A Short Manual for MUpen2D Software

V. Bortolotti[a], L. Brizi[b], A. Nagmutidinova[a], G. Landi[c], F. Zama[c]

[a]*Department of Civil, Chemical, Environmental, and Materials Engineering, University of Bologna*
[b]*Department of Physics and Astronomy, University of Bologna, Italy*
[d]*Department of Mathematics, University of Bologna*

version 1.5 – April 2023

This two-dimensional (2D) inversion program software developed at the University of Bologna (Bologna, Italy), named MUpen2D, is an enhanced version of Upen2DToll that processes 2D ASCII Nuclear Magnetic Relaxation (NMR) data to produce distributions in the two NMR dimensions (2DNMR maps). At present, the following 2DNMR distribution maps are considered: longitudinal-transverse T_1-T_2 relaxation times, transverse-transverse T_2-T_2 relaxation times, and diffusion - transverse D-T_2 NMR parameters. MUpen2D implements the Multiple Uniform Penalty algorithm by extending the Uniform PENalty principle to two-dimensional data, including L1 norm and locally adapted L2 norm penalty terms. The algorithm produces the 2DNMR maps by solving a sequence of regularized least-squares problems. The regularization parameters are computed by an automatic update rule based on the Uniform Penalty principle. The software is written in Matlab® script language and it comes also with a standalone command line executable version and a compiled standalone graphic interface version, developed with App Designer, both for Windows OS and Linux OS. To perform the inversion the tool allows the user to handle some parameters that greatly influence the quality of the 2DNMR maps and also the computation time. There is also a set of diagnostic tools that allows the user to evaluate the quality of the measured data. Remember that in order to have a good inversion it is fundamental to have high quality data (data without too many outliers or electronic baseline, relaxation curves not correctly sampled or distorted, etc…).

After a brief introduction of the theory, the algorithm used in the program is presented. Then the installation of the software, the user interface and how to use the program are described. Finally, some examples of computation of the accompanying data (and of the inversion parameters used) are presented.

1 Introduction and General Theory

The structure of different types of porous media can be analyzed through measurements of the NMR relaxation of ^{1}H nuclei. NMR parameters like the longitudinal (T_1) and the transverse (T_2) relaxation times, as well as the self-diffusion molecular coefficient, can be determined and associated to properties of the ^{1}H fluids and saturated porous media.

In case of heterogeneous porous media samples, one expects to measure a multi-exponential decay signal from the relaxation curves measurements, as well as from the diffusion measurements. So, for example in case of 1D signal, one expects to find:

$$S(t_i) = g_i = \sum_{k=1}^{M} A(T_k)\,\mathrm{e}^{-\frac{t_i}{T_k}} + \varepsilon_i \quad \text{i=1, …, N} \qquad 1)$$

where, T_k are the relaxation times, $A(T_k)$ the unknown system of relaxation times and ε_i represent the signal's noise. N is the number of acquired points of the relaxation curve and M the number of relaxation components of the underlying multi-exponential model.

The problem of determining the two-dimensional distribution (2D problems) of NMR parameters (relaxation times and diffusion coefficients) from NMR data is an inverse ill-posed problem modeled by a first kind Fredholm Integral Equation having separable exponential kernels. For example, in the case of an Inversion Recovery IR-CPMG experiment, the relaxation data $S(t_1, t_2)$ can be expressed with the following continuous model:

$$S(t_1,t_2) = \int\int_0^{\infty} k_1(t_1,T_1) k_2(t_2,T_2) F(T_1,T_2)\, dT_1\, dT_2 + e(t_1,t_2) \qquad 2)$$

where $F(T_1, T_2)$ represents the unknown distribution of T_1 and T_2 relaxation times and $e(t_1,t_2)$ represents additive Gaussian noise. The evolution times t_1 and t_2 are independent variables in the two dimensions: t_1 is the evolution time parameter of the experiment in the first dimension (IR, CPMG or Diffusion experiment) and t_2 is the evolution time parameter in the second dimension (CPMG experiment).

The kernels $k_1(t_1, T_1)$ and $k_2(t_2, T_2)$ are decaying exponential functions whose expression depends on the specific NMR experiment and the unknown distribution $F(T_1, T_2)$ is supposed to be greater or equal to a constant $\ell \in \mathbb{R}$, not necessarily positive.

In the present version, the code implements the kernels for IR-CPMG (T_1-T_2 model), CPMG-CPMG (T_2-T_2 model), and Diffusion-CPMG (D-T_2 model) experiments.

In the T_1-T_2 case, the two kernels have the following expressions:

$$k_1(t_1,T_1) = 1 - 2\, e^{-\frac{t_1}{T_1}}, \qquad k_2(t_2,T_2) = e^{-\frac{t_2}{T_2}}$$

while in the T_2-T_2 case, the two kernels have the following expressions:

$$k_1(t_1,T_{21}) = e^{-\frac{t_1}{T_{21}}}, \qquad k_2(t_2,T_{22}) = e^{-\frac{t_2}{T_{22}}}$$

In the case of D-T_2 the acquired echo-amplitude $S(t_1, t_2)$ has the following expression:

$$S(t_1,t_2) = \int\int_0^{\infty} k_1(t_1,D)\, k_2(t_2,T_2) F(T_2,D)\, dD\, dT_2 + e(t_1,t_2)$$

With the following exponential kernels:

$$k_1(t_1,D) = e^{-t_1 D}, \qquad k_2(t_2,T_2) = e^{-\frac{t_2}{T_2}}$$

Due to the large dimension of the data and the inherent ill-posedness of the inverse problem, a significant issue in 2DNMR inversion is to ensure both computational efficiency and accuracy.

There are several codes, usually developed in Matlab®, that solve this ill-posed problem using different methods mostly based on Tikhonov regularizations.

MUPen2D implements a Multiple Uniform Penalty algorithm by extending the Uniform PENalty principle[1] to two-dimensional data, including L1 norm and locally adapted L2 norm penalty. Therefore, it computes locally adapted regularization parameters and approximate solutions by solving a sequence of regularized least squares problems.

Considering the discretized form of the integral equation (2), in order to avoid artifact peaks in the computed distribution, MUPen2D solves the following minimization problem:

$$\min_f \left\{ \|\mathbf{Kf}-\mathbf{s}\|^2 + \omega_1 \sum_{i=1}^{N} \lambda_i (\mathbf{Lf})_i^2 + \omega_2\, \alpha \|\mathbf{f}\|_1 \right\}$$

Where $\mathbf{K} = \mathbf{K}_2 \otimes \mathbf{K}_1$ is the Kronecker product of the discretized exponential kernels $\mathbf{K}_1 \in \mathbb{R}^{M_1 \times N_1}$, $\mathbf{K}_2 \in \mathbb{R}^{M_2 \times N_2}$, $\mathbf{s} \in \mathbb{R}^M, M = M_1 \cdot M_2$, is the discrete vector of the measured noisy data, $\mathbf{f} \in \mathbb{R}^N$, $N = N_1 \cdot N_2$ is the vector reordering of the distribution, $\|\cdot\|$ is the L_2 norm, and $\mathbf{L} \in \mathbb{R}^{N \times N}$ is the discrete Laplacian operator. The locally adapted regularization parameters λ_i and the L1 norm regularization parameter α are computed according to the Uniform Penalty [5]. Where the default values of the weights are $\omega_1 = \omega_2 = 1$. Depending on the data and problem type, it is possible to introduce further regularization flexibility by setting ω_i, s.t. $\omega_1 + \omega_2 = 1$.

Given an approximated distribution **f**, the automatic selection rule for the regularization parameters can be described as follows:

$$\alpha = \frac{\|\mathbf{Kf}-\mathbf{s}\|^2}{(N+1)\|\mathbf{f}\|_1} \qquad 3)$$

$$\lambda_i = \frac{\|\mathbf{Kf}-\mathbf{s}\|^2}{(N+1)\left(\beta_0 + \beta_p \max_{\mu \in I_i} \left\{ vec(\|\nabla \mathbf{F}\|_\mu^2) \right\} + \beta_c \max_{\mu \in I_i} \left\{ (\mathbf{Lf})_\mu^2 \right\}\right)} \qquad i = 1, \ldots, N \qquad 4)$$

Where $vec(\cdot)$ denotes the operator that stacks a matrix column-wise to produce a column vector and **F** is the 2D distribution map corresponding to **f** (i.e., **f**=*vec*(**F**)). The indices subsets I_i, i=1, …, N, are related to the neighborhood of the point *i* and the β's are positive parameters whose optimum values can change with the nature of the measured sample. See reference [5] for a detailed explanation.

2 Algorithm

The numerical method implemented by MUPen2D is an iterative procedure where, at each iteration, suitable values for the λ_i's and α are determined according to (3) and (4).

Let $\sigma_1^{(1)}$ and $\sigma_1^{(2)}$ be the maximum eigenvalues of the matrices $\mathbf{K_1}$ and $\mathbf{K_2}$ respectively, the steps of the MUPen2D algorithm can be outlined as follows:

Let $p_\mu^{(k)}$ and $c_\mu^{(k)}$ denote the values of the gradient and Laplacian of the reconstructed map $\mathbf{f}^{(k)}$ at index μ, then the steps of the MUpen2D algorithm can be summarized as follows:

1- Compute the SVDs of the matrices $\mathbf{K_1}$ and $\mathbf{K_2}$;

2- Compute $\hat{\mathbf{s}}$ the SVD projection of **s**;

3- Set k = 0. Using the tolerance parameter Tol$_{GP}$ and running few iterations of the Gradient Projection method, compute a starting guess $\mathbf{f}^{(0)}$ for the distribution **f**:

$$\mathbf{f}^{(0)} = \underset{\mathbf{f}\geq\ell}{\operatorname{argmin}} \| (\mathbf{K}_2 \otimes \mathbf{K}_1)\mathbf{f} - \hat{\mathbf{s}} \|_2^2$$

4- repeat

a) compute

$$\lambda_i^{(k)} = \frac{\| (\mathbf{K}_2 \otimes \mathbf{K}_1)\mathbf{f}^{(k)} - \hat{\mathbf{s}} \|_2^2}{(N+1)\left(\beta_0 + \beta_p \max_{\mu\in I_i} (p_\mu^{(k)})^2 + \beta_c \max_{\mu\in I_i} (c_\mu^{(k)})^2\right)} \qquad i = 1,\dots,N$$

$$\alpha^{(k)} = \frac{\left\|\mathbf{K}\mathbf{f}^{(k)}\text{-}\mathbf{s}\right\|^2}{(N+1)\left\|\mathbf{f}^{(k)}\right\|_1}$$

b) using the FISTA method with the tolerance parameters Tol$_{FISTA}$ and constant stepsize $\xi^{(k)} = \left(\sigma_1^{(1)}, \sigma_1^{(2)}\right)^2 + 64 \max_i \left|\lambda_i^{(k)}\right|$, compute

$$\mathbf{f}^{(k+1)} = \underset{\mathrm{f}}{\operatorname{argmin}} \left\{ \left\| \begin{pmatrix} \mathbf{K} \\ \sqrt{\omega_1 \mathbf{\Lambda}^{(k)}}\mathbf{L} \end{pmatrix} \mathbf{f}^{(k)} \text{-} \begin{pmatrix} \mathbf{s} \\ 0 \end{pmatrix} \right\|^2 + \omega_2\, \alpha^{(k)} \left\|\mathbf{f}^{(k)}\right\|_1 \right\}$$

and

residual$^{(k)}$ as the difference between computed and measure data

until $\| \mathbf{f}^{(k+1)} - \mathbf{f}^{(k)} \| \leq Tol_{UPEN} \| \mathbf{f}^{(k)} \|$

5- Save computed distribution on file

The algorithm depends on several parameters that can be grouped as follows:

• Parameters: β_{00}, β_0, β_p, β_c used to control the proper choice of the local regularization parameters λ_i. See references [1, 3, 4, 5] for a detailed discussion.

• Tolerance parameters used to stop the iterative methods: Tol$_{CG}$, Tol$_{FISTA}$, Tol$_{UPEN}$. The typical values, that have proven to work properly for many kinds of different set of data, are reported in table 1.

To avoid possibly infinity loops, maximum possible iteration numbers are used.

Table 1- Typical default Tolerance parameter values.

Tol$_{CG}$	Tol$_{FISTA}$	Tol$_{UPEN}$
1.0E-4	1.0E-8	1.0E-5

3 Installation and Program description

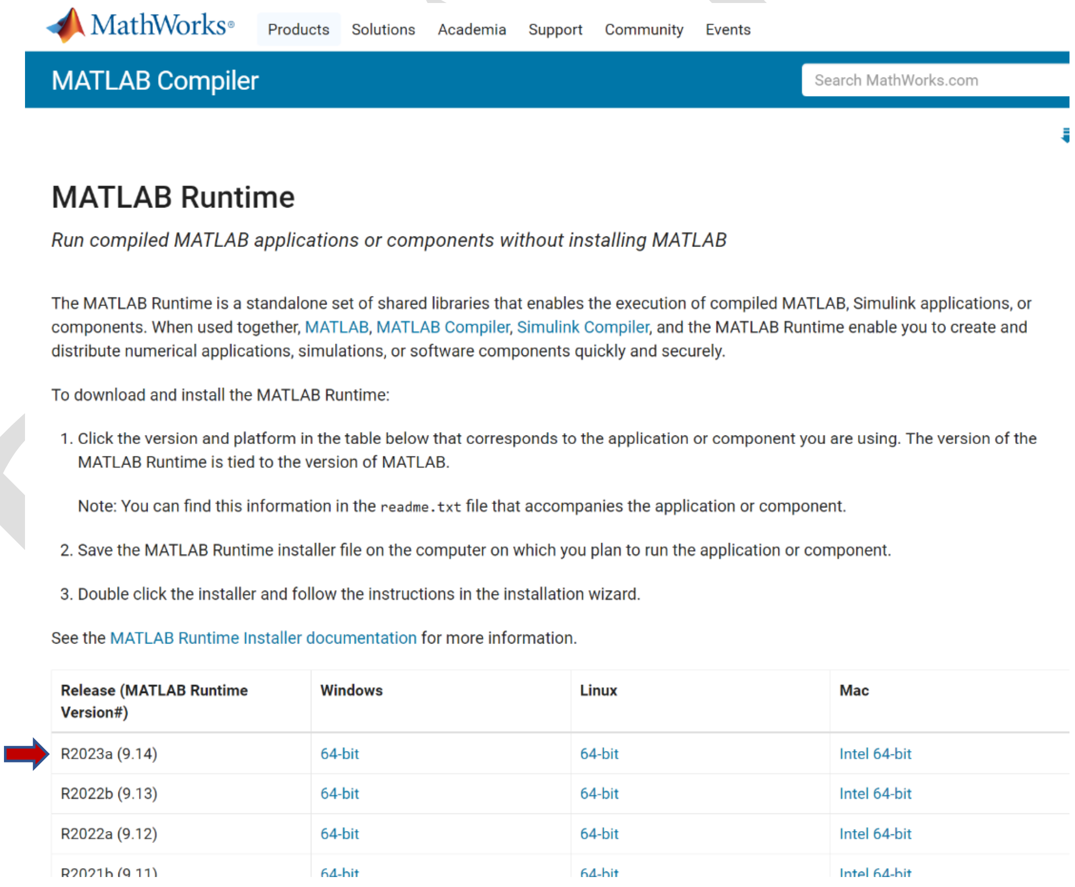
These notes refer to the command line standalone version of MUPen2D. Anyway, the furnished information can be easily used also to works with the GUI (Graphical User Interface) version.

3.1 Installation

This SW was written under Matlab 2023a, but it does not need to have Matlab installed on the user PC. MUpen2D is distributed as both a set of scripts and a freeware portable software, therefore a standalone software, that requires the presence of the appropriate free shared Matlab libraries. For this Mupen2D release the runtime library is named R2023a (9.14). At present there are compiled versions for Windows OS (tested under Windows 10) and Linux (tested under Ubuntu 20.04). Under Windows OS, the installation consists of two steps: 1) install the free Matlab runtime library and 2) unzip the zipped folder with the MUpen2D set of files and subfolders. In the following, the procedure for Windows is described. Follow the link:

"https://www.mathworks.com/products/compiler/matlab-runtime.html"

and download the file "R2023a" (see red arrow below in Fig.1).



MATLAB Runtime

Run compiled MATLAB applications or components without installing MATLAB

The MATLAB Runtime is a standalone set of shared libraries that enables the execution of compiled MATLAB, Simulink applications, or components. When used together, MATLAB, MATLAB Compiler, Simulink Compiler, and the MATLAB Runtime enable you to create and distribute numerical applications, simulations, or software components quickly and securely.

To download and install the MATLAB Runtime:

1. Click the version and platform in the table below that corresponds to the application or component you are using. The version of the MATLAB Runtime is tied to the version of MATLAB.

   Note: You can find this information in the `readme.txt` file that accompanies the application or component.

2. Save the MATLAB Runtime installer file on the computer on which you plan to run the application or component.

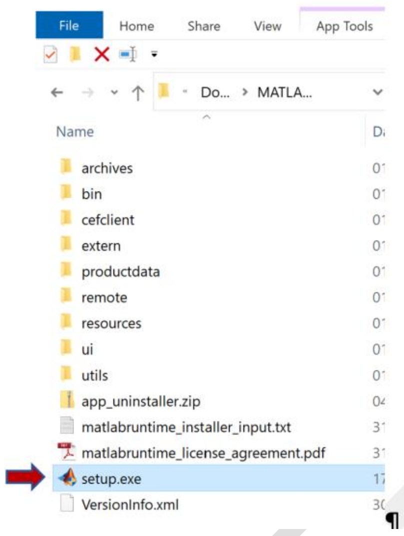
3. Double click the installer and follow the instructions in the installation wizard.

See the MATLAB Runtime Installer documentation for more information.

Release (MATLAB Runtime Version#)	Windows	Linux	Mac
R2023a (9.14)	64-bit	64-bit	Intel 64-bit
R2022b (9.13)	64-bit	64-bit	Intel 64-bit
R2022a (9.12)	64-bit	64-bit	Intel 64-bit
R2021b (9.11)	64-bit	64-bit	Intel 64-bit

Figure 1-Mathworks webpage where to download the Matlab runtime library.

After a while, you will have downloaded a file with a name similar to the following "MATLAB_Runtime_R2023a_Update_1_win64.zip". It might take some time because the file is very big, about 3GB. Once it has been downloaded, unzip the folder and click on the executable file “setup.exe” (see the red arrow in Fig.2) to install the Matlab library.



archives
bin
cefclient
extern
productdata
remote
resources
ui
utils
app_uninstaller.zip
matlabruntime_installer_input.txt
matlabruntime_license_agreement.pdf
setup.exe
VersionInfo.xml

Figure 2 – Matlab runtime library installation.

After having installed the Matlab library, unzip the zipped distributed SW “MUpen2D.zip” and open the folder “MUpen2D” containing the SW and click on the portable command line executable file ‘MUpen2D.exe’ to run the software (or 'GUI_MUpen2D.exe' if you want to run the GUI version). MUpen2D is a command line program that computes 2D multi-exponential inversion of data that must be stored in an ASCII format file.

After a few seconds, required for the SW initialization, the interface folder, where to choose the data to be processed, shown in Fig.3 will be displayed.

The MUpen2D package consists of a folder, named “MUpen2D”, which contains the following folders and files:

• “MUPen2D.exe” is the standalone command line program version. Double-click to launch the application;

• “GUI_MUPen2D.exe” is the standalone GUI program version. Double-click to launch the application;

• “DATA” a folder used to store the input data files and parameter files with the parameters used to drive MUPen2D during the inversion process (see next paragraph for a detailed description of these files). The DATA folder comes with three examples whose data are stored in the “T1-T2”, “T2-T2” and “D-T2” subfolders;

• “output files” a folder that contains the following output files created runtime by the computation:

- “2D_Distribution.txt”, the computed 2D map of relaxation times;
- “Parameters.txt”, with the relevant output parameters such as computation time, iteration numbers, value of the output residual norm;
- “T1.txt” and “T2.txt, with the values of the vectors T_1 and T_2 in eq. (2);
- “t1.txt” and “t2.txt”, with the values of the vectors t_1 and t_2 in (2);

- “Residual.txt”, with the residual of the last computed 2D map is saved in the file.

• “DOC” folder with the manuals;

• “splash.png” the splash screen image (the image that appears while the program is launching).

Under Linux the installation follows almost the same steps as under Windows, although it is, as usual, more complex and can depend on the different distribution/version of Linux.

Anyway, the compiler creates a script (in this specific case named run_GUI_MUPEN2D.sh for the GUI version) that launches the executable under Linux. To do this it is necessary to run the following terminal command:

```
“sudo ./run_GUI_MUPEN2D.sh /usr/local/MATLAB/MATLAB_Runtime/v99”
```

where “/usr/local/MATLAB/MATLAB_Runtime/v99” is the location where has been installed the runtime library of Matlab 2023a.

Instead, to run the command line version, use the following command:

```
“sudo ./run_MUpen2DTool.sh /usr/local/MATLAB/MATLAB_Runtime/v99”
```

Command line version run and data loading

To run the program double click on the file “MUpen2D.exe” contained in the MUpen2D folder. Once the program is launched, an input data dialog window (“Open Data Directory”) will appear asking for the directory where the datafiles and parameter files are stored. The input dialog is shown in Figure 3.

To select data and parameter files select the desired folder present in DATA by clicking on the “Folder Selection” button. Then, the data and parameters inside the DATA folder are automatically loaded by the program and the computation starts.

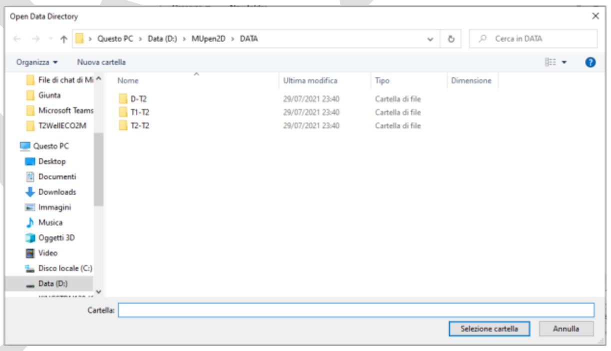


Figure 3- Input dialog to select the data directory. Single-click to select the sub-folder of DATA with the data to be processed.

A message dialog box (see Figure 4) will appear and will remain on until the computation will be finished.

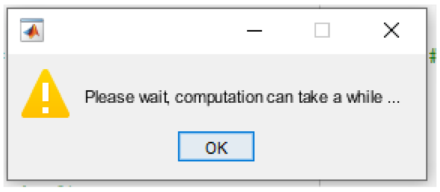

Please wait, computation can take a while ...

Figure 4- Message dialog box.

3.2 Input data, parameter files and keywords

A data folder must contain 6 files: three data files and three parameter files.

Data files

- a column ASCII data files with the list of the M time parameter used to acquire the relaxation curve related to the first dimension. Usually, the first dimension is the Inversion times of an IR sequence or the echo times of a CPMG sequence or the delta time parameters used in the D-T_2 experiment;
- a column ASCII data files with the list of the N time parameter used to acquire the relaxation curve of the second dimension. Usually, it contains the Echo times of a CPMG sequence;
- a 2D M×N matrix data file with the acquired signal. For example, in case of an IR-CPMG acquisition, the M rows of the matrix are the CPMGs acquired with N echoes at M different IR inversion times.

Parameter files with the keywords

The three (3) parameter files contain the keywords that drive the inversion computation. The files are in "fixed format", therefore the position (the column of the "=" symbol) of the value of keyword cannot be changed. Keyword value in a wrong position crashes the program. Rows that do not contain a keyword are interpreted as comments and ignored.

Changing the value of a parameter can improve or worsen the quality of the distribution map computed and can also substantially increase/decrease the computation time in an unpredictable manner.

This because of the parameter changes have also effects on the matrix structure involved in the inversion procedure.

- **FileFlag.par**, contains a list of keywords that modify the functionality of MUpen2D:
  - -FL_typeKernel, 1, 2, 3 or 4. 1 to select the T_1-T_2 kernel model for IR-CPMG experiment 2 for a SR-CPMG experimen, 3 to select the D-T_2 kernel model or 4 for the T_2-T_2 kernel model;
  - -FL_InversionTimeLimits, 0 or 1. If the value is 1 then MUpen2D automatically computes the left and right extreme limits of the relaxation times ($T_1(1)$, $T_1(M)$, $T_2(1)$ and $T_2(N)$) used in the inversion process using 4 times the last measured data times ($t_1(M)$, $t_2(N)$) and ¼ of the first measured data times ($t_1(1)$, $t_2(1)$), respectively. Otherwise if the value is 0, then MUpen2D will use the limits as reported in the FileSetInput.par file (see below);
  - - FL_OutputData, 0 or 1. 1 to create output files;
  - - FL_NoContour, Number of Contour lines to obtain 2D contour maps. Set 0 to obtain the 2D map without contour lines;
  - - FL_Verbose, 0 or 1. 1 to display additional information;

- FL_Debug, 0 or 1. 1 to add further output, useful for diagnostic analysis of the code, in particular on the convergence process of the algorithm FISTA. Its activation can have effect on the computation time and on the quality/shape of the computed map;
- FL_Lower_bound, lower bound to enable possibility to have negative distribution. It impacts on Projected Gradient algorithm and FISTA as well;
- FL_Amp_scale, multiplication factor of the 2D data (default value =1);
- FL_Scale_fact, multiplication factor of the 2D data (default value=1);
- FL_T1T2Filter, 0 or 1. If 1 the filter deletes in the 2DNMR T1-T2 maps the points where $T_2 > 1.5\ T_1$ to exclude nonphysical part of the computed NMR map (default value =0). ;
- FL_EraseCol, exclude from the computation the first "FL_EraseCol" columns of the 2D data matrix (default value =0);
- FL_EraseRow, exclude from the computation the first "FL_EraseRow" rows of the 2D data matrix (default value =0);
- FL_Offset, enable the regression also of a constant term (0 or 1, default value =0);
- FL_Stat, enable computation of postprocessing statistics performed on the residual matrix, the difference between measured and reconstructed data. 0 to disable statistics, a positive value is used as threshold to highlight residual differences;

- **FilePar.par**
  - par.gpnr.tol, convergence tolerance of the Projected Gradient algorithm;
  - par.gpnr.maxiter, maximum number of iterations of the Projected Gradient algorithm;
  - par.nwtp.maxiter, maximum number of iterations of the Projected Newton algorithm (not used in this release version);
  - par.nwtp.tolrho, convergence tolerance of the Projected Newton algorithm (not used in this release version);
  - par.cgn2d.tol, convergence tolerance of the Conjugate Gradient algorithm (not used in this release version);
  - par.cgn2d.maxiter, maximum number of iterations of the Conjugate Gradient algorithm (not used in this release version);
  - par.svd.svd, 0 or 1. 1 enable the use of the SVD algorithm to reduce the size of the problem;
  - par.svd.soglia, threshold of the SVD algorithm (minimum amplitude of the accepted singular value), it can have effects on the quality of the computed map;
  - par.upen.tol, convergence tolerance used by UPEN algorithm (typical default value order of 10^{-5});
  - par.upen.iter, maximum number of iterations of the UPEN algorithm;

- par.upen.beta00, is a scale parameter that depends on the specific material, see references [1], [4] and [5] for a detailed discussion;
- par.upen.beta0, is a compliance floor that prevents division by zero, it should be small enough to prevent undersmoothing, and large enough to avoid oversmoothing (typical default value 10^{-7});
- par.upen.beta_p, the compliance parameter for the local slope smoothing of the 2D distribution;
- par.upen.beta_c, the compliance parameter for the local curvature smoothing of the 2D distribution;
- par.fista.maxiter, maximum number of iterations of the FISTA algorithm;
- par.fista.tol, convergence tolerance used by FISTA algorithm;
- par.fista.weight, a weighting parameters used to balance the weight of the two regularization methods (L2 and L1) during the regression process. A value >1 implies no weighting is used (default value 2);

- **FileSetInput.par**

-Filenamedata, the name of the 2D data files;

-filenameTimeX, the filename of vectors of times of the first dimension;

-filenameTimeY, the filename of vector of times of the second dimension;

-nx, number of relaxations time bins along the first dimension used by MUpen2D, increasing this number will smooth the plotted curve but at the same time it will increase the computation time;

-ny, number of relaxations time bins along the second dimension used by MUpen2D; increasing this number will smooth the plotted curve but at the same time it will increase the computation time;

-T1min, minimum inversion time limit for the second inversion dimension, used if FL_InversionTimeLimits = 0;

-T1max, maximum inversion time limit for the second dimension, used if FL_InversionTimeLimits = 0;

-T2min, minimum inversion time limit for the first inversion dimension, used if FL_InversionTimeLimits = 0;

-T2max, maximum inversion time limit for the first inversion dimension, used if FL_InversionTimeLimits = 0.

It is worth noting that the ranges determined by the limits can have strong effect on the computed map, indeed they act on the matrix used by the inversion algorithm.

3.3 Output files

When the computation terminates, many files are saved in the folder “\DATA\output_files\”. A few of them are saved only if the “FL_verbose” flag is on. The 2D matrix of the computed distribution is automatically saved in an ASCII file with the name “2D_Distribution.txt”. The files with the coordinates (relaxation times) of the distribution in the two dimensions are saved under the folder “\DATA\output_files\” with the names “T1.txt” and “T2.txt.

Also, the times of the acquired data are saved, “t_1.txt” and “t_2.txt”. At last, the list of parameters used, and the positions coordinates of peaks are saved in the file “Parameters.txt” and the residual of the last computed 2D map is saved in the file “Residual.txt”. In addition, at the end a number of plots are shown (they can be saved as “.fig” files as well as “.jpg” files). In particular, the software shows: the projection of the 2D map along the two dimensions (Fig. 3a and Fig. 3b), the 2D distribution map (Fig. 3c) and the 3D distribution map (Fig. 3d).

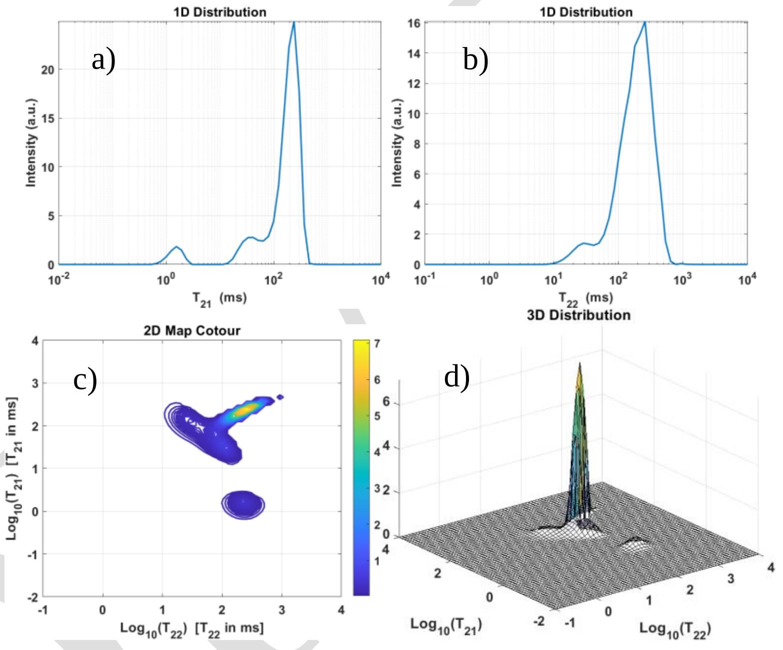


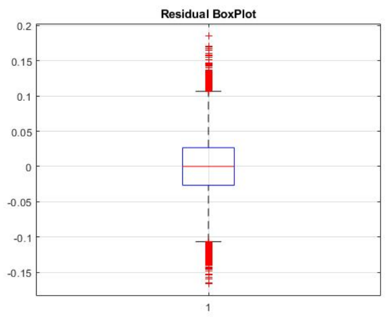
Figure 3 - Pictures of the plots automatically created by MUpen2D.

4 STATISTICS

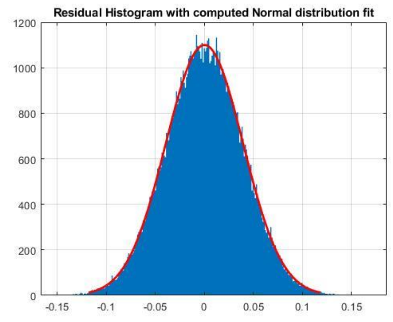
If the parameter flag FL_Stat is enabled, then some statistics on residuals (calculated as: Data-ComputedData) are computed, and the corresponding plots are shown.

Example of plot with a brief explanation are reported below.

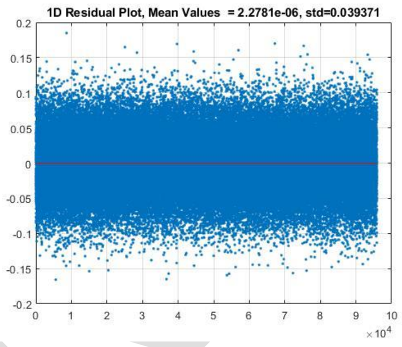
a) The residual Box, where on each box, the central mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers. The outliers are plotted individually using the '+' symbol.



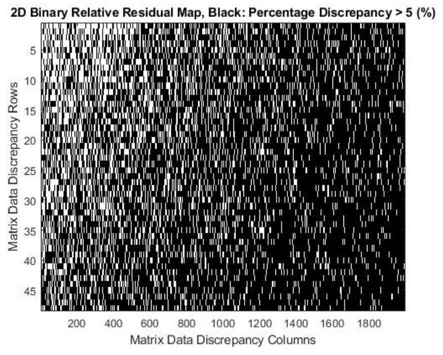
b) The histogram of residuals with the corresponding Normal distribution (red curve) computed using the Matlab function “fitdist()”.



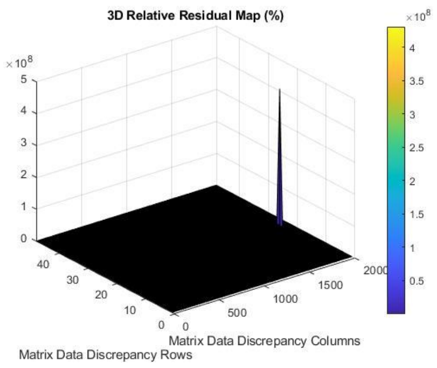
c) The residual 1D plot (array representation) with the arithmetic mean and sample standard deviation computed on the data.



d) In the next figure, the Binary plot of relative residuals (calculated as: 100*(Data-ComputedData)/Data) bigger than a threshold inputted by the user using the value assigned to the flag FL_Stat.

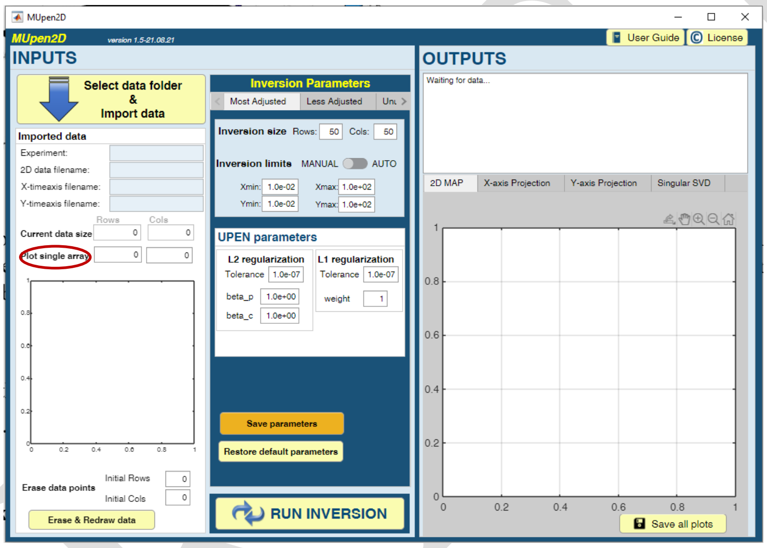


e) A 3D plot with the relative residues in percent, element-by-element (element-wise) computed as: abs(100*(Data-ComputedData)/Data).



5 GUI INTERFACE

The software comes also with an easy-to-use GUI interface, see figure below, wherewith the user can test the effects on the computed map resulting from parameter changes. The parameter meanings are the same as described in the previous paragraphs.



Current data size 0 0

Plot single array 0 0

Erase data points Initial Rows 0 Initial Cols 0

Inversion size Rows: 50 Cols: 50

Inversion limits MANUAL AUTO

Xmin: 1.0e-02 Xmax: 1.0e+02

Ymin: 1.0e-02 Ymax: 1.0e+02

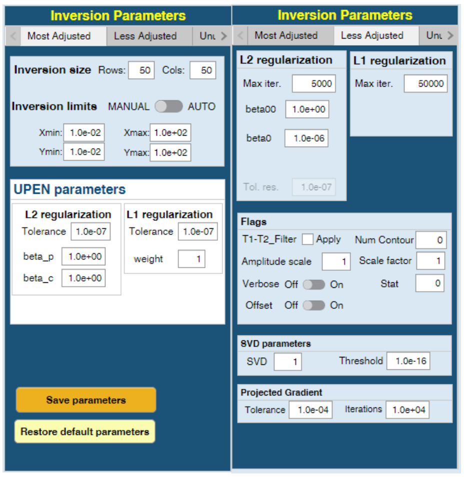
L2 regularization: Tolerance 1.0e-07, beta_p 1.0e+00, beta_c 1.0e+00

L1 regularization: Tolerance 1.0e-07, weight 1

Waiting for data...

Parameter changes only have a local effect. Use the button “Save parameters” to make them permanent by saving the values shown in the interface in the parameter files.

The parameters are subdivided between two “Tab widgets” (see figure below). The is also a third Tab with a list of unused inactive parameters left over for internal compatibility.



Inversion size Rows: 50 Cols: 50

Inversion limits MANUAL AUTO

Xmin: 1.0e-02 Xmax: 1.0e+02

Ymin: 1.0e-02 Ymax: 1.0e+02

L2 regularization: Tolerance 1.0e-07, beta_p 1.0e+00, beta_c 1.0e+00

L1 regularization: Tolerance 1.0e-07, weight 1

L2 regularization: Max iter. 5000, beta00 1.0e+00, beta0 1.0e-06, Tol. res. 1.0e-07

L1 regularization: Max iter. 50000

Flags: T1-T2_Filter Apply, Num Contour 0, Amplitude scale 1, Scale factor 1, Verbose Off On, Stat 0, Offset Off On

SVD parameters: SVD 1, Threshold 1.0e-16

Projected Gradient: Tolerance 1.0e-04, Iterations 1.0e+04

Using the numeric edit field widget named "Plot single array", it is possible to extract and automatically show a column or a row of the data matrix. This is a useful functionality to have a look to the quality of the acquired data.

6 EXAMPLES

In the DATA folder, there are three examples, one inside the sub-folder "T1-T2", a second one inside the subfolder "T2-T2" and a third in the subfolder "T1_T2_Synth".

5.1 Example1 folder "T1-T2"

In this example a sample of fresh cement paste was measured with an IR-CPMG 2D sequence. The number of the inversion points is 48, computed in geometric scale with a common ratio of 1.1896 in the range between 0.8 ms to 2800 ms. During the detection period, the TE was of 108 μs and the number of echoes was 1000. The repetition time was 3 s and number of scans 4.
In Figure 4 the plots computed by MUpen2D.

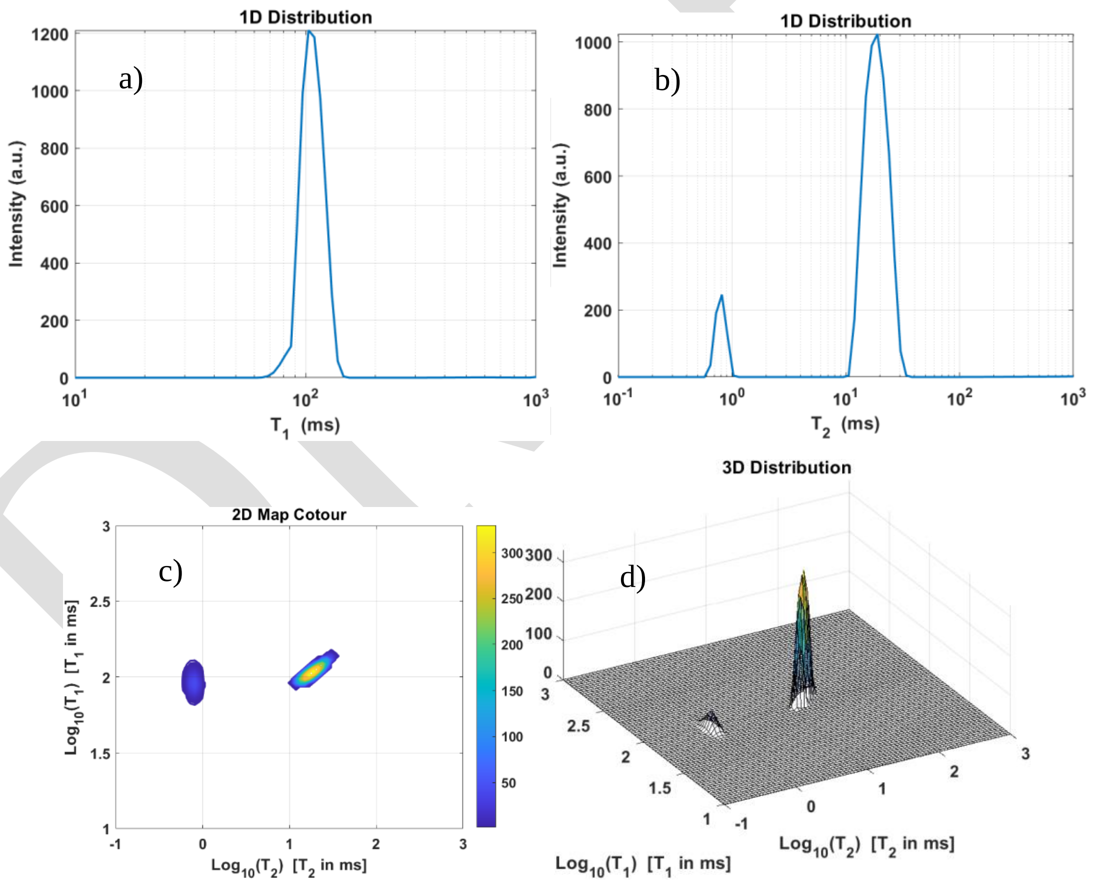


Figure 4 – "T1-T2" experiment. Projection along the T_1 (a) and T_2 (b) dimension, (c) Reconstructed 2D Contour relaxation map, (d) 3D map.

5.2 Example2 folder "T2-T2"

In this example a sample of Maastricht stone saturated with water was measured with a CPMG-CPMG 2D sequence. During the preparation period, the echo time (TE) was 6.61 ms and the number of echoes was 128. The mixing time lasts 100 s. During the detection period, the TE was of 300 μs and the number of echoes was 2800. The repetition time was 3 s and number of scans 16. The output files plots are those reported in the Figure 3 of the paragraph “3.3 Output files”.

5.3 Example3 folder “T1-T2_Synth”

This is a T1-T2 synthetic case. We consider a reference map distribution of size 64x64 and use it to synthesize the simulated IR-CPMG data sequence with 128×2048 non uniformly distributed times in the interval [10^{-3}, 3 10^{3}] ms. The reference map distribution has the following three peaks:

	Height	T_1 (ms)	T_2 (ms)
P1	184.07	807.91	24.95
P2	48.87	572.83	24.95
P3	227.55	6.56	2.48

The data noisy is obtained by adding Gaussian white noise of level 10^{-2} (SNR 20 db) to the synthetic data. The peaks positions and height, computed by 2DUpenTool, are reported in the following table:

	Height	T_1 (ms)	T_2 (ms)
P1	53.30	839.56	21.96
P2	24.00	146.07	16.41
P3	82.87	6.85	2.47

The Relative Error (Err) and Root Mean Squared Error (RMSE) have the following values:
Err = 0.113, RMSE= 1.720. The computed plots are shown in Figure 5.

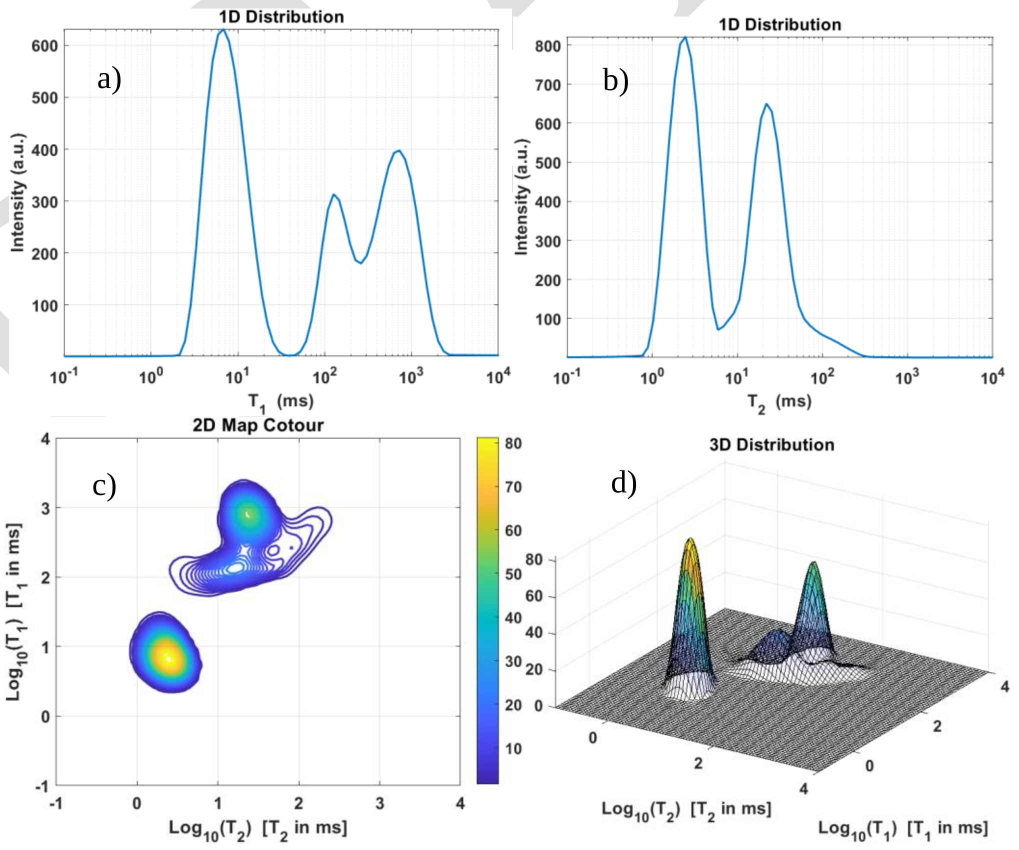


Figure 5 - “T1-T2_synth” example. Projection along the (a) T_1 and (b) T_2 computed relaxation time distribution, (c) Reconstructed 2D Contour relaxation map, (d) 3D map.

7 REFERENCES

[1] V. Bortolotti, R. J. S. Brown, P. Fantazzini, G. Landi, F. Zama, Inverse Problems 33 (1), 2016.

[2] B. Blüemich, Essential NMR, Springer-Verlag, 2005.

[3] V. Bortolotti, L. Brizi, P. Fantazzini, G. Landi, F. Zama, Filtering techniques for efficient inversion of two-dimensional nuclear magnetic resonance data, Journal of Physics: Conference Series 904 (1). 2017.

[4] V. Bortolotti, R.J.S. Brown, P. Fantazzini, G. Landi, F. Zama, UPEN2D: Improved 2DUPEN algorithm for inversion of two-dimensional NMR data, Microporous and Mesoporous Materials 269 195 – 198, 2018.

[5] V. Bortolotti, G. Landi, and F. Zama. 2DNMR data inversion using locally adapted multi-penalty regularization. Computational Geosciences, 25:1215 – 1228, 2021.