# Tomofast-x

Geophysical 3D parallel potential field joint inversion package (v.1.3):

**User Manual**

by Vitaliy Ogarko

*Centre for Exploration Targeting, The University of Western Australia, 2022.*

# 1 Contents

# 2 Introduction

Tomofast-x package solves constrained nonlinear inversion problems of gravity and magnetism for rock density and magnetic susceptibility models [1-6]. Tomofast-x is also capable of performing the Electrical Capacitance Tomography, for details see Ref. [7]. In this manual we mainly focus on gravity and magnetic inversions. To improve the final model by adding geological and petrophysical information into inversion, the code supports several different constraints. Such constraints include cross-gradient constraints, “smart”-gradient, disjoint interval bound constraints, and clustering constraints. For more information on equations being solved and the type of constraints used, see reference [1], and references therein.

The model grid allows to accommodate an arbitrary model shape to be able to simulate surface topography, and continent-ocean boundaries.

For visualization of the output 3D models, the code generates the output results also in the VTK file format (binary version, for smaller size), which can be visualized by a free open-source software Paraview.

The code is written in Fortran 2008, and utilizes classes and modern vectorization compiler features for speed-up. It is fully parallelized using the MPI library. Note that the MPI library is the only external dependency, which is making this code easy to build on different machines, and to extend or extract its parts.

In the parallel mode, all code parts are run in parallel, and the data handling is done via RAM memory (i.e., not via files). The parallel mode allows faster performance and solving much bigger problems due to memory limitations on single machine. The code can be executed in parallel on a supercomputer, with distributed memory system, using thousands of CPUs, or on modern computers/notebooks, with shared memory system, using several (2-40) CPUs.

Some critical parts of the code are covered by automated unit tests, including some parallel tests.

The code with some examples can be downloaded from the following GitHub repository:

https://github.com/TOMOFAST/Tomofast-x

Please, feel free to commit your pull requests with new features, bugfixes, optimizations, new examples, etc.

If you are using this code or some of its parts, please cite references [1] and [2].

# 3 Code compilation

The code compilation is based on Make, which is a build automation tool. It is assumed that the code is compiled in Linux environment, even though a Windows build is also possible (either in PowerShell, or directly in Windows with installed make). To compile the code you need:

- Compiler gcc v.4.9 or more recent, or Intel compiler.
- MPI library (such as OpenMPI).

The Makefile is contained in the root folder and should be used to compile Tomofast-x. Compiling the code is a necessary step to be able to run inversions.

To compile the code run the make command in the code directory as:

```
make
```

To clean the compilation files (to perform a clean compilation), run:

```
make clean
```

To run the code with your parameter file:

```
./tomofast3D -j <Parfile path>
```

For the parallel run on your local machine, execute:

```
mpirun -np <number CPUs> -j <Parfile path>
```

To run unit tests in serial:

```
./runtests.sh
```

and in parallel:

```
mpirun -np 3 ./runtests.sh
```

Note, that the Makefile by default assumes the gcc compiler. It can be switched to the Intel compiler by setting "`FC = mpiifort`", and switching the `FLAGS` variable definition to the Intel one (commented out by default).

For full debugging information, in the Makefile different sets of `FLAGS` are provided. The flags to output the vector optimization report, `OPT_INFO`, can also be enabled.

# 4 Description of code parameters

To run the code, one has to specify various configuration parameters, that include paths to input data files, free parameters of the cost function and constraints, configuration of the solver, and others.

All input parameters are defined in one text file, which we call the Parfile. The Parfile is split into several categories separating different parameter types. For a Parfile that provides a list of all available input parameters with their default values and description, see the file *Parameters_all.txt* located in the code root directory.

A description of each parameter category, with its name prefix specified in brackets, is given below.

## 4.1 Global (`global`)

Contains the path to the output data folder, and a flag that determines which problem to solve (gravity, magnetism, or both).

## 4.2 Model grid (`modelGrid`)

Contains the total number of grid cells, and paths to the model grid files.

Note that if a “true” model exists, one can specify its values in the same grid files, then the forward data generated from that model can be used directly, by specifying in the DATA section:

```
forward.data.grav.dataValuesFile = ./output/grav_calc_read_data.txt
```

```
forward.data.magn.dataValuesFile = ./output/mag_calc_read_data.txt
```

This option is added to facilitate testing the code with different synthetic models. When there is no “true” model known, the model values in the model grid files can be set to zeros.

## 4.3 Data (`forward.data`)

Contains the number of data, and paths to data grids, and the observed (measured) data to be inverted. Note, that the data grid and data values files have the same file format and can point to the same files.

## 4.4 Magnetic field (`forward.magneticField`)

Constants describing the external (Earth) field, required to calculate the forward magnetic problem (inclination/declination angles, field intensity, etc).

## 4.5 Depth weighting (`forward.depthWeighting`)

Contains parameters to configure the depth weighting used. The `type` parameter specifies the type of depth weighting. The type 1, is based on the inverse power law of the distance from the surface to the j[th] grid cell, defined as:

$$W(j) = \frac{1}{(Z_j - Z_0)^{q/2}}$$

The corresponding free parameters, power $q$ and a shift $Z_0$ can be specified in parameters `power` and `Z0`, for gravity and magnetic problems separately. Note that as the magnetic field attenuates with distance faster than gravitational one, the power $q$ for magnetic problem is usually higher. The exact values of the powers should be configured for a model of interest, as they depend on the model grid (cells) dimensions, and data locations.

The depth weighting type 2 corresponds to the depth weighting based on the distance to data (i.e., it varies in all model dimensions). It is a preferable option for models with non-flat topography, and is defined as:

$$W(j) = \frac{1}{\sqrt{\Delta V_j}} \left\{ \sum_{i=1}^{N} \left[ \int_{\Delta V_j} \frac{dv}{(R_{ij} + R_0)^q} \right]^2 \right\}^{\frac{1}{4}}, j = 1, \dots, M,$$

where $\Delta V_j$ is the volume of the j[th] cell, $R_{ij}$ is the distance between the cell subvolume dv and the i[th] data, and $R_0$ is a small constant for integral validity. For more details, see the UBC code manual.

## 4.6 Sensitivity kernel (`sensit`)

In this section one can set a flag to reuse the already calculated sensitivity kernel. The corresponding path to the sensitivity kernel is specified in parameter `folderPath`. This is useful to save the calculation time when the sensitivity kernel does not need to be recalculated. For example, when one changes the constraints parameters and/or types, or when a different background field is used for data subtraction.

## 4.7 Matrix compression (`forward.matrixCompression`)

In this section one can configure sensitivity matrix compression based on the wavelet compression. This is very helpful for running large models on machines with limited memory and speeds up the calculation significantly. The level of compression can be adjusted by varying the compression rate parameter `rate`, where unity corresponds to the full matrix (i.e., no compression).

Note, that another way to solve memory limitations, is to run the code in parallel on a supercomputer, using more CPUs. The information on the compression rate, and compression error (cost) is printed in the log for the reference.

## 4.8 Prior model (`inversion.priorModel`)

The prior model is the one used for adding model damping constraints to the cost function, see Ref. [1]. Parameter `type` defines the type of prior model initialisation. If type 1 is chosen, then all values of the prior model will be initialised from a value specified in parameter `value`. When type 2 is chosen, the prior model will be read from a file, which path is specified in `file` parameter.

## 4.9 Starting model (`inversion.startingModel`)

The starting model is the one that is used as an initial model in the iterative inversion process. The input parameters are same as for the prior model (see above). Note, it can be a good choice to keep the staring model equal to zero, not to propagate any "bad" structures into inversion results, which can be hard to remove due to the "null-space" issues.

## 4.10 Inversion (`inversion`)

Contains the number of nonlinear inversion iterations (major loop), `nMajorIterations`, and two stopping criteria for the inversion solver: (1) based on the number of solver iterations, (minor loop), `nMinorIterations`, and (2) based on the smallest relative residual, `minResidual`. It is recommended to keep the latter two parameters equal to 100 and 1e-13, respectively. While the number of major iterations can depend on the type of constraints added to the inversion.

## 4.11 Model damping (`inversion.modelDamping`)

Contains a damping weight $\alpha$ and the $L_p$ norm power $p$ for the model damping term, see Refs. [1, 7]. Note different values of $\alpha$ for gravity and magnetic problems, due to different scale of physical units. When $\alpha = 0$ the model damping is not active, even though the depth weighting is always active, via the sensitivity matrix preconditioning by $W^{-1}$. It is recommended to always have the non-zero value of $\alpha$ for numerical stability.

## 4.12 Joint inversion (`inversion.joint`)

This section contains weights needed to set-up the balance between the gravity and magnetic inversions in joint inversion (for example, when using cross-gradient, or clustering constraints). The problem weight (`problemWeight`) is applied to respective data misfit and model damping terms. The column weight (`columnWeightMultiplier`) is applied to the columns of the least-squares matrix, essentially performing model mapping to the scaled variables.

## 4.13 Disjoint interval bound constraints (`inversion.admm`)

Contains parameters that define the disjoint interval bound constraints (ADMM). To enable the constraints, set `enableADMM` to unity. The local bounds for every cell and lithology are defined in the bound files (`boundsFile`). For the description of the bounds file format, see section 5.4. The bound constraints terms are added to the misfit function with a corresponding global weight (`weight`). Note that this weight should be adjusted to the problem at hand to achieve best results.

When the bound constraints are enabled, the number of inversions should be greater than ~20, because the problem becomes nonlinear (with respect to mapping used inside the constraints, which is affecting the right-hand side of the least-squares system, but not the least-squares matrix). For more details on disjoint interval bound constraints, see Ref. [2].

## 4.14 Damping-gradient constraints (`inversion.dampingGradient`)

Contains parameters for setting the damping-gradient constraints. This type of constraints can either be a commonly used model gradient term (`weightType=1`), or a "smart"-gradient (`weightType=2`), where the model gradient term is preconditioned with local weights. The local weights are based on the second input model ("smart"-gradient constraints assume single inversion). For more details on "smart"-gradient constraints, see Refs. [3-4].

## 4.15 Cross-gradient constraints (`inversion.crossGradient`)

Contains parameters to set-up the cross-gradient constraints for performing joint inversion. The constraints are enabled when parameter `weight` is greater than zero. For more details on cross-gradient constraints, see Ref. [5].

## 4.16 Clustering constraints (`inversion.clustering`)

Contains parameter to set-up the clustering constraints, which can be used in both single and joint inversions. The constraints are enabled when the respective parameter `weight` is greater than zero. For details on the file formats of the mixture (`mixtureFile`) and cell weights file (`cellWeightsFile`), see section 5.5. For more details on clustering constraints, see Ref. [6].

# 5 Description of input data formats

Tomofast-x requires several input data files to specify:

- Observed data values and positions (data grid),
- Model grid (cells),
- Prior and starting models (optional), and
- Different constraints (optional).

All input files have an ASCII (text) format, and the 3D coordinates are given in a Cartesian coordinate system. It should be noted that the Z-axis points downwards, i.e., all Z-values below the ground are positive, and Z-values above the ground are negative. All the physical units used in the code are SI-units, unless explicitly specified otherwise.

Examples of input data files can be found in *data/gravmag* folder. A detailed description for different type of input data is given below.

## 5.1 Data file format

Data grid and data values files have the same format, for simplicity. The paths to data files are specified in the Parfile sections `forward.data.grav` and `forward.data.magn`, for gravity and magnetic problems, respectively. For more details see the file *Parameters_all.txt*.

The first line contains the number of data values, and the following lines contain 3D data positions (x, y, z), and then the data value, all separated by a space, as follows:

$N$

$p_x^1\ p_y^1\ p_z^1\ d^1$

…

$p_x^i\ p_y^i\ p_z^i\ d^i$

…

$p_x^N\ p_y^N\ p_z^N\ d^N$

Here $N$ is the number of data values, $p_x^i\ p_y^i\ p_z^i$ is the 3D position of the *i*-th data, and $d^i$ is the value of the *i*-th data. The data value is used in the data values files:

`forward.data.grav.dataValuesFile`,

`forward.data.magn.dataValuesFile`,

and it is ignored for the data grid files:

`forward.data.grav.dataGridFile`,

`forward.data.magn.dataGridFile`.

Separated paths for data grid and values are provided in order to be able to observe the forward data generated from the input model (model values stored in the model grid), in the same inversion run, for tests with synthetic data.

## 5.2 Model grid file format

Model grid and model values files have the same format, for simplicity. The paths to model files are specified in the Parfile sections `modelGrid.grav` and `modelGrid.magn`, for gravity and magnetic problems, respectively. For more details see the file *Parameters_all.txt*.

The model grid consists of a set of non-overlapping rectangular prisms. This format allows to specify models with different shapes, to be able to simulate surface topography, and continent-ocean boundaries.

The first line contains the number of model cells, and every following line contains cell coordinates, model value, 3D cell index, and model covariance, separated by a space, as follows:

$N$

$X_{min}^1\ X_{max}^1\ Y_{min}^1\ Y_{max}^1\ Z_{min}^1\ Z_{max}^1\ m^1\ i^1\ j^1\ k^1\ c^1$

…

$X_{min}^i\ X_{max}^i\ Y_{min}^i\ Y_{max}^i\ Z_{min}^i\ Z_{max}^i\ m^i\ i^i\ j^i\ k^i\ c^i$

…

$X_{min}^N\ X_{max}^N\ Y_{min}^N\ Y_{max}^N\ Z_{min}^N\ Z_{max}^N\ m^N\ i^N\ j^N\ k^N\ c^N$

Where $N$ is the number of model cells, $X_{min}^i\ X_{max}^i\ Y_{min}^i\ Y_{max}^i\ Z_{min}^i\ Z_{max}^i$ are the min/max coordinates of the X, Y, Z planes of the cell (rectangular prism), $m^i$ is the model value at this cell (ignored for the model grid file), $i^i\ j^i\ k^i$ is the 3D cell-index (integer), and $c^i$ is the covariance (set to unity when not known). The 3D cell-index is needed for constraints that are using the gradient calculations (such as cross-gradient, and smart-gradient constraints), and to perform the matrix wavelet compression.

The surface topography can be specified effectively draping the mesh under the topography. This way we do not need to specify the air-cells, thus reducing the grid size and memory requirements, see Figure 1.

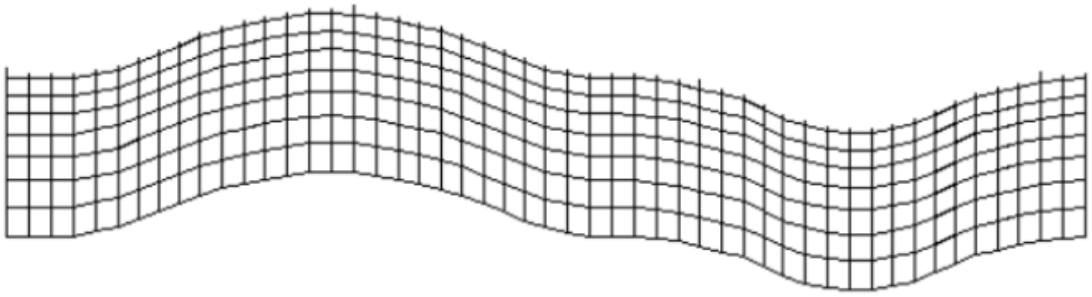

Figure 1. Example of model grid with topography. Note, that we effectively drape the mesh under the topography. Note the increasing cell height with depth too.

## 5.3 Prior and starting models

The prior and starting models can be defined from the input constant value (given in the Parfile), or from files. To define uniform models using a constant value, set the model type to 1, and define the corresponding model value (`grav.value` and/or `magn.value`). To define the models from files, one needs to set the model type to 2 in the Parfile as:

```
inversion.priorModel.type = 2
```

```
inversion.startingModel.type = 2
```

In this case, the model files should be given in the same format as described above, and paths to files should be specified in the corresponding Parfile sections.

## 5.4 Disjoint interval bound constraints (ADMM)

This type of constraints requires providing a bounds file for the gravity and/or magnetic problems. The paths to bounds files are specified in the Parfile parameters:

`inversion.admm.grav.boundsFile`,

`inversion.admm.grav.boundsFile`,

for the gravity and magnetic problems, respectively. The format of the bound constraints file is as follows:

$N\ L$

$m_{min}^{1,1}\ \dots\ m_{min}^{1,l}\ \dots\ m_{min}^{1,L}\ m_{max}^{1,1}\ \dots\ m_{max}^{1,l}\ \dots\ m_{max}^{1,L}\ w^1$

…

$m_{min}^{i,1}\ \dots\ m_{min}^{i,l}\ \dots\ m_{min}^{i,L}\ m_{max}^{i,1}\ \dots\ m_{max}^{i,l}\ \dots\ m_{max}^{i,L}\ w^i$

…

$$m_{min}^{N,1} \ \dots \ m_{min}^{N,l} \ \dots \ m_{min}^{N,L} \ m_{max}^{N,1} \dots \ m_{max}^{N,l} \ \dots \ m_{max}^{N,L} \ w^N$$

Here $N$ is the number of model cells, $L$ is the number of bounds (lithologies), and every following line contains min/max constraints for every lithology $l = 1, \dots L$, for the *i*-th model cell, with $i = 1, \dots N$, ending with the cell local weight $w^i$. The local weight determines the degree of uncertainty of the corresponding constraint among others. When all bound constraints are equally uncertain, all weights should be set to unity. Examples of such bound constraints files can be seen in "*data/gravmag/mansf_slice/*" folder.

## 5.5 Clustering constrains

Clustering constraints require an input mixture file describing the clusters (`mixtureFile`). The mixture file has the following format:

$$n$$

$$w_1 \ \mu_1^G \ \sigma_1^G \ \mu_1^M \ \sigma_1^M \ \sigma_1^{GM}$$

...

$$w_i \ \mu_i^G \ \sigma_i^G \ \mu_i^M \ \sigma_i^M \ \sigma_i^{GM}$$

...

$$w_n \ \mu_n^G \ \sigma_n^G \ \mu_n^M \ \sigma_n^M \ \sigma_n^{GM}$$

Here, $n$ is the number of clusters, and every following line defines, for the *i*-th cluster, the local weight $w_i$, the mean and standard deviation for gravity and magnetic problems, $\mu_i^G \ \sigma_i^G$ and $\mu_i^M \ \sigma_i^M$, and cross-term standard deviation $\sigma_i^{GM}$ (in the multivariate normal distribution).

When the local type of constraints is chosen (`constraintsType=2`), also the cell weights file (`cellWeightsFile`) needs to be provided, in the following format:

$$N \ n$$

$$w_1^1 \dots w_n^1$$

...

$$w_1^N \dots w_n^N$$

With $N$ and $n$ being the total number of cells, and the number of clusters, respectively. Every following line contains the local cluster weights for the *i*-th cell.

For more details on clustering constraints, see Ref. [6].

# 6 Description of the code output files

Tomofast-x produces a number of output files, stored in the output folder specified in the Parfile parameter `global.outputFolderPath`. There are several types of output files produced:

- Data files (txt).
- Model files (txt).
- Visualization files (vtk, csv).
- Analysis files (txt, vtk).
- Code logs (written on the screen).

Each type is described below in more detail. Note that we give file names for the gravity problem, while for the magnetic problems the files have the same name, only the "grav"-prefix is replaced with "mag".

Note, that most of the output can be disabled by setting in the Makefile the flag:

```
SUPPRESS_OUTPUT = YES
```

This option can be useful for performance tests.

## 6.1 Data files

The output data files have the same format as the input data files. They correspond to the gravity and magnetic problems, named with prefixes "grav" and "mag". The files for the gravity problem are:

- grav_calc_read_data.txt – data calculated from the "true" model (relevant for running synthetic model tests, when the true model is provided).
- grav_observed_data.txt – observed data (the same as input data).
- grav_calc_final_data.txt – data response of the final model (obtained from inversion of the observed data).

## 6.2 Model files

The output model files have the same (voxet) format as the input model files. The files are stored in the *Voxet* folder, which is created inside the output folder.

- grav_read_voxet_full.txt – the "true" model (relevant for running synthetic model tests, when the true model is provided).
- grav_prior_voxet_full.txt – the prior model.

- grav_starting_voxet_full.txt – the starting model.
- grav_final_voxet_full.txt – the final model obtained after inversion.

## 6.3 Visualization files

To be able to visualize the final model, the code also generates models in the (binary) vtk-files, which can be visualized by a free open-source software Paraview. For example, for the gravity problem the files produced are:

- grav_read_model3D_full.vtk – the "true" model (relevant for running synthetic model tests, when the true model is provided).
- grav_prior_model3D_full.vtk – the prior model.
- grav_starting_model3D_full.vtk – the starting model.
- grav_final_model3D_full.vtk – the final model obtained after inversion.

The code also produces the vtk-files with model slices (profiles) cutting the model half in x-, y-, in z-directions, whose filenames are marked with "half_x", "half_y", and "half_z", respectively.

Tomofast-x also produces several csv-files with the data, stored in the format suitable for visualization in Paraview. They contain the same data as their txt-counterparts, described in the subsection above, but formatted differently for visualization purposes. To visualize them in Paraview one needs to perform the following steps:

1) Open it in Paraview,
2) Apply "*Table To Points*" filter (*Filters > Alphabetical > Table To Points*),
3) Choose corresponding x, y, z columns in Properties menu (left bottom corner), and click "*Apply*", and
4) Finally choose "*Coloring"* by f-column (which stores the data values).

The point size can be increased in "Styling" menu, to improve the visibility.

## 6.4 Analysis files

The code also produces some additional files for various analysis (convergence, sensitivity, model variance, and others):

- costs.txt – contains the data and model costs, for gravity and magnetic problems, clustering constraints cost, and cross-gradient cost for every direction: x, y, z.
- lsqr_std_prior_grav.txt – contains the prior solution variance.
- lsqr_std_posterior_grav.txt – contains the posterior solution variance.

- Voxet/sensit_grav_voxet_full.txt – contains the integrated sensitivity matrix (in the format of model grid).
- Voxet/clustering_data.txt – contains model cell data followed by Gaussian mixture value, two derivatives (with respect to gravity and magnetic models), and the value of each mixture element (local Gaussian, one per cluster).

## 6.5 Screen log

Finally, during the code execution, a lot of information about current state of code convergence is produced and written to the screen log. This information can be used to better understand the influence of input parameters on the rate and quality of convergence. One of the important parameters to monitor during the inversion process is the relative data misfit, defined as:

$$cost = \frac{\|D_{calc} - D_{obs}\|_2^2}{\|D_{obs}\|_2^2}.$$

Note, that this cost, including other costs (see previous subsection) is also written to "*costs.txt*" file, to the second and third columns, for gravity and magnetic problems, respectively.

Other relevant information written in the log is the LSQR convergence details after every 10 iterations, and the final number of minor iterations, which can be helpful to analyse the stability of the inversion process. For example, when the gravity or magnetic inversion shows some instabilities (e.g., ripples) one may want to increase the value of the model damping weight.

# 7 References

All papers listed below are using Tomofast-x for performing inversions. For convenience, we split the papers to categories corresponding to the type of constraints they use.

- All types of constraints summary:

[1] J. Giraud, V. Ogarko, R. Martin, M. Lindsay, M. Jessell (2021): Structural, petrophysical and geological constraints in potential field inversion using the Tomofast-x open-source code, Geoscientific Model Development Discussions, https://doi.org/10.5194/gmd-2021-14

- Disjoint interval bound constraints:

[2] V. Ogarko, J. Giraud, R. Martin, and M. Jessell (2021), Disjoint interval bound constraints using the alternating direction method of multipliers for geologically constrained inversion: Application to gravity data, GEOPHYSICS 86: G1-G11, https://doi.org/10.1190/geo2019-0633.1

- Smart-gradient constraints:

[3] J. Giraud, M. Lindsay, M. Jessell, and V. Ogarko (2020), Towards plausible lithological classification from geophysical inversion: honouring geological principles in subsurface imaging, Solid Earth, 11: 419–436, https://doi.org/10.5194/se-11-419-2020

[4] J. Giraud, M. Lindsay, V. Ogarko, M. Jessell, R. Martin, and E. Pakyuz-Charrier (2019), Integration of geoscientific uncertainty into geophysical inversion by means of local gradient regularization, Solid Earth, 10: 193–210, https://doi.org/10.5194/se-10-193-2019

- Cross-Gradient constraints:

[5] R. Martin, J. Giraud, V. Ogarko, S. Chevrot, S. Beller, P. Gégout, M. Jessell (2021), Three-dimensional gravity anomaly data inversion in the Pyrenees using compressional seismic velocity model as structural similarity constraints, Geophysical Journal International 225(2): 1063–1085, https://doi.org/10.1093/gji/ggaa414

- Clustering constraints:

[6] J. Giraud, V. Ogarko, M. Lindsay, E. Pakyuz-Charrier, M. Jessell, R. Martin (2019), Sensitivity of constrained joint inversions to geological and petrophysical input data uncertainties with posterior geological analysis, Geophysical Journal International, 218(1): 666–688, https://doi.org/10.1093/gji/ggz152

- $L_p$ norm-based model damping:

[7] R. Martin, V. Ogarko, D. Komatitsch, M. Jessell (2018), Parallel three-dimensional electrical capacitance data imaging using a nonlinear inversion algorithm and Lp norm-based model regularization, Measurement, 128: 428-445, https://doi.org/10.1016/j.measurement.2018.05.099