2D LAPLACE INVERSION INSTRUCTION MANUAL

Sophie Godefroy, Brett Ryland and P. T. Callaghan
Victoria University of Wellington, Wellington, New Zealand

This 2 dimensional so-called Laplace inversion is a program written at Victoria University of Wellington (Wellington, New Zealand) to process 2 dimensional ASCII data measuring in the two directions either diffusion or relaxation characteristics of heterogeneous proton systems. The program interface is written in Matlab whereas the code section that performs the bulk of the computation is written in C, making use of the Matlab "mex" external interface. The program requires at least Matlab 6 to be run, and the instructions for the first use are provided at the beginning of the third section of this instruction manual. After a brief introduction of the theory, we will present the algorithm used in the program. We will then describe the user interface and how to use the program. At last, some examples of computed data will presented.

1 GENERAL THEORY

For heterogeneous samples, we expect to measure multi-exponential decay of the signal, for the relaxation measurements as well as for the diffusion measurements. The processing of the data is performed by using a Laplace inversion written in Matlab, in 1 dimension for the processing of the relaxation and diffusion measurements, and in 2 dimensions for the diffusion and / or relaxation 2D experiments. In 1D, if we consider for example relaxation data, the multi-exponential signal *S(t)* can be written as:

$$S(t_i) = g_i = \sum_k A(T_k) \exp\left(-\frac{t_i}{T_k}\right) + \varepsilon_i \qquad [1]$$

In this expression, T_k and $A(T_k)$ correspond respectively to the relaxation times and the relaxation time distribution of the system under study. ε represents the noise of the signal.

The general procedure to inverse the data and determine the distribution of relaxation times *A(T)* consists of applying the non-negative least square algorithm, as described in the Lawson & Hanson book[1]. This would consist of the minimization of:

$$\min\left\{\chi^2 = \sum_{j=1}^{n} \left(g_j - \sum A(T_i) \exp(-t_j / T_i)\right)^2\right\} \qquad [2]$$

However, because the signal is noisy, the determination of the distribution by a simple non-negative least square fit would lead to a multitude of peaks. The problem is mathematically ill-posed. To avoid this problem, we add a so-called regularization

function (or smoothing function), weighted by the smoothing parameter α[2,3,4,5]. The new least square function to minimize is displayed in the following equation.

$$\min\left\{\chi^2 = \sum_{j=1}^{n}\left(g_j - \sum A(T_i)\exp(-t_j/T_i)\right)^2 + \alpha^{-1}\sum_{i=1}^{m}\left(2A(T_i) - A(T_{i-1}) - A(T_{i+1})\right)^2\right\} \quad [3]$$

The program written to transform the data works on the matrix form of equation [3] such as:

$$Y = KX + E \quad [4]$$

where X is the distribution to be determined, Y is the signal measured, K is the known matrix of the exp($-t_i/T_k$) (with t_i the time and T_k the relaxation times fixed for the processing). The least squares to minimize then has the form:

$$\chi^2 = \|KX + E\|^2 + \alpha^{-1}\|X''\|^2 \quad [5]$$

In 2 dimensions, for example for a relaxation / relaxation correlation, the signal measured follows the equation:

$$s(t_1,t_2) = \sum \exp(-t_1/T_1).\exp(-t_2/T_2)A(T_1,T_2) + E(t_1,t_2) \quad [6]$$

In the matrix form, the signal will be written as:

$$Y = K_1 X K_2' + E \quad [7]$$

where K_1 and K_2 are the known matrices of $exp(-t_{1i}/T_{1k})$ and $exp(-t_{2i}/T_{2k})$ respectively. To get the 2D Laplace inversion of the 2D data defined by Y, the idea is to transform the matrix Y (of dimension $m \times n$) into a vector of length $m \times n$[6]:

$$y = Kx + e,\ y = [Y]_D,\ x = [X]_D,\ K = K_1 \otimes K_2 \quad [8]$$

The problem occurring from this transformation is the huge size of the vector y and the matrix K. The solution is to reduce the size of the matrices before the 2D -> 1D transformation by using the singular value decomposition algorithm on the matrices such as[6]:

$$K_1 = U_1 S_1 V_1';\ K_2 = U_2 S_2 V_2' \quad [9]$$

In these expressions, U and V are unitary matrices. After determining the reduced matrices according to:

$$\widetilde{K}_1 = U_1' K_1;\ \widetilde{K}_2 = U_2' K_2;\ \widetilde{Y} = U_1' Y U_2; \quad [10]$$

equation [7] can be written in a reduced form such as:

$$\widetilde{Y} = \widetilde{K}_1 X \widetilde{K}_2' + E \qquad [11]$$

We can now apply the 2D -> 1D transformation on the reduced data:

$$\widetilde{y} = \widetilde{K}x + e,\ \widetilde{y} = \left[\widetilde{Y}\right]_D,\ x = \left[X\right]_D,\ \widetilde{K} = \widetilde{K}_1 \otimes \widetilde{K}_2 \qquad [12]$$

and process the data according to the Laplace inversion presented in the equations [4] and [5].
For the diffusion experiments, the multi-exponential decay signal (in 1D) can be written as:

$$g_j = s(t_j) = \sum A(D_i) \exp(-(\gamma\delta G)^2(\Delta - \delta/3)D_i) + \varepsilon_i \qquad [13].$$

To process the diffusion data, we use the same procedure as for the relaxation data. But to get the real diffusion distribution, we flip the 1D or 2D diffusion distribution matrix in the direction of the diffusion experiment.

2 ALGORITHMS

Below are given the general algorithms of the Non-Negative Least Square (NNLS) fit[7] (Table 1) and of the Singular Value Decomposition (SVD) (Table 2).

1.Set P = NULL, Z ={1,2,…,n}, x=0
2.Compute the n-vector $w=E^T(f-Ex)$
3.If the set Z is empty or if $w_j \leq 0$ for all $j \in Z$, goto 12
4.Find an index $t \in Z$ such that $w_t = \max\{w_j, j \in Z\}$
5.Move the index t from the set Z to set P
6.Let E_p denote the m*n matrix defined by "column j of E_p={col. j of E if $j \in P$;0 if $j \in Z$}. Compute the n-vector z as a solution of the least squares problem $E_P Z \approx f$. Only the components Z_j, $j \in P$ are determined by this problem. Define $Z_j=0$ for $j \in Z$.7.If $Z_j>0$ for all $j \in P$, set x=Z and goto 2
8.Find an index $q \in P$ such that $x_q/(x_q-Z_q) = \min\{x_j/(x_j-Z_j): Z_j \leq 0, j \in P\}$
9.Set $a= x_q/(x_q-Z_q)$
10. Set x=x+ a(Z-x)
11. Move from set P to set Z all indices $j \in P$ for which $x_j=0$. Goto step 6
12. Comment: the computation is finished

Table 1. NNLS fit algorithm

•Singular Value decomposition of a matrix $\mathcal{A}$ (m×n):
1) Find the eigenvalues of the matrix $\mathcal{A}^T\mathcal{A}$ and arrange them in descending order
2) Find the number of non zero eigenvalues of the matrix $\mathcal{A}^T\mathcal{A}$.
3) Find the orthogonal eigenvectors of the matrix $\mathcal{A}^T\mathcal{A}$ corresponding to the obtained eigenvalues, and

arrange them in the same order to form the column-vectors of the matrix V (n×n). 4) Form the diagonal matrix S (m×n) placing on the leading diagonal of it the square roots $\sigma_i=\text{sqrt}(\lambda_i)$ of p=min{m,n} first eigenvalues of the matrix $\mathcal{A}^T\mathcal{A}$ got in 1) in descending order. 5) Find the first column-vector of the matrix U (m×n) : $u_i = s_i \mathcal{A} v_i$ (I=1:r) 6) Add to the matrix U the rest of m-r vectors using the Gram-Schmidt orthogonalization process.

Table 2. SVD algorithm

3 DESCRIPTION OF THE PROGRAM

This 2 dimensional Laplace inversion is written in Matlab, and requires at least Matlab 6 to be run. It computes 2 dimensional multi-exponential data according to the theoretical description and algorithms presented in the sessions above. The data must be in an ASCII format. The section of code that performs the bulk of the computation has been written in C to reduce computation time, and makes use of the "mex" external interface to Matlab. The mex interface translates the Matlab input and output matrices into C-arrays, which can then be processed in subroutines written purely in C. The C-code is compiled in Matlab via the "mex" command into a file of the same name with a platform-dependant extension (.dll in Windows). Typically, a compiler needs to be selected the first time mex is used on a system, this can be done simply via the "mex -setup" command. More information can be found by calling "help mex" at the Matlab command prompt. The 2D Laplace Inversion program (TwoDLaplaceInverse.m) has to be run form matlab.

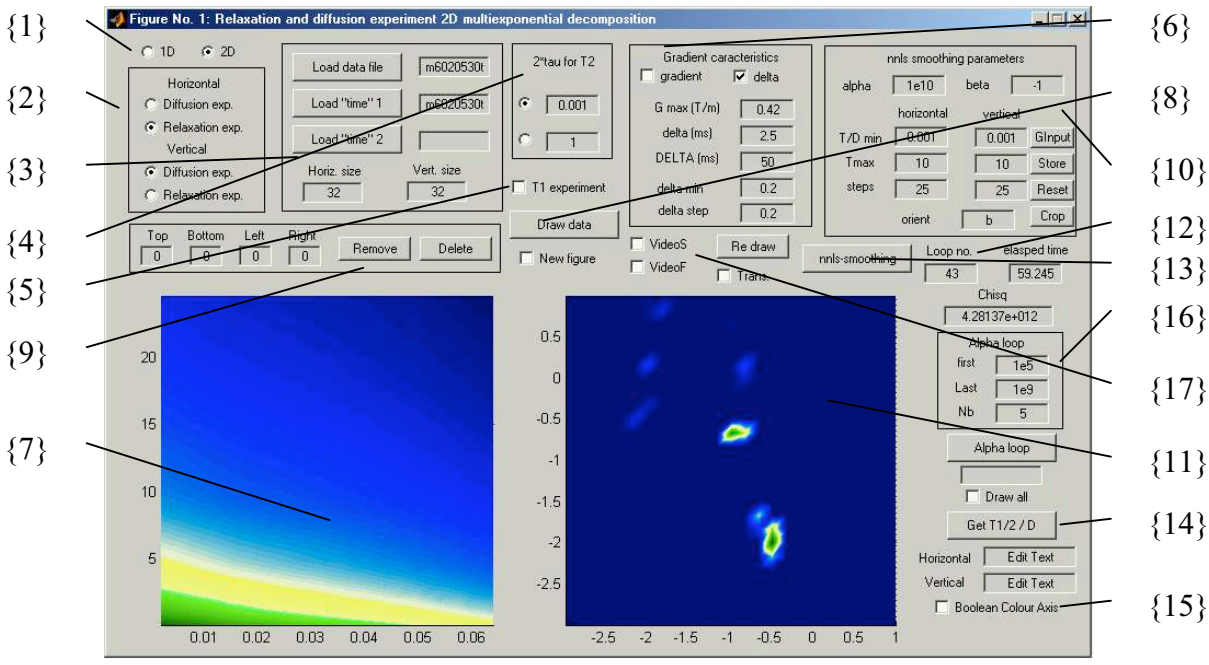

Figure 1. Interface of the Matlab program of the 2D Laplace inversion.

Data loading:

The interface of the 2D Laplace inversion program written in Matlab is presented in Figure 1 above. The program can compute both 1D and 2D relaxation decay curves, as must be indicated in the 1D / 2D choice button of the top left corner {1}. Both the relaxation or diffusion experiments can also be computed as must also be indicated in the choice buttons for the horizontal direction only for a 1D experiment or horizontal and vertical directions for a 2D experiment {2}. The data file and, when relaxation experiments are selected, the characteristic "times" of the experiments (such as the inversion-recovery times, lists, or any other information describing the "coordinates" of the experiments) are entered via the "load" button {3}. In case of a list (for example for a CPMG experiment), a scale factor can be entered in the box on the right of the data-loading box after selection of the button {4}. Of course, for a 1D experiment, only the "time" 1 is entered. For a 1D spin-lattice relaxation time experiment (T_1) the "T_1 experiment" check box has to be selected {5}. In case the diffusion experiments choices are selected in one or two dimensions, the parameters used for the diffusion experiments are to be entered in the "Gradient Characteristics" box in order to get the q^2 coordinates of the experiment {6}. The program allows the determination of q^2 for PGSE experiments with either variation of the amplitude or of the duration of the gradients pulses. In case the gradient variation experiment box is checked, one has to enter the values of the maximum amplitude of the gradient, its duration and the delay between the pulsed gradients. In case the box of variation of delta is checked, one has to enter the value of the fixed amplitude of the gradient, of the duration between the two pulsed gradient (DELTA) and the minimum value as well as the increment of the duration of the pulsed gradients {6}. The q^2 determined by the program consider a linear variation of the gradient amplitude or duration. For any other variation, the q^2 of the experiment will have to be calculated by the user and entered in the direction of the diffusion experiment as a "time file" after checking the choice "relaxation" in the box {2}. The 1D or 2D curve of the signal decay is displayed on the left graph {7} by clicking on the "draw data" button {8}. For example, the one shown in the Figure 1 corresponds to the map of a 2D relaxation/relaxation correlation experiment. The box {9} allows one to remove a certain amount of (too noisy) columns or rows indicated in the corresponding edit text boxes, from the four sides of the decay map, after clicking on the "remove" button. One can go back to the original data by clicking on the "delete" button.

Data processing:

The NNLS parameters are entered in the "nnls smoothing parameters" box {10}, such as α, the minimum and maximum values of the relaxation times or diffusion coefficients on which the distributions are going to be calculated, as well as the number of values (steps) of those series that will determine the size of the distribution {10}. The range values are entered in second for the relaxation times and in $10^{-9}m^2/s$ for the diffusion coefficients. The chosen size of the distributions cannot be bigger than the size of the original data. A 25×25 distribution map should take about 1 minute of computation time. It is therefore recommended to set the different parameters (coordinates ranges, …) by computing a smaller distribution, such as 15×15. For 2D data only, the "orient" text box indicates the orientation in which the 2D to 1D transformation is going to be done ("h" for horizontal, "v" for vertical and "b" for the geometrical average between the horizontal and vertical transformations). The 4 other buttons in nnls smoothing parameters are the following.

"GInput" allows the min/max values to be set using the pointer. "Store" saves the current "min", "max" and "steps" values, the stored values are reset when a new data file is loaded. "Reset" recalls the last stored values. "Crop" attempts to adjust the min/max values so that the right graph {11} has a border of zeros one layer thick. The number of loops used by the program and the computation time are indicated below the box {12}. The 1D or 2D distributions are displayed on the right graph after clicking on the "nnls-smoothing" button {13}. The position of the peaks of the distributions can be picked up by the "Get $T_{1/2}$/D" button {14}. The values of the relaxation times are displayed in second and the values of the diffusion coefficients in m^2/s. The "Boolean Colour Axis" check box {15} changes the colour axis of the right graph to allow one to locate hard-to-see peaks.

Determination of alpha:

The value of the smoothing parameter α can be checked by measuring χ^2 {12} as a function of α, as can be done by the function "Alpha loop" {16}. The parameters of the "Alpha loop", namely the first and last alphas and the number of steps are entered in the box above the button. The curve of χ^2 as a function of α is then drawn and one has to choose the optimum value of α as explained in Figure 2. The box "Draw all" has to be checked if one wants to see the different distribution maps as a function of alpha.

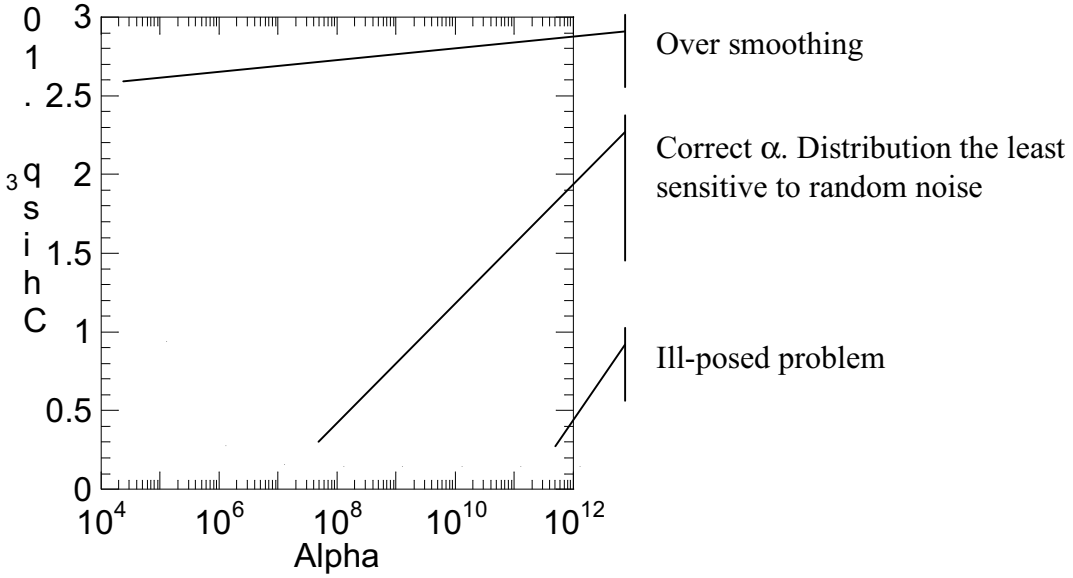

Figure 2. Determination of the smoothing parameter α.

Saving the distributions:

The 2D matrix of the computed distribution are automatically saved in an ASCII file under the same name that the signal data but with the extension ".out". The files of the coordinates of the distributions in the two directions are saves under the same of the "relaxation time" file but with the extension ".out" as well. The coordinates

corresponding to the direction of a diffusion experiment are named from the file name of the data with the addition of "grad.out" to their name.
In addition, one can also save the computed distributions as a .fig file as well as a .jpg file, under their name added to "m.fig" or "m.jpg", respectively. This process can be done from the {17} set of buttons and check boxes. To get the .fig and .jpg file, the check box "new figure" and the "Redraw" button must be clicked on. The "trans" check box allows to inverse the two directions of the distribution map. Of course, the colour map of the distribution can be change from the matlab features of the .fig file. One can also get the "video" of the distribution showing the high of the different peaks and saved under their name added to m.avi".

4 EXAMPLES

Below are a few examples of the capability of the 2D Laplace Inversion. On the left are shown the simulated 2D distribution maps from which 300×300 simulated signal decay maps were generated. Some noise was added to the data (S/N≈200) to make the simulated signal more realistic. On the left are the 2D distribution maps computed with the 2D Laplace Inversion program, showing its reliability.

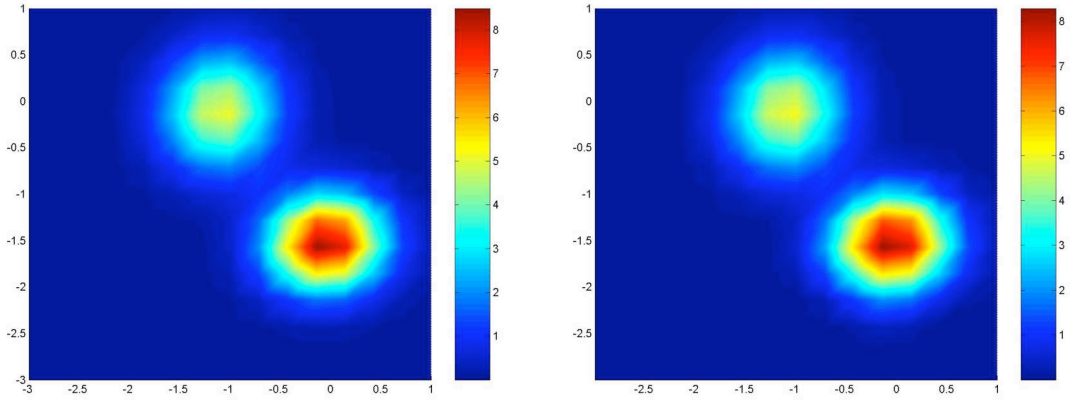

Figure 3. Simulated 2 large peaks 2D distribution map (on the left) used to generate a simulated 2D decay signal, whose computed distribution is on the right.

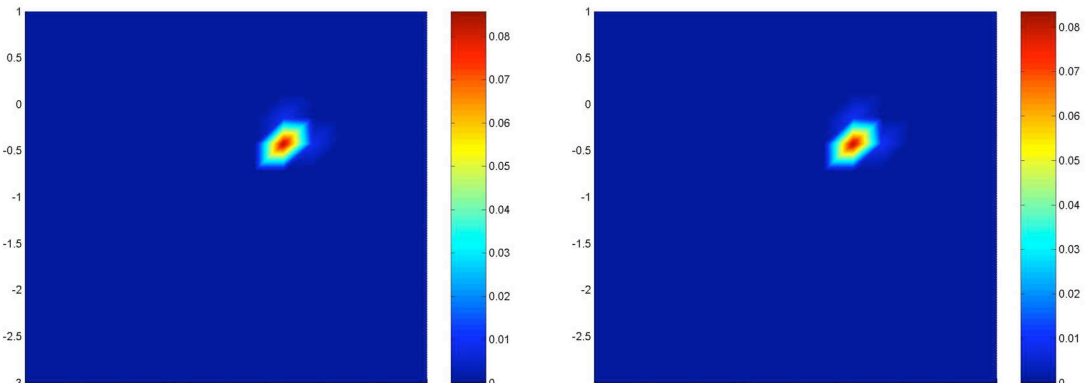

Figure 4. Simulated 1 narrow peak 2D distribution map (on the left) used to generate a simulated 2D decay signal, whose computed distribution is on the right.

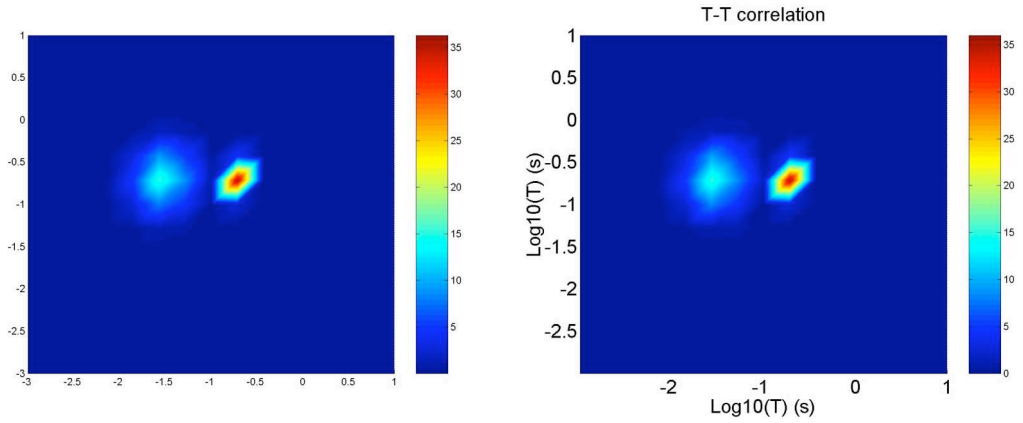

Figure 5. Simulated 2 narrow peak 2D distribution map (on the left) used to generate a simulated 2D decay signal, whose computed distribution is on the right.

---

[1] C. L. Lawson and R. J. Hanson, solving least squares problems, Englewood Cliffs, N.J., Prentice-Hall, 1974.

[2] S.W. Provencher: Inverse problems in polymer characterization: Direct analysis of polydispersity with photon correlation spectroscopy. *Makromol. Chem.* **180**, 201 (1979).

[3] S.W. Provencher: A constrained regularization method for inverting data represented by linear algebraic or integral equations. *Comput. Phys. Commun.* **27**, 213 (1982).

[4] S.W. Provencher: CONTIN: A general purpose constrained regularization program for inverting noisy linear algebraic and integral equations. *Comput. Phys. Commun.* **27**, 229 (1982).

[5] G. C. Borgia, R. J. S. Brown and P. Fantazzini, Uniform-Penalty Inversion of Multiexponential Decay Data, Journal of Magnetic Resonance, **132**, 65-77, 1998.

[6] Y.-Q.-Song, L. Venkataramanan, M. D. Hurlimann, M. Flaum, P. Frulla and C. Straley, T1-T2 correlation spectra obtained using a fast two-dimensional Laplace Inversion, Journal of Magnetic Resonance, **154**, 261-268, 2002

[7] C. L. Lawson and R. J. Hanson, solving least squares problems, Englewood Cliffs, N.J., Prentice-Hall, 1974.