BEL1D MANUAL

AUTHOR:
HADRIEN MICHEL

HADRIEN.MICHEL@ULIEGE.BE

VERSION 1.0 - JUNE 2019

HADRIEN MICHEL IS A PHD FELLOW WHOSE RESEARCHES ARE FUNDED BY THE F.R.S.-FNRS, FOND DE LA RECHERCHE SCIENTIFIQUE, BRUSSELS, BELGIUM.

Table of Contents

1

SECTION

Introduction to BEL1D

BEL1D is a method for the uni-dimensional geological imaging of the subsurface. It is the abbreviation of "Bayesian Evidential Learning 1D imaging". This method relies on the constitution of statistics-based relationships between geophysical data and model's parameters. This manual aims at guiding the user through the use of BEL1D for several applications via a set of Matlab Toolboxes (Michel, Nguyen, Kremer, Elen, & Hermans, Under Review):

- SNMR (see Section 2)
- Surface waves dispersion curves (see Section 3)
- Generalized case (see Section 4)

As the different workflows for each toolboxes are very similar, only the SNMR one will be described fully. For surface waves and the generalized cases, only the key points will be covered. For any misunderstanding in those two cases, the user is referred to the equivalent step in the SNMR case.

1.1 Requirements

In order to run BEL1D, you will need a Matlab license. In the specific cases of SNMR and surface waves, installation of secondary software (for the forward models runs) is required:

- A Windows computer, as the MEX files are compiled for Windows only (**MASW**). Otherwise, the MEX codes can be compiled for Linux (see `https://github.com/`

`eespr/MuLTI` for more details). This latter solution will require renaming the function in the BELDSW.m code.

- MRSMatlab (available upon request at the authors of MRSmatlab: Mueller-Petke, Braun, Hertrich, Costabel, and Walbrecker (2016)), a set of open-source Matlab toolboxes for the processing and the interpretation of Magnetic Resonance Sounding (also called Surface Nuclear Magnetic Resonance) data.

For the technical specifications of the computer, the codes can run on any machine **but** the ability to handle large data-sets will highly depend on the RAM available. At least 8GB of RAM is highly recommended and 16GB is advised. The codes are also able to use the Matlab parallel processing toolbox but the latter is not mandatory (no specific toolbox is required). The better the processor, the faster the computations.

1.2 Downloading BEL1D

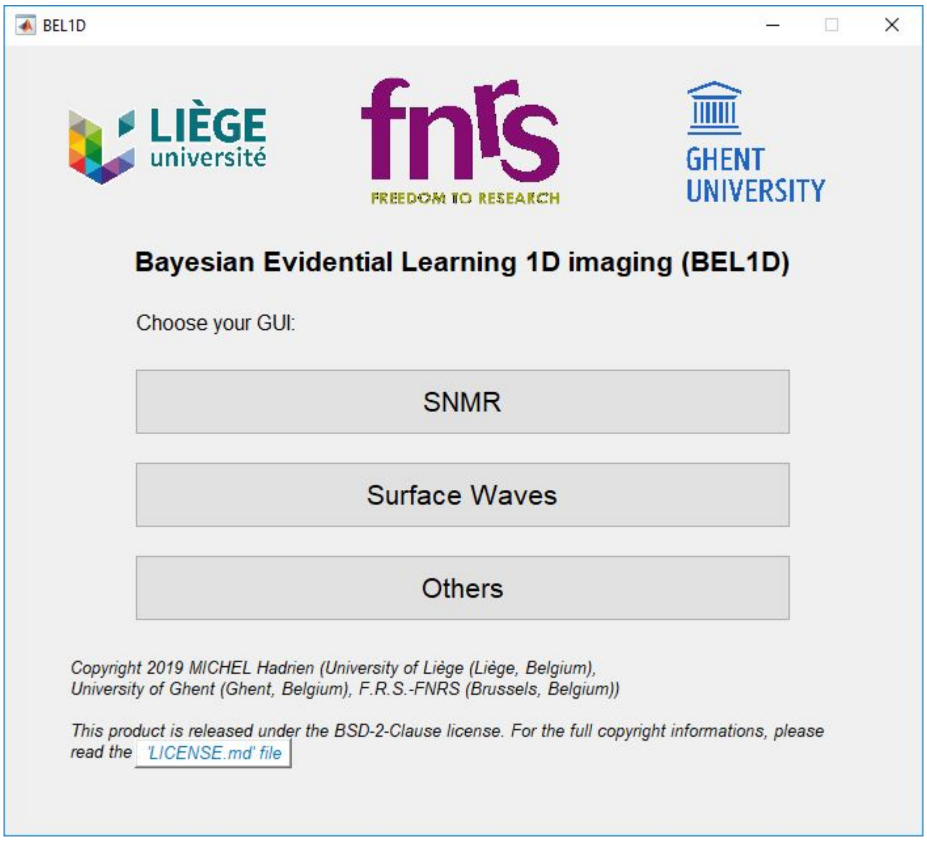
BEL1D is available to download on GitHub (`https://github.com/hadrienmichel/BEL1D`) under the BSD-2-clause license.

1.3 Installing BEL1D

To install BEL1D, simply copy all the files in a folder at a convenient location for your usage. You will need to run BEL1D from this folder later.

1.4 Launching BEL1D

To launch BEL1D, simply type `BEL1D` in Matlab's Command Window. You will be greeted with a menu proposing the different toolboxes (Fig. 1). Select the toolbox that you want in order to obtain the specific GUI's. In this menu, you can also obtain information about the copyright license by clicking on the 'LICENSE.md' file button.



Bayesian Evidential Learning 1D imaging (BEL1D)

Choose your GUI:

Copyright 2019 MICHEL Hadrien (University of Liège (Liège, Belgium), University of Ghent (Ghent, Belgium), F.R.S.-FNRS (Brussels, Belgium))

This product is released under the BSD-2-Clause license. For the full copyright informations, please read the 'LICENSE.md' file

Figure 1: BEL1D introduction menu

SECTION 2

BEL1D for SNMR data

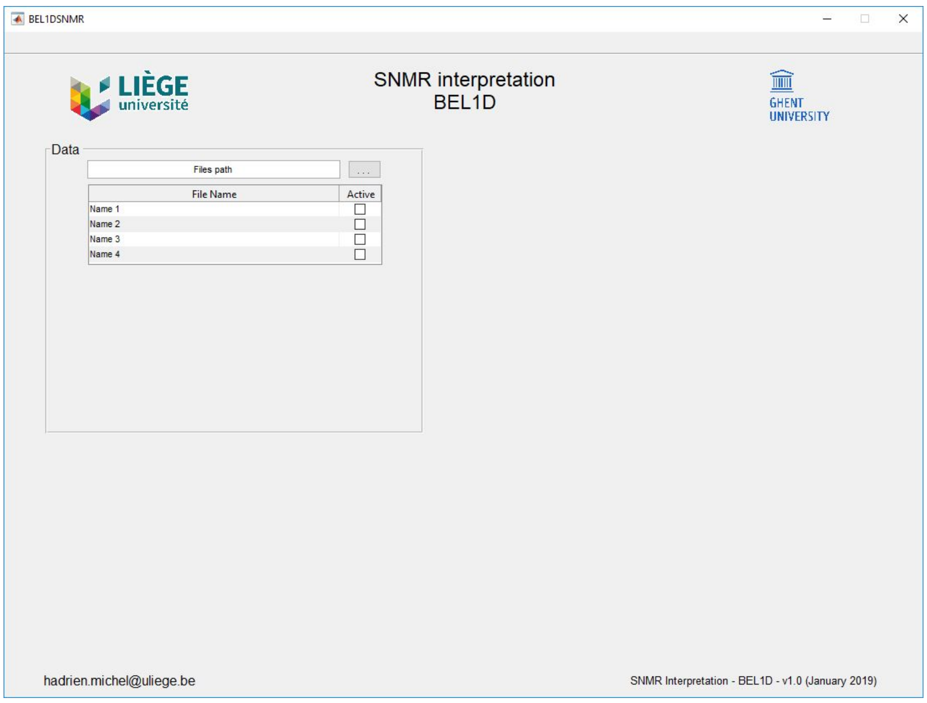


Figure 2: BEL1D for SNMR - initial state

Once you have selected the SNMR toolbox in the main menu, a GUI will appear (Fig. 2). Only some options of the full GUI are available at this stage as the different options will appear progressively through the different steps of computation. This toolbox enables the direct interpretation of coincident transmitter/receiver couples and in-loop receivers. As the use of multiple data-sets at once is also supported, the multi-central

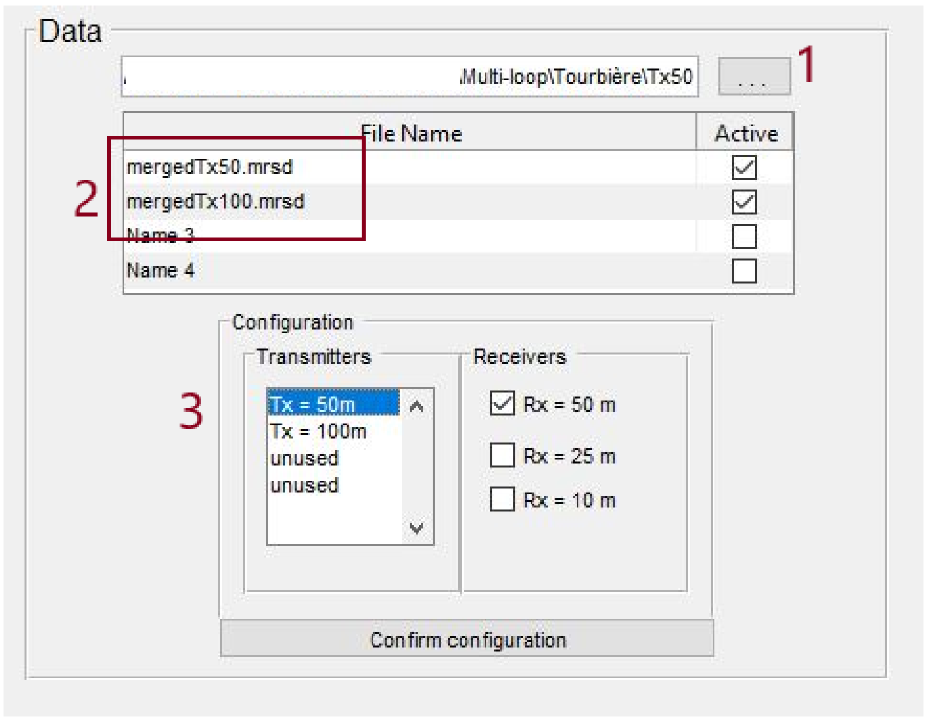
configuration (Kremer, Michel, Irons, et al., 2019) is also natively supported.
The toolbox is composed of four panels:

- Data
- Prior model space
- Kernels
- BEL1D

Many of the inputs of the toolbox are issued from MRSmatlab. For more details on the use of MRSmatlab, the reader is directed towards the manual (Kremer, Michel, & Mueller-Petke, 2019).

2.1 The *Data* panel

In the *Data* panel (Fig. 3), the user is invited to follow the three steps below.



.Multi-loop\Tourbière\Tx50

File Name	Active
mergedTx50.mrsd	✓
mergedTx100.mrsd	✓
Name 3	☐
Name 4	☐

Transmitters: [✓ Tx = 50m], Tx = 100m, unused, unused

Receivers: [✓ Rx = 50 m], ☐ Rx = 25 m, ☐ Rx = 10 m

Figure 3: BEL1D for SNMR - Data panel

1. Introduce the path to the main directory (containing the data files (**.mrsd*) issued from either data processing (MRSSignalPro toolbox from MRSMatlab) or data modelling (MRSModelling toolbox from MRSMatlab). Eventually the kernel files (**.mrsk*) should be present in the same folder if they are already computed with the MRSKernel toolbox from MRSMatlab or if they correspond to specific configurations (see 2.3).

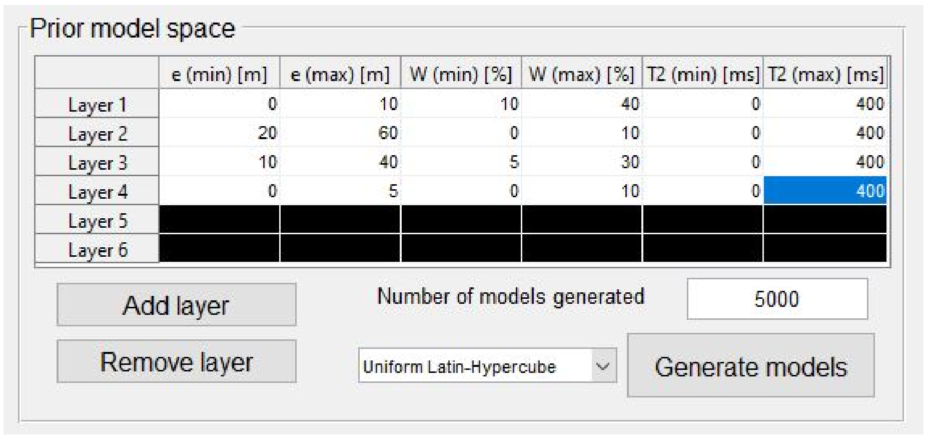
2. Introduce the names of the **.mrsd* files that are used. You must include the extension in the name. Once a name is completed, the user can choose to activate this

file by clicking on the corresponding checkbox. As soon as a file is activated, the configuration menu appears (see point 3 below).

3. Select the specific configuration that you want to test. Each file introduced in the second step corresponds to a transmitter size (the files are in the same order as the transmitters). Then, for each transmitter, the ensemble of corresponding receivers is proposed. Select the combination of transmitters/receivers that you want to use and click the *Confirm Configuration* button. Be aware that no possibilities to go backward are implemented yet!

2.2 The *Prior model space* panel

In the *Prior model space* panel (Fig. 4), you will define the prior model space. This panel appears once you have completed the previous panel. Here, you will need to introduce



Prior model space

	e (min) [m]	e (max) [m]	W (min) [%]	W (max) [%]	T2 (min) [ms]	T2 (max) [ms]
Layer 1	0	10	10	40	0	400
Layer 2	20	60	0	10	0	400
Layer 3	10	40	5	30	0	400
Layer 4	0	5	0	10	0	400
Layer 5						
Layer 6						

Number of models generated: 5000

Uniform Latin-Hypercube

Figure 4: BEL1D for SNMR - Prior panel

the ranges of the prior model space variables. To construct this space, a good idea could be to first look at the results of the deterministic inversion (especially for the determination of the number of layers) provided by MRSMatlab. The three parameters per layer are ranging between the *min* and *max* values introduced in the table. To add or remove layers, use the corresponding buttons.
You can choose between four different types of distributions in the drop down menu (the Latin-Hypercube samplers ensure a good coverage of the full space):

- Uniform (standard or Latin-Hypercube sampler)
- Gaussian (standard or Latin-Hypercube sampler)

The uniform distributions use the *min/max* parameters to define the ranges. The Gaussian distributions transforms the *min/max* parameters to define:

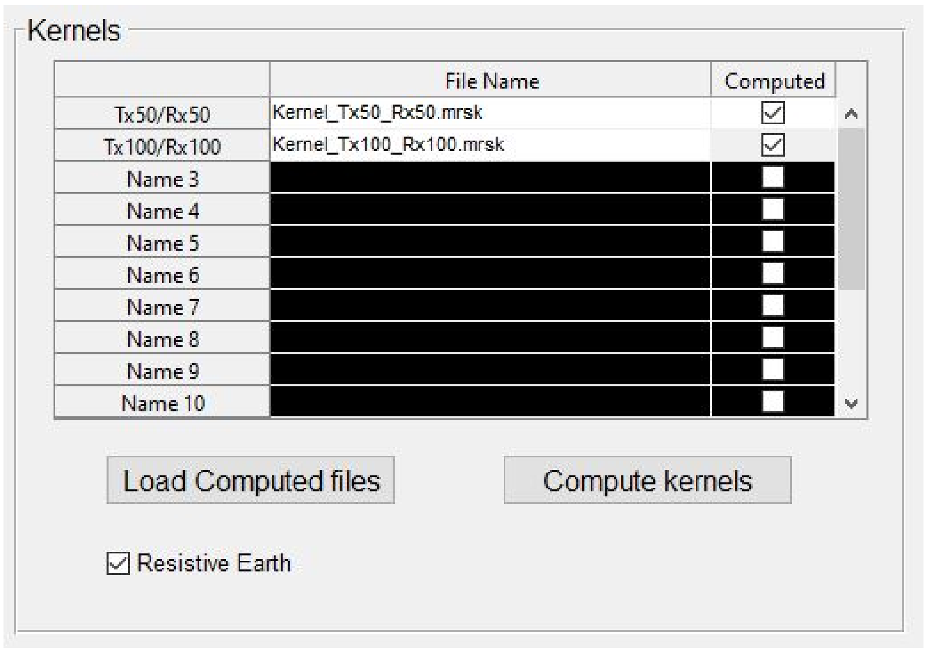
$$\mu = \frac{min + max}{2}$$

$$\sigma = \frac{max - min}{4}$$

Finally, you can choose the number of models that will be sampled in the space (the prior realizations) by changing the value in the corresponding box. Once you have completed all those tasks, you can click on the Generate models button to unlock further panels.

2.3 The *Kernel* panel

In this panel (Fig. 5), you can reference the kernel files (**.mrsk*). The default names for the kernels appears automatically. If the kernels already exists in the main directory (see 2.1) then, the corresponding *Computed* checkbox will automatically be ticked. You can change the name of the kernel file to reference an already existing file in the directory. The computation of the kernels is also possible, provided that MRSMatlab has been started in your current Matlab session. However, the possibilities are very limited, since complex cases are handles in the MRSKernel toolbox form MRSMatlab. You can only compute for coincident transmitter/receiver couples or in-loop configurations under the hypothesis of a resistive earth (neglecting the complex component of the signal) or a constant 100 Ohm.m resistivity. You can thus compute the missing files (if any) by



Kernels

	File Name	Computed
Tx50/Rx50	Kernel_Tx50_Rx50.mrsk	☑
Tx100/Rx100	Kernel_Tx100_Rx100.mrsk	☑
Name 3		☐
Name 4		☐
Name 5		☐
Name 6		☐
Name 7		☐
Name 8		☐
Name 9		☐
Name 10		☐

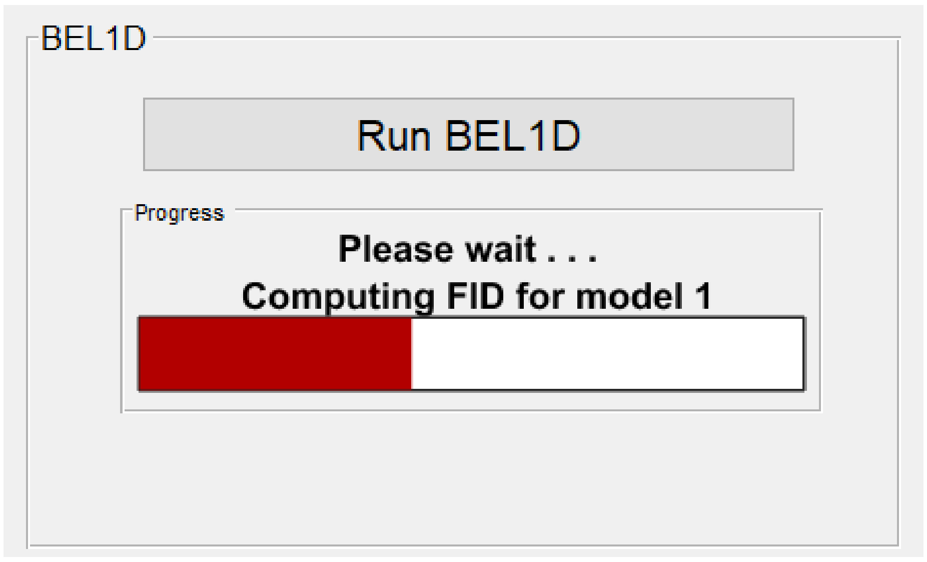
☑ Resistive Earth

Figure 5: BEL1D for SNMR - Kernel panel

clicking on the *Compute kernels* button. Then, Load the existing kernel files from the directory by clicking on *Load Computed files*. Loading all the files and generating the prior realizations (see 2.2) unlocks the final panel where all the computations are handled: *BEL1D*.

2.4 The *BEL1D* panel

This panel only proposes one option when unlocked: *Run BEL1D*. Simply click on the button to run BEL1D. This will launch the computations. If you want to use the parallel capabilities of Matlab (Parallel Computing Toolbox), start a parallel pool before clicking on the button. This enables the use of parallel computations of the forward model runs. However, this is not mandatory and will only increase the speed for very large data-sets or large numbers of prior realizations. While BEL1D is running (Fig. 6), the program will



Please wait . . .
Computing FID for model 1

Figure 6: BEL1D for SNMR - BEL1D running

keep the user up-to-date on the advancements of the computations (when possible). Different figures will also appear during the computations, informing on the possibilities of BEL1D to reduce the uncertainty. The first figure that will appear represents the CCA space (e.g., Fig. 7) and the second is the relative composition of the models dimensions in the CCA space (e.g., Fig. 8). For more information on those elements, please, refer to the reference paper (Michel et al., Under Review).

Then, a menu will appear, asking you if you want to compute the noise impact on BEL1D (Fig. 9). This option takes a lot of time, especially for very large data-sets and large numbers of prior realizations since it requires to run multiple times the principle component analysis. However, in case of high noise levels, it is preferable to run this test.

If you choose to compute the impact of noise, you will need to enter the estimated noise level on the dialogue box that appears. Then, the computation takes place! There is no implemented way to know the advancement of the code. It can take several minutes, and in some cases tens of minutes (very large prior and thus large number of realizations, large data-sets, etc.). Once the impact is computed, it is automatically taken into account and the remaining BEL computations are ran.

Finally, when all computations are made, the *BEL1D* panel changes to a menu with basic options for processing of the results and saving the results (Fig. 10). There, you can either visualize the models in the parameter space (*Show Parameters distributions*) or in the model space (*Show Models distributions*). You can finally save the results (*Save Results*) in a chosen *'*.mrsbel'* file. Attention: in order for the saved results to contain the simulated data-sets for the posterior, you must first show the models in the model space (*Show Models distributions*). More details on those in the next subsection.

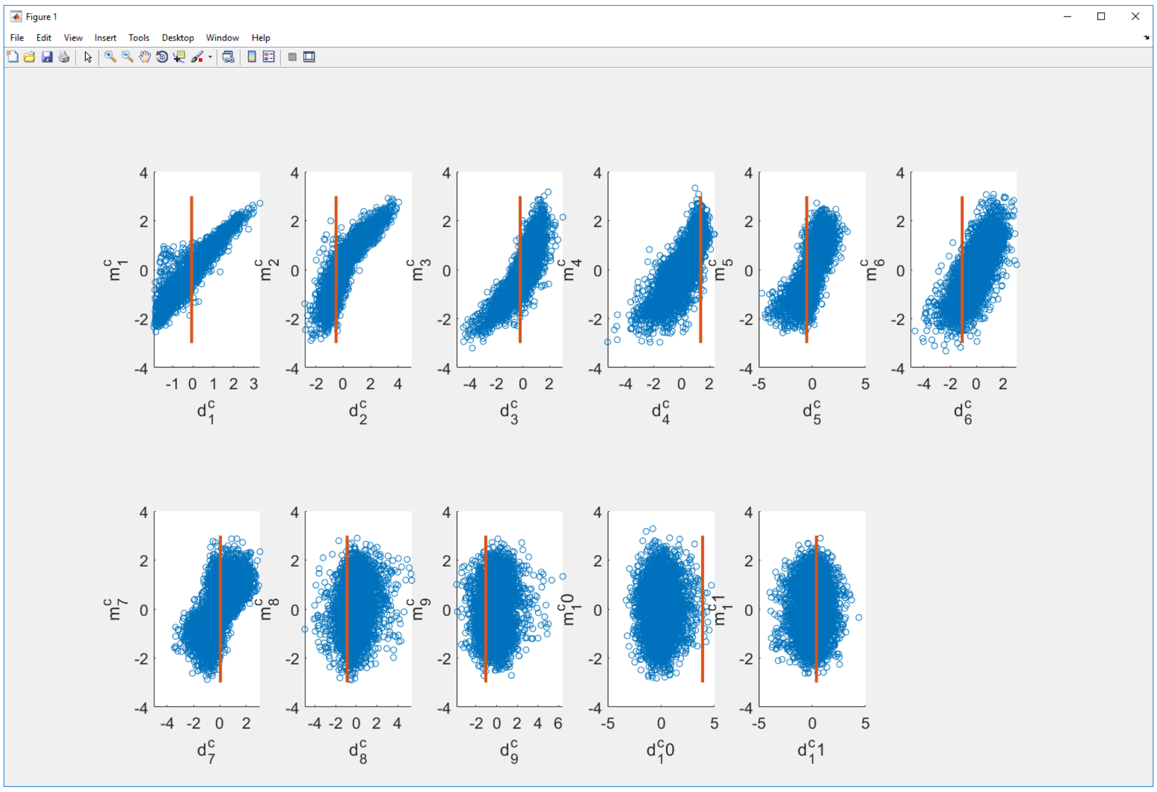


Figure 7: BEL1D for SNMR - example of CCA space figure

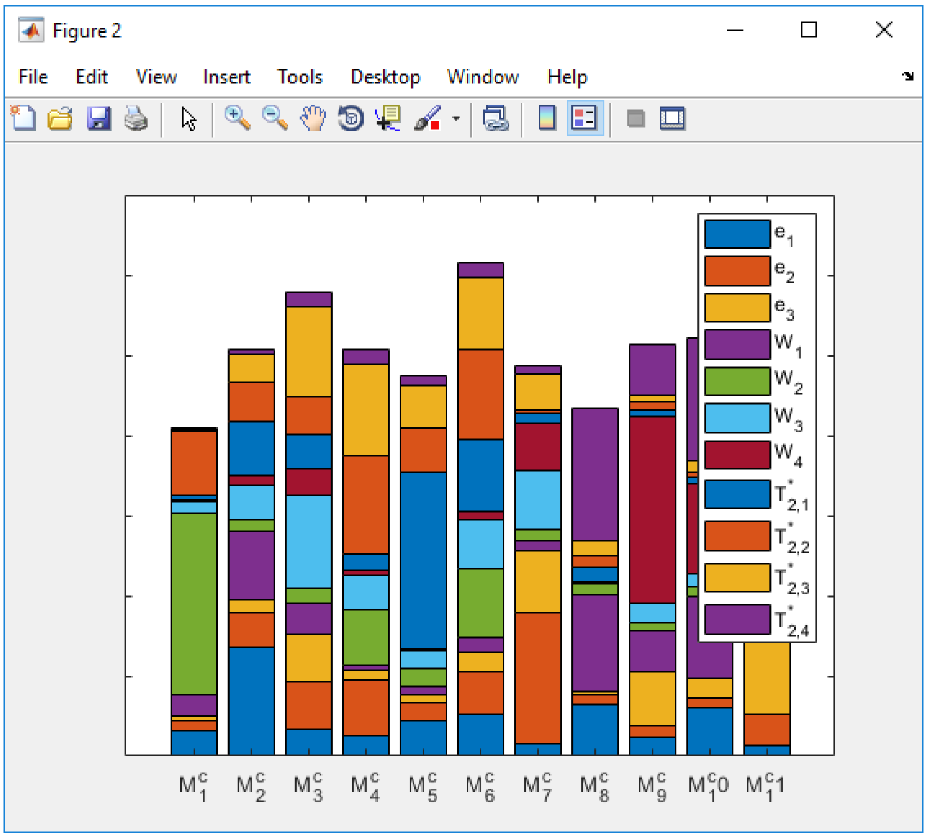
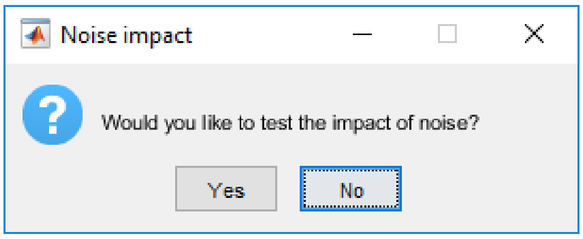


Figure 8: BEL1D for SNMR - example of CCA space component figure



Would you like to test the impact of noise?

Figure 9: BEL1D for SNMR - menu to account for noise level

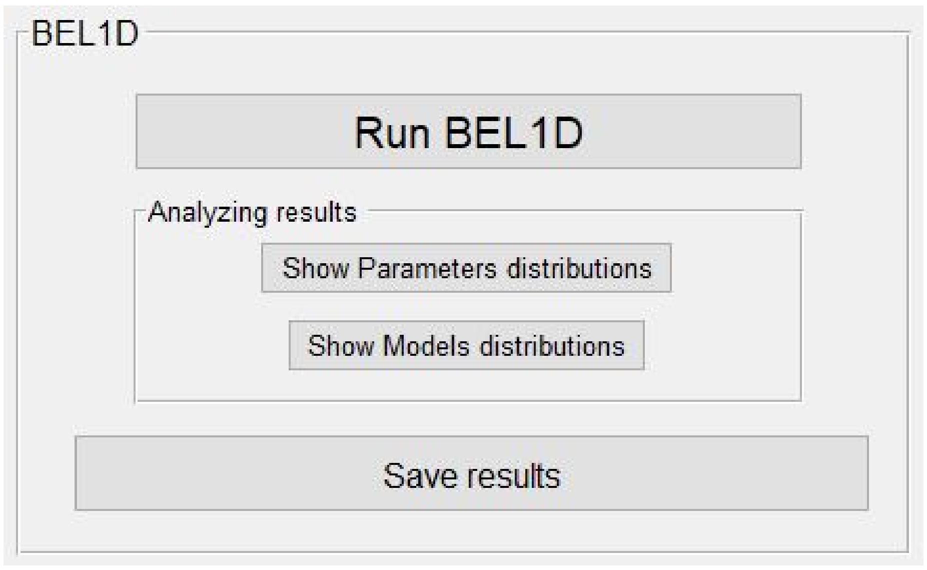


Figure 10: BEL1D for SNMR - BEL1D panel after computations

2.5 Processing the results

The simple processing of the results available inside the toolbox enables to analyze the results rapidly. You can either show the reduction of uncertainty in the parameter space (Fig. 11) or the ensemble of sampled models in the model space (Fig. 12). For the parameter space, the full representation is proposed, with links between the different parameters. Nonetheless, a simplified version is also proposed with the parameters assumed independant (be careful, this is not true since the models are sampled as sets of dependent parameters).

In Fig. 11, you can see the different parameters in the X and Y axis, with for each combinations, scatter plots showing the prior and the posterior models. It is a way to observe directly the reduction of uncertainty for the different parameters, and to eventually spot relations that arise from BEL1D. On the diagonal of the figure, the corresponding histograms are displayed.

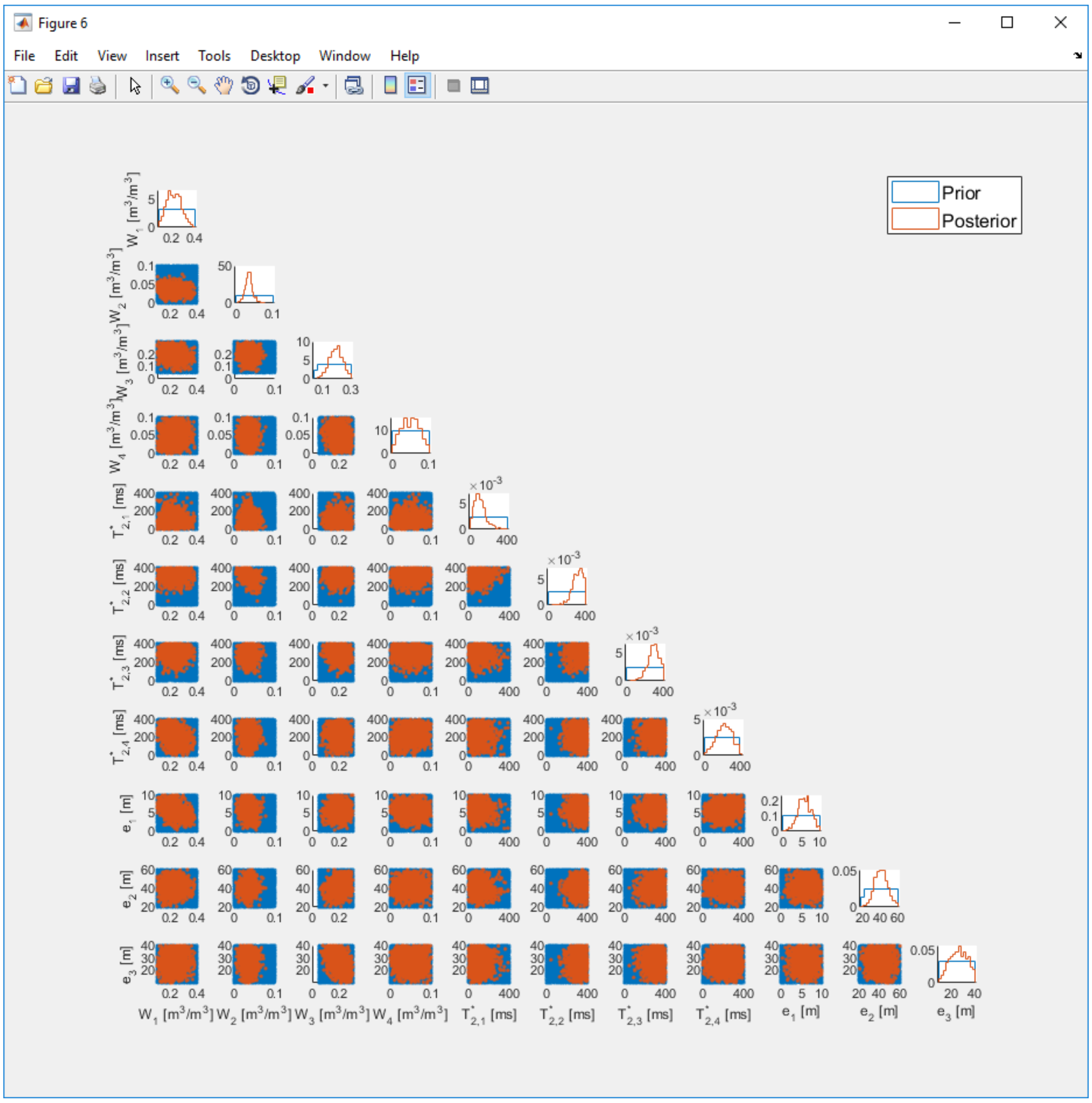


Figure 11: BEL1D for SNMR - Results in the parameter space

In Fig. 12, you can observe the different models that are sampled with distributions along the depth of the parameters. To enhance visualization, a scale-bar showing the RMS is present. For each color in the scale-bar, the same number of models. This scale-bar therefore gives an idea of probability as well, the smaller the bar corresponding to a given color, the more probable the parameters values that are associated. Note that since all the models where acquired uniformly, all the models presented are equiprobable!

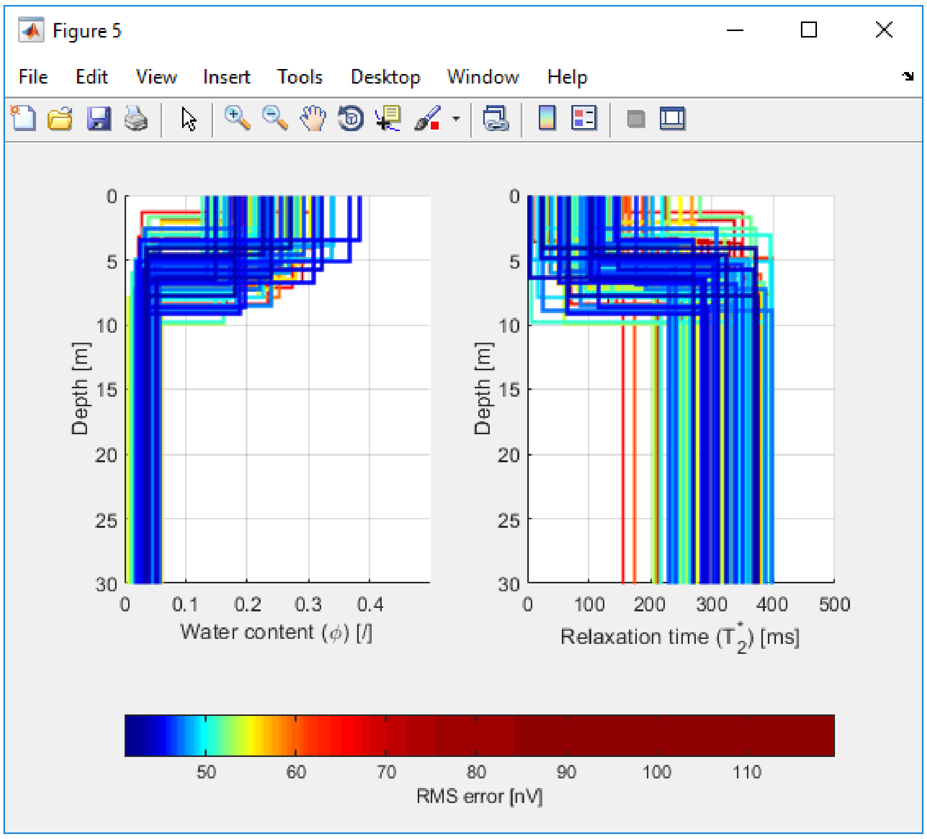


Figure 12: BEL1D for SNMR - Results in the model space

To further analyze the results, one can directly observe the values obtained in the saved '*.*mrsbel*' file. In Matlab, run the command:

```
Tmp = load('Filename.mrsbel','-mat');
SNMR = Tmp.SNMR;
SNMR =
  struct with fields:

        Data: [1x1 struct]
      Models: [1x1 struct]
     Kernels: [1x1 struct]
    Solution: [1x1 struct]
```

The `SNMR` structure that you obtain is composed of four structures:

- Data: the field data
- Models: the prior model space
- Kernels: unused (possibly the kernel files)
- Solution: the posterior model space

```
>> SNMR.Data
ans =
  struct with fields:

            proclog1: [1x1 struct]
            proclog2: []
            proclog3: []
            proclog4: []
                used: [1x1 struct]
           activated: [4x4 logical]
    proclogNamesCode: {1x4 cell}
          d_real_obs: [1x35712 double]
>> SNMR.Models
ans =
  struct with fields:

       param: [6x6 double]
    nbLayers: 4
        type: 2
           N: 5000
       model: [1x1 struct]
>> SNMR.Kernels
ans =
  struct with fields:

    nb: 3
>> SNMR.Solution
```

```
ans =
  struct with fields:

    model: [1x1 struct]
```

In the `Data` structure, you will find the raw data in the fields `proclog1`, `proclog2`, `proclog3` and `proclog4`. Those fields are simple copies of the `proclog` structures contained in the data files. `d_real_obs` is a vector containing the true data as arranged for BEL1D. The other elements in the structure are useless for results processing.
The `Models` structure defines the prior model space and the associated realizations. `param` contains the properties of the prior model space (the table from the Prior model space panel —see subsection 2.2), `nbLayers` is the number of layers in the models, `N` is the number of prior realizations and `type` corresponds to the choice of sampler in the drop-down menu:

- 1 = Uniform distributions
- 2 = Uniform distributions with a Latin-Hypercube sampler
- 3 = Gaussian distributions
- 4 = Gaussian distributions with Latin-Hypercube sampler

The prior realization are contained in the `SNMR.Models.model` structure:

```
SNMR.Models.model
ans =
  struct with fields:

      thick: [5000x3 double]
      depth: [5000x4 double]
      water: [5000x4 double]
         T2: [5000x4 double]
        res: [5000x1 double]
    results: [5000x72x496 double]
```

In this structure, you will find the different parameters defining the models: `thick` (and `depth` the cumulative sum of `thick` with first a `0`), `water` and `T2`. `res` is reserved for future developments. Finally, the simulated data corresponding to all the different models is contained in `results`.
Finally, the posterior distribution is defined in the `SNMR.Solution.model` structure. This structure contains the same elements as the equivalent for the prior model space, but the models are the results of BEL1D.
From this, you can process the results as you wish, as everything is in your hand.

SECTION 3

BEL1D for surface waves dispersion curves

Once you have selected the Surface Waves toolbox in the main menu, a GUI will appear (Fig. 13). Only some options of the full GUI are available at this stage as the different options will appear progressively through the different steps of computation.
The full GUI contains three panels:

- Data
- Prior model space
- BEL1D

3.1 Preamble

Before any use of the surface wave dispersion curve toolbox, you must ensure that the *gpdc.exe* code is added in the path. You can do so by testing gpdc in the command window/bash at any location. To do so, follow the example shown in Gpdc - GeopsyWiki. If you can run *gpdc* from any location, you will be able to proceed to further steps. Otherwise, ensure that *gpdc* is added to your path and eventually restart your computer.

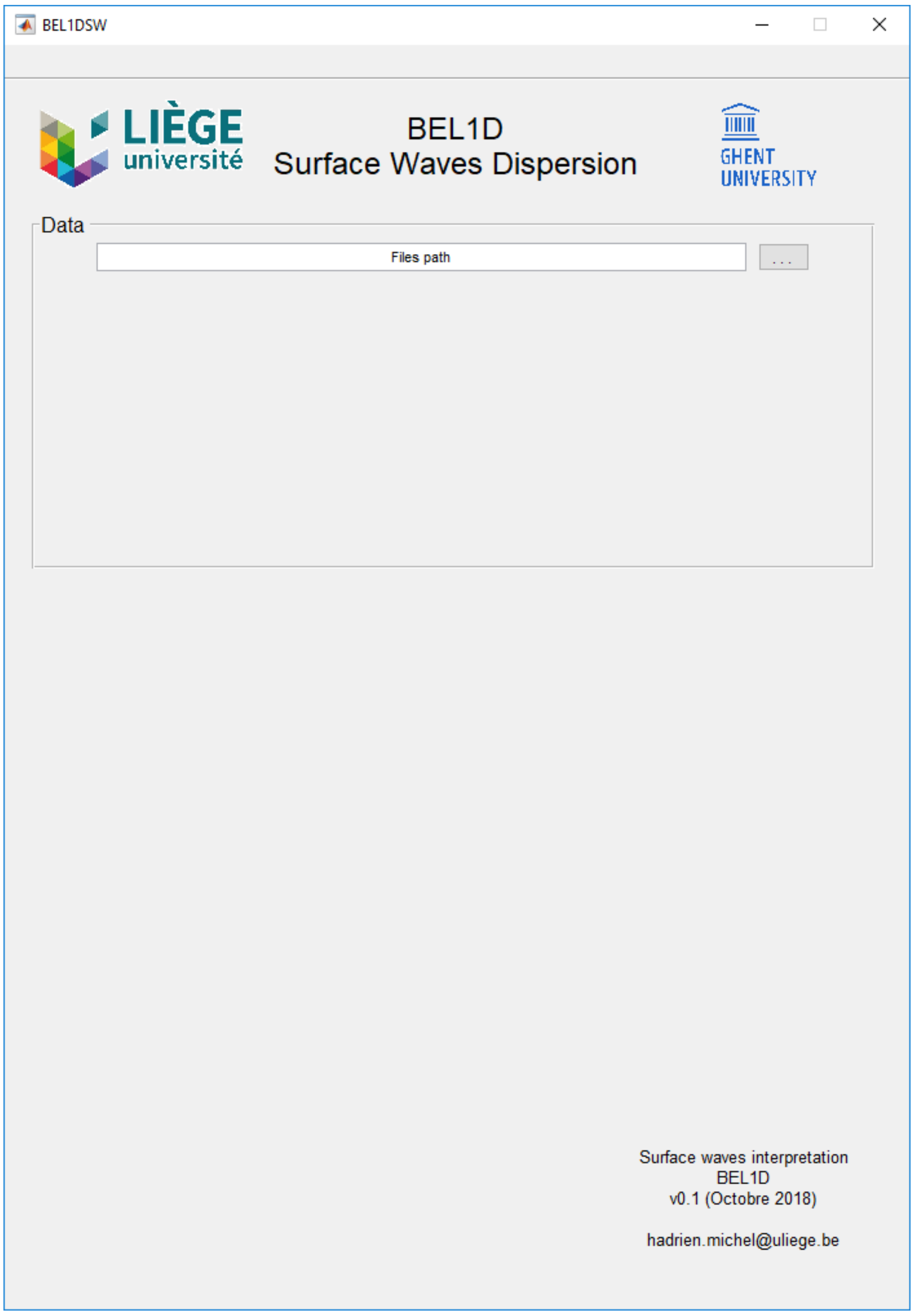


Figure 13: BEL1D for Surface Waves - initial state

3.2 The *Data* panel

In the data panel, you can select the dispersion curve. You can either use the file saved from Geopsy (see tutorial). Otherwise, you can use a custom '**.maswd*' file which is a '**.mat*' file saved with a different extension. This latter is composed as follow.

```
tmp = load('File.maswd','-mat');
dispersion = tmp.dispersion;
dispersion.Curve =
  struct with fields:

         f: [64x1 double] % The sampled frequencies
        ct: [64x1 double] % The corresponding phase velocities
    lambda: [64x1 double] % The corresponding wavelength
```

To load the data, click on the button and select the data file. Once the data is loaded, the *data* panel will be updated with a visualization of the data (Fig. 14).

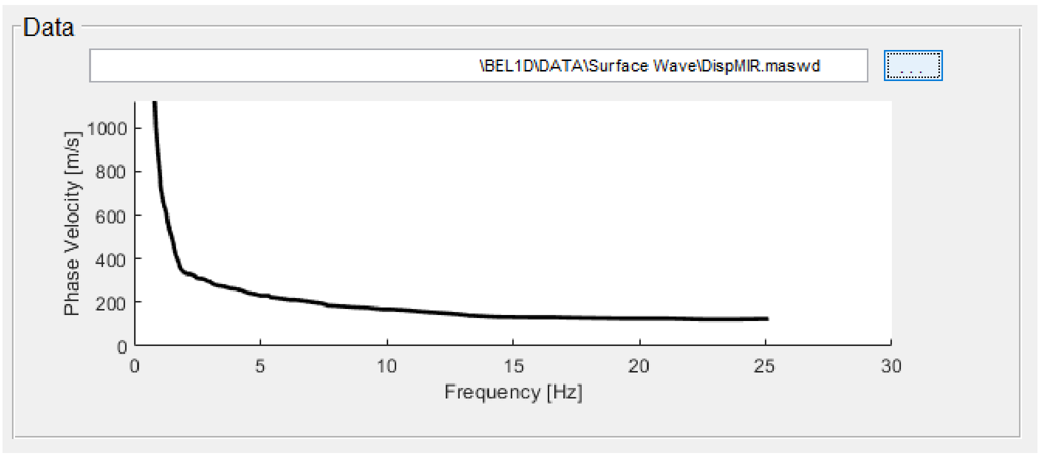


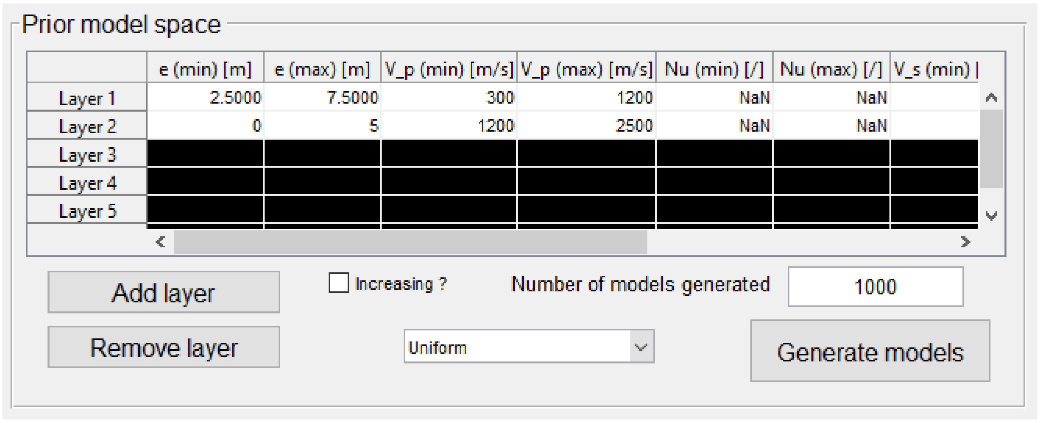
Figure 14: BEL1D for Surface Waves - Data panel after loading

3.3 The *Prior model space* panel

The *Prior model space* panel (Fig. 15) appears when data is loaded. There, you can describe the prior in the same way as proposed in the SNMR case (see 2.2). In this case however, the parameters are:

- Thickness
- P-wave velocity
- Poisson's coefficient (optional: NaNs = not taken into account)
- S-wave velocity
- Density

The addition of constraints on the Poisson's coefficient changes the samplers to rejection samplers in order to obtain models that correspond to the ensemble of constraints. Checking the *Increasing* box ensure that the parameters (P- and S-waves velocities and densities) are all increasing with depth. Again, you can choose between Uniform and



	e (min) [m]	e (max) [m]	V_p (min) [m/s]	V_p (max) [m/s]	Nu (min) [/]	Nu (max) [/]	V_s (min)
Layer 1	2.5000	7.5000	300	1200	NaN	NaN	
Layer 2	0	5	1200	2500	NaN	NaN	
Layer 3							
Layer 4							
Layer 5							

☐ Increasing ?

Number of models generated 1000

Uniform

Figure 15: BEL1D for Surface Waves - Prior model space panel

Gaussian distributions with classical or Latin-Hypercube samplers. However, the use of Latin-Hypercube sampler is impossible with rejection samplers (Poisson's coefficient or *Increasing*).

3.4 The *BEL1D* panel

Here, the options are exactly the same as in the SNMR case (Subsection: 2.4).

3.5 Post-processing

The result of BEL1D on MASW is stored in a '**.maswbel*' file that you can load using the command

```
MASW = load('namefile.maswbel','-mat');
MASW = MASW.MASW;
  struct with fields:

        Data: [1x1 struct]
      Models: [1x1 struct]
    Solution: [1x1 struct]
```

This structure contains the dispersion curve used to constrain the solution (`MASW.Data`), the prior realizations and their associated data-sets (`MASW.Models`) and the models from the posterior (and their associated data-sets if they were already computed) in the `MASW.Solution` structure. Processing of the results can be done similarly to the SNMR case (see Subsection 2.5).

4

SECTION

BEL1D in the generalized case

Once you have selected the General toolbox in the main menu, a GUI will appear (Fig. 16). Only some options of the full GUI are available at this stage as the different options will appear progressively through the different steps of computation.
The full GUI contains three panels:

- Data
- Prior model space
- BEL1D

Here, we will explain the way the general toolbox works. It especially requires two different things:

- A "field" data-set that corresponds to a given format
- A forward modelling function that works with a given set of input/output rules.

The remaining of this toolbox is extremely similar as the previous case (Section 3) and will therefore only be briefly introduced.

4.1 The data formatting

The data-set must be provided in a n-by-2 matrix saved in a **.mat* file under the name *data*. The first column represents the "x" coordinate of the data-set (for a vertical seismic profile (VSP), this would be the depth of the geophone) and the second column ("Y") the corresponding measurement (for VSP, this would be the time of first arrival).

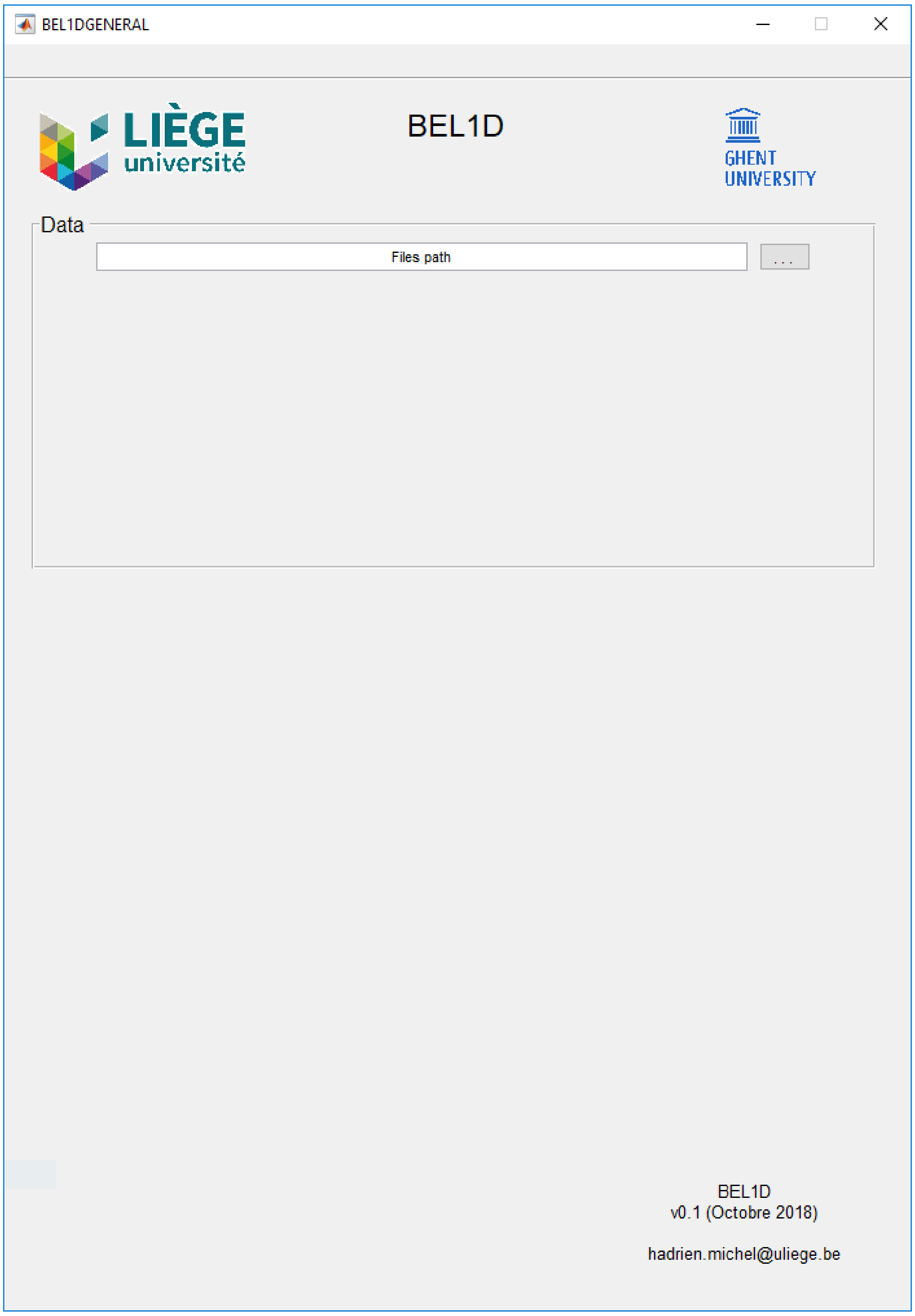


Figure 16: BEL1D for the General case - Initial state

4.2 The forward model function

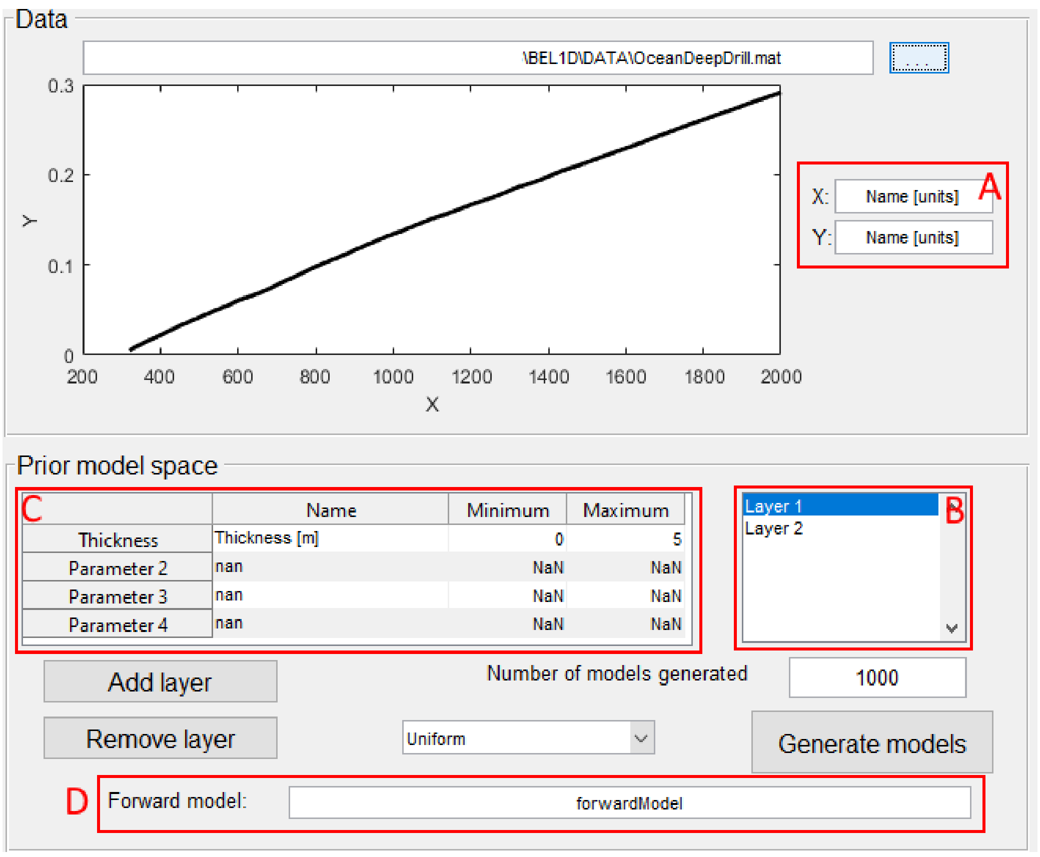
The forward modelling function must correspond to the formatting below. However, the name of the function is not constrained nor is the internal structure. Only the input/output formatting matters. Please, note that the function only computes the forward model for one given model at a time.

```
function [Y] = forwardModel(X,param)
% FORWARDMODEL is a function that performs the forward modelling
% for a given experiment.
% The function takes as arguments:
%       - The X coordinates for which the model must be
%         calculated (X).
%       - The parameters of the model in a vector form
%         (size [1 x ((n x m)-1)])
%           o n is the number of layers
%           o m is the number of parameters
%         The vector is expressed as follow:
% >> param =
%
%        [e_1,e_2,...,e_n-1,param2_1,...,param2_n,param3_1,...,
%        param3_n,...,param4_1,...,param4_n]
%

% The forward model itself, that computes the response of the model
% (Y) for the set of values (X) given the models parameters (param).
Y = YourForwardModel(X,param);

% X and Y must be vectors that have the same length!

end
```

4.3 The toolbox

The initial state of the toolbox is presented in Fig. 16. There, you must enter the path to the '*.*mat*' data file (see Subsection 4.1) by either entering the full path manually or clicking on the corresponding button (. . .). Once the path is entered, the data will appear in a graph just below the path. There, you can enter manually the names of the two dimensions of the data by replacing the names in the corresponding X and Y boxes (box A in Fig. 17). Those names are latter used for visualization purpose. They are however not mandatory.

Once the data is introduced, you can describe the prior model space by defining the boundaries of the different parameters layer-by-layer in box C from Fig. 17 (click on the corresponding layer in the listbox on the right - box B in Fig. 17). You can name and give units to the parameters apart from the thickness. Those names will be used latter for visual purpose. The use of the entirety of the possible parameters is not mandatory, but the number of parameters **must** correspond to the forward model. You can add con-



\BEL1D\DATA\OceanDeepDrill.mat

X: Name [units]
Y: Name [units]

	Name	Minimum	Maximum
Thickness	Thickness [m]	0	5
Parameter 2	nan	NaN	NaN
Parameter 3	nan	NaN	NaN
Parameter 4	nan	NaN	NaN

Layer 1
Layer 2

Number of models generated 1000

Uniform

Forward model: forwardModel

Figure 17: BEL1D for the General case - Data introduced

straints on the model space by modifying the code of the `ModelGenerator.m` function situated in the *General* folder of the codes. Follow the example in the code shown bellow to proceed the addition of those conditions. Pay attention, the use of Latin-Hypercube sampler is not working with added conditions.

```
function [models] = ModelGenerator(type, N, parameters, nb_layer)

    thick = parameters(1:end-1,1:2);
    param2 = parameters(:,3:4);
    param3 = parameters(:,5:6);
    param4 = parameters(:,7:8);
    %% Some code for sampling and initialization (not shown here)
    something;
    something_else;
    etc;
    %% Add here your constraints on the sampled models (/!\ may ...
        not work with latin-hypercube sampler), else put OK = true;
    if passed
        tmp = models.thick+models.param2(:,1);% Here, the ...
            condition is on the limit of the sum of the thickness ...
            and the second parameter.
        N = N - sum(tmp≥10);
        models_save.thick = [models_save.thick; ...
            models.thick(tmp≥10)];
        models_save.param2 = [models_save.param2; ...
            models.param2(tmp≥10,:)];
    else
        tmp = models.thick+models.param2(:,1);
        N = N - sum(tmp≥10);
        models_save = models;
        models_save.thick = models_save.thick(tmp≥10);
        models_save.param2 = models_save.param2(tmp≥10,:);
        end
        passed = true;
        if N == 0
            OK = true;
        end
    end
    %% If no added conditions:
    models_save = models;
    OK = true;
    %% End of the code for the addition of conditions
    models = models_save;
end
```

After the definition of the prior model space, you must provide the name of the forward model function (box D in Fig. 17). The function formatting is covered in subsection 4.2. This function needs to be located in either the *General* folder or in the main directory or any other location that is accessible for Matlab.
Further steps of BEL1D are exactly the same as in the two previous cases and are therefore not described.

4.4 Processing of the result

The result of BEL1D in the genralized case are stored in a '*.mbel' file that you can load using the command:

```
MBEL = load('namefile.mbel','-mat');
MBEL = MBEL.MBEL;
  struct with fields:

               Data: [1x1 struct]
         Data_Xname: 'X Name [units]'
         Data_Yname: 'Y Name [units]'
             Models: [1x1 struct]
           Solution: [1x1 struct]
```

The names and units of the different parameters are stored in the `Models.param.names`. The names of the data dimensions are stores in *Data_Xname* and *Data_Yname*. The remaining structures are the same as in the surface wave and SNMR cases (see Subsection 2.5).

5

SECTION

References

Kremer, T., Michel, H., Irons, T., Hermans, T., Nguyen, F., & Mueller-Petke, M. (2019). Improving the accuracy of 1D surface nuclear magnetic resonance surveys using the multi-central-loop configuration. *Journal of Applied Geophysics*.

Kremer, T., Michel, H., & Mueller-Petke, M. (2019). MRSM*atlab manual* (Tech. Rep.).

Michel, H., Nguyen, F., Kremer, T., Elen, A., & Hermans, T. (Under Review). 1d geological imaging of the subsurface from geophysical data with bayesian evidential learning. *Computers & Geosciences*.

Mueller-Petke, M., Braun, M., Hertrich, M., Costabel, S., & Walbrecker, J. (2016, jul). MRSmatlab — a software tool for processing, modeling, and inversion of magnetic resonance sounding data. *GEOPHYSICS*, *81*(4), WB9–WB21. doi: 10.1190/geo2015-0461.1