case:

$$\begin{aligned} A_{ln}^{(x)} &= \int_0^{x_{max}} \phi_n(x)\phi_l(x)dx, \\ B_{\gamma n}^{(x)} &= \int_0^{x_{max}} \psi_\gamma(x)\phi_n(x)dx, \\ C_{\nu\gamma}^{(x)} &= \int_0^{x_{max}} \psi_\nu(x)\psi_\gamma(x)dx. \end{aligned} \tag{11}$$

3D case:

$$C_{\Gamma\Lambda} = C_{\nu\gamma}^{(x)} C_{\alpha\beta}^{(y)} C_{\lambda\kappa}^{(z)} (\text{outer product}). \tag{12}$$

Similarly matrices **A**, **B**.

Transformation matrices are independent of parameters **m** and χ and can be computed at the beginning of the inversion process.

$\tilde{\chi} \rightarrow \mathbf{m}$: transformations

Parameter transformations:

$$\mathbf{A}\tilde{\chi} = \mathbf{B}^T\mathbf{m}, \tag{13a}$$

$$\mathbf{C}\mathbf{m} = \mathbf{B}\tilde{\chi}. \tag{13b}$$

Jacobian transformation (only forward):

$$\mathbf{J}_{\mathbf{m}} = \mathbf{J}_{\tilde{\chi}}\mathbf{A}^{-1}\mathbf{B}^T \rightarrow \mathbf{J}_{\mathbf{m}} \approx \mathbf{J}_{\tilde{\chi}}\mathrm{diag}^{-1}\left(\mathbf{A}\right)\mathbf{B}^T \tag{14}$$

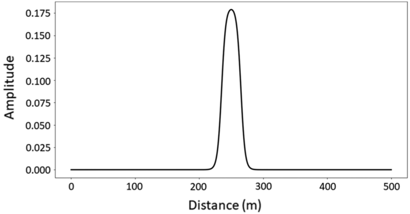
Our choice of basis functions ϕ and ψ

sinc and an adjustable boxcar functions for ϕ and ψ, respectively.

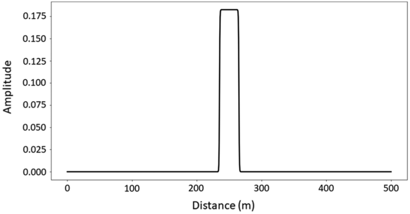
$$\phi_n(x) = \frac{\sin\left[(x - x_n)\,\pi/\Delta x\right]}{(x - x_n)\,\pi/\Delta x} \tag{15a}$$

$$\psi_\gamma(x, x_T) = \frac{\sinh\left[\Delta x_B/(2kx_T)\right]}{\cosh\left[\Delta x_B/(2kx_T)\right] + \cosh\left[(x - x_\gamma)/(kx_T)\right]} \tag{15b}$$

$x_T = x_B$

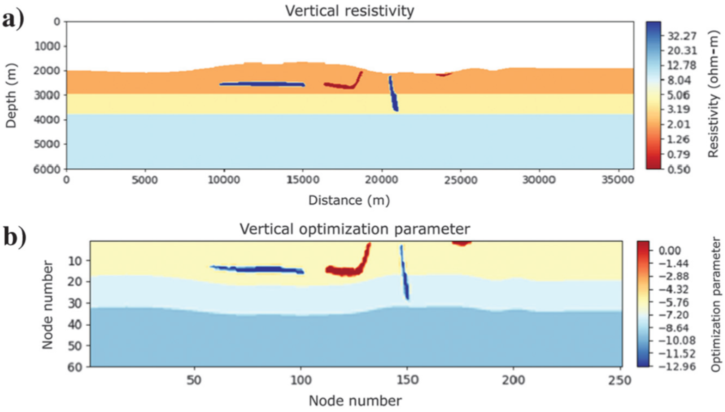


$x_T = 0.1x_B$



$k = 0.1088$, $x_B = 30$ m

$\rho_v \rightarrow m_v$: transformed model example



From Mittet & Avdeeva (2023)

Memory required for transformation matrices

model size	$n_x \times n_y \times n_z$	$n_x^o \times n_y^o \times n_z^o$
2.5D, Atlab-3(2)	$415 \times 1 \times 167$	$157 \times 1 \times 43$
3D, small	$94 \times 169 \times 77$	$26 \times 53 \times 23$
3D, medium, Atlab-3(2)	$228 \times 214 \times 167$	$104 \times 97 \times 66$
3D, large	$622 \times 514 \times 101$	$199 \times 161 \times 18$

model size	**A** (GB)	**B** (GB)	**C** (GB)	$\mathbf{B_J}$ (GB)
2.5D, Atlab-3(2)	0.75	0.32	0.77	0.003
3D, small	5.2	6.5	0.007	0.06
3D, medium, Atlab-3(2)	40.6	45.7	0.15	0.57
3D, large	313	233	0.13	1.1

Table: Examples of memory requirements of the transformation matrices.

Memory required by Jacobian

data size	N_s	N_r	N_{tr}	N_{fr}
2.5D, Atlab-3(2)	80	6	480	4
3D, small	57	12	467	3
3D, medium, Atlab-3(2)	240	18	4320	4
3D, large	700	139	54813	5

problem size	$\mathbf{J}_{\boldsymbol{\sigma}}$ (GB)	$\mathbf{J}_{\mathbf{m}}$ (GB)
2.5D, Atlab-3(2)	3.98	0.39
3D, small	102	2.6
3D, medium, Atlab-3(2)	8392	685
3D, large	527483	9421

Table: Examples of reductions of memory required by the Jacobian matrix. Here we assume that we invert fields $(\mathbf{E}_x, \mathbf{E}_y, \mathbf{H}_x, \mathbf{H}_y)$ and $(\mathbf{E}_x, \mathbf{H}_y)$ in 3D and in 2.5D cases, respectively.

Systems to solve

- At the beginning of the inversion:

$$\mathbf{A}\tilde{\chi} = \mathbf{B}^T\mathbf{m}.$$

- At each GN iteration, transform model to simulation mesh:

$$\mathbf{C}\mathbf{m} = \mathbf{B}\tilde{\chi}.$$

- At each GN iteration, solve system of normal equations:

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

allton

Solvers

- 2.5D: the transformation matrices and their factorizations could be stored in memory. Direct solvers could be used for transformations, and also for the solution of the system of normal equations.
- 3D: All direct solvers must be substituted with iterative solvers. We use preconditioned CG solvers from Eigen (Guennebaud et al., 2010) and PETSc (Balay et al., 2023) C++ libraries.
- For transformation of parameters diagonal Jacobi preconditioner is used.
- For solution of the system of normal equations our preconditioner is based on L-BFGS approximation to the inverse of Hessian matrix.

With CG solver the system matrix or the preconditioner is not computed or stored, rather their multiplication with a vector is implemented.

L-BFGS based preconditioner

- Preconditioner **P** is L-BFGS approximation to the inverse of Hessian matrix (Nocedal & Wright (1999))
- For application of **P** to a vector we need to know models and gradients for previous n_{cp} iterations
- To make sure that **P** is successful we propose a damped version of L-BFGS, so $\mathbf{s}_k = \mathbf{m}_{k+1} - \mathbf{m}_k$ are modified as follows:

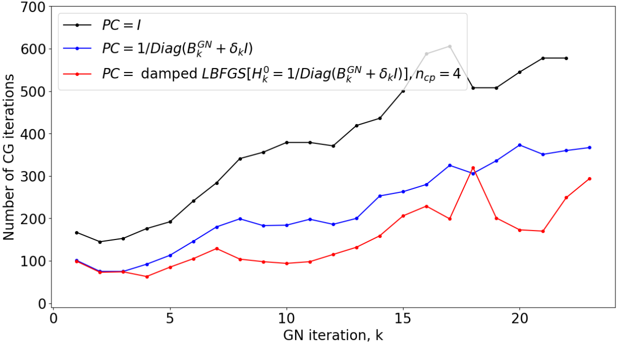
$$\tilde{\mathbf{s}}_k = \theta_k \mathbf{s}_k + (1 - \theta_k)\mathbf{P}_k \mathbf{y}_k, \tag{16}$$

here the scalar θ_k is defined as

$$\theta_k = \begin{cases} 1, & \text{if } \tau_k \geq 0.2 \\ 0.8/\left(1 - \tau_k\right), & \text{otherwise} \end{cases}, \tag{17}$$

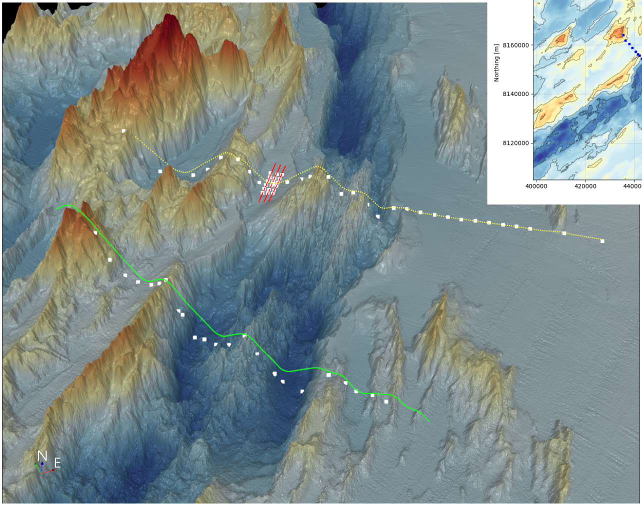
with $\tau_k = \frac{\mathbf{s}_k^T \mathbf{y}_k}{\mathbf{y}_k^T \mathbf{H}_k \mathbf{y}_k}$, $\mathbf{y}_k = \mathbf{g}_{k+1} - \mathbf{g}_k$, $\mathbf{g}_k = \left(\frac{\partial \varphi}{\partial \mathbf{m}}\right)_k$.

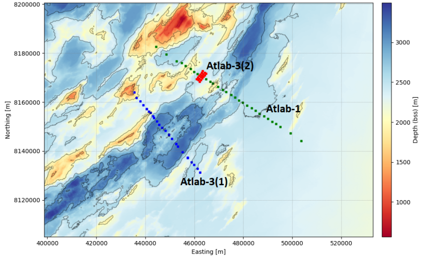
Effect of CG preconditioner



The proposed preconditioner gives significant reduction in number of CG iterations;
Most cases show that the number of CG iterations is below 200, even for relatively large 3D examples.

Atlab data

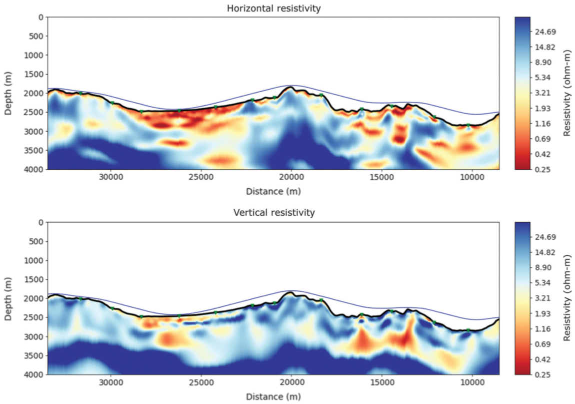




ATLAB Consortium:

The purpose of the consortium is to obtain a better overall understanding of slow-spreading ridges;

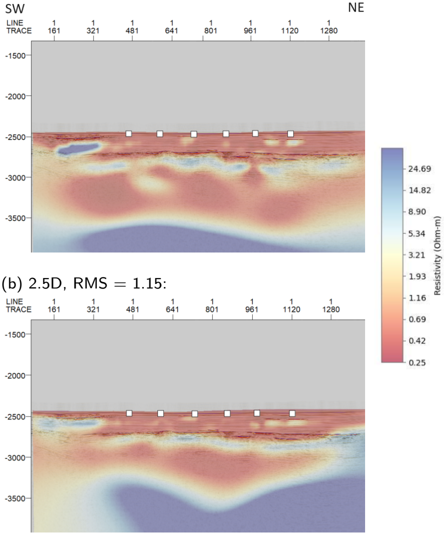
Atlab-1



Acquired in 2017 and inverted in 2019 by Johansen et al. (2019); Re-inverted in early 2022 by Mittet & Avdeeva (2023).

- 14 Rx: separation 2 km
- 89 Tx: current 1200 A, length 300 m, height 100-400 m
- Maximum offset - 6 km
- Frequencies: 2, 3.5, 4.5, 9, 17 Hz
- simulation mesh: 841 $\times$ 281 nodes, 50 $\times$ 25 m^2
- optimization mesh: 510 $\times$ 124 nodes, 70 $\times$ 35 m^2,
- **rms**: 24.1 $\rightarrow$ 1.04

(a) 3D, RMS = 1.12:



(b) 2.5D, RMS = 1.15:

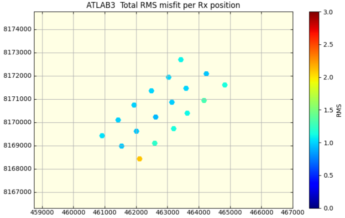
Atlab-3 data was acquired in 2022, triggered by the high-resolution image of Atlab-1.

- 3 × 9 Rx: separation 800 km
- 240 Tx: current 1000 A, length 50 m, height 30-40 m
- Maximum offset - 6 km
- Frequencies: 1.6, 4.8, 8.0, 11.2 Hz
- simulation mesh: 228 × 214 × 167 nodes, 60 × 60 × 30 m^3
- optimization mesh: 104 × 97 × 66 nodes, 121 × 121 × 43 m^3
- **rms**: 9 → 1.12

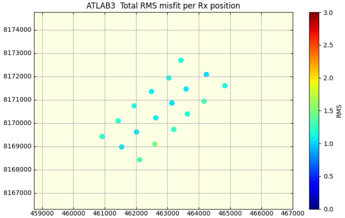
allton

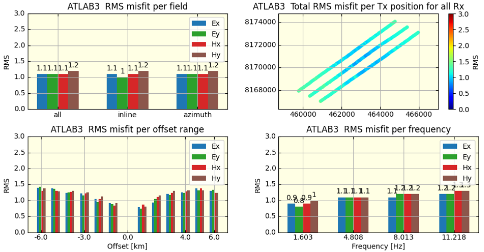
Atlab-3(2) data fits

Initial run



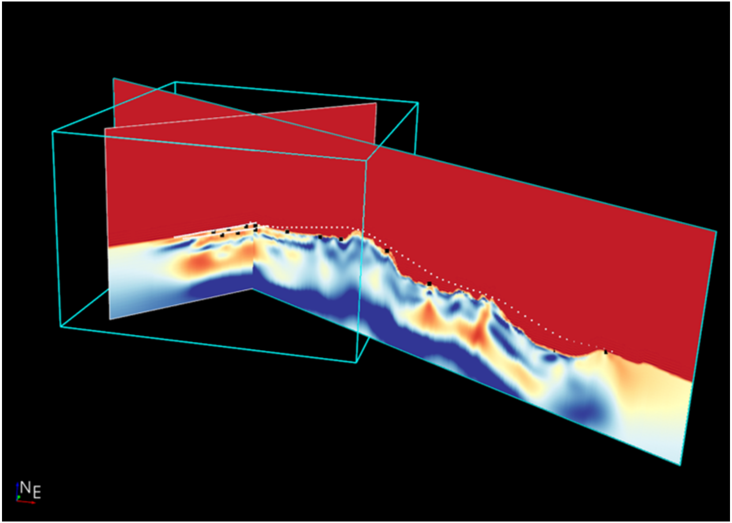
Final run, corrected Rx018





from Allton's software, CSEM_QC

Atlab-1 & Atlab-3(2) inversion results



from Allton's software, Model builder

Conclusions

- We propose a new algorithm for 3D VTI-anisotropic CSEM inversion based on Gauss-Newton optimization method.
- To reduce the memory requirements the simulation and optimization meshes are decoupled, using node-based basis functions. In the 3D case, the memory requirements can be reduced from hundreds to tens of TB.
- With the use of L-BFGS preconditioner we speed up the CG solutions of the system of normal equations.
- 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.

allton

Acknowledgements

- We thank the ATLAB consortium for allowing us to present the results of inversion of Atlab-1 and Atlab-3 datasets. ATLAB consortium has received funding from Norwegian governmental institutions and industry.
- The reprocessed seismic data was provided by ShearWater.

References

Balay, S., Abhyankar, S., Adams, M. F., Benson, S., Brown, J., Brune, P., ... Zhang, J. (2023). *PETSc/TAO users manual* (Tech. Rep. No. ANL-21/39 - Revision 3.20). Argonne National Laboratory. doi: 10.2172/1968587

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., & Arntsen, B. (2019). Deep electrical imaging of the ultraslow-spreading Mohns Ridge. *Nature*, *567*, 379–383.

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.

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