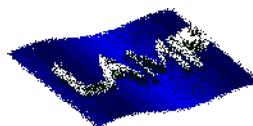


presents

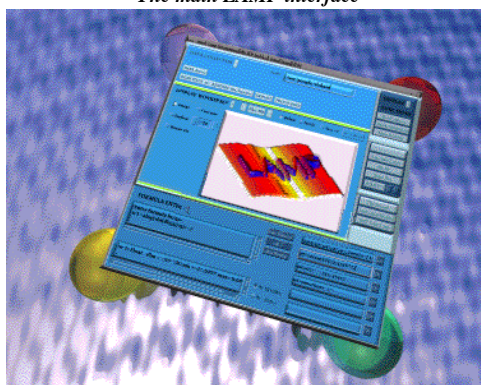


THE LAMP BOOK

LAMP by D. Richard, M. Ferrand, G. J. Kearley.
WWW Manual created by A. D. Bradley

The LAMP Book is an on-line help and reference manual for the Large Array Manipulation Program, developed initially for the treatment of data obtained from neutron scattering experiments here at the Institute Laue-Langevin. LAMP has become more general purpose by adding common features and is seen as a GUI-laboratory for data analyses based on the IDL language.

The main LAMP interface



LAMP provides a predictable and intuitive graphical user interface which integrates scientific visualisation with an enhanced data language. Many high level modules are predefined to enable interactive data analysis and visualisation of 2D , 3D data and atomistic representations.

LAMP also provides for :-

- Input and output of various data-type files with *CUSTOMIZED MODULES*.
- Expansion and simplification of data with *USER-MACROS* in Interactive Data Language IDL.
- General fitting interfaces for *2D and SIMULTANEOUS FITS* with CONSTRAINTS.
- Visualisation of calculated *PHONONS* and molecular *VIBRATIONS*.
- Cristal and Magnetic *STRUCTURES* with propagation vectors *EDITOR*.
- Miller planes in the *CRYSTALLOGRAPHIC CELL* easily animated to help visualise local structure.
- *RECIPROCAL SPACE* densities calculated from multiple instrument acquisition scans.
- Automatic *ELECTRONIC LOG BOOKS* updated in html format by a right click on images.
- Lamp , Nxml , NeXuS *FILE BROWSER* with snapshot images.
- Direct link to a powerful *IMAGE-MANIPULATION* module, Scan.
- The live control of an experiment with *GEORGE*.
- The same behaviour on *Unix* Motif platforms, MS *windows*, *MacIntosh*.
- A distribution package for the scientific community with a *FREE embedded IDL license*.
- A *LIVE-UPDATE* feature to keep your application always up-to-date.

Ftp://Ftp.ILL.FR/pub/cs
Http://Barns.ILL.FR/

[Open Manual](#)

[OPEN TOF MANUAL \(local\)](#) [OPEN TOF MANUAL \(ILL\)](#)

[GEORGE on D7](#)

[D7 Lamp Book](#)



If you just have [downloaded](#) the lamp package, click the *README* below right now.

[Installation README](#)

[NEWS](#) over the lamp package and last changes (text file). mail queries to lamp@ill.fr



LEGAL NOTICE: The LAMP package is distributed in the public domain. If you find this application useful, you may send an electronic mail message to lamp@ill.fr. We would gratefully appreciate any feedback on malfunctions.

The reference in your Publications should be:

() LAMP, the Large Array Manipulation Program. http://www.ill.fr/data_treat/lamp/front.html

Using The LAMP Book

This manual can be used in a number of ways:-

- Use the [main contents](#) below and the more specific contents at the start of each page.
- Use the [imagemap](#) on the [Introduction Page](#) to take you to relevant pages.
- Go through it page by page using the buttons at the bottom of each page. Use your browsers **Back** and **Forward** too.

[Front Page](#)

Main Contents

[1. Introduction](#) - Describes *workspaces* and contains a [clickable image](#) of the basic operations that LAMP performs.

[2. Reading in Data](#) - Instructions for starting LAMP, using the *Data Collector Area*, [saving](#) and emptying workspaces.

[3. Displaying Data](#) - Using the *Display Workspace Area* to plot data, perform cursor operations, set preferences, and print.

[4. Special Display Interfaces](#) - Basics of the separate LAMP interfaces *Scroll Spectra*, *Radial Integration*, *Superplot*, *STR-fit* *Gk-fit* *Qens-fit*, *Atomic structure editor*, *Miller planes*, *phonons and vibrations animation*, *Tomography* ...

[5. Manipulating Workspaces](#) - Using from Layout features the *Formula-Entry Area* and the *Do/Xbu input text* to write and execute fomulae and macros, obtain information about data, save and exit LAMP sessions.

[6. Common Workspace Functions](#) - The syntax used to manipulate data, from maths functions and normalising, to plotting data. Help from Idl contains a decription of all available native functions and all about syntax.

[7. Other Display interfaces and "TOF legacies"](#) - Filling HKL reciprocal space by series of monicristal diffractions , Magnetic scattering cross-section example , George-Layout , Mask & Group , Tof reduction legacies.

[8. LAMP Macros](#) - Making *command files* and compiled IDL procedures that can be executed in LAMP.

[9. Frequently Asked Questions](#) - Plots, printing, instrument-spectrum numbers and what to do if LAMP shows no response.

[Front Page](#)[Next Page](#)

1. Introduction

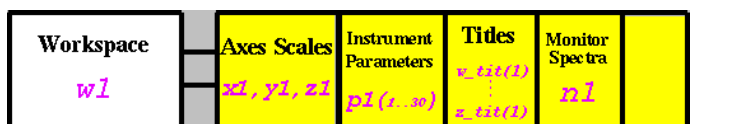
[Main Concepts](#) | [LAMP's Basic Operations \(clickable image\)](#)

Main Concepts

LAMP allows you to read data into a buffer: one of **40 workspaces** (extendable to 80) which can then be handled using the LAMP interface to perform the functions outlined above.

Workspaces are arrays that can contain any type of data. Typically, these are whole experimental runs rather than just single spectra. LAMP also predefines arrays for the instrument parameters, axes information, descriptive text, monitor spectra, errors, ... which are all tied to the individual workspace.

This structure can be visualised as follows:

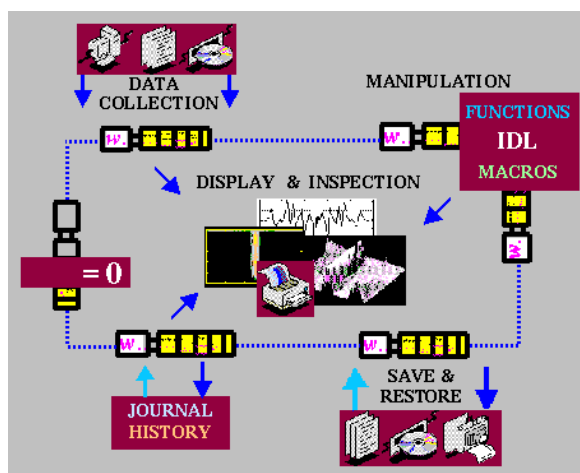


When you read data into a workspace (LAMP predefines 40 workspaces, called **w1...w40**, for your use) it automatically adapts to the dimensions of the data. Also, when you pass data from one workspace to another LAMP automatically passes the tied arrays (scales, parameters etc.) too.

LAMP's Basic Operations (clickable image)

The basic operations of LAMP are summarised in the following diagram.

This diagram is also an *imagemap* which means that you can click on any area you are interested in and automatically be taken to the relevant manual entry.


[Previous Page](#)
[Main Contents](#)
[Next Page](#)

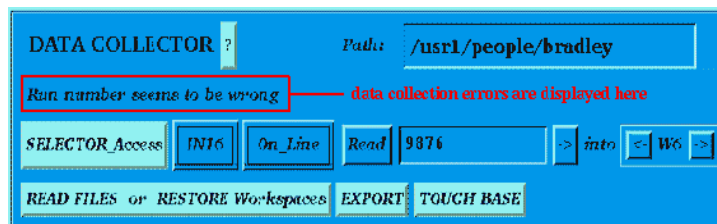

2. Reading in Data

[Running LAMP](#) | [Imagemap](#) | [Selecting Instrument](#) | [Selecting Data-Base](#) | [Selecting a Workspace](#) | [Selecting a Run-Number](#) | [Reading the Run](#) | [Other Methods of Reading Data](#) | [Reading Treated Data](#) | [Saving Workspaces](#)

The Lamp Book describes the classical lamp interface selected from "Lamp/Layout" item in the menu-bar.

To start Lamp see "Installation README" in the front page of this manual.

The correct instrument and data-base may already be set, if not you must select **DATA Instrument** and the Data Collector Area will then appear as :



Imagemap - click on any point to go to the relevant paragraph.

- **Path is the working path**

This is the current path for default inputs and outputs.

- **Selecting Instrument/Data-Type**

If the instrument button (IN16 above) does not indicate the desired instrument then press this button. A pull-down menu of instrument-groups appears. Select the desired group then the desired instrument, or data-type eg. [INX](#) (see [Reading Treated Data below](#)). To add or modify a Data-Type then press the button labelled **Customize**

- **Selecting Data-Base**

The button to the right of the instrument-button contains the name of the data base. Press this button and a pull-down menu appears which normally offers:

- *Current Cycle...* Data for the current cycle on the main data-base
- *Previous Cycle...* Data from the previous cycle on the main data-base
- *Current Path...* Data will be read from the path defined in the field at the top-right of the window. This path is editable.
- *Etc...*

To add or modify a data-base Link then press the button labelled **Customize**

- **Selecting a Workspace**

The workspace selector is at the far-right of the Data Collector area. At start-up w1 will be selected. You may change this by pressing the "<" and ">" buttons on the interface.

- **Selecting a Run-Number (or a file-name)**

Enter the desired run-number in the window (containing run 9376 above). You can increment the run-number using the ">"(+1) button, which causes an automatic read and plot.

If you enter the form "9376 : 9380" or "9376 > 9380" or "9376+9378-9380", functions rdand or rdsum or rdopr are called (see [Other Methods of Reading Data above](#))

- **Reading the Run**

To read the data into the chosen workspace click on **Read** (or hit carriage return in the run-number window), the instrument parameters, monitor spectra etc. are automatically read into their corresponding arrays for that workspace (see [Main Concepts](#)).

- **Other Methods of Reading Data**

The command `w1=rdrun(1234)` can be entered into the [formula-entry window](#) and "Do input commands". This example reads run number 1234 into w1 (and the monitor spectra into n1, axes scales into x1,y1 etc.).

The command `w1=rdsum(1234,1237)` would sum together runs 1234 to 1237 into w1.

The command `w1=rdand(1234,1237)` would join together runs 1234 to 1237 into w1.

The command `w1=rdopr('1234+1237')` would sum runs 1234 and 1237 into w1.

These forms of read-in are useful for [command files](#) and [compiled macros](#).

When used in macros you can add the keyword **DATP= datp** to get all data parameters in the structure variable [datp](#) ex: `w1=rdand(1234,1237,DATP=datp)`.

The **IMPORT file Workspace** button from the *File menu-bar* allows you to read saved data from the path selected (see the File item in the menu-bar).

The **SELECTOR_Access(Self...)** button is expected to be useful for reading in data series (see the File item in the menu-bar).

- **Reading Treated Data (example for TOF)**

Data files output by the time-of-flight treatment program, **INX**, can be read into LAMP by selecting **INX** from the instrument menu. Then select **current path** from the data-base menu. If required change the path in the path-window to point to the data, then enter the filename in the read window. See [Saving Workspaces below](#) for [saving data files](#).

- **** Emptying Workspaces**

LAMP uses dynamic memory-allocation so that empty workspaces take no space. You can empty workspaces that are no longer required by setting them equal to zero eg. typing `w6=0` in the [formula-entry window](#) would empty w6 (and the tied arrays n6,x6 etc.).

- **** Saving Workspaces**

The **EXPORT workspace** button in the *File menu-bar* allows you to save data in the path selected and gives a choice of formats. The **LAMP format** is the default setting, which saves the data in either binary or ASCII file and all the associated parameters, scales, titles, history etc., in a second file which is always a text file. This format has the advantage that the workspaces and their associated arrays can be restored in their entirety. Other formats (tiff.png etc.) are mostly intended for reproducing graphics.

Binary formats are:

- XDR.HTM for data exchange between different platforms (UNIX, LINUX, WINDOWS..),
- F77 for FORTRAN unformatted read.
- HDF for data exchange between different instituts (NeXus),

A snapshot image file (192x192 byte array) is produced as binary stream data.

Data can be saved in any desired format using a user-macro, eg. to write workspace w1 to an **INX file**, say *myfile.dat*, the macro `dumpx.pro`, can be used as follows

```
dumpx,w1,'myfile.dat'
```

which is typed in the [formula-entry area](#).

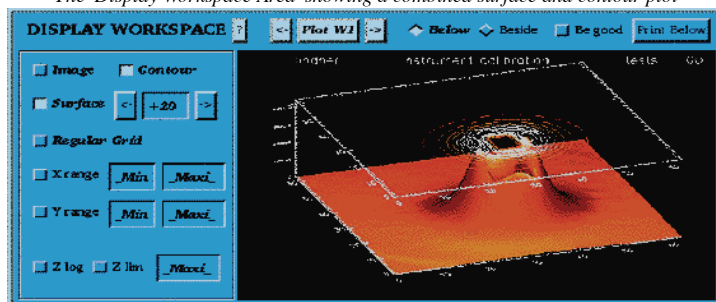
Some on-line help can be obtained by pressing the '?' button.


[Previous Page](#)
[Main Contents](#)
[Next Page](#)


3. Displaying Data

[Imagemap](#) | [Plotting a Workspace](#) | [Below and Beside](#) | [Colours](#) | ["Be Good" Button \(Set Preferences\)](#) | [Image, Contour and Surface](#) | [Ranges](#) | [Zoom](#) | [Cursor](#) | [Printing](#)

The 'Display Workspace Area' showing a combined surface and contour plot



Imagemap - Click on any point to go to the relevant paragraph.

- **Plotting a Workspace**

By default the number of the last workspace that was modified appears in the **Plot W#** button. You can decrease or increase this number by pressing the "<" or ">" buttons, respectively. When the button is pressed a plot is made according to the setting of the other buttons.

- **Below and Beside**

By default the small plot-window (shown above) will be used for graphics, this is the **Below** setting. If **Beside** is selected plots will appear in a new window which can be rescaled and scrolled, each new plot (made by pressing **Plot W#** again) makes a new window which enables plots to be compared easily and stored as icons if desired. The **Beside** window also contains options for [printing](#), changing colours and annotation (which have their own help files).

- **Colours**

Colours can be changed by pressing the **Load new Colors** button on the right of the main LAMP interface or the **Edit menu_bar** item (and also through the **Beside** window described above). This brings up the "XLOADCT" utility which changes the colours of all plots interactively. *Be careful not to plot black on black.* If you experience problems type `loadct,27` in the [formula-entry window](#) (or select *Eos B* in "XLoadCT"), this colour scheme will display most types of bidimensional data and also gives a white background to plots.

- **"Be Good" or "Options->Plot..." Button**

This button brings up a **set preferences** window, which allows you to change the titles (displayed on plots and printouts), **default printer** and various plot options, such as plotting surfaces with shading from a light source (default setting) or *vectors* which can give a multi-plot effect.

- **Image, Contour and Surface**

Any combination of these display styles can be selected (and displayed by pressing **Plot W#** again) the Z rotation angle for surface plots is given in the box next to the **surface** button. This can be modified as desired then plot again or use the right-button of the mouse to dynamically choose an angle; the view angle (X rotation) is set in the **Plot Options** menu.

Remember that contour plots of noisy data can take a long time to generate (particularly with a log scale); *Image* is usually very fast.

- **Ranges**

Once the **"Range etc.."** button has been pressed the **X, Y range** and **W lim** buttons appear (shown [above](#)). If a button is deselected the full range will be taken, otherwise the desired range can be entered into the windows beside the buttons. There is also a **W log** button to display the intensities logarithmically (revealing small features).

N.B. If you change an X-scale from [channels to energy \(t2e\)](#) you may get a shock if you have selected an X-range based on channels! There is also a button to select a **Regular Grid** after converting to energy ie. to give a linear scale in energy rather than channels.

- **Zoom**

To zoom in on a region of a plot press the left mouse-button and drag out a rectangle enclosing the area of interest. When you release the mouse-

button the zoomed area will be plotted. Unzoom by pressing *Plot W#*, or *Replot* in the *Beside* window.

Data in the zoomed area can be saved into a workspace (eg. w4) by entering **trap,w4** in the [formula-entry window](#).

- **Cursor**

Press the left mouse-button with the cursor-arrow in the *Below* plot window and the values of **X**, **Y**, and **I** (Intensity,W) will appear above the plot. Obviously the cursor cannot work on 3D *surface* plots.

Note: For simple x,y plots (eg. single spectra) the X and Y values can be displayed **on the plot** using the left and middle mouse-buttons respectively. Press the right mouse-button to have access to your log-book. The image as other features should be included in this log-book (Html) which can be transformed in PDF file at the end of the experiment or analyses using your browser.

- **Printing**

Normally the correct printing device will be set so that you simply press the **Print Below** button to obtain a hard-copy. If no device is set LAMP will make a postscript file in your current *path*. The printing device is set in **Plot Options**. Also the *Beside* plot can be printed if the printer is set in **Plot Options (Be good)** otherwise just a postscript file is made.

See [Graphics](#) on [Common Workspace Fuctions](#) page for direct methods of plotting data and **setting plot ranges** using the formula-entry window.


[Previous Page](#)
[Main Contents](#)
[Next Page](#)



4. Special Display Interfaces

[SuperPlot](#) - [Overplotting](#) | [Scaling and Zooming](#) | [Cut applied to ...](#) | [Plot Style and Printing](#) | [Tips](#)

[Radial integrations](#)

[Unroll images](#)

[Curve fit](#) - [Str-Fit](#)(complete and generous) | [GK-Fit](#)(limited but easy to use) | [Qens-Fit](#)(for quasi-elastic spectra)

[Tomography](#)

[Magnetic structure](#) - [Editor](#) | [incommensurate example](#) | [commensurate example](#)

[Atomistic representation](#) - [Miller planes/isodensities](#) | [Molecules/displacements](#)

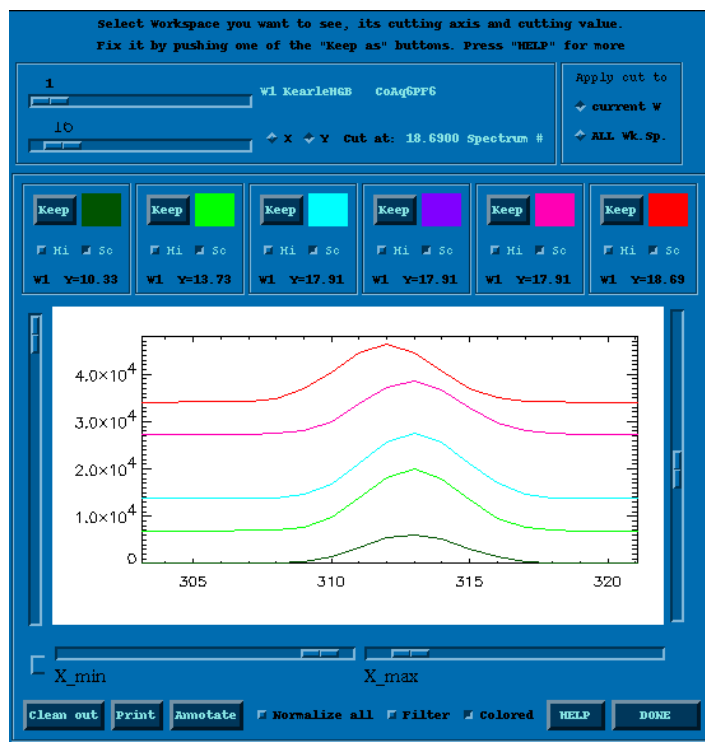
[Scroll Spectra](#) - [Scrolling](#) | [Scaling and Zooming](#) | [Plot Style and Printing](#) | [Tips](#)

The DISPLAY FUNCTIONS list on the right of the classical LAMP interface and the TOOLS list in the Menu-bar contain buttons to select the following interfaces :-

SuperPlot

This interface provides a window for plotting several individual spectra. The current workspace number used in Lamp is automatically used as the **default workspace**. You can change it with the *workspace slider*. The **cut axis** and the **cutting value** can also be easily choosen using *X or Y button* and *cutting value slider*.

The SuperPlot window displaying five spectra from 1 workspace



- **Overplotting**

Select the desired cutting value and cutting axis of the workspace, the requested plot will appear. You can keep this plot by pressing the *Keep as* button, at the top of one of the six local buffer control panels. By selecting another cutting value or cutting axis or reselecting a workspace, you create another plot. To get rid of it, just select an empty workspace.

- **Cut applied to ...**

There are two exclusive buttons : *Current workspace* and *All workspaces*.

In **current workspace** mode, the cut is only applied to the current workspace, in **All workspaces** mode cut is applied to every buffer, subsequent plot is made only if possible (Only if the cut value and axis exist in workspace)

- **Scaling and Zooming**

By default SuperPlot will *autoscale* to suit all spectra in the window.

All spectra are scaled by default because the *scale* button is set for each workspace. The scale toggle button allows you to specify whether or not a plot can be scaled.

The two *bottom sliders* allow you to perform an x range zoom by setting minimum and maximum values to be plotted on the X axis.

- **Plot Style and Printing**

The *Linestyle* for plotting depends on which buffer is chosen. Until a plot is stored in a buffer, the current workspace plot has a thickness of 2. If the coloured mode is set, the lines are in color except the current workspace plot, which is still black, but with a thickness of 1. plots can be printed using the *PRINT* button if the printer is set in *Plotting preferences*, otherwise just a postscript file is generated. If you want to print an annotated plot, use the annotate print option.

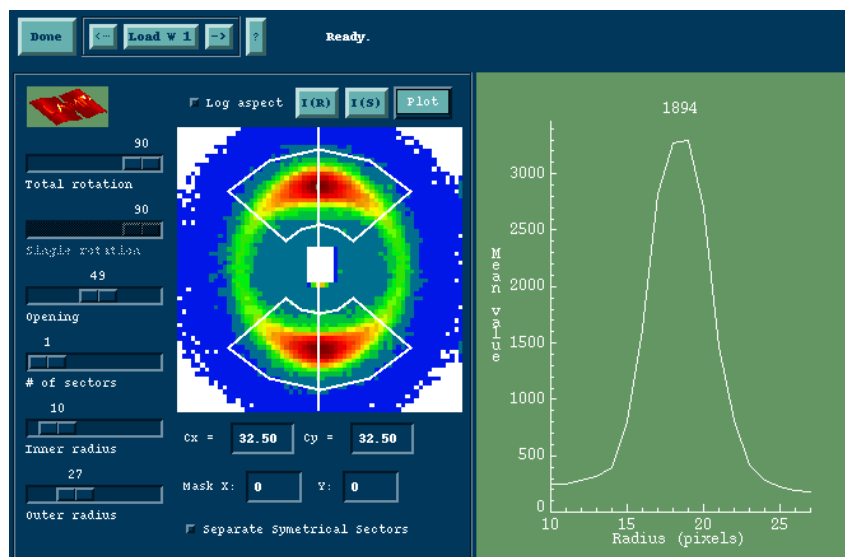
- **Tips**

If you change from a cut applied to the *current workspace* mode to the *All workspace* mode, the buffered workspaces are preserved. For example, plot W1 at x0, W2 at x3 and W3 at y5. Then change to "All W" mode, you get W1, W2 and W3 cut at the current cutting value and axis, which can be changed. Go back to current workspace mode, you will have back your W1 at x0, W2 at x3 and W3 at y5 plots (nice, isn't it !)

Also in the DISPLAY FUNCTIONS area on the right of the main LAMP interface are buttons to select a *Radial Integration* programs, a curve function fitting program (*GK_Fit*), *SCAN* for more elaborate image manipulation, and [Load new Colours](#).

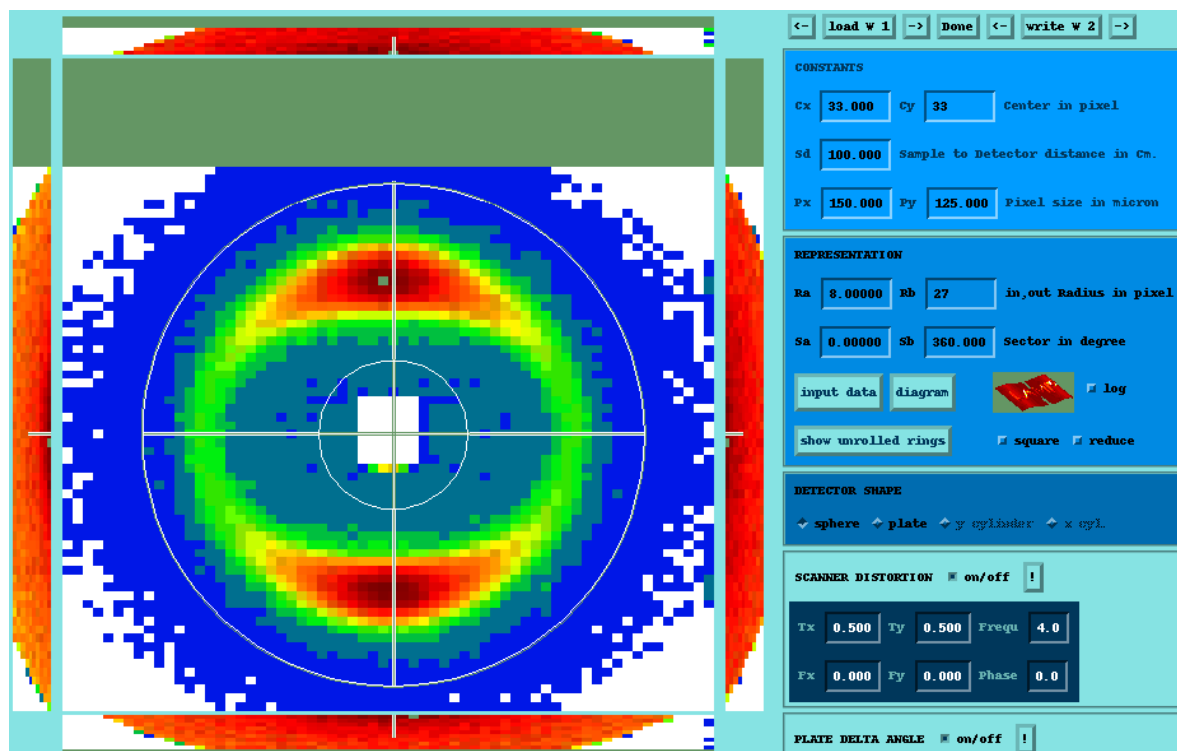
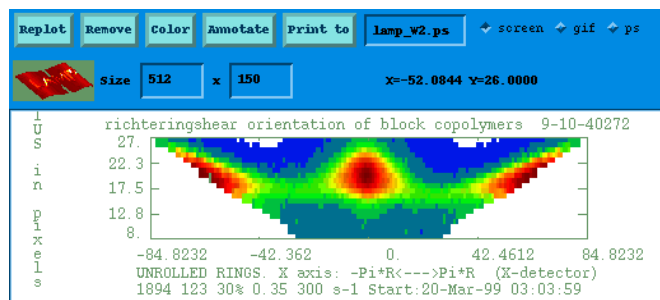
RADIAL INTEGRATIONS

This interface provides a window for radial and azimuthal integrations using user defined sectors.



UNROLLING RINGS

This interface provides a window for unrolling large images. The analysis becomes easier since integrations are simple projections.



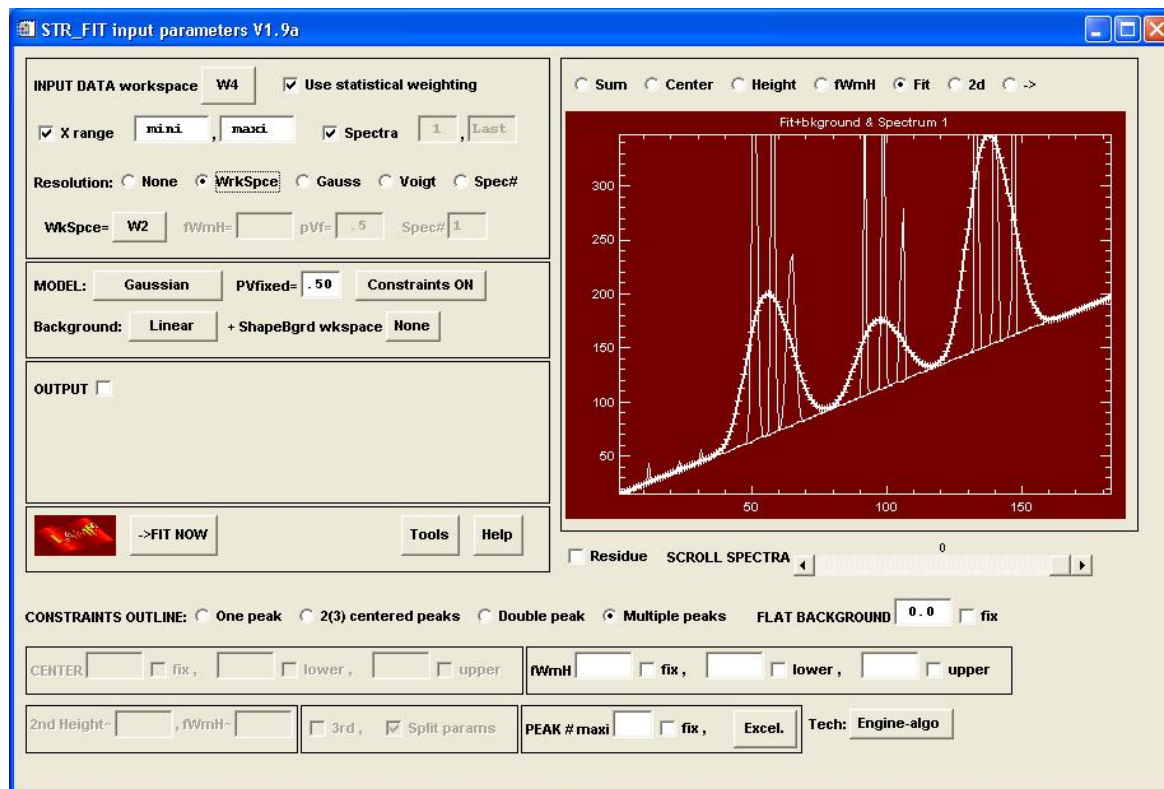
CURVE FITTING with Str_Fit

Originally developed for analysing strain scanning acquisitions in which a Bragg peak is measured as a function of the position in the sample. The diffraction peak in each pattern is fitted with a given theoretical lineshape (e.g. pseudo-Voigt), including convolution with a resolution function.

The fitted data, residuals and parameters can be viewed in different ways.

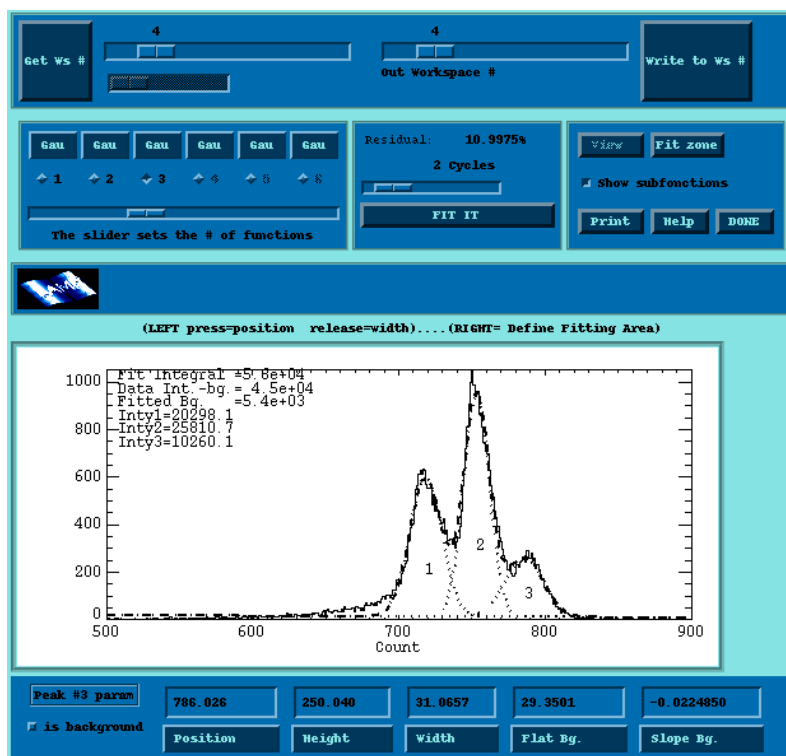
Today the program aims to give users a general fitting interface in which we can:

- Construct a model by entering own formulae and using predefined functions.
- Constraint and tie parameters for independent spectra or 2D spectra (simultaneous fitting).
- Fit parameters in one go using previous spectra results.
- Choose a resolution as a function or numerical input spectra.
- Save models and results. Call "Str_fit" from a macro without using the interface.



CURVE FITTING with GK_Fit

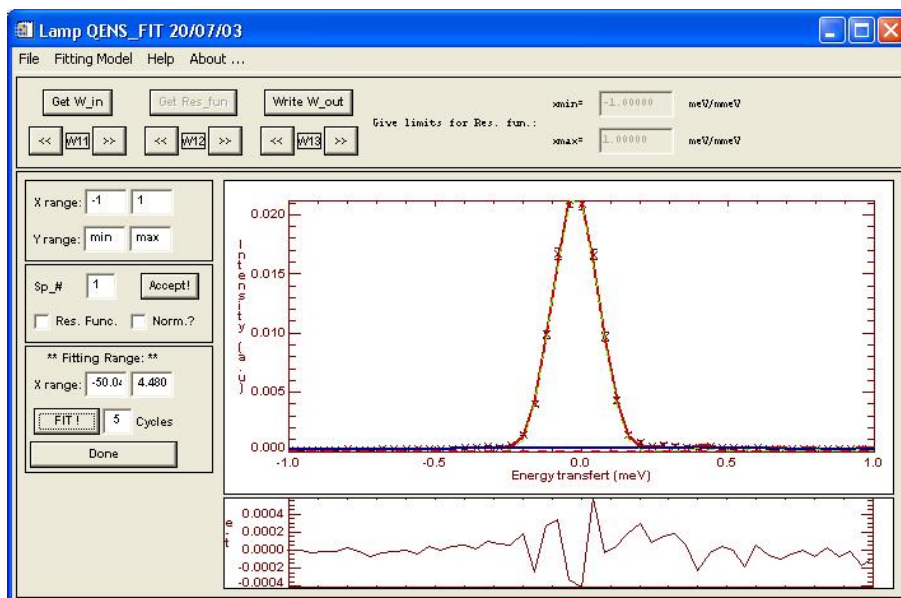
From one to six Gaussian & Lorentzian functions can be fitted from this interface.



CURVE FITTING with Qens_Fit

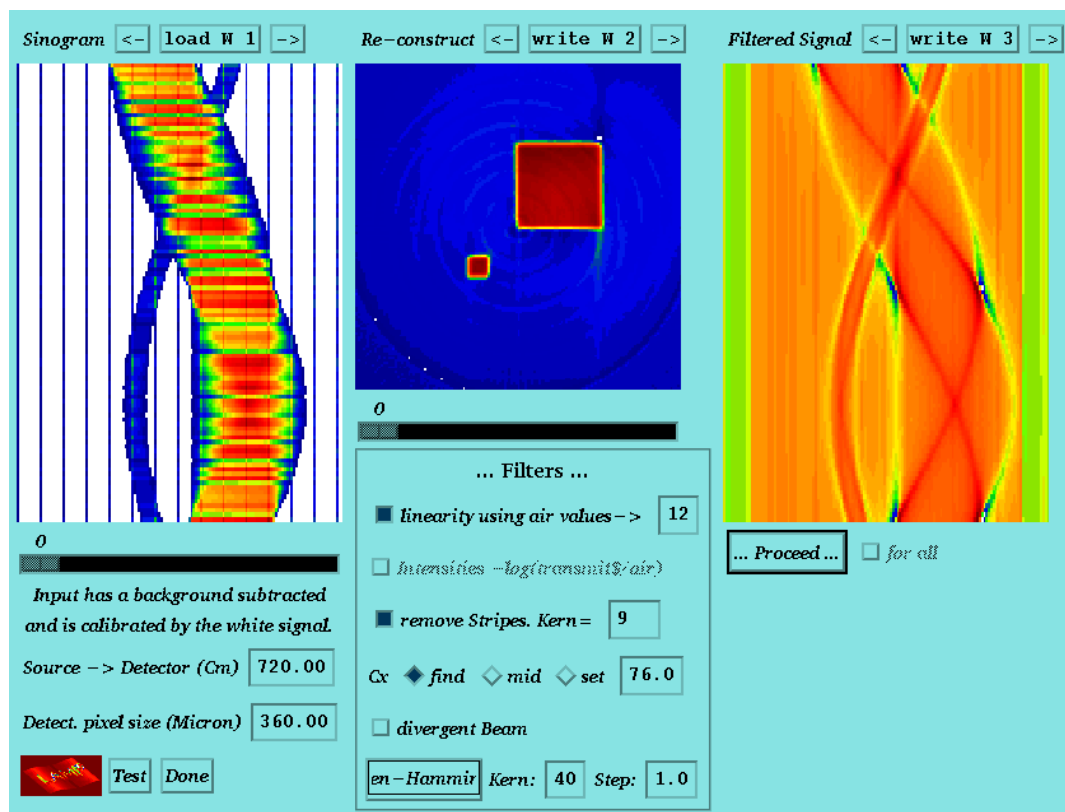
Used to analyse quasi-elastic spectra. Inelastic excitations can also be treated. The input for "Qens_fit" consists of two workspaces, one containing the data as a function of Q (or temperature, pressure etc), the other containing the corresponding set of resolution functions, and a fitting model (typically a delta function for the elastic peak and lorentzians and gaussians for the quasi/inelastic features).

Once one spectrum has been fitted the remaining spectra can be fitted in one go and parameters can be fixed for all spectra. Instead of fitting independently each spectrum, a physical model can be imposed, for example for diffusion on a sphere. Models are added when needed.



TOMOGRAPHY

A straightforward interface for reconstruction.



MAGNETIC STRUCTURE

An editor for magnetic structures representations with the possibility to add planes and do animations. MAGDRAW reads the nuclear and magnetic structures following Rietveld refinement in standard programs like "FULLPROF". The structure is displayed as a 3D object that can be easily rotated.

... GLOBAL CELL PARAMETERS ...

Propagation vectors: Number of cells along a: b: c:

Vectors input: ☒ Cart. M ☐ Sph. M ☐ Dynamic

a = b = c =

Fourier coef. relative to cell atom: ☒ Sk ☐ Tk

alpha = beta = gamma =

Modulation: ☒ Phase ☐ Amplitude

Title = Lattice =

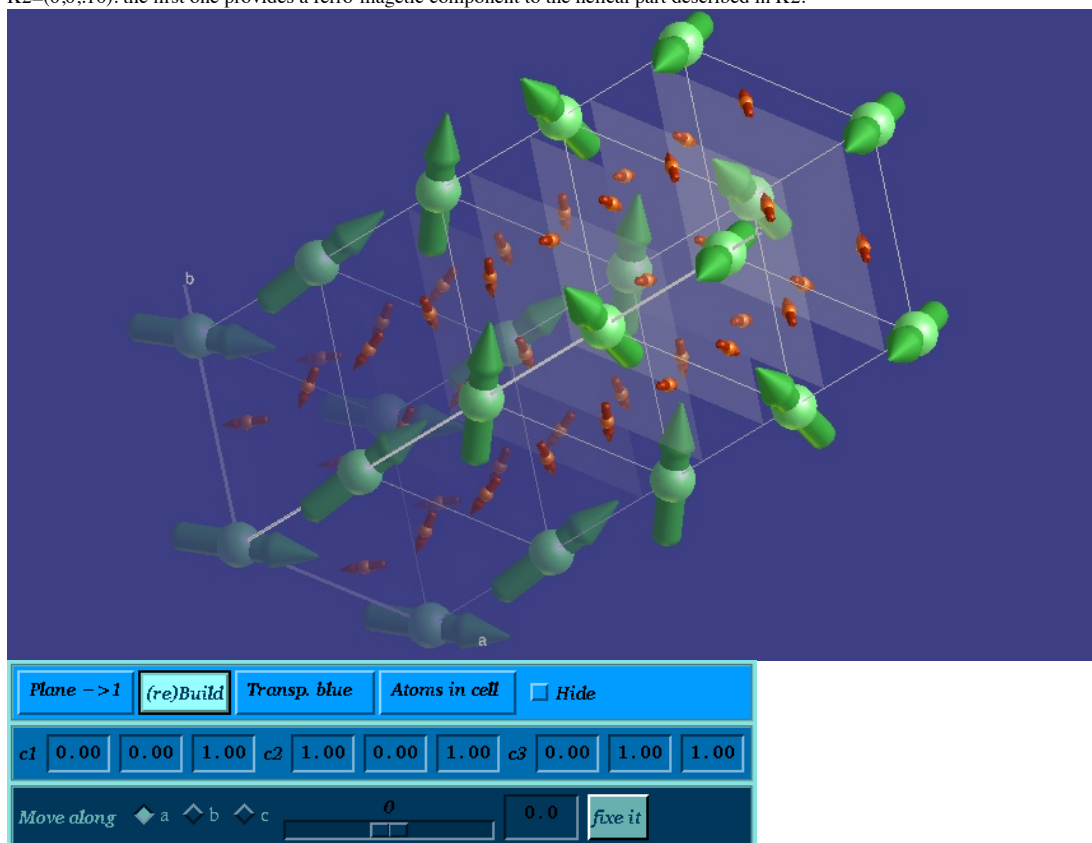
... ATOMS & MAGNETIC MOMENT / DISPLACEMENT INPUTS ...

	Symbol	Atomic coordinates	site	Real vectors -> K1	Phase	Width	Color
1 Dysprosium	Dy	0 0 0	1	6.35		1.00	
2 Manganese	Mn	0.50 0 0.25	1	1.97	0.55	1.00	
3 Manganese	Mn	0 0.50 0.25	1	1.97	0.55	1.00	
4 Manganese	Mn	0.50 0.50 0.25	1	1.97	0.55	1.00	
5 Manganese	Mn	0.50 0 0.74	1	1.97	-0.2	1.00	
6 Manganese	Mn	0 0.50 0.74	1	1.97	-0.2	1.00	

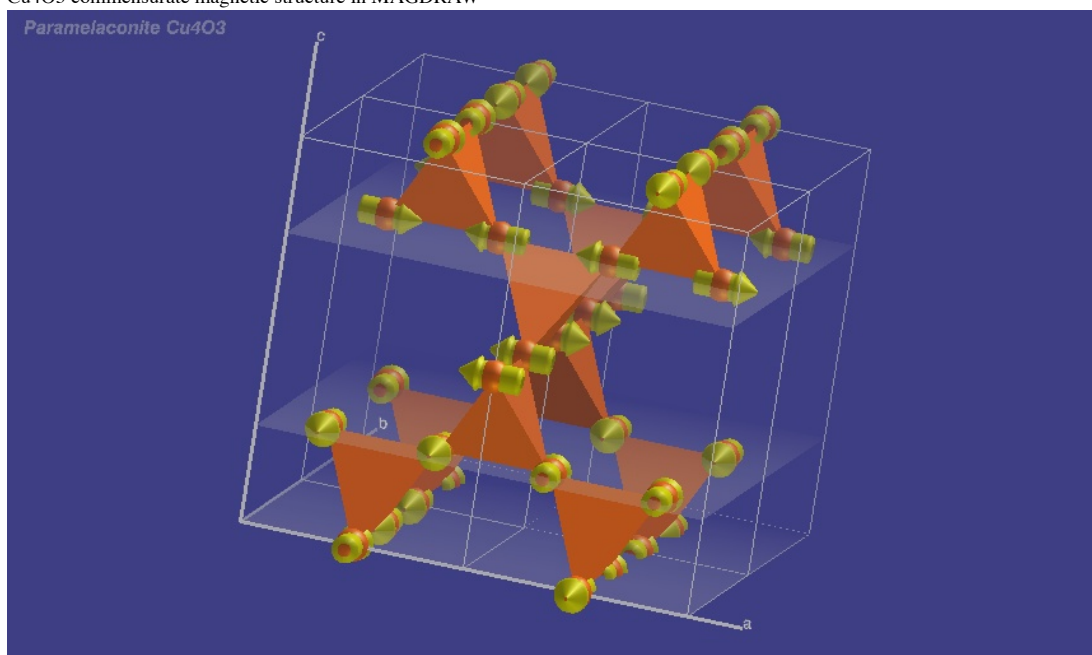
File name =

Workspace:

DyMn6Ge6 incommensurate magnetic structure in MAGDRAW. The magnetic structure is characterized by two propagation vectors $K1=(0,0,0)$ and $K2=(0,0,.16)$. the first one provides a ferro-magnetic component to the helical part described in $K2$.



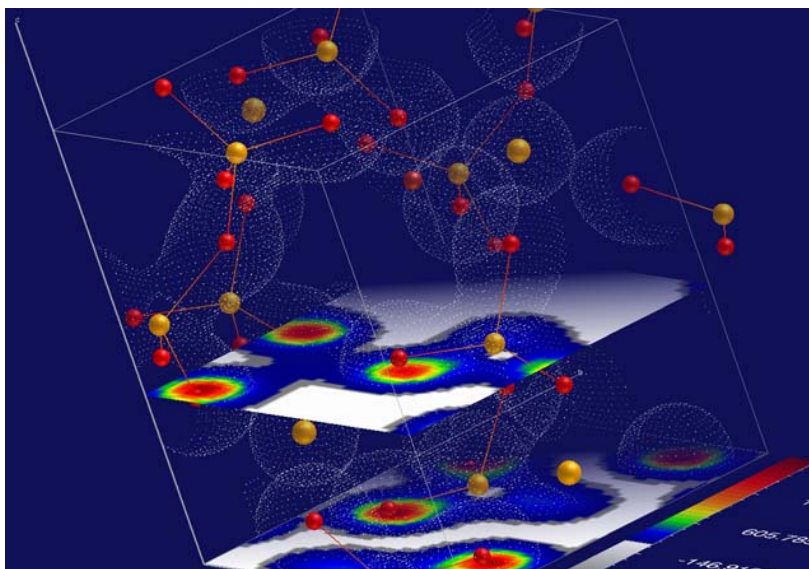
Cu4O3 commensurate magnetic structure in MAGDRAW



MILLER PLANES, ISOSURFACE OF ELECTRON OR SPIN DENSITY

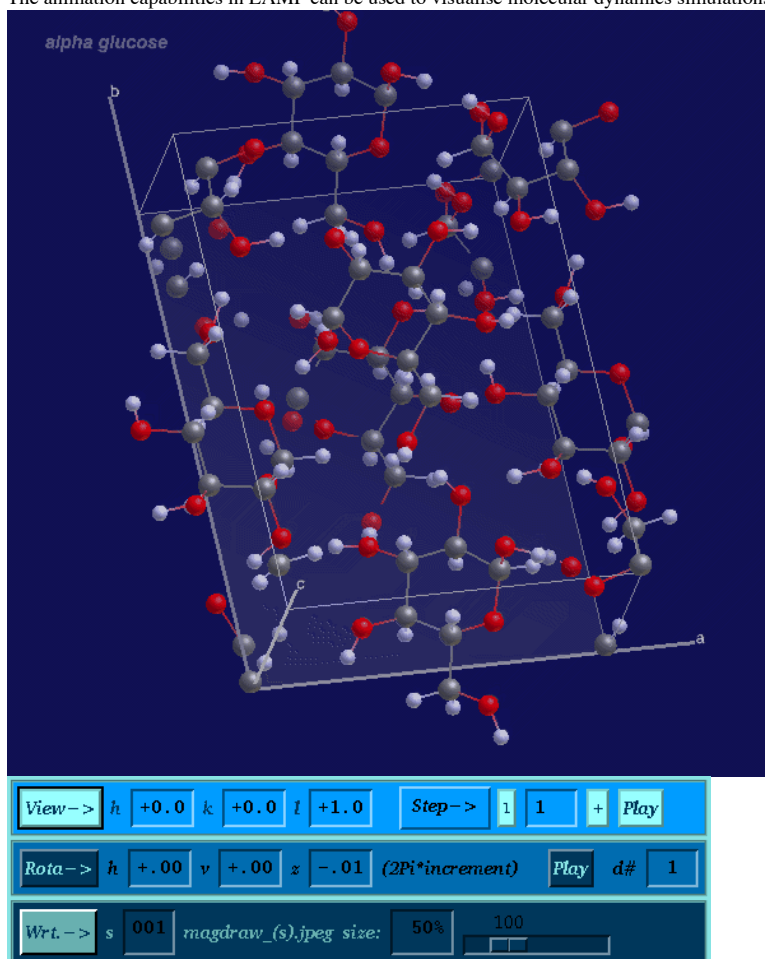
Miller planes can be shown in the crystallographic cell to help visualise local structure. Standard representations of atoms and structural units (balls, sticks, polyhedra etc) are available. High quality images and animations can be saved for publications or presentations. Atomistic representations are also being explored for visualising the results of solid state, first principles calculations on materials. Computational codes like VASP based on density functional theory (DFT), allow electron and spin density to be calculated and these can be displayed with the crystal structure.

3D isosurface and 2D contour plot representations of electron density, calculated by VASP for LiFePO4.



MOLECULES/DISPLACEMENTS

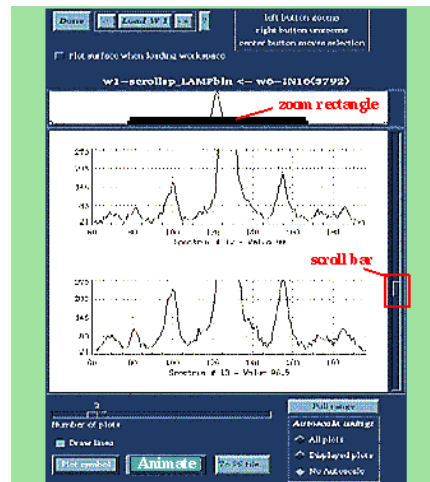
The animation capabilities in LAMP can be used to visualise molecular dynamics simulations and calculated phonons and molecular vibrations.



Scroll Spectra

This provides a window for scrolling through the individual spectra within a workspace. At start-up only triangles are displayed until an input workspace is selected using **Load W#**.

The Scroll Spectra window displaying just two spectra from a workspace



- **Scrolling**

Select the number of spectra to view in one window with the *Number of plots* bar. To scroll through the spectra use the **scroll bar**, scroll one at a time by selecting the scroll bar (click on it) then use the keyboard up and down keys. The *Animate* button allows you to scroll continuously (select either *Slow*, *Normal*, *Fast* or *Stop* to halt the animation).

- **Scaling and Zooming**

By default LAMP will *autoscale* to suit the spectra in the window. If an individual spectrum is zoomed (by dragging with the left mouse-button) all the spectra in the window will zoom. The **zoom rectangle** can be moved (using the middle mouse-button) to view any area of the spectra with the same scale. **N.B.** To zoom in the Y direction *No Autoscale* must be selected. To unzoom use the right mouse-button or the *Full range* button.

- **Plot Style and Printing**

The *Plot symbol* button can be used to plot individual points and the *Draw lines* button to join up the points. The displayed plots can be printed using the *To PS file..* button if the printer is set in *Plotting preferences*, otherwise just a postscript file is made.

- **Tips**

Spectra in different workspaces can be compared by opening further windows with the *Scroll Spectra* button in the main LAMP interface. The [SuperPlot interface](#) is nevertheless more adapted to do this.

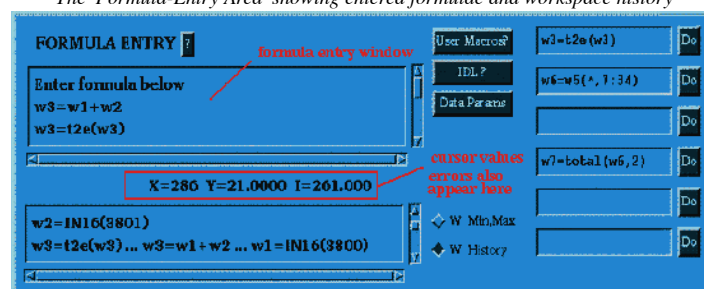
Scroll spectra can be used to visually select spectra that are usefull and you can then [extract spectra](#) from the workspace using the [formula-entry window](#). **N.B.** You must **subtract 1** from the *Spectrum #* to form the true spectrum number because spectra are numbered from 0 not 1 as they are in Scroll Spectra. (*Mask & Group* can also be used to select spectra)


[Previous Page](#)
[Main Contents](#)
[Next Page](#)


5. Manipulating Workspaces

[Imagemap](#) | [Formula-Entry Window](#) | [One-line Macros \(Do buttons\)](#) | [Errors](#) | [Workspace Array and Data Dimensions](#) | [Workspace History](#) | [User-Macro Files](#) | [Internals accessible functions](#) | [IDL Help](#) | [Instrument or Data Parameters](#) | [Session Journal](#) | [Saving the Session](#) | [Exiting LAMP](#)

The 'Formula-Entry Area' showing entered formulae and workspace history



Imagemap - Click on any point to go to the relevant paragraph.

- **Formula-Entry Window**

All formulae and commands entered in this field (by typing them in and hitting return) are scanned by LAMP for [workspace manipulations](#). They are also executed by IDL so that almost all [IDL commands and functions](#) can be entered in this field.

- **One-line Macros (*Do* buttons)**

As well as using the formula-entry window, one-line commands can be also be written in any of the windows on the right of the area in classical Lamp and executed by pressing its neighbouring *Do* button. You can also get them from the menu-bar item "lamp/layout".

Commands can be **copied** from **any field** of the LAMP interface (using left-mouse button and dragging) and **pasted** (using the middle mouse-button) into a one-line macro. These macros are stored when you [exit LAMP](#) and restored next time LAMP is run. They are associated with the working directory (your *path*), so if you change this they will not be restored.

- **Errors**

A **crunch sound** signifies an error and a message will appear beneath the formula-entry window or the DATA COLLECTOR area ([cursor values](#) are also displayed here). *Please report unexplained errors.*

- **Workspace Array and Data Dimensions**

Each time a workspace is altered its dimensions and data minimum and maximum values may change. If the *W Min,Max* button is selected this information will be displayed in the neighbouring window.

- **Workspace History**

Each operation which modifies a workspace is stored in its history. If the *W History* button is selected the histories are displayed in place of the *W Min,Max* values. (see also [Session Journal](#) below)

- **User-Macro Files**

The *User Macros?* button opens a window which allows the [LAMP macros](#) and their descriptions to be visualised and searches for all *.pro files in the directory appearing in the users *path*. It also allows new [macro and command files](#) to be made by the user. You can also open the window from the menu-bar item "Edit".

From the runtime version of Lamp you have only access to the *.prox batch files.

- **INTERNALS ACCESSIBLE FUNCTIONS**

Lot of *internal functions* are accessible and can be called by user. Their descriptions can be visualised from the menu-bar item "Info".

- **IDL Help**

The *IDL?* button activates full IDL on-line help, the ' ? ' button also gives a few tips on using functions and commands.

- **Instrument or Data Parameters**

Each workspace has its own set of instrument parameters (eg. stored in p1(0) to p1(120) for w1, see [Main Concepts](#)), these can be displayed and edited by pressing the *Data Params* button or from the menu-bar item "Info". For example if you [extracted spectra](#) from a workspace you would need to edit the *Number of detectors* by selecting the *Workspace #* and changing the number in the corresponding row.

- **** Session Journal**

All of the workspace operations that you perform (reading data, formulae entered etc.) are logged and can be viewed by pressing the button labelled *The Journal* which is on the right of the main LAMP interface and the menu-bar item "Info". There is also a print option to obtain a hard copy. This journal is saved in local directory as file *lamp.jou*

- **** Saving the Session**

The button labelled *Save Lamp Session* (presents also in the menu-bar item "File") saves the entire current status of LAMP. Next time LAMP is started (from the same directory or *path* as the session was saved in) the session can be restored, ignored or deleted. You may rename the file lamp.ses to prevent it being overwritten.

- **** Exiting LAMP**

The button *EXIT from Lamp* exits cleanly from LAMP and IDL.

N.B. If LAMP shows **no response** ie. appears to have " **crashed** " then close the shell window in which LAMP was started.


[Previous Page](#)
[Main Contents](#)
[Next Page](#)


6. Common Workspace Functions

[Formula-entry syntax \(IDL\)](#) | [Simple Algebra \(+, -, /, * \)](#) | [Extracting sub-arrays](#) | [Integrating \(total\)](#) | [Normalising on Monitors](#) | [Maths-functions](#) | [Showing Values \(show\)](#) | [Shifting axes](#) | [Resizing an array \(congrid\)](#) | [Putting Workspaces together](#) | [Changing Data-type](#) | [Graphics](#)

Formula-entry and Do text area syntax is IDL

All text entered in the formula-entry area is parsed and then sent by LAMP to the Interactive Data Language (IDL). The main reason for parsing is to enable blocks of related arrays in a source workspace (see [Main Concepts](#)) to be passed together to a target workspace.

The passing rule is :-

The workspace, eg. **w2**, on the left of the '=' is the target workspace, whilst the *first* workspace, eg. **w1**, on the right of the '=' is the source workspace.

Extensive on-line help is available on this language by pressing the [IDL? button](#). You can make any IDL command through the [formula-entry area](#), and these will normally be operations on workspaces. LAMP provides maximum flexibility by giving you access to the majority of its [working variables](#), either for one line commands or for [command files](#). You can put data into any variable that you have declared and operate on it with IDL, but these variables will not be managed by LAMP, the variables a,b,c,...z are available for your use. 3.14159... can be obtained by typing in " !PI"

Note: In the following we refer to data in the x-dimension as "channels" and in the y-direction as "spectra". Appologies to crystallography.

- **Simple Algebra with Workspaces (+, -, /, *)**

To add two workspaces w1 and w2 simply type:

```
w3=w1+w2
```

in the [formula-entry window](#). Subtraction, multiplication etc. are similar. In these operations LAMP will transfer the parameters, axes values, titles (and monitors) of the first workspace after the "=" sign into the new workspace (see [passing rule](#) above).

Effectively, **w3=w1+w2** also implies:

p3=p1; n3=n1; x3=x1; y3=y1; titles in w3=titles in w1 ...

- **Extracting sub-arrays (channels/spectra)**

To extract spectra 1 to 40 from w1 into w4

```
w4=w1(*,0:39)
```

In this example the * implies **all** channels. *N.B.* 1 has to be subtracted from the spectrum numbers to suit the IDL array numbering.

To extract channels 250 to 500 for all spectra from w1 into w4

```
w4=w1(249:499,*)
```

To extract channels 250 to 500 for spectra 30 to 40 from w1 into w4

```
w4=w1(249:499,29:39)
```

N.B. LAMP ensures that parameters, axes etc. are transferred correctly but it does not update the values for the number of spectra and number of channels in the new parameters. These can be edited using the [Data Params](#) button.

- **Integrating (total)**

To sum all the spectra of w4 into w5

```
w5=total(w4,2)
```

where **2** represents the second dimension (y) ie. the spectra.

The sum of a range of spectra could be made, say from Spectrum #30 to Spectrum #40 using

```
w2=w1(*,29:39) & w2=total(w2,2)
w2=total(w1(*,29:39),2) is correct but Lamp will not update new axis.
```

To sum across all the channels, eg. for *s(Q)*

```
w3=total(w1,1)
```

If only elastic *s(Q)* is required and the elastic-peak position is known to be from say 200 to 220

```
w3=w1(199:219,*) & w3=total(w3,1)
w3=total(w1(199:219,*),1) is correct but Lamp will not update new axis.
```

- **Normalising on Monitors**

Monitor spectra are stored in the array **n** (ie. n1 for w1, n2 for w2 etc.). If each channel is to be normalised with the corresponding channel in the monitor spectrum (diffraction instruments) then the data in w1 can be normalised using

```
w1=w1/n1
```

If you need to normalise with the integrated monitor spectrum, for spectra of say 512 channels you must sum across the channels using (n1 can be two-dimensions)

```
w1=w1/total(n1(*,0))
```

If the position of the peak in the monitor is known to be say 220 to 240 channels then a better result is obtained with

```
w1=w1/total(n1(219:239,0))
```

If this command is used frequently it should be put into a [one-line macro](#), a [command file or a macro](#).

- **Maths-functions**

These work on workspaces or arrays of any dimensions (including scalars):-

Multiplication: **x*2**

Division: **x/2**
 Powers: **x^2**
 Square root: **sqrt(x)**
 Absolute value: **abs(x)**
 Exponential: **exp(x)**
 Logarithms: **alog(x)** (natural) or **alog10(x)** (base 10)
 Trig. functions eg.: **sin(x)** or **asin(x)** arc-sine

See *IDL help* for more functions.

- **Showing Values** (**show**)

The value of any variable or expression can be displayed in the [formula-entry window](#) with the command

```
show, x          eg.      show, total(w1,1)      show, a*5
```

If the result is an array only the first 10 elements are displayed. The IDL **print** command can be used to display the entire array in the shell window, eg.

```
print, x
```

Minimum and maximum values of workspaces are available in the [formula-entry area](#). Otherwise to get min and max values between say channels 250 to 500 for spectra 30 to 40 of w1

```
show,max(w1(249:499,29:39))
```

The channel and spectrum in which the min or max occurs can be put into a user parameter, say c, as follows

```
a=max(w1(249:499,29:39),min=b,c)
show,a;(display maximum)
show,b;(display minimum)
show,c;(display index where maximum occurs)
```

- **Shifting axes**

The values of the x-axis, in time, energy channels etc., are stored in x1 for w1, and x2 for w2 etc. To shift spectra simply add (or subtract) the appropriate value from the x array. For example, to shift data in w1 (say in energy) by 0.15

```
x1=x1+0.15
```

- **Resizing an image** (**congrid**)

To resize w1 which may, for example, contain a 128 by 512 image to another size, say 64 by 256 pixels in w2

```
w2=congrid(w1,64,256)
```

- **Putting Workspaces together** (**join**)

To join workspaces into first dimension(W4), second dimension(W5), third dimension(W6):

```
w4=[ w1 , w2 , w3 ]
w5=[ [w1] , [w2] , [w3] ]
w6=[ [[w1]] , [[w2]] , [[w3]] ]
```

- **Changing Data-type**

Main data types are:

```
a=0          short integer, limits: -32769 to 32768
b=0.0        floating (real), limits: magnitude 10-38 to 1038-1
c="hello"    string (text), limit: 32767 characters
```

Changing between data-types:

```
d=long(a)    convert 'a' to long-integer (limits: -231 to 231-1) (a=0L is long integer)
e=fix(b)     truncate 'b' to integer
f=float(b)   convert 'a' to floating
f=double(b)  convert 'a' to double float
g=string(b)  convert 'b' to string
h=round(b)   convert 'b' to nearest long integer
```

N.B. By default integers are 2 bytes. If you have numbers greater than 32768 you will get odd results unless you work with long-integers or floats, eg. enter `w1=float(w1)` in the formula-entry window.

- **Graphics**

plot,a,b

Simple line plot of 'a' against 'b'. If only 'a' is given it is plotted as a function of its point number.

oplot,a,b

Simple line plot of 'a' against 'b' over previous plot.

contour,a,b,c

Plots a contour map of the array in 'a'. If 'b' and 'c' are given these are the x and y values for the grid. b and c can be vectors or arrays.

surface,a,b,c

Plots a surface of the array in 'a'. If 'b' and 'c' are given these are the x and y values for the grid. b and c can be vectors or arrays.

shade_surf,a,b,c

As surface but with shading.

N.B. The plot-range buttons in main Lamp window only apply to workspaces. When you issue the above commands you must set `xrange`, `yrange` and

zrange manually, eg. to plot monitor 1 of workspace w3 from 200 to 250 channels:

```
plot,n3(*,1),xrange=[200,250]
```

See *IDL help* for more functions.

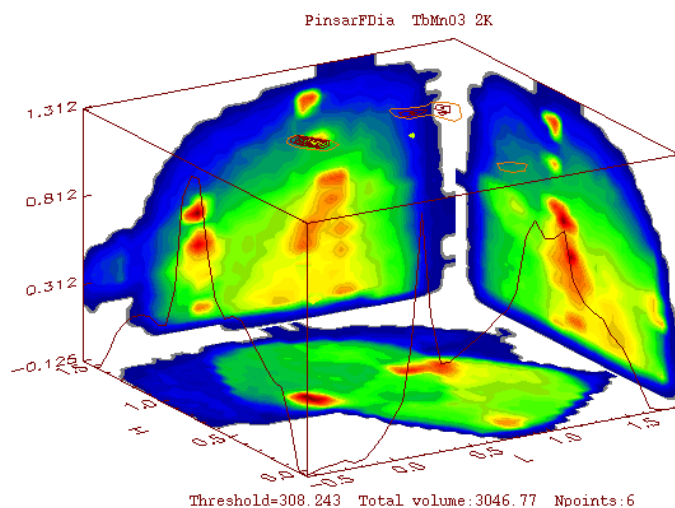

[Previous Page](#)
[Main Contents](#)
[Next Page](#)


7. Other Display interfaces and "TOF legacies"

[Filling reciprocal space](#) | [Magnetic scattering cross-section](#) | [George Layout](#) | [Mask & Group masking](#) | [grouping spectra](#) | [TOF reduction legacies](#)

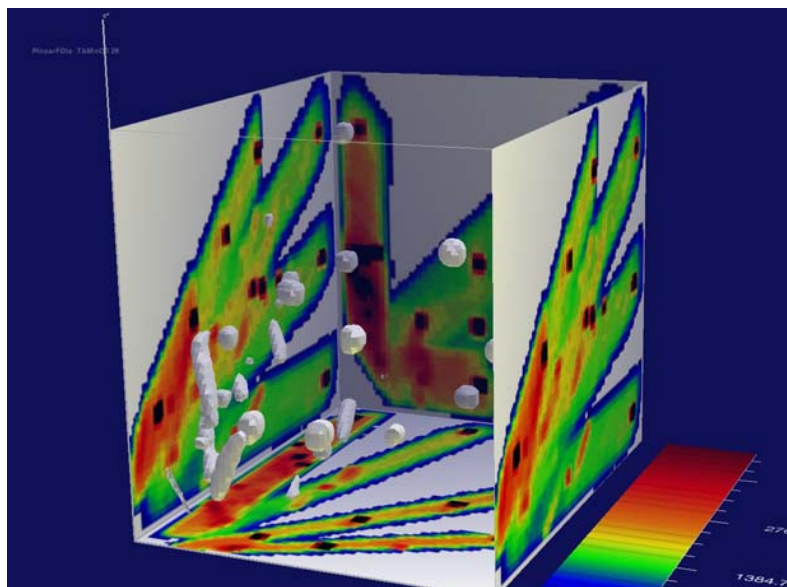
Filling reciprocal space

Representation of a serie of scans during acquisition to survey different regions of the reciprocal space and identify complex features due to incommensurability or diffuse.



Using the function MAKE_VOLUME we obtain a real 3D object which may assist during the measurement to identify the observed intensities appearing outside Bragg positions. **Syntax is:** `w2=make_volume(w1,/hkl)`

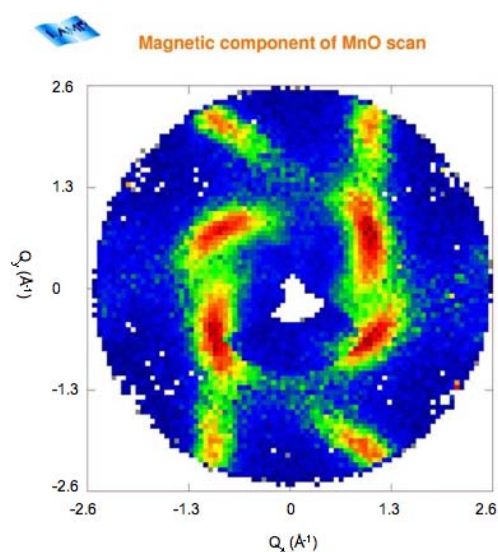
Now the figure below shows nuclear and diffuse, magnetic scattering from many data sets taken on D9 on TbMnO₃. By interactively rotating and zooming the data in (hkl) space, the data can be inspected in detail and indexing the reflections becomes easier. The 2D projections can be moved dynamically through the volume to help locate more precisely diffracted intensity of interest.



Magnetic scattering cross-section example

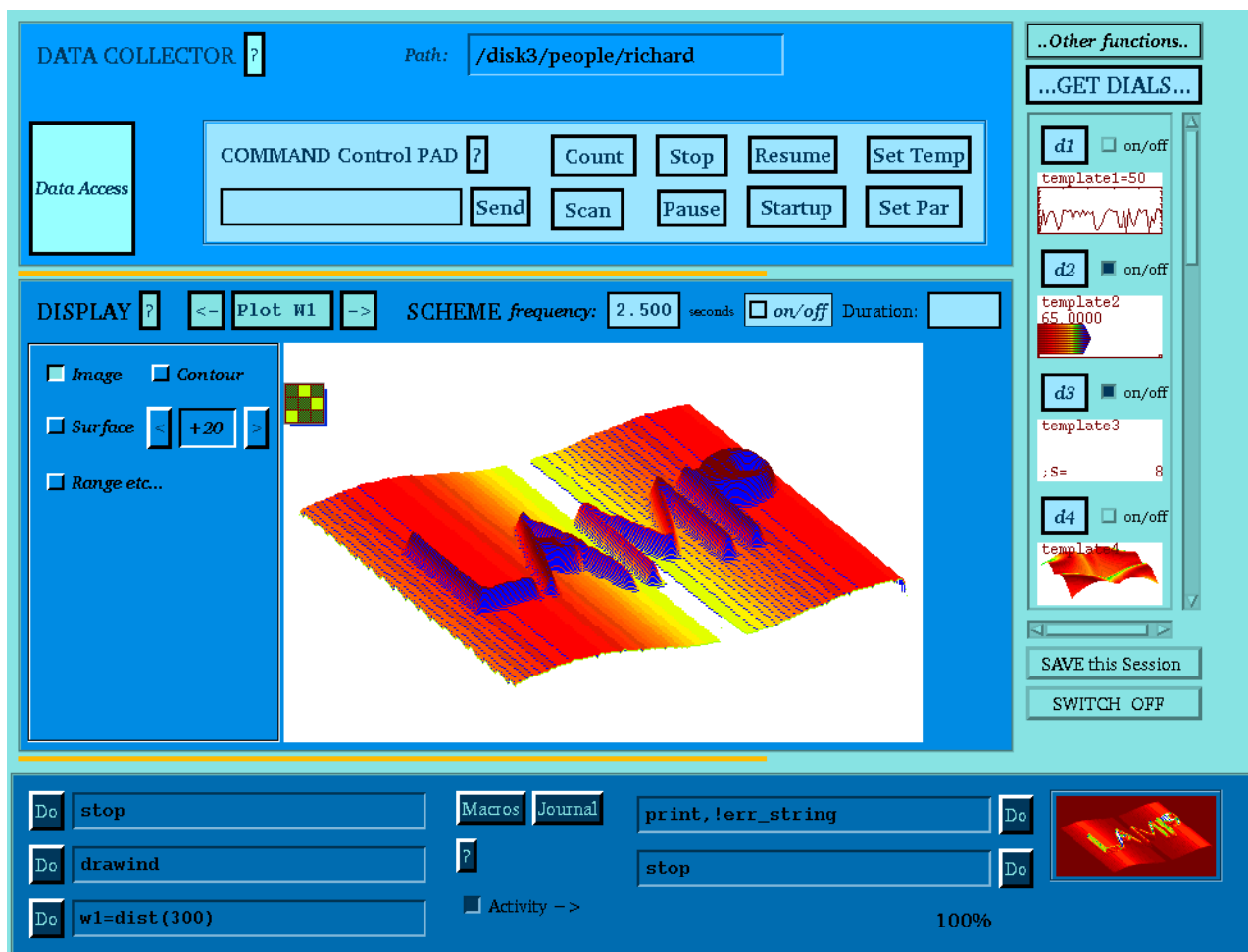
Using the program suite described in the TOF-Manual (linked in the [FRONT page](#)) here is an example of diffuse scattering in manganese oxide (MnO) from D7.

Magnetting scattering cross-section of a 5g MnO single cristal at -122°C , 30°C above the temperature at which the sample orders magnetically. The nuclear contribution to the scattering has been removed using neutron polarisation analysis. Magnetic manganese atoms tend to align themselves at temp greater that the Néel temperature.



George Layout

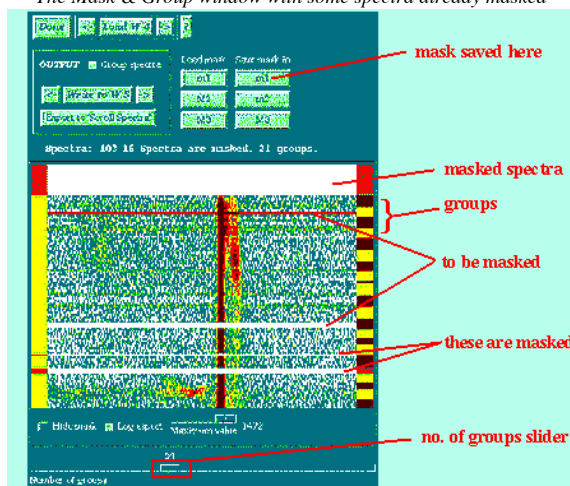
GEORGE (General Experiment Organisation, Response and Guidance Executive) allows **on-line data analysis** to be combined with **any** instrument control and data acquisition sequences. The macros written by the instrument scientist has a **frequency** property which determines how often information is sent to, or received from the instrument control system. Other properties when updated in the macro, allow George to display "live" texts, images, fit results etc... The Control Pad with there pull-down buttons is **dynamically** constructed at launch from a text file edited by the scientist!!!



Mask & Group

This interface allows the noisy or unwanted spectra in a workspace to be *masked* and the remaining spectra to be *regrouped* into a new workspace. At start-up a colour chart is displayed until a workspace is loaded using the **Load W#** button.

The Mask & Group window with some spectra already masked



- **Masking**

By default a log image of the data is shown. Normally this reveals the bad spectra as horizontal lines (shown above). Spectra are selected for masking by clicking on the image with the left mouse-button, and deselected by clicking on a previously selected spectrum. A range of spectra can be selected by clicking on the first with the left mouse-button, and on the last with the right mouse-button. To **save a mask** so that you can apply it to other runs click on one of the **M1..M3** buttons. You can write the *cleaned* data to a new workspace by selecting a workspace and pressing **Write to W#**. **N.B.** Be sure that the **Group spectra** button is not selected if you do not want to group the output workspace. If you want to re-use a saved mask for another run use the **Load mask** buttons, otherwise you can define a new mask.

- **Grouping Spectra**

Normally it is best to filter out the unwanted spectra (masking) before grouping. Move the slider at the bottom of the window to select the number of groups to be generated. The marks on the right of the image will show which spectra will be grouped together. Remember to select the **Group spectra** button before writing to the new workspace.

- **Tips**

This is one interface that it is worth keeping as an icon, otherwise you never know what the mask is. It is also best to treat as many runs as possible in one session with this interface.

You can use **Export to 'Scroll Spectra'** to view the current workspace in Scroll Spectra without having to reload it.

The TOF Manual which will help you to reduce Time-of-Flight data is linked to the [FRONT page](#)

[DidLine](#) | [Convert to Energy \(t2e\)](#) | [Rebinning Irregular Data \(rbin\)](#) | [Convert to s\(Q,ω\)](#) | [Normalise to Vanadium \(vnorm\)](#) | [Normalise Backscattering Spectra \(bsnorm\)](#) | [Line-up Elastic Peaks \(lineup\)](#)

DidLine

A friendly way to inspect rapidly the ToF raw data.

The call is : **Wi=didline(Wj)** Where **Wj** = Workspace containing the raw sample.

This command invokes an intermediate interface to specify essentials inputs.

Before calling didline check the following parameters:

Pj(2)= **Doppler frequency** (Hz)

Pj(11)= **Temperature** (K)

Pj(18)= **Channel width** (microsec.)

Pj(21)= **Wavelength** (angstroms)

Pj(27)= **Distance Det - Sample** (meter)

Convert to Energy (t2e)

LAMP uses the function **t2e** (tee) to convert the x axis from time-of-flight (channels) to energy. Usually it is best to normalise, remove noisy spectra, subtract background etc. before using this transformation.

The function is written in IDL and can be examined by pressing the [User Macros?](#) button and selecting t2e.pro from the list which appears.

- **Example Syntax** (to be entered into [formula-entry](#): **w2=t2e(w1)**)

LAMP makes two simple checks on the input workspace before converting to energy.

1. Does the workspace contain more than 1 point?
2. Does the text **t2e** exist in the history of the workspace?

If the input workspace is suitable, **t2e** will convert the energy scale and make the appropriate corrections to the intensity for changing from time-bins to energy-bins. The title of the x-axis is changed. A "shot" sound indicates success and a "crunch" sound signifies an error. *Please report unexplained errors.*

- **Tips:** If the resulting spectrum does not have the elastic peak centred at zero energy there is probably an error in the elastic-peak position. Either use [lineup](#) to align all positions, or sum all spectra of interest, e.g.: **w2=total(w1(*,20:80),2)** then plot w2 and find the elastic peak position with the cursor (click left mouse-button on peak). Press the [Data Params](#) button and edit the elastic peak position to the correct value. Repeat **t2e**.

The sample energy-loss side of the energy spectrum may go to very high values. If the energy spectrum looks odd, with the elastic-peak at one end, select a more realistic plot-range.

If you have been using plot limits in channels remember to change these values before plotting in energy.

Rebinning Irregular Data (rbin)

Data on an irregular energy-scale can be rebinned to a regular grid by typing **rbin** in the [formula-entry window](#). A window opens in which the energy range and energy increment can be chosen for a given input and output workspace.

The *rbin window* makes and executes the command: **w_out=ebin(w_in, E min, E max, increment)**

If required this command can be entered directly in the formula-entry window as, eg.

```
w2=ebin(w1,-0.7,7.0,0.05)
```

Convert to s(Q,ω)

When time-of-flight versus angle data are transformed to energy versus momentum transfer (Q,ω), the data points fall on a rather irregular grid. A Delaunay triangulation of the planar set of Q, ω points is constructed and used to interpolate the data onto a regular grid which can simplify preliminary analysis.

The function **sqw** performs this task but the syntax is complicated so you are strongly recommended to use the [sqwwin](#) command, but if you wish to make conversion within a [macro or command file](#) you need to use the **sqw** function.

Function sqw

example syntax: **w2=sqw(w1,0.02,0.1,-1.0,1.0,0.1,2.0)**

Order of arguments: (input workspace, energy increment, Q increment, energy min, energy max, Q min, Q max)
 You should use a course grid whenever possible otherwise the processing time will be very long. Choose Q max, Q min within the limits of the data.
 An additional argument **,/fast** can be added if the Q, ω grid of the input workspace is the same as that used for a previous call of **sqw**, the same triangles will then be used, making the calculation faster.

Command sqwwin

Typing **sqwwin** in the formula-entry area will bring up a special interface to set the arguments for the **sqw** function and then perform the calculation.

Adjust the sliders to select the workspaces (*in* and *out*), the energy (*E*) and Q ranges of interest. The default maximum Q corresponds to the maximum energy transfer in the highest-angle spectrum - that is 1 point! Usually you should reduce the maximum Q value. You should avoid generating more than about 30 points in energy and 50 values in Q.

Press the **Do Interpolation** button to start the calculation. If **sqwwin** has been used previously you will get extra options related to the triangles calculated before. If in doubt, select the **Delete Triangles** option. The interface closes when the calculation is complete.

Normalise to Vanadium (vnorm)

The function **vnorm** will take the integral of a given spectral region in one workspace and then divide the corresponding spectra in a second workspace by this integral.

- Example Syntax: **w3=vnorm(w1,w2,230,250)**

Order of arguments: (input workspace, vanadium workspace, lower limit for integral, upper limit for integral)

The limits are those of the elastic peak in the vanadium spectra and must be given in channels not energy.

You can use this to get $s(Q)$ for either an elastic or inelastic peak by [extracting the spectra](#) of interest eg.:

w2=w1(150:200,*) and then normalising: **w4=vnorm(w2,w3,230,250)**

Normalise Backscattering Spectra (bsnorm)

The function **bsnorm** is used to normalise backscattering spectra to the monitor spectrum eg. to normalise workspace, w1, and put the normalised workspace into w2, type

w2=bsnorm(w1)

Line-up Elastic Peaks (lineup)

Because there are frequently small differences between the distance from the sample to individual detector-groups there can be slight differences in the time-of-flight channel in which the elastic peak arises. When keyword NOFIT is set, the function **lineup** first smoothes each spectrum and then estimates the position of the maximum. if NOFIT is not present then a gaussian fit is made for all spectra. An average of these positions is taken and then all spectra are shifted so that their elastic peaks are at the average position. Any peak which is more than 10 channels away from the elastic-peak channel given in the parameters is not shifted. The new average peak position is returned in **elas** but not entered in the parameters automatically.

- Example Syntax: **w2=lineup(w1, elas [,nofit])**

The counting statistics in an individual spectrum need to be adequate to enable the elastic-peak to be found. Otherwise the routine does nothing.

• Tips

You can get a quick estimate of the elastic peak position by summing all spectra (do not include the multidetector on IN5) by entering, for example:

w2=total(w1(*,30:90),2)

The function **shift** can then be used to move channels by a given integer, for example:

w2(*,50)=shift(w1(*,50),3)

will shift spectrum 50 in w1 by 3 channels. Channels shifted off one-end wrap around to the other. (See [IDL help](#) for more info.)


[Previous Page](#)
[Main Contents](#)
[Next Page](#)


8. LAMP Macros

[Command Files](#) | [Command Files with loops](#) | [Compileable Macros \(*.pro files\)](#) | [IDL Basics](#) | [Passing of Common Variables](#) | [An Example Macro: demo.pro](#) | [How LAMP deals with macros](#) | [Statistics errors handling](#) | [Simple IDL programming](#)

Command Files

Command files are simply a list of [functions or commands](#) written into a file as they would be typed in the formula-entry area. The file can then be called up in the [formula-entry and Do input text area](#) and the functions it contains will be executed, line by line.

Files can be written in a window which is opened by pressing [User Macros?](#) button and selecting the **Create a new: Batch** option, the filename should be given a *.prox form, select **Write new file** to save it in your directory.

- An Example Command File: demo.prox

```
w1=w1(*,61:95)/total(n1)
w2=w2(*,61:95)/total(n2)
w3=w3(*,61:95)/total(n3)
```

```
w3=w3-w2
w1=w1-w2
w3=vnorm(w3,w1,200,300)
```

This would be executed by entering @demo in the formula-entry or Do input text area.

Command Files with Loops using the XBU interpreter

You have access to the **XBU interpreter** from item "Lamp/Layout" in the menu-bar of Lamp. You will be able to execute a command file in which you can insert statements like **While** , **for** , **if**. This tool is very usefull for writing simple codes without programming and compiling. You also write some conditionnal codes in the **runtime** version of Lamp (see the Help in the XBU interface).

Compileable Macros (*.pro files)

More elaborate IDL functions and commands (such as **t2e** used to convert time-of-flight channels into energy units) can be written, compiled and saved as macros using LAMP. They can then be executed in the formula-entry area.

A macro procedure can be written by pressing the [User Macros?](#) button and selecting the **Create a new: Macro** option, the filename should be given a *.pro form. After writing the file pressing **Write new file** will compile and save it, any **compile errors** will appear in the **shell window** from which LAMP was started.

• IDL Basics

The main things to be aware of are:-

- IDL works on entire arrays.
- The separator is always a comma (not a space) for procedures and parameters.
- The default for integers is 2 bytes.
- Indices start at 0. For example if w1 is dimensioned at 512 this means w1[0...511].
- Changes to *common block* @lamp.cbk is dangerous.

• INTERNALS accessible functions

Lot of **internal functions** are accessible and can be called by user. Their descriptions can be visualised from "Internal ROUTINES" in item "Info" of the main Menu-bar. (i.e **take_datp** , **give_datp** , **mod_datp** , etc...)

• Passing of Common Variables

All variables relating to a workspace are available. There are two ways to get them :

- by including the parameter : [datp](#) in a user defined macro.
- by invoking the TAKE_DATP procedure with datp as the only parameter, i.e : **take_datp**, [datp](#)
If you use this way and have changed datp, you have to end your procedure or function by **give_datp**, [datp](#)

In both cases, datp variable attached to the workspace invoked in the command line will be internally set by lamp with the following structure :

```
datp= {x:          x,          (vector of x coordinates.)
      y:          y,          (vector of y coordinates.)
      z:          z,          (vector of z coordinates.)
      e:          e,          (the errors associated to DATA, same size as DATA)
      n:          n,          (monitor spectra)
      p:          p,          (vector of parameter values)
      pv:         pv,         (a table of any dimensions containing other parameter values)
      par_txt:    par_txt,    (string array of text associated to DATP.P)
      w_tit:      w_tit,      (main title)
      x_tit:      x_tit,      (x axis title)
      y_tit:      y_tit,      (y axis title)
      z_tit:      z_tit,      (z axis title)
      other_tit:  other_tit,  (subtitle)
      time:       date }      (string date of the experiment.)
```

In fact, this structure allows one to modify all the parameters of a single workspace.

If you want to resize a parameter variable, you have to use the **mod_datp** procedure.

Its calling sequence is **mod_datp**, [datp](#), 'tag_name',value. For example, to replace datp.x, you should use **mod_datp**, [datp](#), 'x' , new_x_value.

In case of a function, i.e : **w1=f(w2,[datp](#))** or **w1=f(w2)**, w2 parameters (and eventual modifications) are reaffected by lamp to w1. W2 parameters are not modified.

◦ Calls accepted using or not [datp](#) parsing (Function or procedure)

```
W1 = f(W2,W3[,datp])    W1 parameters will receive W2 parameters
a = f(W2[,datp])        W2 parameters might be changed
W1 = f(a[,datp])        W1 parameters might be changed
proc,W1[,datp]          W1 parameters might be changed
```

◦ An Example Macro with [datp](#) parsing: demo.pro

```
function demo,w,datp ; (program line, demonstrates use of function)
; ** The call is w1=demo(w2,datp)
wret=w(10:* , *)
mod_datp,datp, 'X', datp.x(10:*)
datp.y=datp.y^2
datp.p(9)=123
return, wret ;return result
end
```

In this example entering **w1=demo(w2,[datp](#))** in the formula entry window would result in workspace w1 equal to w2 cleared of its first 10 channels, its y scale, y1, equal y2 squared (Q^2 for example) and parameter, p1(9), being changed to 123 (for TOF instruments this is the elastic peak channel).

Note that **mod_datp** is used to resize the X parameter.

All other w1 parameters will have the w2 parameters unchanged.

◦ Same example as above without parsing [datp](#):

```
function demo,w
; ** The call is w1=demo(w2)
take_datp,datp
wret=w(10:* , *)
mod_datp,datp, 'X', datp.x(10:*)
```



```

    datp.y=datp.y^2
    datp.p(9)=123
    give_datp,datp
    return,wret
end

```

o **How LAMP deals with macros**

When LAMP parses the formula entry it stores the number of the target workspace on the line in a variable (let's say target) and the second in another variable (let's say source). LAMP transfers all variables associated with the source workspace into [datp](#) variables. after the call, all [datp](#) parameters are copied into target parameters. If there is only one workspace in the command line, source and target are the same, the standalone workspace. If you use a **function** call, as in [example above](#), you must **return** the **w** array. If you use a **procedure** the return has no argument.

o **Statistics errors handling**

To each workspace is associated a variable, named e, which should contain the statistics errors. "w" and "e" must have the same dimensions, and user might fill the "e" variable if needed. The updating of this variable is a user responsibility when you manipulate workspaces using many operators or a function.

• **Getting Common Variables from RDRUN, RDAND, RDOPR used in macros.**

All variables relating to a workspace are available.

When used in macros you can add the keyword DATP= datp to get all data parameters in the structure variable datp
ex: w1=rdand(1234,1237,DATP=datp).

• **Simple IDL programming**

Loops

```

for i=1,n do begin
    a=a+1
endfor
;alternatively
while a lt b do begin
    a=a+1
endwhile

```

If then

```
if a eq b then c=d
```

If then begin

```

if a ne b then begin
    c=d
    e=f
endif

```

If then else

```

if a lt b then begin
    c=d
endif else begin
    e=f
endelse

```

Case

```

case i of
1: j=7
2: j=6
3: j=5
4: j=4
else:
endcase

```

Declaring arrays

To declare an integer array, say 'i', of dimensions 3x4:

```
i=intarr(3,4)
```

To declare an long array, say 'k', of dimensions 5x6x10:

```
k=lonarr(5,6,10)
```

To declare a floating array, say 'r', of dimensions 3x3x2:

```
r=fltarr(3,3,2)
```

To declare a string array, say 's', of dimensions 5x5:

```
s=strarr(5,5)
```

see [Changing Data-type](#) for methods of changing between these.


[Previous Page](#)
[Main Contents](#)
[Next Page](#)

9. Frequently Asked Questions

o **I see no plot!**

You may be plotting black-on-black, see [manual entry for colours](#) on Displaying Data page.

o **I get a wrong number value after a normalization.**

This error occurs when you divide by a too small number (or zero). This can occurs also by dividing non-floating spectra.

o **When I press print I only get a PS file.**

The LAMP default printer must be set in *Begood*, see [manual entry for printing](#) on Displaying Data page.

o **LAMP is stuck (no response)**

Certain operations on large arrays can take a long time, such as contour plots and the sqw command, but if you are sure that LAMP has *crashed* then see [Exiting LAMP](#) on Manipulating Workspaces page.



mail queries about LAMP or The LAMP Book to lamp@ill.fr

[Previous Page](#)

[Main Contents](#)

End of Last Page

Legal Notice

The LAMP package is distributed in the **public domain**. If you find this application useful, **you may send** an electronic mail message to lamp@ill.fr. We would gratefully appreciate any feedback on malfunctions.

The reference in your Publications should be:

() LAMP, the Large Array Manipulation Program. http://www.ill.fr/data_treat/lamp/front.html

Installation Notice

INSTALL RUNTIME "lamp_runtime_windows.zip" on Windows 2000, XP

- Extracting **lamp_runtime_windows.zip** to drive C: will create the Free_Lamp directory.
- Right click on the shortcut file "Free_Lamp\lamp.exe" and choose **Properties** item.
- Change Target to: "C:\Free_Lamp\lamp\IDL\bin\bin.x86\lamp.exe" -em=..\..\lib\hook\runtime.sav
- Change Start in to: "C:\Free_Lamp\lamp\IDL\bin\bin.x86"
- Change Icon... which is located at Free_Lamp\lamp\lamp.ico
- double-click on lamp.exe and enter **YES** to "Do you want to import preferences?"

INSTALL RUNTIME "lamp_runtime_unixs.tar.gz" on Unix , Linux , MacOSX

- entering "**tar -xzf lamp_runtime_unixs.tar.gz**" will create the Free_Lamp directory.
- Edit **START_lamp** to set **lampd** to the full lamp path directory (ex: "/usr/local/Free_Lamp/lamp"), avoid spaces in directory names on MacOSX.
- Launch the script file **START_lamp** each time you want to start Lamp.

Warning: this script assumes that csh is in /bin (MacOSX).

INSTALL SOURCE "lamp.tar.Z" on Unix , Linux , (MacOsX) , (Vms)

- A running IDL must be obviously already installed.
- entering "**zcat lamp.tar.Z | tar tf -**" will create the lamp directory.
- Execute the script file **MAKE_LAMP.unix**
- Edit **START_LAMP.unix** to set **lampd** to the full lamp path directory.
- Execute the script file **START_LAMP.unix** each time you want to start Lamp.

INSTALL SOURCE "lamp.zip" on MS-Windows , MacOSX , (MacOs9)

- Start IDL (IDLde) and select File -> Preferences -> Path
 - Add the "LampPath"
 - Add the "LampPath\lamp_mac" (+ enable subfolders)
- Enter **@MAK_LAMP.win** from the IDL command line to produce the lamp.sav file.
- To start Lamp: start IDL then enter **lamp** or **@lamp.ini** (You can also edit File->Preferences->Startup)

SOURCES

New versions of LAMP can be downloaded via anonymous ftp at :
<ftp.ill.fr> in subdirectory /pub/cs/ files lamp.tar.Z or lamp.zip
or in subdirectory /pub/cs/lamp_runtimes/ for embedded packages.

You can retrieve the LAMP source by [Downloading it](#), now !!!

You can retrieve the LAMP runtime by [Downloading it](#), now !!!

MAIL BOX

Suggestions and problems may be sent to lamp@ill.fr.