Mfit Home Page

E. Farhi [farhi@ill.fr]

Table of Contents: MFit

Table of Contents: MFit

Table of Contents: MFit

M*fit* user guide

- Basic topics is the starting point for new users. The topics take you through the steps necessary to load your data, fit a model function to it, and save the results.
- Extending M*fit* explains how to enhance M*fit* by adding your own load functions, fitting functions, and fitting routines.
- Batch files allow you to automate fitting and provide a permanent record of fitting procedures using a simple macro language.
- Customizing M*fit* describes the structure of the initialization file read when M*fit* starts. This allows you to change various default options to suit your needs and preferences.
- Obtaining and installing M*fit* explains how to download and set up M*fit* on your system.
- Send email for more help, or with comments or bug reports (to E. Farhi or D. McMorrow)

Basic topics

To start M*fit*, first open a Matlab session, and type `mfit`
If you encounter any problem here, you may need to check that the install process is complete.

- **Loading data files**
- **Choosing fitting functions**
- **Fitting**
- **Saving results**

Going further with M*fit*

- **More advanced Fitting**
- **Options when loading data files** (Column selector, MFit parameter file, General Load routine, Direct Data entry)
- **Manipulating data** (X−axis transform, data transformation, direct operations...)
- **Building a complex fitting function** (multiple function fit, convolutions...)
- **How to write fitting functions**
- **How to write load functions**
- **How to write fitting routines**

Batch files

- **M*fit* batch files** (INI format ans usage, saved user configurations)

Customizing M*fit*

- **The `mfit.ini` file**

How to obtain and install M*fit*

- **Download and Install M*fit***

Possible Future evolutions of Mfit

M*fit* is still under developpement. Some envisaged enhancements are (as far as we can find time to implement that !):

- Multiple data windows
- Multiple data in one data window

- More complex data treatment (such as in *Mview*)
- Possibility to constrain parameters during fits
- Interface with the McStas program for 3−axis instrument simulations
- Improvement of Rescal

Back to the **M*fit* home page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

How to obtain and install M*fit*

M*fit* consists of about 40 Matlab .m functions, as well as the load functions, fitting functions, and fitting routines. To install **M***fit* on your system, you need to download an archived version, unpack it, and set up the configuration file. Follow the steps below:

This installation only considers **M***fit*. For the whole package look at
http://www.ill.fr/tas/matlab/doc/matprgs.html

1. Download the archives below

- The Install.txt file (containing descriptions, installation procedures and Whatsnew).
- Mfit 4.2 − tar gzip compressed archive
- the sublibraries for **M***fit*.

load Load routines for **M***fit* and Mview (with looktxt as Mex)
funcs Fit functions for **M***fit* (with `trix` and `convlv`)
nllsq Fit methods for **M***fit* (simplex and adapatative gradient)

2. Unpack the archives

- `cd` to the directory in which you want to install **M***fit*, e.g. `c:\matlab`, `/usr/local/matlab5/toolbox/local` or `~/matlab`
- type **`tar -xzf mfit4.tar.gz`** to extract the tar gz archive. Winzip can handle this on PC's, or MacGzip on Macintosh

Unpacking archive will create subdirectories called:

`mfit4` for the main function files
`load`, `funcs` and `nllsq` for the load functions, fitting functions, and fitting routines.
The `trix` subdirectory is included in `funcs` for TAS 4−D neutron data fits, but requires to install the *Rescal* package..

3. Create the required Mex files ffind and texmex

- The General Load Routine (load/multibatch) uses a Mex function 'texmex' including the 'looktxt' C code.
  To install it, go into the '`load`' directory and type in Matlab:

  ```
  >> mex -O -output looktxt texmex.c
  ```

  This will create a 'looktxt.mex???' file.

- The ffind routine exists in two versions (.m and .mex) in the load directory

  If you want to boost the importation of some specific data files (e.g. SPEC ILL and TASCOM...) you can optionally use the ffind MEX−function.
  Move to the '`load`' dir, and type (on terminal prompt):

```
>> mex -O ffind.c
```

Otherwise the normal (slower) ffind.m routine is used.

- Optionally, if you plan to install Rescal, you will need also (in `funcs/trix`) to type

```
>> mex -O mcint.c    or    mex -O mcint2.c
```

This is needed if you want to perform some 4D fits of neutron scattering data acquired on a TAS instrument.

4. Edit the Mfit configuration file to match your installation location

5. Add the M*fit* directory to your Matlab path

- See your Matlab manual.and the example startup.m file.

6. Run M*fit*

- Type '`mfit`' at the Matlab command prompt

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

[Choose a load function][Choose a data file]

Loading data files

Choose a load function

M*fit* uses load functions to get data from your data files into a form it can recognize. The first step in loading data is to choose the appropriate load function from the 'MFit:Control:Load Routine' menu in the control window:

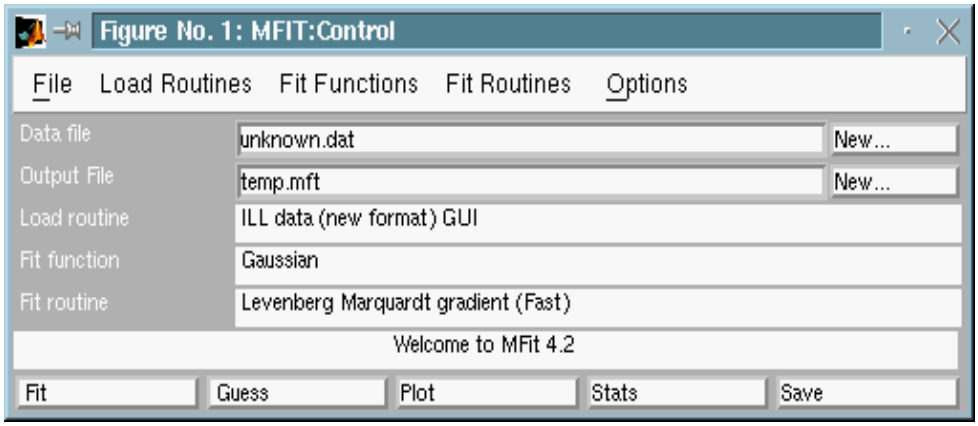


Fig 1 : **M***fit* : control window

The functions that appear on the menu are those that **M***fit* read from the mfit.ini file when it started. Choose a load function from the menu by clicking on it. If you choose '*Other*...' you can choose a load function that is not one of those listed in the .ini file. **M***fit* shows the currently selected load function in the control window.

Some load routines :

- The *General Load routine* enables to load any data stored into a text (ascii) file. It does analyze the file contents (with looktxt), separate and sort numerical parts (single numbers, vectors and matrices).
- The *MFIT parameter file* enables to get the parameter values saved in an **M***fit* output file (see Saving results), for instance plot an amplitude *vs.* a temperature.
- The *Direct User data entry* enables, when pressing 'New...' button or selecting 'Load New Data file...' in the control window, to enter directly an expression for x, y and the error. One can use here the user variables from the matlab session. A blank field will re−use the active data.
- It is also possible to transfert directly (by typing matlab commands) data between **M***fit* and the matlab workspace. See Exchanging data between **M***fit* and the `matlab` workspace.

Usually, load routines are available as *Graphic User Interface* (GUI, asks user preferences) and *Automatic* (tries to guess, usefull for batch runs) modes
You can define your own load routines for special data formats.

Choose a data file

The next step is to choose your data file. There are actually three ways you can do this: the simplest is to press the '**New..**' button to the right of the 'Data file' box in the control window (see the above figure 1).
Alternatively, you can choose 'Load new data file...' from the control window 'File' menu (or Control−L key).
In either you will be presented with a dialog box to allow you to choose a data file:

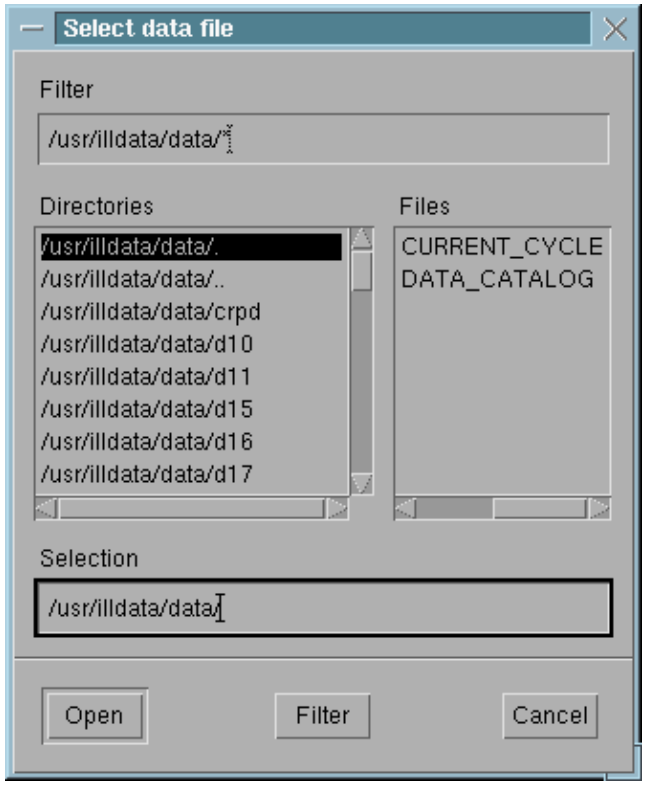


Fig 2 : **M***fit* : selecting a file to load.

Choose a file, changing directories if necessary. **M***fit* will attempt to load the selected file using the current load function. When using a Graphic User Interface (GUI) routine, the load routine might ask for additional informations about how to import the data. Often, the column selector will appear.

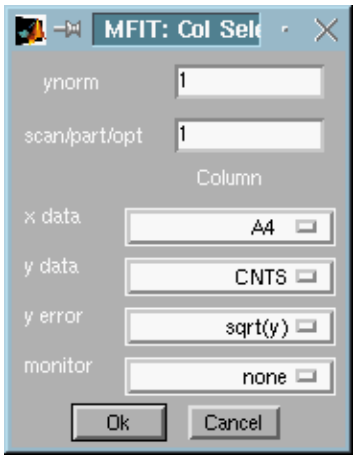


Fig 3 : **M***fit* : column selector in GUI mode

Click OK button. If successful, a data window, containing a graph of your data will be opened. Get the example ILL data file here. It looks something like this:

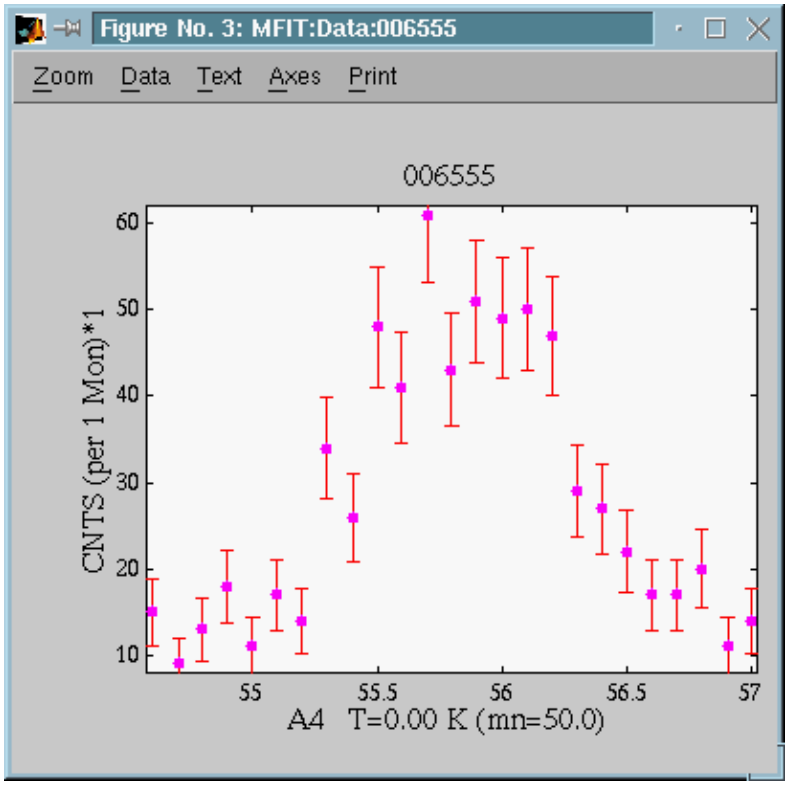


Fig 4 : **M***fit* : the data window (with ILL data GUI load).

If the data file you want to load is in the current data directory (i.e. the same directory as the last data file loaded), you can just edit the name in the data file box in the control window, right next to the 'Data File' indication in the MFit Control window.
You may click on any text to edit it, zoom, select/deselect points with the left button. The middle button in the graph will indicate the point coordinates. You may also manipulate your data ('Change X−axis' and 'Transform' items of the 'MFit : Graph/Axes' menu). If you encounter some problems in viewing your data, you can still close the data window (MFit : Graph : Data menu/Close), and reload it (MFit : Control : File menu). You can also change the color mode with the matlab command '`colordef`'.

Next: Choosing fitting functions.
See also: Options when loading data files, **M*fit* batch files** , **The `mfit.ini` file** , **Manipulating data** , **How to write load functions**.

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

Choosing a fitting function

The Matlab functions which calculate the model functions you fit to data are called fitting functions. The procedure for choosing a fitting funtion is similar to that for choosing a load function: you pick a function from those listed on the control window 'Fit Functions' menu:

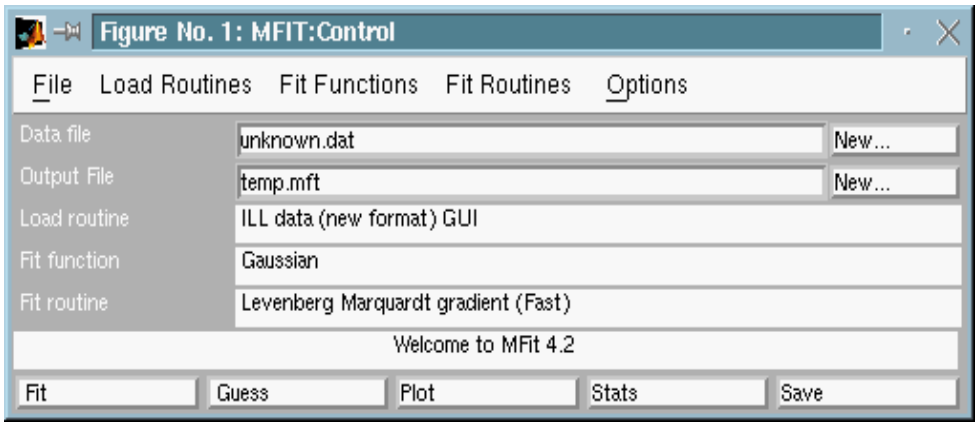


Fig 1 : **M***fit* : control window

The functions listed are those that **M***fit* read from the mfit.ini file when it started. Choose the function you want by clicking on it. You can choose an unlisted function by choosing '*Other*...' and then using the dialog that appears to tell **M***fit* about your function. **M***fit* show the currently selected fit function in the control window. It is possible to build a more complex function (such as a gaussian plus a lorentzian, convoluted with an experimental resolution function) with the multifunction feature.

If **M***fit* successfully loads the fit function, it opens a parameters window which shows the values of the parameters associated with the particular function. The parameters windows looks something like this:

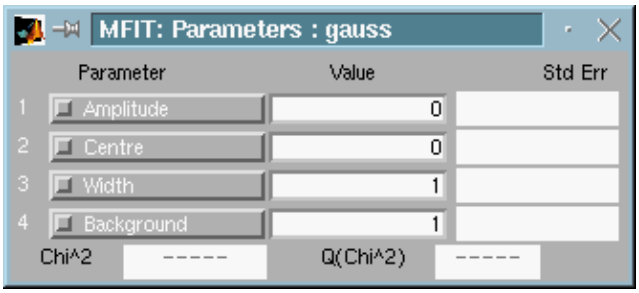


Fig 2 : **M***fit* : parameter window

The number and names of the parameters of course depends on the function you chose, and the boxes will be blank when you first load a function.

Next: **Fitting**.
See also: **M*fit* batch files** , **The `mfit.ini` file** , **Building a complex fitting function** , **How to write fitting functions**.

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

By now you should have successfully loaded a set of data, and chosen a fitting function. There should be three **M***fit* windows open: the control window, the data window, and the parameters window. You can now begin fitting.

Basic fitting

1. Getting starting parameters
2. Performing a fit

More advanced fitting

1. Fix and free parameters
2. Select data to fit
3. The autoguess feature
4. Change fit routine options
5. Choosing an alternate fit routine

Basic fitting

Get starting parameters

Non−linear fitting works by using an algorithm to attempt to improve on a first 'guess' that you make for the function parameters that fit your data. The first step in fitting is to give **M***fit* a set of starting parameters to work with. You can do this in two ways. The first is simply to type them into the parameters window:

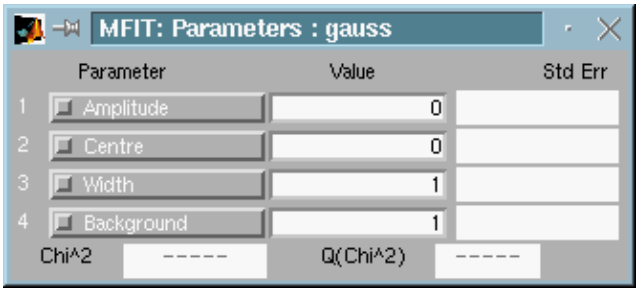


Fig 1 : **M***fit* : parameter window

you click on the box showing the value of the parameter you want to change, and type the new value. After you have entered values for all the parameters, press the 'plot' button in the control window, and the function will be drawn in the data window.

This is slow, and for many functions it's not obvious what starting parameters might be. Many **M***fit* fit functions allow you to get the starting parameters with the mouse, by clicking on points in the data window. Click on the '**guess**' button in the control window:

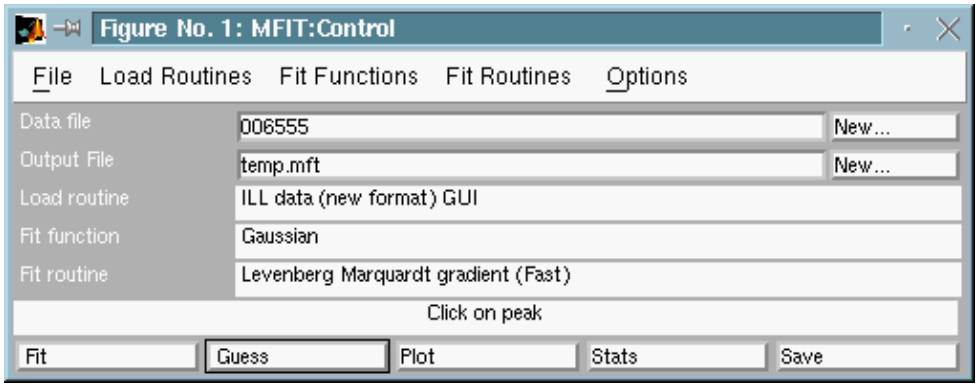


Fig 2 : **M***fit* : control window asking for a peak position.

and follow the prompts displayed in the message box, the wide test box just above the row of buttons in the control window. For example, many peak fit functions ask you to click on the centre of the peak (in the data window), then on the width (i.e. the half height position), and then on the background. When you've fiinished, the function will be draw in the data window.

Beginning a fit

Having chosen starting parameters, you can now begin the fit by pressing the 'fit' button in the control window. As the fit progresses **M***fit* shows the number of fit iterations in the message box, and draw the function with the current parameters in red in the data window. When the fit finishes, the word 'Done' is displayed in the message box, and the values in the parameters window are the results of the fit. The parameters window also displays the uncertainties associated with each parameter value, and the value of the reduced chi squared, and Q, the probability that chi squared should exceed its current value by chance.

Next: Saving results

Mfit Back to the *Mfit* user guide page

More advanced fitting

Fix and free parameters

You will often want to fix some of the function parameters to preset values, preventing the fitting algorithm from changing them during the fit. You do this simply by clicking in the check box to the left of the parameter name in the parameters window:

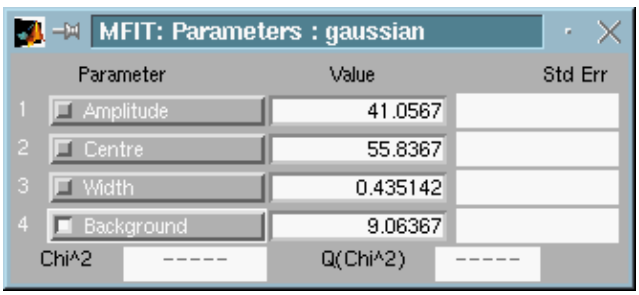


Fig 3 : **M***fit* : parameter window with a fixed backgroung

Here, the 'Background' parameter has been fixed at 9.06. You can free a parameter by clicking on the checkbox again.

Select data to fit

Sometimes, you may want to restrict the data points you want to fit: you might just want to fit a small feature in your data, or you may want to exclude data points that you believe are spurious (dangerous, unless you have very good reasons!) You can include and exclude points from the fit by choosing operations from the 'select data' menu in the data window:

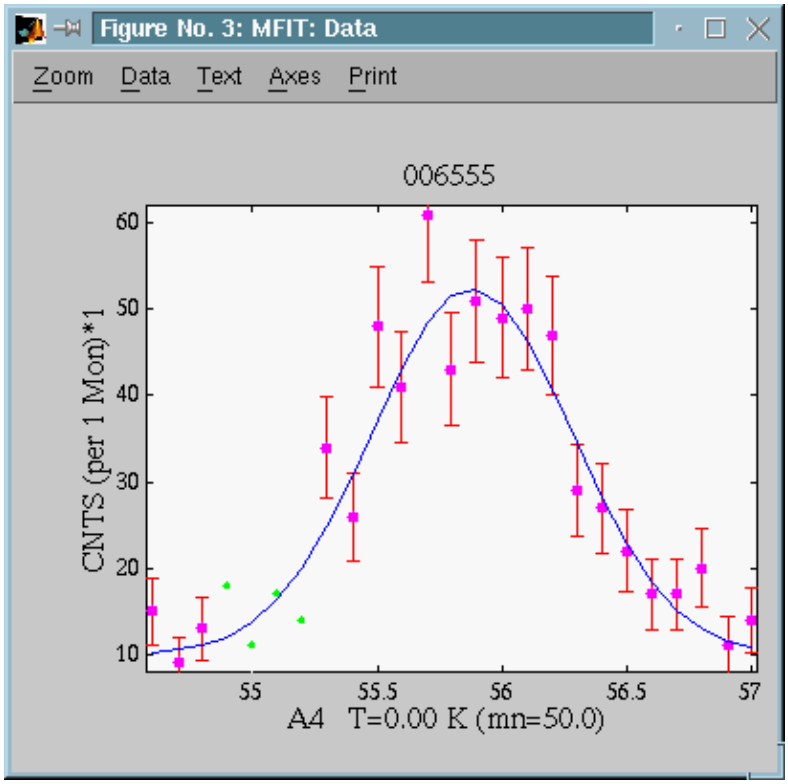


Fig 4 : **M***fit* : Deselected points in the Data window

To select data points to include in the fit, choose 'Select Data' in the 'Data' menu of the '**M***fit* : Data' window . The cursor will change to a cross hair, and you click and drag on the data window to define a box enclosing the data points you want to select. Keep clicking and dragging until you have selected all the points you want to include. You deselect points in the same way. To set the zoom mode again (default mode), just use the Zoom menu.

The AutoGuess feature

Most fit functions require peak positions, widths, amplitudes and backgrounds. **M***fit* can perform an automatic guess for such parameters. When checking the 'AutoGuess' item in the Option menu of the control window, it analyzes the data, looking for peak characteristics, and sets the guessed values to parameters, according to their *names*. It does work fine for most functions, but the data must contain enough points to enable an efficient data analysis.

Change fit routine options

It is possible to change some parameters controling the fitting process. To do so, select the *Fit Control* item of the *Options* menu in '**M***fit* : *Control* window':

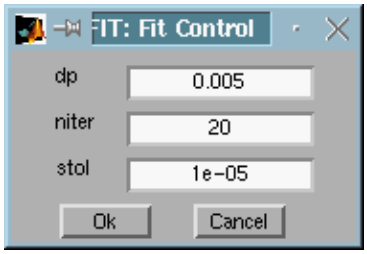


Fig 5 : **M***fit* : The Fit control window

You can there change :

- **dp** − this parameter depends on the fit routine (see Choosing an alternate fit routine)
  - for the Marquadt Levenberg method, this is the step used to compute partial derivatives `f(x,p+dp) - f(x,p)/dp` (in percent of parameter). Reasonable value should be below 0.1 (10 %) ; when higher, this could lead to uncertainties in derivative estimates.
  - for the Simplex method, it indicates the exploration range around starting parameter set (in %). For instance, a value of 0.1 will indicate that each parameter will first be investigated between +10%/−10% around starting value. This range can be extended during simplex fitting process. Reasonable values should be between, say 0.1 (10 %) and 0.5(50 %). High values are to be used when your starting parameters are very approximative.
- **niter** − maximum number of iterations
- **stol** − sets the convergence tolerancy for fit. When the fit criterion (usually least−square) variation between two fit iterations is less than that value, the fit is considered to be stable, and finishes.

Choosing an alternate fit routine

The standard fit procedure (optimization) is to minimize the least−square criterion by mean of an adaptative gradient (Marquardt−Levenberg method). This routine is supplied in a graphic mode (showing the fit process, and the converging parameters), as well as a non−graphic faster routine.
But you may choose to use some other fitting procedure in the 'Fit Routine' menu of the Control window. For instance, a **simplex** method is also supplied.

- **Marquardt−Levenberg method** − is fast and efficient, but can converge towards a wrong solution in some cases (local criterion minimum)
- **Simplex method** − is slower, but can easely avoid local criterion minima solutions, and finally converge to the global minimum

Refer to the How to write fitting routines page in order to add your own minimization method.

Next: **Saving results**
See also: **How to write fitting routines** , **How to write fitting functions**

Back to the ***Mfit* user guide page**
Back to ILL/TAS Matlab Page

Mfit

Version **4.2**. (1999)

[Set the output file][Save the results][Saving automatically][Mfit results file format]

Saving data

M*fit* allows you to save the results of your fitting. You can save the fit paramaters and uncertainties, but you can also save the fitted curve (to import into a graphing program for instance), and you can re−save your data. This last feature provides a convenient way to use **M***fit*'s load functions to convert special format data to standard x, y, error column ascii format. that can be easily read by, for example, graphing and spreadsheet programs.Saving results

Set the output file

M*fit* appends saved data to the end of the current output file. The current output file name is displayed in the control window 'Output File' field:

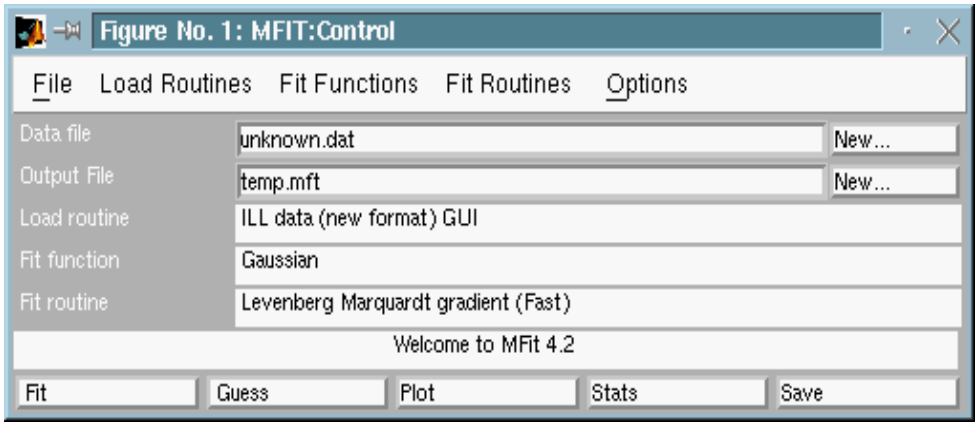
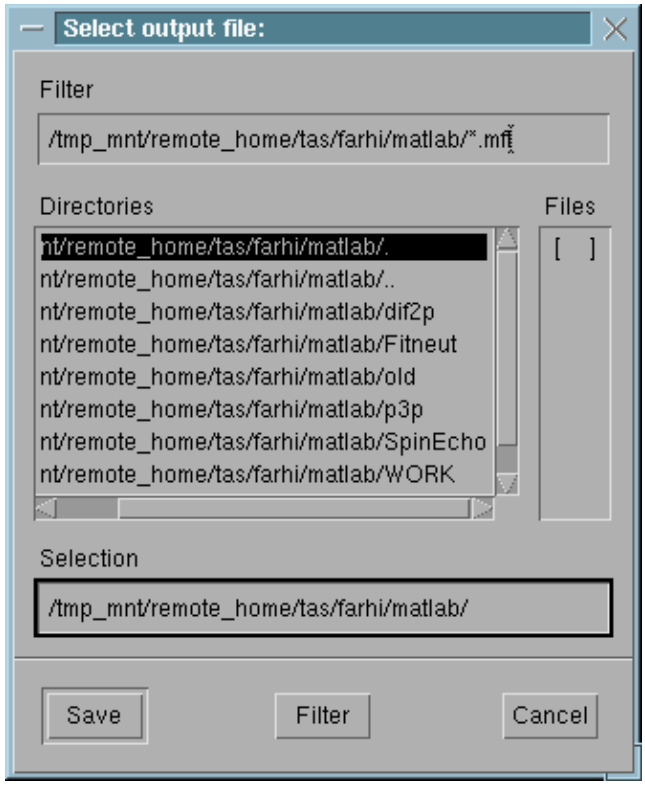


Fig 1 : **M***fit* : control window

You can set the output file in one of three ways:

- Press the 'New...' button to the right of the output file name in the control window.
- Choose 'Set output file...' from the control window 'File' menu
- Edit the output file name displayed in the control window (this is quick, but you can't change directory)

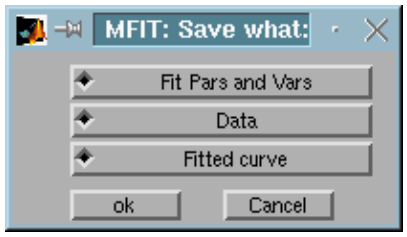
If you use either of the first two methods, **M***fit* displays a dialog that lets you choose a directory and file. It looks something like this:



You can either select an existing file to which the saved information is appended, or enter a new file name, in which case a new output file is created.

Save the results

If you select the Press the 'Save...' item of the File menu in the Control window, this brings up a dialog like that below, which allows you to choose what to save.



You can check one or more options, then click ok.

- Fit parameters − save the fit parameters in standard M*fit* format.
- Data − saves your data as a x, y, error ascii file
- Fitted curve − saves the fitted curve as an x,y ascii file. If the Data button is not checked then the x limits and number of points can be specified in the boxes provided. If the Data button is checked then the fitted curve is evaluated with the same x values as the data and saved with the data in a four−column x, y, error, fitted curve ascii file.

Pressing the 'Save' button at the bottom right corner of the Control window will save results directly with whichever options are currently selected in the Save what window.

Saving automatically

The '**AutoSave**' feature enables to save automatically the fit results in the active output file just before loading a new data file. This assumes that the last parameter set is the good one, and all fit results of the fitting session are stored in a single file. You can then use the *Mfit parameter file* load routine to analyze these results.

Back to the **M*fit* user guide page**

M*fit* results file format

The Mfit results file format is designed to be complete and easily readable, both by you and by **M***fit*. In addition to the parameter values and errors, **M***fit* records the save date and time, the data file, the output file, the fit function, load function, and fit routine, and chi squared for the fit.

Example:

```
%MFIT Date 3.1.2000  Time 16:46:51
% Section : Vars - Data : 006555 - Function : gauss .
DataFile        = 006555
DataDir         = /usr/illdata/data/in20/
FitFuncName     = Gaussian
FitFuncFile     = gauss
FitFuncDir      = /home/tas/matroot/matlab/funcs
OutFile         = temp.mft
OutDir          =
LoadRoutineName = ILL data (new format) GUI
LoadRoutineFile = illgui
LoadRoutineDir  = /home/tas/matroot/matlab/load
FitRoutineName  = Levenberg Marquardt gradient (Fast)
FitRoutineFile  = mf_flsqr
FitRoutineDir   = /home/tas/matroot/matlab/nllsq
% [006555] is : [A4   T=0.00 K (mn=50.0)] versus [CNTS (per 1 Mon)*1].
load
% Section : Parameters (4) - Data : 006555 - Function : gauss .
par   1 Amplitude    4.127740e+01   3.661250e+00   0
par   2 Centre       5.602750e+01   3.606060e-02   0
par   3 Width        2.595580e-01   2.926780e-02   0
par   4 Background   9.725880e+00   0.000000e+00   1
% CorCoef 0.317 -- RV 4.249 -- ChiSq 4.957 -- Q ChiSq 0.000
% Section : Data (21)- Data : 006555 - Function : gauss .
%             x             y         err:y          yfit
5.460100e+01 1.500000e+01 3.872983e+00 9.725891e+00
5.470500e+01 9.000000e+00 3.000000e+00 9.725975e+00
5.479800e+01 1.300000e+01 3.605551e+00 9.726434e+00
5.529800e+01 3.400000e+01 5.830952e+00 1.052099e+01
5.540300e+01 2.600000e+01 5.099020e+00 1.200974e+01
5.550100e+01 4.800000e+01 6.928203e+00 1.500107e+01
5.559500e+01 4.100000e+01 6.403124e+00 2.002488e+01
```

```
5.570500e+01 6.100000e+01 7.810250e+00 2.880159e+01
5.579800e+01 4.300000e+01 6.557439e+00 3.764786e+01
5.589700e+01 5.100000e+01 7.141428e+00 4.610236e+01
5.600100e+01 4.900000e+01 7.000000e+00 5.078871e+01
5.610000e+01 5.000000e+01 7.071068e+00 4.942405e+01
5.619900e+01 4.700000e+01 6.855655e+00 4.290854e+01
5.629800e+01 2.900000e+01 5.385165e+00 3.370708e+01
5.640200e+01 2.700000e+01 5.196152e+00 2.430259e+01
5.649600e+01 2.200000e+01 4.690416e+00 1.782146e+01
5.660000e+01 1.700000e+01 4.123106e+00 1.335075e+01
5.669900e+01 1.700000e+01 4.123106e+00 1.117912e+01
5.679800e+01 2.000000e+01 4.472136e+00 1.022962e+01
5.690200e+01 1.100000e+01 3.316625e+00 9.867387e+00
5.699600e+01 1.400000e+01 3.741657e+00 9.764999e+00
% End of Data 21 lines, 4 columns
```

The format for the parameters is parameter number, parameter name, value, and uncertainty. An uncertainty of 0 indicates that the paramter was fixed during the fit. This information is also given by the last column (0 : free, 1 : fixed).

Mfit results files are also batch files. You can run them by clicking the 'Batch' button in the control window.

See also: ***Mfit* batch files** , **Mfit parameter file load routine**

Back to the ***Mfit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

[Exchanging data][Data transformations][X−axis transform]

Manipulating data with Matlab and M*fit*

Exchanging data between M*fit* and the `matlab` workspace

The **M***fit* program handles some local variables, refering to the active data window and parameter window:

x − [length *n*, vector] the X axis data
y − [length *n*, vector] the Y axis data
err − [length *n*, vector] the error bars amplitude
selected − [length *n*, vector] the selected points for fit (0 : unselected, 1 : selected)
p − [length *m*, vector] the fit function parameters
dp − [length *m*, vector] the fit function parameters uncertainties after fitting
fixed − [length *m*, vector] the fixed parameters (0 : free, 1 : fixed)
fit − [usually length *n*, vector] the fitted curve, evaluated from fit function

These are not normally accessible from the Matlab workspace. Anyway, there are two functions that enable to manipulate these variables directly. In all **M***fit* routines (load routines, fit functions, fitting routines...), these variables are available as local variables.

Matlab −> M*fit* : tomfit

Suppose you have some data stored into the Matlab workspace. You can send these data directly to **M***fit* by mean of the `tomfit` function:

```
>> tomfit(x,y,err,selected,p,fixed);
```

If you use empty parameters for that function (for instance `tomfit([],y)`), the corresponding **M***fit* data will be kept (not modified).

M*fit* −> Matlab : fromfit

Similarly, you can retreive **M***fit* data into the Matlab workspace by mean of the `tomfit` function:

```
>> [x,y,err,selected,fit,p,dp,fixed]=fromfit;
```

This enables to perform some operations directly on matlab variables coming from **M***fit*, and then send them back to **M***fit*.with tomfit.

Data transformations

Once you have loaded some data into **M***fit* (see Loading data files), it is possible to modify **M***fit* internal variables `x`, `y` and `err`, when selecting the '*Transform...*' item of the *Data* menu in the **M***fit* : Data window.

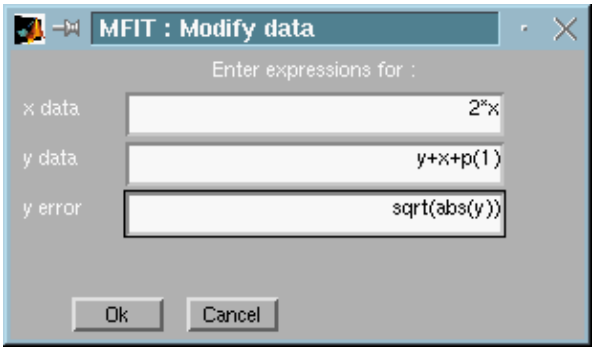


Fig 1 : **M***fit* : Transform pop−up from Data window.

Internal variables can be used here. This method does not require to type commands into the Matlab workspace, as iwhen you use the Exchanging data with Matlab workspace method.

In fact, it acts exaclty the same as the *Direct User data entry* , but uses the **M***fit* variables instead of the Matlab workspace ones.

X−axis transform (rescale)

In some cases, the X−axis needs to be **rescaled** (for instance when a non linearity occurs in scanning during an experiment).
In that case, select the '*Change X−Axis*' item of the *Data* menu in the **M***fit* : Data window.

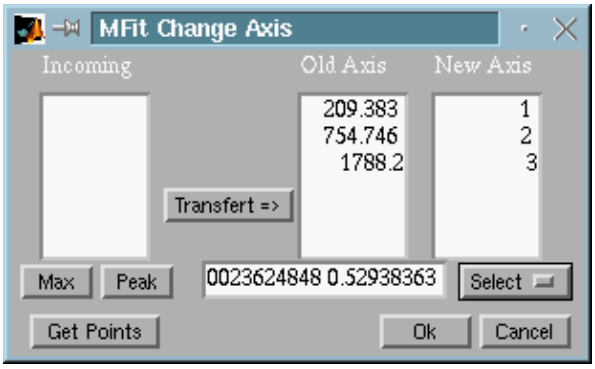


Fig 2 : **M***fit* : Change X−Axis pop−up from Data window.

1− For this operation, you first select some X−values in the current Data window. Either you enter the numerical values with the keyboard, or you click on '***Get Points***' button. In that case a pointer will appear in the Data window. Select the reference points and hit a keyboard key to end acquisition.

Optionally, you can ask to modify these values either into the nearest maxima (**Max** button), or to compute nearest peak estimates (**Peak** button).

2− Then transfert these values to the '***Old Axis***' catalog, and enter for each of them the real value that better suits you (the one that you would expect) in the ***New Axis*** list.

3− Select the **order** of the polynomial transformation. The corresponding polynome coefficients are displayed.

4− Click on **OK** button. The Old Axis will then be changed into a New Axis. In the given example (Fig 2), the original axis ranged from 0 to 2000. With a second order transformation, it finally ranges from 0.5 to 3.06, and the data appears stretched:

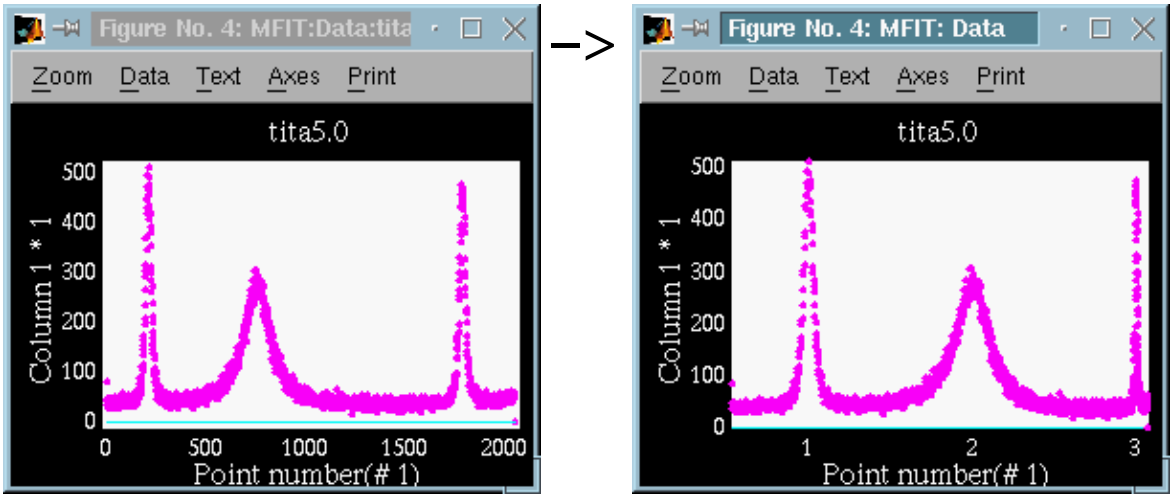


Fig 2 : **M***fit* : Effect of an X−Axis change on data aspect

If you need to perform an X−Axis change for each loaded data set, with the same polynome, you can then activate the ***Auto Rescale*** feature in the *Options* menu of the **M***fit* : Control window.

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

[Column selector][**M***fit* parameter file importation][General Load routine][Direct Data entry]

Some loading options and comments

The column selector

Most of the GUI versions of the Load routines will cause a column selector pop−up window to appear:

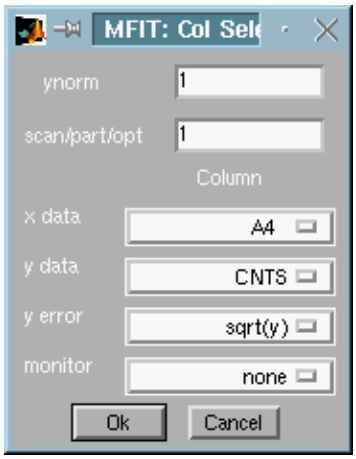


Fig 1 : **M***fit* : column selector in GUI mode

You may modify here, if necessary, the *x*, *y* ,*error* and *monitor* columns assignments. The final signal is *y* divided by the *monitor* (none means 1), and multiplied by *ynorm*.
In some cases where many signals are stored in a single file (for instance the SPEC files), you can enter the scan number (#) in a given experimental file.
Also, if the load routine handles it, you can ask for an automatic parameter search in the data file (use '`1,setpar`' as scan/part/opt specification). For instance, the ILL data load routine can get the temperature, the 3−axis instrument parameters, etc...and transfert them to the Mfit/Rescal windows.

The M*fit* parameter file importation

This routine enables to plot fit parameters *versus* some others.
An **M***fit* parameter file usually contains a serie of similar fit results (see the **Saving results** page, and the AutoSave feature).

When loading the data file, you are first asked to choose among the different results stored in it with a window like:

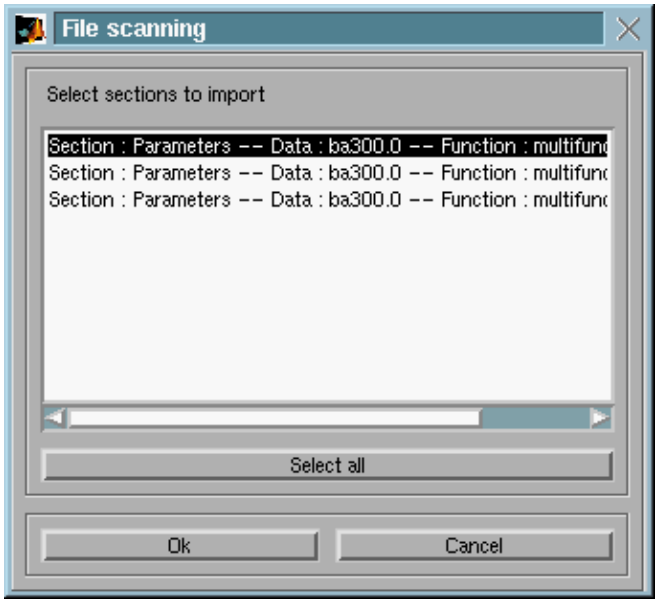


Fig 2 : **M***fit* : parameter file part selector for importation

Then the parameter file is analysed, and a list of parameter names found inside the selected sections is given. Choose the X and Y data in that set. Multiple choices are possible (concatenation).

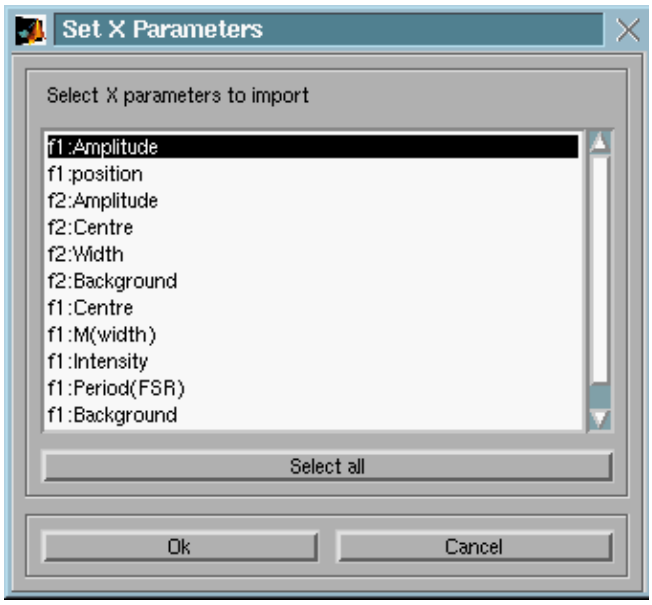


Fig 3 : **M***fit* : parameter file X−Y selector for importation

The General Load routine

This load routine can virtually import any kind of data stored inside a text file. The format of the data is not important. In fact, `texmex`, the MeX version of looktxt , looks for single numbers, vectors or matrices, and can pass−over the character strings and comments.
For instance, when importing a file, the Matlab window will display the analysis as follow:

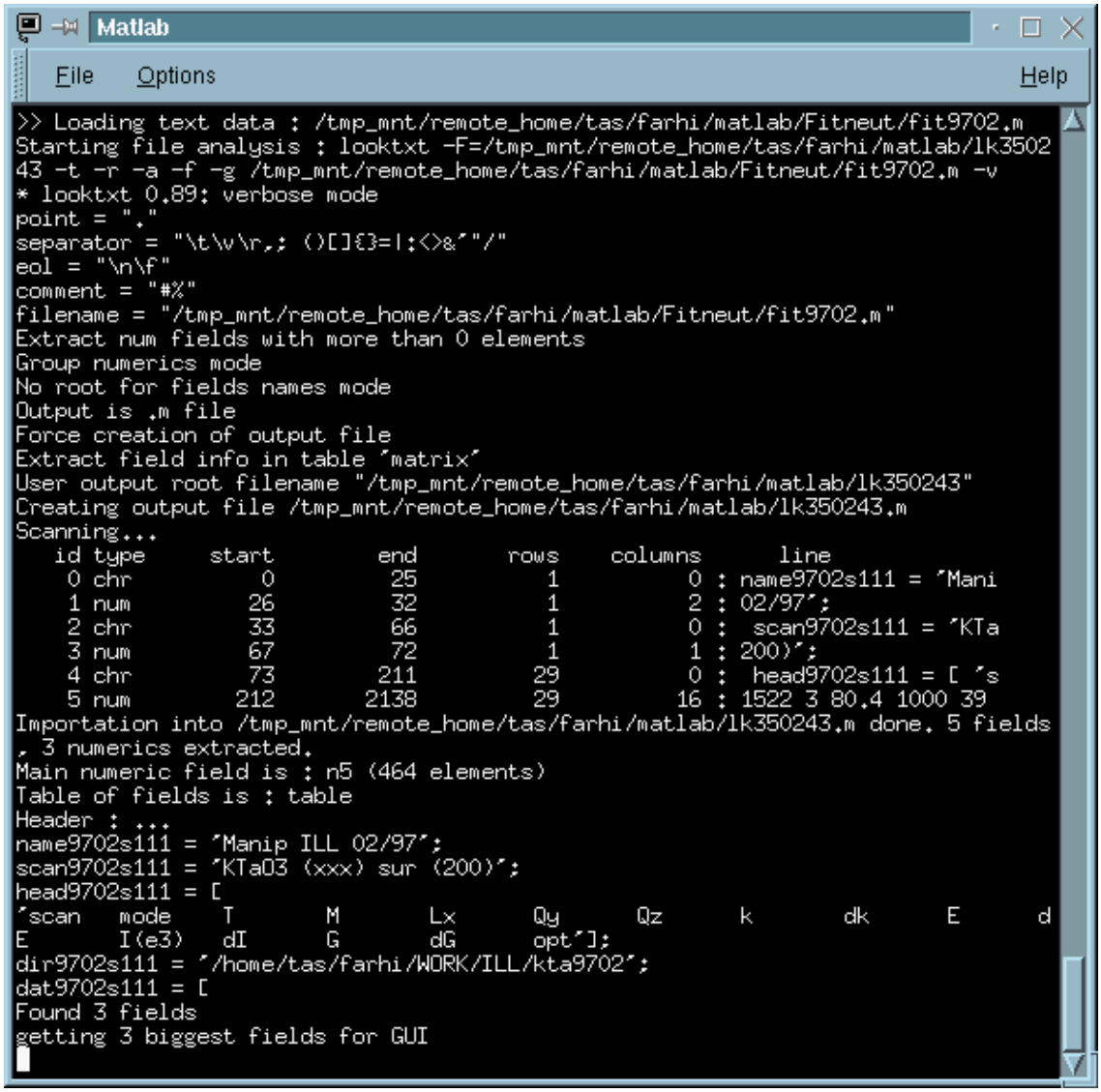


Fig 4 : Matlab workspace : The **M***fit* Multiload file analysis with looktxt/texmex.

This shows the analysis/importation process, sorting character strings, comments, and numerical values. Then, a pou−up window will ask what numerical part should be used as data to import. Of course, you should choose a vector or matrix numerical field (the size of each part is indicated between quotes):

Fig 5 : **M***fit* : The Multiload choice after file analysis.

In that example, the General Load routine automatically selects the bigger numerical field, which is a matrix of dimension 29x16 numbers. Then, the usual Column selector is displayed, and the importation process goes on. Usually, the data files contain a header indicating the column significance just before the data, as shown in that example. This indication is to be used for selecting the column affectation for X, Y, error and monitor.

The Direct Data entry

This load routine enables to import a Matlab variable or an expression for X, Y, error and monitor.
When selecting 'New...' data load button or 'Load new data file...' from the File menu in the MFit : Control window, a window pop−up appears. You can use variables from the matlab workspace, and expressions:

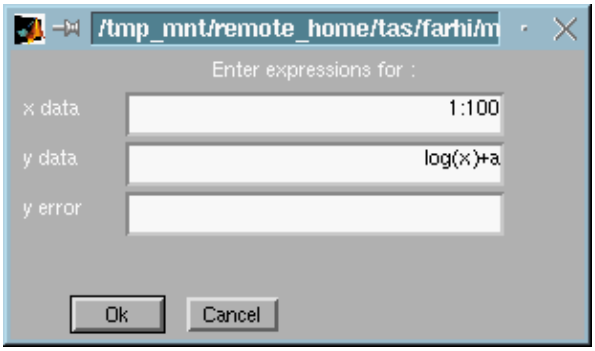


Fig 6 : **M***fit* : The Direct data entry pop−up

In that example, we have previously defined the variable '`a`' in the matlab workspace

```
>> a=2;
```

After selecting OK, `x` will be the vector `1:100` , `y` will be assigned the evaluated expression using `a` and `x`. The error will be constant (not specified by user), but it could have been '`sqrt(abs(y))`'.

See also : Exchanging data between **M***fit* and the `matlab` workspace (**M***fit* variables, *fromfit* and *tomfit* commands)

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

Writing your own fitting functions

Before starting to write a new function, you should wonder if your function can be written as a combinaison of various simple and already available functions. For those common cases, **M***fit* provides a tool, called the multifunction, that does not require to type in a single program line.

The Matlab functions that code the mathematical functions you fit to your data are called fiting functions. One of the most useful features of **M***fit* compared to other fitting packages is that the Matlab language makes it quick and easy to write even complex functions. This section describes how to write your own fitting functions. If you have not used the Matlab language and .m files before, you may want to read the Matlab user guide alongside this document.

Function format

Fitting functions are .m files with the syntax:

`function [y, name, pnames, pin]=function_name(x, p, flag)` The function takes two or three parameters...

- **`x`** − vector of values for which to evaluate the function y=f(x)
- **`p`** − vector of function parameters, i.e. parameter 1 = p(1), parameter 2 = p(2) etc.
- **`flag`** − can be absent, or if present can take the values 1, 2 or a vector:
  - ♦ If absent the function returns y=f(x).
  - ♦ If `flag=1` (identify) the function also returns (see below) the function name, and an array of parameter names.
  - ♦ If `flag=2` (guess) the function returns a vector of intial parameter values (see example).
  - ♦ If `flag=y` (auto−guess) uses (x,y) to guess starting parameters.

...and returns a minium of 1 and a maximum of 4 parameters:

- **`y`** − a vector of function values corresponding to the value of the function for the input vector `x` and parameters `p`.
- **`name`** − a Matlab string describing the function
- **`pnames`** − an array of Matlab strings (one per parameter) giving the parameter names.*fit* appends saved ata to the end of the current output file. The current output file name is displayed in the control window
- **`pin`** − a starting parameter set

Example:

Here is the code for the Gaussian fitting function that is one of the default functions. This can be used as a template for your own functions. Note that you can use the mf_msg([text]) function to display user prompts in the control window message box.

```
function [y, name, pnames, pin]=gauss(x,p, flag)
%
% MFIT function [y, name, pnames, pin]=gauss(x,p, flag)
%       Gaussian fitting function
%       MZ 29.11.94
%
% Check to see whether flag exists: if not, evaluate the function
if nargin==2;
   y=p(4)+p(1)*exp(-0.5*((x-p(2))/p(3)).^2);
```

```
%
% If flag does exist, return the function and parameter names
else
   y=[];
   name='Gaussian';
   pnames=str2mat('Amplitude','Centre','Width','Background');
   pin=[0 0 1 0];
   if flag==2
      mf_msg('Click on peak');
      [cen amp]=ginput(1);
      mf_msg('Click on width');
      [width y]=ginput(1);
      width=abs(width-cen);
      mf_msg('Click on background');
      [x bg]=ginput(1);
      amp=amp-bg;
      pin=[amp cen width bg];
   end
end
```

Note also that the only essential line in a fitting function is the one beginning y=... i.e. the minimal Gaussian fitting function would read:

```
function [y, name, pnames, pin]=gauss(x,p, flag)
   y=p(4)+p(1)*exp(-0.5*((x-p(2))/p(3)).^2);
```

However, **M***fit* would not know the parameter names (it would use names like 'parameter 1', etc.) and the user would have to input initial parameter values manually.

If you want to change a minimal fit function into a Mfit one, just add the `flag` input parameter, the output parameters '`name, pnames, pin` , and the following lines at the end of the function (modify the parameters default values and names anyway):

```
if (nargin>2)
        y=[];
        name='my_function';
        pnames=str2mat('Amplitude','Centre','Width','Background'); %
params names
        pin=[0 0 1 0];                  % default parameters
        if flag==2
                mf_msg('This function cannot handle Guess feature.');   %
user asking section
        end
end
```

See also: **multifunction**

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

Writing your own fitting routines

Non−linear fitting algorithms usually begin with a starting guess for the function parameters that minimize some target function (usually chi squared), and then attempt to improve upon the initial values by an iterative procedure. Different algorithms do this in different ways, and may also estimate the resulting uncertainties in the parameter values obtained in different ways. You can supply your own fitting routines which allows you to determine both the minimization algorithm and the target function.

The Gradient routine supplied with Mfit is adapted from the Levenberg−Marquardt algorithm `leasqr.m` by Richard I. Shrager, A.Jutan, and Ray Muzic. A `simplex` method adapted from the matlab `fmins` function is also provided. Des McMorrow has also written a FORTRAN .mex file interface to the excellent CERN package MINUIT, but this is machine dependent, and so is not included. Ask if you're interested.

Function format

Fitting routines are .m files (or .mex files − see the Matlab external interface guide) with the syntax:

`function [p, std]=function_name(x,y,err,pin,dpin,func)` The input parameters are:

- **`x`**, **`y`**, **`err`** − vectors containing the data. `err` is the unceratinty in `y`
- **`pin`** − vector of initial parameter values
- **`dpin`** − vector the same length as pin. If pin(n)=0 then parameter n is held at the `same value during the fit`
- **`func`** − the name of the fitting function called. (See 'Writing fitting functions' for details)

and the function returns **`p`** and **`std`**, vectors containing the best fit estimates of the paramters, and the associated uncertainties respectively.

Back to the ***Mfit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

[The mfit.ini file][Using more than one .ini configuration file][Saving an active configuration for future use]
[mfit.ini File format][Create a mfit.ini file for your system][Example]

The mfit.ini file

When it starts, **M***fit* reads a lot of configuration information from the `mfit.ini` file. It specifies things like:

- Where to find all your fitting functions, and what to call them
- Where to find all your load functions, and what to call them
- Where to find all your fitting routines, and what to call them
- Data window default position, size, and colors
- Graph fonts and font sizes
- Control window position
- Text height (improves appearance and readability on some OS's)
- The default data directory
- What to do when starting

The advantages of forcing you to register all the functions individually are:

- You can give your routines sensible names, rather than just referring to them by the .m file name − so you choose 'Lorentzian + Lorentzian squared (sloping background)' rather than slorlor2.m, for example.
- You can put the functions in any directory − e.g. with the data they apply to. It's probably advisable to group most of the functions together, however.

The penalty is that you have to set the `mfit.ini` file up manually before you can run **M***fit*. The following sections explain how to set up a suitable configuration file for your installation.

You can still modify some **M***fit* configuration variables after launch by selecting the *Mfit Config* item from the *Mfit : Control : Options* menu.

Using more than one `.ini` configuration file

It is possible to maintain various configuration files. When typing

```
mfit
```

the mfit.ini file is used, but you can also type for instance

```
mfit('my_inifile.ini')
```

In that case, the file that you specified is used for the **M***fit* configuration, and if it is not found, the `mfit.ini` file is used instead.

Saving an active configuration for future use

If you use a special configuration of **M***fit*, for instance after selecting the **M***fit Config* item from the **M***fit : Control : Options* menu, you can save your own configuration into the `inifile` (hich is `mfit.ini` by default but can be changed in *Options : MFit Config*) by selecting 'Save Configuration File' from the *Mfit : Control : File* menu.

The saved the configuration will store (and restore at each **M***fit* startup using your configuration) the active data set, multifunction, data transformation, fit control options it will be included in your configuration.

mfit.ini File format:

The `mfit.ini` file (see example) consists of four or five sections, each of which starts with the section name in curly braces. The sections are `{general}`, which specifies default files, directories and graphical properties, then the `{Load Routines}`, `{Fit Functions}`, and `{Fit Routines}` sections which register load routines, fit functions, and fit routines for use with **M***fit*. The format of the `{general}` section is clear from the example below. The other sections consist of a series of lines, each of which is of the format:

- `[file name], [Directory], [Description]`

where `[file name]` is the name of an .m file (without extension), `[Directory]` is the directory in which it is found, and `[Description]` is a text description of the function. For example, the line

- `gauss, c:\matlab\mfit3\funcs, Gaussian (width sigma)`

registers a function c:\`matlab\mfit3\funcs\gauss.m`, which has a menu entry `'Gaussian (width sigma)'`. Note that a blank line inserts a separator into the corresponding menu.

A fifth optional `{startup}` section can contain some **M***fit* batch commands to execute on starting. Of course, you can use the `exec` batch command to extend this feature to any matlab command.

Create a mfit.ini file for your system:

- Load a copy of a basic `mfit.ini` file into your favourite text editor

- Use 'search and replace' to make change the directories to wherever you have installed **M***fit* on your system: e.g. replace all occurences of `'c:\matlab\mfit\funcs'` with `'~\matlab\mfit3\funcs'`, and so on.
- Save the results as mfit.ini in your **M***fit* directory
- Run **M***fit*, and make sure you can pick functions from the menus Note that you can have more than one `mfit.ini` file: **M***fit* reads the mfit.ini file it finds in directory in which it is started, or the .ini file specified on the command line, eg. `mfit('c:\expdata\mfitexp.ini')` (see Using more than one `.ini` configuration file)

Example:

Here is what a typical mfit.ini file looks like at the moment. Click here to get a copy:
(I've added some comments in red here, other items are intuitive and self−explicite I hope)
Some of the **M***fit* configuration variables are accessible from the *Mfit Config* item of the *Option* menu in the **M***fit* : Control window

```
{General}
```

```
LoadRoutineFile =
LoadRoutineDir  = /home/tas/matroot/matlab/load
LoadRoutineName =
FitFuncFile     = gauss
FitFuncDir      = /home/tas/matroot/matlab/funcs
FitFuncName     = Gaussian
DataFile        = unknown.dat
DataDir         =
OutFile         = temp.mft
OutDir          =
BatchFile       = temp.mbt
BatchDir        =
FigureBgColor   = black
AxesColor       = white
FitColor        = cyan
EbarColor       = red
DataColor       = green
DataColorSelected       = magenta
FigurePosition  = 200 50      % for data graph specifications
FigureSize      = 400 350
ContWinPosition = 37 231
ParWinPosition  = 29 70
AxesFont        = Times
AxesFontSize    = 12
LabelFont       = Times
LabelFontSize   = 16
FitPoints       =             % points to show when drawing fit line,
empty means 'use data point number'
ContWinPosition = 100 500
ParWinPosition  = 50 50
TextBoxHeight   = 18
MarkerSize      = 10
DataLineStyle = none
AutoGuess       = 0
AutoSave        = 0
ShowPwinAtStart = 0           % 0 : hide, 1 :show parameter window at
MFit start
IniFile         = mfit.ini    % you can change this in
MFit:Control:Options:MFit Config:Inifile
ShowBatchLine   = 1           % 0 : hide, 1: show batch command field in
Mfit:Control
ExecAfterLoad   =             % a command to execute after each data load
ExecAfterFit    =             % a command to execute after each data fit

{Load Routines}
===General purpose================    % this appears as a menu separator
multigui, /home/tas/matroot/matlab/load, General Load routine GUI
multibatch, /home/tas/matroot/matlab/load, General Load routine Auto/Batch
%xyeload , /home/tas/matroot/matlab/load, X Y error columns text file
%multi   ,  /home/tas/matroot/matlab/load, Multicolumn GUI
%pargui  , /home/tas/matroot/matlab/load , MFIT parameter file (v0.0)
mf_gpar , /home/tas/matroot/matlab/mfit4, MFIT parameter file (v1.0)
frombase, /home/tas/matroot/matlab/load , Direct User data entry
===Special Neutron data formats============
illgui,   /home/tas/matroot/matlab/load, ILL data (new format) GUI
illbatch,         /home/tas/matroot/matlab/load, ILL data (new format) Auto/Batch
tasgui,  /home/tas/matroot/matlab/load, TASCOM (Riso data format) GUI
```

```
tasbatch, /home/tas/matroot/matlab/load , TASCOM (Riso data format) Auto/Batch
specgui ,  /home/tas/matroot/matlab/load, SPEC GUI
specbatch, /home/tas/matroot/matlab/load , SPEC  Auto/Batch
===Special local data formats============
mcabatch ,  /home/tas/matroot/matlab/load, MCA text data
{Fit Functions}
===Single peaks==================
gauss,   /home/tas/matroot/matlab/funcs, Gaussian
gauss2,  /home/tas/matroot/matlab/funcs, Gaussian squared
sgauss,  /home/tas/matroot/matlab/funcs, Gaussian (sloping background)
lorz,    /home/tas/matroot/matlab/funcs, Lorentzian
lorz2,   /home/tas/matroot/matlab/funcs, Lorentzian squared
airyfp,  /home/tas/matroot/matlab/funcs, Airy (for Fabry Perot)
cusp,    /home/tas/matroot/matlab/funcs, Cusp (double power law)
dho,     /home/tas/matroot/matlab/funcs, Damped Harmonic Oscillator + Inc.
triangl,  /home/tas/matroot/matlab/funcs, Triangular peak (width hwhm)
voigt,   /home/tas/matroot/matlab/funcs, Voigt function
dirac,   /home/tas/matroot/matlab/funcs, Dirac peak
===Double peaks==================
gaussx2,  /home/tas/matroot/matlab/funcs, Two Gaussians
lorzx2,   /home/tas/matroot/matlab/funcs, Two Lorentzians
lorgss,   /home/tas/matroot/matlab/funcs, Lorentzian + Gaussian
green,    /home/tas/matroot/matlab/funcs, Green function
===Multiple peaks================
ngauss,   /home/tas/matroot/matlab/funcs, N Gaussians
nlorz,    /home/tas/matroot/matlab/funcs, N Lorentzians
multifunc, /home/tas/matroot/matlab/funcs, MultiFunctions...
===Miscellaneous=================
%rescon , /home/tas/matroot/matlab/funcs, Lorentzian + Lorentzian squared with 3D resolution co
pow,     /home/tas/matroot/matlab/funcs,        Power law (below x0)
pow1,    /home/tas/matroot/matlab/funcs,        Power law (above x0)
ellipse, /home/tas/matroot/matlab/funcs,        Ellipse
background,  /home/tas/matroot/matlab/funcs,    Constant (Background)
strline, /home/tas/matroot/matlab/funcs,        Straight line )ax+b)
quadrat, /home/tas/matroot/matlab/funcs,        Quadratic (ax^2+bx+c)
polynomial, /home/tas/matroot/matlab/funcs,     Polynomial
expon,   /home/tas/matroot/matlab/funcs,        Exponential decay
==Specific=======================
refl, /home/tas/matroot/matlab/funcs/, Reflectivity (N layers)
trix,  /home/tas/matroot/matlab/funcs/trix, 4D-TAS resolution function
trix2,  /home/tas/matroot/matlab/funcs/trix, 4D-TAS resolution function v2
%sdk, /home/tas/matroot/matlab/funcs/, 2 tandem Fabry Perot (SDK)
%couplph, /home/tas/matroot/matlab/funcs, Spin-Phonon coupled function
{Fit Routines}
=== Non-linear least squares routines ====
mf_flsqr, /home/tas/matroot/matlab/nllsq, Levenberg Marquardt gradient (Fast)
mf_glsqr, /home/tas/matroot/matlab/nllsq, Levenberg Marquardt gradient (Graphic)
mf_simplx, /home/tas/matroot/matlab/nllsq, Simplex (Nelder-Mead method)
mf_simplg, /home/tas/matroot/matlab/nllsq, Simplex Graphic (Nelder-Mead method)
```

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

Writing your own load functions

Data files come in many different formats, and a strnegth of **M***fit* is that it allows you to write your own load functions so that you can load any data format transparently. **M***fit* includes load functions for some common (and some not−so−common!) data formats, but you may well want to write your own to cope with special formats.

Usually, you should try first the *General Load routine*, but if you plan to import massively some specific data format files, you may write a faster and adapted load routine.

Function format

Load functions are .m files with the syntax:

`function [x,y,err,xlab,ylab]=xyeload(filename)` The function takes `filename`, the name of the data file including the path, as a parameter, and returns three equal length vectors `x`, `y`, and `err` corresponding to the x, y, and uncertainty−in−y values of the data. The function can also return `xlab` and `ylab`, Matlab strings which, if supplied, are used to label the x and y axes of the graph.

If an error occurs in the load function it should exit with one (or all) of `x`, `y`, `err` an empty vector.

Example:

Here is the code for the x, y, error column ascii file load function (`xyeload.m`). This function loads ascii files with a text header followed by three column numeric data. The header is ignored, except for the last line, which, if it consists of three words, is assumed to give the column labels which are then extracted.

```
function [x,y,err,xlab,ylab]=xyeload(file)
%
% MFIT function [x,y,err]=xyeload(file)
%       MZ 29.11.94
%
% This is a basic load routine for MFIT, illustrating the required
% syntax. The routine takes the name of a data file (including path) as a
% parameter and returns the column vectors x, y, err.

%------------- Open data file---------------------
fid=fopen(file,'r');
if (fid<0)
   x=[];
   return
end

%-------------- Initialize arrays-----------------
data=[];
header='dummy';
text=fgetl(fid);

%============= Load data ==========================================

%------Read header and first row of data -------------------------
istext=1;
while (istext==1)
   [row1 count]=sscanf(text,'%f');
    if isempty(row1)
       header=str2mat(header, text);
      text=fgetl(fid);
    else
```

```
      istext=0;
    end
end

%------ Read data and reshape into matrix --------------------------
data=fscanf(fid,'%f');                          % Read data into vector (for speed)
ncol=length(row1);                              % Use row1 to work out how to reshape
nrow=length(data)/ncol;
if (nrow*ncol~=length(data))
   error('Bad data format');
end
data=[row1'; reshape(data,ncol,nrow)'];         % Reshape vector to matrix
fclose(fid);                                    % close input file

x=data(:,1);
y=data(:,2);
err=data(:,3);


%===== Make x and y column labels ==================================

%-----Try to be clever and make labels from last line of header ----------
s=deblank(header(size(header,1),:));                        % last line of header
i=find(s==9);                                    % replace tabs by spaces
s(i)=32*ones(size(i));
[xlab s]=strtok(setstr(s),' ');                  % x label is first word
[ylab s]=strtok(setstr(s),' ');                  % y label is second word
[elab s]=strtok(setstr(s),' ');                  % elabel is third word

%----- Wrong number of columns? Then just 'x' and 'y'-----------------
if ~(length(elab)>0 & length(s)==0)              % s should be empty if 3 words
   xlab='x';
   ylab='y';
end
```

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

[Simple usage][More advanced multifunctions][Function names]

Building a complex fitting function : Multi functions

Simple usage

1. Loading data
2. Selecting the multifunction
3. The multifunction parameter window
4. Guessing the multifunction parameters
5. Doing the fit

More advanced multifunctions

1. Plot/Hide separate sub−functions contributions
2. Using expressions and user variables as sub−functions
3. Adding user parameters
4. Using advanced sub−functions combinaisons (convolutions,...)
5. Setting a constrain on parameters
6. Using internal or external multifunction storing

Function names

Simple usage

Loading data

To explain the possibilities and usage of the **M***fit* multifunction feature, we shall start by loading an example file with the *General Load routine* . Select in the column selector 'Point number' as X, 'Column 1' as Y, 'sqrt(y)' for error, and 'none' for monitor.

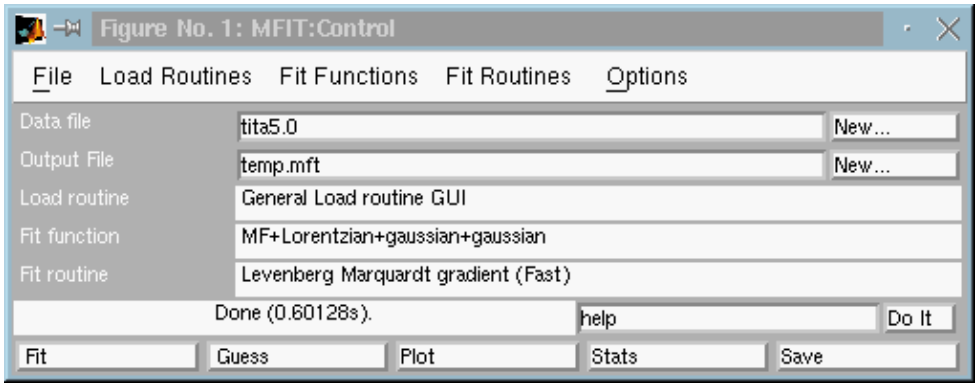


Fig 2 : **M***fit* : The Control window for loading the example file

Selecting the multifunction

Then select the '***Multifunctions...***' item in the 'Fit functions' menu of the **M***fit*:Control window. The window that appears is intended to describe a fit function involving a combinaison of more simple **M***fit* fit functions. In the loaded data (see Fig 4 below), there are 3 peaks. We shall fit them with 2 lorentzians and a gaussian function.

Enter '3' in the 'Number of functions' field, and then click on '*Make function*' button There are now 3 lines `fn=` to describe the multifunction, one for each peak..

To precise the nature of each sub−function, you need to enter the ***matlab name*** of the fit function (the one wich has an extension `.m`).
For instance, the *Lorentzian* function is described in the `lorz.m` matlab file, and the Gaussian corresponds to the file `gauss.m`.
The names of all availables **M***fit* fit functions are given in the `funcs` directory (see installation). You can also get them by typing at matlab prompt (see generated function list below):

```
>> help funcs
```

Then you need to **assemble sub−functions** as a matlab expression (in *Final y=*). In the following example, we just add them.

Finally, the Multifunction window looks like:

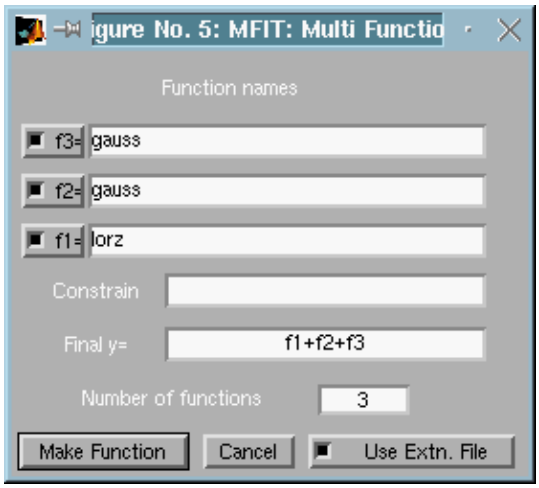


Fig 2 : **M***fit* : The multifunction description, composed of 2 Lorentzians, and a Gaussian.

Re−create the multifunction by clicking on '*Make function*' button. This has to be done each time you change the function number or a function name. An other possibility is to re−select '*Multifunctions*...' in the *Fit Functions* menu.

The multifunction parameter window

The parameter window corresponding to the multifunction appears. Usually, each function has a background parameter. In order to avoid redondant parameters, we fix two of these to 0 (see Fix and free parameters), and obtain the following parameter window:

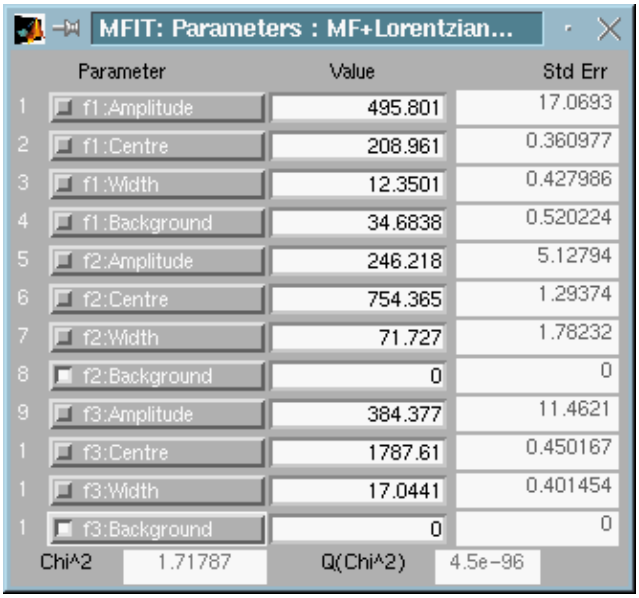


Fig 3 : **M***fit* : Parameter window for a multifunction. Background is only free in one function..

Guessing the multifunction parameters

The guess feature works the same way as for a single fit function. Follow the indications given in **M***fit*:Control message line (on top of buttons), and the additional informations in the matlab workspace (showing what function is beeing guessed). The autoguess also works, and automatically guesses some starting parameters in the case of peak−like fit functions.

Doing the fit

The last step is the fit, performed through the usual procedure (see Fitting).
The final plot shows the result of the multifunction, as well as each sub−function contribution.

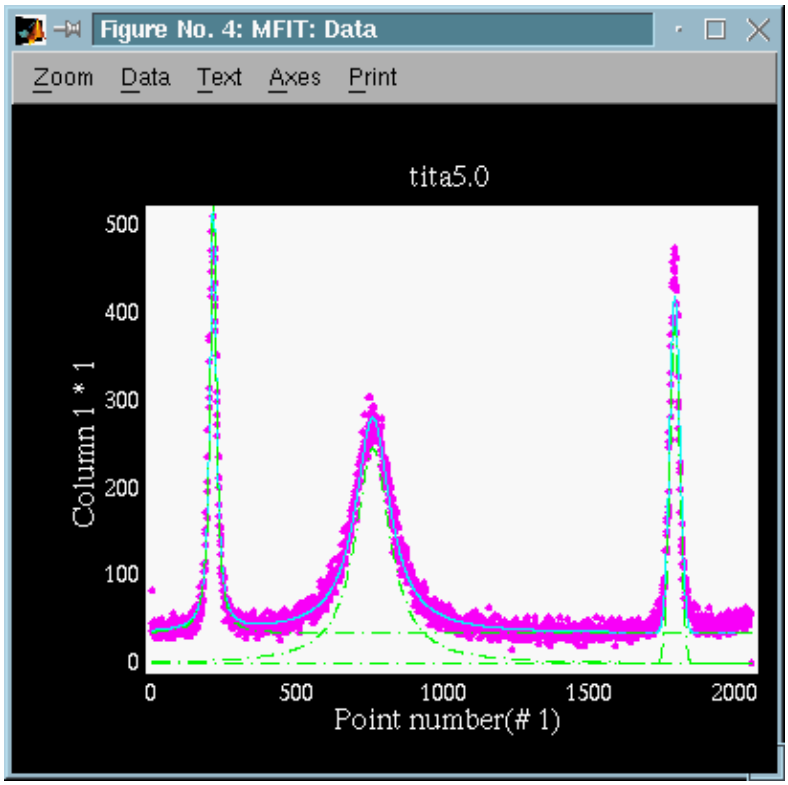


Fig 4 : **M***fit* : The data window, with multifunction and separate contributions.

More advanced multifunctions

In all multifunctions features, you can use the **M***fit* internal variables `x,y,err,selected,fit,p,dp,fixed`.

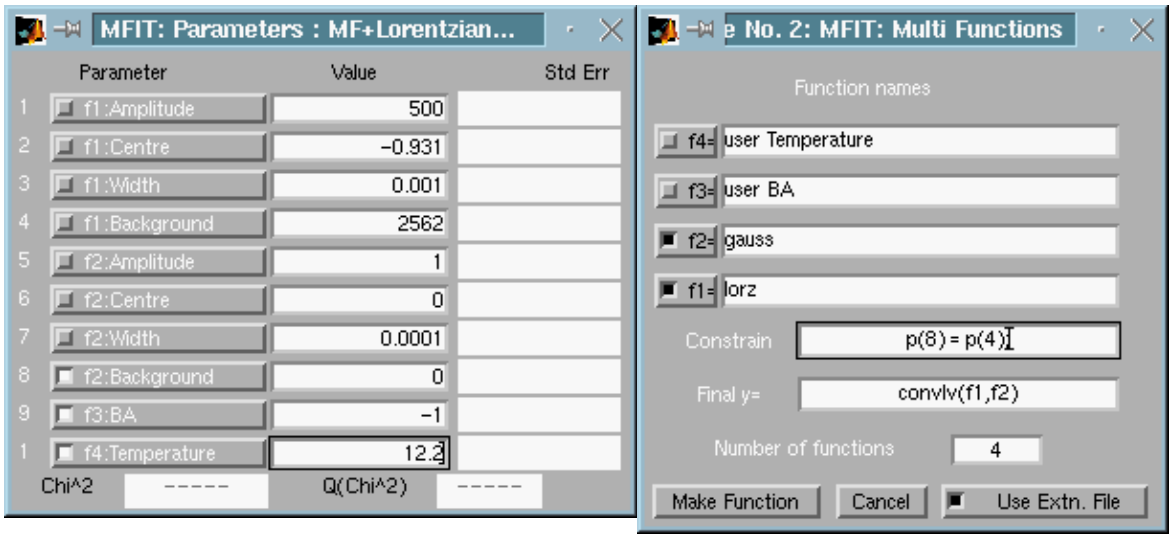


Fig 5 : **M***fit* : A more advanced multifunction usage using parameter variable 'p', and 2 user variables.

Plot/Hide separate sub−functions contributions

If you unselect the 'fn=' butons, the separate sub−functions will not be drawn.

Using expressions and user variables as sub−functions

It is possible not to use a function name for fields `fn=`. Names that are not function names, are evaluated in the following order:

- first as global matlab variables
- as base workspace variables (not global)
- then as matlab expressions using **M***fit* internal variables in 'Multi Functions' window:

For instance, if you used a variabl;e 'a' in matlab workspace, you can type:

'`fn=a;`'.or '`fn=log(a);`'.

Adding user parameters

It is possible to use in multifunctions some additional parameter that are not related to any fit function. For instance it could be a temperature that you measured during an experiment, or a laser power, etc...
This is usefull if you want to gather all informations about a data file in the same MFit parameter file (see Saving results). Then you can plot a parameter versus an other, including your private user parameters (see **M***fit* parameter file importation).

To do so, use lines like:

```
fn=user my_parameter
```

It will appear as a one parameter function. You may fix it, in order not to affect the fitting process. Of course, you can also use it in the '*Final y=*' expression.

Using advanced sub−functions combinaisons

In the '*Final y=*' expression, you may use very complex expression, such as **1D convolutions**:

```
Final y= convlv(f1,(f2−min(f2))./norm(f2))
```

This convolution procedure is provided in the `funcs` directory.
Any matlab expression is possible.

Note: for convolutions, it is *highly* recommanded that the 'f2' signal be **centered** (around x=0), with **no background** (`f2 = f2 − min(f2)`), and possibly **normalized** (`f2 = f2 ./norm(f2)`). The previous example normalizes and substracts the background.

Setting a constrain on parameters

The *Constrain* field is evaluated just before the sub−functions. You can modify the parameters according to your own criteria. Parameters are denoted 'p', and you can also use variable 'x', as these are passed to any fit function.

For instance

```
Constrain p(3) = p(4);
```

will force parameter 3. You'd better fix it in the Parameter Window in that case.

Using internal or external multifunction storing

When unselecting the 'Use External File' button, you ask the multifunction description to be stored into memory. The access is then slower, as a matlab precompilation is not possible. On the other hand, when this button is selected, matlab stores the multifunction into a file. You can even copy (rename it) and install it in the Fit Functions menu, with the *Browse...* item.

Matlab **M*fit*function names** given by `help funcs` command:

```
user      : user defined constant for Multifunc
multifunc : Multi Function handler window
triangl   : Triangular
cusp      : Power law cusp.
dho       : Damped harmonic oscillator
bose      : bose factor
green     : Green function
ellipse   : Ellipse
gauss     : Gaussian
gauss2    : Gaussian Squared
gaussx2   : 2 gaussians
lorgss    : Lorentzian + gaussian
lorz      : Lorentzian
lorz2     : Lorentzian squared
```

```
  lorzx2    : 2 lorentzians
  gaussn    : Gaussian Power n
  ngauss    : N gaussians
  nlorz     : N lorentzians
  polynomial: polynomial
  pow       : Power law y=0 x>x0
  pow1      : Power law y=0 x<x0
  quadrat   : quadratic
  sgauss    : Gaussian plus a sloping background
  strline   : slope/line
  refl      : reflectivity function
  voigt     : Voigt
  lorzn     : Lorentzian Power n
  sdk       : 2 Airy functions product power 3 for SDK
  airyfp    : Airy function for Fabry Perot
  rescon    :  3D resolution convolution of
Lorentzian+Lorentzian squared
  background: constant
  dirac     : Dirac peak
  couplph   : Phonon coupled with pseudo spin system.
  constant  : constant
```

Back to the **Mfit user guide page**

Back to ILL/TAS Matlab Page

Version **4.2**. (1999)

The M*fit* batch language

M*fit* is designed to make interactive fitting as quick and easy as possible. However, there may be times when you want to automate your fitting, especially if, as frequently happens, you need to go through a the same fitting procedure for many different data files. The **M***fit* batch language lets you automate most fitting operations. An additional advantage is that the batch files serve as a record of the fitting procedure used.

In fact, the standard format in which **M***fit* saves fit results (see saving fit results) is as a batch file. This makes it particularly easy to write batch files, as you can perform a fit manually once, save the results, and then edit the resulting file (the example below was produced in this way).

The batch commands can be used in various ways:

- In the **`mfit.ini` file** , the statup section can contain batch commands to execute at **M***fit* launching.
- The **M***fit*:Control window can contain a *batch command field* when setting `ShowBatchLine = 1` in the **`mfit.ini` file**. Press the *Do It* Button to execute command.
- In the matlab workspace, you can type
  `>> mf_batch('command string')` or
  `>> mf_batch('filename')` . Try for instance `mf_batch('help').`
- In the Option menu of the **M***fit*:Control window, the 'Extensions...' item has a *mfit batch command* field

See:

- **Example batch file**
- **Batch file command reference**

Back to the **M*fit* user guide page**
Back to ILL/TAS Matlab Page

Batch language command reference

Below is a list of batch file commands, and an brief description of what they do. Square brackets indicate necessary arguments, eg. `[Parameter number]`, and angle brackets indicate optional arguments, eg. `<Parameter name>` Commands are case insensitive. Scroll down to browse.

plot
pause
axes
guess
exec
vars
help
rem
stop

Back to the **Mfit user guide page**

Back to ILL/TAS Matlab Page

All Mfit variables can be modified (see the vars command). Also, in matlab expressions (see par and exec), you can use the variables x, y, err, selected, p, dp, fixed.

DataFile − Set name of data file to load.

Syntax: `DataFile = [File name]`
Example: `DataFile = temp.dat`
Note: The data file is not actually loaded until the load command is issued. The = sign is optonal
See also: load,vars

DataDir − Set the data directory.

Syntax: `DataDir = [directory name]`
Example: `DataDir = c:\temp\`
Notes: The data file is not actually loaded until the load command is issued . The = sign is optonal
See also: load,vars

FitFuncFile − Set fit function to use.

Syntax: `FitFuncFile = [.m file name]`
Example: `FitFuncFile = gauss`
Notes: omit the .m extension from the file name. The function is not actually loaded until the load command is issued. . The = sign is optonal
See also: choosing a fit function, load, FitFuncDir,vars

FitFuncDir − Set the directory for the fit function.

Syntax: `FitFuncFile = [.m file name]`
Example: `FitFuncFile = gauss`
Notes: omit the .m extension from the file name. The function is not actually loaded until the load command is issued. . The = sign is optonal
See also: choosing a fit function, load, FitFuncFile,vars

OutFile − Set the output file name.

Syntax: `OutFile = [File name]`
Example: `OutFile = temp.mft`
Notes: the output the file to which fit results are saved. New results are appended to the end of the current

output file. . The = sign is optonal
See also: save, OutDir, vars

OutDir − Set the output directory.

Syntax: `OutDir = [Directory name]`
Example: `OutDir = c:\temp\`
Notes: the directory for the output file. . The = sign is optonal
See also: save, OutFilevars

LoadRoutineFile − Set the load routine to use to load data.

Syntax: `LoadRoutineFile = [.m file name]`
Example: `LoadRoutineFile = xyeload`
Notes: omit the .m extension from the file name . The = sign is optonal
See also: choosing a load routine, LoadRoutineDir

LoadRoutineDir − Set the directory for the load routine.

Syntax: `LoadRoutineDir = [Directory name]`
Example: LoadRoutineDir = c:\matlab\mfit3\load
Notes:. The = sign is optonal
See also: LoadRoutineFilevars

load − Load data file and fit function.

Syntax: load <filename <with funcname>>
Example: `load ba300.0,X=#,Y=1 with multibatch` will load that file with multibatch load routine, X as point number and Y as column 1.
Notes: Load the data file specified by the `DataFile` and `DataDir` commands, and load the fit function specified by the `FitFuncFile` and `FitFuncDir` commands. Using the optional `with` keyword will affect the DataFile and LoadRoutineFile.
See also: DataFile, DataDir, FitFuncFile, FitFuncDir

par − Set the value of a fit function parameter.

Syntax: `par [Parameter number] <Parameter name> [Value or expression]`
Example: `par 4 123.45,` or equivalently `par 4 Background 123.45`
Notes: the parameter name can be omitted
See also: fix, free

fix − Fix the specified parameter during the fit.

Syntax: `fix [Parameter number or all]`
Example: fix 3 or `fix all`
Notes: Fixes the specified parameter at its current value. By default all parameters are free.
See also: par, free

free − Free the specified parameter during the fit.

Syntax: `free [Parameter number or all]`
Example: free 3
Notes: Frees the specified parameter at its current value. By default all parameters are free.

See also: par, fix

fit − Begin fit.

Syntax: fit
Example: fit
Notes:
See also:guess

save − Save fit results.

Syntax: `Save <Type>`
Examples: `save`, `save data`, `save curve`.
Notes: Saves fit results to the file specified by the `OutFile` and `OutDir` commands (see above). The type argument specifies exactly what is saved. If type is omitted the results are saved in **M***fit* batch file format (see saving fit results). If `type` is `data`, then the current data is saved in x, y, error colum format. If type is `curve` the evaluated fitting function is saved in x, y column format.
See also: OutFile, OutDir

plot − Update the plot.

Syntax: plot
Example: plot
Notes: Uses the current data, axis limits, fit function, and function parameters
See also:axes

axes − Set axis limits for the current graph.

Syntax: `axes [xmin] [xmax] [ymin] [ymax]`
Example: axes −1 1 0 100
Notes:
See also:plot

pause − Pause execution of the batch file until the user presses a key.

Syntax: pause
Example: pause
Notes: This can be useful to monitor progress of the batch file, or to display data as a 'movie'
See also:plot

vars − list Mfit variables and their current values.

Syntax: vars
Example: vars
Notes: In fact any Mfit variable that is listed with the vars command can be modified.
See also: OutFile, OutDir, DataFile, DataDir, FitFuncFile, FitFuncDir

guess − execute an autoguess to find starting parameters

Syntax: guess
Example: guess
Notes: Signal should contain enough points, and function should use parameters such as *Amplitude, width, center, background*

See also:fit

help − list batch known commands

Syntax: help
Example: help
Notes:
See also:vars

rem − comment

Syntax: rem <string>
Example: rem hello world
Notes:
See also:

stop − abort batch execution

Syntax: stop
Example: stop
Notes:
See also:

exec − execute a matlab expression

Syntax: exec <expression>
Example: exec set(findobj('tag','mf_dp'),'string',0.1)
Notes: you can also call any mfit function
See also:

Back to the **M*fit* user guide page**

Example:

Here is a sample batch file. Any line beginning with a '%' is treated as a comment. See the batch language command reference for details of the commands.

```
% Sample batch file. This section (apart from the comments) was
% produced by Mfit, by saving the results of a manual fit.

% Set data file and directory
datafile = RB_27502.DAT
datadir = P:\RBZNCL\X20\ORD2A\

% Set fit function and directory
fitfunctionfile = LORZ2X2
fitfunctiondir = C:\USERS\MARTIN\MATLAB\MFIT2\FUNCS\MARTIN\

% Set the output file
outputfile = fit1.mft
outputdir = P:\RBZNCL\X20\ORD2A\
```

```
% Set the load routine to use to load data
loadroutinefile = xyeload
loadroutinedir = c:\users\martin\matlab\mfit2\load\

% Set the fit routine (algorithm used for fitting)
fitroutinefile = mf_lsqr
fitroutinedir = c:\users\martin\matlab\mfit2\nllsq\

% Load the data file and fit function set above
load

% Set starting values of parameters
%-------------------------------------------------
par 1 Amplitude_1 1.898640e+004 5.066810e+002
par 2 Centre_1 -1.393110e+000 1.376960e-005
par 3 Width_1 9.5e-4
par 4 Amp_2/Amp_1 0.1
par 5 Offset_2 1.227e-003
par 6 Width_2 1.7e-003
par 7 Background 24
par 8 Index 275.02
%-------------------------------------------------

% Fix some parameters
fix 3
fix 4
fix 5
fix 6
fix 7
fix 8

% Do the fit
fit

% Save the results to the output file specified above
save

% Set a new datafile
datafile=rb_28002.dat

% Load the new file (nothing else needs to change)
load

% Set the 'Index' parameter (all other parameters have the values
from
% the end of the last fit)
par 8 280.02
fit
save

% New data file
datafile=rb_28502.dat
load
par 8 285.02
```

```
fit
save
```

Back to the **M*fit* user guide page**

Back to ILL/TAS Matlab Page
