EUROPEAN SYNCHROTRON RADIATION FACILITY

INSTALLATION EUROPEENE DE RAYONNEMENT SYNCHROTRON

FIT2D V10.3 Reference Manual V4.0 ESRF98HA01T

Copyright A P Hammersley/ESRF 1987-1998
Email: hammersley@esrf.fr

August 26, 1998

Use of **FIT2D** implies acceptance of the "Conditions of Use" (see Page 278).

Copyright 1987-1997 Andy Hammersley / ESRF
Use of FIT2D implies acceptance of
the " Conditions of Use "
(Click on "CONDITIONS")

V10.3

Abstract

This manual describes use of **FIT2D**, a 2-D data analysis program, specialising in model fitting. **FIT2D** is a program which allows difficult data analysis problems to be tackled using 2-D dimensional fitting of user specified models. To enable model fitting to be performed on a wide variety of input data, many other more basic data analysis operations are also available. Calibration and correction of detector distortion has become an important part of **FIT2D**. **FIT2D** is still under development, but already a wide range of basic data analysis operations and a number of more sophisticated methods are available. Normal graphics display methods are available.

INTENDED AUDIENCE: This document is intended for scientists and engineers who intend to use **FIT2D** for basic data analysis, data visualisation, or data format conversion. This manual is intended for reference and most users should not need to read more than small sections of the manual. The index should be of use in finding information on specific subjects, and users are recommended to use the index extensively. The manual contains many cross-references so useful inter-relating sections should be easy to find. (If you have helpful suggestions for further index entries or cross-references please contact me.) For newcomers to **FIT2D** the user guide **"FIT2D: An Introduction and Overview"** is recommended as a starting point.

MAJOR VERSION CHANGES: The following major changes have occurred which are signalled through the change in the major version number.

At **Version 10:** WindowsNT and Windows 95 support is available. This is still a testing stage, but gradually **FIT2D** will become available on Windows based systems as well as Unix systems.

The "symbol" system was replaced by the "variable" system. The "variable" system allows internal program variables to be defined from the command-line or within **FIT2D** and used within macros to allow attempts like file names to be automatically varied. The "variable" system is very similar to the previous "symbol" system, but variables have the concept of type i.e. a variable may be an integer value, a logical (boolean) state value, a floating point real value, or a character string value. The commands `DEFINE SYMBOL, SYMBOL, LIST SYMBOLS` and `UN-DEFINE SYMBOL` have been replaced by the commands `DEFINE VARIABLE, VARIABLE, LIST VARIABLES` and `UN-DEFINE VARIABLE`. The `DEFINE VARIABLE / VARIABLE` command now prompts for the type of variable to be defined. (The `SYMBOL` command is now an alias for the `DEFINE VARIABLE` command.)

At **Version 9:** all NAG code was removed from **FIT2D**, so that no NAG licence (or other commercial licence) is necessary to run **FIT2D**. This mainly affected the fitting code and associated options.

At **Version 8:** the "Graphics User Interface" (GUI) became the default interface to **FIT2D**. (The command line ("Keyboard") interface can still be used directly as the default interface, by entering the option `-key` or `-keyboard` on the command line when starting **FIT2D**.)

NEW FEATURES: The following major changes and new features have been added since Version 9.129 (as well as many minor ones):

- **Graphics Display**
  - "Feature" which made "two-click" mode input unusable corrected (V9.130).[1]
  - Support for X-11 displays with 24-bit "TrueColor' colour mapping (V9.136).
  - Allow PostScript output of images to rebin by differing factors in each direction (V9.137).

[1] In fact the numerical overlay was being written on top of the spy-glass image. This was in a way intended, so I term a "feature", but nevertheless made the function unusable so many may well regard this as a bug. Still now it's fixed.

- Support for “eXceed” and “LAN Workplace Pro” PC X-servers in 24-bit plane ’TrueColor’ set-up (V9.138).
- Re-draw graphics when an ’Expose’ event occurs with 24-bit ’TrueColor’ and reverse RGB coding. This attempts to work around a bug in the X-server with “LAN Workplace Pro” in 24-bit mode. The X-server wipes out all graphics whenever an expose event occurs ! (V9.139)
- **CURVE STYLES** control option added within **OPTIONS** sub-menu (V9.148).
- **DISTANCE** command added to **DISPLAY** sub-menu to calculate the distance between two coordinates (V9.156).
- Output all graphically input coordinates to the terminal window (V9.156).
- Masked out coordinates no longer accidently drawn on 1-D graphs. (This was the cause of red crosses being incorrectly drawn.) (V9.156).
- Re-set linear axis scaling after log X/Y graph (previously this caused a problem with subsequent 2-D axis drawing), and put masked points in correct place on Log scales (V9.157).
- Graphical value input now has a graphical **O.K.** button to accept the current value (V9.157).
- Work around problem with Silicon Graphics IRIX 5.3 Indigo workstation screens reporting that they have 2-bit plane depth default “visuals” (V9.160).
- Correction to graphical coordinate input so that **TWO CLICK** mode input is disabled for X/Y graph coordinate input (previously hung program) (V9.166).
-
- Change Windows image display to use much more efficient ’CreateDIBSection’ routine (V9.171).

- **Fitting and Graphical MASKING Menu**
  - New scaling method used in fitting resulting in much faster fitting and much better convergence (V9.142).

- **POWDER DIFFRACTION Menu**
  - Change array index order for 2-theta and azimuthal directions in ”CAKE” and ”INTEGRATE” commands. The 2-theta output scan is now in the X-direction, which is the natural order for fitting with the **MFIT** interface (V9.130).
  - Handle **CANCEL** button requests properly in the **INPUT** and **CORRECTION** commands forms (V9.133).
  - Correct problem with 2-D Gaussian beam centre sometimes being returned out by a large factor (V9.143).
  - Always output D-spacings when the user clicks on the image (even if distance, wavelength have not been explicitly set) (V9.152).
  - Tilt angles saved between calls to **FIT2D** (V9.156).
  - Correct **CAKE** integration so that the region is correctly integrated when the data has an offset axis start (V9.174).

- **IMAGE PROCESSING GUI Menu**
  - **EXTEND** command added to the **GEOMETRIC** sub-menu, to allow the size of the defined data region to be easily increased (V9.144).
- **ON-LINE CRYSTALLOGRAPHY Menu**
  - Output D-spacings for peak search, and use centroiding to improve peak centres (V9.161).
- **SAXS/GISAXS GUI Menu**
  - Option of 1-D projected scan output for grazing incidence small angle scattering (V9.149).
  - Add **SUMMATION** option and option to correct intensities for flat plate geometry in **PROJECTION** command (V9.150).
  - Take account of detector tilt angle in **PROJECTION** command (V9.151).
  - Tilt angles saved between calls to **FIT2D** (V9.156).
  - **1-D TRANSFORMS** command added to allow the I(q) versus q scans to be transformed to [Log] I(q)**a * q**b versus [Log] q**c scans (V9.157).
  - Draw projection region in correct place for offset images (V9.157).
  - Option of **FIT 1-D PROJECTION** in **BEAM CENTRE** command to allow the symmetry of 1-D profile to be used to define a beam centre (V9.158).
- **FILE SERIES GUI Menu**
  - **INTEGRATE** command added, allowing automatic integration of a whole series of files (V9.130).
  - **AVERAGE, COMPOSITE,** and **INTEGRATE** commands within the "FILE SERIES" interface now generate file names in the same manner as the **MACROS/LOG FILE** interface **RUN SEQUENCE** command (V9.131).
  - Option to output 1-D integrated scan files in the **INTEGRATE** command (V9.133).
  - Correction to **INTEGRATE** command. Previously, the intensity tended to fall high angles owing to the spatial correction only being performed on the zoomed in region (V9.137).
  - First correct input image of **INTEGRATE** command prior to display for masking and ROI selection (V9.138).
  - Will now work for 'TIFF' and 'IMAGEQUANT' formats without asking for the input re-binning form (V9.163).
- **KEYBOARD Interface menu**
  - Option to input "PDS" format (V9.145).
  - Add more predictors to `PREDICTOR` command (V9.153).
  - Update `FUJI LINEARISATION` command so that full user control is given to parameters of linearisation formula (V9.164).

  - Change name of commands `DEFINE SYMBOL`, `LIST SYMBOLS`, etc. to `DEFINE VARIABLES`, `LIST VARIABLES`, etc. (V10.1).

- **File Input and Output**
  - Cure problem with "PRINCETON INSTRUMENTS" format input (and potentially others) when the image is larger than the current data array (V9.132).
  - Support input of 4-byte integers Princeton Instruments format (V9.134).
  - Support input of 4-byte reals with Princeton Instruments format (V9.135).
  - Improve support for Fuji BAS**** scanners. Now a variety of possible files will be tried for the binary data, and if they are not found the user will be prompted for the file to use. The linearisation is now performed using a LUT so should be much faster (V9.140).
  - Correct file selection call for the "FIT2D" format option for an output file (previously was for an input file) (V9.154).
  - Set title based on input file name for "2-D GAS" format input using the GUI (V9.157).
  - Correct GUI selection of output files for "CBF", "CHIPLOT", "TIFF", and "HUFFMAN" formats and for output of detector distortion correction table file (previously the file selection tool was looking for an input file) (V9.160).
  - Byte swap input of ESRF format Real*4 data if necessary (V9.163).
  - CHIPLOT input now gives error message if there is a problem opening a file (V9.163).
  - Convert "CHIPLOT" input to new I/O system calls so that a file with only 'read' permission can be input on Linux machines (the Absoft Fortran compiler requires read/write permission unless non-standard keywords are used) (V9.167).
  - Convert "FUJI" and "BINARY" format and spatial distortion function file input to new I/O system calls so that a file with only 'read' permission can be input on Linux machines (V9.168).
  - Convert "BSL" format file input to new I/O system calls so that a file with only 'read' permission can be input on Linux machines (V9.169).
  - Correct error introduced in V9.166 which meant that "TIFF" file 'XResolution' and 'YResolution' tags were not correctly defined (V9.170).
  - Add support for the "Klora" file format in the GUI input routine (V9.171).
  - "TIFF" output inverted, so now is the correct way up for the TIFF standard (V9.172).

- **Detector Distortion Calibration and Correction**
  - Bi-Spline and image plate spatial distortion correction made about 33% faster (V9.144).

- **Macros**
  - The **RUN SEQUENCE** command in the GUI, take account of any unchanging part of the file name prior to any extension (V9.131).

- **Other:**

– Help texts now gives references to web pages (V9.167).
– Store symbol data type within internal symbol database (now the variables database) (V9.175).
– Allow evaluation of arithmetic expressions within command menus (V9.176).
– Replacement of all 'SYMBOL' routines with 'VARIABLE' routines, so now all internal variables have a type and are stored in integer, logical, floating point, or character string form (V10.0).
– Command-line option '-gdoc' added to allow control of the option of saving menus and forms to PostScript files. (This is to help produce documentation.) (V10.3).

FURTHER DEVELOPMENT: FIT2D is presently under development and many new features are likely to be added in the future. If users have particular requirements which fit within the general frame-work that **FIT2D** provides, I will be happy to try to include new features to fulfil their needs.

When all else fails ... Read the documentation (anonymous)

Documentation is like sex; even when it's bad, it's better than nothing (anonymous, Internet)

These two quotations are more or less consistent with the authors' view on documentation. I do not like to have to read documentation, but it is necessary and at times indispensable. Most of this reference manual should not be read ! It is a reference manual and as such, apart from some general considerations which are given in the first few sections, the remainder of the manual can be considered for reference only.

FIT2D *should* be very largely self-documenting. All graphical menus and forms contain a **?** button and/or a **HELP** button. Clicking the **?** button will produce a list of the available commands and a short one-line explanation of the command. The **HELP** button will produce interactively scrolled help text, describing the whole menu or form, and the options and choices that are available. Within the keyboard interface you can obtain an explanation of the required input for every prompt by entering a question mark (?). The "HELP" option is also available within many menus.

Nevertheless the Reference Manual exists, and I feel it can be useful. If you want to perform an operation but do not know the name of the command there is a large index to help you. If you want to know precise details on the workings of a command, the reference manual contains extra information and cites articles and books containing more information.

Contents

1 Prerequisites

To be able to run **FIT2D** the following tasks must have been fulfilled:

Installation: **FIT2D** since Version 7.0 is a single executable which does not depend on any other files, although a PostScript version of this reference manual may also be useful, as might be the introductory manual: "FIT2D: An Introduction and Overview" (both these manuals are now available as web pages).[2]. **FIT2D** executables are available for most workstation systems and can be obtained from the world-wide web. The **FIT2D** home page URL is:

```
http://www.esrf.fr/computing/expg/subgroups/data_analysis/FIT2D
```

From the home page links to installation documentation, executables, and user and reference manuals are available. With a minimum of computer knowledge you should be able to obtain and set-up **FIT2D** yourself. Otherwise ask your system administrator for help.

On Unix systems it is generally necessary to set the "executable bit". Assuming the **FIT2D** executable has been copied to a file named `fit2d`, this can be achieved with the following Unix command:

```
chmod +x fit2d
```

Environment Variables: On Unix systems The following environment variable needs to be defined:

`DISPLAY` As with all X-11 programs the environment variable `DISPLAY` needs to be set to your X-11 display. Normally this will be defined automatically, but sometimes it may be necessary to define or re-define this variable manually. e.g. Using the "C" shell or (better) the "T" shell with a workstation called `experiment1` the following command could be typed on the "T" shell command line:

```
setenv DISPLAY experiment1:0
```

`FIT2D_KEY` variable needs to be set to an individually determined code: "the software key". This is only necessary outside of the ESRF, and not on ESRF computer systems. For external users an e-mail message will be with the value of the individual code (key value) and instructions on how to set it.

The current "values" of the environment variables may be checked by using the "C"/"TC" shell command `printenv`.

[2]If the Version of **FIT2D** being used has been linked "dynamically" then it is dependent on the required system libraries being available. If it has been linked "statically" then it should be completely independent.

2 Running FIT2D

To run **FIT2D** on Unix systems simply enter `fit2d` < Return> on the command line[3]. On Windows systems you may either run **FIT2D** from a command window as on Unix systems, or click on a **FIT2D** icon. If necessary **FIT2D** will open a "terminal window" for text output, and keyboard input.

A "welcome" banner should appear in the terminal window with the version number of the program. If this does not happen check with your system administrator that **FIT2D** has been installed and that your `PATH` is properly defined (see Section 25.1, Page 238).

After the header information a graphics window should be started as a separate X-11 window. If **FIT2D** cannot open the graphics window for some reason it will output an error message in the terminal window with suggested causes of the problem and possible solutions. If no error message appears, but the graphics window does not appear, check the definition of the `DISPLAY` variable e.g. `printenv DISPLAY` < Return>. Other potential start-up problems are covered in Section 25.2, Page 238.

Assuming everything has been set-up properly, the graphics window should have been opened and **FIT2D** will be waiting for graphical input in one of four buttons: **CONDITIONS, I ACCEPT, HELP** or **DON'T ACCEPT**.

Whilst **FIT2D** is running keep the terminal window visible, even when running in the graphics mode. Useful information, warning, and error messages often appear in the terminal window.

2.1 FIT2D Executable Directory

Ideally systems would be set-up such that users would type `fit2d` and **FIT2D** would automatically start. However this depends on the system directories and the users "path" being defined appropriately.

On all ESRF computers **FIT2D** *should* be found in the `/usr/local/bin` directory, which is usually already in a standard "PATH". This is the recommended directory for external users as well, but for various reasons **FIT2D** may be in a different directory. Ask your local system administrator for help if necessary.

(The recommended directory for the **FIT2D** executable on Windows system is not presently defined.)

2.2 Start-up of FIT2D

When **FIT2D** is started you should see a screen of text similar to this one[4] on the terminal screen from where you typed `fit2d`:

[3]Command line options may also be useful, see Section 20, Page 226.

[4]The version number may well be different, and some of the messages notifying you of recent changes may be different.

```
                      ------------------------------
                      PROGRAM  FIT2D  Version: V10.3
                      ------------------------------

        Copyright 1987-1998 Andy Hammersley / ESRF (hammersley@esrf.fr)

FIT2D: 2-D Detector Calibration/Correction; File re-formatting; 2-D Fitting

                      YOU CAN ALWAYS ENTER:   ?
              FOR FURTHER EXPLANATION OF REQUIRED INPUT

                    No commercial software used !
See ``CHANGES'' (keyboard menu) for important changes


!!! NOTE: ``SYMBOLS'' commands replaced with ``VARIABLES''
!!! commands in "KEYBOARD" menu
```

Note the version number that you are using[5].

At the same time the graphics window should be filled with the **FIT2D** graphics "banner" and the **FIT2D** logo and four buttons (unless **FIT2D** has been started in the "keyboard" interface mode).

2.2.1 Graphics User Interface (GUI) Start-Up

The default interface to **FIT2D** is the graphics user interface (GUI) which uses the graphics window with "point and click" user interaction.

The areas with yellow background and blue text are graphically active i.e. they can be clicked upon to select a command, or in later forms a file name.

In this case (see front cover, Page 1) there are four buttons:

- **CONDITIONS** This explains the conditions of use.
- **I ACCEPT** To accept the conditions of use, and enter the main interface selection menu.
- **DON'T ACCEPT** To exit **FIT2D** immediately.
- **HELP** For general explanation on the graphics user interface.

Now and at any time the graphics window is active i.e. the cursor and cross-hair can be moved with the mouse (or other graphics pointing device), the graphics window can be resized by

[5]This is useful if you encounter bugs or problems.

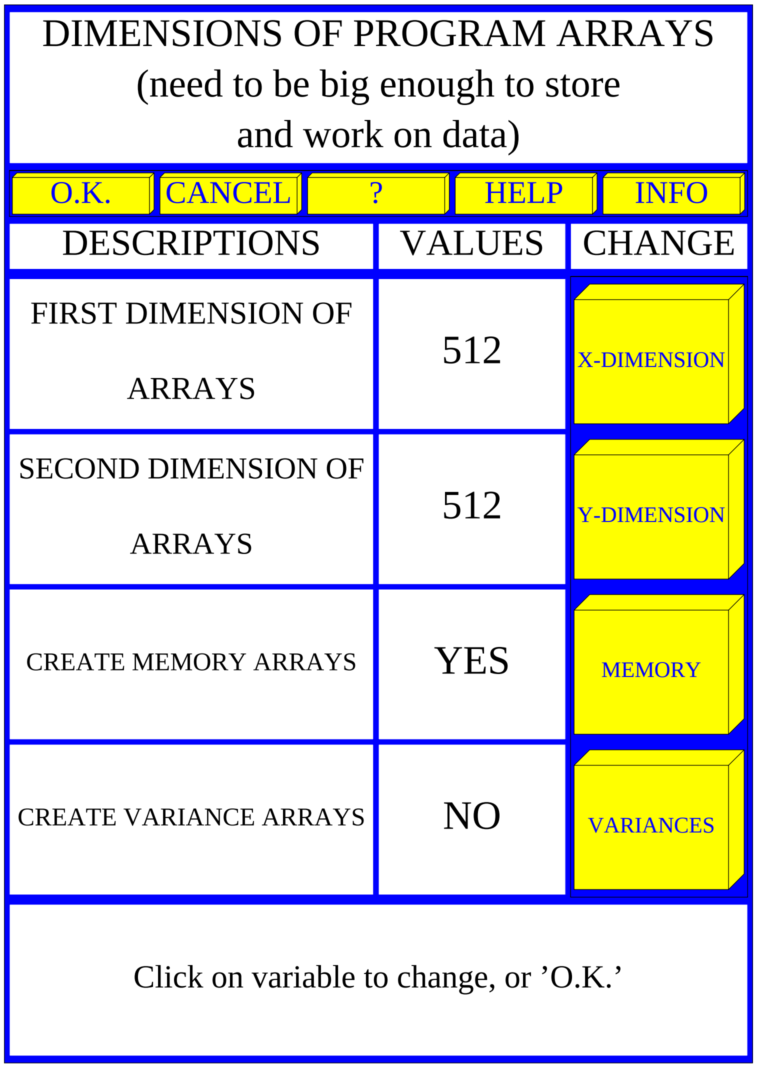
dragging on one of the corners of the window. After user re-sizing **FIT2D** will re-size the window to the largest window with the same aspect ratio as previously within the user defined region, although a minimum size is specified below which the window cannot be reduced.

To enter **FIT2D** you must click on the **I ACCEPT** button which conveys acceptable of the "Conditions of Use" which are detailed within the text form obtained by clicking on the **CONDITIONS** button.

If you don't accept the "Conditions of Use" then click on the **DON'T ACCEPT** button and **FIT2D** will exit immediately. (This is also useful when you remember that you should be doing something else, or want to work from a different directory, etc.)

After the **I ACCEPT** button has been clicked the following form should appear (unless command-line options have been used to specify the equivalent information; see Section 20, Page 226):

Figure 1: **FIT2D** Program Array Dimensions Form



DIMENSIONS OF PROGRAM ARRAYS (need to be big enough to store and work on data)

DESCRIPTIONS	VALUES	CHANGE
FIRST DIMENSION OF ARRAYS	512	X-DIMENSION
SECOND DIMENSION OF ARRAYS	512	Y-DIMENSION
CREATE MEMORY ARRAYS	YES	MEMORY
CREATE VARIANCE ARRAYS	NO	VARIANCES

Click on variable to change, or 'O.K.'

The four vertical buttons allow the size (number of elements) of the internal program arrays and the number and type of arrays to be specified. Unlike many other programs **FIT2D** is completely versatile in the size of images (and 1-D data-sets) which can be treated (although system constraints will impose limits at some level).

Thus when **FIT2D** is started it is necessary to specify the size of the internal arrays used to store data. Normally you will want the arrays to be as large as the data to be input. The precise size will vary from application to application. The arrays can be larger or smaller than the data input from file. If the arrays are smaller then not all of the data can be input at full resolution. If the arrays are bigger, then only the region input will be used, but sometimes it can be useful to specify array sizes larger than the size of the input data.

The **X-DIMENSION** button allows the number of pixels in the first array dimension (horizontal as displayed) to be specified.

The **Y-DIMENSION** button allows the number of pixels in the second array dimension (vertical as displayed) to be specified.

The **MEMORY** button allows the creation of "memory" arrays or not. All operations except for very basic display require the "memory" to exist, so by default this boolean variable has the value **"YES"**. Normally this should not be changed. Only if there is a problem of insufficient computer memory, and only image display is required would this be set to **"NO"**.

The **VARIANCES** button allows variance arrays to be created or not. Variance arrays are necessary if error propagation is to be carried out [3], but this nearly doubles total computer memory requirements and will slow down many operations. This should only be selected if error propagation is being used in the data analysis.

The other buttons (blue text on a yellow background) are general forms buttons. See Section 4.3, Page 35 for general information on the interactive graphical forms.

When the form values are correct, click on the **O.K.** button and the program array will be created and the main interfaces menu will appear:

Assuming the "Scientific Interfaces" menu has appeared you have entered **FIT2D** and are ready to select one of the graphical interfaces or the **KEYBOARD INTERFACE**. These are described below.

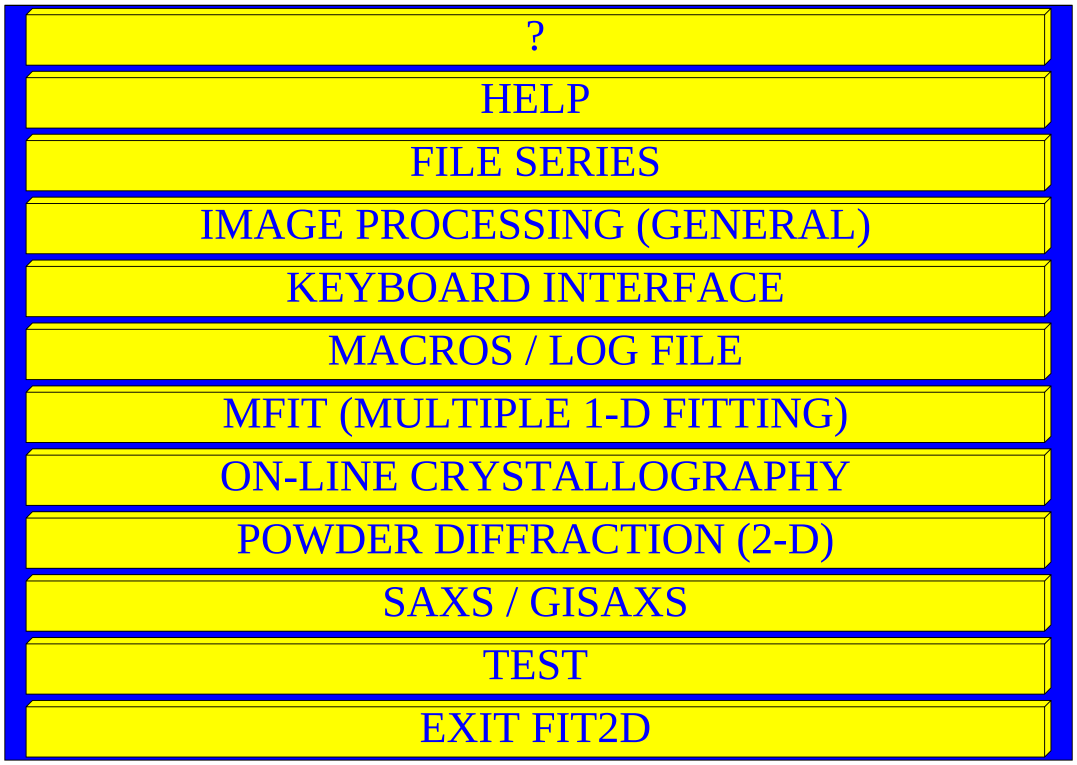
2.2.2 "Keyboard" Interface Start-Up

When **FIT2D** is started-up in the "keyboard" interface mode e.g. by using the `-key` option on the command-line (see Section 20, Page 226 for further information), or when the graphics system cannot start-up for some reason, the line:

```
X DIMENSION FOR ARRAYS (Range: 1 to 1000000) [512]:
```

appears beneath the **FIT2D** banner text within the terminal window.

Figure 2: **FIT2D** Scientific Interfaces Main Menu



This is a prompt line for user input. If you click inside the terminal window[6] and enter a question mark (?), you should see the following text:

```
X DIMENSION FOR ARRAYS (Range: 1 to 1000000) [512]:?
Here you are asked to define the size of the program arrays.  These will
be used to store data "inside" FIT2D.  Normally you will want the arrays
to be at least as large as the image data to be input. If the dimensions
are larger this does little harm,  but is wasteful of system  resources.
If the arrays are smaller then not all of  an image can be input at full
resolution.  Some input options allow an image to  be re-binned on input
or for a sub-region of the image to be input.
```

This prompt and the next prompt:

```
Y DIMENSION FOR ARRAYS (Range: 1 to 1000000) [512]:
```

define the sizes of the internal arrays which are used to store data. As images can vary in size, **FIT2D** allows great flexibility in the size of data that can be input. Very large images may be treated, **but** the computer system on which **FIT2D** is running may not be set-up to allow

[6]This is nothing to do with **FIT2D**, but is a property of the X-window system. Depending on how your X-terminal is set-up it may not be necessary to click, simply placing the graphics cursor inside the terminal window may be sufficient.

enough memory to be used. Users are recommended to use as little memory as they need. The program arrays do not need to be exactly the size of input data: if they are bigger the extra space is not used initially, but may be used by certain commands; if the arrays are smaller than images to be input, then only part of the image will be input, or with some input options there is the choice to "re-bin" the data on input.

The next prompt allows you to create variance arrays or not:

```
CREATE VARIANCE ARRAYS [NO]:
```

If you don't know whether or not to create variance arrays it is best not to create them (the default is not to create them). Creating variance arrays doubles the requirements for virtual memory, and is only useful for a few specialist commands.

Assuming no error occurs the next prompt is:

```
Main menu: ENTER COMMAND [INPUT DATA]:
```

You have now started **FIT2D** and entered the "Main menu". See Section 14, Page 104 for help on general usage of the **FIT2D** "keyboard" interface .

If instead you get an message like the following:

```
STATUS: The error was identified in module:
EXPG_IO: Input/Output and Status

STATUS: Position where error condition was identified
Subroutine EXPG_IO_MALLOC V0.1

STATUS: The error condition has been classified as:
Bad memory allocation: Memory allocation failed
RESET ‘‘status’’ [YES]:
```

then the memory allocation has failed i.e. the system cannot give **FIT2D** as much virtual memory as you have requested. This means that the program arrays have not been created and normal operations will not work. It may be possible to re-define smaller memory requirements using the `DIMENSIONS` command (See Section 15.28, Page 122), or to ask other users to stop some processes to free more virtual memory, or to ask the system administrator to increase the system quotas.

2.3 FIT2D Old and New Versions

On most systems where **FIT2D** is installed and updated a previous version is saved in the same directory as the current **FIT2D** program. The old version is called `fit2d_old`. If ever you find that a bug has appeared in a new version which was not present in the old version, you can use the command `fit2d_old` instead of `fit2d` to obtain an older version. This will take all the command line options exactly as for `fit2d`. (It cannot be guaranteed that sites outside the ESRF are set-up like this, but this is the recommended set-up.)

At the ESRF new features may be added to a development version of **FIT2D** during an experimental run owing to user requests. Since it is preferable not to change the standard version during a run, normally any changes will be made to a version called `fit2d_new`. Only if a major bug is discovered (one which is likely to affect many users) will the standard version (`fit2d`) be changed during a run.

Between experimental runs, the standard version may replace the old version, and a new version may replace the standard version.

(An external site may also choose to set-up a `fit2d_new` version.)

3 Important Principles And Common Features

A number of general principles persist throughout **FIT2D**. These should be largely intuitive and obvious, but some points may be missed by a new user, so are explained here.

3.1 Dynamic Memory Allocation

FIT2D uses dynamic memory (or virtual memory) allocation to allow as much flexibility as possible in the size of data that can be treated. This has obvious advantages, but also has disadvantages. 2-D dimensional data (images) from modern detector systems can be very large, so the demands of **FIT2D** for memory allocation can be similarly large. Users are recommended to try to use the minimum of memory necessary for their data. At start-up the user is prompted for the initial size of program arrays i.e. the size necessary to input their data.

The user is also prompted as to whether or not error estimate arrays (variance arrays) are to be created. If created, the dynamic memory demands are roughly double and many operations will take longer as error propagation is being carried out. If error estimate arrays are not created, no error propagation is carried out, and any fitting will thus be unweighted.

If at any stage during the running of FIT2D the user gets a message that memory allocated has failed, it must be assumed that the previous command has failed. Recovery may be possible, but this depends on the circumstances.

(It is thought that Windows 95 limits the maximum memory size to 64 Mbytes. If this is the case it will be a severe limitation for those wanting to work with large image sizes i.e. greater than 1024 $\times$ 1024. WindowsNT does not impose this limitation so may be a solution for those wanting to use **FIT2D** from a Windows system. Whether or not Windows 98 solves this problem is not presently known, and **FIT2D** has yet to be tested on such a system.)

3.2 The “MEMORY”

FIT2D uses two main internal arrays to store image data and to perform almost all operations. The default array will be referred to as the “current data array” ; this is where data is stored when input, and where data to be displayed graphically must be situated. The second array will be referred to as the “memory”. The “memory” is like the memory facility on many calculators except that the single memory may take up 50 Mbytes or more, and may store a full image.

To move data from the current data array to the memory and vice versa, most graphical menus contain the **EXCHANGE** button, and the keyboard interface contains the `EXCHANGE` command. The output of all operations, which produce output data, within the GUI is in the current data array. In many cases the original data is transferred to the memory, and thus may be recovered using the **EXCHANGE** button. In other cases the original data may be recovered using the inverse operation e.g. **SUBTRACT** after **ADD**.

3.3 "Active Data Region" (*ADR*) / "Region of Interest" (ROI)

All commands which operate on data elements may be limited as to the range of data elements upon which they act. The region or "window" over which operations take place is referred to as the "Active Data Region" (*ADR*) or the "Region Of Interest" (ROI). . In other programs and contexts this concept is sometimes referred to as a "Data Window".

When data is input the *ADR* is initially set to include all the data. The menu commands **ZOOM IN**, **UN-ZOOM**, and **FULL** buttons allow the *ADR* to be changed (see Section 4.2.3). The **MOVEMENT** graphics sub-menu also contains many commands which change the *ADR* e.g. all the commands which move around the image; **UP, DOWN, LEFT,** etc.

It should be noted that the *ADR* concept does not just affect graphical display, but also affects arithmetic and other operations, and the output of data to file (**OUTPUT**). **FIT2D** follows the WYSIWYG ("what you see is what you get") principle in data operations, so only the area of the data which is currently displayed will be operated upon. (The exception is the **EXCHANGE** command which swaps the entire data-sets including the definitions of the *ADR*.)

3.4 The Graphics Window

The graphics window may be resized, to suit user needs, by dragging on one of the corners of the window border. When the graphics window is resized, it will change size, but will keep the same "aspect ratio" i.e. the relative lengths will be kept the same[7]. To do this the correctly proportioned window is fitted within the region defined by the dragged cursor. Thus, to define a larger window, it is important to drag the cursor in both directions. The window can also be made smaller, but there is a minimum size.

By default the graphics window is created in the "portrait" format, in the same ratio as A4 paper. However other formats may be used and in particular "landscape" A4 format is available (see Section 20).

3.5 False Colour Image Display

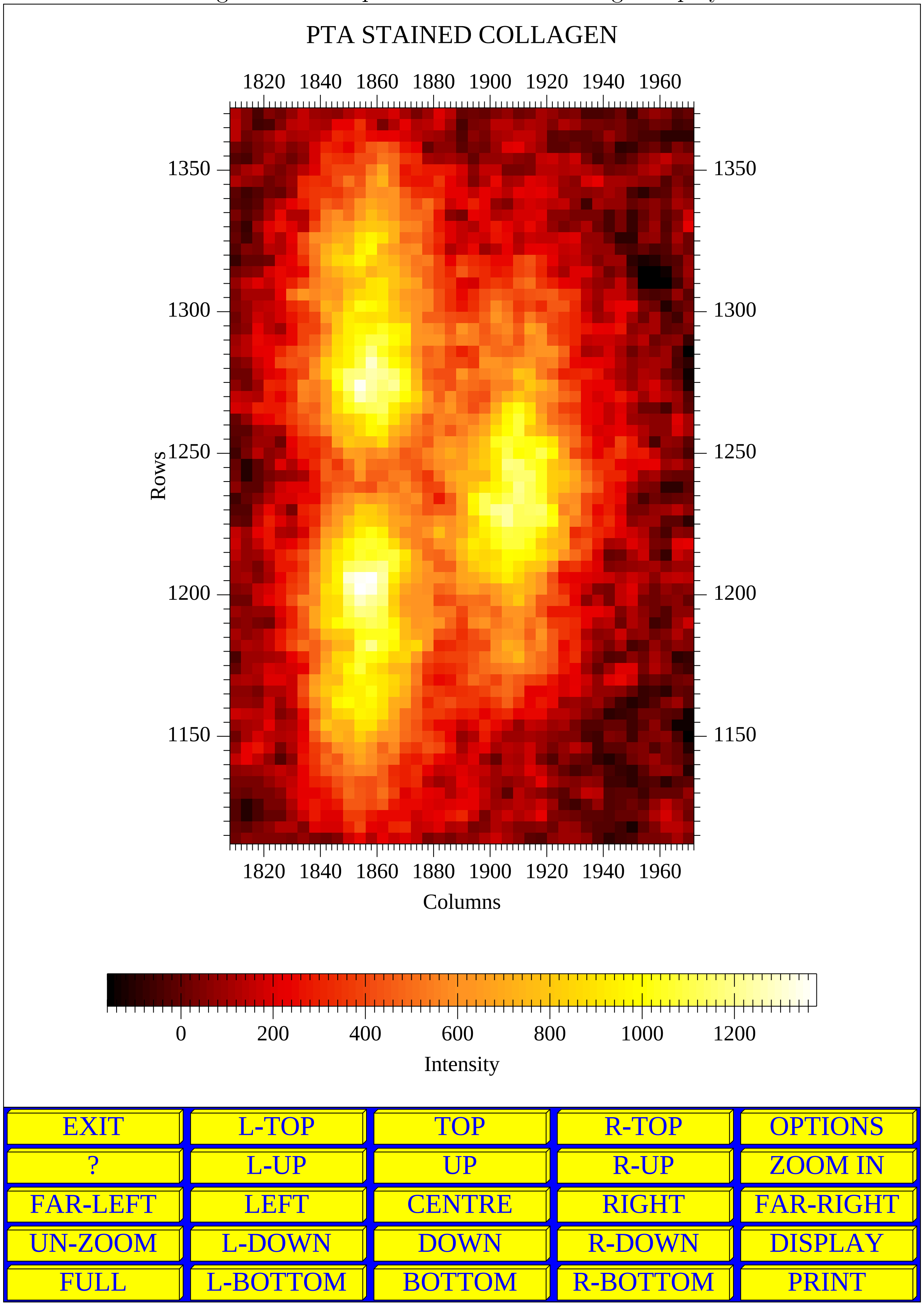
The basis of all interactive data analysis in **FIT2D** is the false colour image display for 2-D data, or an X/Y Graph plot for 1-D data. The flat image display allows easy and unambiguous interactivity. An example of the false colour image display and a control menu is shown in Figure 3[8]. The whole of the ROI is displayed as an image even if there are less screen pixels than data elements, which is often the case.

For general data analysis the style options are generally not important, with the exception of the intensity or "z-scaling". It is necessary to understand the manner in which the data element

[7]This automatic re-sizing does not presently work on Windows systems.

[8]The images displayed in this manual deliberately have relatively small numbers of pixels. This is to prevent the PostScript file of the manual getting too big, and to make printing reasonably quick. Naturally, most analysis will involve images with many more pixels. **FIT2D** can handle very large images and display them efficiently.

Figure 3: Example of False Colour Image Display



values are changed into displayed pixel colours, to interpret the display images.

Generally usage is very obvious, but it is necessary to understand the different scaling modes, which may be present. The mode of intensity scaling can be changed by using the **Z SCALING** command and subsequent sub-menu (see Section 5.3, Page 43). This sub-menu may also be obtained simply by clicking on the intensity scale colour bar, in the main menus of the different interfaces.

By default **FIT2D** uses automatic full range linear scaling of the *ADR* data values. This means that the minimum value and maximum of the *ADR* are found and all values in-between are converted to a colour (black, white and grey's are colours) in the current colour table, using linear interpolation.

With automatic range intensity scaling the whole of the range of data values will always be visible, although some images with a very few very extreme intensity values may not appear very interesting. Full range automatic scaling may be selected using the **FULLY AUTOMATIC** button. As the ROI changes then so do the current minimum and maximum values, and same intensity values may be displayed using different colours. This can be confusing and undesirable in some circumstances.

“Fixed” range intensity scaling is also available. With completely fixed scaling the minimum and and maximum data values which correspond to the extremes of the colour table are fixed to some values. Thus, as the ROI changes the intensities displayed with particular colours stay fixed. However, if data values outside the defined range are encountered then they are just displayed using the colour corresponding to the minimum or maximum of the range.

The full range of scaling options are (together with linear or logarithmic scaling):

FULLY AUTOMATIC: Automatic full range re-scaling. Every time the ROI or the data is changed the minimum and maximum are found and the scaling is set to cover the whole range. This is often suitable, but when a few very high or low valued pixels exist, the detail may be lost in the image.

WEAK PEAKS: Automatic limited range re-scaling. Every time the ROI of the data is changed, a number of regions are used to calculate averages, and standard deviations. The range is set to hopefully display best weak features in the data.

USER MIN/MAX This uses user entered fixed minimum and maximum values for the displayed intensity range.

USER MINIMUM This uses an user entered fixed minimum value for the displayed intensity range, but the maximum is automatically defined by the ROI data values.

USER MAXIMUM This uses an user entered fixed maximum value for the displayed intensity range, but the minimum is automatically defined by the ROI data values.

If data is input but nothing appears to be displayed, it is likely that the intensity scaling is fixed, and that the data values are not within the range. The Z-scaling mode and range values are remembered between calls to **FIT2D** (V9.113).

For further information on controlling the intensity scaling see Section 5.3, Page 43.

Aspects of the false colour diagram style can be changed using the **OPTIONS** command, which is present in many menus (see Section 5.5, Page 49), and other forms of graphical display are available through the **DISPLAY** command which is also generally available (see Section 5.4, Page 45).

3.6 Information, Warnings, and Error Messages

Different types of prompt text, information text, and warning and error messages will appear in the graphics window and also extra information text will appear in the terminal window. The main "dialogue" is in always in the graphics window, except within the **KEYBOARD INTERFACE**, but often more detailed explanation is given in the terminal window.

Information and warning text in the terminal window is identified by one of four initial "key-words":

`INFO:` The text is of a purely informative nature e.g. values calculated as the result of some operation, or information on an input data-set.

`NOTE:` The text is also informative, but highlights a potential for misunderstanding by an inexperienced user. Thus, such messages should be carefully noted.

`WARNING:` Something has not worked as it should. This may be of greater or lesser importance depending on the circumstances. e.g. An input file was not found, or was of the wrong format.

`ERROR:` Something "serious" has gone wrong. This may require exiting **FIT2D** or may be "recoverable" depending on the circumstances. e.g. The system has failed to allocate dynamic memory as requested.

It should be noted that these categories are not strict i.e. in some circumstances a `WARNING` message may be more serious than an `ERROR` message, nevertheless, this show a generally increasing importance and need for user comprehension and maybe action.

4 Graphics User Interface: Introduction And General Use

The "Graphical User Interface" (GUI) is designed to make easily accessible the most used functionality which is available within **FIT2D**. To achieve this, functionality has been arranged into different "interfaces" for different scientific or technical areas. Figure 2, Page 21 shows the menu for the present choice of "scientific interfaces". (Further interfaces are likely to be added in future versions.) It should be emphasized that data can be moved from one interface to another without restriction e.g. 2-D Powder diffraction data could be input in the **IMAGE PROCESSING (GENERAL)** menu, and manipulated, then Debye-Scherrer rings could be integrated within the **POWDER DIFFRACTION (2-D)** menu.

By clicking on the different "buttons" the user enters the different "interfaces". The buttons are:

? The question mark in this menu, and other menus gives a brief description of each of the available choices.

HELP The "HELP" button enters the "context" related help associated with the menu. In this case there is information describing the different available interfaces. Clicking on "HELP" enters a "pager" which allows the user to scroll down and up the help information.

FILE SERIES Is an interface for performing operations on a whole series of files, or images as single commands. This is a newly available interface and much functionality is yet to be added.

IMAGE PROCESSING (GENERAL) Is an interface for general purpose image processing and display operations on data input from a file.

KEYBOARD INTERFACE This enters the "keyboard" command-line interface to **FIT2D**. Certain functionality is presently only available from this interface.

MACROS / LOG FILE Commands to create and run macros, and to open and close log records of the data analysis session.

MFIT (MULTIPLE 1-D FITTING) This interface allows 1-D "peak" functions such as Gaussian, Lorentzian, and Voigt functions to be fitted to data along with other functions such as a polynomial "background" function, and exponential decay or trigonometric functions. The fit model may be fitted repeatedly to a whole series of 1-D datasets. (This provides the same functionality as was previously available through the program **MFIT** [7].)

ON-LINE CRYSTALLOGRAPHY This interface allows on-line display and examination of crystallographic data, with the option of automatic updating of new image data as they are obtained. Simple statistics to evaluate data quality are produced.

POWDER DIFFRACTION (2-D) This interface specialises in the integration of Debye-Scherrer rings from 2-D detectors, to 1-D "2-theta" scans, and to other scans. This is useful for standard powder refinement, but is also useful for texture analysis and other types of integrations and "scans".

SAXS/GISAXS This interface is similar to the **POWDER DIFFRACTION (2-D)** interface, but is under development for specialised support of Small Angle X-ray Scattering (SAXS) and Grazing Incidence SAXS (GISAXS). The interface should also be useful for other types of small angle scattering.

TEST This interface is for test purposes, and is used to generate test data. (Perhaps it will evolve into a data simulation interface.)

EXIT FIT2D Allows **FIT2D** to be exited, after confirmation.

Most of the interfaces contain common sub-menus for generally important and useful functionality such as inputting, displaying, and outputting data, as well as specialist commands particular to the interface. First the general commands and sub-menus will be described.

4.1 Graphics User Interface Colour Coding

Throughout the GUI a simple colour coding is used to indicate: user input, information, or warning messages. "Active" regions such as buttons and region for inputting text have a yellow background with blue writing. (Note: the image of the data and the whole graphics window may be "active" for certain operations, but these are left in their normal colours.) General information text has a white background and black text, and warning or error messages have a red border and within black writing on a white background.

When warning or error messages occur there will usually be extra information in the terminal window, and other useful information can appear, so keep the terminal window visible at all times.

4.2 Parameter Value Input

FIT2D has been developed in a very modular fashion and the user interface exhibits this modularity. This means efficient development and easy use. The same "components" are re-used many times and the user will soon become familiar with the input style.

At the most basic level of user input is parameter input. This may be a number e.g. a value for the number of pixels for the internal arrays in one direction, a character string e.g. a title for a data-set, or a choice between a limited number of values.

The following different types of input exist:

Integer Value

Real Value

Boolean Value

Character String

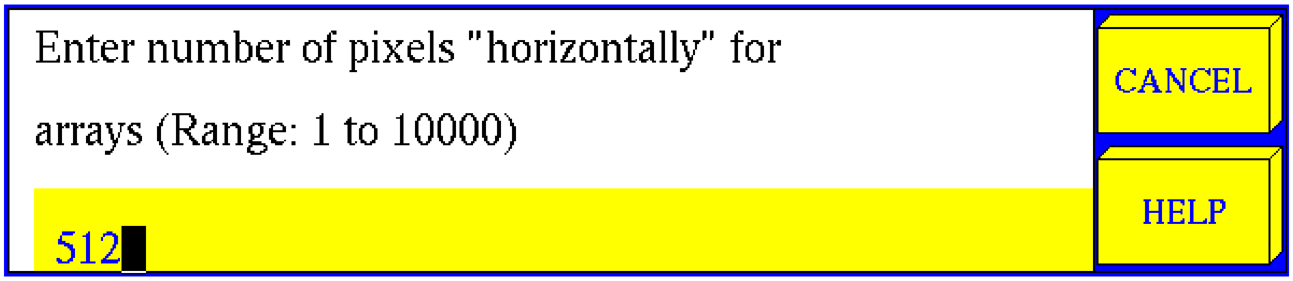
Graphical Menu Choice

Graphical Coordinate

4.2.1 Integer, Real Value, and Character String Input

Figure 4 shows the typical display box for the input of an integer value. A very similar style is used for the input of real and character string values.

Figure 4: Example of the User Input of an Integer Value


Enter number of pixels "horizontally" for arrays (Range: 1 to 10000)

512

The following "components" of the dialogue box are apparent:

Descriptive Text: At the top (with the white background) is a short text describing the required input. This includes the text `(Range: 1 to 10000)` which tells that the value must by between 1 and 10000. For parameters where any value may be input this text is not present.

CANCEL The cancel button cancels the parameter input without changing its value. This may also cancel the operation for which the parameter value was being requested.

HELP The help button may be clicked to obtain more detailed information concerning the parameter value being input and its context in relation to the operation being undertaken.

O.K. The O.K. button may be clicked to accept the current value which is in the text input box. Initially this is a default value from **FIT2D**. This is equivalent to typing the <ENTER> key.

Text Input Box: The yellow lower rectangle is the area for the display of input text. The box already contains the text `512`, which is the default value for the parameter, and the cursor to the right of the number. Text entered from the keyboard will appear in this box at the cursor position.

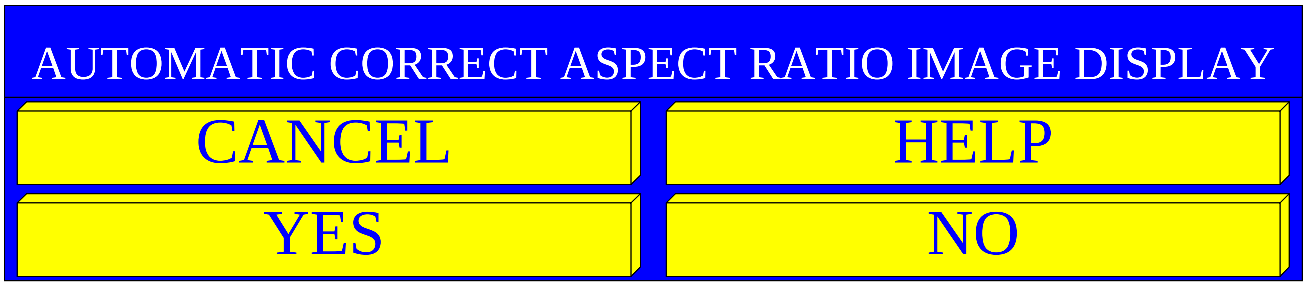
The text may be "edited" using the left and right arrow keys to move the cursor, the "backspace" or "delete" key to remove text and other numerical or alphabetic keys to enter characters. Additionally the up arrow and down arrow keys allow previous inputs to be scrolled. The up-arrow key allows previous input to be "re-called". Each time the up-arrow key is pressed the previous input replaces any entered text. The down-arrow key can be used after the up-arrow key to scan through the previous inputs. If used by itself, or when the current input is being displayed, it removes all characters. This can be useful for quickly removing all the characters of a presented default response.

When the text is as required, the <ENTER> (or < RETURN>) key is used to enter the value. If the value is correct the parameter input is completed, otherwise an error message will presented, followed by the full information "help" message, and the user will be required to re-enter the parameter value.

4.2.2 Boolean Value Input

Many times a simple "yes/no" choice is offered to the user. This sort of "boolean" parameter value input is controlled with a simple menu of four buttons. An example is shown in Figure 5.

Figure 5: Example of the User Input of a Boolean Value



The buttons are:

? Help information on the parameter value being requested

CANCEL Cancel input without changing the value of the parameter. This will also in many cases cancel the operation for which the parameter was requested.

YES Set the value to "Yes".

NO Set the value to "No".

The "CANCEL" button will cancel the operation and the parameter will keep its previous value.

The "?" button will display help text describing the purpose of the parameter and the type of value required.

4.2.3 Graphical Menu Choices

Graphical Menus are the normal manner in which data analysis operations may be chosen by the user. The choice of "scientific interfaces" is a graphical menu, and each of the interfaces is a menu. Various menu commands may themselves produce sub-menus e.g. the **Z-SCALING** command produces a sub-menu.

When a button is pressed, it will be re-drawn pressed-in, to show that it has indeed been pressed. (Note: afterwards it may be re-drawn normally, and if this happens quickly enough the pressed-in button may not be noticed.)

Different interfaces have different commands, but the following commands are common to several or all menus:

EXIT Every menu has an exit button which is used to exit the menu and return to the previous menu, or in the case of the "scientific interfaces" menu to exit **FIT2D**, provided confirmation is given.

? The question mark button provides a message with the available button commands and a short explanation of the commands.

HELP The help button provides a "context" related help message on the menu and the overall functionality which it provides.

INPUT Input data from a file (see Section 4.4 for more details).

OUTPUT The output button is used to save the current *ADR* of the data to an output file.

EXCHANGE Exchange the current data with the memory data. If the memory was previously undefined, then the current data array becomes un-defined (see Section 3.2).

ZOOM IN Define a smaller *ADR* using graphical cursor input (see Section 3.3).

FULL Set the *ADR* to be the whole of the currently defined data (see Section 3.3).

UN-ZOOM Increase the size of the *ADR*. This is done symmetrically if possible (see Section 3.3).

PRINT Save the currently displayed graphical image to a PostScript file. The name for the PostScript file is requested.

DISPLAY Sub-menu with variety of alternative data display possibilities to the false colour image e.g. contour or 3-D surface views (see Section 5.4).

OPTIONS Sub-menu offering control of graphical display style options (see Section 5.5).

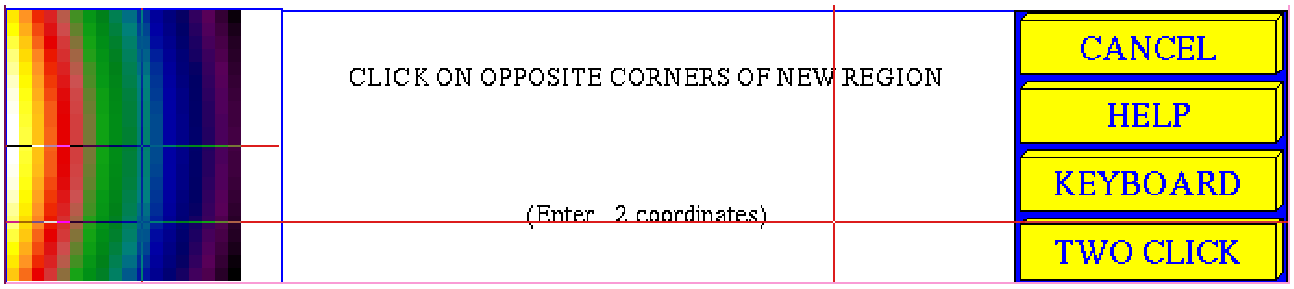
4.2.4 Graphical Coordinate Input

FIT2D has a true **graphics user interface** in that the user interface includes the graphics display, as opposed to only buttons and dialogue boxes, of most programs with "GUI's". For many operations the most convenient form of user interaction is for the user to indicate some feature in the data. This is performed by clicking on a false colour image of the data (or an X/Y graph for 1-D data).

An example of the dialogue box associated with graphical coordinate input is shown in Figure 6.

The various components of this dialogue region are:

Figure 6: Example of Graphical Coordinate Input



"Spy-Glass": On the left is the "spy-glass" or zoom window which shows the region around the cursor at full pixel resolution. As the cursor is moved within the data image, so the image within the spy-glass changes. The cross-hairs in the centre of the spy-glass show the position of the cursor. Unlike the main image display, the spy-glass is always full range automatic intensity scaling.

User Prompt: The white background rectangle contains "prompt" information describing the purpose of the required coordinate input. Note: For some coordinate input, this region can be replaced by a yellow button, to be pressed to end input e.g. when the user can enter a variable number of coordinates.

Cross-Hair: Drawn on top of all the graphics is the cursor cross-hair. These vertical and horizontal lines follow the cursor, and may be useful when the input coordinate needs to be aligned with features in the data.

HELP The help button will provide a message describing in greater detail than the "prompt" the required input and its purpose.

CANCEL The cancel button allows the input to be cancelled. This will usually also cancel the operation for which the coordinate was required.

KEYBOARD The keyboard button may be used to enter the X and Y coordinate position, by entering two numbers from the keyboard instead of the graphical click. This can be useful when some precise value is required.

TWO CLICK Enter "two click mode". In this mode for graphical coordinate input the first click within the image stops the spy-glass, and then a second within the spy-glass defines the input coordinate. This allows more accurate input, particularly for very large images which will be displayed at much lower than single pixel resolution. When in "two click mode" this button will change to **ONE CLICK**, and may be used to return to normal "one click mode".

When a coordinate is selected, a double cross is drawn on the image at the appropriate position. The double cross is made from a black horizontal/vertical cross, and a white diagonal cross. Thus, regardless of the colour of the data the cross will be visible.

Often when two or more coordinates are input, the later coordinates are related in some way to the first, or previous coordinate. In such a case, a "rubber-band" line or rectangle is drawn and re-drawn as the cursor moves. e.g. The **ZOOM IN** command requires two coordinates to define the new *ADR*; and a rectangular box will be drawn defined by the first coordinate and the cursor position, whilst the second coordinate is being input.

4.3 Graphical forms

Often a number of control parameters for a data analysis operation may be varied by the user. Since for many of the parameters the default values will be suitable "graphical forms" are presented. The same basic "form" is used for many different purposes. The form for defining the size of program arrays when **FIT2D** is first started-up is an example. All forms have the same basic components and functionality, although the number of user variable parameters may change. The form presented to the user for modification when the **INTEGRATE** command of the **POWDER DIFFRACTION (2-D)** interface is shown as an example in Figure 7.

Each user controllable parameter has a button which may be clicked to change the value of the parameter. This button will contain some short text to describe the parameter. To the far left will be a short phrase giving more detail as to the purpose of the parameter, and immediately to the left is the current value.

Clicking on the buttons of parameters which require a "YES/NO" value will simply toggle the value, whilst clicking on other types of parameters will produce a "dialogue" region at the bottom of the graphics window. Here the parameter value can be entered from the keyboard (see Section 4.2.1), or a choice or inputs will be displayed as a graphical menu.

In addition to the parameter buttons are the following general buttons:

O.K. Clicking on the O.K. button will exit the form, saving any changes to parameter values which may have been made.

CANCEL Clicking on the cancel button will exit the form without making any changes. This may also cancel the operation for which the parameters were being input and return to a menu.

INFO This gives general information explaining the use of the graphical form.

? This gives a short phrase explaining the use of each of the buttons in the form

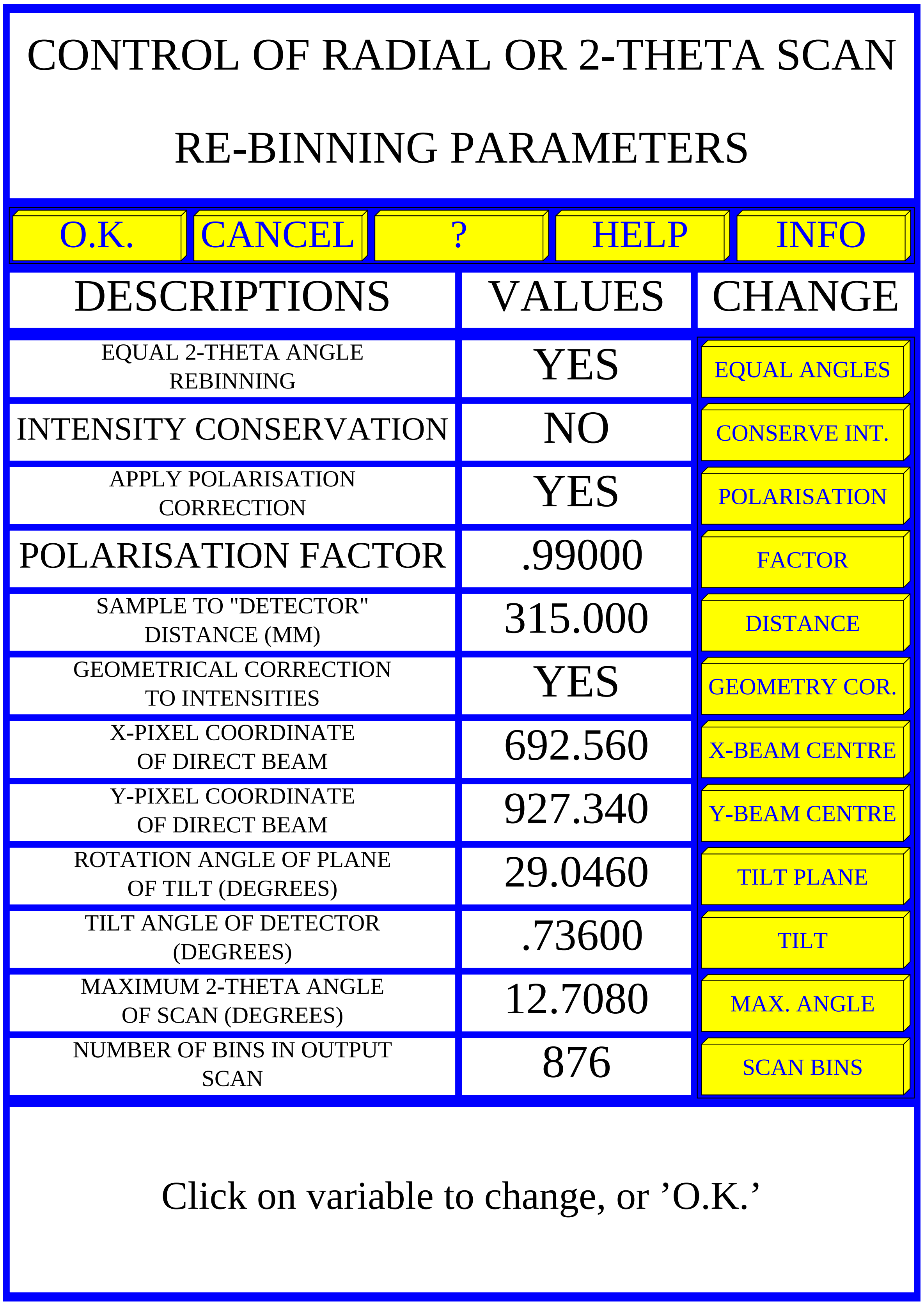
HELP This gives help information specific to the particular graphical form and the parameters being controlled. This gives an overview of how the parameters control or vary the operation.

4.4 File Selection and input

Another often repeated operation is the selection of a file for input of data (or for other purposes). To facilitate this operation a "file selection tool" is used. This allows the directory tree to be searched as well as displaying available files. An example is shown in Figure 8.

At the top of the display is the "prompt" describing the type of file to be entered. The line underneath shows the current directory and this text will change if the directory is changed.

Figure 7: Example of a Graphical Form



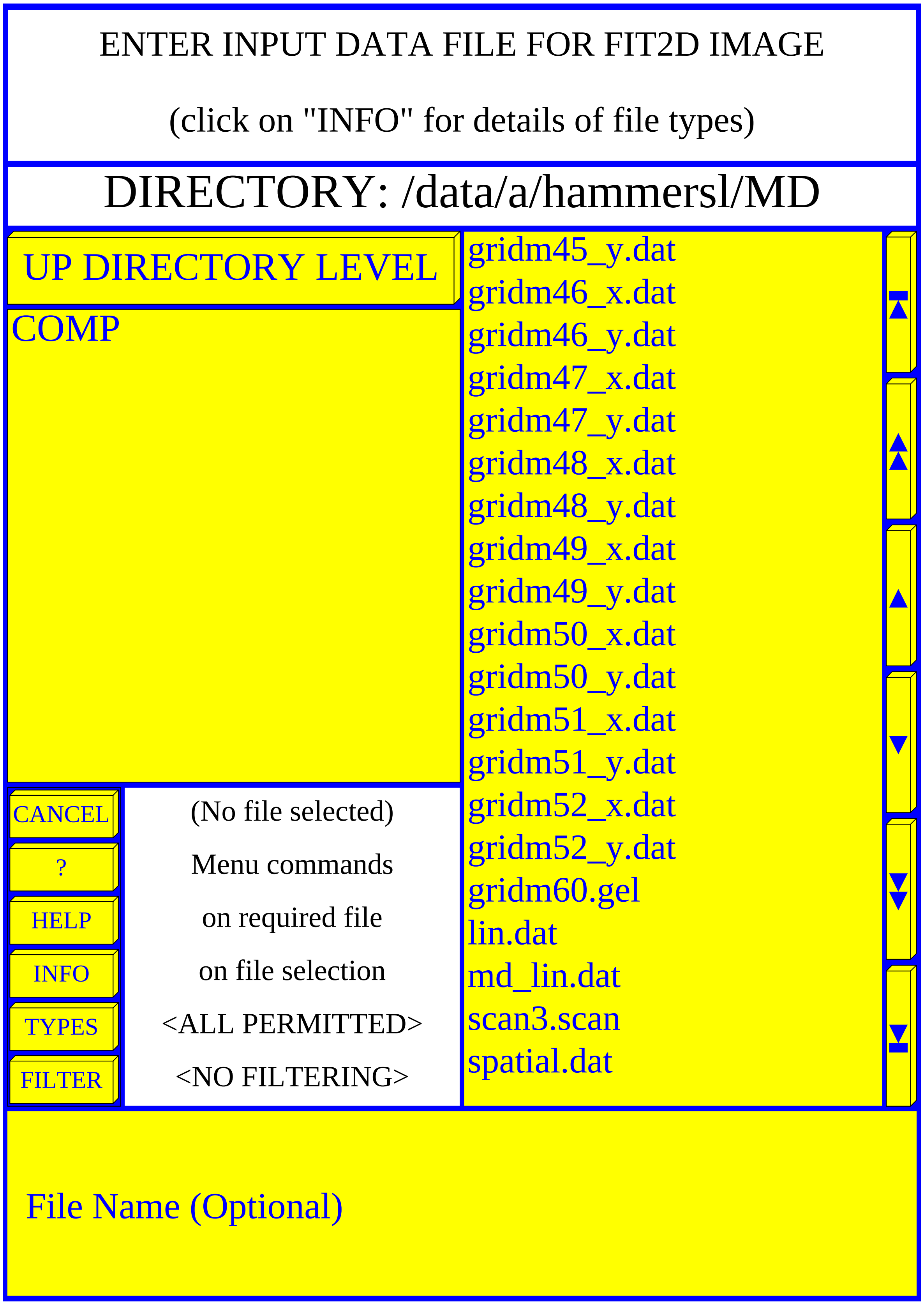
CONTROL OF RADIAL OR 2-THETA SCAN

RE-BINNING PARAMETERS

DESCRIPTIONS	VALUES	CHANGE
EQUAL 2-THETA ANGLE REBINNING	YES	EQUAL ANGLES
INTENSITY CONSERVATION	NO	CONSERVE INT.
APPLY POLARISATION CORRECTION	YES	POLARISATION
POLARISATION FACTOR	.99000	FACTOR
SAMPLE TO "DETECTOR" DISTANCE (MM)	315.000	DISTANCE
GEOMETRICAL CORRECTION TO INTENSITIES	YES	GEOMETRY COR.
X-PIXEL COORDINATE OF DIRECT BEAM	692.560	X-BEAM CENTRE
Y-PIXEL COORDINATE OF DIRECT BEAM	927.340	Y-BEAM CENTRE
ROTATION ANGLE OF PLANE OF TILT (DEGREES)	29.0460	TILT PLANE
TILT ANGLE OF DETECTOR (DEGREES)	.73600	TILT
MAXIMUM 2-THETA ANGLE OF SCAN (DEGREES)	12.7080	MAX. ANGLE
NUMBER OF BINS IN OUTPUT SCAN	876	SCAN BINS

Click on variable to change, or ’O.K.’

Figure 8: The File Selection Tool


ENTER INPUT DATA FILE FOR FIT2D IMAGE

(click on "INFO" for details of file types)

DIRECTORY: /data/a/hammersl/MD

(No file selected)

Menu commands

on required file

on file selection

<ALL PERMITTED>

<NO FILTERING>

gridm45_y.dat
gridm46_x.dat
gridm46_y.dat
gridm47_x.dat
gridm47_y.dat
gridm48_x.dat
gridm48_y.dat
gridm49_x.dat
gridm49_y.dat
gridm50_x.dat
gridm50_y.dat
gridm51_x.dat
gridm51_y.dat
gridm52_x.dat
gridm52_y.dat
gridm60.gel
lin.dat
md_lin.dat
scan3.scan
spatial.dat

File Name (Optional)

Underneath, on the left hand side is the "UP DIRECTORY" button which allows the directory tree to be mounted. The sub-directories of the current directory, if any, are displayed in a region on the left of the window and any files are displayed on the right. Both, when present, are displayed in blue writing on a yellow background (i.e. active), and if more directories of files are present than can be displayed "scroll" buttons appear to allow the list to be scrolled. Clicking on a directory name will change the current directory, and a new list of sub-directories and files will be displayed.

As well as the directory and file selection regions, there is a region for text input. The text is displayed at the bottom of the window. Thus, rather than clicking on a displayed file name it is possible to enter file names from the keyboard. The <ENTER> (or < RETURN>) key is used to complete entry of the name. If a valid file name is entered then this will be the selected file. If a valid directory name is entered, then the directory will be changed and a new set of files and sub-directories will be displayed. The <TAB> key may be used for automatic file name completion. Pressing the <TAB> key will search for possible completions of the entered text. If only one file or directory is possible, then this will be automatically selected. In case that more than one completion is possible then common characters will be automatically completed as far as possible. Immediately using the <TAB> key again will update the lists of files and directories which are possible completions of the file name. (Note: when this file completion facility is used, any filters or type matching (see below) is ignored.)

Additionally the following buttons follow the directory list:

CANCEL To cancel the selection of a file. This will cancel the input and return to a menu.

? List of button commands and short explanatory text.

HELP Help information on the particular type of file required.

INFO Help information on the use of the file selection tool.

TYPES A sub-set of files may be selected and displayed by specifying only files of a certain file extension to be "selectable". By clicking on the **TYPES** button the require extension of the files may be entered. e.g. `image` may be entered to select only MarResearch image plate scanner files. When a file type has been defined it is displayed in the information box, to the right of the **TYPES** button. In the example no file types have been defined so the information box contains the text `<ALL PERMITTED>`.

FILTER A sub-set of files may be selected and displayed by defining a file "filter". The asterisk is used as a "wild-card" so that a pattern may be defined and only files matching the pattern will be displayed in the file window. e.g. If "f*.dat" is entered, then only files which start with lower case "f" and are of file extension "dat" will be selectable. Any active filter is displayed to the right of the "FILTER" button. In the example no file filter has been defined so the information box contains the text `<NO FILTERING>`. (It should be noted that both "FILTER" and "TYPES" will act together if they are both activated.)

Having selected a file **FIT2D** will try to use the file extension to determine the data format. If the file extension is of a number of "known" types the appropriate input code will automatically be tried. e.g. The Molecular Dynamics image plate scanners save files with the extension `gel`, so all such files are read in automatically. If a file type is unknown, then a menu of available input formats is displayed, and the user can click on the correct format, or the **CANCEL** button if none are suitable. (The **BINARY** input type allows a very wide range of file types to be input, so long as the user knows the basic details of the format.)

5 Common GUI Commands and Sub-Menus

Important commands which appear in many different interfaces are listed below together with a brief description, and they are described in detail in the sections which follow:

INPUT: Input data from a file with the help of the file selection tool.

EXCHANGE: Swap the data in the current data array with the data in the "memory". If no data is present in the "memory" then the current data array becomes undefined.

Z SCALING: Sub-menu controlling the displayed intensity range and the choice of linear of logarithmic scaling. This sub-menu may be obtained directly from all interfaces when the intensity scaling colour bar (look-up table) is displayed.

DISPLAY: Sub-menu which allows a number of different forms of graphical display.

MOVEMENT: This sub-menu allows easy movement within a large data image.

OPTIONS: Sub-menu controlling graphical style options, such as grid, title, and axis text.

OUTPUT: Sub-menu allowing choice of output formats to save the contents of the current region of interest to a file.

5.1 The GUI INPUT Command

Most interfaces contain the **INPUT** command, and usually this will be the first command to be used on entering an interface. After pressing the **INPUT** command the file selection tool will appear, and help the user to find and select the file containing the required data. See Section 4.4, Page 35 for general guidelines on file selection.

Depending on the extension of the file selected and type of file format the data may be automatically input, or a file format menu, or an input form may appear.

The following file extensions and corresponding formats are automatically recognised as are upper case, or mixed case versions[9]:

`bin`: BINARY user specified format

`bsl`: BSL (Daresbury) / OTOKO (Hamburg) format

`chi`: CHIPLOT ASCII 1-D X/Y graph format

[9]Other file formats are likely to be added as needed, so this list is likely to grow.

`cor`: BINARY user specified format

`cor2`: BINARY user specified format

`corr`: BINARY user specified format

`edf`: "Klora" format **or** BINARY user specified format. The "Klora" format is a subset of the "ESRF Data format". The "ESRF Data format" is too general and wrongly specified to be usable. The "Klora" format has a header section of 1024 bytes, stores one image, and does not use any data compression. This is the default data format on many of the beam-lines at the ESRF.

`f2d`: FIT2D format

`final`: BINARY user specified format

`gel`: Molecular Dynamics IMAGEQUANT (tiff) format

`inf`: FUJI BAS-2000 (BAS-1500) format

`info`: BINARY user specified format

`image`: MarResearch IP scanner format

`img`: FUJI BAS-2000 (BAS-1500) format **or** HAMAMATSU CCD format. The file type is recognised from the start of the file.

`mar1200`: MarResearch IP scanner format

`mar1600`: MarResearch IP scanner format

`mar2000`: MarResearch IP scanner format

`mar2300`: MarResearch IP scanner format

`mar3450`: MarResearch IP scanner format

`pck`: Old compressed MarResearch format

`pmi`: PHOTOMETRICS CCD format

`spe`: PRINCETON INSTRUMENTS CCD format

`tif`: Adobe TIFF format

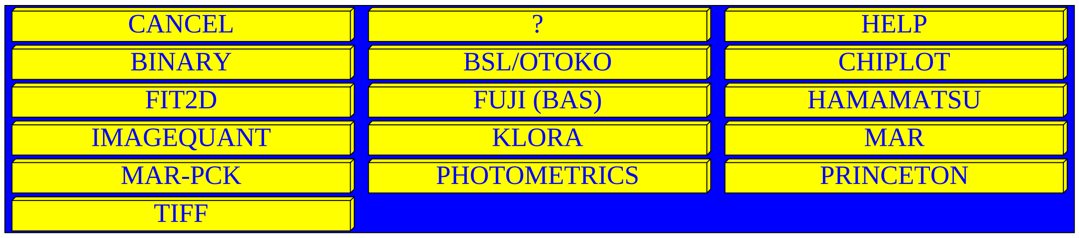
`tiff`: Adobe TIFF format

If the file has an unrecognised extension, then the **UNKNOWN EXTENSION: SELECT FILE FORMAT:** menu will appear. This menu is shown in Figure 9.

Most file formats are automatically input, but some do not have headers which fully define the necessary information to input the data. The `BINARY` format is the best explain of this. This is a very flexible input option which allows the user to define all the information concerning the formats and allows almost any non-compressed binary data to be input provided the user knows the details of the format, or finds them by trail and error. The `BINARY` format input parameter form is shown in Figure 10.

The form buttons allow the following input parameters to be controlled:

Figure 9: The **UNKNOWN EXTENSION: SELECT FILE FORMAT:** Menu



X-PIXELS: The number of pixels to be defined and input in the fastest changing direction

Y-PIXELS: The number of pixels to be defined and input in the more slowly changing direction

DATA TYPE: The data type used to store a pixel value. Presently the following types are supported:

BYTE VALUES: Single byte per pixel integers

4-BYTE INTEGER: 4-bytes per pixel integers

INTEGER (2-BYTE): 2-bytes per pixel integers

REAL (4-BYTE IEEE): 4-byte per pixel IEEE floating point reals.

SIGNED: For integers data types, whether or not the values are signed or unsigned.

BYTE SWAP: For multiple byte per pixel data types, this sets whether or not the bytes are to be reversed in their ordering on input. This is necessary for data written on a little-endian machine and read in on a big-endian machine, or vice versa.

STARTING BYTE: This allows the possibility of ignoring header sections at the start of binary files. If, for example, a file has a fixed length header of 1024 bytes, so that the image data starts at byte number 1025 (numbering from 1), then this value can be set to 1025.

When the form has been set, the **O.K.** button can be clicked and the data will be input.

Even with unknown formats trail and error can be used to find the correct parameters of the format provided the correctly input data can be recognised.

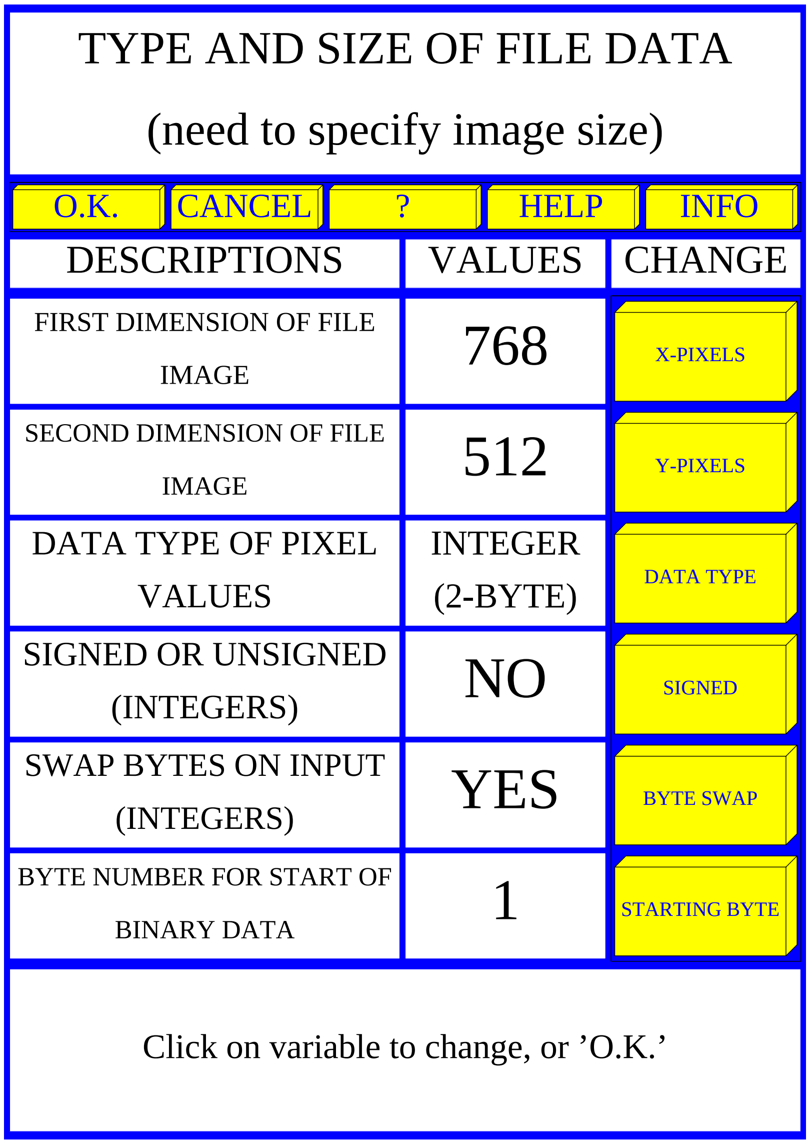
Since some of the formats are often used with very large files, a sub-region input menu can also appear.

Further information on the available file formats for input may be found in Section 15.51, Page 129.

5.2 The GUI EXCHANGE Command

The **EXCHANGE** command is present in many interface menus and sub-menus, and is the essence to using **FIT2D** for data analysis. **EXCHANGE** swaps the data in the current data

Figure 10: The `BINARY` Format Input Parameter Form



TYPE AND SIZE OF FILE DATA

(need to specify image size)

DESCRIPTIONS	VALUES	CHANGE
FIRST DIMENSION OF FILE IMAGE	768	X-PIXELS
SECOND DIMENSION OF FILE IMAGE	512	Y-PIXELS
DATA TYPE OF PIXEL VALUES	INTEGER (2-BYTE)	DATA TYPE
SIGNED OR UNSIGNED (INTEGERS)	NO	SIGNED
SWAP BYTES ON INPUT (INTEGERS)	YES	BYTE SWAP
BYTE NUMBER FOR START OF BINARY DATA	1	STARTING BYTE

Click on variable to change, or 'O.K.'

array with any data in the "memory" array. If no data is present in the "memory" the current data array is left undefined.

EXCHANGE is the way in which to "charge" data in the "memory" so that binary image processing operations can be used. See Section 7.4, Page 73 for a detailed example of dividing one image by another.

EXCHANGE is also very useful for recovering the original data after many operations within the GUI. The output of GUI operation, if any, is always in the current data array, but the original data is often transferred to the "memory". so after such an operation, **EXCHANGE** can be used to recover the original data for further analysis.

5.3 The GUI Z SCALING Command

The **Z-SCALING** menu allows the range of intensity values displayed to be changed, and to control the manner in which data array intensity values are converted to output pixel colours.

Figure 11 is an example of the false colour display of 2-D data, together with the **Z-SCALING** menu for changing the manner in which pixel intensity values are displayed as different colours or grey levels.

The following commands are available:

EXIT: Exits **Z SCALING** sub-menu and return to calling menu.

+ MAXIMUM: Set fixed range scaling and increase maximum limit by 10% of the scaling range.

USER MIN/MAX: Prompt user for `MINIMUM DISPLAY VALUE` and for `MAXIMUM DISPLAY VALUE`, and set fixed range scaling to these values.

?: List of commands with short description of each command.

- MAXIMUM: Set fixed range scaling and decrease maximum limit by 10% of the scaling range.

USER MINIMUM: Prompt user for `MINIMUM DISPLAY VALUE` and set fixed minimum value for the displayed intensity range. The maximum is automatically defined by the ROI data values, and re-defined each time the ROI is changed or new data are displayed.

FULLY AUTOMATIC: Full range automatic scaling is set. The ROI is searched for the minimum and maximum values, and the scaling range is set. Each time the ROI is changed or new data are displayed the scaling range is re-calculated.

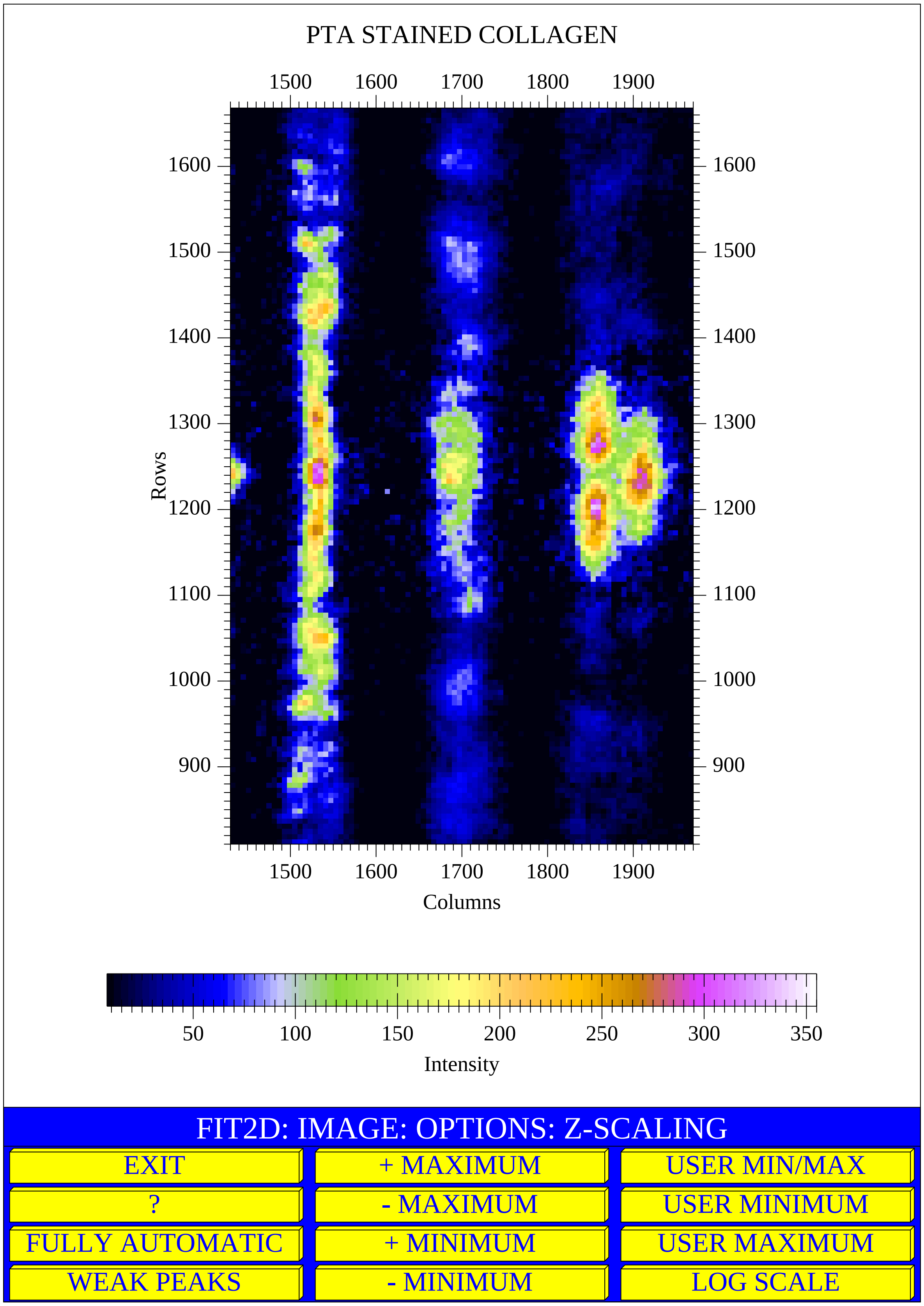
+ MINIMUM: Set fixed range scaling and increase minimum limit by 10% of the scaling range.

USER MINIMUM: Prompt user for `MAXIMUM DISPLAY VALUE` and set fixed maximum value for the displayed intensity range. The minimum is automatically defined by the ROI data values, and re-defined each time the ROI is changed or new data are displayed.

WEAK PEAKS This selects reduced range automatic scaling. This is designed to display the weak features of diffraction data. Often full range scaling is not very informative since the intensity range is dominated by a very few strong reflections, and little detail is seen in the majority of the image. In this mode, the average value and standard deviation of intensity values are calculated in several regions of the image. The minimum of the display intensity range is then calculated from the minimum of the mean minus three times the standard deviation, and the maximum is set to the maximum of the mean plus five times the standard deviation from each of the regions. Each time the ROI is changed or new data displayed the range is re-calculated.

- MINIMUM: Set fixed range scaling and decrease minimum limit by 10% of the scaling range.

Figure 11: Example of False Colour Image Data Display



LOG SCALE: Display the data on a logarithmic scale. (The **LOG SCALE** button is replaced with the **LINEAR SCALE** button.)

LINEAR SCALE: Display the data on a linear scale. (The **LINEAR SCALE** button is replaced with the **LOG SCALE** button.)

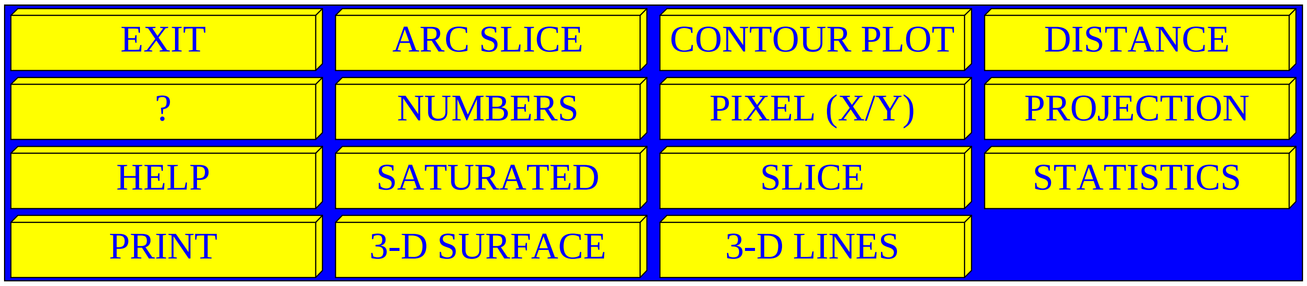
As well as linear scaling, logarithmic scaling is available through the **LOG SCALE** button. When logarithmic scaling is enabled this button is replaced by the **LINEAR SCALE** button. If zero or negative range limits are encountered when automatic range logarithmic scaling is being used, a small positive value will be used instead.

The **Z SCALING** menu may also be obtained by clicking within the intensity scale colour bar when in the main menus of the different interfaces. A "rubber-band" line appears from where the user clicked. Any of the menu commands can be clicked, or the intensity scale colour bar may be clicked a second time. If clicked a second time the two values within the scale are used to set fixed range scaling. If the user clicks outside both the menu and the intensity scale colour bar, then fully automatic scaling will be set. This allows quick and reasonably precise re-scaling from any interface.

5.4 The GUI DISPLAY Command

The **DISPLAY** menu allows other graphical display possibilities in addition to the standard false colour image display. Figure 12 shows the menu button choices.

Figure 12: The Display Menu



The buttons have the following functions:

EXIT Exit display menu

ARC SLICE Define an arc, by entering the end coordinate and a coordinate on the arc, and calculate the 1-D "slice" along this arc. This is displayed as an X/Y graph and is saved in the memory.

CONTOUR PLOT Display the *ADR* as a contour plot

DISTANCE Allows two coordinates to be entered graphically, and calculated the distance between the coordinates in pixel units, and in millimetres, using the currently defined pixel sizes.

? Explanation on each of the menu options

NUMBERS Allow the user to click on coordinates and output the 11 by 11 square of pixel values around the coordinate. (If a log file is open this will be saved in the log file.)

PIXEL (X/Y) Allow the user to click on pixels and display information on the coordinate and pixel values. If the geometry of a diffraction experiment has been defined, then angles and equivalent D-spacings will be calculated and displayed.

PROJECTION Allow the user to define an arbitrary line, and a region either side of the line, and "project" the pixel values in the defined region onto the line. This is displayed as an X/Y graph and is saved in the memory. (This is similar to **SLICE** but allows averaging over several or many pixels.)

HELP Help text explaining the available commands.

SATURATED Count the number of pixels with an intensity greater or equal to an input value.

SLICE Allow the user to click on two coordinates to define an arbitrary "slice" through the data. The nearest pixels to the line define the output data values. This is displayed as an X/Y graph and is saved in the memory.

STATISTICS Allow the user to define a arbitrary polygon region, and calculate a number of statistics within this region. e.g. number of pixels, total intensity, mean and standard deviation values. By first defining a background region, followed by a region around a diffraction peak, a primitive form of peak integration is available.

PRINT Save the current graphics display to a PostScript file.

3-D SURFACE The *ADR* is displayed a 3-D surface perspective view. The surface may additionally be coloured by the current colour table. The viewing angle and many aspects of the display are controlled by a menu. An example of the display and menu are shown in Figure 13. This menu is described in the next section.

3-D LINES Draw the *ADR* as a 3-D line plot, with hidden line removal.

5.4.1 The 3-D SURFACE Sub-Menu

The **3-D SURFACE** control sub-menu is shown in Figure 13.

The following commands are available:

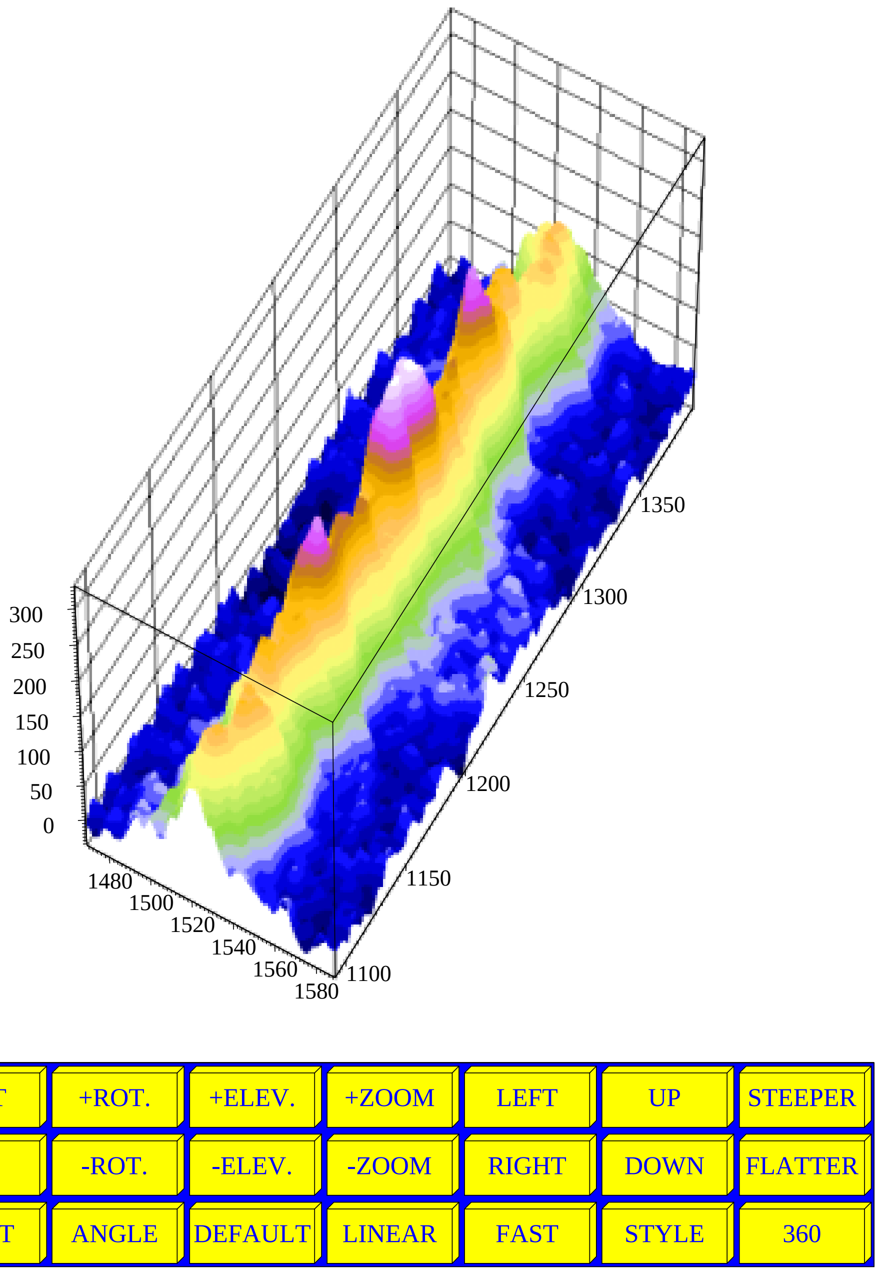
EXIT Exit 3-D surface viewing sub-menu.

+ROT. Increase longitude angle of viewer.

+ELEV. Increase elevation angle of viewer.

+ZOOM Increase size of object.

Figure 13: Example of the 3-D Surface "Viewer"



LEFT View further to the left of the object.

UP View further above the object.

STEEPER Increase importance of Z-dimension (vertical).

? Explanation of menu options.

-ROT. Decrease longitude angle of viewer.

-ELEV. Decrease elevation angle of viewer.

-ZOOM Decrease size of object.

RIGHT View further to the right of the object.

DOWN View further below the object.

FLATTER Decrease importance of Z-dimension (vertical).

PRINT Output displayed 3-D surface to PostScript output file.

ANGLE Keyboard entry of arbitrary viewing angles.

DEFAULT Re-set default view angles, and zoom factor.

LOG Logarithmic image intensity scaling. When selected this button is replaced by **LINEAR** to allow selection of linear intensity scaling.

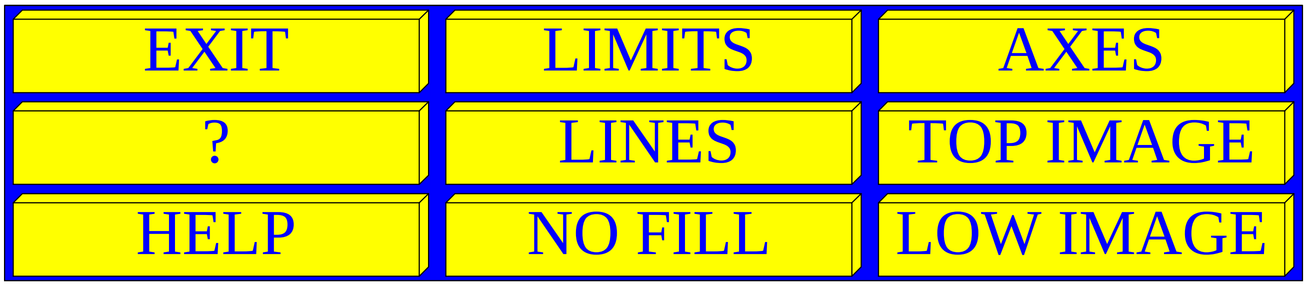
FAST Fast mode of display: lower resolution, and few colours are displayed. This results in many fewer polygons being calculated so display is much faster. This is useful for finding the best viewing angle. When the required view has been found the display style can be re-set to **NORMAL**.

STYLE Control of 3-D surface display diagram style. This starts a sub-menu which is described below.

360 360 degree rotation in longitude, in 10 degree steps. (Note: This may take some time on older systems and there is no cancel button !)

The **STYLE** button starts a sub-menu which controls style aspects of the 3-D surface display. This sub-menu is shown in Figure 14.

Figure 14: 3-D Surface Style Control Menu



The style sub-menu has the following commands:

EXIT Exit style sub-menu.

? Explanation of menu options.

HELP Help text.

LIMITS Set output pixel resolution limit and number of colours.

LINES Draw 3-D lines on surface around the re-binned pixels.

FILL Fill areas of 3-D surface with colour.

AXES Draw enumerated axis around the 3-D surface.

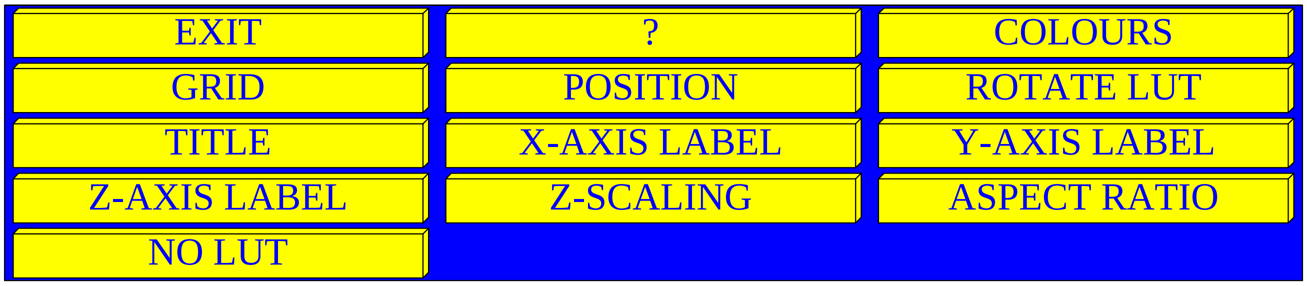
TOP IMAGE Add 3-D projected false colour image above 3-D surface.

LOW IMAGE Add 3-D projected false colour image below 3-D surface.

5.5 The GUI OPTIONS Command

The **OPTIONS** menu allows the style of the displayed false colour image to be altered. The sub-menu is shown in Figure 15.

Figure 15: The Graphics Display Style Options Menu



The following command are available:

EXIT: Exit **OPTIONS** menu

?: Explanation on each of the menu options

COLOURS: Choice of different colour tables e.g. grey scale, inverse grey scale (black is more intense), “temperature”. Several of the different colour tables are shown throughout this manual.

CURVE STYLES: Allows the style of curves to be altered. Each set of data coordinates may be displayed by a line through the coordinates, markers drawn at the coordinates, and error boxes drawn around the coordinates. (Error boxes will only be drawn if error estimates are defined.) All aspects of the lines and markers may be set. e.g. line type (solid, dotted, dashed, dot-dashed), line widths, colours, and marker types (15 choices). If the markers define closed shapes, the interiors may be optionally filled with a different colour.

GRID: Add or remove horizontal and vertical grid lines to the image display. Both coarse (every large axis tick mark) and fine (every small axis tick mark) grid lines may be set.

POSITION: Change the position for the image display in the graphics window. This allows you to change the size of the image display, or other graphics, within the graphics window. You click on opposite corners of the position where the image should be displayed. The coordinates input are those of the maximum region used for the axes frame of the image, contour plot or graph. Since the title and axis label positions are 'attached' to the axes frame this also changes their position. Note: This is a maximum size, if correct aspect ratio display is being used (the default) the actual region may be smaller if the image aspect ratio is not the same as the region.

ROTATE LUT: Change cyclically the colours used to display different intensity levels. This can emphasise different features in the data.

TITLE: Change the title of the image.

X-AXIS LABEL: Change the text used to label the X-axis of the image.

Y-AXIS LABEL: Change the text used to label the Y-axis of the image.

Z-AXIS LABEL: Change the text used to label the intensity scale (Z-scale) of the image. This is drawn with the look-up table intensity scale

Z-SCALING: Change the mapping of data values to pixel colours (see Section 5.3 for further details).

ASPECT RATIO: Change the "aspect ratio" for image display. By default pixels are drawn square, so if the *ADR* is rectangular the image is rectangular. When many more pixels are defined in one direction than the other, it can be more useful to allow the pixels to be rectangular and allow the image to fill the full region available. These modes can be toggled.

NO LUT: Don't draw the intensity scale colour bar, or "look-up table". When clicked, this is replaced by the button **DRAW LUT**.

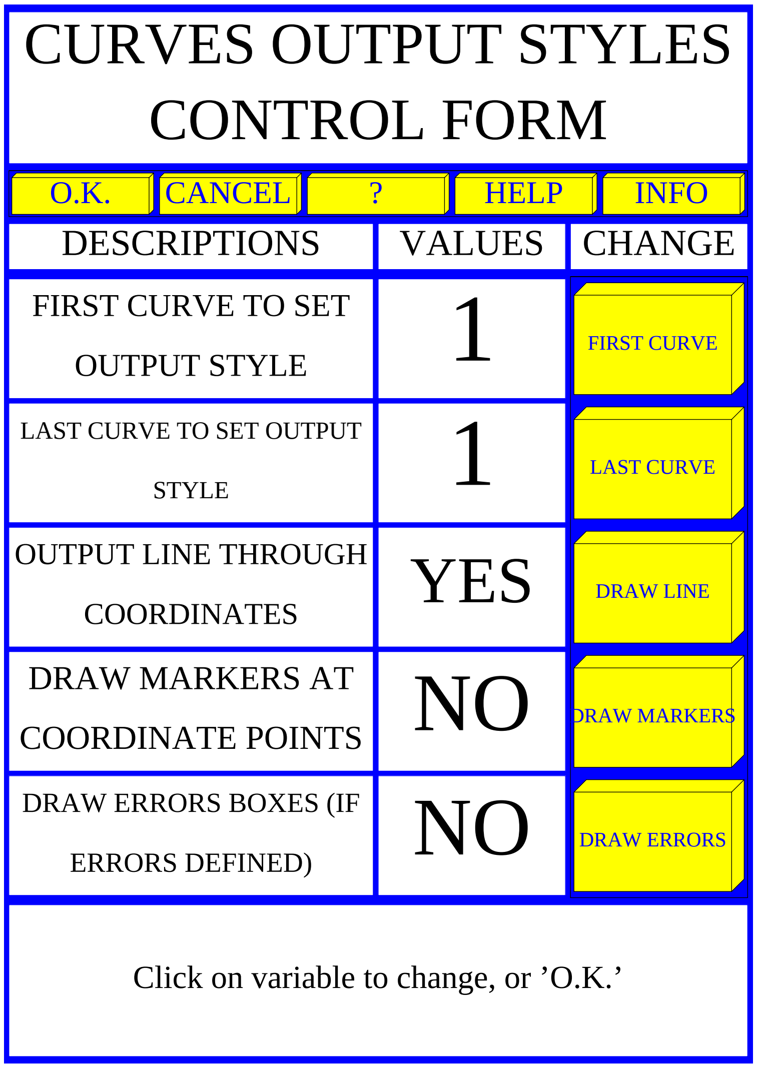
The **CURVE STYLES** command allows the output style of 1-D data-sets to be controlled. A data-set may be drawn with a series of lines through the coordinates, a series of markers, and if error estimates are defined as a series of error boxes. The **CURVES OUTPUT STYLES CONTROL FORM**, shown in Figure 16, Page 51, allows the output styles to be set for individual curves. (Up to 15 different curve styles may be set, although generally in **FIT2D** only 1 or 2 1-D data-sets are output at a time.)

If lines are to be output the **LINE OUTPUT STYLE CONTROL FORM** appears. This is shown in Figure 17, Page 52. This allows the following attributes of the line to be set:

LINE TYPE: Type of line drawn between the coordinates:

1. Solid line
2. Dashed line
3. Dotted line

Figure 16: The Curves Output Styles Control Form



CURVES OUTPUT STYLES CONTROL FORM

DESCRIPTIONS	VALUES	CHANGE
FIRST CURVE TO SET OUTPUT STYLE	1	FIRST CURVE
LAST CURVE TO SET OUTPUT STYLE	1	LAST CURVE
OUTPUT LINE THROUGH COORDINATES	YES	DRAW LINE
DRAW MARKERS AT COORDINATE POINTS	NO	DRAW MARKERS
DRAW ERRORS BOXES (IF ERRORS DEFINED)	NO	DRAW ERRORS

Click on variable to change, or 'O.K.'

4. Dot-dash line

COLOUR: Colour of line: Choice of primary, and secondary colours, white, and black.

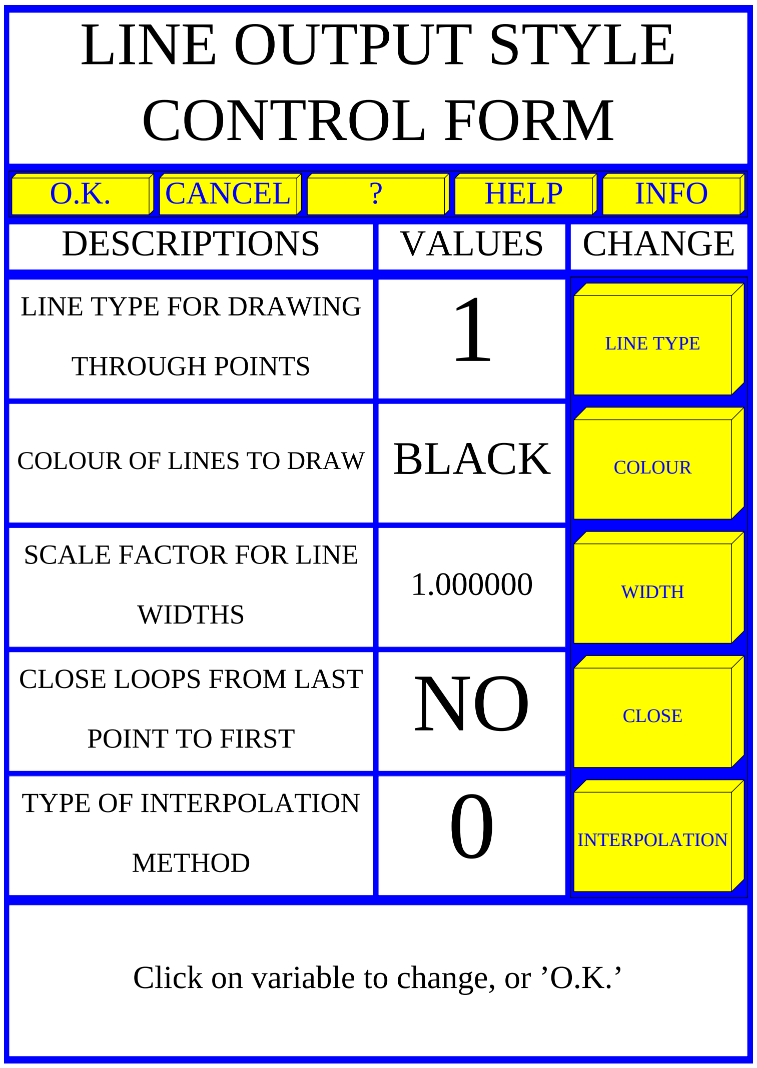
WIDTH: Scale factor for line thickness. (Graphics terminals are too low resolution to see the difference in thickness, except for very thick lines, but lines will appear thicker on PostScript hardcopy output.)

CLOSE: Whether a line should be drawn from the last data point to the first data point (usually not).

INTERPOLATION: Type of interpolation to use to draw the line between the data points:

- 0: Linear i.e. straight lines
- 1: Cubic spline i.e. smooth lines going through the data points, but with the maximum and minimum possibly going outside the range set by the two coordinates.
- 2: Piecewise monotonic i.e. smooth lines going through the data points, and with the maximum and minimum within the range set by the two coordinates.

Figure 17: The Line Output Style Control Form



LINE OUTPUT STYLE CONTROL FORM

O.K. | CANCEL | ? | HELP | INFO

DESCRIPTIONS	VALUES	CHANGE
LINE TYPE FOR DRAWING THROUGH POINTS	1	LINE TYPE
COLOUR OF LINES TO DRAW	BLACK	COLOUR
SCALE FACTOR FOR LINE WIDTHS	1.000000	WIDTH
CLOSE LOOPS FROM LAST POINT TO FIRST	NO	CLOSE
TYPE OF INTERPOLATION METHOD	0	INTERPOLATION

Click on variable to change, or 'O.K.'

If markers are to be output the **MARKERS OUTPUT STYLE CONTROL FORM** appears. This is shown in Figure 18, Page 53. This allows the following attributes of the markers to be set:

MARKER: Type of marker:

1. Dot "." (Not scalable)
2. Cross "+"
3. Asterisk "*"
4. Circle "o"
5. Diagonal cross "X"
6. Square
7. Rotated square (Kite)
8. Triangle (Base a bottom)

9. Triangle (Point a bottom)
10. Six-pointed star
11. Convolution sign (circle with diagonal cross)
12. Circle with cross
13. Two coloured cross

COLOUR: Colour of lines used for drawing markers.

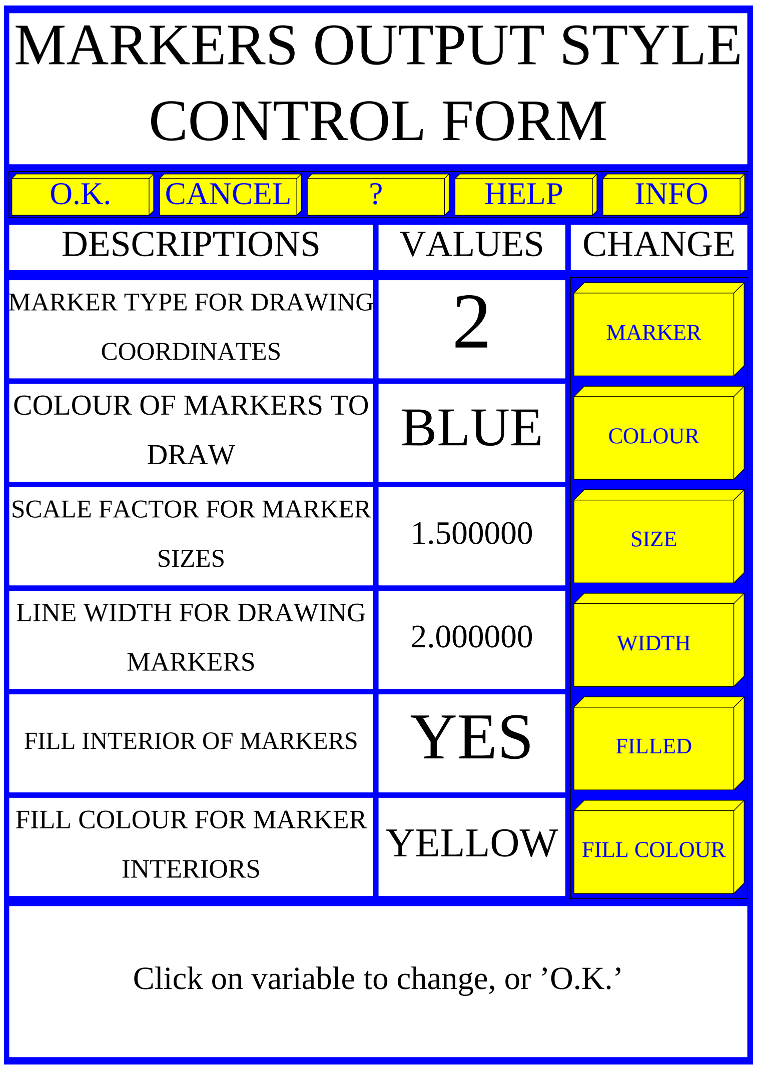
SIZE: Size scale factor.

WIDTH: Line width scale factor of lines used to draw markers.

FILLED: Whether or not the interior of markers which defined a closed area is to be filled with colour.

FILL COLOUR: The colour used to fill the interiors of any filled markers.

Figure 18: The Markers Output Style Control Form


MARKERS OUTPUT STYLE CONTROL FORM

DESCRIPTIONS	VALUES	CHANGE
MARKER TYPE FOR DRAWING COORDINATES	2	MARKER
COLOUR OF MARKERS TO DRAW	BLUE	COLOUR
SCALE FACTOR FOR MARKER SIZES	1.500000	SIZE
LINE WIDTH FOR DRAWING MARKERS	2.000000	WIDTH
FILL INTERIOR OF MARKERS	YES	FILLED
FILL COLOUR FOR MARKER INTERIORS	YELLOW	FILL COLOUR

Click on variable to change, or ’O.K.’

If error boxes to be output the **ERROR BOXES OUTPUT STYLE CONTROL FORM** appears. This is shown in Figure 19, Page 55. This allows the following attributes of the error boxes to be set:

BOX TYPE Type of representation for the error estimates:

- 0: Cross hair
- 1: Kite
- Rectangle

Note: If the error estimates are only defined in one direction these are all drawn the same.

LINE TYPE Type of line used to drawn the error boxes:

1. Solid line
2. Dashed line
3. Dotted line
4. Dot-dash line

COLOUR: Colour of lines used for drawing the error boxes.

WIDTH: Line width scale factor of lines used to draw the error boxes.

5.6 The GUI MOVEMENT Command

The **MOVEMENT** command allows easy "movement" within a large image. i.e. The user can zoom-in to a small sub-region of the image, and change the position of the sub-region to move around the image. This is very useful for examining large images. This menu is also obtained by using the `IMAGE` command from with the "KEYBOARD" menu. The **MOVEMENT** sub-menu is shown in Figure 3, Page 26.

The commands available are:

EXIT: Exit sub-menu and return to calling menu.

L-TOP: Move to the top left-hand corner of the current defined data image.

TOP: Move to top of the current defined data image.

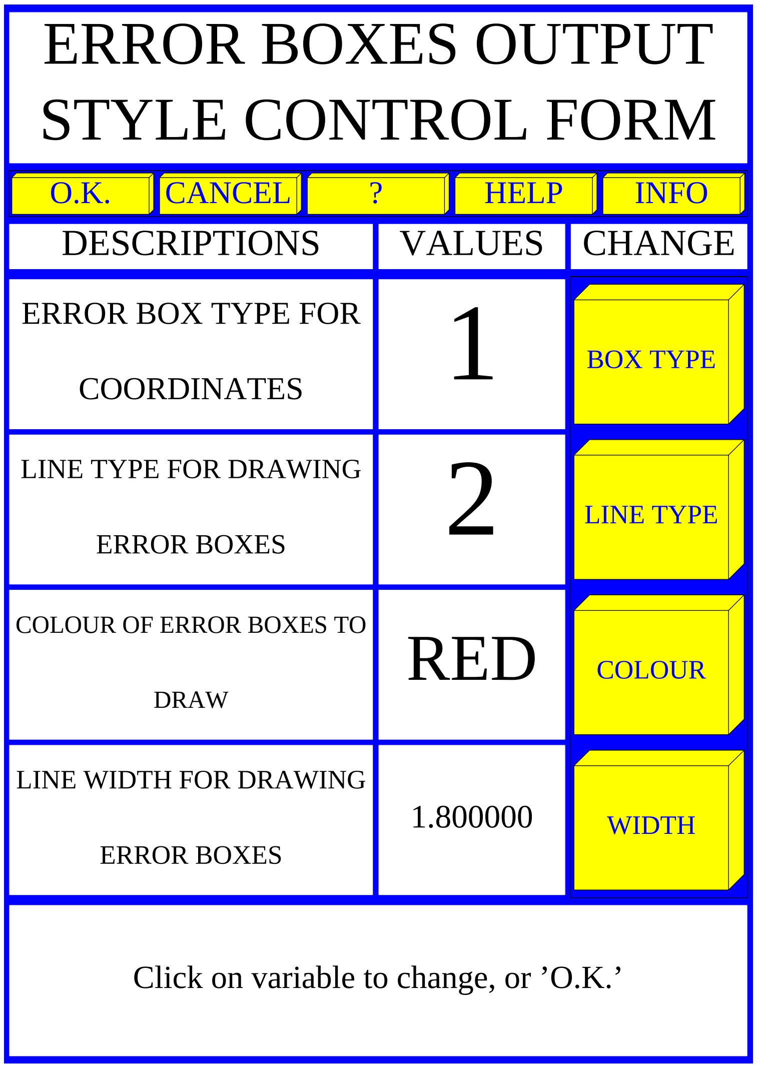
R-TOP: Move to the top right-hand corner of the current defined data image.

OPTIONS: Enter the graphical style control **OPTIONS** sub-menu (see Section 5.5, Page 49).

?: List of commands with brief description.

L-UP: Move left and upwards. The centre of the displayed sub-region will move to the top left-hand corner, provided the data is defined.

Figure 19: The Error Boxes Output Style Control Form



ERROR BOXES OUTPUT STYLE CONTROL FORM

DESCRIPTIONS	VALUES	CHANGE
ERROR BOX TYPE FOR COORDINATES	1	BOX TYPE
LINE TYPE FOR DRAWING ERROR BOXES	2	LINE TYPE
COLOUR OF ERROR BOXES TO DRAW	RED	COLOUR
LINE WIDTH FOR DRAWING ERROR BOXES	1.800000	WIDTH

Click on variable to change, or 'O.K.'

UP: Move upwards. The centre of the displayed sub-region will move to the top edge, provided the data is defined.

R-UP: Move right and upwards. The centre of the displayed sub-region will move to the top right-hand corner, provided the data is defined.

ZOOM IN: Zoom in to a new, smaller sub-region by interactive graphical coordinate clicking.

FAR-LEFT: Move to the left edge of the current defined data image.

LEFT: Move left. The centre of the displayed sub-region will move to the left edge, provided the data is defined.

CENTRE: Move to the centre of the defined data image.

RIGHT: Move right. The centre of the displayed sub-region will move to the right edge, provided the data is defined.

FAR-RIGHT: Move to the right edge of the current defined data image.

UN-ZOOM: Make the ROI bigger, symmetrically about the existing ROI if possible. The size is increase by 50 pixels or by 50%; whichever is the larger.

L-DOWN: Move left and downwards. The centre of the displayed sub-region will move to the lower left-hand corner, provided the data is defined.

DOWN: Move downwards. The centre of the displayed sub-region will move to the lower edge, provided the data is defined.

R-DOWN: Move right and downwards. The centre of the displayed sub-region will move to the lower right-hand corner, provided the data is defined.

DISPLAY: Enter the **DISPLAY** sub-menu which contains a variety of alternative graphics display styles (see Section 5.4, Page 45).

FULL: Set the ROI to be the whole of the defined data.

L-BOTTOM: Move to the bottom left-hand corner of the current defined data image.

BOTTOM: Move to lower edge of the current defined data image.

R-BOTTOM: Move to the lower right-hand corner of the current defined data image.

PRINT: Save the currently defined graphics as a PostScript file.

It should be noticed that the menu movement commands are arranged in a cross. The position in the cross corresponds to the affect the command has in moving within the image.

5.7 The GUI MASK Command

The **MASK** command is available in a number of interfaces, and will gradually be generalised. At present it is used to mask out "bad" or contaminated data pixels from regions to be integrated, or fitted.

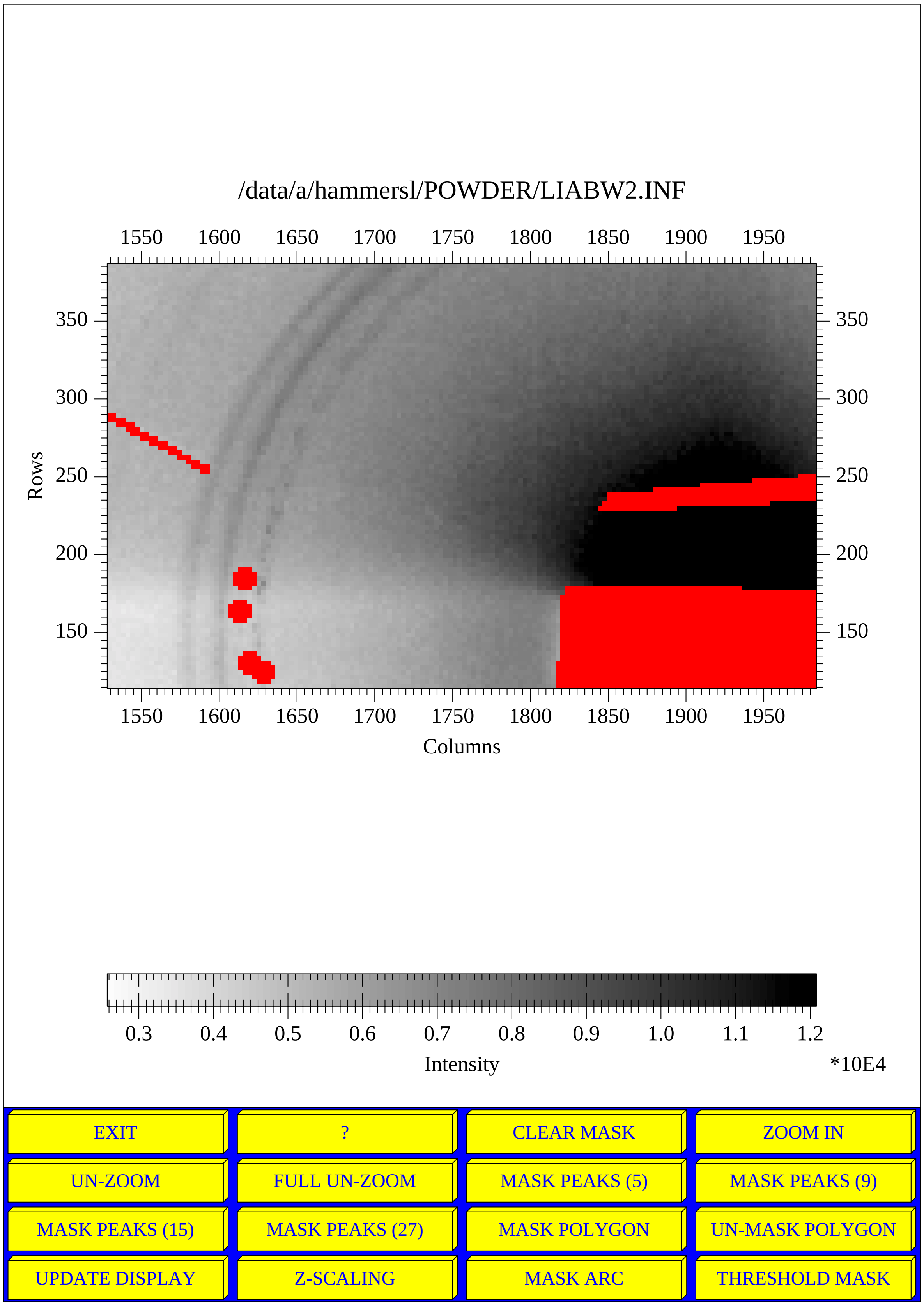
Allowing elements to be arbitrarily included or excluded from fitting and other operations is a very powerful facility which allows much flexibility in the data analysis process. "Bad" data points can be excluded from operations. Optimum non-rectangular data regions can be selected reducing calculations for fitting and other calculation intensive operations.

The "mask" is a 2-D area of the same size as the current data and memory arrays. Each data element may be "masked" or not. By default no elements are masked.

"masked-off" areas can be defined graphically for small blemishes, or for large regions, or from the data values themselves by using the **THRESHOLD MASK** command. This allows a threshold value to be defined and all data elements with values above, or below (as required), this value are defined as "masked-off"

The current ROI is displayed as a 2-D pixel image together with the graphical menu giving a choice of commands allowing the masking and un-masking of regions of the data. "Masked-off" pixels are displayed using a single colour. An example of this display and menu are given in Figure 20.

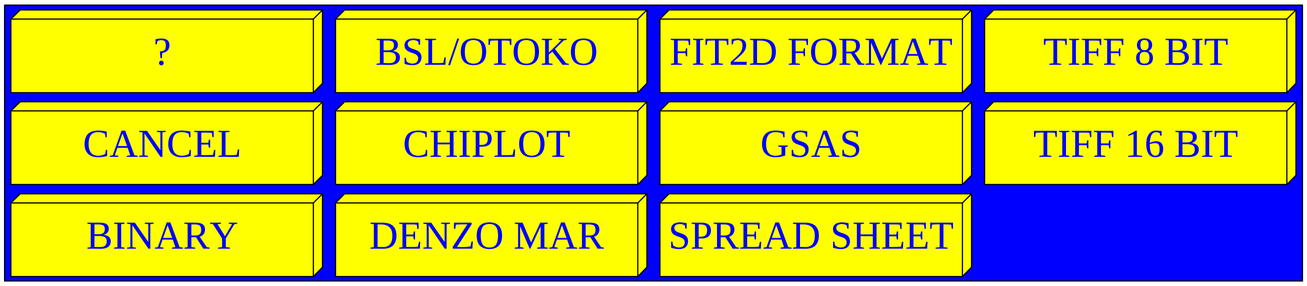
Figure 20: Example of the **MASK** Sub-Menu



The available choices are:

- **EXIT**: Exit from **MASK** sub-menu and return to calling menu.
- **?**: Display list of commands with a short description.
- **CLEAR MASK**: Set all elements in the ROI to be un-masked.
- **ZOOM IN**: Graphical region definition.
- **UN-ZOOM**: Make region displayed bigger.
- **FULL UN-ZOOM**: View whole of data image.
- **MASK PEAKS (5)**: Mask out peaks (Diameter 5 pixels). The user is prompted to select graphically the centre of each peak to be masked out with a circle of diameter 5 pixels. When finished click within the graphical prompt box. The display is updated with the maksed out pixels being dispalyed in the masking colour (red by default).
- **MASK PEAKS (9)**: Mask out peaks (Diameter 9 pixels). Works in the same manner as **MASK PEAKS (5)**.
- **MASK PEAKS (15)**: Mask out peaks (Diameter 15 pixels) Works in the same manner as **MASK PEAKS (5)**.
- **MASK PEAKS (27)**: Mask out peaks (Diameter 27 pixels) Works in the same manner as **MASK PEAKS (5)**.
- **MASK POLYGON**: Coordinate definition of region to mask off The user clicks on the corners of an arbitrary polygon region. When finished click within the graphical prompt box. The polygon is closed by a line going from the last entered coordinate to the first. Normally this should be a simple convex shape.
- **UN-MASK POLYGON**: Coordinate definition of region to un-mask. Works in the same manner as **MASK POLYGON**.
- **UPDATE DISPLAY**: Re-draw image, including masked regions
- **MASK ARC**: Define an arc region to be masked-off. If the arc is made large enough a circular region can be masked. .
- **THRESHOLD MASK**: Allows pixels within the ROI to be masked if they are less than an entered minimum threshold value, or alternatively if they are greater than a maximum threshold value. First the user is prompted for **LESS THAN COMPARISON**. If **NO** is selected a greater than comparison with be used. Then the **DECISION THRESHOLD DATA VALUE** is entered. The masked off pixels are added to any existing masked pixels, and the display is updated.

Figure 21: The GUI **OUTPUT** Command File Format Menu



5.8 The GUI OUTPUT Command

The **OUTPUT** command allows the current ROI to be save to a user specified file. If all of the data is to be saved, make sure that the ROI is set to all of the data prior to using the **OUTPUT** command. The **OUTPUT** command file format choice menu is shown in Figure 21.

The presently available commands are:

?: Short description of each of the available choices

CANCEL: Exit menu without saving data

BINARY: Save ROI as pure binary "dump" with a choice of data types.

BSL/OTOKO: Daresbury BSL / Hamburg OTOKO: KOHALA format

CHIPLOT: 1-D row or column output for X/Y graph plotting in a simple ASCII column format

DENZO MAR: Output format in Mar format for input to DENZO

FIT2D FORMAT: Self describing readable binary for re-entry to **FIT2D**

GSAS: GSAS powder diffraction format

SPREAD SHEET: ASCII, one line of ADR in one record

TIFF 8 BIT: TIFF integer pixel storage with 1 byte per pixel. This is suitable for exporting data to image display programs such as `xv`.

TIFF 16 BIT: TIFF integer pixel storage with 2 bytes per pixel

Depending on the format chosen, further prompts and graphical forms may appear. The name of the output file will be defined.

6 The FILE SERIES Interface

The commands within the **FILE SERIES** Interface allow operations on a whole series of files. (This is also possible using macros, but here the most useful operations are "hard-coded" and this allows greater efficiency in some cases.) Further commands are likely to be added in the near future, so you may find an expanded menu.

The following commands are available:

EXIT: Exits **FILE SERIES** interface and returns to main "scientific interface" menu. Note: data can be moved from one interface to another.

?: Help text with short explanation of commands.

HELP: Detailed help text on the interface and the available commands.

AVERAGE: Calculates the average value from a series of input files. See below for details of the specification of the file series.

COMPOSITE: Inputs a series of files and displays them all together in a composite image. The composite image is actually formed in the main data array, so can be used for data processing afterwards. See below for details of the specification of the file series.

DISPLAY: Enter the GUI general display menu, assuming that data has been defined. This is used to display the result of an **AVERAGE** or **COMPOSITE** operation, in a variety of different styles.

EXCHANGE: transfers data from the "main program array" to the "memory array", and vice versa. This may be useful to save some data whilst another operation is overwriting the "main program array".

INTEGRATE: Integration of a series of images from a file series to a choice of 1-D scans (V9.130). The integration is exactly the same as the **INTEGRATE** command in the **POWDER DIFFRACTION** interface (see Section 11.3, Page 90). This option is described in greater detail in Section 6.3, Page 64.

OPTIONS: Starts the general GUI **OPTIONS** sub-menu (See Section 5.5, Page 49) which allows various display options to be modified, and control of the graphical output style. Used to alter the colour table, etc. after data has been input.

OUTPUT: General GUI output menu. Saves the current "region of interest (ROI)" in a selected file with a choice of file formats.

PRINT: Save the currently displayed data image as a PostScript file. You are prompted for the name of the output file to create. After the file is fully written, it may be sent to a PostScript printer for printing. Note: It can take some time to create a file of a large 2-D data-set, and often even longer for the file to be printed.

6.1 Defining a File Series

A file series is a series of files which are related to each other. **FIT2D** does not need to know the relationship, but it does require that the files are numbered in a similar manner.

The files must all be of the same format, as only the first file is used to define the input format and all other files are input in the same manner.

A large variety of numbering schemes will be successfully input, but schemes based on alphabetic characters will not be recognised.

A typical file series would be:

```
file_001.dat, file_002.dat, file_003.dat,
file_004.dat, file_005.dat, file_006.dat,
file_007.dat, file_008.dat, file_009.dat,
file_010.dat, file_011.dat, file_012.dat
```

An alternative scheme which is also often found is:

```
file_1.dat, file_2.dat, file_3.dat,
file_4.dat, file_5.dat, file_6.dat,
file_7.dat, file_8.dat, file_9.dat,
file_10.dat, file_11.dat, file_12.dat
```

(The former series is preferable, as the directory listing command will output the files in numerical order, which will not be the case for the second series, but **FIT2D** will input both series correctly.)

FIT2D looks for the changing numerical part and works out whether or not the number of characters used for the numerical part is fixed or changing.

The file series is defined by selecting the first file using the GUI file selection tool. If the file format is known and fully defined the data will be input automatically. Otherwise one or more menus will be presented to define the file format and parameters necessary for the input.

After input the data will be displayed and a **FILE O.K.** form will appear. If the data is not correct, enter **NO** and you can select another file or re-define the format. Enter **CANCEL** to stop the operation completely and return to the main **FILE SERIES** menu.

After confirming the first file input is correct, you are prompted for the last file in the series. This is not input immediately. Again you can use the **CANCEL** button on the file selection form, if you don't what to continue the operation.

Next you are prompted for the **FILE INCREMENT**. (For some operations this value is part of a larger graphical value form.) This is the increment to increase the file "number" for each input. If all files in a simple series are to be input, this number should be 1. However, if only every second file is required, then enter 2. This increment is generally intended to allow skipping of files, however some series are defined in a "confusing" manner. e.g. if the following files exist:

```
file_10.dat, file_20.dat, file_30.dat,
file_40.dat, file_50.dat, file_60.dat,
file_70.dat, file_80.dat, file_90.dat
```

but no files in between. Selecting `file_10.dat` as the first file and `file_90.dat` as the last file will make **FIT2D** think there are 81 files to input. However defining the "FILE INCREMENT" as 10 allows the correct files to be input.

The file series is then defined. If any files are missing in the series, a warning prompt will appear with the name of the missing file, and the **O.K.** button must be clicked before the operation will continue.

The effect of missing files will depend on the operation, but in general the operations will try to continue, and leave blank any affected region, or not include the data in the case of the **AVERAGE** operation.

6.2 Defining a Composite Image

It is often useful to display a whole series of images together. This can be achieved in a very general manner using the **COMPOSITE** command. An example of such a composite image is shown in Figure 22. The files containing the data are defined as explained above. In addition a graphic form allows, a number of options:

SUBTRACT: Allows a background image to be input from file and subtracted from each image input in the series, prior to further operations.

ROI: Allows interactive graphical selection of a small region to be input and used to built up the composite image. If **ROI** is not selected the whole of the input images is used.

INCREMENT: Increment number between input files (see above).

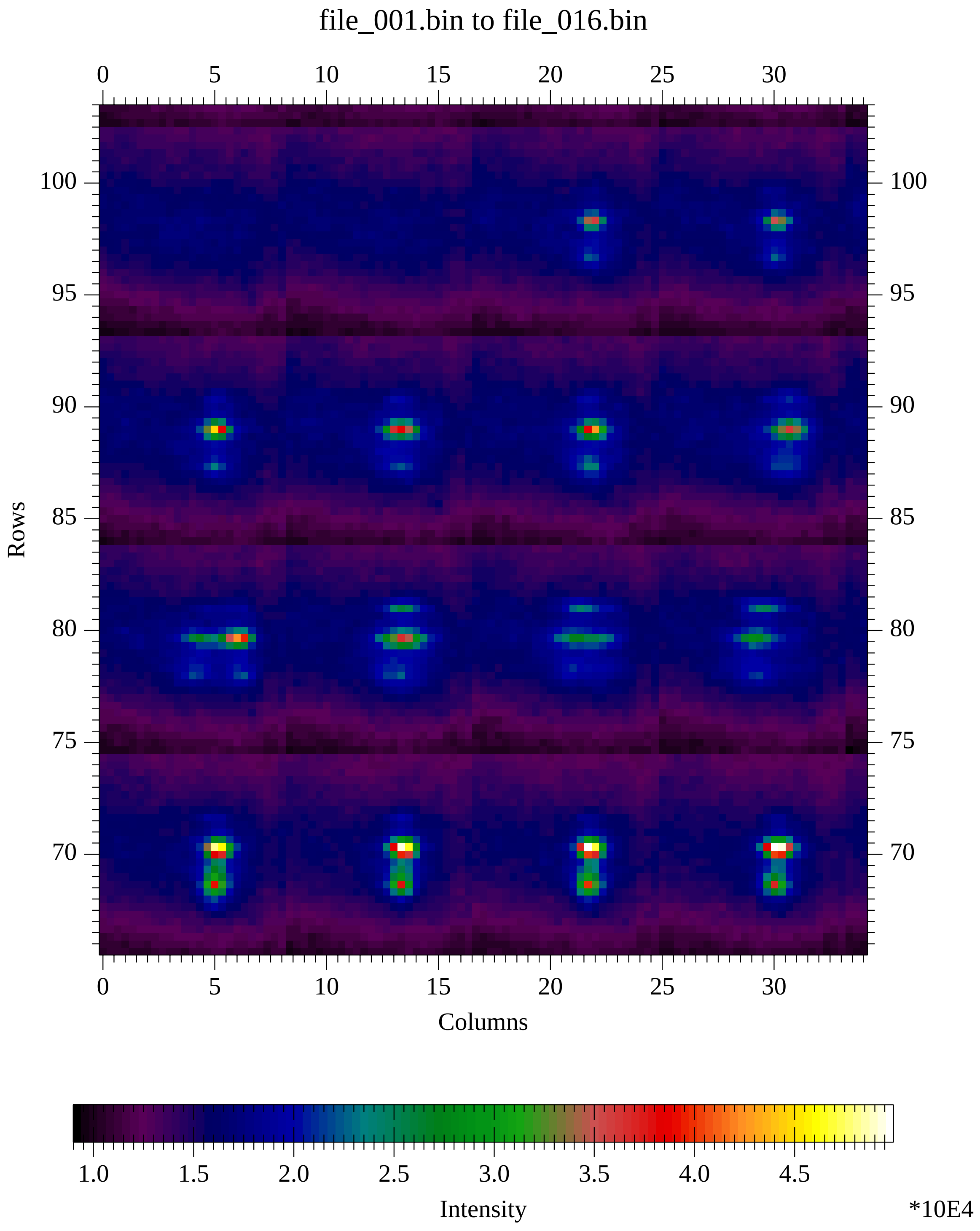
NO. PER ROW: Number of input images to be displayed together on one row of the composite image. The number of rows is calculated automatically from the number of files and the number per row.

RE-BIN NO.: This is the factor by which the input image pixels should be re-binned (in both directions) before being assembled into the composite image. If many images are being input fully it is recommended to re-bin them to reduce the computer memory requirements.

If **SUBTRACT** is **YES** then the file selection tool will help you input an image to be subtracted from each of the file series images. The file format must be the same.

If **ROI** is **YES** the "FIT2D: REGION OF INTEREST MENU" will appear. The **ZOOM IN** button can be used to select a sub-region. Coordinates of the region may be entered graphically,

Figure 22: An Example of a **COMPOSITE** Image



or by clicking the **KEYBOARD** button and entered from the keyboard. The other buttons allow un-zooming, if the original choice was not correct. When satisfied click on the **EXIT** button.

If the size of the data array needed to construct the composite image is larger in either dimension than the current program array sizes, then you will be prompted **DESTROY AND CREATE BIGGER PROGRAM ARRAYS**. If you answer **YES** any current data and "memory" data will be lost, and the **program array sizes will be set exactly to the size necessary for the composite image.** Entering **NO** will abort the operation. Assuming the computer memory is available the operation will start inputting data, otherwise an error message will appear.

When the data has been input, the composite image is displayed as a false colour image. The data is in the main program array, so operations such as **DISPLAY** may be used to change the display style. The **FILE SERIES** menu can also be exited, and the data can be analysed within another interface.

6.3 Integrating a Series of Images

The **INTEGRATE** command is equivalent to the **INTEGRATE** command within the **POWDER DIFFRACTION INTEGRATE** command (See Section 17.18, Page 215).

First you need to know geometrical parameters e.g. beam centre and detector tilt angles. For this you may want to first input data and treat it within the **POWDER DIFFRACTION** interface.

The **FILE SERIES INTEGRATE** command prompts for the first and last file in the series as with the other commands. A detector distortion correction form allows optional correction of non-uniformity of response and spatial distortion. The first image in the series will be input and corrected as specified.

The masking menu appears automatically and allows both masking and selection of a sub-region with the **ZOOM-IN** command. When satisfied click **O.K.**

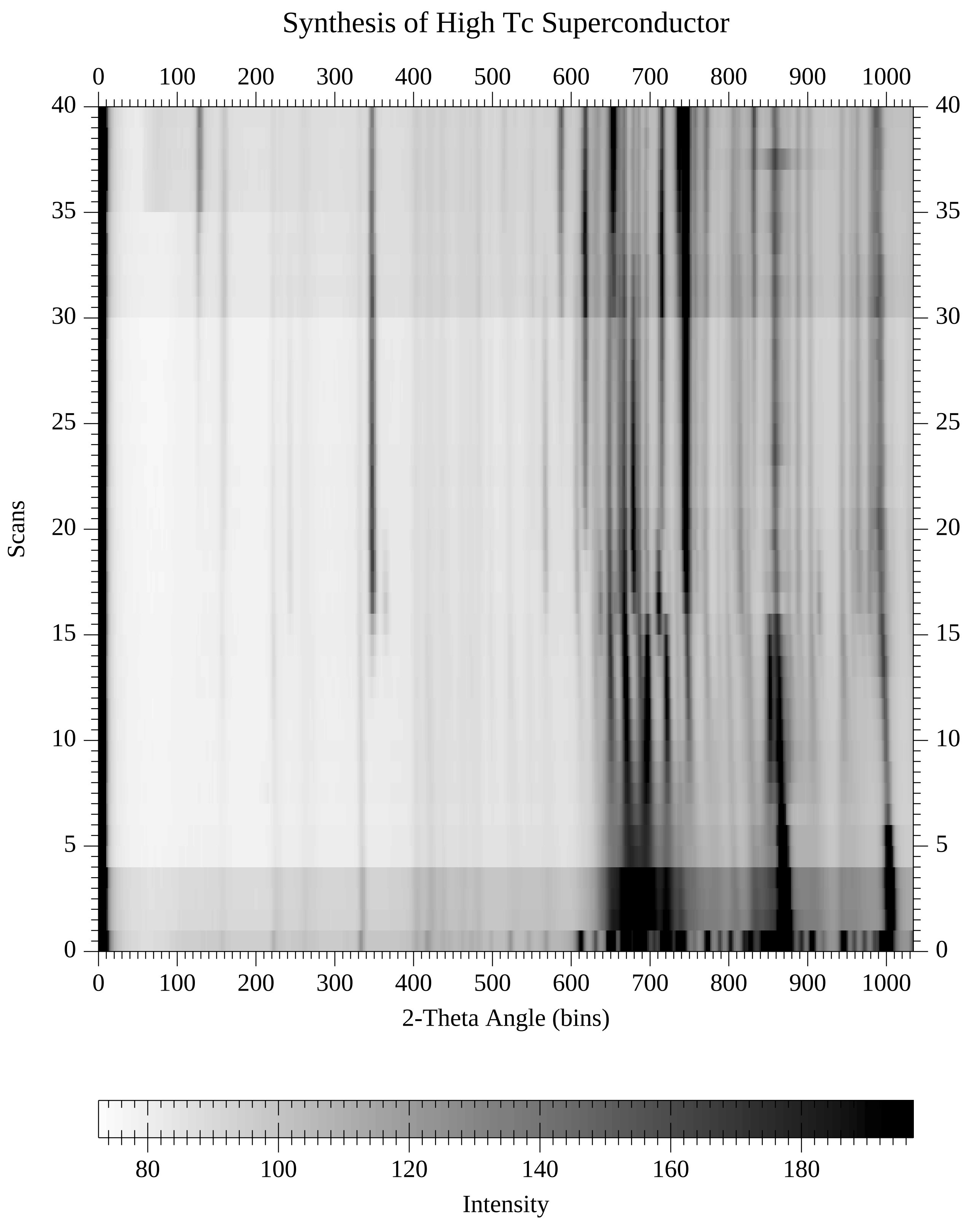
The experiment geometry and the integration control forms are presented for completion. There is the option to save each 1-D integrated scan to an output file as the integration is completed. If this option is selected the output file type and the file extension will be asked. The output files will have the same base name as the input files.

Once the menus and forms have been completed the integration of the first image will proceed, and the same treatment will then be repeated automatically for all subsequent images in the defined file series. Each 1-D integrated scan is displayed as the data is integrated. At the end the whole of the scans are left as a 2-D image in the main data array. The X-direction is the 2θ direction, and the Y-direction has the direction scans, with the first at the bottom. This data can be saved or used for further analysis.

Often there will be many more pixels in the horizontal direction than in the vertical (the scans). The output will be a thin rectangle, and details may not be seen. To see the full detail use the **OPTIONS** menu command **ASPECT RATIO** to turn off the automatically correct aspect ratio. The image will then fill the whole of the current image output area.

An example of the results left in the main program array after the **INTEGRATE** command are shown in Figure 23.

Figure 23: An Example of the Results of the **INTEGRATE** Command



7 The IMAGE PROCESSING (GENERAL) Interface

The **IMAGE PROCESSING (GENERAL)** interface contains general commands for operations which manipulate images, such as adding, dividing, transposing, and filtering. These are the typical commands you will find in any image processing software. The **IMAGE PROCESSING (GENERAL)** main menu is shown in Figure 24.

The following commands are available (in order displayed in the menu from left to right, top to bottom):

EXIT: Exits the **IMAGE PROCESSING (GENERAL)** interface and returns to main "scientific interface" menu. Note: data can be moved from one interface to another.

DISPLAY: Enter the GUI general display menu, assuming that data has been defined. This is used to display the data in the current program array in a variety of different styles.

EXCHANGE: transfers data from the "main program array" to the "memory array", and vice versa. This may be useful to save some data whilst another operation is overwriting the "main program array".

FILTER: starts a sub-menu which allows various filtering and smoothing of the data (see sub-section below).

?: Help text with short explanation of commands.

FULL: Sets the extent of the "region of interest (ROI)" or "active data region (ADR)" to be the whole of the currently defined data. This may be useful after the "ZOOM-IN" command.

GEOMETRIC: starts the Geometric operations sub-menu which allows various transformations to be applied to the data (see sub-section below).

INPUT: starts an interactive file selection tool to allow selection of directories, and of a file for input (see Section 5.1, Page 39).

HELP: Detailed help text on the interface and the available commands.

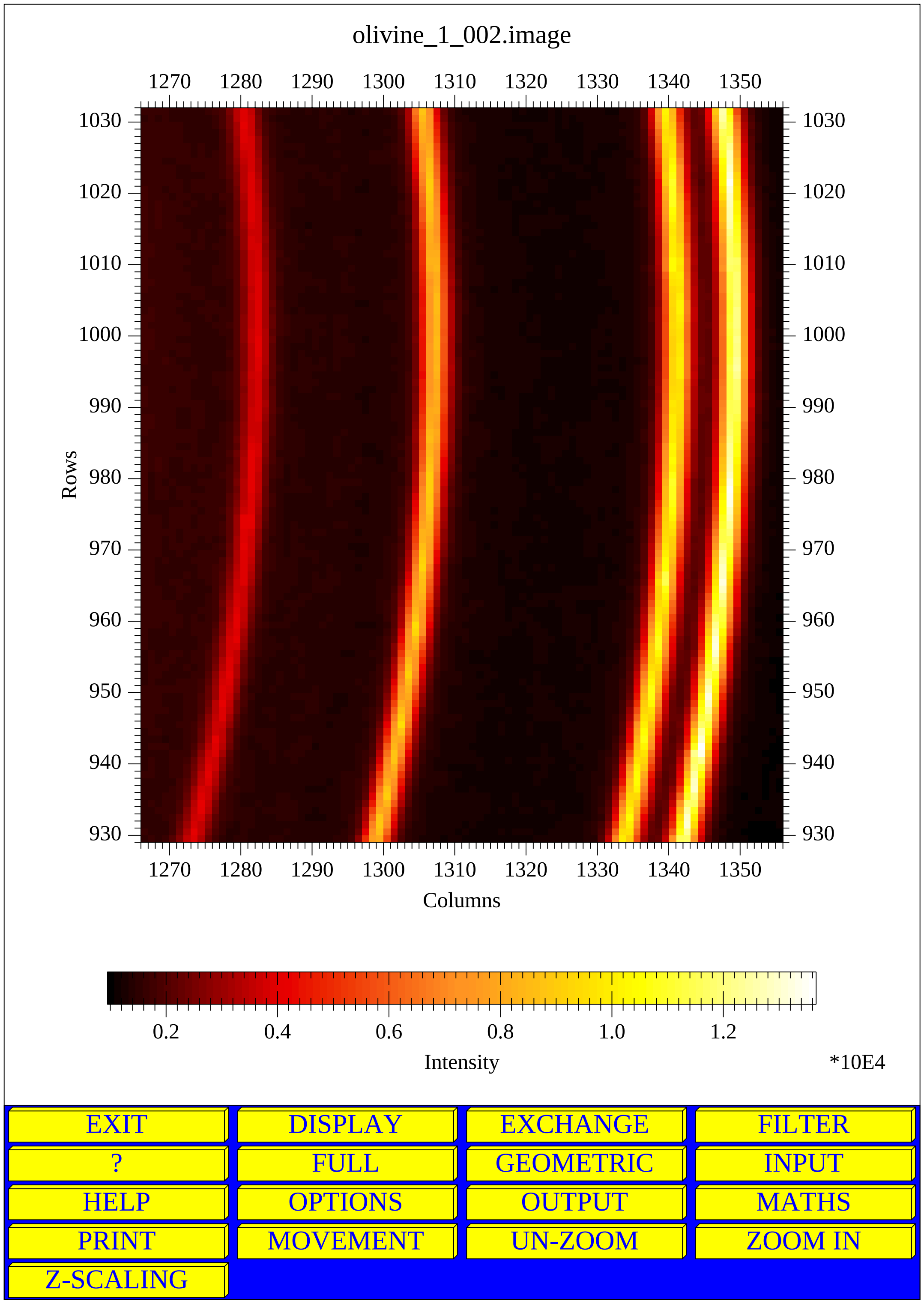
OPTIONS: Starts the general GUI **OPTIONS** sub-menu which allows various display options to be modified, and control of the graphical output style. Used to alter the colour table, etc. after data has been input (see Section 5.5, Page 49).

OUTPUT: General GUI output menu. Saves the current "region of interest (ROI)" in a selected file with a choice of file formats.

MATHS: starts a sub-menu which allows various unitary and binary mathematical operations to be performed pixel by pixel on the data.

PRINT: Save the currently displayed data image as a PostScript file. You are prompted for the name of the output file to create. After the file is fully written, it may be sent to a PostScript printer for printing. Note: It can take some time to create a file of a large 2-D data-set, and often even longer for the file to be printed.

Figure 24: The **IMAGE PROCESSING (GENERAL)** Main Menu



MOVEMENT: starts a sub-menu which allows movement within the currently defined data.

UN-ZOOM: increases the size of the current "region of interest" (ROI).

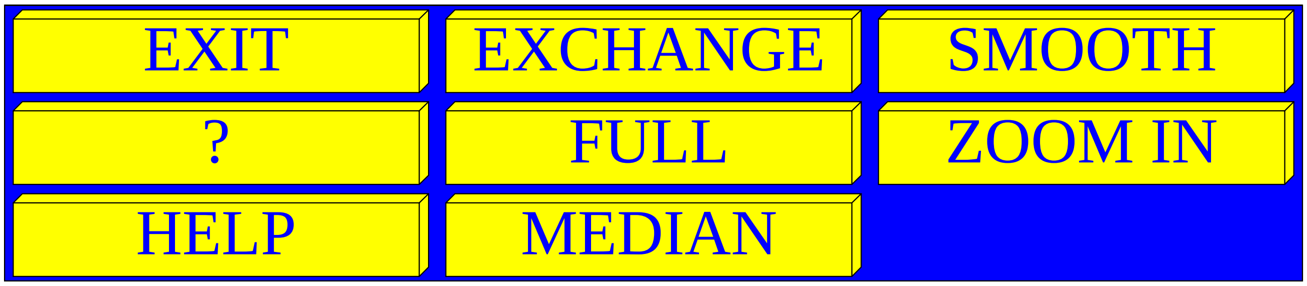
ZOOM-IN: allows a sub-region of the data to be selected and displayed. It should be noted that **all operations including OUTPUT only work on the current selected region**. Care needs to be taken when using **ZOOM-IN** that subsequent operations are intended for only the current sub-region. The full available data region may be selected by using the **FULL** button.

Z-SCALING: starts a sub-menu which allows the false colour/ grey scale used to display image intensities to be changed. Different automatic and fixed scaling methods can be selected.

7.1 The FILTER Sub-Menu

The **FILTER** sub-menu contains commands which allow filtering and smoothing of data. (Other commands exist in the "KEYBOARD MENU" which allow other forms of filtering.) The **FILTER** sub-menu is shown in Figure 25.

Figure 25: The **FILTER** Sub-Menu Commands



The available commands and their functions are:

EXIT: Leave **FILTER** sub-menu, and return to **IMAGE PROCESSING (GENERAL)** main menu.

EXCHANGE: transfers data from the "main program array" to the "memory array", and vice versa. This may be useful to save some data whilst another operation is overwriting the "main program array".

SMOOTH: Apply a top-hat smoothing function of specified size to the current region of interest. Top-hat smoothing is the convolution of the data with a rectangular region of uniform intensity (1.0 divided by the number of pixels in the rectangle). This is a low-pass filter, and will filter out noise and high spatial frequencies. You are prompted for `X BLUR SIZE` and `Y BLUR SIZE`. This defines the size of the top-hat function in the X and the Y directions. The original data is transferred to the memory, whilst the current data array contains the smoothed data after the operations is completed.

?: Help text with short explanation of commands.

FULL: Sets the extent of the "region of interest (ROI)" or "active data region (ADR)" to be the whole of the currently defined data. Note: the filtering commands only affect the current region of interest.

ZOOM-IN: allows a sub-region of the data to be selected and displayed. It should be noted that **all operations only work on the current selected region**.

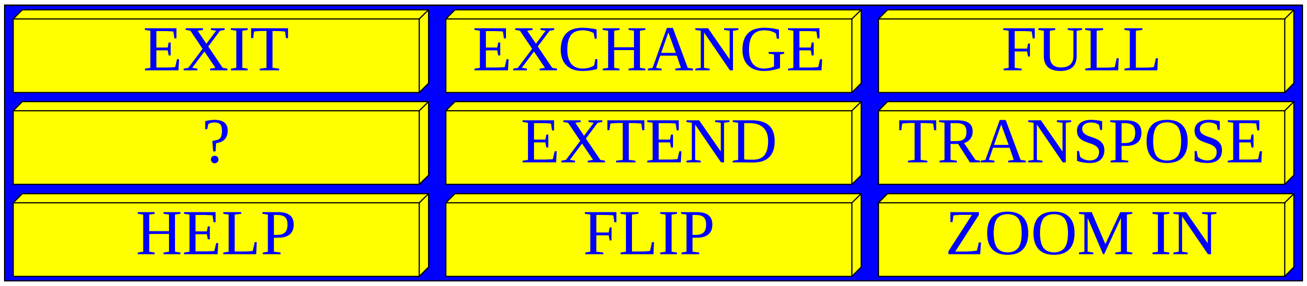
HELP: Detailed help text on the interface and the available commands.

MEDIAN: Applies "median" filter of user specified size. Median filtering is replacing each data value by the median value in a rectangular region centred on the pixel. This is a non-linear filtering which filters out noise whilst tending to preserve edges and other sharp features better than top-hat smoothing. The user is prompted for the `MEDIAN FILTER X-SIZE` and the MEDIAN FILTER Y-SIZE. This defines the number of pixels used to define the rectangle around each pixel for the calculation of the median value. The original data is transferred to the memory, whilst the current data array contains the smoothed data after the operations is completed.

7.2 The GEOMETRIC Sub-Menu

The **GEOMETRIC** sub-menu contains commands which allow geometrical transformations to be applied to the current region of interest. (Other commands exist in the "KEYBOARD MENU" which allow other transformations such as re-binning.) The **GEOMETRIC** sub-menu is shown in Figure 26.

Figure 26: The **GEOMETRIC** Sub-Menu Commands



The available commands and their functions are:

EXIT: Leave **GEOMETRIC** sub-menu, and return to **IMAGE PROCESSING (GENERAL)** main menu.

EXCHANGE: transfers data from the "main program array" to the "memory array", and vice versa. This may be useful to save some data whilst another operation is overwriting the "main program array".

EXTEND: Allows the size of the defined data to be increased. Normally the size is defined when the data is input from file, but it may be useful to make the number of defined pixels artifically larger. This command inputs the new size in the X and Y-directions and extends the image. The new pixels are set to zero. If a variance array exists the pixels are also set to zero.

TRANSPOSE: Transpose elements of the current region of interest. The X and Y-axis values and labels are also swapped. The transposed data is output replaces the original data, which is saved in the "memory".

?: Help text with short explanation of commands.

FLIP: Swaps pixels in the current region of interest, from left to right, and/or from top to bottom, as specified by the user. The operation is performed in-place. Note: If **ZOOM IN** is used to define a sub-region, **FLIP** swaps the pixels, and **FULL** is used, a strange effect will result, since only part of the full region will have been affected.

ZOOM-IN: allows a sub-region of the data to be selected and displayed. It should be noted that **all operations only work on the current selected region**.

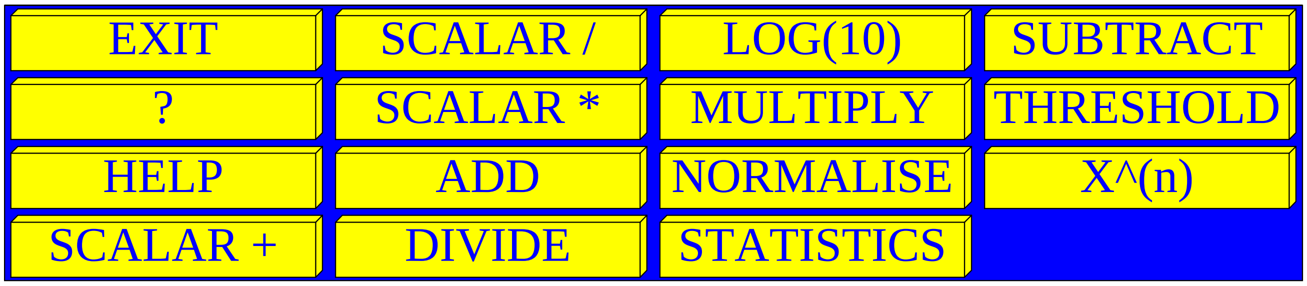
HELP: Detailed help text on the interface and the available commands.

FULL: Sets the extent of the "region of interest (ROI)" or "active data region (ADR)" to be the whole of the currently defined data. Note: the filtering commands only affect the current region of interest.

7.3 The MATHS Sub-Menu

The **MATHS** sub-menu contains commands which allow mathematical operations to be applied to the pixels of the current region of interest. The **MATHS** sub-menu is shown in Figure 27.

Figure 27: The **MATHS** Sub-Menu Commands



The available commands and their functions are:

EXIT: Leave **MATHS** sub-menu, and return to **IMAGE PROCESSING (GENERAL)** main menu.

SCALAR /: Pixel by pixel division of the current region of interest by an input scalar value. The user is prompted `DIVISION CONSTANT` for the scalar which will be used for division.

HELP: Detailed help text on the interface and the available commands. The operation is performed in-place.

LOG(10): Take the logarithm (base-10) of all the pixel values in the current region of interest. If zero or negative values are encountered the user is prompted `LOWER THRESHOLD`, and the value entered will be used as the lowest value in the output. The operation is performed in-place. (Note: This changes the data values, unlike the log scaling option in the **Z-SCALING** menu, which only affects the display.)

SUBTRACT: Subtracts pixel by pixel, the image in the "memory" from the image in the current data array, within the current region of interest. The "memory" data must be defined throughout the current region of interest or a warning message will be produced. The operation is in-place.

?: Help text with short explanation of commands.

SCALAR *: Pixel by pixel multiplication of the current region of interest by an input scalar value. The user is prompted `MULTIPLICATION CONSTANT` for the scalar which will be used for the multiplication.

MULTIPLY: Multiples pixel by pixel, the image in the "memory" and the image in the current data array, within the current region of interest. The "memory" data must be defined throughout the current region of interest or a warning message will be produced. The operation is in-place. The original data is replaced by the result of the multiplication.

THRESHOLD: Allows minimum and maximum thresholding values to be set for the current region of interest. The user is prompted `MINIMUM THRESHOLD VALUE` and **MAXIMUM THRESHOLD VALUE**. Values outside of the set range are replaced will the minimum or the maximum value. The original data is replaced. Note: This operation is in general non-reversible.

HELP: Detailed help text on the interface and the available commands. The operation is performed in-place.

ADD: Add pixel by pixel, the image in the "memory" and the image in the current data array, within the current region of interest. The "memory" data must be defined throughout the current region of interest or a warning message will be produced. The operation is in-place. The original data is replaced by the result of the addition.

NORMALISE: Divide pixel by pixel by the largest value, throughout the current region of interest. The original data is replaced by the result of the division.

X(̂n): Calculate pixel by pixel x^n where x is a user specified constant, and n is the value of each input pixel. The user is prompted ENTER POWER for the value of x. If x is 10.0 this operation is the partial inverse of the **LOG(10)** operation. It is only a partial inverse in the case that values have been thresholded in the **LOG(10)** operation.

SCALAR +: Pixel by pixel addition of the current region of interest with an input scalar value. The user is prompted `ADDITION CONSTANT` for the scalar which will be used for the addition.

DIVIDE: Divides pixel by pixel, the image in the current data array by the image in the "memory", within the current region of interest. The "memory" data must be defined throughout the current region of interest or a warning message will be produced. The operation is in-place.

STATISTICS:

Allows the user to define an arbitrary polygon region with graphical coordinate input, and calculates a number of statistics in the region. If only two coordinates are input, they define opposite corners of a rectangular region for the calculation. The result is displayed in a text window, and can be saved to an output file.

A typical result of the **STATISTICS** operation is shown below.

```
Extremes of polygon (X/Y min, X/Y max) =
    704.85    993.16    761.94   1136.49
Number of pixels inside polygon =     3237
Smallest value =   1.0841     Largest value =   1.4180
Total intensity =   3688.0     (square root =   60.729    )
Average intensity =   1.1393
Root Mean Square (RMS) value =   1.1416
Standard Deviation =  7.15076E-02 Gain (?) =  4.48806E-03
Integrated intensity minus previous average =   61.607
Square root of above =   7.8490
```

The `Integrated intensity minus previous average` contains the integrated sum of pixel values minus the previous average value. Thus, if first a background region is defined, and then a peak is defined a simple form of peak integration is performed.

7.4 Example of Dividing One Image by Another

As an example of a typical binary operation on images, the division of *image1.dat* by *image2.dat* will be used. (Exactly the same logic applies to other binary operation on images.)

The following steps can be used:

1. Enter the **IMAGE PROCESSING (GENERAL)** Interface.
2. Use the **INPUT** command to input data from *image2.dat*
3. Use the **EXCHANGE** command to place the data in the "memory".
4. Use the **INPUT** command to input data from *image1.dat*
5. Use the **MATHS** command to enter the **MATHS** sub-menu.
6. Use the **DIVIDE** command to divide the current data by the "memory" data.

The result is left in the current data array, and the *image2.dat* data is left in the memory.

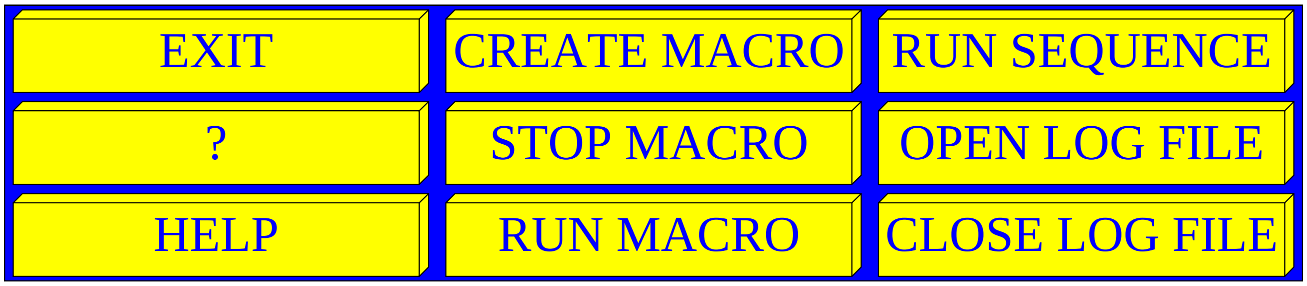
Note: The images can be input in the reverse order, but an additional **EXCHANGE** command will be necessary, so that the division is in the correct sense.

The same logic applies to the subtraction of two images. With the image addition and multiplication the order is not important.

8 The MACROS / LOG FILE Interface

The **MACROS / LOG FILE Interface** is a support interface which allows creation and running of macros, including running simple macros on whole sequences of files, and the creation of log files of the **FIT2D** output and user input. The main menu is shown in Figure 28.

Figure 28: The **MACROS / LOG FILE** Menu Commands



EXIT: Leave the **MACROS / LOG FILE** sub-menu, and return to the interfaces menu.

CREATE MACRO: Open a user specified file and start saving a macro of the current series of operations. The user is prompted with the name of a file in which to save the macro (`fit2d.mac` by default). Enter **YES** to use the suggested file, or **NO** to select another file using the file selection tool. See Section 8.1, Page 75 for full details on creating and using simple macros.

RUN SEQUENCE: Run a user specified macro on a series of files. At present this only allows for relatively simple macros i.e. macros which work on zero or one input file, and zero or one output file. This covers most real cases, but more complicated macros can be run using the **SEQUENCE** command in the “KEYBOARD” Interface. The user selects the macro file to run using the file selection tool, and the first and last files in the sequence. The user is then prompted `FILE INCREMENT` which is the step between inputting files in the sequence. Normally this will be 1, but to skip every other file, the value 2 would be entered. The `OUTPUT FILES EXTENSION` is then defined. If the macro includes input and output files, the output files will be created in the same directory and with the same file names as the input files, but a different file extension will be used. Here the file extension for the output files is defined. If no output files are defined then this value can be ignored.

?: Help text with short explanation of commands.

STOP MACRO: Closes a previously opened macro file; stops the recording of a sequence of operations. See Section 8.1, Page 75 for full details on creating and using simple macros.

OPEN LOG: Open a user specified file and start saving a macro of the current series of operations. The user is prompted with the name of a file in which to save the macro (`fit2d.log` by default). Enter **YES** to use the suggested file, or **NO** to select another file using the file selection tool.

HELP: Detailed help text on the interface and the available commands.

RUN MACRO: Run a macro once. The file selection tool is used to select the macro file to run. Whilst it must be done manually, a single macro file can be set up to run on a whole sequence of input and/or output files.

CLOSE LOG FILE: Close a previously opened log file; stop outputting a record of the **FIT2D** output and user input to a file.

8.1 Creation and Using Simple Macro Files

Often it is necessary to process a large number of files using the same sequence of data analysis operations. This task is made much easier through "macro" files and the **RUN SEQUENCE** command. Here macros are designed to do the simple job of inputting data from one file, and outputting it in another (although the output file can be missing).

To create the macro file perform the following steps:

1. Enter the **MACROS / LOG FILE** interface
2. Select the **CREATE MACRO** command, and define the name of the file to contain the macro.
3. **EXIT** the **MACROS / LOG FILE** interface and perform the full sequence of input, analysis, and output commands on one of the files in the sequence. Be careful to make sure all values which may change or remember defaults are set. (Normally, the user can rely on most default values, but for the macro to be really robust, it is better to spend the time setting the values, so the macro contains all the necessary values, and there is no reliance on default values which may be wrong.
4. **EXIT** whichever interface you have been using and re-enter the **MACROS / LOG FILE** interface.
5. Select the **STOP MACRO** command to finish recording the sequence.
6. Edit the file with you favourite editor (you can use <CONTROL>⊗Z to use the same terminal window as **FIT2D**, and then `fg` to return to **FIT2D**, or work from another terminal window). Replace the input file name with the "symbol" `#IN` and any output file name with `#OUT`.

The macro is now ready to be run.

The macro can be run repeatedly on a series of files by using the **RUN SEQUENCE** command. See Section 6.1, Page 61 for details of allowable file series. The macro file to run is selected, as are the first and last files in the series. The "increment" between input file is input. Files in a series must be named with a changing numerical part. This is automatically recognised and the first and last numbers in the series defined. To process every file between these numbers (assuming the files exist) enter 1 (or -1 in the case that the last file has a larger number than the first). To skip alternate files enter 2 (or -2), and so on. If for some reason all the files have an extra 0 after the changing numerical part e.g. they are numbered `file_10.dat, file_20.dat, file_30.dat`, etc. enter 10.

The **RUN SEQUENCE** command will automatically generate the input file names from this information, and they will replace the `#IN` "symbol" which has been placed within the macro.

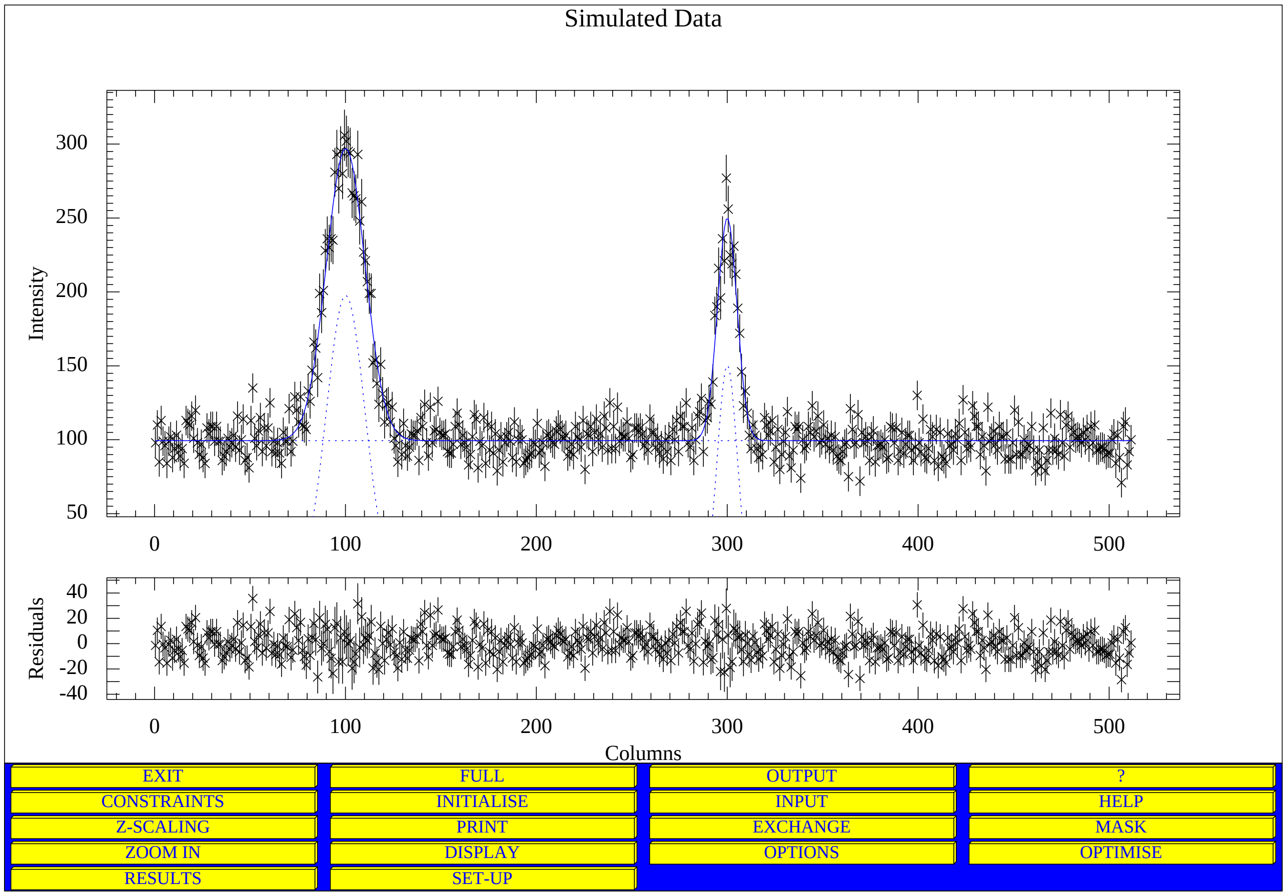
If the macro defines an output file using the `#OUT` "symbol", then the command will also generate the series of output file names. These are the same as the input file names, but with a user specified extension replacing any extension of the input files. The last user input before the macro is automatically run is `OUTPUT FILE EXTENSION` which is the output file extension used to form the output file names. The generated output file names automatically replace the `#OUT` "symbol" each time the macro is run in the sequence.

The progress of the macro is output in the terminal window.

9 The MFIT (MULTIPLE 1-D FITTING) Interface

Both 1-D and 2-D models may be fitted to experimental data using a variety of minimisation methods. 1-D model fitting is available within the **MFIT (MULTIPLE 1-D FITTING)** interface. This interface contains the equivalent of the **MFIT** program [7]. (2-D fitting is available in the “KEYBOARD” interface, within the **FIT** sub-menu.) An example of the graphics output and the main menu is shown in Figure 29, Page 77.

Figure 29: The **MFIT (MULTIPLE 1-D FITTING)** Main Menu



The main menu contains the following commands:

EXIT: Exits **MFIT (MULTIPLE 1-D FITTING)** interface and returns to main “scientific interface” menu. Note: data can be moved from one interface to another.

FULL: Sets the extent of the “region of interest (ROI)” or “active data region (ADR)” to be the whole of the currently defined data. Note: the filtering commands only affect the current region of interest.

OUTPUT: General GUI output menu. Saves the current ”region of interest (ROI)” in a selected file with a choice of file formats.

?: Help text with short explanation of commands.

CONSTRAINTS: Allows individual fitting parameters to be constrained to fixed values or left free to evolve.

INITIALISE: Interactive graphical definition of a fit model. With a choice of fitting functions. (See Section 9.1, Page 78.)

INPUT: starts an interactive file selection tool to allow selection of directories, and of a file for input (see Section 5.1, Page 39).

HELP: Detailed help text on the interface and the available commands.

Z-SCALING: starts a sub-menu which allows the false colour/ grey scale used to display image intensities to be changed. Different automatic and fixed scaling methods can be selected.

PRINT: Save the currently displayed data image as a PostScript file. You are prompted for the name of the output file to create. After the file is fully written, it may be sent to a PostScript printer for printing.

EXCHANGE: transfers data from the "main program array" to the "memory array", and vice versa. This may be useful to save some data whilst another operation is overwriting the "main program array".

MASK: Interactive graphical definition of data regions to be masked off (or un-masked). Masked pixels and elements are ignored in the fitting.

ZOOM-IN: allows a sub-region of the data to be selected and displayed. It should be noted that **all operations only work on the current selected region**.

DISPLAY: Enter the GUI general display menu, assuming that data has been defined.

OPTIONS: Starts the general GUI **OPTIONS** sub-menu which allows various display options to be modified, and control of the graphical output style. Used to alter the colour table, etc. after data has been input.

OPTIMISE: Tries to optimise the fit model to the data values. This button may be pressed more than once.

RESULTS: Display of the fitted model parameters. This is displayed in a text window and may be saved to an output file.

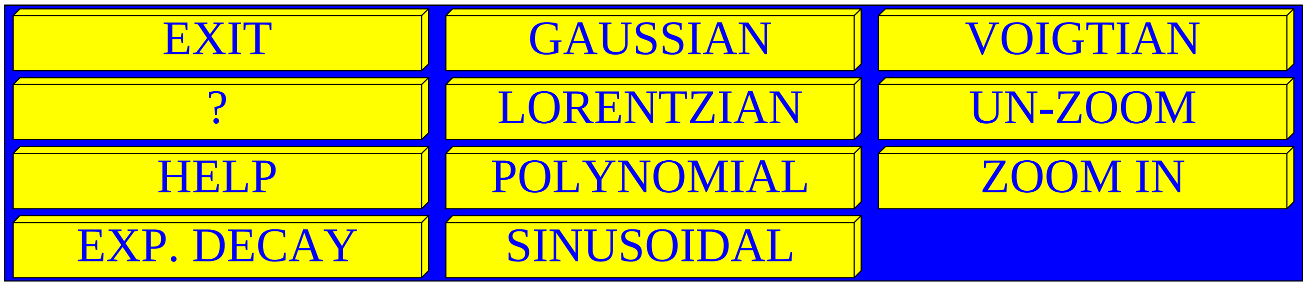
SET-UP: Graphical form allowing setting of control parameters.

TRANSPOSE: Transpose data so that columns before the rows, and vive versa. The fitting works on the rows, so the **TRANSPOSE** command is necessary to fit the column wise data.

9.1 Entering an Initial Model and Model Optimisation

Fit models may be defined using the **INITIALISE** sub-menu. A fit model may be constructed from an arbitrary number of peak functions (Gaussian, Lorentzian, and Voigt) and

Figure 30: The **INITIALISE** Sub-Menu



trigonometric or exponential decay functions, and a polynomial function of specified order. The **INITIALISE** sub-menu is shown in Figure 30, Page 79.

A model is built up by successfully selecting a function type to add to the model. e.g. To add a Gaussian peak function to the model:

1. Select the **GAUSSIAN** button.
2. The message **CLICK ON PEAK TOP** appears, and the graphics cursor is active. You should click reasonable accurately on the estimated top of the peak. This position is used for the initial peak position, and as part of the initial peak height calculation.
3. The message **CLICK ON PEAK AT HALF HEIGHT** appears, and the graphics cursor is active. You should click reasonable accurately on the estimated side of the peak at half height. Either side will do. This position together with the peak top position is used to calculate the initial standard deviation width of the peak, and it's maximum height.

As functions are added they are drawn on the diagram in blue.

For a general background function, the **POLYNOMIAL** button allows an user specified polynomial to be added to the model. The user is prompted for the order of the polynomial to add, and the order of the initial polynomial function to be fitted. The initial polynomial is automatically fitted to the residuals between the data and the currently input function values. Therefore, generally it is best to add any polynomial background function to the model last.

The **SET-UP** form allows the control parameters to be varied e.g. Weighted fitting or not (provided variance arrays exist). The **CONSTRAINTS** form allows individual model parameters to be constrained, or set to freely vary.

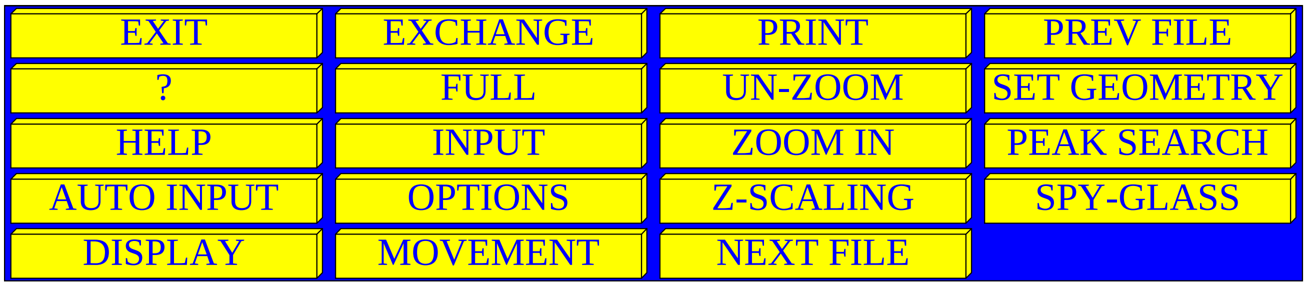
The **OPTIMISE** button optimises the model to the data, according to the current control parameters. The fitted data is output in the "memory". If the current region of interest contains more than one row, the subsequent rows of data will also be fitted using the fitted parameters of the previous fit, as their initial parameters.

The display of the fitted model and the data is shown in the upper X/Y graph in Figure 29, Page 77, and beneath is a display of the residuals. If error estimates are available for the data values, these too will be displayed. This graphical display of the results may be saved as a PostScript file using the **PRINT** button (while this is displayed in the graphics window).

10 The ON-LINE CRYSTALLOGRAPHY Interface

As well as an “off-line” data analysis program, **FIT2D** may be used as an “on-line” data display and manipulation program. Figure 31, page 80 shows the **ON-LINE CRYSTALLOGRAPHY** interface. Data may be input and displayed as with the other menus, and additionally the **AUTO INPUT** button allows files in a sequence to be automatically input and displayed as they are saved by the data acquisition system. An optional “alarm” may be set which will ring if the time for an image to appear is greater than a specified length.

Figure 31: The **ON-LINE CRYSTALLOGRAPHY** Main Menu



EXIT: Exit Interface and return to main interfaces menu

EXCHANGE: Swap current data with the “memory”; see Section 5.2, Page 41.

PRINT: Output current graphics to a PostScript file.

PREV FILE: Input “previous” file in a file sequence.

?: List of commands with short descriptions.

FULL: View image of full data; set the ROI to be the whole of the defined data.

UN-ZOOM: Zoom out to see more of the data. (Also changes the ADR.)

SET GEOMETRY: Define diffraction geometry i.e. beam centre, wavelength, sample to detector distance. This must be defined if the corresponding d-spacings of diffraction peaks are to be calculated.

HELP: Detailed help text.

INPUT: Input data from a file on disk. This defines the file type, and may be the size for further input using the ”NEXT FILE” or ”PREV FILE” commands; see Section 5.1, Page 39.

ZOOM IN: View a smaller region in greater detail. (Also changes the ADR.)

PEAK SEARCH: Find, display, and calculate statistics on diffraction peaks. A number of control parameters allow the peak search to be optimised to find the maximum number of genuine peaks, whilst rejecting noise and other defects. The found peaks and estimates of integrated intensities and sigmas may optionally be saved in an ASCII file.

AUTO INPUT: Automatically input files from a sequence with option of performing a peak search on input, and of setting an alarm if a new image is not available after a defined interval.

OPTIONS: Display style control menu; see Section 5.5, Page 49.

Z-SCALING: Automatic or fixed user control of intensity display range. It is important to consider whether fixed range scaling, or automatic re-scaling is more suitable when displaying a sequence of images; see Section 5.3, Page 43.

SPY-GLASS: Cursor controlled "real-time" zoom window.

DISPLAY: Graphical display possibilities; see Section 5.4, Page 45.

MOVEMENT: Easily controlled movement around an image; see Section 5.6, Page 54.

NEXT FILE: Input "next" file in a file sequence.

MASK: Interactive masking of "bad" data regions. These are ignored in the **PEAK SEARCH** command; see Section 5.7, Page 56.

The peak search option identifies peaks, uses centroiding (with background removal) to improve the initial peak positions, calculates D-spacing, and performs a simple box integration. This allows estimates of the number peaks with different $I/Sigma(I)$ to be displayed, and if the diffraction geometry has been defined, the number of peaks in different resolution shells, and the average $I/Sigma(I)$ in each shell is displayed. The peak positions, D-spacings and intensities may be saved to an ASCII output file for further use e.g. Input to an indexing program.

The **AUTO INPUT** command allows a sequence of images to be input, displayed, and optionally searched for peaks, and statistics output. Similarly the **NEXT FILE** and **PREV FILE** allow easy input of following or previous files in a sequence. (All files are assumed to be of the same format and sizes as the a file input using the **INPUT** button.)

11 The POWDER DIFFRACTION (2-D) Interface

The **POWDER DIFFRACTION (2-D)** interface allows Debye-Scherrer rings which have been recorded of a 2-D detector to be integrated to the equivalent of a 2-theta scan with a zero-dimensional detector [10]. An example of an input image and the main menu are shown in Figure 32, Page 83. To aid with this the **MASK** sub-menu allows interactive definition of masked areas i.e. pixels which will be ignored during further operations. The **TILT** button allows the beam centre and any non-orthogonality of the detector to the main beam to be refined, and this tilt is used by the **INTEGRATE** command. Other types of output are also available. The **CAKE** sub-menu allows different regions of the image to be defined with graphical coordinate input for the starting azimuth, end azimuth and inner and outer radii of the integrated region. The number of output bins in both the azimuthal and the 2-theta or radial direction are user specified. This allows a variety of different output possibilities e.g. A number of different 2-theta scans, for different azimuth ranges; a 1-D profile of intensity of a ring as a function of azimuth (e.g. for texture studies); or a polar transform of the data. The output formats include one of the GSAS formats [15], one of the Cerius formats[10], and a general purpose ASCII two column format, which may easily be edited.

The **POWDER DIFFRACTION (2-D)** Main Menu contains the following commands:

EXIT: Exit **POWDER DIFFRACTION (2-D)** menu and return to main interfaces menu

BEAM CENTRE: Determine beam centre by a choice of methods

FULL: View image of full data

OUTPUT: Save data in an output file

?: List of available commands with short explanations

CAKE: Versatile multiple 2-theta scans integration

INPUT: Input data from a file on disk

TILT: Fit tilt and beam centre to powder rings

HELP: Help text on this graphical menu

CORRECTION: Spatial distortion correction

INTEGRATE: 2-D to 1-D 2-theta integration

Z-SCALING: Automatic or user control of intensity display range (see Section 5.3, Page 43)

PRINT: Output current graphics to PostScript file

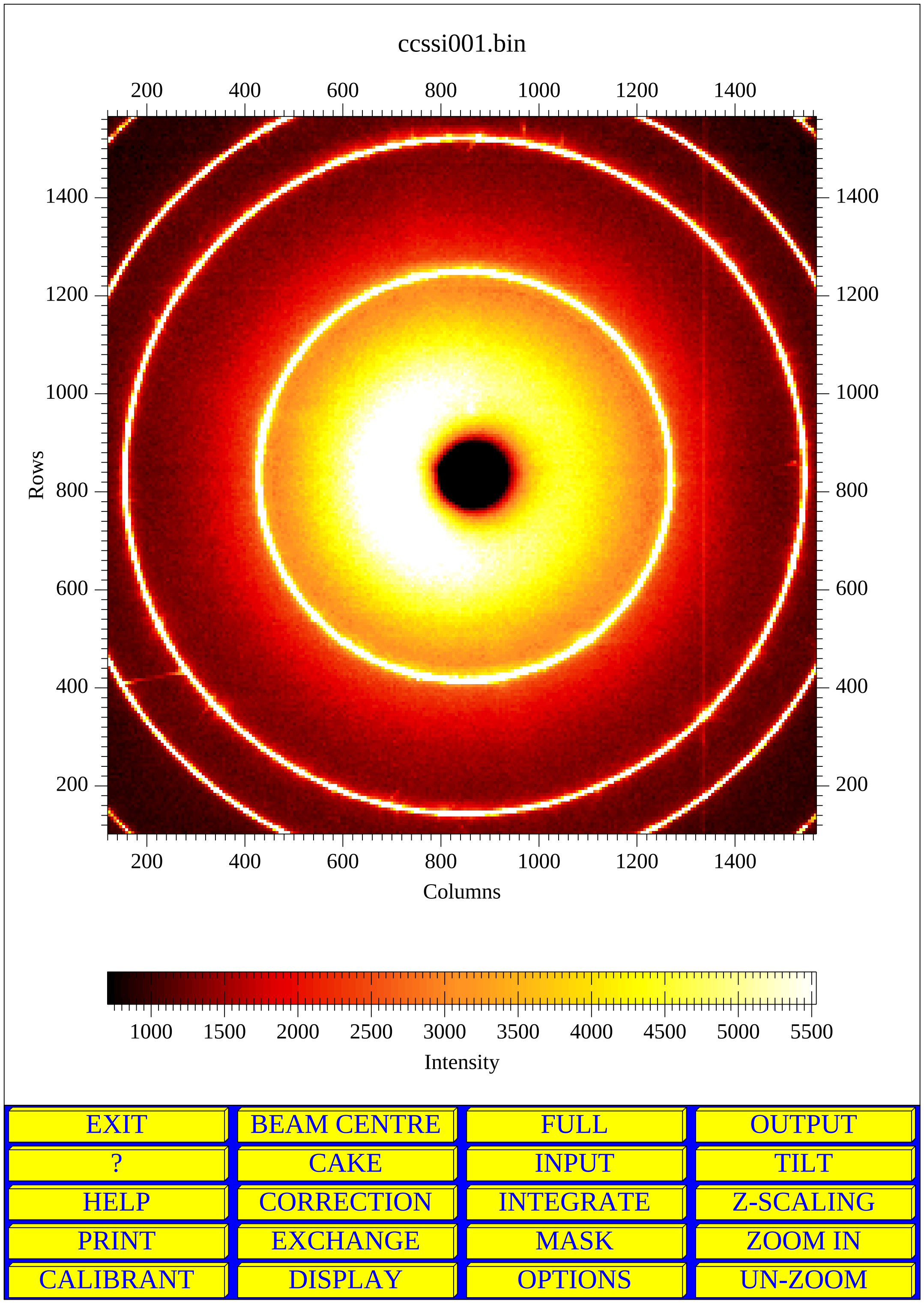
EXCHANGE: Swap current data with the "memory"

MASK: Defined masked-off regions of the image

ZOOM IN: Define smaller graphical display region

[10]Crystallographic software package from Molecular Simulations

Figure 32: The **POWDER DIFFRACTION (2-D)** Main Menu



CALIBRANT: Refine wavelength, distance, tilt, etc from powder pattern with known D-spacings.

DISPLAY: Further graphical display possibilities (see Section 5.4, Page 45)

OPTIONS: Graphics display control menu

UN-ZOOM: Zoom out to see more of the data

11.1 The INPUT Command

The **POWDER DIFFRACTION (2-D)** Interface **INPUT** command is different from other **INPUT** commands in that it has an additional graphical form to control correction of spatial distortion and non-uniformity of intensity response on input.

The initial selection of the input file and format is the same as the other **INPUT** commands. See Section 5.1, Page 39 for further information.

When the raw data has been input a graphical form similar to the one shown in Figure 33 is displayed.

The form allows the following options:

FLAT-FIELD: Controls the application of a flat-field correction to the data immediately on input. If **YES** is selected the input data will be divided pixel be pixel by the data input from the file defined by **FF FILE**.

FF FILE: This uses the file selection tool to input a file to be used for flat-field correction.

FF SCALE: If the flat-field file data has not been normalised you may want multiply the data by a scale factor. In this case, this should be set to **YES**. The multiplier is set with **FF MULTIPLIER**.

FF MULTIPLIER: The scale factor to multiply the result of any flat-fielded data.

SPATIAL DIS.: Control correction for spatial distortion or not. If **YES** then the file defined by **SD FILE** is used to define the distortion function.

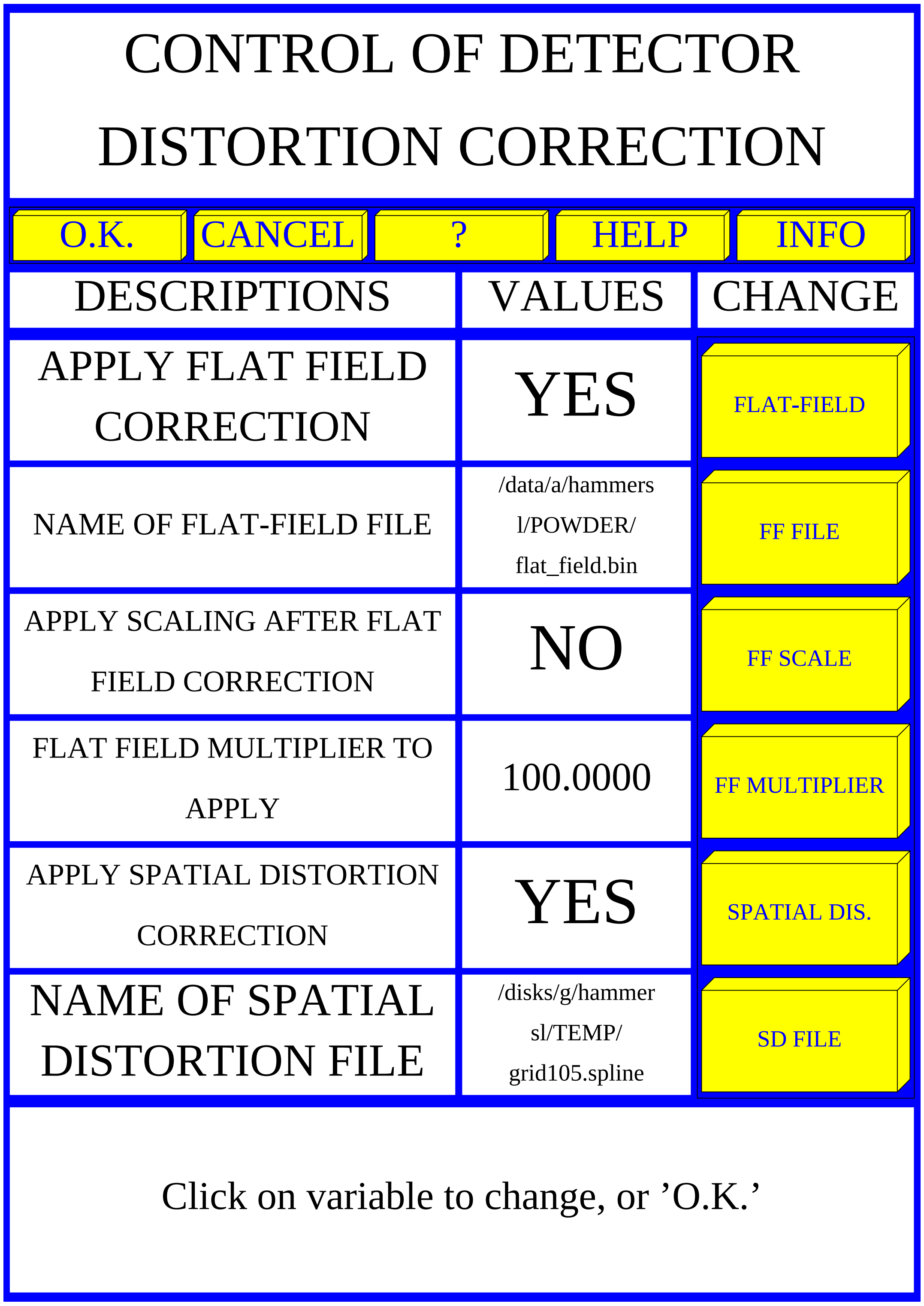
SD FILE: This uses the file selection tool to input a file to be used for the spatial distortion correction.

As any spatial distortion correction is applied, a progress report is output in the terminal window.

11.2 The TILT Command

The **TILT** command allows any non-orthogonality of the detector to the direct beam (tilt), and the direct beam centre on the detector to be determined [10]. This is achieved by least squares fitting of powder rings (or rings from other randomly orientated samples e.g. wax).

Figure 33: The Distortion Correction Control Form



CONTROL OF DETECTOR DISTORTION CORRECTION

DESCRIPTIONS	VALUES	CHANGE
APPLY FLAT FIELD CORRECTION	YES	FLAT-FIELD
NAME OF FLAT-FIELD FILE	/data/a/hammersl/POWDER/flat_field.bin	FF FILE
APPLY SCALING AFTER FLAT FIELD CORRECTION	NO	FF SCALE
FLAT FIELD MULTIPLIER TO APPLY	100.0000	FF MULTIPLIER
APPLY SPATIAL DISTORTION CORRECTION	YES	SPATIAL DIS.
NAME OF SPATIAL DISTORTION FILE	/disks/g/hammersl/TEMP/grid105.spline	SD FILE

Click on variable to change, or 'O.K.'

One or more powder rings are used to refine beam centre and non-orthogonality parameters. This form allows a number of choices in the manner in which this works.

This operation works on the current selected data region, and does not take into account "masked-off" data. You may want to use the **MASK** command to mask-off data prior to using the **TILT** command. Or, if problems are encountered during the **TILT** evaluation, problem data regions may be identified and masked-out prior to another try with **TILT**.

An initial approximate beam centre is obtained from a choice of methods. This beam centre is then used to define a search region around one powder ring. This ring should be strong and well defined. The whole of the ring should be within the search region through the defined azimuthal range.

Other rings may also be selected to be used to refine the beam centre and tilt parameters. Ideally two or more rings should be used, but one will work. Complete rings at high angle will have the largest effect.

From the starting ring coordinates a best fit circle is found, followed at an ellipse. The powder ring section centres are re-calculated for all the rings. The beam centre and tilt angles are refined to these positions. There is the option of rejecting badly fitting positions and re-fitting the data.

For a given set of ellipses, two possible beam centre / tilt angle combinations are theoretically possible and indistinghable. Only if the beam centre and/ or the tilt angles are fixed is there an unique solution. If the beam centre and the tilt are being fitted you will be asked to choose between the two solutions. (In practice owing to the uncertainty in the data the same solution may be produced for both minimisations.) If there are two solutions you may be able to choose the correct solution by other knowledge from the data e.g. scatter from the beam-stop. If you can't choose between the solutions, do not worry; the calculated two-theta angles from the data will be the same.

First the **Experimental Geometry Control Form** is presented. An example is shown in Figure 34.

This form is not too important, as many of the values will be determined automatically by the command. However, the pixel size should be correct if strange results are to be avoided, and **DISTANCE** does affect slightly the interpretation of the powder rings. Other values, such as the wavelength, are not important for the **TILT** command, but can be usefully set for later operations.

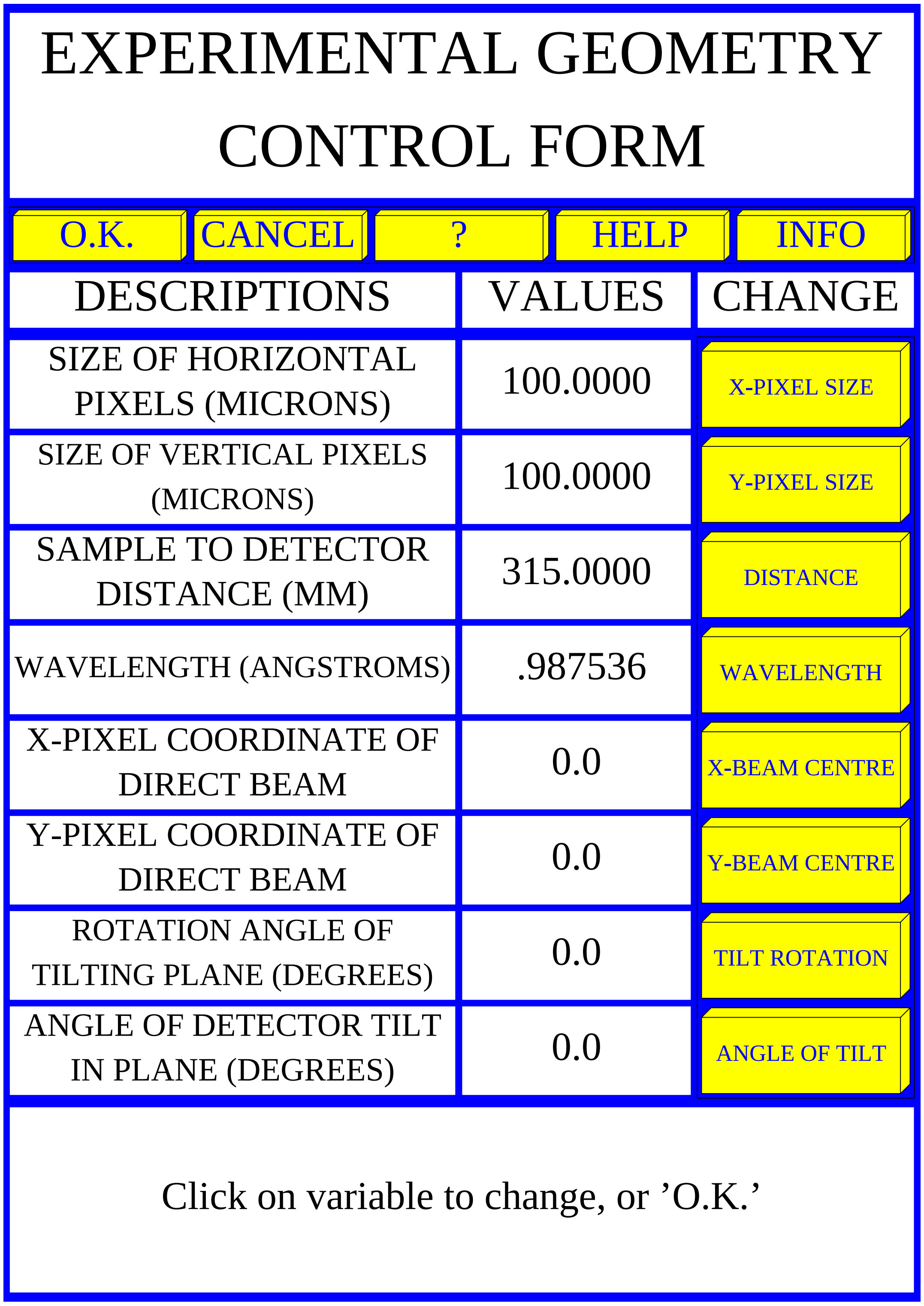
When the correct values have been set, click on **O.K.**.

The **TILT/ BEAM CENTRE REFINEMENT** Control form now appears. This is shown in Figure 35.

The following "buttons" are available:

ANGULAR SECTIONS: This the number of angular sections around 360 degrees of the powder rings which are used to calculate the centre of the rings. The beam centre and tilt are fitted to these positions on each ring. If the data is very noisy or shows poor powder averaging then a smaller number may be better. (No theoretical criterion exists to set an

Figure 34: The Experimental Geometry Control Form



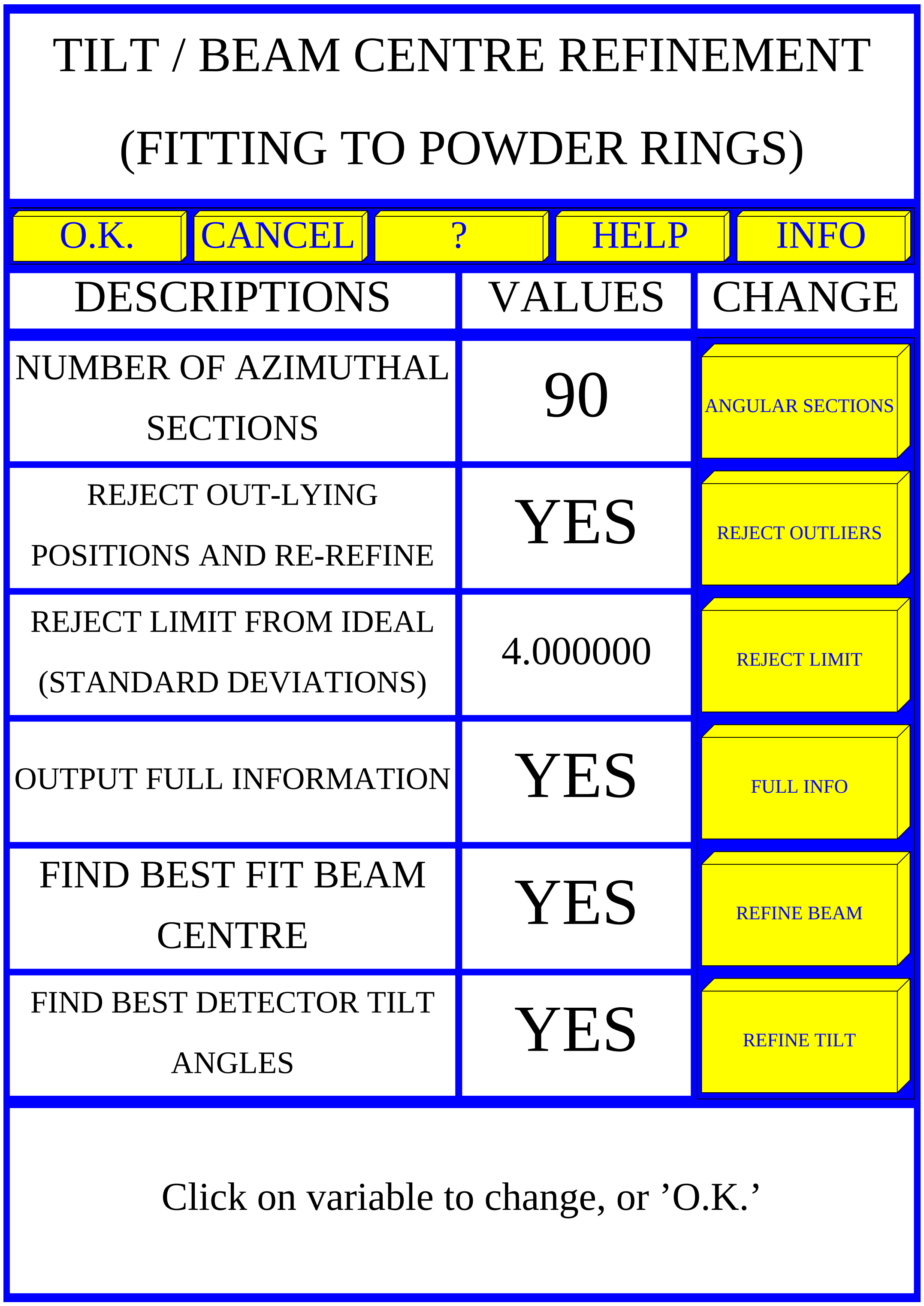
EXPERIMENTAL GEOMETRY CONTROL FORM

O.K. | CANCEL | ? | HELP | INFO

DESCRIPTIONS	VALUES	CHANGE
SIZE OF HORIZONTAL PIXELS (MICRONS)	100.0000	X-PIXEL SIZE
SIZE OF VERTICAL PIXELS (MICRONS)	100.0000	Y-PIXEL SIZE
SAMPLE TO DETECTOR DISTANCE (MM)	315.0000	DISTANCE
WAVELENGTH (ANGSTROMS)	.987536	WAVELENGTH
X-PIXEL COORDINATE OF DIRECT BEAM	0.0	X-BEAM CENTRE
Y-PIXEL COORDINATE OF DIRECT BEAM	0.0	Y-BEAM CENTRE
ROTATION ANGLE OF TILTING PLANE (DEGREES)	0.0	TILT ROTATION
ANGLE OF DETECTOR TILT IN PLANE (DEGREES)	0.0	ANGLE OF TILT

Click on variable to change, or ’O.K.’

Figure 35: The TILT/ BEAM CENTRE REFINEMENT Control Form



TILT / BEAM CENTRE REFINEMENT

(FITTING TO POWDER RINGS)

DESCRIPTIONS	VALUES	CHANGE
NUMBER OF AZIMUTHAL SECTIONS	90	ANGULAR SECTIONS
REJECT OUT-LYING POSITIONS AND RE-REFINE	YES	REJECT OUTLIERS
REJECT LIMIT FROM IDEAL (STANDARD DEVIATIONS)	4.000000	REJECT LIMIT
OUTPUT FULL INFORMATION	YES	FULL INFO
FIND BEST FIT BEAM CENTRE	YES	REFINE BEAM
FIND BEST DETECTOR TILT ANGLES	YES	REFINE TILT

Click on variable to change, or ’O.K.’

optimum value, so trail and error is recommended. Clearly, this value should not be too small.)

REJECT OUTLIERS: This option allows badly fitting "ring" positions to be rejected, and the data re-fitted. This is to make the fit procedure more robust and to allow for erroneous positions owing to contaminating Bragg peaks, etc.

REJECT LIMIT: If **REJECT OUTLIERS** is **YES** then this is the number of standard deviations away from the best fit predicted position after which "outliers" are rejected.

FULL INFO: **YES** for terminal screen output of information relating every stage of the fitting of the powder rings.

REFINE BEAM: **YES** to fit the beam centre position, as well as the tilt angles. **NO** to refine only the tilt angles (if variable). This is usually used when the beam centre has been determined from a direct beam mark.

REFINE TILT: **YES** to fit the detector non-orthogonality to the beam (the tilt). **NO** will keep the tilt angles fixed and only fit the beam centre (if variable). **NO** is often used when the tilt angles have been determined accurately from a high quality calibrant measurement.

Set the required parameters and click **O.K.** to continue.

Next the **BEAM CENTRE MENU** appears. This is shown in Figure 37, Page 92. This is used to set an initial beam centre, which will then be refined if **REFINE BEAM** has been set to **YES**.

If the beam centre has been accurately recorded, e.g. with a semi-transparent beam-stop, then fitting with a 2-D Gaussian may be the best method of setting the beam centre, and it may then be better not to refine it using the powder ring positions.

For further information on the **BEAM CENTRE MENU** see Section 11.4, Page 92.

Depending on whether or not a circle has been selected in defining the beam centre, the user may be requested to select a powder ring for the initial tilt determination.

Next the **RING PROFILE SEARCH LIMIT** is defined by graphical input. Relative to the defined circle this defines an annuli symmetric about the ring. This profile limit will also be used for other rings which are selected for fitting. The search is limited to ring positions within the annuli, so this should be large enough that all rings fit inside, but small enough not to include other rings.

The annuli are draw in white on the image.

Other rings to be used for the tilt refinement are now selected by clicking on them. A single ring is sufficient, but more well defined rings will clearly lead to more accurate results. When finished click within the prompt box.

Starting from the initial beam centre the centre of each sector of powder ring will be determined for the different sectors. This is then fitted with an ellipse to obtain a better estimate of the beam centre. The central positions of all sectors of all the selected rings are then found and the tilt and/or beam centre parameters are refinement to give the best fit to the experimental data positions.

Unless the beam centre is known there are theoretically two equally possible and indistinguishable solutions to the tilt angles from the shape of the ellipses. These have opposite tilt angles, but differing beam centres. **FIT2D** tries to find both solutions, but owing to noise it may find the same one twice. Details of both solutions are output in the terminal window and the user is asked to choose one of them. If some knowledge of the beam centre exists it may be possible to determine the true solution. If not, it is probably not important as both solutions theoretically lead to the same 2θ angles when the data is integrated.

11.3 The INTEGRATE Command

The **INTEGRATE** command allows integration of arbitrary 2-D regions to a variety of 1-D scans. Presently, 2θ, D-Spacing, Q-Space, and equal radial distance element scans may be produced. The **ZOOM IN** command may be used to select a ROI for integration, and the **MASK** command may be used to mask-off "bad" pixels within the ROI so that their value do not influence the results of the integration.

First the **EXPERIMENTAL GEOMETRY CONTROL FORM** appears. An example of this form is shown in Figure 34, Page 87. The values may already have been defined e.g. within the **TILT** command. Clearly, these values are extremely important if the resulting scan is to be correct. When the values are correctly set click the **O.K.** button.

Next the **CONTROL OF RADIAL, 2-THETA, OR Q SCAN RE-BINNING PARAMETERS** control form appears. An example of this form is shown in Figure 36.

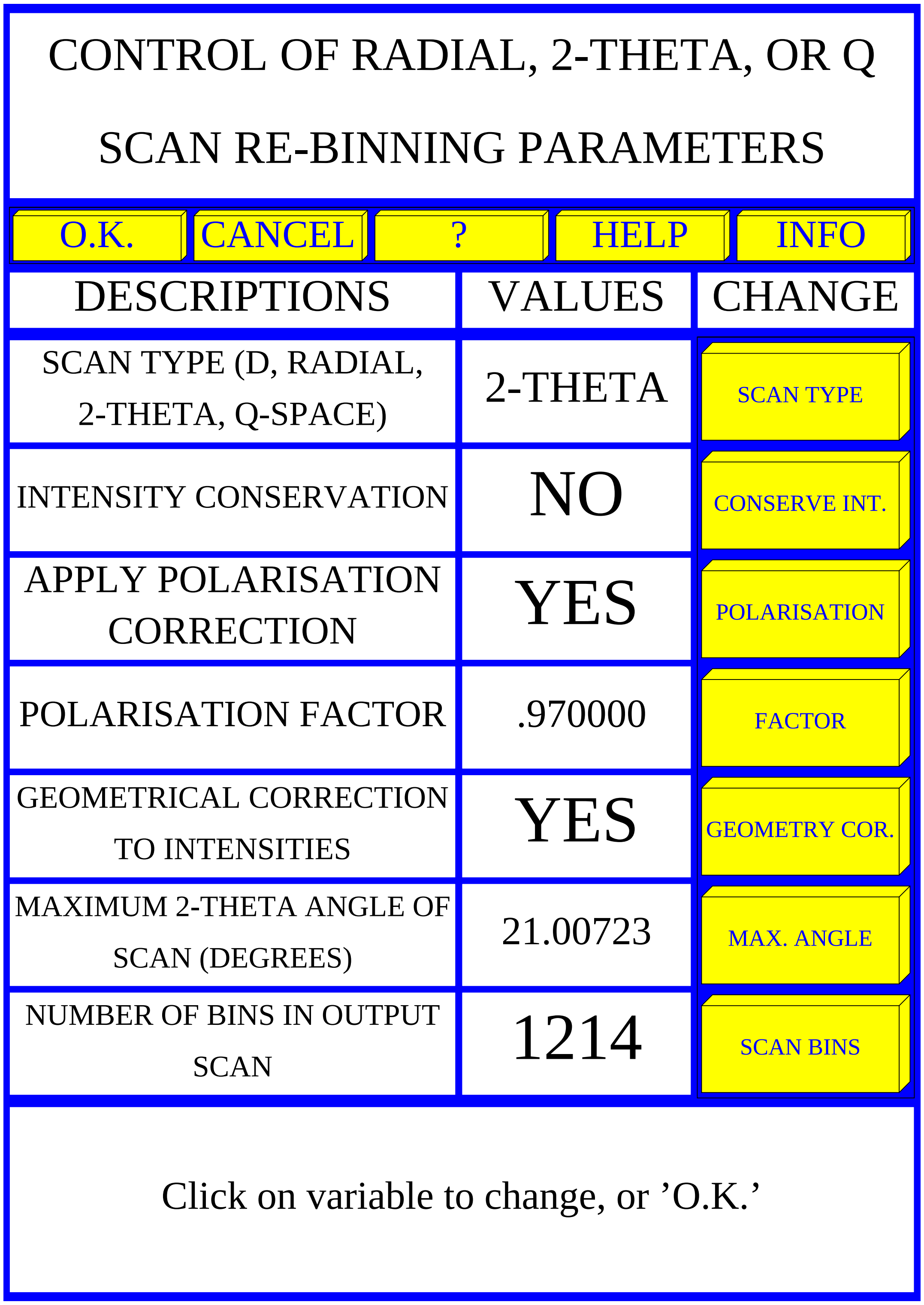
The following control parameters are available:

SCAN TYPE: Allows one of 4 different types of integrated scans to be selected:

- **2-THETA**: This is an equal angle scan, equivalent to a 2θ scan on a powder diffractometer. The scale is in degrees.
- **Q-SPACE**: This is similar to the **2-THETA** scan, but the output elements are in equal Q-range bins. The scale is in inverse nanometres. The definition of Q is $(4\pi/\lambda)sin(2\theta/2)$, where 2θ is the angle of the scattering as recorded on the detector relative to the direct beam.
- **RADIAL**: This is an equal radial distance element scan. The scale is in millimetres.
- **D-SPACINGS**: This converts pixel angles to equivalent D-spacings and outputs an scan in equal D-spacing distance pixels.

CONSERVE INT.: **NO** means that the output intensities are normalised by the number of contributing pixels (although geometrical corrections may be applied). This is appropriate for producing the equivalent of a 2θ scan, and for a Q-space scan, but does not converse total intensity. For applications which require integrated intensities to be conserved, this should be set to **YES**.

Figure 36: The CONTROL OF RADIAL, 2-THETA, OR Q SCAN RE-BINNING PARAMETERS Control Form



CONTROL OF RADIAL, 2-THETA, OR Q SCAN RE-BINNING PARAMETERS

DESCRIPTIONS	VALUES	CHANGE
SCAN TYPE (D, RADIAL, 2-THETA, Q-SPACE)	2-THETA	SCAN TYPE
INTENSITY CONSERVATION	NO	CONSERVE INT.
APPLY POLARISATION CORRECTION	YES	POLARISATION
POLARISATION FACTOR	.970000	FACTOR
GEOMETRICAL CORRECTION TO INTENSITIES	YES	GEOMETRY COR.
MAXIMUM 2-THETA ANGLE OF SCAN (DEGREES)	21.00723	MAX. ANGLE
NUMBER OF BINS IN OUTPUT SCAN	1214	SCAN BINS

Click on variable to change, or ’O.K.’

POLARISATION: **YES** to apply a polarisation correction to the intensities during the integration.

FACTOR: This is the polarisation factor which is used if the polarisation correction is applied.

The polarisation factor is defined as $(I_h - I_v)/(I_h + I_v)$, where I_h is the horizontal component and I_v is the vertical component. (Horizontal should normally correspond to the X-direction of the image.)

GEOMETRY COR.: **YES** corrects a flat "scan" to the equivalent of a 2θ scan, or a Q-space scan. (**CONSERVE INT.** should be set to **NO**). These are the effect of change of distance and obliqueness at higher angles for the flat image plate compared to a detector on a 2θ arm, always facing the sample.

MAX ANGLE: Maximum angle of the output scan in degrees.

SCAN BINS: Number of bins in the output scan. This may be increased up to the size of the program array in the first dimension. Larger values lead to over-sampling with may help obtain better profiles.

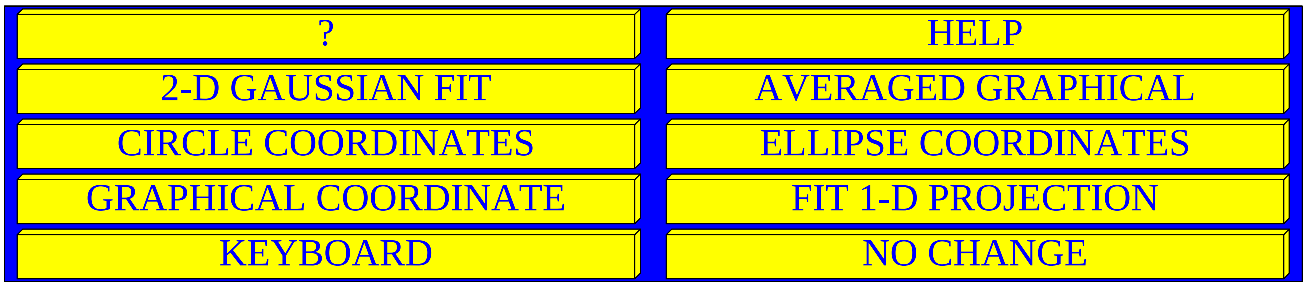
When the required parameters have been set click on the **O.K.** button. Whilst the data is being integrated a progress report will appear in the terminal window.

The 2-D data will be replaced by the 1-D integrated scan, but the 2-D is saved in the "memory" and can be re-called using the **EXCHANGE** command.

11.4 The BEAM CENTRE Command

The **BEAM CENTRE** Command allows the beam centre to be defined or re-defined without changing any other experimental geometric parameters. This may be useful when the detector tilt angles are accurately determined using a high quality calibrant measurement, but data images suffer some non-reproducibility in positions. The **BEAM CENTRE MENU** is shown in figure 37.

Figure 37: The BEAM CENTRE MENU



The following commands are available:

?: List of commands with short description

HELP: More detailed help on available commands

2-D GAUSSIAN FIT: Automatic fitting of a 2-D Gaussian function on the direct beam mark. The user should click on the beam mark to start the fitting.

AVERAGED GRAPHICAL: Set beam centre from the average of entered coordinates. This will work well if there are clearly defined symmetric peaks.

CIRCLE COORDINATES: Set the beam centre from the least squares fit of a circle to 3 or more entered coordinates. This is useful if a well defined powder ring is present.

ELLIPSE COORDINATES: Set the beam centre from the least squares fit of an ellipse to 5 or more entered coordinates. This is useful if a well defined powder ring is present.

GRAPHICAL COORDINATE: Single graphical coordinate click to define the beam centre. This is clearly rough and ready, but may be sufficient for many purposes, in particular an initial beam centre for refinement.

FIT 1-D PROJECTION: This option is designed for grazing incidence SAXS data. A rectangular region is specified in the same manner as for the **PROJECTION** command. Two coordinates define the projection line, and other two coordinates define the width of the 2-D projection region. The 1-D projection is displayed and a menu allowing masking and un-masking of data points. The symmetry point of the 1-D data is refined. For this to work optimally the intensities must be symmetric e.g. there must not be differing absorption effects. The best symmetry point on the 1-D projection line is used to calculate the 2-D coordinate. This is presented to the user as the default X/Y coordinate for the beam centre.

KEYBOARD: Enter beam centre coordinate from input values from the keyboard.

NO CHANGE: Use the existing position.

11.5 The CALIBRANT Command

The **CALIBRANT** button allows the fitting of the powder pattern from a standard calibrant sample. The known D-spacings of the diffracting planes allow the beam centre and tilt to be defined with greater accuracy than with an unknown sample, and if reasonably high angle data is present allow the sample to detector distance, and the wavelength to be refined independently. This has been verified to agree to within the accuracy that may be obtained from scanning an absorption edge using a Ge detector on ID-30 at the ESRF.

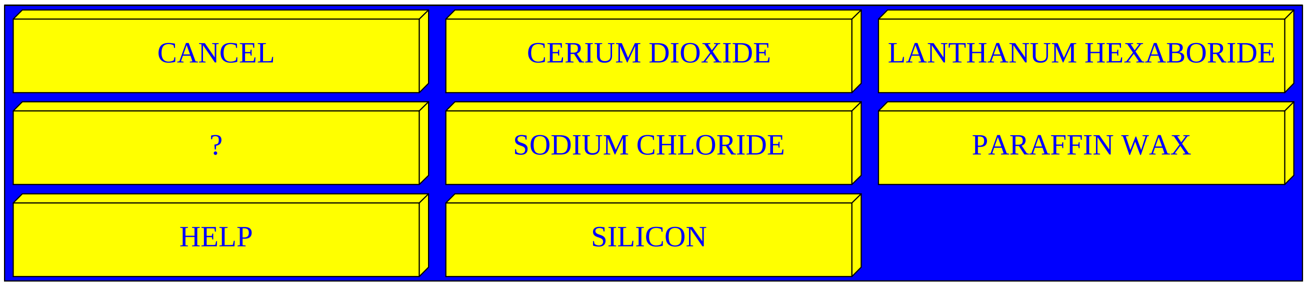
Clearly for best results a very high quality powder sample should be used, and a well exposed image should be obtained.

First the **SELECT CALIBRATION SAMPLE** choice menu appears. This is shown in Figure 38.

The following choice of calibrants is available, and includes the commonly used samples from the American National Institute of Standards and Technology:

CERIUM DIOXIDE: Ceria powder sample e.g. NIST standard

Figure 38: The **SELECT CALIBRATION SAMPLE** Choice Menu



LANTHANUM HEXABORIDE: LaB_6 powder sample e.g. NIST standard

SODIUM CHLORIDE: NaCl Common salt (beware of moisture)

PARAFFIN WAX: Macromolecular crystallographers "standard" calibration wax. **PARAFFIN WAX** is included since many crystallographers use this for distance calibration. However don't expect high accuracy, and if only low angle data is available don't try to refine both wavelength and distance together.

SILICON: Silicon powder sample e.g. NIST standard

After the appropriate calibrant sample has been selected, the **CALIBRANT PATTERN REFINEMENT** control form appears. This is shown in Figure 39.

The following "buttons" are available:

DISTANCE: The sample to detector distance in millimetres.

WAVELENGTH: The radiation wavelength in Ångstroms.

X-PIXEL SIZE: The horizontal (as displayed) size of detector pixels in microns.

Y-PIXEL SIZE: The vertical (as displayed) size of detector pixels in microns.

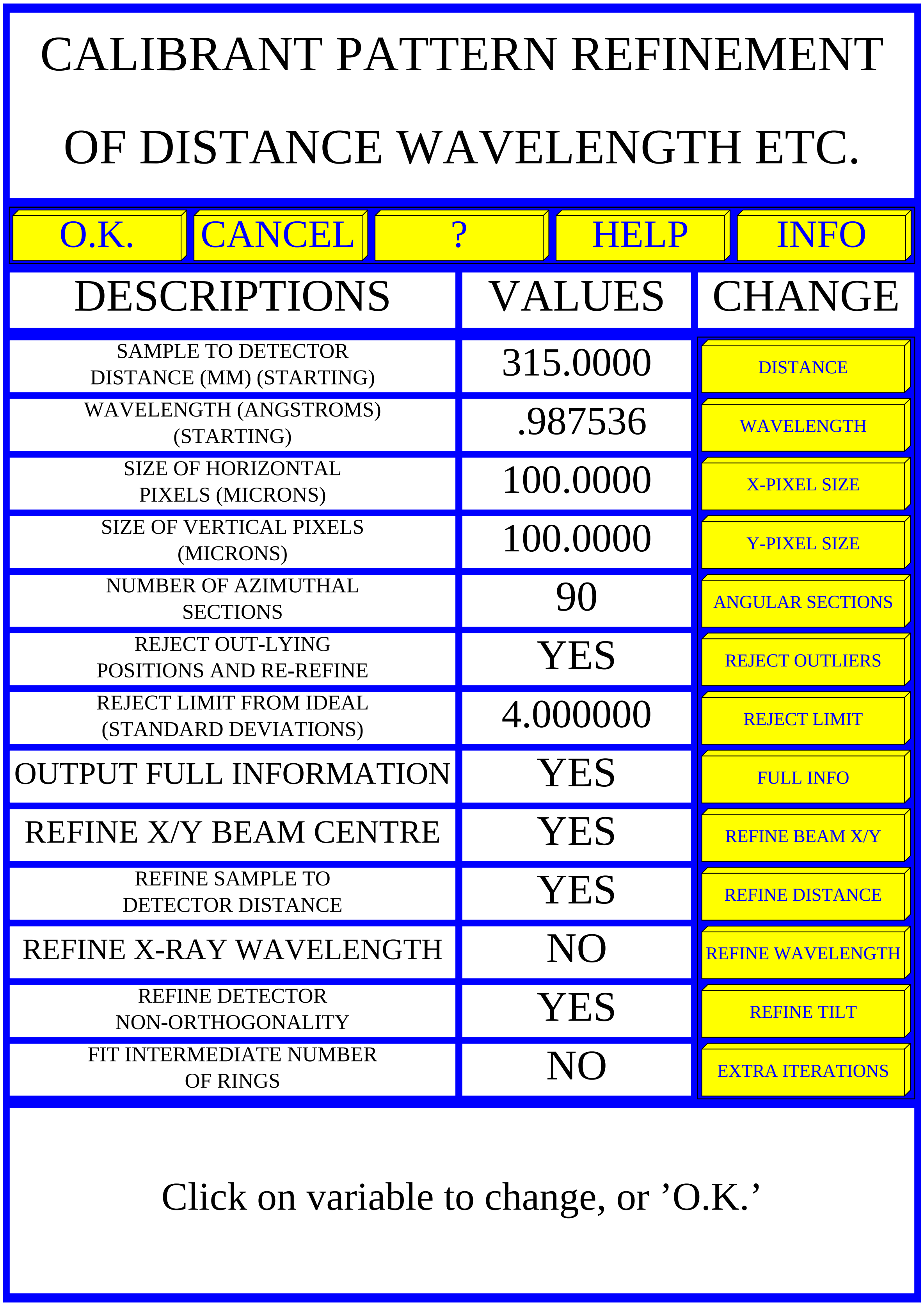
ANGULAR SECTIONS: This the number of angular sections around 360 degrees of the powder rings which are used to calculate the centre of the rings. The beam centre and tilt are fitted to these positions on each ring. If the data is very noisy or shows poor powder averaging then a smaller number may be better. (No theoretical criterion exists to set an optimum value, so trail and error is recommended. Clearly, this value should not be too small.)

REJECT OUTLIERS: This option allows badly fitting "ring" positions to be rejected, and the data re-fitted. This is to make the fit procedure more robust and to allow for erroneous positions owing to contaminating Bragg peaks, etc.

REJECT LIMIT: If **REJECT OUTLIERS** is **YES** then this is the number of standard deviations away from the best fit predicted position after which "outliers" are rejected.

FULL INFO: **YES** for terminal screen output of information relating every stage of the fitting of the powder rings.

Figure 39: The **CALIBRANT PATTERN REFINEMENT** Control Form



CALIBRANT PATTERN REFINEMENT

OF DISTANCE WAVELENGTH ETC.

DESCRIPTIONS	VALUES	CHANGE
SAMPLE TO DETECTOR DISTANCE (MM) (STARTING)	315.0000	DISTANCE
WAVELENGTH (ANGSTROMS) (STARTING)	.987536	WAVELENGTH
SIZE OF HORIZONTAL PIXELS (MICRONS)	100.0000	X-PIXEL SIZE
SIZE OF VERTICAL PIXELS (MICRONS)	100.0000	Y-PIXEL SIZE
NUMBER OF AZIMUTHAL SECTIONS	90	ANGULAR SECTIONS
REJECT OUT-LYING POSITIONS AND RE-REFINE	YES	REJECT OUTLIERS
REJECT LIMIT FROM IDEAL (STANDARD DEVIATIONS)	4.000000	REJECT LIMIT
OUTPUT FULL INFORMATION	YES	FULL INFO
REFINE X/Y BEAM CENTRE	YES	REFINE BEAM X/Y
REFINE SAMPLE TO DETECTOR DISTANCE	YES	REFINE DISTANCE
REFINE X-RAY WAVELENGTH	NO	REFINE WAVELENGTH
REFINE DETECTOR NON-ORTHOGONALITY	YES	REFINE TILT
FIT INTERMEDIATE NUMBER OF RINGS	NO	EXTRA ITERATIONS

Click on variable to change, or ’O.K.’

REFINE BEAM X/Y: **YES** to fit the beam centre position, as well as the tilt angles. **NO** to refine only the tilt angles (if variable). This is usually used when the beam centre has been determined from a direct beam mark.

REFINE DISTANCE: **YES** to refine the sample to detector distance, from the angles of the calibrant powder rings.

REFINE WAVELENGTH: **YES** to refine the radiation wavelength from the angles of the calibrant powder rings.

REFINE TILT: **YES** to fit the detector non-orthogonality to the beam (the tilt).

EXTRA ITERATIONS: Sometimes the initial values are not good enough to find higher angle rings. If **YES** an extra step is introduced which finds intermediate angle rings and refines using them, prior to trying to use all available rings. This should normally not be necessary.

As can be seen it is possible to refine, or not, all the different parameters. It is possible to refine all parameters together, but normally only the sample to detector distance or the wavelength should be refined. However, with high quality, high angle data is possible to refine both together, and get results as good as those obtained by scanning edges.

Set the required parameters and click **O.K.** to continue.

The user is prompted to define a ring of the calibrant pattern, by clicking on the ring 3 or more times. Normally the inner ring is requested. This defines the inner ring, and an initial beam centre.

This should be last user intervention, provided the initialisation is not too inaccurate. If the position of the ring does not correspond within 10% of the given distance and wavelength, then a **WARNING** message will be output and the operation will terminate. In such a case check the initial values and change as required.

When the initialisation is good enough, the routine will successively search for more ring positions and refine the refinable parameters as requested. The coordinates use and the rings predicted are drawn on the image as the command proceeds.

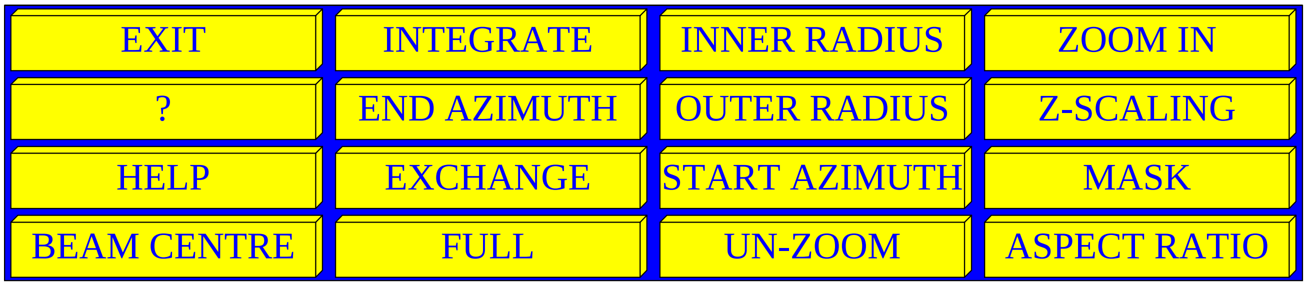
11.6 The CAKE Command

The **CAKE** command is so called, because it allows an arbitrary user "cake" of data to be defined, and integrated to one of a large choice of single and multiple scans.

When the **CAKE** command is first entered, an initial "cake" is defined. First the beam centre is defined in the same manner as the **BEAM CENTRE** command; see Section 11.4, Page 92. This is followed by graphical input for the **STARTING AZIMUTH**, **END AZIMUTH**, **INNER LIMIT**, and lastly the **OUTER LIMIT**. All these values expect the **OUTER LIMIT** have defaults which may be used instead, by clicking within the prompt boxes.

The defined "cake" or integration area, will be drawn on top of the image in inverse video. The **CAKE** sub-menu now appears. This is shown in Figure 40. If the **CAKE** sub-menu is exited, and re-entered within the same **FIT2D** session, the defined "cake" is remembered.

Figure 40: The **CAKE** Sub-Menu



The following commands are available:

EXIT: Exit **CAKE** sub-menu and return to calling menu.

INTEGRATE: Integrate currently defined "cake" region, with a choice of the number of azimuthal and radial/2-theta output pixels. This allows enormous flexibility in defining intensity versus azimuthal angle scans, 2θ scans, multiple 2θ scans, and polar transformations.

INNER RADIUS: Change inner radius/2-theta angle of the currently defined "cake" integration region.

ZOOM IN: Graphical region definition

?: List of available commands and short descriptions.

END AZIMUTH: Change end azimuth of the currently defined "cake" integration region.

OUTER RADIUS: Change outer radial/2-theta angle of the currently defined "cake" integration region.

Z-SCALING: Automatic or user control of intensity display range; see Section 5.3, Page 43.

HELP: Detailed help text

EXCHANGE: Swap current data with the "memory"; see Section 5.2, Page 41.

START AZIMUTH: Change starting azimuthal angle of the currently defined "cake" integration region.

MASK: Mask or Un-mask data (masked pixels are not re-binned); see Section 5.7, Page 56.

BEAM CENTRE: Change beam centre, which defines the zero angle for integration; see Section 11.4, Page 92.

FULL: Set ROI to be all of the currently defined data.

UN-ZOOM: Zoom out to see more of the data

ASPECT RATIO: Control automatic correct aspect ratio (or not). After integration, the output is often highly non-square. For better display it is often better to set **AUTOMATIC CORRECT ASPECT RATIO IMAGE DISPLAY** to **NO**.

When a suitable "cake" has been defined and "bad" pixels masked out, the **INTEGRATION** command should be selected. The **TYPE OF AZIMUTHAL/RADIAL OR 2-THETA TRANSFORM** control form appears. An example is shown in Figure 41.

The following parameters may be controlled:

START AZIMUTH: Change starting azimuthal angle of the currently defined "cake" integration region.

END AZIMUTH: Change end azimuth of the currently defined "cake" integration region.

INNER RADIUS: Change inner radius/2-theta angle of the currently defined "cake" integration region.

OUTER RADIUS: Change outer radial/2-theta angle of the currently defined "cake" integration region.

SCAN TYPE: Allows one of 4 different types of integrated scans to be selected:

2-THETA: This is an equal angle scan, equivalent to a 2θ scan on a powder diffractometer. The scale is in degrees.

Q-SPACE: This is similar to the **2-THETA** scan, but the output elements are in equal Q-range bins. The scale is in inverse nanometres. The definition of Q is $(4\pi/\lambda)sin(2\theta/2)$, where 2θ is the angle of the scattering as recorded on the detector relative to the direct beam.

RADIAL: This is an equal radial distance element scan. The scale is in millimetres.

D-SPACINGS: This converts pixel angles to equivalent D-spacings and outputs an scan in equal D-spacing distance pixels.

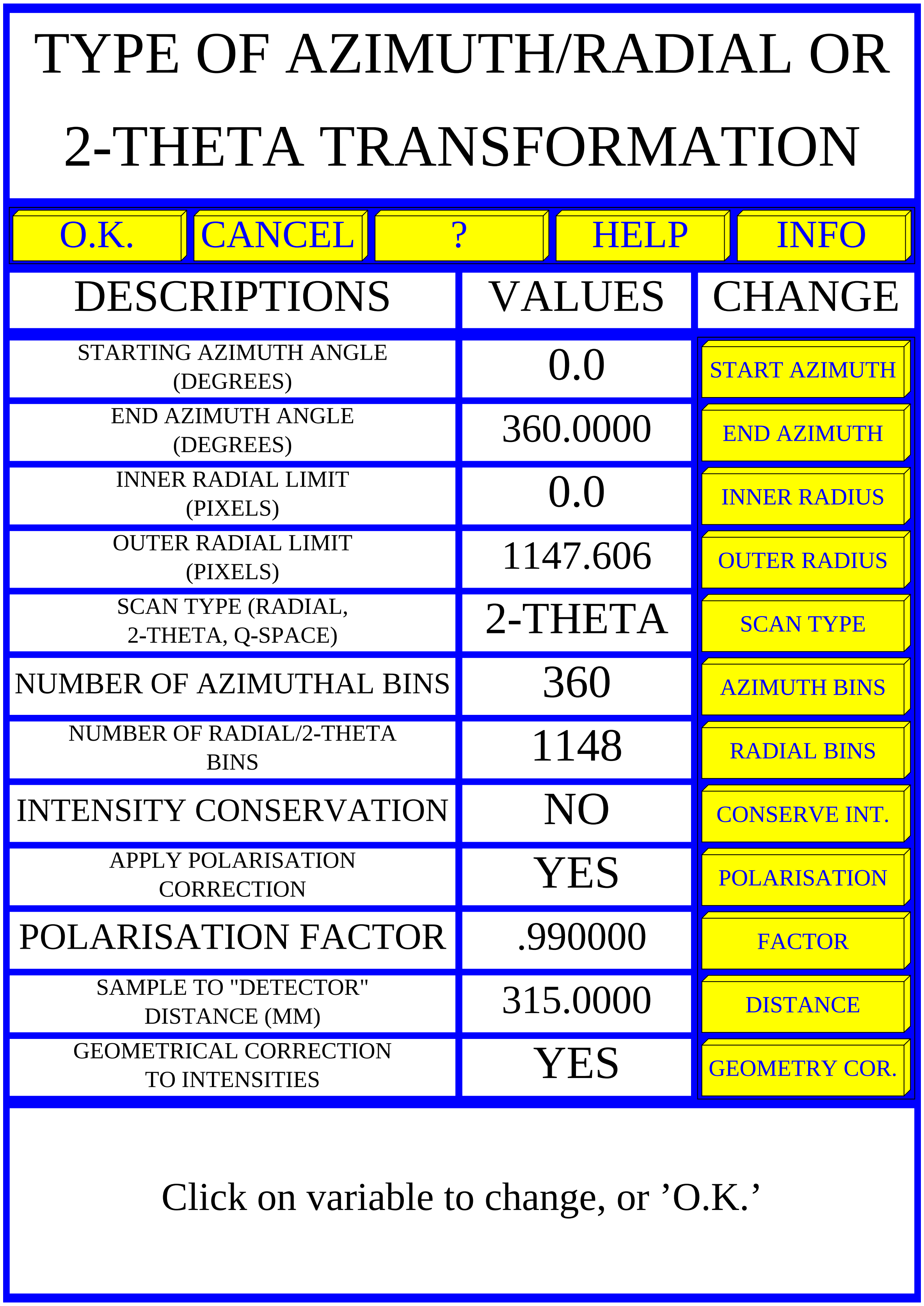
AZIMUTH BINS: Number of bins in the output scan in the azimuthal sense. This may be increased up to the size of the program array in the second dimension.

RADIAL BINS: Number of bins in the output scan in the radial, 2θ, or Q-space sense. This may be increased up to the size of the program array in the first dimension.

CONSERVE INT.: **NO** means that the output intensities are normalised by the number of contributing pixels (although geometrical corrections may be applied). This is appropriate for producing the equivalent of a 2θ scan, and for a Q-space scan, but does not converse total intensity. For applications which require integrated intensities to be conserved, this should be set to **YES**.

POLARISATION: **YES** to apply a polarisation correction to the intensities during the integration.

Figure 41: The **TYPE OF AZIMUTHAL/RADIAL OR 2-THETA TRANSFORM** Control Form



TYPE OF AZIMUTH/RADIAL OR 2-THETA TRANSFORMATION

DESCRIPTIONS	VALUES	CHANGE
STARTING AZIMUTH ANGLE (DEGREES)	0.0	START AZIMUTH
END AZIMUTH ANGLE (DEGREES)	360.0000	END AZIMUTH
INNER RADIAL LIMIT (PIXELS)	0.0	INNER RADIUS
OUTER RADIAL LIMIT (PIXELS)	1147.606	OUTER RADIUS
SCAN TYPE (RADIAL, 2-THETA, Q-SPACE)	2-THETA	SCAN TYPE
NUMBER OF AZIMUTHAL BINS	360	AZIMUTH BINS
NUMBER OF RADIAL/2-THETA BINS	1148	RADIAL BINS
INTENSITY CONSERVATION	NO	CONSERVE INT.
APPLY POLARISATION CORRECTION	YES	POLARISATION
POLARISATION FACTOR	.990000	FACTOR
SAMPLE TO "DETECTOR" DISTANCE (MM)	315.0000	DISTANCE
GEOMETRICAL CORRECTION TO INTENSITIES	YES	GEOMETRY COR.

Click on variable to change, or 'O.K.'

FACTOR: This is the polarisation factor which is used if the polarisation correction is applied. The polarisation factor is defined as $(I_h - I_v)/(I_h + I_v)$, where I_h is the horizontal component and I_v is the vertical component. (Horizontal should normally correspond to the X-direction of the image.)

DISTANCE: The sample to detector distance in millimetres.

GEOMETRY COR.: **YES** corrects a flat "scan" to the equivalent of a 2θ scan, or a Q-space scan. (**CONSERVE INT.** should be set to **NO**). These are the effect of change of distance and obliqueness at higher angles for the flat image plate compared to a detector on a 2θ arm, always facing the sample.

By setting azimuthal bins to 1, a single 1-D radial/2θ/Q-space scan can be obtained.

By setting the number of radial bins to 1, and selecting the "cake" as an annulus about a ring, or an arc peak, an integration of intensity versus azimuth may be obtained.

Leaving both these numbers greater than 1, a 2-D polar transform is obtained. In the output array the X-direction is used to store the radial/2θ/Q-space bins and the Y-direction the azimuthal bins[11].

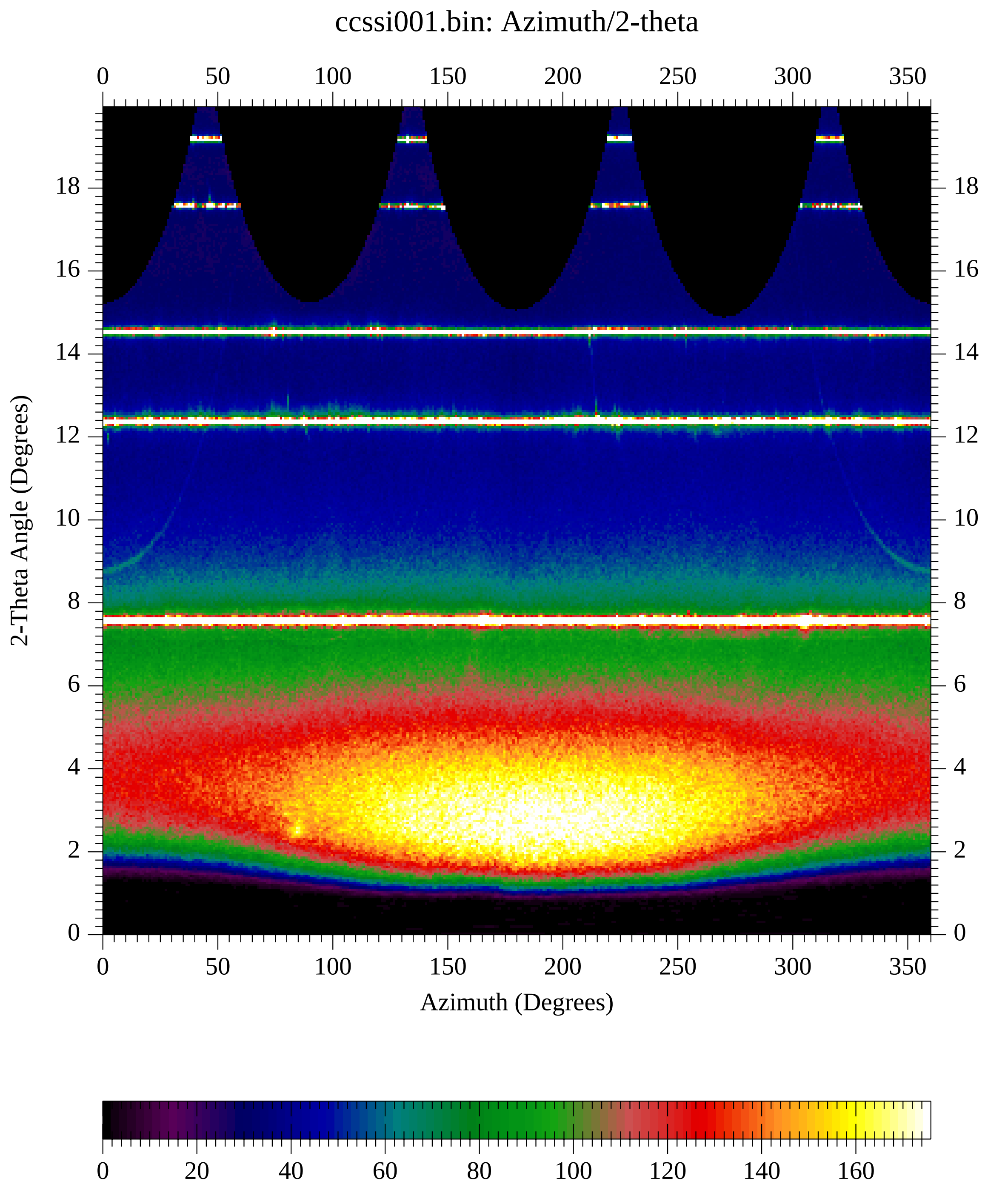
An example of the result of such a transform is shown in Figure 42.

Here the powder rings have become straight lines, at equal 2θ angle. Any wobble in the lines shows inaccuracy in the beam centre, or the detector tilt angles, or spatial distortion in the detector, or maybe deviatoric stress in the sample.

The black region at zero and low 2θ is the beam-stop shadow, and the black regions at high 2θ are area outside the rectangular input region.

[11]Prior to V9.130 the storage was the opposite way around. It was changed so that output of radial/2θ/Q-space scans could easily be input and fitted in the **MFIT** interface. If fitting is required in the azimuthal direction the command **TRANSPOSE** can be used.

Figure 42: Example of a Polar Transform of a Powder Pattern

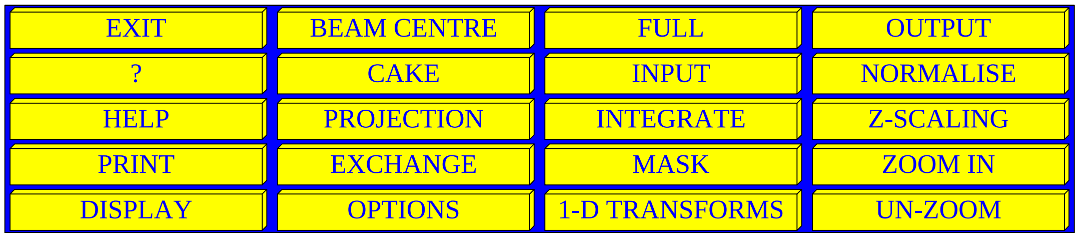


12 The SAXS/GISAXS Interface

This interface is similar to the **POWDER DIFFRACTION (2-D)** interface, but is being tailored for the needs of small angle scattering. It is still under development, so not all the commands have been implemented at present.

The main menu is shown in Figure 43.

Figure 43: The **SAXS/GISAXS** Main Menu



The **SAXS/GISAXS** Main Menu contains the following commands:

EXIT Exit menu.

BEAM CENTRE Determine beam centre by a choice of methods.

FULL View image of full data.

OUTPUT Save data in an output file.

? This help on the menu choices.

CAKE Versatile multiple 2-theta scans integration (see Section 11.6, Page 96).

INPUT Input data from a file on disk.

NORMALISE Apply intensity normalisation corrections. (Not yet implemented.)

HELP Help text on this grphical menu.

PROJECTION Integrate rectangular region to 1-D scan. This uses the beam centre and a user input coordinate to define an integration line. A second user input coordinate defines the width of the integration region. Pixels within this region are effectively “collapsed” onto the integration line, and then re-binned to the required 1-D output scan. This type of integration is specifically for grazing incidence data.

INTEGRATE 2-D to 1-D 2-theta integration.

Z-SCALING Automatic or user control of intensity display range.

PRINT Output current graphics to PostScript file.

EXCHANGE Swap current data with the ”memory”.

MASK Defined masked-off regions of the image.

ZOOM IN Define smaller graphical display region.

DISPLAY Further graphical display possibilities.

OPTIONS Graphics display control menu.

1-D TRANSFORMS Choice of conversion of intensity and Q scales. Allows 1-D integrated scan data to be transformed by the general equation:

$$[Log]I(q)^a * q^b versus [Log]q^c \tag{1}$$

The conversion of the intensity and Q scales to log is optional, and the values of a, b, and c are user controlled.

UN-ZOOM Zoom out to see more of the data.

13 The TEST Interface

The **TEST** integerface is for program development and testing purposes. Normally users will not be concerned with this interface.

14 The KEYBOARD INTERFACE: Introduction And General Use

14.1 FIT2D Prompts and Required Input

By default all user prompts are short but further information is always available by entering a question mark (?).

Menu commands may be shortened to any non-ambiguous command; upper, lower, or mixed case input is fine.

Generally prompts propose to the user a default suggestion, and often define a limited range for acceptable input values. Values outside the defined range will not be accepted and **FIT2D** will re-prompt with a warning message and an explanation of the required input. When a default is given the user can enter < RETURN> to obtain the default value. Default values are used to suggest sensible sizes of values and to guide the user through some of the menu commands.

e.g. To define the X-dimension (horizontal as seen on the screen) the following prompt is issued and information produced to an input which is outside the acceptable range of input:

```
X DIMENSION FOR ARRAYS (Range: 1 to 1000000) [512]:-1
Must be an integer in the defined range
Here you are asked to define the size of the program arrays.  These will
be used to store data "inside" FIT2D.  Normally you will want the arrays
to be at least as large as the image data to be input. If the dimensions
are larger this does little harm,  but is wasteful of system  resources.
If the arrays are smaller then not all of  an image can be input at full
resolution.  Some input options allow an image to  be re-binned on input
or for a sub-region of the image to be input.
X DIMENSION FOR ARRAYS (Range: 1 to 1000000) [512]:
```

The return of default values should save you much typing, but leads to a problem for entering blank character strings. If you try to enter a blank character string for a character input which has a default, the default string will be taken. To allow easy input of blank strings a special program variable has been defined: `##BLANK` . (Program variables are covered in Section 29.1, Page 251). Here all you need know is that a method is available to enter a blank character string. e.g. To set the title of a data-set to be blank the following prompt and input would be used:

```
Main menu: ENTER COMMAND [GAUSSIAN]:title
NEW TITLE [Simulated Data]:##BLANK
```

Previously issued commands may be recovered, edited, and re-used using the arrow keys ("command line history recall") in the same manner as the "TC-shell" or **emacs**. The up-arrow key may be used to go backwards through the "history", and the down-arrow key forwards. Once a history line has been selected it may be edited using the left and right-arrow keys, the delete key, and by typing text which will be inserted automatically. The < RETURN> key may be used to enter the whole line of text regardless of the position of the cursor[12].

When entering a file name it is possible to type only part of the file name and have the name automatically completed. After typing one or more characters of the file name you may type the `<TAB>` key and **FIT2D** will complete or partially complete the file name up to a difference between two file names. If the characters uniquely define a file the name will be completed. **emacs** uses will be familiar with this functionality as it is provided by the GNU Readline library.

Any text which extends for more than a page (24 lines) is controlled by a "pager" which allows forward and backward scrolling and keyword searching.

Generally the user can enter a double backslash (\\) instead of the normal input to "escape" from a particular command[13]. This should return the program to the main menu[14].

Commands may be sent to the operating system from any of the keyboard input menus (V7.26). A dollar needs to be typed before the command to be sent. e.g. To send the command `ls -al` to a Unix system, you would enter `$ls -al`. e.g.

```
Main menu: ENTER COMMAND [IMAGE]:$ls -al
```

Note: i. Unix systems do not seem to be able to interpret `aliases` when sending a command in this fashion. ii. Any output to the terminal from the operating system is not included in a log file.

14.2 Important Main Menu Commands

The keyboard interface main menu features commands of many different types, to be able to perform basic image operations, visualisation, style customisation, and other support operations such as input and output. Only specialist commands related to 2-D detector system calibration and to fitting have been "hidden" in sub-menus[15].

The commands may be grouped by type, i.e.:

Arithmetic `CADD, CDIVIDE,` and `CMULTIPLY` are all scalar operations which prompt for the value of the scalar and apply the operation throughout the *ADR*. `LOG` takes the logarithm base 10 throughout the *ADR*, and `RAISE TO A POWER` takes all elements in the *ADR* to

[12]More commands are available, but these are the most important ones. GNU Readline documentation describes fully the available possibilities.

[13]Previously, <CONTROL>⊗D (Unix) or <CONTROL>⊗Z (VMS) were the normal "User escape" commands, however **FIT2D** now uses the GNU Readline library for terminal input which enables command line history recall and file name completion, but using <CONTROL>⊗D for its own purposes.

[14]It is possible that there are still some occasions when this "user escape" option does not work as it should. If this happens please inform me of the command so that it can be rectified for future versions.

[15]For keyboard driven programs many users dislike having to go up and down many menus and sub-menus to find commands. The calibration and fitting operations have nevertheless been put in separate sub-menu to reflect the much greater specialist knowledge required to use these routines.

the requested power. `ADD, SUBTRACT, DIVIDE, MULTIPLY,` and `LOG` are all commands which work on two images (one in the memory) and perform the stated operation element by element throughout the *ADR*.

Geometry `MOVE` and `REFLECT` allow rotation, translation, and reflection of data, the output is in the memory. `FLIP` allows the data to be reflected about its horizontal or vertical middle. `TRANSPOSE` transposes the *ADR*, output in the memory i.e. `DATA(x, y)` is transferred to `MEMORY(y, x)`.

Filtering `BLUR, MEDIAN FILTER` and `SPATIAL FILTER` perform filtering operations of the *ADR*, output in the memory.

Visualisation `IMAGE, CONTOUR PLOT` and `3-D PLOT` allow the *ADR* to be visualised in a number of different forms.

Style Control `COLOUR TABLE` and `ROTATE LUT` allow the colour table to be changed. Many other commands allow aspects of the graphics output style to be controlled e.g. `GRID` allows horizontal and vertical grid lines to be added to the graphics.

The most important menu commands are:

`INPUT DATA, REGION, ROI, IMAGE, PRINT GRAPHICS, CALIBRATION, FIT,`
and `EXIT`.

`INPUT DATA` allows data to be input one of a number of defined file formats. More formats will be added as appropriate.

`REGION` allows the region that commands act upon to be changed. Initially the region will be the whole data, but often it is very useful to restrict actions to only a sub-set of the data. This sets the ACTIVE DATA REGION (*ADR*). Understanding the *ADR* concept is very important to efficient and powerful use of **FIT2D**.

`ROI` also allows the *ADR* to be changed, but works using pixel numbers instead of data coordinates.

`IMAGE` plots previously input data as a false colour image according to the current attributes (style) which have been set by the user.

`PRINT GRAPHICS` will output the current graphics in a PostScript output file. The first time this is called the user is prompted for the name of the output file.

`CALIBRATION` enters the calibration sub-menu with commands allowing calibration data to be analysed and calibrations defined and applied to subsequent scientific data.

`FIT` enters the fitting sub-menu with commands allowing a fit model to be defined and minimised.

`EXIT` (or `QUIT`) is used to exit, **FIT2D** demands confirmation, to avoid accidental loss of results.

14.3 FIT2D Terminal Output

Initially header text is output to the terminal. This contains the version number of the version of **FIT2D** being used, and may contain some information on new possibilities or warnings concerning changed functionality. In the case of problems it is useful to note the version number being used (See Section 24.3, Page 236).

On start-up you are prompted for the size of the program arrays to be created to hold image data, and whether or not to create variance arrays. When these questions are answered **FIT2D** enters the main menu loop. This is recognisable by a prompt of the following type:

```
Main menu: ENTER COMMAND [INPUT DATA]:
```

The text `Main menu:` tells you that you are in the main menu, as opposed to one of various possible sub-menus. The text `ENTER COMMAND [******]:` tells you that **FIT2D** is waiting for a menu command to be entered. The text within the square brackets is a suggested default command. On start-up the suggested default is the `INPUT DATA` command, but at later stages many other commands are suggested as defaults.

When one of the possible sub-menus is entered the command prompt is changed to reflect the sub-menu entered. e.g.

```
Calibration sub-menu: ENTER COMMAND [FIND PEAKS]:
```

is the prompt output when the `CALIBRATION` command is issued and the calibration sub-menu is entered.

Commands which require various numerical constants or text strings to be defined will output prompts similar to those output for the size of the program array dimensions. e.g. The `CADD` command needs to input the addition constant value:

```
ADDITION CONSTANT [1.00000]:
```

All such prompts are in capital letters. The `[1.00000]` is as for the menu command prompts a default return value. Further information on required inputs (obtained by entering a question mark (?)), is in mixed case. e.g.

```
Enter real value to add to active data region
ADDITION CONSTANT [1.00000]:
```

In addition to the normal prompt text, other program output may appear spontaneously during the execution of certain commands. These messages fall into three general categories and are all started with text to indicate the category:

`INFO:` These are information messages, such as progress reports for operations which may take a long time, or quantities and results calculated by **FIT2D**.

`WARNING:` These are generally warning messages which the user should take note of, but which hopefully do not require exiting **FIT2D**. Such a warning may be issued when a file was not found.

`ERROR:` These are generally important error messages which may mean further processing is impossible. Such a message is issued when virtual memory allocation fails.

The exact difference between an information, warning, or serious error, message is impossible to define. Thus these are only general categories. Sometimes a `WARNING:` message can be treated as information, and processing can carry on without a change of strategy. Similarly, an `ERROR:` message can sometimes be treated as only a warning by an experienced user. e.g. If virtual memory allocation fails it may be possible to reduce the requirements and try again.

14.4 The "Pager"

When more than a page of text is output to the user it is controlled by a "pager". This allows the user to move easily backwards and forwards through the text as well as allowing some more powerful options such as keyword searching.

The text is output a number of lines at the time. `f` or < RETURN> gives the next set of lines, `b` may be used to get the previous set of lines, `s` goes to the start of the text, `q` to quit the text, and a line number followed by < RETURN> may be typed to go immediately to a required line number in the text.

`?` or `h` gives a list of the available commands, which are all a single letter or a number. The letters may be entered in upper or lower case. `n` allows the number of lines in a "page" to be user set.

Keywords may be searched for with the `>` and `<` commands, for forwards and backwards searching respectively. The searching is "auto-wrapping" i.e. if a keyword is not found at by the end of the text (forwards search) the search is carried on from the beginning of the text. The user is prompted for repeat searching. By default the searching is insensitive to letter case. This behaviour may be toggled on and off using the command `c`.

14.5 On-line HELP

On-line help is available using the `HELP` command. This provides "pages" of information similar to the information in this document. The output of the help text is controlled by a "pager" so that the user can scroll backwards and forwards, and search through the text for keywords (See Section 14.4, Page 108).

Further information is available for every prompt, which may be obtained by entering a question mark (?). (The initial user prompt messages are deliberately kept to one line.)

14.6 Modifying Text

The commands `TITLE`, `X-AXIS LABEL`, `Y-AXIS LABEL`, and `Z-AXIS LABEL` allow the corresponding items to be replaced or defined. e.g. `TITLE` allows a new title to be defined and `TEXT` allows accompanying text to be defined.

14.7 Modifying Graphics Style

The style of all graphical items can be altered using the `SET **** STYLE` commands. e.g. `SET ANNOTATION STYLE` can be used to change the text font, size, and colour used to draw the annotation labels.

15 The KEYBOARD INTERFACE: Command Summary

Here the "keyboard" interface main menu commands are described in detail in alphabetic order.

15.1 `1-D INTERPOLATION`

Applies a 1-D linear scaling interpolation to data points in the current *ADR*.

15.2 `3-D SURFACE PLOT`

Allows the *ADR* of the current data to be viewed as a 3-d surface, together with colour. The view angle may controlled by the user.

Users should not try to view large numbers of pixels with many colours as this can require very large amounts of system memory. Users are recommended to use `REGION` to restrict the viewing region and `REBIN` to further reduce the number of pixels. `COLOUR TABLE` should be used to reduce the number of colour indices by setting `MINIMUM INDEX` and `MAXIMUM INDEX` to use a limited range of colour indices. (System memory requirements will be roughly proportional to the number of pixels multiplied by the number of colour indices.)

Based on the starting view position the surface is calculated and drawn from back to front. This means that no hidden surface removal is necessary, yet the surface will appear correctly. Graphical menu "buttons" allow the user to change the angle and zoom in and out. However, it is possible to change the viewing angle such that it is outside the range for which the back to front drawing method is correct.

To obtain hard-copy is it necessary to click on the `PRINT` "button" within the menu and specify a file name.

Note: This command is still under development.

15.3 `?`

Entering a question mark will produce a list of all the available main main commands followed by a brief description of the corresponding operation. The list is controlled by the pager (See Section 14.4, Page 108). (Further information on the commands may be obtained by the command `HELP`).

15.4 `ADD`

Adds the contents of the "memory" to the current data array throughout the *ADR*. If the "memory" data is not defined for any of the pixels in the current data *ADR* this operation will fail and a warning message will be produced. The `INFORMATION` command may be used to obtain the current sizes of the current data and "memory" images, and the *ADR* for each.

15.5 ANNOTATION LABEL

All graphical display (images, contour plots, X/Y graphs) may contain additional annotation labels. These text labels may be arbitrarily placed and may include arrows to point to areas of the graphics. The annotation labels and arrows may be defined in either page coordinates or data coordinates (See Section 21, Page 231). By using data coordinates the relative position of the label or arrows will not change if different regions of the data are displayed.

Many different annotation labels may be added and their style may be controlled individually using `SET ANNOTATION STYLE` (See Section 15.91, Page 165) and `SET ARROW STYLE` (See Section 15.92, Page 165).

15.6 ASPECT RATIO

This option controls the "aspect ratio" of data displayed as images. By default all images will be displayed in their correct aspect ratio i.e. the size of the image will be proportional to the number of pixels in each dimension. An alternative way of thinking of this, is to say that the individual pixels are drawn as squares.

Sometimes, especially with images with highly differing numbers of pixels in their dimensions, it is preferable to not use the correct aspect ratio for display. The prompt:

```
AUTOMATIC CORRECT ASPECT RATIO IMAGE DISPLAY [YES]:
```

allows this to be controlled.

The following example sets the graphics system so as to not use the correct aspect ratio:

```
AUTOMATIC CORRECT ASPECT RATIO IMAGE DISPLAY [YES]:?
Enter  ``YES''  if  you want image display with automatic correct aspect
ratios i.e. the pixels are square. Enter ``NO'' to use all the available
display region.  This  may  result in non-square pixels,  but  may  be
preferable for very non-square images.
AUTOMATIC CORRECT ASPECT RATIO IMAGE DISPLAY [YES]:n
```

If automatic correct aspect ratio is not set then the image will fill all the region which has currently be defined for it.

(The aspect ratio can also be controlled from the **OPTIONS** menu of the GUI.)

15.7 AUTOCORRELATION

Calculates the autocorrelation function of the current ADR. At present this must be a power of two pixels in length for both of the two dimensions. The circular autocorrelation function is calculated. To calculate a non-circular autocorrelation, first place the region in a larger region of zero valued pixels. At least 2n-1 elements are necessary to completely avoid any wrap-around.

The prompt:

```
CENTRE OUTPUT [YES]:
```

appears. If the default "YES" is accepted, then the zero "frequency" will be centred in the output, otherwise the zero "frequency" correlation will be the first pixel.

15.8 AXES SCALES

`AXES SCALES` allows the scales for the X and Y-axes to be changed. This can be useful and sometimes necessary since various commands, e.g. 3-D Surface Display, use the size of the axis elements to determine display and calculations.

Each axis has a starting value, for the first element, and a step value, between all successive elements.

You will be prompted:

```
1ST X-AXIS ELEMENT VALUE [.500000]:
X-AXIS ELEMENT INCREMENT [1.000000]:
1ST Y-AXIS ELEMENT VALUE [.500000]:
Y-AXIS ELEMENT INCREMENT [1.000000]:
```

15.9 BANNER

This command allows the **FIT2D** graphics window header, or banner, page to be saved to a PostScript file for printing or inclusion in a document [16].

You are prompted for the name of the file to store the PostScript output.

15.10 BLUR

Blurs (smoothes) data in the *ADR* by convolution with a top-hat function of user specified size[17]. The output is in the memory.

[16]This is a command that I use for making **FIT2D** posters, and documentation.

[17]The present implementation is much faster than the previous version (20 times for 10×10 top-hat).

The user specifies the number of pixels which define the top-hat rectangle. If the number of pixels if even then the centre of gravity of peaks will be slightly changed (right and up). If the number is odd then the effect of the convolution is symmetric and no change in centre of gravity occurs.

15.11 BRAGGS' EQUATION

`BRAGGS' EQUATION` allows a number of different simple uses of Braggs' equation. From known 2θ angles and wavelength (or energy), the corresponding d-spacings may be calculated using the `D-SPACING` sub-menu option. From known d-spacings and wavelength the corresponding 2θ angles may be calculated using the `TWO THETA ANGLE` option. From known d-spacings and 2θ angles the wavelength (and photon energy) may be calculated using `WAVELENGTH/ENERGY`. The commands `EXIT` or `QUIT` can be used to return to the main menu.

Here is an example of use, which converts a 2θ angle at one wavelength into the equivalent diffraction peak angle at a different wavelength:

```
Main menu: ENTER COMMAND [INPUT DATA]:bragg
Bragg equation sub-menu: ENTER COMMAND [D-SPACING]:
WAVELENGTH (Angstroms) (0.0 for keV )
 (Range: 0.0 to 1.000E+05) [1.54061]:
INFO: Energy =    8.047779     keV
TWO THETA ANGLE (Degrees) (Range: 0.0 to 180.000) [20.0000]:69.79
INFO: D-spacing (Angstroms) =      1.34652
Bragg equation sub-menu: ENTER COMMAND [TWO THETA ANGLE]:
WAVELENGTH (Angstroms) (0.0 for keV )
 (Range: 0.0 to 1.000E+05) [1.54061]:0.
ENERGY OF RADIATION (keV) (Range: 1.000E-03 to 10000.0)
 [8.04778]:16.5
INFO: Wavelength =    .7514255     Angstroms
D-SPACING (Angstroms) (Range: 1.000E-03 to 1.000E+06)
 [1.34652]:
INFO: Two theta angle (Degrees) =     32.40415
Bragg equation sub-menu: ENTER COMMAND [D-SPACING]:exit
```

15.12 CADD (Constant ADDition)

Adds a constant to every element throughout the *ADR*. The user is prompted for the value of the constant.

15.13 CALCULATOR

The `CALCULATOR` command enters the reverse Polish notation calculator sub-menu. The sub-menu commands are mainly the normal calculator arithmetic and function commands. At any time values may be entered which are stored one by one on the "stack", or an operator command may be entered to perform an operation on one or more of the values previously entered on the stack. The result replaces the values operated upon. In addition to operator commands such as `*, +, -, /` stack manipulation and display commands are available. For a full list of the available commands type `?`. As always the command `EXIT` or the command `QUIT` is used to return to the main menu. The calculator may be used in normal arithmetic mode, but also supports complex arithmetic. To enter complex constants type two values separated by a space and/or a comma.

The complete list of available commands and their function is:

`?` List of available operators and commands

`+` Add top two elements of stack

`-` Subtract lower element of stack from top element

`*` Multiple top two elements of stack

`/` Divide top element of stack by lower element

`1/X` Form reciprocal of top of stack

`ABSOLUTE` Absolute value of top element: Sqrt(r**2 + i**2)

`ACOSINE` Take Arc Cosine of top element of stack

`ARCCOSINE` Take Arc Cosine of top element of stack

`ARCSINE` Take Arc Sine of top element of stack

`ARCTANGENT` Take Arc Tangent of top element of stack

`ASINE` Take Arc Sine of top element of stack

`ATANGENT` Take Arc Tangent of top element of stack

`ADDITION` Add top two elements of stack

`CLEAR` Clear stack (remove all elements)

`COSINE` Take cosine of top element of stack

`DIVISION` Divide top element of stack by lower element

`DUPLICATE` Copy top of stack onto stack

`ENERGY` Convert wavelength (Angstroms) to photon energy (keV)

`EXCHANGE` Swap top two elements of stack

`EXIT` Exit fit sub-menu

`EXPONENTIAL` e to the power of the top element of stack

`INTEGER` Truncate value to an integer

`LN` Take natural logarithm of top element

`LOGARITHM` Take logarithm base 10 of top element

`MAXIMUM` Greater of top two elements of the stack

`MEMORY` Output value of memory

`MODULUS` Remainder when top of stack divided by lower value

`MINIMUM` Minimum of top two elements of the stack

`MULTIPLICATION` Multiple top two elements of stack

`NEGATION` Multiple top element by -1.0

`PI` Ratio of circumference to diameter of a circle

`POP` Remove top element from stack

`POWER` Raise top element to the power of the lower element

`PUSH` Duplicate top element (adds to stack)

`QUIT` Exit calculator sub-menu

`R1` Place value from register 1 onto stack

`R2` Place value from register 2 onto stack

`R3` Place value from register 3 onto stack

`R4` Place value from register 4 onto stack

`QUIT` Exit fit sub-menu

`RECALL` Place value from memory onto stack

`RECIPROCAL` 1/x, reciprocal of top element

`REGISTERS` Values of current registers

`S1` Store top element in register 1

`S2` Store top element in register 2

`S3` Store top element in register 3

`S4` Store top element in register 4

`SINE` Take sine of top element of stack

`SQUARE ROOT` Replace value by its square root

`SQRT` Replace value by its square root

`STACK` Shows contents of the stack

`STORE` Save value of top element in memory

`SUBTRACTION` Subtract lower element from top element

`SYMBOL` Store top of stack as a named program variable

`TANGENT` Take tangent of top element of stack

`VARIABLE` Store top of stack as a named program variable

`WAVELENGTH` Photon energy (keV) to wavelength(Angstroms)

Note: Since the commands may be entered in any shortened non-ambiguous form, common forms of naming the functions will work e.g. `sin, cos, exp, tan` are all understood.

The `VARIABLE` command allows the value in the top of the stack to be stored as a program variable for later use, and use in macros (see Section 19, Page 225).

15.13.1 Examples of Calculator Usage

At first a reverse polish notation calculator may seem complicated to use, but with a small amount of practice it is very easy, but of course different [18].

Before any operation, e.g. multiplication of two numbers, can be performed the operands, e.g. the two numbers, must be first entered onto the "stack". Operands (numbers) are entered onto the stack simply by typing their value followed by the < Return> key. Each new value is placed on the top of the "stack", previous values being stored underneath it.

The following example shows the program output produced when multiplying 0.496 and 3.567.

```
Main menu: ENTER COMMAND [INPUT DATA]:calc
ENTER VALUE OR OPERATOR:0.496
ENTER VALUE OR OPERATOR:3.567
ENTER VALUE OR OPERATOR:*
RESULT:   1.769232      .0000000E+00
ENTER VALUE OR OPERATOR:exit
```

The following example shows how to calculate $\cos^2(20)+\sin^2(20)$ (note trigonometric functions work in degrees).

```
ENTER VALUE OR OPERATOR:20
ENTER VALUE OR OPERATOR:cos
RESULT:   .9396926      .0000000E+00
ENTER VALUE OR OPERATOR:2
ENTER VALUE OR OPERATOR:pow
RESULT:   .8830222      .0000000E+00
ENTER VALUE OR OPERATOR:20
```

[18] There is a mis-conception that reverse polish notation calculators are "better". This is completely un-true; they are neither better nor worse, just much simpler to program, as no parser is necessary.

```
ENTER VALUE OR OPERATOR:sin
RESULT:   .3420201      .0000000E+00
ENTER VALUE OR OPERATOR:2
ENTER VALUE OR OPERATOR:pow
RESULT:   .1169778      .0000000E+00
ENTER VALUE OR OPERATOR:+
RESULT:   1.000000      .0000000E+00
```

The following example shows the entry of complex numbers and their multiplication.

```
ENTER VALUE OR OPERATOR:1 -1
ENTER VALUE OR OPERATOR:0 -1
ENTER VALUE OR OPERATOR:*
RESULT:  -1.000000     - 1.000000
```

The following example shows how to find out the photon energy (in keV) corresponding to a wavelength of 1.0Å.

```
ENTER VALUE OR OPERATOR:1
ENTER VALUE OR OPERATOR:energy
RESULT:   12.39852      .0000000E+00
```

15.14 CALIBRATION

Enters sub-menu for calibration and correction or spatial distortions. See Section 16, Page 176 for full description of available commands and estimation of calibration functions and application for correcting images.

15.15 CDIVIDE (Constant DIVIDE)

Divides every element throughout the *ADR* by a constant. The user is prompted for the value of the constant.

15.16 CHANGES

Paged text with details of changes which occurred in different versions of **FIT2D**.

15.17 CLEAR DATA

Sets all elements in the *ADR* to zero. If variance array exists the corresponding elements are also set to zero.

15.18 `CLOSE LOG`

Closes a log file if already open (if not, does nothing).

15.19 `CMULTIPLY` (Constant MULTIPLY)

Multiplies every element throughout the *ADR* by a constant. The user is prompted for the value of the constant.

15.20 `COLOUR TABLE`

`COLOUR TABLE` allows the type of false colour table used to represent intensity values of 2-D images to be changed. A choice of a number of different colour schemes is available to the user[19]. The different available colour schemes are referred by a name e.g. `TEMPERATURE, GREY-SCALE, GEOGRAPHICAL`. As with entering menu commands the shortest non-ambiguous letter sequence may be entered, and a question mark (`?`) may be entered to obtain a list all available choices.

Users are recommended to try out the different colour tables, as different data and different aspects of of data may be highlighted by different colour tables. The choice includes grey scales. Generally colour allows most detail to be seen in a single image, but colour can sometimes be confusing and grey scales may be useful to allow the brains natural image processing to spot features or trends in the data.

In addition to the variety of colour tables, the user has control on the range of colour indices used on the workstation or X-terminal to display the image and on the number of separate colour levels used to display the image.

Since windowing systems allow several different windows to have their own colour tables it is possible to have more defined colours than the physical possibilities of the hardware (normally 256 colours for ESRF workstations). If this happens then when a user clicks in a different window the colours of other windows may well change (this effect is known as "flash"). To try to avoid "flash" **FIT2D** does not by default use all of the colour table. `MINIMUM INDEX` and `MAXIMUM INDEX` control the range of colour table indices that **FIT2D** uses for setting colours and displaying images. (Because the precise number of colours used by other windows can vary it is possible that "flash" occurs even using the default settings.) If these values are changed the colour table indices affected will be altered as part of this command, but the indices used to display the image will not be changed, so it is necessary to use the `PLOT DATA` command to re-display the data as desired.

By displaying a very limited number of different colours it is possible to obtain a kind of colour contour plot. Whilst not perfect this is far "cheaper" in terms of calculation than a true colour contour plot. `NUMBER OF LEVELS` allows the number of separate displayed colours to be changed.

[19] This range is likely to be added to in the future.

15.21 CONCATENATION

During complicated macros it is often useful to be able to define new variables, e.g. file names, based on the values of other variables (see Section 29, Page 251). The joining together of two or more character strings to form a longer character string is known as concatenation.

This command allows two strings to be joined together and used to define the value of a new program variable. This is best demonstrated with a realistic example:

If the two variables `#FILE_BASE` and `#EXTENSION` have been defined, we can define a new variable `#FILE_NAME` to be the concatenation of the two with the following:

```
Main menu: ENTER COMMAND [PLOT DATA]: concat
ENTER FIRST STRING [fit2d.]: #FILE_BASE
ENTER SECOND STRING [f2d]: #EXTENSION
INFO: Concatenation:
      file.bin
ENTER VARIABLE NAME [#FILE_OUT]: #FILE_NAME
```

15.22 CONTOUR PLOT

This will display the *ADR* of the current data in contour plot form. It should be noted that this is more calculation intensive than producing a false colour image.

(At present no attribute control is available, but if control over the number of contour lines, and their style is required it can be added.)

15.23 CREATE DATA

Normally if you try an instruction such as `IMAGE` before any data has been input you will receive a warning message telling you that first you must `INPUT DATA` or `CREATE DATA`. `CREATE DATA` makes **FIT2D** create program data, although initially the data array elements are all zero. This can be useful for simulation purposes.

The user is prompted for the size of artificial data to create. This must clearly be not greater in size than the current program arrays.

15.24 CURVE STYLES

This is a shorter form of the command `SET CURVE STYLES` (See Section 15.96, Page 166).

15.25 DEDUCE FILE SEQUENCE

Often it is desired to run a series of data analysis operations on a whole series of files. This can be achieved by running a macro on a whole series of files using the "SEQUENCE" command (Section 15.90, Page 160), or by writing a single macro with a loop which runs on different files (see Section 29, Page 251).

When a macro is to be run on a series of files, it is necessary to deduce the file sequence. i.e. the number of the first and last file in the sequence, the base part of the file name, and the file extensions. `DEDUCE FILE SEQUENCE` allows a starting file name and an end file name to be entered, which may of course be program variables, and if possible deduces the parameters of the sequence, and sets a number of internal program variables.

- `##PREFIX` : This is a character string set to the non-changing characters at the start of the file names.
- `##START` : This is an integer value from the starting file in the series
- `##END` : This is an integer value from the last file in the series
- `##STEP` : This is an integer value for the step from one file in the series to the next. This will normally be set to 1, or to -1 if the last file number is less than the first file number.
- `##VARIABLE` : This is a logical (boolean) value (`TRUE` or `FALSE`) which tells whether or not the number of characters used to store the changing numerical part is variable or fixed. e.g. `file_1.dat` to `file_100.dat` has a variable number of characters, whereas `file_001.dat` to `file_100.dat` has a fixed number of characters.
- `##NUM_CHARS` : When `##VARIABLE` is `FALSE` this is an integer value which is the number of characters used to store the changing numerical part of the file names.
- `##POSTFIX` : This is a character string with any non-changing part of the file names after the changing numerical part, but before any file extension. (Often this will be blank.)
- `##EXTENSION` : This is a character string with any non-changing part of the file name after, and including the decimal point[20].

e.g. If `#FILE_1ST` equals `/users/a/smith/data/poly_001.dat` and `#FILE_LAST` equals `/users/a/smith/data/poly_180.dat` then:

```
Main menu: ENTER COMMAND [PLOT DATA]: DEDUCE
ENTER NAME OF STARTING FILE [fit2d_999.f2d]: #FILE_1ST
ENTER NAME OF END FILE [fit2d_001.f2d]: #FILE_LAST
Main menu: ENTER COMMAND [CONCATENATION]: LIST VARIABLES
##PREFIX = /users/a/smith/data/poly_
##START = 1
##END = 180
```

[20]Normally, a "file extension" does not include the decimal point i.e. is the characters after the decimal point, however, defining the "extension" with or without the decimal point allows more flexibility.

```
##STEP = 1
##VARIABLE = FALSE
##NUM_CHARS = 3
##POSTFIX =
##EXTENSION = .dat
```

This command is useful for creating macros which work with file series.

15.26 DEFINE VARIABLE

Internal program variables may be defined using the `DEFINE VARIABLE` command. This is similar to the Unix 'alias' command, or the VMS 'DEFINE' or 'ASSIGN' commands. A "token" can be defined as a variable and given a data type and a value. The variable names, data types, and "values" are stored in a variable translation table. Once defined, whenever the variable is encountered, it will be replaced by its "value", prior to further input processing. The variable must be separated from other characters by one or more spaces, a comma, or a <Tab>.

This is very powerful, but also very dangerous, and potentially very confusing. Users are recommended to only use unlikely character strings for variables e.g. Start every variable with a hash sign (#). Certain commands define variables and values; these all start with a double hash (##).

Re-defining an existing variable will overwrite its previous value. If the data type changes a warning message will be output. Variables may be removed from the variable translation table by using the `UN-DEFINE VARIABLE` command (see Section 15.119, Page 172). A list of currently defined variables, types, and values may be output using the `LIST VARIABLES` command ((see Section 15.54, Page 142).

Variables may be used as a manner of "parametrising" a macro. By pre-defining variables for file names, variables may be used within a macro. By re-defining the variables with different file names the macro may be re-run on different files. Variables are further explained in Section 29, Page 251 and in particular an example is given of defining variables for "parametrising" a macro in Section 29.4, Page 256.

Here is an simple example of setting the variable `#VALUE` to store a floating point real value will be value of 0.7071. It may be noticed that the choice of data types is unique in the first character so single letters can be entered to set the data type.

```
Main menu: ENTER COMMAND [INPUT DATA]:define
ENTER NAME OF VARIABLE [#IN]:#VALUE
ENTER DATA TYPE OF VARIABLE:?
BOOLEAN VALUE: (Same as ``LOGICAL (BOOLEAN) VALUE'')
CHARACTER STRING VALUE: Alpha-numeric character string value
FLOATING POINT (REAL) VALUE: Floating point value
INTEGER VALUE: Store as an integer value
LOGICAL (BOOLEAN) VALUE: Logical ``true'' or ``false'' value
REAL VALUE: (Same as ``FLOATING POINT (REAL) VALUE'')
STRING VALUE: (Same as ``CHARACTER STRING VALUE'')
```

```
ENTER DATA TYPE OF VARIABLE:r
ENTER FLOATING POINT VALUE:0.7071
```

15.27 DIFFRACTION PATTERN

This allows the diffraction pattern produced by an arbitrary tri-clinic unit cell to be predicted (at present this assumes 360 degree rotation about a rotation axis). For the diffraction pattern to be predicted it is necessary to have defined the geometry of the experimental set-up; this must be done beforehand using `GEOMETRY (EXPERIMENT)` (See Section 15.43, Page 126).

FIT2D calculates the position of the centre of each Bragg peak for a range of *hkl* indices. The user is prompted for the cell parameters and for the range of indices.

The user is allowed the option of creating an overlay diagram of the predicted diffraction pattern on top of an image of the *ADR*.

(At present the unit cell interaxial angles are entered in a form different from the usual crystallographic conventions. If crystallographers are interested in this command the form of input can be changed.)

15.28 DIMENSIONS

Changes the size of program arrays and allows the "memory" arrays and variance arrays to be used or not. The user is prompted for the new size of program arrays. **WARNING:** this operation destroys any data that was already present in the current data array or in the memory. The user should save any data (`OUTPUT DATA`) before performing this operation. If the "memory" arrays are not created many of the menu commands with be disabled. Similarly certain commands require that the variance arrays exist so cannot work without defined variance arrays.

As with all other operations which allocate memory this may fail owing to system limits. If this does fail trying to continue regardless will almost certainly lead to false results or further errors. See Section 3.2, Page 24.

This example shows the program dialogue when the `DIMENSIONS` is used to create program arrays which are 1000 × 1000 elements, with "memory" arrays and variance arrays.

```
Main menu: ENTER COMMAND [INPUT DATA]:dim
ARRAY X DIMENSION (Range: 1 to 100000) [500]:1000
ARRAY Y DIMENSION (Range: 1 to 100000) [500]:1000
CREATE MEMORY [YES]:
CREATE VARIANCE ARRAYS [NO]:yes
```

This example shows the program dialogue when the `DIMENSIONS` is used to create program arrays which are 2000 × 2000 elements, but without "memory" arrays nor variance arrays to reduce computer memory usage.

```
Main menu: ENTER COMMAND [INPUT DATA]:dim
ARRAY X DIMENSION (Range: 1 to 100000) [1000]:2000
ARRAY Y DIMENSION (Range: 1 to 100000) [1000]:2000
CREATE MEMORY [YES]:n
CREATE VARIANCE ARRAYS [YES]:n
```

15.29 DISPLAY LIMITS

When a large region of data is displayed as an image there is an upper limit of the maximum number of pixels to be output by the graphics system to a PostScript file[21]. If necessary, prior to display, the region is automatically re-binned to a smaller size. The data is re-binned by an equal factor in both directions, and scaled by the number of re-binned pixels, so the range is approximately the same as the range of the full data. This both reduces image file size, and makes printing faster.

The upper limit may changed by the user, so that higher or lower resolution images may be printed.

15.30 DIVIDE

Divides element by element the *ADR* of current data by the memory. If a zero is encountered in the memory data, the output element will be set to `-1.7014117e38` and an error message will be output at the end of the operation. If error propagation is being carried out and a zero occurs in a memory element the variance array value is similarly set to `-1.7014117e38`.

(Overflow and underflow may occur if the numbers are sufficiently large or small.)

15.31 END GRAPHICS FILE

Closes the file used for graphics output (after the `PRINT GRAPHICS` command). This allows different files to be used for different graphics. A subsequent `PRINT GRAPHICS` command will prompt for the name of a new file.

(This may be useful for putting different diagrams in different files for later inclusion into a LaTeX, or other type, of document, or for forcing the same question sequence during a macro.)

15.32 ENTROPY

The histogram of pixel intensity values is calculated from -65535 to 65535 units. The raw data should be integer valued for this operation to be meaningful.

Calculates the zero order entropy of the ADR, and the theoretical maximum compression ratio compared to 16 bit raw storage which may be achieved by entropy encoding.

[21]For screen output the screen itself will provide a limit.

(Other information may also be output. This is being used to investigate and develop data compression algorithms.)

15.33 `EXCHANGE`

Exchange current data with the memory contents. The previous data in the memory becomes the current data, and the previous current data becomes the memory data. Axis values, text labels, and the *ADR* are similarly swapped.

(The program in fact exchanges the pointers to various arrays instead of swapping element by element. This makes `EXCHANGE` a fast operation independent of the size of the program array, unlike the `STORE` and `RECALL` commands which must operate element by element.)

15.34 `EXIT`

Exit from program. **FIT2D** prompts for confirmation to avoid accidental loss of work. After confirmation the graphics window should disappear and in a few seconds the normal operation system prompt should appear in the terminal window.

15.35 `FAST IMAGE`

This command was used as a faster method of printing when the GPHIGS graphics system was being used. Now this command is obsolete.

This command will be removed in 1996, so do not use in macros.

15.36 `FILTER`

Not implemented at present. This can be re-implemented if required.

Fourier Filtering of data with sharp cutoff filter.

15.37 `FIT`

This command enters the fitting sub-menu. The fitting sub-menu contains a variety of commands which allow fit models to be defined, minimised, and the inspection of subsequent results.

Fitting is a complicated and much more specialist subject than the other data analysis operations described in this document. The commands within the `FIT` sub-menu are explained in Section 17, Page 208.

15.38 FLIP

The `FLIP` command allows the *ADR* to be reflected about its middle either horizontally or vertically. **FIT2D** gives the following prompt:

```
FLIP LEFT TO RIGHT ("NO" = TOP/BOTTOM)
```

Entering `Y` will flip the image about its vertical middle, and `N` will flip the *ADR* about its horizontal middle. The operation is performed in place so the memory is not affected.

15.39 FONT

Same as `SET FONT`, See Section 15.98, Page 166.

15.40 FUJI LINEARISATION

This command should generally, no longer be used, as the Fuji BAS format is automatically corrected on input. However, the command remains in the case that it may still be useful.

The `FUJI LINEARISATION` command allows Fuji image plate intensities from a BAS Image Plate scanner as read in from a binary file to be converted to a linear scale. At present the images are stored in a log form and must be converted to a linear scale. The conversion formula is:

$$PSL/mm^2 = 100 * (4000/S) * 10^{L(D_x y/2^b - 0.5)} \qquad (2)$$

where S is the sensitivity
L is the latitude
b number of bits
$D_x y$ is the stored integer pixel value

These values are Fuji terms and are given in the `.inf` file.

The user is prompted for the values of the conversion parameters S, L, and b.

NOTE: No error propagation is performed[22].

15.41 FULL REGION

Fast method of making the *ADR* cover all the defined data.

[22]Prior to V9.164 a similar formula was used, with less user control.

15.42 GAUSSIAN

The user can define a 2-D Gaussian function which is added to the data within the current *ADR*. The user defines the peak centre, the maximum intensity, the orientation, and the standard deviation size in pixels of the two axes. (The peak intensity may be negative to allow peaks to be subtracted.)

Here is an example of the program dialogue:

```
Main menu: ENTER COMMAND [INPUT DATA]:gauss
PEAK CENTRE X-COORDINATE [250.000]:
PEAK CENTRE Y-COORDINATE [250.000]:
PEAK MAXIMUM INTENSITY [1000.00]:
ORIENTATION OF FIRST AXIS (Range: -360.000 to 360.000) [0.0]:30
STANDARD DEVIATION WIDTH FOR FIRST AXIS
 (Range: 0.0 to 1.700E+38) [50.0000]:
STANDARD DEVIATION WIDTH FOR SECOND AXIS
 (Range: 0.0 to 1.700E+38) [25.0000]:
```

15.43 GEOMETRY (EXPERIMENT)

A number of commands are only possible if an experimental geometry has been defined e.g. `DIFFRACTION PATTERN` which predicts the centres of Bragg peaks on a detector.

`GEOMETRY (EXPERIMENT)` allows the user to define general aspects of a real or simulated experiment such as energy (or wavelength), sample to detector distance, and pixel sizes.

The centre of the beam on the detector may be defined in one of three different methods:

`AVERAGED GRAPHICAL` The user clicks on a series of pairs of symmetric points (peaks) and the program calculates the beam centre to be the average of the input coordinates.

`ELLIPSE (BEST FIT)` The user clicks on five or more coordinates which should lie on an ellipse. From the coordinates entered the best fit ellipse in the least squares sense is found.

`GRAPHICAL COORDINATE` The user clicks on the estimated centre of the beam (useful for semi-transparent beam-stops).

`KEYBOARD` The user is prompted for keyboard input of the X and Y coordinates of the beam centre. (The user must know the values by some other method.)

`LEAST SQUARES` The user clicks on three or more coordinates which should lie on a circle. From the coordinates entered the best fit circle in the least squares sense is found.

(The `PEEP` command will output angles and corresponding d-spacings once the geometry has been defined.)

15.44 GRID

User can turn on and off horizontal and vertical, coarse and fine grid lines. The coarse grid lines are drawn wherever there are 'large ticks' on the axes, and the fine grid lines are drawn where there are 'small ticks'.

15.45 GUI

This command enters the **FIT2D** "Graphics User Interface" (GUI) from the "keyboard" interface. Thus, even if **FIT2D** is started-up in the "keyboard" interface, all of the GUI commands are still accessible. The only difference to normal GUI operation, is when the GUI is exited. Control returns to the "keyboard" interface main menu without any confirmation.

For information on the GUI interface see Section 4, Page 29.

15.46 HELP

Help text, controlled by a 'pager' allowing backwards and forwards scrolling, searching for keywords, and many other possibilities (See Section 14.4, Page 108). The `HELP` will produce on-line documentation similar to this document.

(Within the `HELP` facility type ? for more information on the pager, and the available paging commands.)

15.47 HISTOGRAM

Calculates the frequency histogram of the pixel values within the current *ADR*.

First the range of pixel values is calculated and displayed, and the user is asked to specify the minimum and maximum values over which the range of values for the frequency histogram is to be defined. Then the number of histogram bins is defined. This can be at most the number of pixels in the first array dimension. These values together define the size of each histogram bin.

Here is an example which defines each histogram bin width to be one intensity unit.

```
Main menu: ENTER COMMAND [Z-SCALE]:histogram

INFO: Active data region minimum value =   .0000000E+00
      Active data region maximum value =   .2820523E+04

HISTOGRAM MINIMUM VALUE (Range: 0.0 to 2.820523E+04) [0.0]:
HISTOGRAM MAXIMUM VALUE (Range: 0.0 to 2.820523E+04)
 [2.820523E+04]:1000
NUMBER OF HISTOGRAM BINS (Range: 1 to 1000) [1000]:
```

The frequency histogram is output in the “memory”.

15.48 I2C

Convert an integer value to a character string and store as a user named program variable of type character string. This is designed especially for use within macros.

The character string can be defined with a fixed number of characters, or a variable number:

```
VARIABLE LENGTH OUTPUT [NO]:
```

and if fixed the number of characters is defined:

```
ENTER NUMBER OF OUTPUT CHARACTERS [3]:
```

The character string is saved as a named program variable:

```
ENTER VARIABLE NAME [#CVALUE]:
```

The following example shows the command being used together with a program variable:

```
Main menu: ENTER COMMAND [PLOT DATA]:Define
ENTER VARIABLE NAME [#CVALUE]:#IVALUE
ENTER DATA TYPE OF VARIABLE:INTEGER
ENTER VARIABLE VALUE [0500]:500
Main menu: ENTER COMMAND [Z-SCALE]:i2c
ENTER INTEGER [1]:#IVALUE
VARIABLE LENGTH OUTPUT [NO]:n
ENTER NUMBER OF OUTPUT CHARACTERS [3]:4
ENTER VARIABLE NAME [#CVALUE]:
Main menu: ENTER COMMAND [CONCATENATION]:list
s #CVALUE = 0500
i #IVALUE = 500
```

15.49 IMAGE

Display of the *ADR* of the current data as a 2-D false colour pixel image with graphical input to change displayed region and set display attributes. This is the same as the graphical user interface **MOVEMENT** command; see Section 5.6, Page 54 for further information.

Figure 3, Page 26 shows an example of the display window complete with the graphical menu choices.

Many of the keyboard interface main menu commands change the behaviour of the displayed graphics:

`ANNOTATION LABELS` Add annotation labels to the graphics (See Section 15.5, Page 111)

`DISPLAY LIMITS` Set the maximum number of pixels which will be used in either the X or the Y-direction to output the image (this only affects the output to file). Larger images, or image regions, will be automatically re-binned to make the number of pixels smaller than the limit. (See Section 15.29, Page 123)

`GRID` Control horizontal and vertical, coarse and fine grid lines on top of image (See Section 15.44, Page 127)

`SET ****` The set commands will change the aspect of the displayed graphical items (See Section 15.91, Page 165 and following `SET` commands)

`TITLE` Change title of image. Section 15.116, Page 172)

`X-AXIS LABEL` Change text for X-axis label (See Section 15.126, Page 174)

`Y-AXIS LABEL` Change text for Y-axis label (See Section 15.127, Page 174)

`Z-AXIS LABEL` Change text for Z-axis (intensity) label (See Section 15.128, Page 174)

`Z-SCALE` Change scaling mode for the Z-scale (intensity). By default the data is always automatically scaled to display the full range of data values within the display region (See Section 15.129, Page 175)

15.50 INFORMATION

Produces information on the internal state of **FIT2D**, such as the present size of program arrays and whether or not current data and memory data are presently defined. If so the current values of the *ADR* are also output.

15.51 INPUT DATA

Input of data from defined file formats into **FIT2D**'s internal data-base. A number of different formats are available and more will be added as appropriate. Different formats have different levels of sophistication (or lack of sophistication) and hence the amount of user input varies. At the time of writing the following formats are available:

`1-D ASCII FREE FORMAT` This allows 1-D data from ASCII text files to be input

`2-D ASCII FREE FORMAT` This allows flexible input of 2-D data from ASCII text files containing lists of pixel values

`BINARY (UNFORMATTED)` This allows unformatted binary data to be input

`BSL FORMAT` This is the Daresbury BSL format based on the Hamburg OTOKO format

`CHIPLOT FORMAT` Simple ASCII X/Y column format data as used by the program CHIPLOT

`COMPRESSED DIFFRACTION DATA` Inputs data which has been stored in the "Compacted Diffraction Data" format

`DIP-2000 (MAC SCIENCE)` This allows a special two byte binary 2500×2500 image to be input

`ESRF DATA FORMAT` Inputs files which have been stored using a limited sub-set of the "ESRF data format" (files from the SAXS end-station on BL-4).

`FIT2D STANDARD FORMAT` This is the standard format which a flexible self describing format

`FUJI BAS-2000` Fuji BAS-2000 image plate scanner format (also BAS-1500)

`GAS 2-D DETECTOR (ESRF)` Direct input of raw format files written on the beam-lines (separate ASCII header file and a binary "histogramming memory" file)

`HAMAMATSU PHOTONICS` `Integer*2` binary data described by a short header. Produced by the Hamamatsu Photonics K.K. C4880 CCD cameras which are used at the Photon Factory for the X-ray image intensifier/CCD read-out detectors.

`IMAGEQUANT` For input of data from the Molecular Dynamics Imaging plate PC systems (a TIFF based format file)

`MAR RESEARCH FORMAT` MarResearch image plate system format.

`NEW MAR CODE` Same as `MAR RESEARCH FORMAT`

`PDS FORMAT` The "Powder Diffraction Standard" format [24]

`PHOTOMETRICS CCD FORMAT` `Integer*2` binary data described by a short header. Produced by the X-ray image intensifier/CCD read-out detector system[23].

`PMC FORMAT` PhotoMetrics Compressed XRII/CCD data. This is data from the Photometrics CCD camera, but which has been compressed using the program `pmi2pmc`.

`PRINCETON CCD FORMAT` `Integer*2`, `Integer*4`, or `IEEE REAL*4` binary data described by a short header. Produced by the X-ray image intensifier/CCD read-out detector system. (Very similar but slightly different format to the Photometrics format.)

`TIFF` Input of **simple** 1 and 2-byte per pixel TIFF files

`UNKNOWN` Attempt to deduce format on non-compressed data

`USER INTENSITIES` Interactive entry of data values

`WESS FORMAT` For input of film densitometer data (Unformatted unsigned byte data)

Some of these formats are described in greater detail below:

[23]Note: For a short time this input option was called `XRII`, until it was realised that there were two different file formats produced by the two different CCD cameras used with the X-ray image intensifier systems.

15.51.1 1-D ASCII FREE FORMAT

This allows 1-D data to be input from an ASCII file with a large amount of flexibility in treating column information.

The program prompts for the file name:

```
Enter name of file containing 1-D data
FILE NAME [fe_2.scan]:
```

A sample of the start of the file is output to help the user chose the correct columns for input of the angular and intensity data:

```
Sample of start of the input file:
 -30.0000    39689        0           0
 -28.0000    40200        0           0
 -26.0000    40682        0           0
 -24.0000    40892        0           0
 -22.0000    40914        0           0
 -20.0000    40569        0           0
```

A variety of questions allow the user to specify the starting line, and character position for the data, and which columns of data to use for X-coordinates and the Y-coordinates. By entering `0` for the number of coordinates to be input the program automatically inputs as many coordinates as possible. Alternatively the number of coordinates to input may be specified.

Here is an example of the program dialogue:

```
Main menu: ENTER COMMAND [INPUT DATA]:
FILE FORMAT [FIT2D STANDARD FORMAT]:1-D
Enter name of file containing 1-D data
FILE NAME [source.scan]:/scratch/EXPERIMENTS/bl10/cell_1_edit.scan
Sample of start of the input file:
  #  DETECTOR vct6 Ch5  Monitor    Seconds
  0  -27.0000     7394        0          45
  1  -26.0000     7491        0          45
  2  -25.0000     7672        0          45
  3  -24.0000     7680        0          45
  4  -23.0000     7675        0          45
TYPE OF DATA FORMAT [VERTICAL COLUMNS]:
NUMBER OF LINES TO IGNORE (Range: 0 to 1000) [0]:1
NUMBER OF CHARACTERS TO IGNORE (Range: 0 to 255) [0]:
NUMBER OF COORDINATES (Range: 0 to 500) [500]:0
Columns of numbers for data-set  1
COLUMN NUMBER FOR X-COORDINATES (Range: 0 to 80) [1]:2
COLUMN NUMBER FOR Y-COORDINATES (Range: 1 to 80) [3]:
INFO:   56 X/Y coordinates per data-set have been found
```

15.51.2 2-D ASCII FREE FORMAT

This allows a 2-D image to be defined and input from numbers in an ASCII file. The numbers may be input in free format, so a variety of column formats may be input.

The numbers may be separated by one of more spaces and <Tabs>, or by commas, spaces, and <Tabs>. Most common forms of scientific notation are supported.

The program prompts for the file name:

```
Enter name of file containing 1-D data
FILE NAME [file.ascii]:
```

The size of the image to be defined is then input:

```
X NUMBER PIXELS (Range: 1 to 10000000) [30]:
Y NUMBER PIXELS (Range: 1 to 10000000) [30]:
```

This defines the width and the height of the image to be input.

A sample of the start of the file is output to help the user chose the correct line to start input:

```
Sample of start of the input file:
Test File for 2-D input
1.0 2.0 3.0
3.0 2.0
3.0
4.0
5.9
```

You are asked how many lines to ignore, so that a short header section may be "jumped" over. In the above example, the first line is a text line, so should be ignored. e.g.

```
NUMBER OF LINES TO IGNORE (Range: 0 to 1000) [0]: 1
```

Every "number" which can be converted will be input and used to define the next pixel value. Thus, the number of values per line may be variable, as in the above example.

15.51.3 BINARY (UNFORMATTED)

.

Unformatted binary data can be input provided the user knows the size of the data image. Presently single byte integer "`Integer*2`", "`Integer*4`" and "`Real*4`" data can be input using this option. The single byte, "`Integer*2`" and "`Integer*4`" data may be signed or unsigned and either byte order may be treated. **FIT2D** prompts for data size and options for byte order and sign/unsigned data.

If the size of the data is not known it may be possible to set a small value for the Y-direction and vary the X-direction until the image wraps properly (provided of course it is possible to identify a true image). Similarly using the wrong byte order should produce very strange results when the image is viewed.

(It should be noted that the GUI input form has the additional ability to be able to specify an offset from the start of the file in order to jump over known length headers.)

The following example shows a a binary image of 1242 × 1152 pixels being input in 2-byte unsigned integers, without byte swapping:

```
Main menu: ENTER COMMAND [INPUT DATA]:input
FILE FORMAT [PRINCETON CCD FORMAT]:bin
INPUT FILE NAME [no_data.dat]:grid.bin
WARNING: No '.info' file, so unknown image size, and pixel sizes
         (Pixel sizes defaulted to 100 x 100 microns.)
X NUMBER PIXELS (Range: 1 to 10000000) [1242]:
Y NUMBER PIXELS (Range: 1 to 10000000) [1152]:
DATA TYPE [INTEGER (2-BYTE)]:?
4-BYTE INTEGER: Signed or unsigned, either byte order
BYTE VALUES: Single signed or unsigned integers
INTEGER (2-BYTE): Signed or unsigned either byte order
REAL (4-BYTE IEEE): Floating point real numbers
DATA TYPE [INTEGER (2-BYTE)]:
PERFORM BYTE SWAPPING [NO]:n
SIGNED DATA [NO]:n
Main menu: ENTER COMMAND [IMAGE]:plo
```

15.51.4 BSL FORMAT

The BSL format is essentially the same as the OTOKO format developed at Hamburg. It consists of an ASCII header file and a number of binary image data files. The images are stored in floating point reals. (This may cause problems when files are moved between non-IEEE and IEEE float point reals systems.)

This may also cause problems between little-endian and big-endian systems since the endianess of the stored data is not recorded.

FIT2D will open the header file and obtain information concerning the images which may be input. The user will be prompted for the number of the image to input.

15.51.5 CHIPLOT FORMAT

The standard χ**PLOT** data format is a very simple format allowing input of one or more data-sets together with title and axis labels, and accompanying text. Error estimates may be defined.

The file should be a formatted variable length record ASCII file. Such a file may be created with an editor, or a BASIC program using `PRINT #` statements, or a Fortran program using formatted `WRITE` statements.

The minimum file format is as follows:

Title and Axes Labels The first three lines contain character strings (i.e. text) that will be used to label the graph. The first line is the title of the graph, the second is the label to be written for the X-axis, and the third line is the label for the Y-axis.

Number of Data points and Data-Sets The fourth line specifies how many data points are present in each of the data-sets, and how many data-sets to input. This line should contain two integers separated by a one or more spaces or a comma. If the second integer is missing it is assumed that only one curve is required.

The Data Points The next '*' lines of the file should contain the data to be plotted; where '*' is the number of data points specified in line four. Each line should contain the value for the X-coordinate, and the values of the Y-coordinates for the different curves. (Note: All the curves share the same X-coordinates.) e.g. If only one curve is to be drawn each line should have two values, the values being separated by a blank space or a comma. Such a line may be as follows:

```
1 30.3
```

or equivalent

```
1.0, 3.03e1
```

Here the lines represents the coordinate (1.0,30.3). If the number does not contain a decimal point it is assumed to be whole number.

The following example shows a file which produces a single curve. Note on line four of the file the number of curves is not specified so one is assumed.

```
Figure 1. SINGLE DATA-SET
X AXIS UNITS
Y AXIS UNITS
10
1 5e-2
2., 5.9e-2
3. 6.7e-2
4 6.8e-2
5 6.6e-2
6 5e-2
7 4.1e-2
8 2e-2
9 0.6e-2
10 -2.6e-2
```

ASCII X/Y column output from other programs can be easily converted to this format for input.

15.51.6 `DIP-2000 (Mac Science)`

The DIP-2000 Mac Science scanner produces images of 2500×2500 pixels with each pixel intensity stored in two bytes. As the scanner uses two different photo-multiplier tubes the "gain" is different for different pixels.

The intensity values are decoded according to the following scheme:

$If\ (I_e \geq 0): \quad I_d = I_e$

$If\ (I_e < 0): \quad I_d = -I_e * 256 + 32768$

Where I_e are the intensity values of the input encoded pixels, and I_d are the intensities of the corrected values.

The user is asked for the name of the file to input, and whether byte swapping is necessary. For a Sun, Silicon Graphics, or HP workstation byte swapping will generally not be necessary.

(Note: The orientation of the image is not necessarily correct at present. When the input orientation is correctly known, this input may be adjusted to conform with the normal ESRF image display convention (looking from sample towards the detector.)

15.51.7 `‘‘ESRF Data Format’’`

FIT2D now contains its own code to input a **limited sub-set** of the so called "ESRF data format". Unfortunately the format is very poorly defined, and in practice is more defined by the software which writes the data, than by the documentation. The two being completely different in a number of fundamental aspects. This option should only be used on the understanding that it is unsupported (unsupportable).

This routine should hopefully be able to input files produced by the SAXS end-station of ESRF Beam-line 4. However, some older files may fail owing to 4-byte floating point numbers being written across word boundaries. If this happens a warning message will explain the problem, and inform the user of the number of spaces which should be added to the header to make the file conform to word boundaries. These should be added at the end of the header section before the curly bracket (}).

Only files with all the header information occurring before the binary image data are "supported", and no data compression is supported.

As one file can potentially contain many images, the user is prompted for the number of the required image.

15.51.8 `FUJI BAS-2000`

The Fuji BAS-2000 image plate scanners produce an ASCII header file and a binary data file. This option will read the header file and find out the size of the image and parameters necessary to convert the stored logarithmic scaled data to a linear scale. The binary data is input and automatically converted to the linear scale.

This input option will also work for BAS-1500 scanners.

15.51.9 `GAS 2-D DETECTOR (ESRF)`

WARNING: The data format used to output data from the ESRF 2-D Gas-filled detectors is poorly defined and may change. This option is believed to work, but no guarantee is given that it is either presently correct, or will continue to be correct. Please check input data carefully. If problems are encountered, or the format changes FIT2D will be modified accordingly.

This format is used to input data directly as produced by the 2-D Gas-filled multi-wire proportional counter detectors used at the ESRF (the detectors themselves are produced by André Gabriel at the EMBL Grenoble). The data is output within two files: an ASCII header file describing the size, type of data, and the number of images held within a binary image file. The binary image file is general referred to as the "histogramming memory". This may contain many images.

The user is prompted for the file name of the header file:

```
INPUT HEADER FILE NAME [wes019-004]:wes019-004
```

The header file is checked to exist and be a valid header file. **FIT2D** outputs the number of images in the file, the size of the images and the number of bits used to store each pixel value. The program array size must be big enough to input an entire image.

If more than one image is contained in the binary image file **FIT2D** prompts for the number of the image to be input. The choice of byte-swapping and of inputing signed or unsigned data is given. Since the data is written by a Motorola 68000 series processor the raw data is written in big endian byte order. Byte-swapping should not be necessary on HP, Sun, and Silicon Graphics workstations, but should be necessary on Vax workstations (and on PC's).

From the header file name the name of the binary image file is constructed (`hm` is concatenated to the header file name). The data is input in the format defined in the header file.

This is an example of the **FIT2D** log for the input of a gas-filled detector image:

```
Main menu: ENTER COMMAND [INPUT DATA]:
FILE FORMAT [FIT2D STANDARD FORMAT]:gas
INPUT HEADER FILE NAME [wes019-004]:wes019-004
INFO: The file contains one image (frame) of   1024 *   1024 pixels.
      Each pixel is stored using 16 bits.
PERFORM BYTE SWAPPING [NO]:?
Enter ``YES'' to swap the byte order on input. Normally byte swapping
should not be necessary for HP, Sun, and Silicon Graphics
workstations. It will normally be necessary for VAX workstations.
PERFORM BYTE SWAPPING [NO]:
SIGNED DATA [NO]:
```

15.51.10 HAMAMATSU PHOTONICS

The `HAMAMATSU PHOTONICS` format inputs data from the Hamamatsu Photonics K.K. C4880 CCD cameras which are used at the Photon Factory for the X-ray image intensifier/CCD read-out detectors. This is an `Integer*2` binary data format described by a short header. Any byte swapping which needs to be performed will be carried out automatically.

If the image is too big for the current program arrays, then a warning message will be produced and only part of the image will be input.

15.51.11 IMAGEQUANT

This is the file format used by the Molecular Dynamics Imaging Plate scanner. It is a TIFF based file format, but does not strictly follow the TIFF standard. The Molecular Dynamics scanner produces images from A4 imaging plates which are 1152×1482 pixels for the coarse resolution scan (176 μm), and 2304×2964 pixels for the fine resolution scan (88 μm).

The Molecular Dynamics scanner produces and displays the image as seen from behind the detector, but the ESRF standard is for images as seen from the sample. Thus the image is left to right reversed when view by **FIT2D** as compared the PC.

FIT2D offers the possibility to input only a region of the total image, and the possibility to re-bin pixels on input to make the very large images smaller and more manageable. An example prompt and user input is as follows:

```
DATA FILE NAME [lysip1.gel]:data.gel
X REBIN NUMBER (Range: 1 to 1152) [1]:2
Y REBIN NUMBER (Range: 1 to 1482) [1]:2
LEFT-HAND PIXEL OF IMAGE REGION [1]:500
LOWER PIXEL OF IMAGE REGION [1]:500
RIGHT-HAND PIXEL OF IMAGE REGION (Range: 500 to 1152) [1152]:1000
UPPER PIXEL OF IMAGE REGION (Range: 500 to 1482) [1482]:1000
INFO: Full image size =    1152 *    1482 pixels
```

15.51.12 MAR RESEARCH FORMAT / NEW MAR CODE

The MarResearch on-line image plate scanners produce a number of different file formats. The raw output from the scanner is a spiral read-out scan. This is not suitable for input to **FIT2D**, but usually this is immediately transformed to a Cartesian raster file.

The original file format contained a fixed length header of one record, followed by the data in a simple binary format, and maybe a number of overload records at the end of the file. **FIT2D** will input this format including information such as the sample to detector distance from the header and the overload records.

A new format now exists and there is the possibility to store compressed data. From **FIT2D** V9.114 support exists to input these new files, including the compressed data.

(Note: At present the orientation of the images may still be wrong, since there appears to be a complicated historical scheme where different sized images are stored in a different manner.)

15.51.13 PMC FORMAT

This is data from the Photometrics CCD camera, but which has been compressed using the program `pmi2pmc`. `pmi2pmc` is a program which can run on a PC or workstation which keeps the same header (presently the first 172 bytes) but compresses the image data according to a simple compression algorithm,

with the value of the first image pixel being stored immediately after the header bytes.

`pmi2pmc` is designed to allow loss-less data compression at source prior to data transfer over ethernet. A typical transfer time is 20 seconds per uncompressed image, and the compressed images may be nearly one third of the size of the raw data. Thus, the transfer may be much faster if the data can be compressed on the PC. Once compressed it is faster to read it in compressed, so **FIT2D** has this input option.

`pmi2pmc` does not change the byte raster order, which is normally from left to right as seen by a camera-man, and from top to bottom. On input **FIT2D** will correct the image so that it is seen from the crystal side and the byte order is from bottom to top.

The raw data can suffer from appreciable spatial distortion and non-uniformity of response. Commands within the `CALIBRATION` sub-menu exist to help calibrate and correct these effects (see Section 16, Page 176).

(Note: At present the orientation of the CCD camera can change which will also change the sense of the output image. Soon the mechanical holder should stop this uncertainty.)

15.51.14 PRINCETON CCD FORMAT

This is the file format produced by the Princeton CCD camera; one of three CCD cameras used with the ESRF X-ray Image Intensifier systems. It is a simple binary format with a 4100 byte header followed by signed or unsigned two byte integer data, signed 4-byte integer data, or IEEE floating point real data. There is an old version of the format and a newer, similar but slightly different version. Both are automatically recognised and input (V9.104). All input types are now supported (V9.136). The full size image of the present ESRF systems is 1242×1152 pixels, but smaller images may be produced (the header contains the size information).

The Princeton camera and software produces and displays the image as seen from behind the detector, but the ESRF standard is for images as seen from the sample. Thus the image is left to right reversed when view by **FIT2D** as compared the PC. The orientation may be changed by the PC software and by the actual orientation of the CCD camera. **FIT2D** will display up with the same sense as up in the PC images, but this may or may not be up in the sense of the experiment.

The raw data can suffer from appreciable spatial distortion and non-uniformity of response. Commands within the `CALIBRATION` sub-menu exist to help calibrate and correct these effects (see Section 16, Page 176).

If more than one image has been stored in a file, **FIT2D** will output a prompt asking for the required image (V7.32), e.g.:

```
IMAGE NUMBER (Range: 1 to 3) [1]:
```

15.51.15 TIFF

This allows input of **simple** TIFF files. The TIFF "standard" allows many different options, many of which are **not** supported. Simple 16 bit per pixel and 8 bit per pixel formats are presently supported. Note: no image compression is presently supported. This format has been added primarily to allow input of data from the EMBL Drum scanner which is used for neutron diffraction at the ILL.

NOTE: This option should not be used for input of data from the Molecular Dynamics 400E scanner. The option `IMAGEQUANT` should be used instead (see Section 15.51.11, Page 137).

FIT2D offers the possibility to input only a region of the total image, and the possibility to re-bin pixels on input to make the very large images smaller and more manageable. An example prompt and user input is as follows:

```
DATA FILE NAME [test.tiff]:data.tiff
INFO: Full image size =    4000 *    4000 pixels
```

```
X REBIN NUMBER (Range: 1 to 4000) [1]:2
Y REBIN NUMBER (Range: 1 to 4000) [1]:2
LEFT-HAND PIXEL OF IMAGE REGION [1]:500
LOWER PIXEL OF IMAGE REGION [1]:500
RIGHT-HAND PIXEL OF IMAGE REGION (Range: 500 to 4000) [4000]:3000
UPPER PIXEL OF IMAGE REGION (Range: 500 to 4000) [4000]:3000
```

Note: If TIFF files need to be input into **FIT2D** which use options not presently supported, please inform me and I will try to cater for them.

15.51.16 UNKNOWN

This option will try to deduce the data format purely from byte to byte and then pixel to pixel correlations within the data. This **cannot work with compressed data.**

With raw non-compressed data the bytes should should regular correlations so that it is possible to estimate with a good success rate the number of bytes used in each pixel.

Having determined the number of bytes in a pixel it is relatively easy for integer data to deduce the byte ordering since the lower significant bytes tend to change more rapidly than the high significant bytes.

The number of pixels in a row can then be determined by peaks in the 1-D linear autocorrelation function, although very structured images can lead to false results.

Finally visual inspection can determine the start of the image.

15.51.17 USER INTENSITIES

Data values may be input interactively to form a 1-D line of data values. This can be very useful for quickly calculating statistical quantities using the `STATISTICS` command (See Section 15.109, Page 170) after the data has been defined. Data values can also be easily entered and plotted using the `PLOT` command (See Section 15.69, Page 153).

The program prompts continual for more data values; the user must use "user escape" to terminate entry of data values. "User escape" is two backslashes (\\)[24]. After all the data values have been entered the user may change the entered values to correct mistakes. A typical prompt and user input session may be as follows:

```
INFO: Continue entering numbers as required, then use "USER ESCAPE"
INFO: "USER ESCAPE" is double backslash (\\)
ENTER DATA VALUE:10.5
ENTER DATA VALUE:15
```

[24]Previously <CONTROL>⊗D (Unix) or <CONTROL>⊗Z (VMS) was used for "user escape", but this has had to be changed owing to the usage of GNU Readline which provides command history recall and file name completion.

```
ENTER DATA VALUE:1.1e2
ENTER DATA VALUE:.45
ENTER DATA VALUE:13.6
ENTER DATA VALUE:\\
DATA VALUE TO CHANGE (0 = quit) (Range: 0 to 5) [0]:4
ENTER NEW DATA VALUE [0.450000]:45.0
DATA VALUE TO CHANGE (0 = quit) (Range: 0 to 5) [5]:0
```

Where "user escape" was entered for the last `ENTER DATA VALUE` prompt.

15.52 INTERNAL MEMORY

The `INTERNAL MEMORY` command allows the *ADR* of a number of different arrays or regions of arrays to be stored internally (V7.27). (This is in addition to the program "memory"). This command is powerful and **dangerous** since each extra array that is stored requires more virtual memory from the operating system. Thus, the total demands for virtual memory grow and it may easily be possible to reach the system limits. If this happens a warning message should be produced, but you should be able to carry on using **FIT2D**, but without the extra saved array. However excessive memory usage should be avoided as it can interfere with other users and the operating system.

Note: This "internal memory" operation only saves and recovers the current *ADR* and not whole of the data arrays. If variance arrays are defined these will also be saved and recovered.

The user is prompted whether to save the current data, or to recover previously saved data. e.g.

```
INFO: There are currently  0 active data regions stored internally
SAVE ACTIVE DATA REGION (‘‘NO’’: TO RECOVER) [YES]: ?
You are given the choice of saving the current active data region (ADR)
or recovering a previously saved region (if one exists). Enter ‘‘YES’’ to
save the current ADR within internal program memory, or ‘‘NO’’ to recover
a previously saved ADR.  Note:  Each time  an  ADR is saved the program
needs to allocate more dynamic memory,  so  this command should be used
with care,  and  may  fail if  the computer system cannot allocate more
memory.
SAVE ACTIVE DATA REGION (‘‘NO’’: TO RECOVER) [YES]:
```

If the current *ADR* is to be saved, the user is prompted for the number of the memory store to use. If none have been used there is no choice, however if memory stores have already been defined there is a choice between overwriting the a previous store or using a new one. If possible use an existing store to avoid using even more virtual memory. e.g.

```
INTERNAL STORE NUMBER (Range: 1 to 1) [1]:?
Enter number of internal memory  to  use  to  store  active data region.
By default a new memory will be used,  unless they are  all  being used,
however each time a new memory is used more virtual memory is necessary.
INTERNAL STORE NUMBER (Range: 1 to 1) [1]:
```

When data has been stored, a list of the titles of each stored *ADR* will be output when the `INTERNAL MEMORY` command is issued. e.g.

```
INFO: There are currently  2 active data regions stored internally
      1: Simulated Data
      2: A sub-region of the data
```

To recover a previously saved region enter `NO` to the `SAVE ACTIVE DATA REGION` prompt and specify the memory store from which to recover data. e.g.

```
SAVE ACTIVE DATA REGION (‘‘NO’’: TO RECOVER) [NO]:
INTERNAL STORE NUMBER (Range: 1 to 2) [2]:?
Enter number of internal memory to recover a previously stored active
data region.
INTERNAL STORE NUMBER (Range: 1 to 2) [2]:1
```

The data is copied from the memory store into the program array in the same *ADR* as was used to store the data. Other data which may be present in the program array is not affected. The data in the memory store is still present.

(Note: At present memory stores, once created, cannot be completed destroyed, but virtual memory may be recovered by defining very small *ADR*'s and overwriting the previously stored *ADR* with the new very small one.)

15.53 LINEARISE FILM

Corrects the non-linearity response of film data. The correction applied is a quadratic which has been found to agree with experimental results. It is necessary to know the type of film, and the maximum optical density measured by the micro-densitometer.

15.54 LIST VARIABLES

Outputs to the terminal (and to a log file if open) a list of the currently defined program variables, data types, and their values. If no variables are currently defined, an information message to this effect is output.

e.g. After the `STATISTICS` command has been used, a number of program variables are automatically defined, so the following output may be obtained:

```
Main menu: ENTER COMMAND [CLOSE LOG]:list
```

```
r ##MINIMUM =   .8627196E+02
r ##MAXIMUM =   .9997000E+03
r ##MEAN =   .5117823E+03
r ##RMS =   .5739011E+03
r ##SIGMA =   .2597079E+03
r ##SKEWNESS =   .1288433E+00
r ##TOTAL =   .5117823E+07
```

The data type is given first. The following letters are used with the corresponding meaning for the data type of the value:

i Integer value

l Logical (boolean) value

r Floating point real value

s Character string value

u Data type unknown (e.g. enter with `-sym` from the command-line

* Invalid variable (This should be rarely seen, but can occur if there is no more space available for storing string values.)

15.55 LOGARITHM

Take logarithm base 10 of all elements in the *ADR*.

If any of the elements are zero or negative the user will be prompted for the lower threshold value to be used to set these elements. If variances exist they will be set to the absolute value of the input lower threshold.

15.56 MACRO

Allows a previously defined macro file to be run. The user is prompted for the file name of the macro file. The manner to define and use macros are explained in Section 19, Page 225. (Also see `START MACRO` Section 15.108, Page 169).

FIT2D will try to follow the instructions in the file until the end of the file. The file replaces the normal keyboard and graphics cursor input.

15.57 MEDIAN FILTER

Filters data within *ADR* by taking median value within a user defined window for each data point. The output is in the memory.

Median filtering is a non-linear filtering technique which is sometimes useful as it can preserve sharp features (e.g. lines) in an image whilst filtering noise.

The disadvantage is that it is difficult to treat analytically the effect of a median filter. **There is no error propagation.**

15.58 MESSAGE

The MESSAGE command allows user messages to be included in interactive macros. Lines of text are defined and finished with the user escape (\\). The message will appear on the graphics system. This is normally used within macros.

Here is a very simple example of the MESSAGE command:

```
Main menu: ENTER COMMAND [DEFINE VARIABLE]:message
INFO: Enter message text or user escape (\\) to exit
TEXT:
TEXT:Hello World
TEXT:
TEXT:\\
```

The line “Hello World” will appear in the middle of the graphics screen.

15.59 MOVE/ROTATE

The user can specify a rotation about a fixed point **followed** by a translation. The user is also prompted for the output region of the memory where the data may be output. The pixel values within this region will initially be set to zero and incremented by any input pixels which are transformed to lie on top of output pixels. The output is in the memory.

Alternatively it is possible to specify the coordinates of two independent points on the input data and two corresponding output points. The program will use these to calculate a required rotation and translation. For this to be correct the distance between the two input and two output coordinates should ideally be the same. If the distances are different the rotation angle will be calculated to align exactly the lines, and the translation will be calculated so that the centre point between each pair of points is aligned i.e. the discrepancy will be equal for two points.

15.60 MULTIPLY

Multiply element by element each element of the current data *ADR* with the corresponding element in the memory. The memory *ADR* must be defined for every element in the current data *ADR* otherwise a warning message will be output.

15.61 NORMALISE

Divides element by element each element of the current data *ADR* by the initial maximum element in the *ADR* (provided the maximum is greater than 1.0^{-19})[25]. If the maximum is below this value, zero, or negative, a warning message will be issued and the data will be left unchanged. If a variance array exists error propagation will be performed.

This operation is performed in-place in the current data arrays. The *ADR* of the current data is replaced by the normalised values.

15.62 OFFSET/SCALE

When two datasets have been taken of essentially the same experiment, but perhaps with a difference of scaling, this command tries to estimate the scale factor between the two data sets. This process is complicated by signal independent noise such as film fog for film data and electronic noise in many other detectors.

`OFFSET/SCALE` estimates a signal independent offset and scale factor between two images (one in the memory). For this to be possible there must be a range of intensity values in both images.

(**NOTE:** This operation is presently under development. Tests show the results to be correct for non-noisy images, but the results in the presence of noise are not very stable. Hopefully the stability can be improved.)

15.63 OPEN LOG

Allows a file to be opened to contain a log of the program until the `CLOSE LOG` command is issued, or the program is terminated. The user is prompted for the name of file to be created to contain the log. The log file contains all the input and output as seen on the screen. Producing log files can be particularly useful if problems are encountered, or bugs are suspected. The log file can be an invaluable record of the analysis that a set of data has undergone.

After the `CLOSE LOG` command or the program has been exited the log file may be edited and the useful information extracted for many purposes e.g. tables of figures may be entered to a spread-sheet program for further manipulation and display.

(The `OPEN LOG` command cannot be used for "replaying" an analysis, but this can be achieved by creating a macro, see Section 19, Page 225.)

[25]This value is chosen to stop overflow and underflow when performing the normalisation. For the error propagation it is necessary to divide by the square of the maximum value.

15.64 OUTPUT DATA

Output data in *ADR* to named data file in one of a choice of data formats. More output formats will be added as is appropriate. At present the following output formats or commands are available:

`3CAM` 1-D powder diffraction format for input to CERIUS

`4-BYTE INTEGERS` Binary integer*4 output of active data region

`BINARY (UNFORMATTED)` Simple dump of active data region

`BSL FORMAT` Daresbury SAXS format, based on Hamburg OTOKO format

`CBF` Prototype output of the proposed IUCr "Crystallographic Binary File" Format

`CHIPLOT` 1-D row or column output for X/Y graph plotting

`COMPRESSED DIFFRACTION DATA` Compacted diffraction data format

`DENZO MAR FORMAT` Experimental output format for input to DENZO

`DUMP` (Same as "BINARY (UNFORMATTED)"; backwards compatibility)

`FIT2D STANDARD FORMAT` Self describing readable binary

`GSAS` GSAS powder diffraction format

`HEADER FILES` ASCII header files are created describing information contained in the `BINARY (UNFORMATTED)` output files (this is the default behaviour)

`MCA FORMAT` ASCII 8 columns per line g10.3 data only

`NO HEADER FILES` Do not create ASCII header files when the `BINARY (UNFORMATTED)` format is used

`SPREAD SHEET` ASCII, one line of ADR in one record

`TIFF INTEGERS` Simple 1 or 2-byte output (no compression)

Each output option is described more fully in the following sub-sections.

15.64.1 3CAM

This is simple ASCII text format for the output of 1-D data. It is the recommended input format for inputing data into the Cerius^{TM} software suite.

The user is asked for the name of the output file, and whether to output rows, or columns, and which row or column to output. This allows a single line from a 2-D image to be output. e.g.

```
Enter name of output file
FILE NAME [cerius.3cam]:
OUTPUT ROWS [YES]:
NUMBER OF ROW TO OUTPUT (Range: 1 to 1) [1]:
```

15.64.2 BINARY (UNFORMATTED)

This is an unformatted binary "dump" of the *ADR* of the current data. At present only 2-byte per pixel unsigned integers are available. This can be used to output data in a suitable format for input to MOSFLM , DENZO [26], INTLAUE, IDL, or many other programs[26]. The format can be read into **FIT2D** using the BINARY (UNFORMATTED) choice from the $\widehat{\text{I}}$NPUT DATA command. Whilst flexible this format has the disadvantage that the output file does not contain the information as to the size of the image. To help **FIT2D** can output a separate "header information" file, and the number of pixels in the X and the Y-directions is output to the user.

The data is output in a 2-D raster starting from the bottom left-hand pixel. X (horizontal) is the fastest changing pixel index. Going from left to right. Y (vertical) changes more slowly. After all the pixel on one row have been output, the row number is incremented and pixels on the next row are output. The rows are output from bottom to top. The top right-hand pixel is the last to be output.

By default **FIT2D** will create a "header" with the same base name as the binary output file, but with .info appended to the base name. This file is human "readable" and contains various information telling not only the size of the data, but also information on the manner in which the data has been output. If header files are not wanted this facility may be disabled by typing NO HEADER FILES to the FILE FORMAT prompt. It may be re-enabled by typing HEADER FILES to the FILE FORMAT prompt.

The user is prompted with various questions by this option. The explanation of the questions is best demonstrated by an example.

The following text is the program input/output sequence generated by **FIT2D** when the current active data region contains 20 by 20 pixels.

```
Main menu: ENTER COMMAND [INPUT DATA]:output
FILE FORMAT [FIT2D STANDARD FORMAT]:binary
BINARY FILE NAME [fit2d.bin]:
INFO: Data region is     20 *     20 pixels
RECORD LENGTH (BYTES) (Range: 1 to 6000) [40]:?
Length of fixed length records in file in bytes
RECORD LENGTH (BYTES) (Range: 1 to 6000) [40]:
FIRST RECORD FOR OUTPUT (Range: 1 to 100000) [1]:?
Number of first record to write data from array
FIRST RECORD FOR OUTPUT (Range: 1 to 100000) [1]:
BIG ENDIAN FORMAT INTEGERS [NO]:?
YES for big endian integer (Sun/HP/SG), NO for little endian (VAX, PC, MAR)
BIG ENDIAN FORMAT INTEGERS [NO]:yes
INFO: Data minimum value =    16.43577
INFO: Data maximum value =    991.7908
```

[26]MOSFLM is a program for the initial indexing and integration of protein crystallography diffraction patterns written by Andrew Leslie at Cambridge, England. DENZO is also a protein crystallography indexing and integration program, written by Zbyszek Otwinowski, now at the University of Texas, USA. INTLAUE is written by various people and is available from John W Campbell, at DRAL, England. IDL is a commercial package available for those with money.

```
LOWER LIMIT OF RANGE (Range: -1.700E+38 to 991.791) [0.0]:?
Enter lowest value to be scaled to output range
LOWER LIMIT OF RANGE (Range: -1.700E+38 to 991.791) [0.0]:
UPPER LIMIT OF RANGE (Range: 0.0 to 1.700E+38) [65535.0]:?
Enter Highest value to be scaled to output range
UPPER LIMIT OF RANGE (Range: 0.0 to 1.700E+38) [65535.0]:
```

Note: The entry `?` has been used to show on-line prompt information.

The `Main menu: ENTER COMMAND [INPUT DATA]:output` selects the `OUTPUT DATA` command. `FILE FORMAT [FIT2D STANDARD FORMAT]:binary` selects the `BINARY (UNFORMATTED)` format for file output. `BINARY FILE NAME [fit2d.bin]:` will create an output file called `fit2d.bin`, as the default is accepted.

`INFO: Data region is 20 * 20 pixels` is information showing how many pixels are output in each direction in the data file. This information will be useful when inputing the data into another program.

`RECORD LENGTH (BYTES) (Range: 1 to 6000) [40]:` asks the length of (Fortran) records to be created. Generally the default value is just accepted, however a larger value may be used, and the user is given the choice to pad out records with blanks. This can be useful for creating a pseudo-MarResearch format file .

`FIRST RECORD FOR OUTPUT (Range: 1 to 100000) [1]:` allows blank records to be created at the beginning of the file. Again, generally this can be ignored (default is no blank records), but for pseudo-Mar-Research output this may be used.

`BIG ENDIAN FORMAT INTEGERS [NO]:yes` determines the byte order which is used to output the integer pixel values. Different computers write out several byte integers numbers (2-byte, 4-byte integers) in one of two different orders. “Big endian” means that the first byte of the integer number is the most significant byte, whereas “little endian” means that the first byte is the least significant and the most significant is the last byte[27]. DEC-Vax's, IBM-PC's and compatibles use little endian format integers, whereas Hewlett Packard 700 series, Sun-4, and Silicon Graphics use big endian integers. (**FIT2D** converts integers into the required byte ordering prior to output so can create data suitable for either format regardless of the type of computer which is being used.)

```
INFO: Data minimum value =    16.43577
INFO: Data maximum value =    991.7908
LOWER LIMIT OF RANGE (Range: -1.700E+38 to 991.791) [0.0]:
UPPER LIMIT OF RANGE (Range: 0.0 to 1.700E+38) [65535.0]:
```

This final section shows the range of data values present in the *ADR*, and allows the user to set the range of data values which will be scaled to the 2-byte unsigned integer output range (0: 65535). The reason for this complication is simple: within **FIT2D** the data values may range from approximately -1.7^{38} to $+1.7^{38}$, and values may be as small as 1.0^{-38}. Such a large range

[27]“Big endian” format is consistent with the convention for the numbering bits with a byte, and with normal order of digits within a number. Whereas “Little Endian” is consistent with the manner in which we write (in the “west”) and think of a bit/byte stream arriving.

cannot be scaled into 65536 intensity levels without enormous discretisation. Since, in general not all the range is used, the problem is usually not too severe. By asking for the `LOWER LIMIT OF RANGE` and the `UPPER LIMIT OF RANGE` **FIT2D** allows the user to control any truncation of the data range, and the level of discretisation of the range.

If the data range falls within 0.0 to 65535.0, these values will be offered as the default values. If this is the case data values will be rounded to their nearest integer value prior to being output (any fractional intensity information is lost in the output file). Apart from this rounding, values in the output file will have the same intensity as within **FIT2D**.

If there are values above 65535.0, then the maximum value will be offered as top of the range. If this is accepted, then there will be no thresholding truncation of data values, but all values will be scaled down proportionally to fit into the range 0.0 to 65355.0.

Values of `LOWER LIMIT OF RANGE` apart from 0.0 should be considered very carefully. If this is not zero then the output data has this value as an off-set and is no longer linear, unless this off-set is re-applied.

Here is an example of a "header" file which was created from an *ADR* from (120, 130) to (400, 350). From this information it is possible to know the *ADR* which was used to produce the data. This is not possible from the binary data file alone. Pixel sizes and the byte order in which the integers are output are recorded.

```
Name of binary file containing image data = fit2d.bin
Number of pixels (horizontal/vertical) =       281       221
Integer data: Number of bytes per pixel =  2 (signed)
The data has been written in "little endian" format. (Normal for VAX/PCs)
Nominal horizontal pixel size =      100.000 (microns)
Nominal vertical pixel size   =      100.000 (microns)
Pixel number of first output pixel (pixel offsets) =       120       130
Title = fit2d.bin
X-axis label = Columns
Y-axis label = Rows
Z-axis label = Intensity
Input data value scaled to lowest file value  =   .0000000E+00
Input data value scaled to highest file value =  6.5535000E+04
Record length =      562 (bytes); First output record =       1
```

15.64.3 BSL FORMAT

The BSL format is essentially the same as the OTOKO format developed at Hamburg. It consists of an ASCII header file and a number of binary image data files. The images are stored in floating point reals. (This may cause problems when files are moved between non-IEEE and IEEE float point reals systems.)

This may also cause problems between little-endian and big-endian systems since the endianess of the stored data is not recorded.

FIT2D outputs one image in the file.

15.64.4 CBF

“CBF” is the “Crystallographic Binary File” which is analogous to the “Crystallographic Information File” (CIF). CIF is the standard for transporting and archiving ASCII data within the international crystallographic community. “CBF” is a related format which is as close to CIF as possible, whilst allowing large dat-sets to be stored in efficient binary forms.

This option is presently only for development and testing purposes since the data names are not completely fixed, and may change.

15.64.5 CHIPLOT

Single rows or columns of the data may be output as 1-D series of X/Y coordinates for plotting with **CHIPLOT**. The values are output in a simple ASCII format together with title and axis label information.

As well as the name of the output file, the user is prompted for output of rows of data, or columns of data, and which row or column to output e.g.

```
Main menu: ENTER COMMAND [GAUSSIAN]:output
FILE FORMAT [FIT2D STANDARD FORMAT]:chiplot
Enter name of output file
FILE NAME [chiplot.dat]:
OUTPUT ROWS [YES]:
NUMBER OF ROW TO OUTPUT (Range: 1 to 20) [1]:10
```

The first line of the file contains the title of the data, the second line the X-axis label, and the third line the Z-axis (intensity) label. The fourth line indicates the number of following data points, and whether or not error values are given.

e.g. Here is the start of such a data file:

```
Simulated Data
Columns
Intensity
         20
 5.0000000E-01   .0000000E+00
 1.5000000E+00  1.0000000E+00
 2.5000000E+00  6.5600000E+00

   ...
```

15.64.6 COMPRESSED DIFFRACTION DATA

This format allows typical diffraction data to be stored in an efficient 2-byte integer format.

15.64.7 DENZO MAR FORMAT

This is a temporary manner in which to output data corrected for detector distortions in a format suitable for processing with **DENZO** and certain other programs. The format is very similar to the `BINARY (UNFORMATTED)` output (see Section 15.64.2, Page 147), but the data is transposed prior to output, and an extra blank record is output before the start of the image data.

This is not a true Mar format file, as the header record is left blank, but at present this appears to be good enough for inputting data to **DENZO**.

15.64.8 DUMP

This is exactly the same as `BINARY (UNFORMATTED)` format (see Section 15.64.2, Page 147). (The option was previously called `DUMP` but `BINARY (UNFORMATTED)` is preferable as this is the option for input of the same data type. The `DUMP` option is conserved for backwards compatibility.)

15.64.9 FIT2D STANDARD FORMAT

This flexible format saves the *ADR* of the current data together with any variances, axis data, and text labels. Data may be saved and input into **FIT2D** using the `INPUT DATA` command with no loss of information (within the *ADR*).

You are recommended **NOT** to use this format for taking data away from the ESRF, unless you have **FIT2D** or equivalent input routines.

15.64.10 GSAS

This format is for input to the GSAS (General Structural Analysis System) program [15] of 1-D 2-θ scans such as are created by integrating 2-D powder rings to 1-D scans.

15.64.11 MCA FORMAT

An ASCII dump of data values in 8 columns per line Fortran g10.3 format.

15.64.12 SPREAD SHEET

This is an ASCII format, where each line of the *ADR* is output as one line of the file. Each value is output in an adaptable format (Fortran 1pe12.5 format[28]) separated by a space. A header line is output first which contains the size of the data (number of X-axis elements and number of Y-axis elements) followed by the starting pixel of the output region.

[28]Fortran "g" format would have been used, but it is not handled correctly on HP-UX.

If the *ADR* is small enough this may be convenient for outputting values to examine of the the terminal. e.g. Here a 5 by 8 region has been output which contains a diffraction spot.

```
       5        8 Start pixel = (     530     708)
8.10267E+02  7.32946E+02  7.37758E+02  7.88272E+02  7.65070E+02
8.83738E+02  8.23922E+02  8.39668E+02  8.63968E+02  8.12600E+02
9.62089E+02  1.31421E+03  1.51055E+03  1.18729E+03  8.35718E+02
2.01099E+03  1.08955E+04  2.80248E+04  4.63233E+03  1.03409E+03
2.80316E+03  1.52684E+04  2.86683E+04  4.57030E+03  1.19153E+03
1.12099E+03  1.75127E+03  2.34980E+03  1.70756E+03  1.04862E+03
8.16105E+02  8.79685E+02  9.20219E+02  9.37691E+02  8.76449E+02
7.77960E+02  8.14157E+02  8.16885E+02  8.20791E+02  7.79102E+02
```

15.64.13 TIFF INTEGERS

This allows the current *ADR* to be stored in a TIFF file, in either 1 or 2-bytes per pixel format. The user is prompted for the name of the output file, and whether 1 or 2-byte per pixel output is required. The minimum and maximum values encountered within the *ADR* are output and the user is asked for the minimum and maximum data values to be scaled to the (limited dynamic range) output values. With 1-byte output only 256 output levels are available, and with 2-byte output 65536 levels are available. For more details on this scaling see the `BINARY (UNFORMATED)` documentation, Section 15.64.2, Page 147.

This is an example of the program output:

```
FILE FORMAT [FIT2D STANDARD FORMAT]:tiff
NAME OF TIFF OUTPUT FILE [fit2d.tif]:
DATA TYPE FOR PIXEL VALUES [1 BYTE UNSIGNED INTEGERS]:
INFO: Data minimum value =    .0000000E+00
INFO: Data maximum value =    999.7000
LOWER LIMIT OF RANGE (Range: -1.700E+38 to 999.700) [0.0]:
UPPER LIMIT OF RANGE (Range: 0.0 to 1.700E+38) [999.700]:
```

Note: For input into `xv` only 1-byte per pixel files should be created. 2-byte per pixel files will cause `xv` to crash. (This is a limitation of `xv`). Many other image treatment programs do not deal properly with 2-byte per pixel TIFF files.

15.65 PAGE POSITION

Allows the position and size of the graphics, e.g. image or X/Y graph, on the 'page' to be controlled. The coordinates input are those of the maximum region used for the axes frame of the image, contour plot or graph. Since the title and axis label positions are 'attached' to the axes frame this also changes their position.

Change the position for the image display in the graphics window. This allows you to change the size of the image display, or other graphics, within the graphics window.

(Images and contour plots are drawn with the correct aspect ratio, so if the aspect ratio of the page position region is different not all of the region will be used. The graphic will be centred within the region.)

15.66 PAUSE

The PAUSE command waits for the user to press the < Return> (or <Enter>) key before continuing (c.f. the SLEEP command). This is the case also within macros and allows interactive macros to wait for user response.

15.67 PEEP

PEEP allows the user to use the graphics cursor to click on pixels within the current *ADR* and obtain information associated with those pixels e.g. pixel coordinate, pixel number, data coordinate, and pixel intensity.

If the experimental geometry has been defined (See Section 15.43, Page 126) PEEP will also produce angular information and corresponding d-spacings.

The user clicks in a large "button" to exit this option.

15.68 PIXEL REGION

PIXEL REGION is an alternative method to REGION to set the *ADR*. PIXEL REGION uses pixel numbers to set the *ADR* whereas REGION uses data coordinates which may be very different owing to offsets and re-binning of images.

The user is prompted for the X/Y lower and upper limits of the required *ADR*. The values must be within the given ranges, which are between 1 and the number of defined values for the lower limits, and between the lower limits and the number of defined values for the upper limits. e.g. The prompts and input values may appear as follows:

```
X-LOWER LIMIT (Range: 1 to 1152) [1]:500
Y-LOWER LIMIT (Range: 1 to 1482) [1]:600
X-UPPER LIMIT (Range: 500 to 1152) [1152]:800
Y-UPPER LIMIT (Range: 600 to 1482) [1482]:900
```

15.69 PLOT DATA

Plot *ADR* of data as a 2-D false colour image (or X-Y graph if 1-D). The *ADR* is plotted as a false colour image using the current graphics style e.g. current choice of colour table and current text style. If the image is larger than the resolution of the screen window, the data is automatically rescaled to fit within the window.

The maximum extent of the 'page' used to output the image may be changed using `PAGE POSITION` (See Section 15.65, Page 152). The area set by `PAGE POSITION` is a maximum extent; to preserve aspect ratio the image may well be drawn smaller in either the horizontal or vertical direction. The image is always drawn centred within the maximum extent.

15.70 POISSONIAN NOISE

Adds noise with Poisson statistics to data within *ADR*. Each element within the *ADR* is taken to be the mean of a Poisson distribution. A random sample is generated from each distribution. (This can take a long time for large images.)

If variance arrays are present then the values of the data array **before** adding Poisson statistics will be used as the variance values.

15.71 POLARISATION EFFECT

Simulates the effect of beam polarisation on Bragg diffracted intensities.

A graphical form is used to control the polarisation factor, the pixel sizes, the sample to detector distance, and the position of the direct beam on the detector. The detector is assumed to be orthogonal to the direct beam.

The polarisation factor is defined as $(I_h - I_v)/(I_h + I_v)$, where I_h is the horizontal component and I_v is the vertical component. (Horizontal should normally correspond to the X-direction of the image.)

Intensities in the *ADR* are reduced by the effect that the defined polarisation and geometry would produce.

15.72 POSTSCRIPT OPTIONS

The `POSTSCRIPT OPTIONS` command toggles the PostScript output between colour mode and black and white mode. If the colour table is purely black and white, the black and white mode will produce better quality output on colour printers.

15.73 POWER SPECTRUM

Calculates the power spectrum of the current ADR. At present this must be a power of two pixels in length for both of the two dimensions.

The output is in the "memory".

15.74 PREDICTOR

The `PREDICTOR` command applies one of a choice of simple “predictors” to the current *ADR*. The output is in the “memory”. Predictors are an important part of data compression algorithms for image data. The idea of a predictor is to predict the value of a pixel from surrounding pixels. If the predictor algorithm works well the number of bits needed to store the difference between the predictions and the actual data is less than the number of bits needed to store the raw data values. Thus they can greatly help in data compression.

A choice of eleven different predictors is provided:

If x is the current pixel, and a, b, c, d are the “previous” pixels as shown:

```
a x
c b d
```

1. $predictor = x - a$
2. $predictor = x - Int((a + b)/2)$
3. $predictor = x - Int((a + b + c)/3)$
4. $predictor = x - Int((a + b + c + d)/4)$
5. $predictor = x - ((a + b - c)$
6. $predictor = x - Int((3a + 3b - 2c)/4)$
7. $predictor = x - Int((2a + 2b - c)/3)$
8. For pairs of values a and b, replace with $a' = Int((a + b)/2)$ and $b' = a - b$ (this is reversible) This is applied alternately horizontally and vertically to a specified level, then the series of a' values are stored as a simple previous value differential
9. $predictor = x - Int((a + b + 1)/2)$
10. $predictor = x - Int((a + b + c + 1)/3)$
11. $predictor = x - Int((a + b + c + d + 2)/4)$

(All in integer arithmetic)

This is used to investigate data compression algorithms.

15.75 PRINT GRAPHICS

Output current screen graphics to a PostScript file. The first time `PRINT GRAPHICS` is used it will prompt for the name of an output file. Subsequent commands will send the graphics to the same file, unless an `END GRAPHICS FILE` command is issued to close the file (See `END`

`GRAPHICS FILE`, Section 15.31, Page 123). (For graphics to be included in LaTeX documents only one graphic per file is presently suitable.)

Further `PRINT GRAPHICS` commands will add new "pages" to the existing file.

After exiting **FIT2D** the files can be set to any PostScript printer or may be viewed using a PostScript previewer such as `ghostview`. **FIT2D** will produce colour Postscript if colour has been used on the screen. The file can be sent to a colour printer, or to a black and white printer in which case the colours will be converted to a grey-scale.

The user can name the files as they want, but I suggest you always use the "file extension" `.ps` to distinguish the files.

Beware: PostScript files can be large, so after printing it is recommended to delete the unnecessary files.

15.76 PUBLICATION QUALITY

This command is designed to select quickly all the attributes for graphics quality suitable for publication. Principally this means that all line widths, except those of the grid, are set to double thickness, and all fonts are set to the 'best' available font[29]. Thicker lines are generally needed for photo-reduction.

15.77 QUESTION

The `QUESTION` command is used to add interactive input to a macro. Either the GUI or the "keyboard" interface may be used.

The following types of user input may be defined:

- **CHARACTER STRING**: Input a line of text
- **GRAPHICAL COORDINATE**: Input a coordinate by mouse click
- **INPUT FILE**: Specify an input file name
- **INTEGER VALUE**: Input an integer value
- **LOGICAL VALUE**: Input a logical "YES/NO" value
- **OUTPUT FILE**: Specify an output file name
- **REAL VALUE**: Input a real value

The user prompt is defined, as is an optional default value, and optional range checking. The entered value or values are stored in one or more program variables.

[29]At present the graphics system only supports one font family so that the font do not change, but the font family used: "Times-Roman" should be suitable for most purposes.

Clearly **GRAPHICAL COORDINATE** input will always come from the graphics window.

The following example shows the definition of a question to input an integer value from the user:

```
Main menu: ENTER COMMAND [Z-SCALE]:question
TYPE OF DATA VALUE TO DEFINE [INTEGER VALUE]:int
ENTER USER PROMPT FOR VALUE INPUT:ENTER NUMBER OF THE DAY OF THE MONTH
USE GRAPHICAL USER INTERFACE [YES]:yes
GIVE USER DEFAULT VALUE [YES]:yes
ENTER DEFAULT VALUE FOR INTEGER:1
RESTRICT INPUT VALUE RANGE [YES]:yes
LOWER LIMIT OF RANGE FOR INTEGER:1
UPPER LIMIT OF RANGE FOR INTEGER (Range: 1 to 2147483647):31
INFO: Enter help text or user escape (\\) to exit
TEXT:Enter an integer number between 1 and 31 for the number
TEXT:of the day of the month.
TEXT:\\
ENTER VARIABLE NAME [#VALUE]:#DAY

NOTE: !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
             INTERACTIVE INPUT REQUIRED FOR THE MACRO
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

After the `#DAY` is entered the GUI integer value window appears and will accept a valid integer between 1 and 31.

15.78 QUIT

Exit from the program (See `EXIT`, Section 15.34, Page 124).

15.79 RAISE TO A POWER

`RAISE TO A POWER` allows all the elements of the *ADR* of the current data to be raised to a desired power. Depending on the power entered this allows the elements to be squared (`POWER = 2`), the square root of the elements obtained (`POWER = 0.5`), the reciprocal of the elements calculated (`POWER = -1`), or an arbitrary power calculated. The operation is performed in place, the memory is not affected.

Depending on the power the range of valid input element values changes e.g. negative numbers are not valid if the square root is being taken. If invalid numbers are encountered the output will be given a special value and a warning message will be issued.

The user is prompted for the required power to which the *ADR* elements are to be raised. e.g. The prompt and input may be as follows to cube all the elements:

```
POWER [2]:3
```

15.80 REBIN

Alternative name for `RE-BIN` command, see Section 15.81, Page 158.

15.81 RE-BIN

Re-bins data in the *ADR* by integer and non-integer amounts. The user specifies the number of input pixels to be re-binned into an output pixel in each direction. Output is in the “memory”.

e.g. To re-bin every two pixels in each direction into one output pixel:

```
Main menu: ENTER COMMAND [INPUT DATA]:rebin
X REBIN NUMBER (NUMBER INPUT TO 1 OUTPUT)
 (Range: .10000 to 500.000) [2.00000]:
Y REBIN NUMBER (NUMBER INPUT TO 1 OUTPUT)
 (Range: .10000 to 500.000) [2.00000]:
X START OUTPUT (Range: 1 to 500) [1]:
Y START OUTPUT (Range: 1 to 500) [1]:
Main menu: ENTER COMMAND [EXCHANGE]:
```

15.82 RECALL

Recalls a data-set from internal memory, Data previously stored as current data is lost.

15.83 REFLECT

Reflects data in the *ADR* about a line of reflection specified by the user. The user is prompted for the coordinates of two distinct points to define a line of reflection, and for the extent of the output region in the “memory”.

The reflected data is output in memory. Only the region of the memory corresponding to the current data *ADR* is overwritten.

15.84 REGION

User input of coordinates to specify a new active data region. The user is prompted for the coordinates of the lower left and upper right limits of the required *ADR*. The coordinates are all in data coordinates, which are then converted to pixel numbers.

Note: The current *ADR* affects almost all data manipulation and data display commands.

15.85 RING (ADD POWDER RING)

Adds a simulated powder ring to the current *ADR*. The powder ring has a Gaussian profile of a specified sigma, and may by elliptical owing to detector non-orthogonality (tilt). The simulated powder ring is added directly to the current data array.

The following example shows such a ring being simulated:

```
Main menu: ENTER COMMAND [QUESTION]:ring
X-DIRECTION PIXEL SIZE (MICRONS)
 (Range: 1.000000E-03 to 10000.00) [100.0000]:
Y-DIRECTION PIXEL SIZE (MICRONS)
 (Range: 1.000000E-03 to 10000.00) [100.0000]:
X-PIXEL COORDINATE OF BEAM CENTRE [64.00000]:
Y-PIXEL COORDINATE OF BEAM CENTRE [64.00000]:
SAMPLE TO DETECTOR DISTANCE (MILLIMETRES)
 (Range: .100000 to 1.000000E+05) [200.0000]:
TILT PLANE ROTATION ANGLE (DEGREES)
 (Range: -180.0000 to 180.0000) [0.0]:30
DETECTOR TILT ANGLE (DEGREES) (Range: -180.0000 to 180.0000)
 [0.0]:5
OPENING ANGLE OF DIFFRACTION RING (DEGREES)
 (Range: 0.0 to 90.00000) [2.000000]:
PEAK MAXIMUM INTENSITY [500.0000]:
STANDARD DEVIATION WIDTH (X-PIXELS) OF RADIAL PROFILE
 (Range: 0.0 to 1.700000E+38) [2.000000]:
Main menu: ENTER COMMAND [PLOT DATA]:
```

15.86 ROI (Region Of Interest)

`ROI` is an alternative name for the `PIXEL REGION` command (See Section 15.68, Page 153). The effect is exactly the same.

The command `ROI` has been added as this is tending to be the name used for this operation in an increasing number of PC-based image display programs. It has been added to **FIT2D** for compatibility.

15.87 ROTATE LUT

Interactive rotation of the colour table controlled by a graphics menu. A number of "buttons" are produced within the graphics window which allow the current colour table to be rotated by set amounts. By clicking in the different "buttons" the user should see the immediate effect of rotating the colour table. The "button" labelled `DEFAULT` returns the colour table to the values that were defined at the start of the operation. The "button" labelled `EXIT` is used to return to the main menu.

Note: As the background and text are drawn using two of the colour levels, they will change colour during the rotation of the colour table. Text may no longer be visible. After the `ROTATE LUT` command a `PLOT DATA` command can be issued to draw the data with the rotated colour table and correct background and text colours.

15.88 RUN MACRO

Same as `MACRO`, See Section 15.56, Page 143.

15.89 SELECT PIXEL OPERATION

The `SELECT PIXEL OPERATION` command allows particular pixels to be selected for either multiplication or addition operations with a scalar, depending on an input intensity value criterion. The user can select either pixels which have an intensity greater than an input constant, or pixels which have an intensity less than an input constant. The pixels selected may either be multiplied by an input scalar, or added to an input scalar. The operations take place within the main program array.

This operation may be of use in "flagging" over-loaded pixels.

Here is an example of the program output when all pixel values with greater than 700 have been multiplied by 1.5.

```
Main menu: ENTER COMMAND [HELP]:select
GREATER THAN COMPARISON [YES]:?
"YES" if pixels are to be selected by a greater than comparison (exclusive)
"NO" if pixels are to be selected by a lesser than comparison (exclusive).
GREATER THAN COMPARISON [YES]:
DECISION PIXEL VALUE [16382.5]:?
Input pixel value for comparison
DECISION PIXEL VALUE [16382.5]:700
MULTIPLICATION ("NO" = ADDITION) [YES]:?
"YES" if the pixel values are to be multiplied by a constant, "NO" if a
constant is to be added to the selected pixels.
MULTIPLICATION ("NO" = ADDITION) [YES]:
OPERAND VALUE [0.0]:?
Input value to multiply or add to pixel values
OPERAND VALUE [0.0]:1.5
```

15.90 SEQUENCE

The `SEQUENCE` command is almost certainly the most powerful and also the most complicated command within **FIT2D**. To cope with its complication much explanation of required input is available by entering a question mark (?) for each prompt. The `SEQUENCE` command allows a pre-defined macro to be run automatically on a whole series of files with automatic definition of program variables (See **Macros** Section 19, Page 225, and the `START MACRO` command Section 15.108, Page 169). The `SEQUENCE` command can also be used to repeatedly run a macro without changing any file names, but this possibility is largely uninteresting unless an input or output file name is changed.

Often a large number of files exist in some sequence and it is required to perform some operation on the files and produce some output or a similar sequence of outputs. The output may data files, or may be graphics PostScript files. Such a sequence of input files typically looks like:

```
lys1_001.pmi
lys1_002.pmi
lys1_003.pmi
```

...

```
lys1_178.pmi
lys1_179.pmi
lys1_180.pmi
```

File sequences of this sort can be treated, but in fact much more flexibility is available:

- The value of the first file does not need to be one
- Steps other than unity are also available
- The number of characters used to store the numerical part of the file name is user controlled
- The number of characters used to store the numerical part of the file name may be variable
- The variable numerical part of the file name may be at the start, middle, or end of the file name
- The file name (variable value) does not have to vary
- Output file names can be very similar to input file names e.g. Only the file extension changes, or the name can be defined very differently e.g. The output files can be output in the reverse numerical order to the input files

The `SEQUENCE` command allows a macro to be run for each file in the sequence, with control over the variable values e.g. file names, of a variable number of program variables. A loop is run a number of times, controlled by a loop variable, which has a start value, a maximum value, and a step increment value. The value of the loop variable can be used to define the values (translations) of the program variables.

The starting value of the loop counter variable is set and for each loop:

- Program variables are defined, based directly, based indirectly, or independent of the loop value
- A named macro is called, and run with the defined variable values
- The loop counter value is incremented
- If the loop counter value is within the end condition value, the previous operations are repeated

To define the sequence, first the loop counter variable's initial value, end limit value (maximum or minimum inclusive limit value depending on increasing or decreasing counter values), and the increment step value are defined. A valid finite loop must be declared or **FIT2D** will re-prompt for the input values:

```
LOOP COUNT START VALUE [1]:
LOOP COUNT END LIMIT VALUE (INCLUSIVE) [10]:3
LOOP COUNT INCREMENT STEP [1]:
```

The increment step value may be a value other than 1, and may be negative. If the increment step is negative the `END LIMIT VALUE` must be less than or equal to the `START VALUE`. The loop will always be run at least once; if you decide that you do not want to continue the `SEQUENCE` command now or later, enter user escape (\\).

When the looping conditions are properly defined, **FIT2D** prompts for the name of the macro file to be run.

```
INPUT MACRO FILE NAME [fit2d.mac]:
```

The entered file name is checked to exist and be a valid macro file.

Next the number of automatically controlled program variables and variable values (translations) is defined. Program variables can also have been previously defined using the `DEFINE VARIABLE` command, but their values (translations) will be static, whereas here they can be defined to change dynamically.

```
NUMBER OF VARIABLES TO DEFINE (Range: 0 to 20) [2]:
```

For each such definable program variable **FIT2D** prompts for the name of the variable, followed by a number of other prompts which allow different values to be defined depending on the loop counter value. e.g.

```
ENTER VARIABLE NAME [#IN]:
VARIABLE VALUE BASE [lys1_]:test_
VARIABLE PART FIXED LENGTH [YES]:
NUMBER OF CHARACTERS IN VARIABLE PART (Range: 0 to 20) [3]:2
```

```
ARITHMETIC ON LOOP COUNT VALUE [NO]:y
MULTIPLIER FOR LOOP COUNT VALUE [-1.00000]:
CONSTANT TO ADD TO MODIFIED LOOP COUNT VALUE [21.0]:
VARIABLE VALUE EXTENSION [.pmi]:.temp
```

The `ENTER VARIABLE NAME` prompt requires names of required variables to be entered. Any string may be entered, but clearly it only makes sense to enter program variables which are used within the macro to be run (See Section 19, Page 225). By default the first variable name is `# IN` and the second is `# OUT`.

Variable values are constructed from a fixed initial component, a numerically based changing middle component, and a fixed end component. The variable value may be up to a total of 256 characters in length. Any of the three parts may be blank, allowing the changing component to be at the beginning or end, and allowing non-changing variable values. To define a component as blank a special program variable exists: `##BLANK`[30]. (`##BLANK` is a special variable which is defined by **FIT2D** on start-up.)

First the base part of the variable values is entered: `VARIABLE VALUE BASE [lys1_]:`.

`VARIABLE PART FIXED LENGTH [YES]:` prompts for whether or not the changing part of the variable values should contain a fixed number of characters or a variable number.

If a variable number of characters is selected **FIT2D** prompts for the number of characters: `NUMBER OF CHARACTERS IN VARIABLE PART (Range:  0 to 20) [3]:`. Otherwise this prompt does not occur.

If a fixed number of characters is chosen, numbers which take less characters than the input number are padded out with preceding zeros. If at any time during the sequence the number is too big to be represented in the given number of characters, the sequence processing will stop. If a variable number is chosen, only the characters necessary to represent the number will be used.

The loop counter value can be used directly or indirectly to define the variable part of the variable values. If indirect use is required enter `YES` to the `ARITHMETIC ON LOOP COUNT VALUE [NO]:` prompt. The default is `NO`. Entering `YES` allows the counter value to be multiplied by an input multiplier and then added with an input addition constant prior to it being used to form the variable value. The constants are input as real values, and real arithmetic is performed. The result is converted to the nearest integer prior to being used to form part of the variable value. The integer used is defined:

$$variable\_integer\_value = Nint(Real(loop\_value) * multiplier + add\_constant) \tag{3}$$

Where: Real converts an integer number to a real number
Nint converts a real number to the nearest integer

`MULTIPLIER FOR LOOP COUNT VALUE [-1.00000]:` inputs the multiplier to be used and `CONSTANT TO ADD TO MODIFIED LOOP COUNT VALUE [21.0]:` inputs the addition constant.

[30]This special variable is necessary, since just entering a space or returning will select the default value

By using a negative multiplier and defining a suitable addition constant it is easy to define a sequence of program variables which count in the opposite direction to the loop counter value.

Finally the latter part of the variable value is entered: `VARIABLE VALUE EXTENSION [.pmi]:`. Note: The required input is not the file name extension, but all the characters following the changing part of the value, including the dot (.).

Note: The `SEQUENCE` command cannot be used within a macro. Thus, if you have entered the `START MACRO` command and are in the process of creating a macro, entering the `SEQUENCE` command will result in a warning message.

15.90.1 Examples of the `SEQUENCE` Command Usage

For normal use the `SEQUENCE` command should be relatively straightforward to use, but to exploit the full potential may require some careful thought. In either case some examples are useful.

The following example shows how the macro to correct for spatial distortion defined in Section 29.4, Page 256 may be used to process a file sequence starting with `lys04.gel` and ending with `lys20.gel`. The corresponding output files should have the same name as the input file, but with the extension `.dat` instead of `.gel`. The macro file is called `lyso.mac` and uses two program variables: `#IN` for the input file and `#OUT` for the output file. The following is the program output and user input for this sequence definition:

```
Main menu: ENTER COMMAND [INPUT DATA]:sequence
LOOP COUNT START VALUE [1]:4
LOOP COUNT MAXIMUM VALUE (INCLUSIVE) [10]:20
LOOP COUNT INCREMENT STEP [1]:
INPUT MACRO FILE NAME [fit2d.mac]:lyso.mac
NUMBER OF VARIABLES TO DEFINE (Range: 0 to 20) [2]:
ENTER VARIABLE NAME [#IN]:
VARIABLE VALUE BASE [lys1_]:lys
VARIABLE PART FIXED LENGTH [YES]:
NUMBER OF CHARACTERS IN VARIABLE PART (Range: 0 to 20) [3]:2
ARITHMETIC ON LOOP COUNT VALUE [NO]:
VARIABLE VALUE EXTENSION [.pmi]:.gel
ENTER VARIABLE NAME [#OUT]:
VARIABLE VALUE BASE [lys]:
VARIABLE PART FIXED LENGTH [YES]:
NUMBER OF CHARACTERS IN VARIABLE PART (Range: 0 to 20) [2]:
ARITHMETIC ON LOOP COUNT VALUE [NO]:
VARIABLE VALUE EXTENSION [.cor]:.dat
```

As an example of a slightly more complicated task we look at re-numbering a file sequence in reverse numerical order. The first file is `ice1.dat` and the last file is `ice10.dat`. The output file corresponding to `ice1.dat` is to be called `ICE10.OUT` and the output file corresponding to `ice10.dat` is to be called `ICE1.OUT`. The macro is called `ice.mac` and uses the program variables: #IN for the input file and #OUT for the output file. The following is the program output and user input for this sequence definition:

```
Main menu: ENTER COMMAND [INPUT DATA]:seq
LOOP COUNT START VALUE [4]:1
LOOP COUNT MAXIMUM VALUE (INCLUSIVE) [20]:10
LOOP COUNT INCREMENT STEP [1]:
INPUT MACRO FILE NAME [ice.mac]:
NUMBER OF VARIABLES TO DEFINE (Range: 0 to 20) [1]:2
ENTER VARIABLE NAME [#IN]:
VARIABLE VALUE BASE [lys]:ice
VARIABLE PART FIXED LENGTH [NO]:
ARITHMETIC ON LOOP COUNT VALUE [NO]:
VARIABLE VALUE EXTENSION [.gel]:.dat
ENTER VARIABLE NAME [#OUT]:
VARIABLE VALUE BASE [ice]:ICE
VARIABLE PART FIXED LENGTH [NO]:
ARITHMETIC ON LOOP COUNT VALUE [NO]:y
MULTIPLIER FOR LOOP COUNT VALUE [1.00000]:-1
CONSTANT TO ADD TO MODIFIED LOOP COUNT VALUE [0.0]:11
VARIABLE VALUE EXTENSION [.dat]:.OUT
```

15.91 SET ANNOTATION STYLE

The user may define the text style (font, size, character spacing), and direction individually for all possible annotation labels (independently of whether or not they are being used).

15.92 SET ARROW STYLE

The user may define the arrow style (colour, line width, arrow head type), for all possible annotation arrows (independently of whether or not they are being used).

15.93 SET AXES STYLE

The user may define the axes style (colour, line width, number of small tick marks), for horizontal and vertical axes.

15.94 SET BACKGROUND STYLE

By default the background of the graphics window is white, and axes lines and text are black. SET BACKGROUND STYLE allows the background colour to be set. A number of basic colours are available.

Note: if a chosen colour is not available in the current colour table it will be approximated by the "closest" available colour.

15.95 `SET COLOUR`

By default the axes lines and text are black. `SET COLOUR` allows the general foreground colour to be set. A number of basic colours are available. When used this will replace all colours as previously specified for individual graphics items.

Note: if a chosen colour is not available in the current colour table it will be approximated by the "closest" available colour.

15.96 `SET CURVE STYLES`

`SET CURVE STYLES` allows the style of representation of all possible curves which may be displayed by the program to be changed. (Typically 100 different curve styles may be set.) The user specifies the range of curves to be affected, and specifies whether line, marker, and/or error bars are to be displayed. If a line is to be displayed the user is requested to choose colour, line type, and line width attributes. If markers are to be displayed the user can select the type of markers to be displayed, together with their colour, size, line width and interior fill colour. Similarly for error bars the style, colour, and line width may be selected. (If variance estimates are not present no error bars will be displayed, even if they are selected herewithin.)

15.97 `SET ENUMERATION STYLE`

The user may control output and style of the enumeration for the X and Y-axes. The size and colour of characters may be set, and the number of figures used to label the axis positions. (At present only one font family is available, but for future uses, the user is prompted for the required font.)

15.98 `SET FONT`

The user may select the text font to be used for all subsequent graphics Roman text e.g. titles, labels, etc. The user is prompted for a number corresponding to each of the available fonts [31].

(The command `PUBLICATION QUALITY` (Section 15.76, Page 156) also (potentially) changes the text fonts to the type of font considered best for high quality output.)

[31] At present the graphics system only supports one font family so that the font do not change, but the font family used: "Times-Roman" should be suitable for most purposes.

15.99 SET GRID STYLE

The user may define the style of horizontal and vertical, fine and coarse grid lines (colour, line width, line type) e.g:

```
Main menu: ENTER COMMAND [INPUT DATA]:set grid
HORIZONTAL COARSE GRID LINE TYPE (Range: 1 to 4) [4]:?
1 = solid, 2 = dashed, 3 = dotted, 4 = dot-dash
HORIZONTAL COARSE GRID LINE TYPE (Range: 1 to 4) [4]:2
HORIZONTAL COARSE GRID LINE COLOUR [BLACK]:?
Enter one of the following available colours:
BLACK BLUE BROWN CYAN GREEN GREY MAGENTA ORANGE RED VIOLET WHITE YELLOW
HORIZONTAL COARSE GRID LINE COLOUR [BLACK]:orang
HORIZONTAL COARSE GRID LINE WIDTH SCALE FACTOR
 (Range: 0.0 to 100.000) [1.00000]:
VERTICAL COARSE GRID LINE TYPE (Range: 1 to 4) [4]:2
VERTICAL COARSE GRID LINE COLOUR [BLACK]:oran
VERTICAL COARSE GRID LINE WIDTH SCALE FACTOR
 (Range: 0.0 to 100.000) [1.00000]:
HORIZONTAL FINE GRID LINE TYPE (Range: 1 to 4) [3]:
HORIZONTAL FINE GRID LINE COLOUR [BLACK]:red
HORIZONTAL FINE GRID LINE WIDTH SCALE FACTOR
 (Range: 0.0 to 100.000) [1.00000]:
VERTICAL FINE GRID LINE TYPE (Range: 1 to 4) [3]:
VERTICAL FINE GRID LINE COLOUR [BLACK]:red
VERTICAL FINE GRID LINE WIDTH SCALE FACTOR
 (Range: 0.0 to 100.000) [1.00000]:
Main menu: ENTER COMMAND [PLOT DATA]:close
```

15.100 SET LAYOUT STYLE

The `SET LAYOUT STYLE` command allows the distances in the graphics diagrams to be changed. All the distances are specified in page coordinate units (i.e. the longer edge is one unit long).

The following example shows the gaps between the axis labels and enumeration being reduced:

```
Main menu: ENTER COMMAND [Z-SCALE]:set lay
TITLE TO AXIS GAP (Range: -1.000000 to 1.000000)
 [5.000000E-02]:3e-2
X-LABEL TO AXIS GAP (Range: -1.000000 to 1.000000)
 [5.000000E-02]:3e-2
Y-LABEL TO AXIS GAP (Range: -1.000000 to 1.000000)
 [5.000000E-02]:3e-2
X-AXIS ENUMERATION TO AXIS GAP
 (Range: -1.000000 to 1.000000) [1.500000E-02]:1e-2
Y-AXIS ENUMERATION TO AXIS GAP
```

```
 (Range: -1.000000 to 1.000000) [1.500000E-02]:1e-2
Main menu: ENTER COMMAND [PLOT DATA]:
```

15.101 SET TICK POSITIONS

By default axes are produced automatically with large tick marks at convenient positions and small tick marks in-between. Owing to the great variety of number ranges it is not always possible to provide ideally spaced tick marks. This command allows the user to specify their own positions for the large tick marks.

For the X and the Y-axes the user is prompted for automatic positions or not. If non-automatic positions are selected, the user will be prompted for the start position (in data coordinates), the interval between large tick marks (data coordinates) and the number of large tick marks to draw on the axis. (The number of large tick marks is defaulted to a value which covers the axes, but not outside.)

15.102 SET TITLE STYLE

The `SET TITLE STYLE` command allows the style of the title text output at the top of graphical diagrams to be changed from the default values. The output of the title, or not, the colour, and the character size may be changed. (Other questions are asked, but their answers have no effect at present.)

In the following example the title text is made twice as large and red:

```
Main menu: ENTER COMMAND [Z-SCALE]:set title
OUTPUT TITLE [YES]:y
TITLE FONT INDICE (Range: -100 to 100) [1]:
TITLE COLOUR [BLACK]:red
TITLE CHARACTER HEIGHT SCALE FACTOR (Range: 0.0 to 100.0000)
 [1.000000]:2
TITLE CHARACTER WIDTH SCALE FACTOR (Range: 0.0 to 10.00000)
 [1.000000]:
TITLE CHARACTER SPACING (Range: -5.000000 to 10.00000) [0.0]:
Main menu: ENTER COMMAND [PLOT DATA]:
```

15.103 SET X-LABEL STYLE

The `SET X-LABEL STYLE` command allows the style of the horizontal axis label text output underneath the frames of graphical diagrams to be changed from the default values. The output of the X-axis label, or not, the colour, and the character size may be changed. (Other questions are asked, but their answers have no effect at present.)

See the `SET TITLE STYLE` command (Section 15.102, Page 168) for an example of the user prompts.

15.104 SET Y-LABEL STYLE

The `SET Y-LABEL STYLE` command allows the style of the vertical axis label text output to the left to the frames of graphical diagrams to be changed from the default values. The output of the Y-axis label, or not, the colour, and the character size may be changed. (Other questions are asked, but their answers have no effect at present.)

See the `SET TITLE STYLE` command (Section 15.102, Page 168) for an example of the user prompts.

15.105 SLEEP

The `SLEEP` command allows macros to pause for a determined time and then continue without any user intervention (c.f. the `PAUSE` command). The length of pause is specified in seconds. e.g. The following example pauses for 5 seconds:

```
Main menu: ENTER COMMAND [PLOT DATA]:sleep
LENGTH OF PAUSE (seconds) (Range: 0.0 to 1000.000)
 [1.000000]:5
```

This command is mainly useful for demonstration macros, which should run continuously without any user intervention, but nevertheless need to pause so that message text can be read.

15.106 SMOOTH

This is an alternative name for the `BLUR command` (see Section 15.10, Page 112).

15.107 SPATIAL FILTERING

High-pass or low-pass filtering of the *ADR* using simple spatial filters. The user may select low or high-pass filtering with a choice of three degrees of filtering for each. The output is in the memory.

NOTE: At present there is no error propagation for this command.

15.108 START MACRO

Start recording commands within a named file for later re-use as a macro. After the file is opened all subsequent commands are recorded in the file until the `STOP MACRO` command is issued. The macro can later be run with the `MACRO` command. If the `MACRO` command itself is used, the contents of the macro are written into the new macro file. The manner to define and use macros are explained in Section 19, Page 225. (Also see `STOP MACRO`, Section 15.110, Page 170 and `MACRO` Section 15.56, Page 143.)

15.109 STATISTICS

Calculate a number of basic statistical quantities from the data within the current *ADR* e.g. mean, total counts, and standard deviation. A typical program display may be as follows:

```
Main menu: ENTER COMMAND [INPUT DATA]:stat
Statistics of active data region (ADR):
  DC limits: Min (    .50     ,    .50    ) Max (  100.     ,   100.    )
  Pixel number limits: Min (     1 ,     1) Max (  100 ,   100)
  Total number of data values =    100 *    100 =        10000
  Minimum data value =    1.146601
  Maximum data value =    3.099321
  Average (mean) data value =    2.000000
  Sum of data values =    20000.00
  Root mean square (RMS) value =    2.063915
  Standard deviation value =    .5096754
  Skewness parameter =    .1356223
```

(By changing the *ADR* and using `STATISTICS` a basic from of peak integration may be carried out. See `IMAGE` command, Section 15.49, Page 128, for the `DISPLAY` graphical sub-menu command `STATISTICS` for much more flexible polygon integration possibilities.)

The following program variables, all of data type floating point real are automatically defined by the `STATISTICS` command with corresponding values:

`##MINIMUM` Minimum data value within the *ADR*

`##MAXIMUM` Minimum data value within the *ADR*

`##MEAN` Mean (average) data value within the *ADR*

`##TOTAL` Sum of data values within the *ADR*

`##RMS` Root mean square value within the *ADR*

`##SIGMA` Standard deviation of data values within the *ADR*

`##SKEWNESS` The skewness parameter of the data values within the *ADR*

15.110 STOP MACRO

Closes a previously opened macro definition file (The `STOP MACRO` command is not recorded within the macro.) (See `START MACRO`, Section 15.108, Page 169 and `MACRO` Section 15.56, Page 143.)

15.111 STORE

Stores current data set in internal memory. The data is copied from the current data into the memory. Any previous data in the memory is destroyed.

Note: `EXCHANGE` may often be a much more efficient alternative to `STORE` for large data-sets.

15.112 SUBTRACT

Subtract the memory data from the current data throughout the current *ADR*.

15.113 SURFACE INTERPOLATION

The user defines a number of points in the current data *ADR* by clicking with the graphical cursor. The program calculates an average intensity in the 3*3 pixel region centred on each user defined coordinate. Triangulation and bi-cubic spline interpolation is used to define a surface throughout the *ADR* based on the user defined points. The output is in the memory.

For best results the input coordinates should be spaced throughout the *ADR* and more coordinates need to be defined where the surface should change quickly.

(An alternative and generally superior method of estimating background surfaces is available as part of the `FIT` sub-menu.)

15.114 SYMBOL

This is an alternative name for the `VARIABLE / DEFINE VARIABLE` command; they are identical (see Section 15.26, Page 121).

15.115 SYMMETRIC FUNCTION

Adds a circularly symmetric function to the current data, calculated from a 1-D profile. The 1-D profile needs to have been previously loaded into the memory (the first row in the memory is used to store the 1-D profile). The user specifies the centre of the required function in data coordinates. The function is added to the current data[32].

(A 1-D radial profile can be calculated within the `FIT` sub-menu using the `POWDER`

`DIFFRACTION` command, see Section 17.18, Page 215.)

[32]From V6.11 onwards this option takes account of radial and image pixel sizes.

15.116 TITLE

Allows the user to enter text for the title of the graphics. This will replace the old text.

15.117 THRESHOLD

The user is prompted for a minimum and a maximum threshold value. Any element of the *ADR* which is below the minimum value is set to this value and any value which is above the maximum value is set to the maximum value.

Note: There is no error propagation for this operation.

15.118 TRANSPOSE

Provided that the program array sizes are big enough the *ADR* will be transposed and output in the memory. i.e. `DATA(x,y)` will be transferred to `MEMORY(y,x)`. This will not only set the memory *ADR*, but will also re-define the memory defined data sizes.

15.119 UN-DEFINE VARIABLE

Program variables are created by the `DEFINE VARIABLE` (see Section 15.26, Page 121) and `VARIABLE` commands, and automatically by various commands e.g. `STATISTICS`. As the variable translation table is of a finite size[33] and since the `DEFINE VARIABLE` and `VARIABLE` automatically pole through the defined variable names as default suggestions, it can be useful to be able to remove variables.

`UN-DEFINE VARIABLE` allows variables to be removed from the translation table. If no variables exist a warning message is returned. If one or more variables exist, the user is prompted for the "name" of the variable to be removed. One of the existing variables will be proposed as a default.

The following text shows an example of defining a number of variables automatically using the `STATISTICS` commands, and then removing all of the variables until none are left.

```
Main menu: ENTER COMMAND [INPUT DATA]:stat
Statistics of active data region (ADR):
  DC limits: Min (    .50     ,    .50    ) Max (  100.     ,   100.    )
  Pixel number limits: Min (     1 ,     1) Max (  100 ,   100)
  Total number of data values =    100 *    100 =        10000
  Minimum data value =    75.00000
  Maximum data value =    1074.000
  Average (mean) data value =    511.5939
  Sum of data values =    5115939.
```

[33] Typically 500 variables can be stored.

```
  Root mean square (RMS) value =    574.2042
  Standard deviation value =    260.7468
  Skewness parameter =    .1356223
Main menu: ENTER COMMAND [CLOSE LOG]:un-define
ENTER VARIABLE NAME [##MINIMUM]:
Main menu: ENTER COMMAND [MACRO]:un-define
ENTER VARIABLE NAME [##MAXIMUM]:
Main menu: ENTER COMMAND [MACRO]:un-define
ENTER VARIABLE NAME [##MEAN]:
Main menu: ENTER COMMAND [MACRO]:un-define
ENTER VARIABLE NAME [##RMS]:
Main menu: ENTER COMMAND [MACRO]:un-define
ENTER VARIABLE NAME [##SIGMA]:
Main menu: ENTER COMMAND [MACRO]:un-define
ENTER VARIABLE NAME [##SKEWNESS]:
Main menu: ENTER COMMAND [MACRO]:un-define
ENTER VARIABLE NAME [##TOTAL]:
Main menu: ENTER COMMAND [MACRO]:un-define
WARNING: No variables are presently defined so you cannot "UN-DEFINE" any
Main menu: ENTER COMMAND [MACRO]:close
```

15.120 UNIT CELL PARAMETERS

Allows the user to convert the standard real space unit cell parameters, a, b, c, α, β, and γ into the reciprocal space parameters a^*, b^*, c^*, α^*, β^*, and γ^*. All angles are entered and displayed in degrees, and all distances should be in Angstroms or inverse Angstroms.

The user starts by choosing the direction of the conversion, followed by the parameters of the unit cell.

15.121 VARIABLE

This is an alternative name for the `DEFINE VARIABLE` command; they are identical (see Section 15.26, Page 121).

15.122 VARIANCES DEFINITION

Often data is obtained which essentially is affected by counting statistics, but which has been scaled by an unknown scaling factor. To be able to estimate the variance of each data point it is necessary to be able to estimate the scaling factor. This command estimates the scaling factor and hence estimates the variance of each data-point in the *ADR*.

This task is made slightly more complicated by detector background and by saturation of the detector.

The user inputs the range of non-background and non-saturated data-values. The program outputs the scaling factor and calculates estimated variances. The variances are calculated by applying local smoothing to the input data values and multiplying the result by the square of the estimated scale factor.

15.123 `V2C`

`V2C` exchanges the current data with the variance array (if defined). This can be used to allow the variances to be altered. **This is clearly a highly specialist command, so great care should be taken if this command is used.**

15.124 `WAIT`

This is an alternative name for the `PAUSE` command (see Section 15.66, Page 153).

15.125 `WEIGHTED AVERAGE`

Combines two images (one in the memory), both of which **must** have corresponding variance estimates, in the “optimal” manner of the weighted average. The operation is truly optimal only if the variances are correct and the error statistics are independent Gaussian errors. In practical situations neither of these conditions are likely to be true, but with care in estimating the variances, the results should be statistically better than simple addition.

15.126 `X-AXIS LABEL`

Allows the user to enter text for the label of the X-axis of the graphics. This will replace the old text.

15.127 `Y-AXIS LABEL`

Allows the user to enter text for the label of the Y-axis of the graphics. This will replace the old text.

15.128 `Z-AXIS LABEL`

Allows the user to enter text for the label of the Z-axis (intensity) of the graphics. This is the text that will appear alongside the bar showing the current colour table. This will replace the old text.

15.129 Z-SCALE

By default displayed images are automatically scaled to display the complete range of intensities in the displayed region. This behaviour can be modified using `Z-SCALE`. Five Z-axis (intensity) scaling modes are available:

- 0: Automatic full data range (default)
- 1: User set minimum and maximum
- 2: User set minimum, but automatic maximum
- 3: Automatic minimum, and user set maximum
- 4: Automatic "Weak Diffraction peak" scaling

Depending on the scaling mode selected you may be prompted for the minimum and the maximum values to be displayed. Once set the scaling mode continues to operate on all subsequent images, until `Z-SCALE` is used again to change the mode. (This may lead to strange results if you forget that you have set a particular scaling range.)

The `Automatic ''Weak Diffraction peak'' scaling` option allows an adaptive scaling option which is designed to automatically adjust the scaling range to display effectively weak data peaks. This is done by examining the image statistics in the the centre of the image, or all of the image if the *ADR* is small. From the average, the standard deviation and the skewness of the distribution an appropriate display range is calculated. This option is still under development to find the best scaling under a wide variety of circumstances.

As well as automatic or manual selection of the intensity scaling range, the intensity scale may be either linear or logarithmic (V7.25). The user is prompted for logarithmic scaling (the default is "NO"). e.g.

```
LOGARITHMIC IMAGE SCALING [NO]:?
The false colour image display may use either linear intensity scaling
(the default), or logarithmic intensity scaling. Here you can choose
which to use. Enter ''YES'' for logarithmic scaling, ''NO'' for linear
scaling.
LOGARITHMIC IMAGE SCALING [NO]:
```

If logarithmic scaling is used, and the linear scale includes 0.0 or negative numbers, automatic thresholding is applied to avoid the problem of very small numbers. The threshold level depends of the scale being displayed.

16 Calibration Sub-menu

Spatial distortions, intensity non-linearity, flat-field non-uniformity of response, and for image plates the decay of the latent image with time, can be calibrated and corrected. The process of measuring a spatial calibration grid, defining an approximate interpolating function, and of subsequent correction to images is divided into a number of separate operations. This is to allow the maximum of flexibility to cover different detector systems and different approaches to the calibration process e.g. different interpolating functions. The program tries to guide the user through a logical sequence of commands for the spatial distortion calibration process, but this may not always correspond to a user's needs. (Use the "?" to obtain a full list of available commands.)

The calibration and correction process may be divided into two processes:

Distortion Calibration

Distortion Correction

In the case of spatial distortion this means: a. producing a calibration interpolating spline function, b. Using the spline function to correct a distorted image. For the uniformity of response the calibration stage means producing a normalised "flat-field" image (or its inverse), and the correction stage means pixel by pixel division (or multiplication) of the distorted data by the calibrated flat-field image.

The two operations may be performed completely separately; hopefully a detector system may be calibrated once and the measured distortion corrected on many subsequent data images. The correction process may eventually be performed on an on-line processor as the data are being acquired. (The correction process is much simpler than the calibration process.)

16.1 Defining a Spatial Distortion Calibration Function

At present the spatial distortion is measured from a known grid of holes on a calibration mask [8, 30, 34]. The calibration mask should cover the whole of the detector area and should ideally be placed flat against the detector and exposed to X-rays (or other radiation) at some distance from the detector. (The larger the distance the less the problem of parallax.) Alternatively, a point source at the sample position can be used to produce exactly the same parallax as the sample.

The calibration data should be input in the normal manner (See Section 15.51, Page 129). Before entering the calibration sub-menu the *ADR* may need to be defined (using the `REGION` command, the `PIXEL REGION` command, or the `IMAGE` graphics sub-menu). The calibration measurement and function definition is limited to the *ADR*. Experience shows that it is important to inspect the calibration grid image and to define an *ADR* which includes all valid grid peaks, but does not include partially recorded grid peaks or other spurious features[34]. It may

[34]At the edge of imaging plate images it is possible to have a false line of peaks owing to light scatter off the edge. Such a false line of peaks could be mistaken for genuine peaks and lead to false distortion values which can lead to a false distortion function. Partially recorded peaks may also lead to false values for the distortion.

also be useful to use the `Z-SCALE` command to select an intensity range to be displayed.

The normal order of commands for distortion calibration is (commands with a dagger (†) are recommended but not strictly necessary):

`FIND PEAKS`: Find centres of grid peaks as measured on the detector, and estimate initial distortion values

`DISPLAY DISTORTION`†: View (verify) calculated distortion values as 2-D images

`FIT GRID PEAKS`: Fit 2-D interpolating function to measured X and Y distortions

`OUTPUT DISTORTION FUNCTION`†: Save distortion function coefficients in an ASCII file for subsequent correction of data.

Additionally the command `RE-CALCULATE DISTORTION` may be used to change the assumptions used to define the distortion, and the command `VIEW PEAKS` may be useful to view by row and by column the measured peak centres and the calculated distortion. `SAVE PEAKS` may be used to output all the measured peaks to an ASCII file. This can then be used as input for a spread-sheet or graph plotting program for further processing or display.

16.2 Spatial Distortion Correction

The distorted data should be input in the normal manner (See Section 15.51, Page 129). Before entering the calibration sub-menu the *ADR* may be defined if only a sub-region of the data is of interest[35] (using the `REGION` command, the `PIXEL REGION` command, or the `IMAGE` graphics sub-menu `ZOOM IN` graphics command). If only part of the data is of interest, defining a smaller *ADR* will save time. The correction will be applied throughout the *ADR* (provided the interpolating function is valid). If the *ADR* extents outside the valid region of the interpolating function, only the region within these limits will be corrected and the user will be informed of the limits.

Enter the calibration sub-menu using the command `CALIBRATION`. In general the command `INPUT DISTORTION FUNCTION` will be used to recover a previously calculated spatial distortion function. (If such a function has already been defined this is not necessary.)

The correction is applied to the data using the command `SPATIAL CORRECTION`. This re-bins all pixels in the current *ADR* and outputs them in the memory. This process may take several minutes for large images (further optimisation of this process is possible).

Having corrected the spatial distortion the result is in the memory, so after using `EXIT` to leave the calibration sub-menu, you should use `EXCHANGE` to make the corrected data the current data (this is the default option).

[35]Previously it was necessary to limit the *ADR* to the valid region of the interpolating function. This is now performed automatically.

16.3 Non-Uniformity of Sensitivity Response (Flat-Field Correction)

(Note: The approach described to calibrate and correct for the flat-field response has been shown at the ESRF to be able to produce high quality data when used in conjunction with a technique to produce smooth flood-field illumination. Such a technique has been suggested by Jean-Pierre Moy using fluorescence from elements held in an amorphous lithium borate glass [23].)

Owing to geometry, and artifacts of manufacture, the sensitivity of different areas of the detector may vary. This variation may be a low frequency effect e.g. geometry effects, or may be high frequency e.g. pixel to pixel sensitivity differences in a CCD chip. With the assumption that the effect of the point spread function is not important, this can be calibrated by uniform exposure of the detector [35]. It is necessary that the error owing to counting statistics is less than the required accuracy of the calibration e.g. if 1% accuracy is required in the calibration image, then a minimum of 10000 true counts per pixel are necessary.

In practice it is quasi impossible to satisfy the two requirements. Any source which is close enough to give a reasonable flux will not give completely uniform response. If the source is an isotropic point source then the non-uniformity may be a simple geometrical effect. In many cases a point source is required as the sensitivity of the detector will change not only as a function of the detector position, but also as a function of (3-D) source position. In such a case the way to calibrate the detector for a diffraction experiment is to have a point source at the sample position.

Unfortunately, we have not found convenient X-ray sources which are isotropic at the 1% level. Thus, we need to characterise the source distribution. With the assumption that the sources are at least circularly symmetric the intensity distribution of the sources has been characterised using a zero-dimensional detector on the two-theta arm of a diffractometer.

Three different types of flood-field [36] sources have been used, each with advantages and disadvantage:

- Radioactive sources give a reasonably smooth distribution and are easily portable, allowing source rotation to ensure circular symmetric intensity distribution. They offer only low count rates and are only conveniently available in a limited range of energies. Over-night exposures are necessary.
- Amorphous diffuse scatterers can give reasonable count rates at the required energy provided a synchrotron source is available, but often contain unwanted structure in the angular intensity distribution.
- Fluorescent cells containing a solution of the fluorescent material can be very nearly isotropic sources at 90^o to the illumination and can give reasonable count rates with a synchrotron source. However, the energy will always be slightly lower than that of the incoming radiation. We have found that Bragg scattering structure can be minimised by using lithium borate glasses doped with the fluorescent material [23]. These glasses may

[36] The term "flood-field" is used to refer to a smooth, but not completely uniform radiation, whereas the term "flat-field" is used to refer to a completely uniform radiation.

be thin and machined parallel, and may be mounted in the sample position, orthogonal to the beam. As they are solid, there is no problem of evaporation, and there emission distribution does not change with time.

With synchrotron sources polarisation will cause the distributions of any Compton scattering component not to be circularly symmetric, but this effect can be calculated and corrected. Having calibrated the source intensity distribution, the recorded flood-field image may be corrected for source anisotropy, polarisation, geometrical $\frac{1}{r^2}$ fall-off, and any absorption effects.

16.4 Data Collection Sequence for the X-ray Image Intensifier/CCD Detector System

This is an overview of a full calibration/ data collection strategy necessary to obtain quantitative results from the X-ray Image Intensifier/CCD read-out detector system (XRII/CCD) [21, 22] or similar detector system. It is intended as a check list for anyone performing a crystallography experiment. Aspects which require further consideration are discussed in Section 16.5.

The following list is for measurements which may be performed separately from the data collection, before or after.

1a: Calibration measurements to determine intensity non-linearity. At present the ESRF XRII/CCD system appears to be sufficiently linear to avoid the need for correction.

1b: Collect response of CCD camera to uniform light source (illuminated piece of paper). "Correct for dark count" to get CCD uniformity of response function. (Averaging images will improve accuracy.) (Probably performed before the experiment, at the normal operating temperature.)

1c: Place diffuse scatterer/fluorescent cell at the sample position.

1d: Take 1-D 2θ scan of diffuse source/fluorescent cell distribution. At each angular position the counter must be left long enough to reduce counting statistics to the required level. (Scans at other angles than the plane of the synchrotron may be useful if it is possible to obtain these.)

1e: Recording the 2-D source distribution on X-ray film or image plates may be useful to verify the extent to which the intensity distribution is circular symmetric.

1f: Collect reference 2θ scan without diffuse scatterer/fluorescent cell, or with cell containing only solvent.

This is a proposed sequence for efficient collection crystallography and calibration data. This pre-supposes that the detector linearity is known, that the CCD pixel to pixel flat-field response is available, and that a diffuse scatterer with a known angular distribution of intensity for the required wavelength is available. (The order is the result of practical experience.)

2a: Place calibration grid on the detector.

2b: Place diffuse scatterer/fluorescent cell at sample position.

2c: Measure sample to mask distance.

2d: Collect spatial distortion calibration grid image.

2e: Collect flood-field image with diffuse scatterer/fluorescent cell.

2f: Remove diffuse scatterer/fluorescent cell.

2g: Collect reference image for flood-field exposure integration time without fluorescent cell, or with cell containing only solvent.

2h: Collect beam centre image or "wax" image, unless a semi-transparent beam-stop is being used.

2i: Place sample at sample position.

2j: Collect diffraction data images (without changing the magnetic environment e.g. Do not change the position of large metallic objects such as the diffractometer cradle).

2k: Collect "dark count" (empty capillary) images for standard data exposure integration time(s). (Averaging images will improve accuracy.)

2l: If data must be collected in a different magnetic configuration the spatial distortion and ideally the flood-field response should be re-calibrated. (Repeat steps 2a-2h.)

2m: Remove sample

2n: Place diffuse scatterer/fluorescent cell at sample position.

2o: Collect flood-field image.

2p: Collect spatial distortion calibration grid image.

2q: Collect reference image for flood-field exposure integration time without fluorescent cell, or with cell containing only solvent.

(Steps 2n-2q are included to verify if that the spatial distortion has remained constant since it was last calibrated.)

16.5 Notes on the Data Collection Sequence for the XRII/CCD Detector System

16.5.1 Step 1b: CCD Camera Uniformity of Response

Obtaining highly accurate uniformity of response calibration measurements from the system as a whole may be very difficult, however, the high spatial frequency variation will mainly be restricted to the CCD chip, and the other components will generally contribute only low and medium frequency components. Hence, it is useful to calibrate the CCD chip uniformity of response separately (an easier experiment).

The effect of different temperatures and read-out speeds should not be too important. Nevertheless, the read-out conditions might as well be kept identical.

The CCD camera can be illuminated by an uniform source of visible light e.g. an illuminated piece of paper. The exposure camera could be rotated by 90° during the measurement several times to reduce any low frequency unevenness in illumination, or such effects could be removed in software. An appropriate dark count image should be subtracted from the flat-field images, and several flat-field images should be averaged to reduce statistical noise, although this should be less of a problem than for the dark count images.

16.5.2 Steps 1f, 2k: Reference or Dark Count Images

It may be useful to obtain reference images with and without X-rays. Probably reference images which include X-ray background are the best for subtracting from the fluorescent cell measurements. In this manner X-rays not coming from the source position should be eliminated from the flat-field response. Using a fluorescent cell only containing the solvent should allow structure coming from the solvent, e.g. the water ring, to be removed from the fluorescence flood-fields. Some scaling will be necessary to account for the absorption of the fluorescent sample. In the case of the diffraction data, a good reference image will be an empty capillary.

The detector dark count is assumed to vary from pixel to pixel and consist of a constant and a time varying quantity. It is temperature dependent, so the CCD camera should be at the temperature that subsequent data is going to be taken. With the Photometrics camera this means -40°C will be used[37].

The dark count characteristics change as a function of read-out speed, so this should be fixed. Again, with the Photometrics camera the read-out speed is fixed to 200kHz, whereas the Princeton Instruments system allows 50, 100, and 150 kHz.

The dark count images will be subject to statistical noise, which should be relatively appreciable, since (hopefully) the dark count images will not be very strong. To reduce this noise source, the average of a few different images should be used.

The simplest approach is just to integrate dark count images for exactly the same time period as the subsequent data images, and the flood-field images. However, if the build up of dark counts is really as simple as a constant plus a linear time dependent quantity, two different measurements at different time periods would be sufficient to obtain all these parameters for all different exposure lengths. This model could be tested by making dark current measurements for many different integration times, and seeing if the values do fit this simple model.

16.5.3 Steps 2a-2c: Spatial Distortion Calibration Grid

The grid should be carefully placed on the detector in a standard orientation (so that the orientation is reproducible). The sample to mask distance should be measured (as it is effectively the sample to detector distance for the corrected data).

[37]At lower temperatures condensation occurs, unless dry nitrogen is used.

16.5.4 Step 1c-1f, and 2e-2g: Flat-Field Calibration

The aim is to calibrate the the flat-field response of the detector to X-rays coming from the sample position. Owing to the lack of an isotropic X-ray source it is necessary to use a source with a anisotropic distribution and calibrate the distribution with a 2-theta scan of a 0-D detector.

In Steps 1d and 1f a collimator may be used such that the 0-D scintillation counter (or similar detector) is only sensitive to X-rays from the sample position. The counting time at each point should be sufficient, so that counting statistics errors are well below 1% . This will probably require a long scan time, so either the beam decay needs to be monitored with an I_0 monitor, or the scan needs to be carried out in both directions and averaged (more sophisticated treatment would be possible, but averaging is probably quite good enough given the ESRF beam decay[38]).

In Step 2e the flood-field illumination of the XRII will include illumination from sources other than the sample e.g. hutch background. We are not interested in the response from other sources, so we include Step 2f where the flood-field source is removed and only the X-rays from other sources are measured. By subtracting the two images (measured for equal times) the flood-field image we measure is that from the sample point alone (except for secondary scattering by the air in-between the sample point and the detector; this should be negligible).

In subtracting the reference image from the fluorescent cell image we are automatically taking into account the dark current. Again it may be useful to average several images to reduce counting statistics errors.

When correcting the diffraction patterns we will correct an image with both peak and background combined by the flat-field response to the sample position. Since subsequent software will separate the peak from the background, the fact that the background (which does not necessarily come from the sample position) is also corrected by the flat-field response to the sample position is unimportant.

16.6 Calibration Sub-Menu Commands

16.6.1 `?`

List of available commands in the calibration sub-menu together with brief description of their function. (See `HELP`.)

16.6.2 `CALCULATE DISTORTION FUNCTION`

The complete interpolated distortion spline may be calculated for the whole of the *ADR* with this command, provided of course a distortion spline has been fitted, or input. The values of the distortion spline are output in the "memory". This may be useful to check that unwanted oscillations are not present in the fitted spline surface.

[38]This works to the extent that the decay is linear.

16.6.3 DECAY CORRECTION

It has been noted by various investigators that the X-ray signal recorded on an imaging plate decays between the exposure and the read-out by a scanner [4, 6, 9, 25, 29, 33]. This decay can both reduce the measured intensities, and more seriously lead to variable intensity response along the scanned plate. The magnitude of this effect can be 5 to 10 % for short exposures which are scanned soon after exposure, so has an appreciable effect on the integrated data statistics.

The decay of Fuji white and blue image plates has been accurately measured at the ESRF in a long series of calibration measurements by Olof Svensson. The measurements show the decay to be independent of the small temperature variations present in the ESRF Experimental Hall. (Note: The decay curve for the Kodak plates is different.) The decay curve measurements have been fitted by the least squares criterion to the following equation [9]:[39]

$$F_t = 0.13e^{-t/180} + (1.0 - 0.13)e^{-t/6600} \qquad (4)$$

F_t is the fraction of the original recorded signal remaining after a time t in seconds from the exposure.

The user has the option of using this equation with the decay constants specified, and using scan timing results measured for the Molecular Dynamics 400E scanner, or entering their own constants for the exponential decay and the scanner times.

The user is prompted for:

```
FULL USER CONTROL [NO]:
```

If the default of `NO` is entered **FIT2D** will prompt for the choice between 88 μm and 176 μm scan, as this affects the time of scanning:

```
176 MICRON SCAN ("NO" FOR 88 MICRON) [YES]:
```

Otherwise **FIT2D** outputs the basic equation used to approximate the decay and inputs user values for the decay time constants and the proportion of the "fast" and "slow" decay components:

```
The decay is approximated by the sum of two exponential decays. The
fraction is remaining signal t seconds after exposure I_t is defined by:

   I_t = P_fast * Exp(-t/T_fast)+ (1.0 - P_fast) * Exp(-t/T_slow)

where: P_fast is the proportion of the fast decay component
```

[39] Originally a different equation was used, "fitted" by eye; the values produced by the two equations are very similar.

```
       T_fast is the time constant of the fast decay (in seconds)
       T_slow is the time constant of the slow decay (in seconds)

PROPORTION FAST DECAY (Range: 0.0 to 1.00000) [.13426]:
TIME CONSTANT OF FAST DECAY (SECONDS)
 (Range: 0.0 to 1.000E+10) [179.411]:
TIME CONSTANT OF SLOW DECAY (SECONDS)
 (Range: 0.0 to 1.000E+10) [6631.40]:
```

This is followed by an explanation of the times used to calculate the minimum and maximum decay times for each line, and the input of the scanner delay time from the user start (button press) to the start of the physical laser scan of the first line, the time to scan the image, and the number of lines scanned:

```
The program calculates for each line in the image the average decay that
has taken place from the longest time from first exposure to the scan,
to the shortest time from the end of the exposure to the scan. To know
this it needs to know how long was the exposure, the time between the
exposure and the user start of the scan (both these questions are asked
later), the time from the user start of the scan (click with with the
mouse, or similar) to the physical scan of the first line of the image,
and the time to scan each line of the image. The time to scan each line
is calculated from the total time of the physical scan divided by the
number of scanned lines.

START-UP TIME (SECONDS) (Range: 0.0 to 1000.00) [36.0000]:
IMAGE SCAN TIME (SECONDS) (Range: 0.0 to 3600.00) [120.000]:
NUMBER OF SCANNED LINES (Range: 1 to 50000) [1482]:
```

For both `FULL CONTROL` or not the user is prompted for the length of the exposure, and the time from the end of the exposure to the start of the scan (user button pressed). Both times are in seconds.

```
EXPOSURE LENGTH (SECONDS) (Range: 0.0 to 1.000E+10)
 [10.0000]:
ELAPSE TIME (FROM END OF EXPOSURE TO SCAN, SECONDS)
 (Range: 0.0 to 1.000E+10) [60.0000]:
```

From this information the normalised integral of decay is calculated for each line of the scanned image (the difference within a scanned line is insignificant). From the decay values correction values **relative** to the first line of the ADR are calculated and applied to every pixel in the line.

At the end of the correction **FIT2D** will output the maximum correction factor which was applied to the last line to be scanned. This information may be useful in estimating the relative importance of the decay. e.g:

```
INFO: Maximum correction factor =   1.05994
```

The correction takes place in the current data, the old data values are replaced by the corrected values.

Note: If the oscillation method is used to collect crystallographic data, and if the total exposure times are long, then it is important to oscillate the crystals several times during each exposure. This is necessary since the correction assumes that all recorded X-rays have arrived uniformly with time [40].

16.6.4 `DESTROY GRID PEAKS`

Sometimes false peaks may be found on the edge of the grid which can lead to problems in fitting a smooth interpolation function. To overcome this problem `DESTROY GRID PEAKS` allows the user to specific peak positions from the found grid and turn the peak to a missing peaks. **Note: This operation is not reversible** without re-commencing the `FIND PEAKS` operation.

16.6.5 `DISPLAY DISTORTION`

After the `FIND PEAKS` command it is possible to display the measured X or Y-distortion as estimated over the 2-D grid of peaks. This command will display the distortion values as a 2-D image. False colour is used to show the values of the distortion. Each pixel is a peak of the grid. Missing peaks are displayed with a zero distortion value. (Strictly, the measured positions are not regularly spaced, but the deviation from ideal positions is hopefully not too great, so this display should give a reasonable idea of the distortion function over the detector surface.)

(If nothing appears to be displayed in the image it may be because `Z-SCALE` has been used to select a fixed intensity range for display. If this is the case, return to the main menu, set the z-scaling to automatic and return to the calibration sub-menu.)

16.6.6 `EXIT`

To return to the main menu `EXIT` should be entered. As the calibration sub-menu uses extra dynamic memory to store peak positions and the spline coefficients, the user is given the choice of saving these values **or** recovering the memory, which means the values are lost. If you answer `YES` any spline correction coefficients are lost and need to be re-input or calculated.

16.6.7 `FAST CORRECTION`

Corrects data in the current data *ADR* using a look-up table which must have been previously defined using `LOOK-UP TABLE (SPATIAL DISTORTION)` or input from a file using `LOAD LOOK-UP TABLE`.

[40]Phil Evans at the MRC Cambridge has written software to correct for the decay taking into account the time that particular hkl reflections were in the diffracting condition.

This method is much faster than having to re-calculate geometrical areas for every data image, but the stored look-up table is very large so a large amount of computer RAM should be available on the system.

16.6.8 FIND PEAKS

`FIND PEAKS` tries to measure the central position of peaks defined in a regular grid, as the result of a calibration measurement with a grid of known hole to hole distances. The entire *ADR* will be searched for peaks, so prior to `FIND PEAKS` the *ADR* should be set to the required region. Ideally, the *ADR* should just contain all peaks to be measured.

The search is started by the user defining three peaks graphically. A 200×200 region of the *ADR* is displayed as an image and the user is invited to click on the centre of the starting peak, one peak horizontally from the starting peak, and one peak vertically from the starting peak[41]. The size of the displayed region may be changed by prior use of the `SIZE (IMAGE DISPLAY)` option (see Section 16.6.24), Page 200). Clicking on the exact centre is not necessary, but the input coordinate needs to be close to the peak, or the peak searching can get lost.

After selecting an initial starting peak it is necessary to select the peak horizontally to the right of the starting peak, and then the peak vertically upwards from the starting peak[42].

Two parameters are available to allow rejection of noise and spurious features:

`MAXIMUM PEAK SEARCH DISTANCE` and `PEAK DETECTION RATIO`.

`MAXIMUM PEAK SEARCH DISTANCE` sets the maximum distance in pixels from predicted position of a peak to a found "peak" position. Apart from the first three user input peaks, every peak has a predicted position based on the positions of previously found nearby peaks. The predicted position is used for the start of the peak search. If the peak search goes further than the `SEARCH DISTANCE` value from the initial position then the peak is deemed to be missing. This value may be reduced if the peak search is "following" spurious features, or may be increased if the distortion changes quickly relative to the grid peaks and genuine peaks are not being identified.

`PEAK DETECTION RATIO`: This parameter allows rejection of noise and spurious features which have been found within the search distance. This is the ratio of the cross-correlation value found for a new "peak" relative to that of the last accepted peak. (For genuine peaks the cross-correlation values are proportional to intensity.) If too low a value is set noise may be mistaken for peaks, whereas if too high a valid is set genuine peaks may be rejected.

`PEAK STANDARD DEVIATION WIDTH`: Initially, the peaks are assumed to be circularly symmetric 2-D Gaussians in profile. The user must enter the standard deviation size of the peaks in pixel units. (If this value is wrong, or the profile is not Gaussian, the centres should still be correct, so long as the actual profile is also symmetric, and the entered value is about right.) The `FIT` sub-menu may be used to fit 2-D Gaussians to the peaks prior to calibration (See Section 17, Page 208).

[41] Almost any starting peak may now be used, whereas previously it had to be the lower left-hand peak of the grid.

[42] It is possible to skip peaks and have the search only find every other peak or every third peak, but under normal circumstances every peak will be used.

`NUMBER OF SUB-PIXELS`: This sets the precision to which the peak centres should be determined; the number of sub-divisions into which each pixel should be divided when finding the best peak position. e.g. A value of 5 will mean that the centre of each peak will be determined to the best $\frac{1}{5}$ pixel bin in each of the X and Y-directions. The larger the value the longer the peak search will take, and the higher the accuracy provided the data statistics are good enough.

The program will now try to find all peaks in the *ADR* on the regular grid. First a "cross" of peaks is searched. Starting from the initial three peaks all peaks along the positive "horizontal" vector are found, followed by all peaks along the positive "vertical" vector, negative "horizontal" vector, and the negative "vertical" vector. This "cross" is used as starting points for the peak search in each of the four defined quadrants. As such it must not contain any missing peaks[43]. If the "cross" is found to contain missing peaks the peak search stops immediately. Usually an alternative starting position will overcome this problem, otherwise the *ADR* may need to be re-defined. If the central "cross" is found without missing peaks this sets the maximum size of the grid search. Each of the four quadrants is searched using starting estimated peak positions from the "cross". Missing peak positions may be encountered and the search will continue to look for subsequent peaks[44]. The peak search can take many minutes for large grids, so the program will periodically inform you on the progress.

When finished **FIT2D** outputs the number of peaks found and the size of the grid of peaks. Following the grid size is the number of holes in the whole grid as found. From this numbers the number of missing peaks can be calculated. (When the beam-stop covers some holes, or where the edge of the grid is not recorded owing to a circular detector, some missing peaks are normal.)

`GRID SPACING (CENTRE TO CENTRE IN microns)`: In order to calculate the distortion from ideal positions the program needs to know the distance between grid holes (this is the centre to centre distance). The input spacing is in microns. (If this distance is wrong the distortion shape will still be correctly calculated, but the deduced average pixel sizes will be wrong.)

The option to correct peak positions for the effect of off-axis mask vignetting is now given:

```
CORRECT OFF-AXIS MASK VIGNETTING [YES]:
```

Mask vignetting is the shadowing caused by the finite thickness of the calibration mask. (This option was introduced in V7.12.) For off-axis hole positions the X-ray illumination is non-orthogonal to the mask (the mask is assumed to be orthogonal to the direct X-ray beam). Thus the finite thickness of the mask leads to a shadowing of the hole, and the effective centre of the grid peak will move outwards slightly. This option allows this effect to be estimated and the peak positions corrected.

If this option is chosen the user will be prompted for the necessary experiment geometrical information:

```
MASK THICKNESS (microns) (Range: 1.00000 to 1.000E+05)
 [200.000]:
```

[43]This constraint is caused by the software not by any fundamental reason.

[44]The peak search is now much more robust than was previously the case.

```
SAMPLE TO DETECTOR DISTANCE (MILLIMETRES)
 (Range: .10000 to 1.000E+05) [140.000]:
INFO: The beam centre does not need to be specified accurately. An
      approximate centre is appropriate.
X-PIXEL COORDINATE OF BEAM CENTRE [512.500]:
Y-PIXEL COORDINATE OF BEAM CENTRE [512.500]:
```

Note: An approximate beam centre is quite good enough for this correction. The correction should normally be a small effect, and is only worthwhile at short sample to mask distances (¡ 150mm) or with unusually thick masks. Near the centre there will be almost no effect.

Based on the measured peak centres and the input grid spacing, the program calculates average pixel sizes based on the average distance between the furthest peaks on the rows and columns, and based on the RMS pixel sizes found from adjacent peaks. As the grid may not be perfectly aligned to the detector raster, an average rotation is also calculated.

`IDEAL X PIXEL SIZE (MICRONS):` / `IDEAL Y PIXEL SIZE (MICRONS):`
The user is prompted for values for the ideal raster pixel sizes (in microns). These values will be used to calculate the "distortion" values and eventually will be the pixel sizes for spatially corrected images. The RMS X and Y-sizes are given as default values. Depending on the flexibility of subsequent processing software it may be necessary to set square pixels. By default the pixel sizes will in general be non-square.

`GRID ROTATION ANGLE (DEGREES)`: This is the angle of rotation of the grid "horizontal" relative to the detector raster. This avoids grid mask mis-alignment being counted as distortion. By default the average value of mask rotation based on the ends of the columns and rows of peaks is given. For Molecular Dynamics scanner data the value based only on the columns may be better, as the scanner is continual moving in the Y-direction as the X-direction is being scanned.

`X-NUMBER OF IDEAL HOLE:` / `Y-NUMBER OF IDEAL HOLE:` Defines zero distortion reference position. By default the centre of the grid, but the defined grid position must not be a missing peak[45].

These values are used to define initial values for the spatial distortion at each peak position. (The distortion is defined as the ideal position minus the measured position.) (The distortion values can be changed using the `RE-CALCULATE DISTORTION` command.)

It is strongly recommended to use the `DISPLAY DISTORTION` command to verify that the peak searching has produced reasonable results prior to further processing.

Here is an example of the program dialogue:

```
Calibration sub-menu: ENTER COMMAND [FIND PEAKS]:
MAXIMUM PEAK SEARCH DISTANCE (PIXELS) (Range: 1 to 100) [3]:4
PEAK DETECTION RATIO (Range: 0.0 to .99000) [.10000]:.2
PEAK STANDARD DEVIATION WIDTH (Range: 0.0 to 1.000E+05)
 [1.30000]:2.63
NUMBER OF SUB-PIXELS (Range: 1 to 100) [9]:3
```

[45] Again this is not for any fundamental reason.

```
INFO: Starting peaks found O.K.
INFO: Starting peak is at position    474.83334   453.16669
INFO: Next first grid axis peak is at position    508.50000   453.50000
INFO: Next second grid axis peak is at position    475.50000   487.16669
PROGRESS REPORT FREQUENCY (PEAKS) (Range: 1 to 100000) [400]:100

INFO: Starting peak search, this takes some time for big grids
INFO: Found    100  peaks
INFO: Found    200  peaks
INFO: Found    300  peaks
INFO: Found    400  peaks
INFO: Found    500  peaks
INFO: Found    600  peaks
INFO: Number of peaks found =    682
INFO: Number of grid peaks (X/Y) =     28    27 (     756)
INFO: Time taken =    14.     seconds
GRID SPACING (CENTRE TO CENTRE IN microns)
 (Range: 1.00000 to 1.000E+06) [2000.00]:5000
INFO: Average pixel size in X-direction =     131.0585 +-   2.5767 microns
INFO: Average pixel size in Y-direction =     130.6994 +-   3.4362 microns
INFO: Overall average pixel size =     130.8787 +-     3.0057 microns
INFO: RMS pixel size in X-direction =     128.3574
INFO: RMS pixel size in Y-direction =     128.3764
INFO: Average rotation based on rows =     -.324837 +-    .4656 degrees
INFO: Average rotation based on columns =     -.841206 +-    .6721 degrees
INFO: Average rotation (both rows and columns) =
INFO:     -.583021 +-      .6277 degrees
IDEAL X-PIXEL SIZE (MICRONS) (Range: 1.000E-04 to 1.000E+06)
 [128.357]:
IDEAL Y-PIXEL SIZE (MICRONS) (Range: 1.000E-04 to 1.000E+06)
 [128.376]:
GRID ROTATION ANGLE (DEGREES)
 (Range: -360.0000000 to 360.0000000) [-.583021399]:
X-NUMBER OF IDEAL HOLE (Range: 1 to 28) [14]:
Y-NUMBER OF IDEAL HOLE (Range: 1 to 27) [14]:
```

16.6.9 FIT GRID PEAKS

Once peak positions have been found and values of the spatial distortions in the X and Y directions have been calculated, the distortion may be fitted by an interpolating function which will allow the value of spatial distortion to be estimated throughout the *ADR* region. At present two 2-D bi-cubic spline functions are fitted to the measured distortions. The spline functions are defined on an irregular grid of knot points. It should be noted that whilst the holes are defined in a regular grid, their measured positions will no longer be on a grid. (By defining the interpolating function on a grid it will be much more efficient to correct data.)

`FIT ACCURACY`: The user is prompted for the accuracy with which the spline functions should fit the measured positions. This is the required RMS discrepancy between the measured positions and the value of the interpolating spline. This would ideally be equal to the RMS error in peak positions. Typically a starting value of 0.1 pixels is useful. (If the value is too small the spline will fit the noise.)

The routine will find the minimum number of spline knots necessary to achieve the required fit, i.e. the larger the value of the `FIT ACCURACY`, the smoother the function. The number of knot points used to produce the spline is output. It is recommended to keep this number to a maximum of about half the number of grid peaks in each direction. By keeping the number of knot points relatively low the function is kept smooth, and more importantly is better constrained by the data. (If the number of knot points is equal to the number of data points the cubic functions may well oscillate wildly in-between the measured positions.)[46]

16.6.10 FLAT-FIELD CORRECTION

(Note: You should subtract off any dark current, or other non-signal related background prior to this operation. Spatial distortion correction may also be needed prior to this operation, since a flat uniform pixel size image is assumed.)

Ideally an isotropic source would be used to produce a "flat-field" image which could then be used to correct any non-uniformity in detector response simply by dividing by a normalised version. In practice such isotropic sources are unavailable [9, 32, 35]. At present this option corrects an input "flood-field" image to a "flat-field" by inputing a 1-D scan of the source profile from a scintillation counter on a two theta arm or similar experimental set-up [9].

First the user must have an appropriate ASCII file containing the 2θ scan information. The file is read-in in a free format, so the data can be in any vertical column. The scan may have been obtained using the **spec** `ascan`, `dscan`, or other methods at other facilities. Prior to input the scan it may need to be edited to remove undesirable data points. As the data points are to be fitted by a polynomial it is important to remove "erroneous" data points such as those from behind any beam-stop[47].

The program prompts for the file name:

```
Enter name of file containing 1-D 2-theta scan of source
FILE NAME [fe_2.scan]:
```

A sample of the start of the file is output to help the user chose the correct columns for input of the angular and intensity data:

```
Sample of start of the input file:
 -30.0000    39689        0          0
```

[46]At Version 5.7 a bug was fixed (I hope) within NAG routines which could cause **FIT2D** to crash depending on the value of an non-initialised variable.

[47]Here the true value of the function is unknown, but since the data images will also miss this data, the precise values are unimportant. By removing these points the polynomial will simply interpolate in this region. If the data points are not removed a much higher order polynomial would be necessary to fit adequately the sharp feature of the beam-stop shadow. This may introduce unwanted oscillations elsewhere in the fit.

```
 -28.0000    40200        0           0
 -26.0000    40682        0           0
 -24.0000    40892        0           0
 -22.0000    40914        0           0
 -20.0000    40569        0           0
```

A variety of questions allow the user to specify the starting line, and character position for the data, and which columns of data to use for X-coordinates (angle) and the Y-coordinates (intensity). By entering `0` for the number of coordinates to be input the program automatically inputs as many coordinates as possible. Alternatively the number of coordinates to input may be specified.

```
TYPE OF DATA FORMAT [VERTICAL COLUMNS]:
NUMBER OF LINES TO IGNORE (Range: 0 to 1000) [0]:
NUMBER OF CHARACTERS TO IGNORE (Range: 0 to 255) [0]:
NUMBER OF COORDINATES (Range: 0 to 300) [0]:
Columns of numbers for data-set  1
COLUMN NUMBER FOR X-COORDINATES (Range: 0 to 80) [1]:
COLUMN NUMBER FOR Y-COORDINATES (Range: 1 to 80) [2]:
INFO:   31 X/Y coordinates per data-set have been found
```

The intensity values may be corrected for detector dead-time assuming the dead-time is non-paralysable i.e. an event arriving during a dead-time period does not increase the length of the dead-time period [18]. To correct for dead-time the user must enter the count time (integration time) per data value in the 2θ scan. (This is used to calculate the count rate per second from the values input in the scan file, which should be raw total counts.) The user must also enter the detector dead time for a single event. **Note: Some "intelligent" detectors may already take account of dead-time; if this is the case the dead-time correction should be "turned-off" by entering a value of 0.0 seconds.**

For details on methods on calibrating the detector dead-time see [18].

```
COUNT TIME PER DATA POINT (SECONDS)
 (Range: 0.0 to 1.000E+06) [10.0000]:
EVENT DEAD TIME (SECONDS) (Range: 0.0 to .10000) [1.000E-06]:1e-5
```

The original data values are corrected for the dead-time, and will be replaced by higher values (unless 0.0 dead-time was entered).

Note: The input dead-time, and the input count time per data point places a maximum limit on the possible counts in a data point. If a value in the scan exceeds this value, an error message will be output and that value will not be corrected. However, this almost certainly means that all the other values are incorrect. Presumably, either the estimate of the dead-time is incorrect, or the counting time has been entered incorrectly.

The scan data is fitted with a number of different polynomials of different orders, and displays the order and residual value for each polynomial fit order.

```
Order =   0 residual =  6.82410E+02
Order =   1 residual =  6.94029E+02


...


Order =  28 residual =  1.51192E+02
Order =  29 residual =  8.70165E+01
```

The user must choose the order required. The order with the lowest residual (adjusted for number of parameters) is given as a default. However, this is often too high a value. To help decide the fit or the chosen order is displayed graphically each time a different order is entered (e.g. Figure 44, Page 193).

```
ORDER OF POLYNOMIAL (Range: 0 to 29) [30]:10
ACCEPT FIT ("NO" TO TRY ANOTHER ORDER) [NO]:
ORDER OF POLYNOMIAL (Range: 0 to 29) [10]:3
ACCEPT FIT ("NO" TO TRY ANOTHER ORDER) [NO]:
ORDER OF POLYNOMIAL (Range: 0 to 29) [3]:2
ACCEPT FIT ("NO" TO TRY ANOTHER ORDER) [NO]:yes
```

The user should try to find a good fit to the data, but without unwanted oscillations, and be careful not to fit to the noise. When satisfied enter `YES` to the `ACCEPT FIT` prompt.

The centre of the “flat-field” image is specified together with the sample to detector distance and the pixel sizes in the X and the Y-directions[48]. The corresponding angle to the sample position to be calculated for each pixel in the image. If the “flood-field” image has been corrected for spatial distortion the pixel sizes are the corrected pixel sizes.

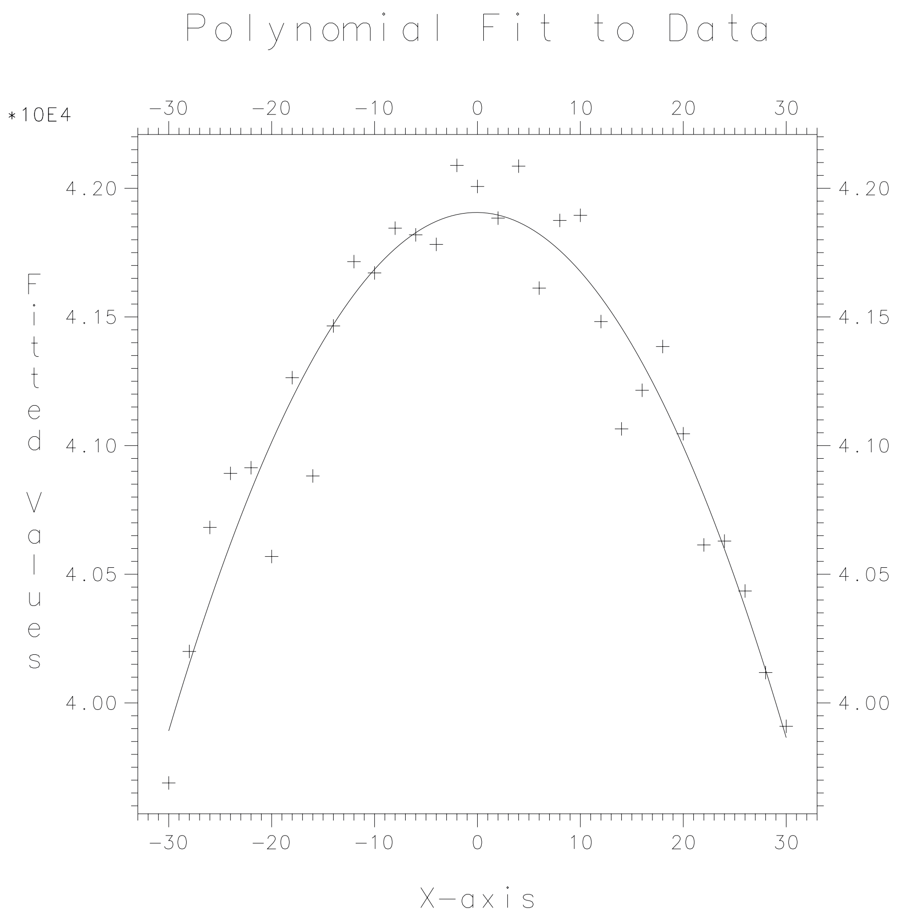
```
FLAT-FIELD CENTRE X-COORDINATE [621.500]:553
FLAT-FIELD CENTRE Y-COORDINATE [576.500]:567
SAMPLE: DETECTOR DISTANCE (METRES)
 (Range: 1.000E-04 to 1.000E+05) [.25000]:.21
PIXEL X-SIZE (MICRONS) (Range: 1.000E-03 to 10000.0)
 [0.0]:131.2
PIXEL Y-SIZE (MICRONS) (Range: 1.000E-03 to 10000.0)
 [0.0]:131.4
```

If significant on-axis absorption occurs there will be an addition correction necessary to account for the increased path length for off-axis positions. Thus, the user is prompted for the on-axis absorption fraction.

```
ON-AXIS ABSORPTION (FOR OFF-AXIS CORRECTION)
 (Range: 0.0 to 1.00000) [0.0]:?
```

[48] This is the sample to mask distance for data which has already been corrected for spatial distortion.

Figure 44: `FLAT-FIELD CORRECTION` Graphics Output



```
Enter fractional absorption; to be used to calculate off-axis absorption
correction. If no off-axis correction is required enter 0.0. If
off-axis correction is required enter the on-axis axis absorption as a
fraction e.g. If the transmission is 80% enter 0.2 for the absorption.
ON-AXIS ABSORPTION (FOR OFF-AXIS CORRECTION)
 (Range: 0.0 to 1.00000) [0.0]:
```

The input "flood-field" data is corrected pixel by pixel for the source profile as a function of angle, $\frac{1}{r^2}$ intensity drop-off, and increased absorption as a function of increased path length with angle. The $\frac{1}{r^2}$ intensity drop-off correction needs to be applied because the 2θ scan is at a constant distance from the source, but the detector is (assumed to be) flat.

The output over-writes the original image.

It may be normalised (`CDIVIDE`) usually with a low value, thresholded (`THRESHOLD`), and perhaps smoothed (`BLUR`) prior to being used for flat-field correction.

As initially produced the resulting "flat-field" correction image should be used for pixel by pixel division (`DIVIDE`) to correct input data[49]. However, it may be preferred to store the reciprocal

[49]The input data should be divided by the "flat-field" image.

of the “flat-field” image using the `RAISE TO A POWER` command with `-1` as the power, and to use pixel by pixel multiplication (`MULTIPLE`).

The following is an example of the use of the `FLAT-FIELD CORRECTION` command:

```
Calibration sub-menu: ENTER COMMAND [FIND PEAKS]:flat
Enter name of file containing 1-D 2-theta scan of source
FILE NAME [fe_2.scan]:Fe_2.scan
Sample of start of the input file:
 -30.0000    39689       0          0
 -28.0000    40200       0          0
 -26.0000    40682       0          0
 -24.0000    40892       0          0
 -22.0000    40914       0          0
 -20.0000    40569       0          0
TYPE OF DATA FORMAT [VERTICAL COLUMNS]:
NUMBER OF LINES TO IGNORE (Range: 0 to 1000) [0]:
NUMBER OF CHARACTERS TO IGNORE (Range: 0 to 255) [0]:
NUMBER OF COORDINATES (Range: 0 to 300) [0]:
Columns of numbers for data-set  1
COLUMN NUMBER FOR X-COORDINATES (Range: 0 to 80) [1]:
COLUMN NUMBER FOR Y-COORDINATES (Range: 1 to 80) [2]:
INFO:   31 X/Y coordinates per data-set have been found
COUNT TIME PER DATA POINT (SECONDS)
 (Range: 0.0 to 1.000E+06) [10.0000]: 5
EVENT DEAD TIME (SECONDS) (Range: 0.0 to .10000) [1.000E-06]:5.0e-5
Order =   0 residual =  6.82410E+02
Order =   1 residual =  6.94029E+02
Order =   2 residual =  1.99863E+02
Order =   3 residual =  2.03530E+02
Order =   4 residual =  2.05713E+02


...


Order =  26 residual =  1.38680E+02
Order =  27 residual =  1.31981E+02
Order =  28 residual =  1.51192E+02
Order =  29 residual =  8.70165E+01
ORDER OF POLYNOMIAL (Range: 0 to 29) [29]:10
ACCEPT FIT ("NO" TO TRY ANOTHER ORDER) [NO]:
ORDER OF POLYNOMIAL (Range: 0 to 29) [10]:3
ACCEPT FIT ("NO" TO TRY ANOTHER ORDER) [NO]:
ORDER OF POLYNOMIAL (Range: 0 to 29) [3]:2
ACCEPT FIT ("NO" TO TRY ANOTHER ORDER) [NO]:yes
FLAT-FIELD CENTRE X-COORDINATE [621.500]:553
FLAT-FIELD CENTRE Y-COORDINATE [576.500]:567
SAMPLE: DETECTOR DISTANCE (METRES)
 (Range: 1.000E-04 to 1.000E+05) [.25000]:.21
```

```
RAW PIXEL X-SIZE (MICRONS) (Range: 1.000E-03 to 10000.0)
 [0.0]:131.2
RAW PIXEL Y-SIZE (MICRONS) (Range: 1.000E-03 to 10000.0)
 [0.0]:131.4
ON-AXIS ABSORPTION (FOR OFF-AXIS CORRECTION)
 (Range: 0.0 to 1.00000) [0.0]:?
Enter fractional absorption; to be used to calculate off-axis absorption
correction. If no off-axis correction is required enter 0.0. If
off-axis correction is required enter the on-axis axis absorption as a
fraction e.g. If the transmission is 80% enter 0.2 for the absorption.
ON-AXIS ABSORPTION (FOR OFF-AXIS CORRECTION)
 (Range: 0.0 to 1.00000) [0.0]:
```

16.6.11 HELP

On-line paged help text on available CALIBRATION sub-menu commands controlled by the "pager" (see Section 14.4, Page 108).

16.6.12 INPUT SPATIAL FUNCTION

Input previously stored fitted spatial distortion function. A fitted spatial distortion function may be recovered for correction of data.

16.6.13 INVERSE DISTORTED/IDEAL

INVERSE DISTORTED/IDEAL command inverses the definition of the spatial distortion function. By default the spatial distortion is defined **from** the distorted grid **to** an ideal grid, with position referring to the distorted grid position. This command toggles the definition, so that after one command the definition is **from** an ideal grid **to** a distorted grid, with position referring to ideal grid position. Using the command twice (or any even number of times) returns to the default definition of the distortion function. When a fitted spline is output an extra line is output for splines which are defined from an ideal grid.

This option is necessary to be able to define a distortion mapping for a program such as **MADNES** [50] which predicts positions on an ideal grid and then apply a distortion value to give the corresponding position on the distorted grid (see Section 16.7, Page 202).

16.6.14 LEARN HOLE PROFILE

After FIND PEAKS has been used to find the centres of grid peaks LEARN HOLE PROFILE may be used to calculate an average hole profile from the found peaks at sub-pixel resolution. The user prompts for the level of the sub-pixel resolution required. The output is in the memory.

[50]**MADNES** actually needs both distortion mappings.

16.6.15 LINEARISE INTENSITIES

LINEARISE INTENSITIES allows the *ADR* intensities to be corrected for intensity non-linearity, provided that an ASCII file is available which contains measurements of raw intensities together with the estimated true linear intensities. (It is beyond the scope of this manual to describe the problems and pitfalls of trying to obtain adequate intensity linearity calibration measurements [9])[51]. At present the data is read-in using a "free format" routine which allows different columns to be selected for the raw and the linearised intensities. Thus the data must be in a column format, but the order or number of columns is not important.

The intensity information is converted to a Log(10) raw intensity scale, and the ratio between the linearised intensity and the raw intensity is calculated. This modified data is fitted by a variety of orders of Chebyshev polynomials. The residual fit statistic for each order is output, and the user is invited to choose a polynomial fit order. The modified data points are displayed together with the fitted polynomial. The user can choose to accept the polynomial order and correct the data, or try a different order of polynomial. Thus, an order which adequately fits the data, but does not exhibit unwanted oscillations between data points can be selected.

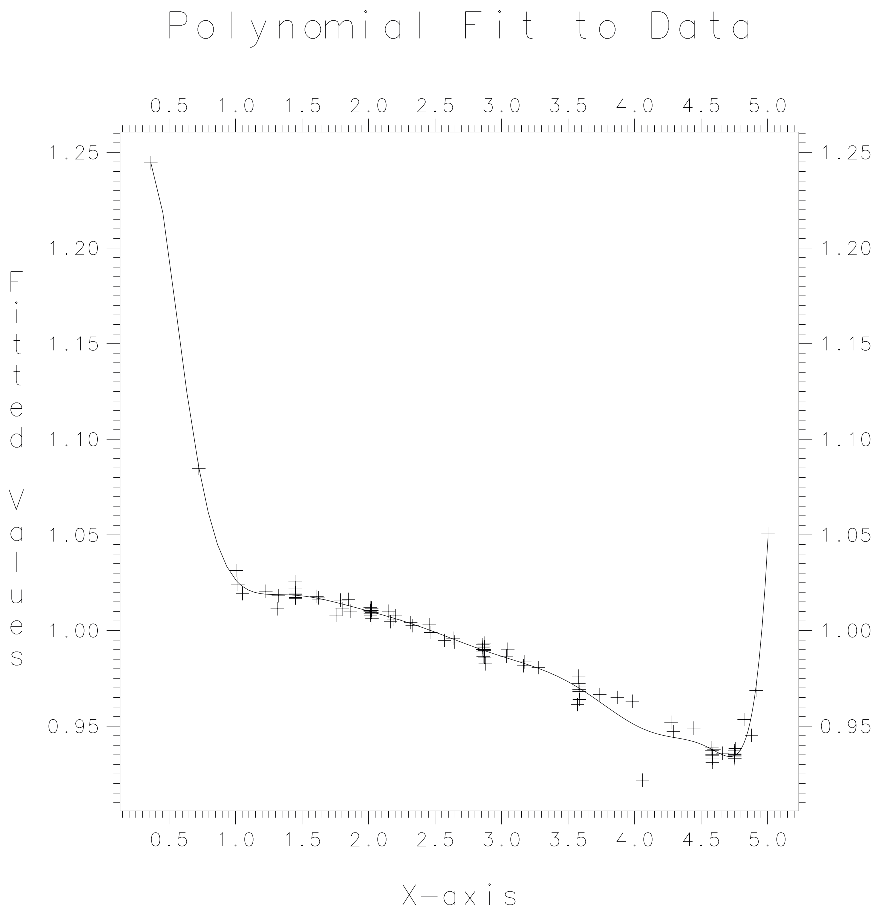
The selected polynomial is used to interpolate between the measured data points when correcting the data. All data values within the range of the defined polynomial (between the minimum and maximum raw intensities in the raw/linearised intensity file) are replaced by the interpolated linearised values. Values outside the range are treated as if the non-linearity function continues outside the range, but is then linear, but offset from the end points. Raw image intensities are replaced in-place by the linearised intensities.

As an example of a non-linearity correction, Figure 45 (Page 197) shows a polynomial fit to data which was produced from a series of linearity calibration tests on an image plate scanner. The program dialogue which produced this graphic and the correction follows:

```
Main menu: ENTER COMMAND [INPUT DATA]:calibration
Calibration sub-menu: enter command [FIND PEAKS]:linearise
INFO: To linearise the data it is necessary to give the name of a file
      which contains values of the raw intensity values together with
      corresponding values on the required (linearised) scale. The raw
      intensity values correspond the X-coordinates and the corresponding
      linearised intensities the Y-coordinates.
Enter name of file containing non-linear and linear intensities values
FILE NAME [source.scan]:linear.dat
Sample of start of the input file:
      56595.5      52929.5
      100770.      105859.
      56674.8      52929.5
      27886.5      26464.8
      56482.0      52929.5
      81965.5      79394.3
TYPE OF DATA FORMAT [VERTICAL COLUMNS]:
NUMBER OF LINES TO IGNORE (Range: 0 to 1000) [0]:
NUMBER OF CHARACTERS TO IGNORE (Range: 0 to 255) [0]:
```

[51] In future versions of the manual references will be given to further papers.

Figure 45: **`LINEARISE INTENSITIES`** Graphics Output



```
NUMBER OF COORDINATES (Range: 0 to 500) [0]:
Columns of numbers for data-set  1
COLUMN NUMBER FOR X-COORDINATES (Range: 0 to 80) [1]:
COLUMN NUMBER FOR Y-COORDINATES (Range: 1 to 80) [2]:
INFO:   96 X/Y coordinates per data-set have been found
Order =   0 residual =  4.15320E-02
Order =   1 residual =  2.45350E-02
Order =   2 residual =  2.25695E-02
Order =   3 residual =  2.12051E-02
```

(Here some program output has been omitted.)

```
Order =  28 residual =  3.45833E-03
Order =  29 residual =  3.43466E-03
Order =  30 residual =  3.43787E-03
ORDER OF POLYNOMIAL (Range: 0 to 30) [31]:7
ACCEPT FIT ("NO" TO TRY ANOTHER ORDER) [NO]:
ORDER OF POLYNOMIAL (Range: 0 to 30) [10]:11
ACCEPT FIT ("NO" TO TRY ANOTHER ORDER) [NO]:y
Calibration sub-menu: ENTER COMMAND [EXIT]:
```

16.6.16 `LOAD LOOK-UP TABLE`

Input a distortion correction look-up table from a disk file. The file must previously have been created using the `STORE LOOK-UP TABLE` command.

16.6.17 `LOOK-UP TABLE (SPATIAL DISTORTION)`

Create a distortion correction look-up table internally from an interpolated spline. Once created this is a much faster way of correcting for spatial distortion, but the internal look-up table is very large so a large amount of computer RAM must be available to **FIT2D**.

16.6.18 `OUTPUT SPATIAL FUNCTION`

Output fitted spatial distortion function to an ASCII file. A fitted spatial distortion function may be input later for correction of data.

16.6.19 `PLATYPUS CORRECTION FILE`

This option allows a calibration file to be created in the correct format for the program **PLATYPUS** [27][52]. `PLATYPUS` needs a 2-D grid of true positions in millimetres corresponding to reg-

[52] A powder diffraction data analysis program written by R Piltz which inputs 2-D data and allows spatial distortion and detector tilt to be taken into account when producing a 1-D circularly averaged powder scan.

ularly spaced pixel positions from a Molecular Dynamics imaging plate scan. The calibration file corresponds to a full A3 image plate even if only an A4 scan was performed.

Because **FIT2D** works with the data looking from the opposite side of the detector (from the sample) as does **PLATYPUS**, and because **FIT2D** works with the data going from bottom to top which is the opposite way round to **PLATYPUS**[53], it is necessary to `FLIP` the full data in both directions prior to the `FIND PEAKS`, `FIT GRID PEAKS`, and `PLATYPUS CORRECTION FILE` commands.

Having fitted a calibration grid image to define the spatial distortion, or after using `INPUT SPATIAL FUNCTION` to recover a previously calculated spatial distortion function, you may use the `PLATYPUS CORRECTION FILE` command to produce a calibration file. You will be prompted with a warning explaining the present need to have reversed the image using the `FLIP` command twice. You must reply "YES", that you understand the requirements to continue producing the file.

FIT2D needs to know where the scan was started using the Molecular Dynamics coordinate system. This system which is used on the controlling PC starts at A1 and each letter or number increment represents 2cm squares. The letters refer to the laser scanning direction and the numbers to the mechanical scan direction. e.g. At the ESRF presently an A4 scan starts at coordinate E5. This may not be the same for other scanners. e.g. The High Pressure group at Daresbury uses A4 plates in a different orientation.

The next input is the name if the output file for the calibration correction.

Below is an example of a log showing the creation of a file called `calib.pos` for an ESRF A4 image plate scan.

```
Calibration sub-menu: ENTER COMMAND [SPATIAL CORRECTION]:platy

IMPORTANT NOTE: The image MUST have been flipped in both
directions using the "FLIP" command.

LIMITATIONS O.K. [NO]:y
MD STARTING COORDINATE [E5]:E5
PLATYPUS CALIBRATION FILE NAME [calib.pos]:
Calibration sub-menu: ENTER COMMAND [EXIT]:
```

16.6.20 QUIT

Same as `EXIT`

16.6.21 RE-CALCULATE DISTORTION

`FIND PEAKS` calculates initial values of the distortion. The distortion values may be re-calculated using `RE-CALCULATE DISTORTION` without needing to perform the peak search again. (See `FIND`

[53]Although visually the data is only left to right reversed.

`PEAKS` for explanation of user prompts.)

16.6.22 `RESIDUALS OF FIT`

Calculates in the "memory" the residuals between the fitted spline function and the measured peak positions for each peak position, for either the X or the Y-distortion values. Missing peaks are given a zero residual.

16.6.23 `SAVE PEAKS`

The user may save the measured peak positions and calculated distortion values to an ASCII file. This can then be used for display by graph plotting programs e.g. **CHIPLOT**, or as entry to a spread-sheet program for further calculation of average pixel sizes, grid rotation, etc.

16.6.24 `SIZE (IMAGE DISPLAY)`

By default the `FIND PEAKS` option displays a 200×200 central region of pixels, for the user to select peaks for the start of the peak search. (This number is compromise between having sufficient detail to be able easily to click on the peaks, and having enough peaks displayed.) If owing to a large beam-stop or another reason the default is not suitable it may be changed with the `SIZE (IMAGE DISPLAY)` command.

The user is prompted for the number of pixels (maximum) in each dimension to be displayed graphically for entry of the initial grid peaks.

16.6.25 `SPATIAL CORRECTION`

Once a spatial distortion interpolation function has been calculated, or a previously calculated function has been input, data may be corrected for spatial distortion. The correction is applied throughout the *ADR*, so `REGION` or `IMAGE` should be used to select the required *ADR* prior to entering the calibration sub-menu.

The user is prompted for an `OVER-LOADED PIXEL VALUE`. This value allows over-loaded pixels intensity not to be distributed below the overload value. Any pixel intensities above the input limit will be added to all the over-lapped output pixels, rather than the intensity being distributed proportional amongst the covered pixels. This means that the output pixels will have at least the overload value so may be easily found and ignored in later processing. A very large value may be entered to turn-off this facility.

```
OVER-LOADED PIXEL VALUE (Range: 0.0 to 1.700E+38) [16382.5]:
```

Next the user must enter the number of rows of the distortion function which will be calculated in a block. The value will not affect the results, but too large a value will require a lot of memory, which may cause problems. To small a number will cause the correction to take longer. The default value should be about right, but every different machine will have a different optimum value depending on the usage and availability of memory.

```
NUMBER OF ROWS OF DISTORTION FUNCTIONS TO CALCULATED IN A BLOCK
 (Range: 2 to 501) [25]:
```

The input pixels are re-binned to output pixels in the memory data image, taking into account the spatial distortion as defined from the interpolating function at the edges of each pixel. The spatially corrected input pixels are assumed to be rectangular on the output grid. The intensity in each input pixel is distributed between one or more output pixels in proportion to the covered area [30].

Owing to the spatial distortion, in general, the size of each input pixel is different. Thus, a uniform input image will not be uniform on output. (By recording the fraction of input pixels added to each output pixel, this non-uniformity could be corrected in the software, but this is not presently the case. It is felt that it is better to correct spatially a flat-field image and use the spatially corrected flat-field image to correct for non-uniformity.)

The corrected image is in the memory at the end of the operation.

Users are kindly asked to cite A P Hammersley, S O Svensson, and A Thompson, “Calibration and correction of spatial distortions in 2D detector systems”, *Nucl. Instr. Meth.*, **A346**, 312-321, 1994 **and generally to acknowledge the use of FIT2D for data corrected by this method.**

Here is an example of the program output produced by the `SPATIAL CORRECTION` command:

```
Calibration sub-menu: ENTER COMMAND [SPATIAL CORRECTION]:
INFO: The corrected pixel dimension in X is     128.4577 microns
INFO: The corrected pixel dimension in Y is     128.4577 microns

OVER-LOADED PIXEL VALUE (Range: 0.0 to 1.700E+38) [16382.5]:?
In order to avoid over-loaded pixels being re-binned and their intensity
spread out to an undetermined value, you can enter a "over-loaded" pixel
value. All input pixels which have this value or more, will cause one or
more output pixels to be incremented by the value regardless of the normal
proportional are re-binning algorithm. Thus over-loaded pixels in the
output image can be easily identified and ignored.
(This can be turned-off by entering a very large value.)
OVER-LOADED PIXEL VALUE (Range: 0.0 to 1.700E+38) [16382.5]:
NUMBER OF ROWS OF DISTORTION FUNCTIONS TO CALCULATED IN A BLOCK
 (Range: 2 to 501) [25]:
INFO: Starting to correct data for spatial distortion
WARNING: One input pixel is trying to be binned into more than three
 output pixels in the X-direction (Pixel    1,    1)
INFO: Number of rows re-binned =    300 ( 60%)
INFO: Time for re-binning =         4.42 seconds
```

```
WARNING:          1 input pixels have been ignored, because they require
 re-binning into more than three output pixels (X-direction).
```

16.6.26 STORE LOOK-UP TABLE

Save a distortion correction look-up table in a named disk file. The internal look-up table must previously have been created using the `LOOK-UP TABLE (SPATIAL DISTORTION)` command.

The `LOAD LOOK-UP TABLE` command can be used to recover a previously stored look-up table.

16.6.27 TRANSFER DISTORTION

The distortion values for either the X or Y-directions may be transferred to the memory through the `TRANSFER DISTORTION` command. The user is prompted for `X-DISTORTION`. Enter "YES" to save the X-distortion values in the memory or "NO" in order to save the Y-distortion values. (Missing peaks are given a zero value for distortion.) By exiting the `CALIBRATION` sub-menu and using the `EXCHANGE` command, the distortion values may be processed (e.g. `STATISTICS`), stored in an output file, etc.

16.6.28 VIEW PEAKS

This command allows the measured positions and calculated distortion values of individual peaks, and rows and columns of peaks, to be output to the screen.

(**Hint:** Use the `OPEN LOG` command in the main menu to save all screen text to an ASCII output file. The output values from rows and columns of peaks may subsequently be displayed using **CHIPLOT** or a spread-sheet program.)

16.6.29 XRII FLAT-FIELD

WARNING: This option is under development and should be presently assumed not to work. The aim is calculate the theoretical flat-field response of the ESRF X-ray Image Intensifier for an arbitrary source position according to a simple model. This model will only be an approximation for the true detector so the theoretical correction will only ever be approximately correct.

16.7 Spatial Distortion Correction in MADNES

The program **MADNES** written by A Messerschmidt and J W Pflugrath [20] performs on-line protein crystallography data collection and integration with auto-indexing and auto-orientation. It includes the possibility to account for detector spatial distortion. Often a spatial distortion calibration and correction system produced by David Thomas [34], or by Marty Stanton [30], is used to correct for the spatial distortion. It is now possible to use **FIT2D** to produce calibration files for **MADNES**. This may have certain advantages:

- Can cope with any detector system (that can be input)
- Can cope with missing grid peaks, owing to the beam stop or to other effects

The version of **MADNES** must have had the following subroutines changed for ones which use the **FIT2D** method of correction:

`lodspd` Load spatial distortion interpolation

`pxtomm` Convert distorted pixel grid to ideal millimetre grid

`mmtopx` Convert ideal millimetre grid to distorted pixel grid

These subroutines can be made available by the author, although some care must be taken to ensure that the code corresponds to the detector read-out system and diffractometer geometry being used (some modification of the code to account for the different coordinate system is likely).

MADNES needs both a function to convert from the distorted detector pixel grid to an ideal detector coordinate, **and** a function to convert from an ideal detector coordinate system to the distorted detector pixel grid. **FIT2D** allows these two functions to be created as interpolation spline functions. Normally (by default) **FIT2D** creates a spline function to convert from the distorted (measured) pixel grid to an ideal grid, as a function of 2-D distorted pixel position. To be able to create an interpolation spline to convert from an ideal detector to a distorted grid detector the command `INVERSE DISTORTED/IDEAL` exists. This command toggles between the two definitions of the distortion. By using the `INVERSE DISTORTED/IDEAL` command once, followed by the `FIT GRID PEAKS` command a spline function is produced for the ideal to distorted grid function. When the `OUTPUT SPATIAL FUNCTION` command is used the file created has an extra first line if it is from an ideal to distorted grid. Thus, **FIT2D** and **MADNES** can recognise the correct files.

MADNES creates automatically the file names to be input containing the two spatial distortion functions, based on a base name plus fixed file extension. The required extension for the file containing the function from the distorted grid to the ideal grid is `.d2i`, and from the ideal grid to the distorted grid is `.i2d`. In addition to the two spatial distortion files **MADNES** needs to know the beam centre in pixel coordinates on the detector. This number is input according to the **MADNES** detector coordinate system[54].

16.7.1 Image Format Conventions

The following is **MADNES** documentation describing the different image storage formats that it can handle.

[54]At present this is experimental and experience may show that other parameters need to be input, or that parameters may be obtained from other sources.

The image DATA can be stored in many orders. MADNES needs to know the order that your images are stored. When one looks at the image from a camera-man's viewpoint, that is from behind the detector towards the crystal, there are 8 possible orders, depending on the origin and the slow and fast directions of pixel order:

```
 +------------+     O-slow->-----+     +-----<-tsaf-O     +------------+
 |            |     |            |     |            |     |            |
 ^            |     f            |     |            s     |            ^
 |            |     a            |     |            l     |            |
 w            |     s            |     |            o     |            t
 o     1      |     t     2      |     |     3      w     |     4      s
 l            |     |            |     |            |     |            a
 s            |     V            |     |            V     |            f
 |            |     |            |     |            |     |            |
 O-fast->-----+     +------------+     +------------+     +-----<-wols-O

 +------------+     O-fast->-----+     +-----<-wols-O     +------------+
 |            |     |            |     |            |     |            |
 ^            |     s            |     |            f     |            ^
 |            |     l            |     |            a     |            |
 t            |     o            |     |            s     |            w
 s     5      |     w     6      |     |     7      t     |     8      o
 a            |     |            |     |            |     |            l
 f            |     V            |     |            V     |            s
 |            |     |            |     |            |     |            |
 O-slow->-----+     +------------+     +------------+     +-----<-tsaf-O
```

Simply for historical reasons, in these routines,
in specifying the detector pixel coordinates:

The FAST direction is called Yms (i.e. is first dimension of a
Fortran array)
The SLOW direction is called Zms (i.e. is second dimension of a
Fortran array)

```
Now for the mm coordinate system.  For the same viewpoint as above,
the X mm points down along the omega axis (vertical for most goniostats,
often horizontal for synchrotrons).  Z goes into the beam and Y makes a
right-handed system. The origin of the mm coordinate system is the
position of the primary beam on the detector, when the tilt (aka swing,
aka theta) angle is zero.

       +-------------+
       |             |
       |             |
       |             |
       |             |
       | Y<-- o      |   Z into beam.
       |      |      |
       |      V      |
       |      X      |
       +-------------+
```

16.7.2 Example of Producing Calibration Files

The following is part of the log produced by **FIT2D** showing the sequence of commands used to produce the two calibration files suitable for use by **MADNES**. Here is it assumed that the base file name for the calibration files will be `gridi20`.

```
Main menu: ENTER COMMAND [INPUT DATA]:
FILE FORMAT [FIT2D STANDARD FORMAT]:princeton
DATA FILE NAME [grid600.spe]:gridi20.spe
INFO: Size of image =  1242 *  1152
Main menu: ENTER COMMAND [IMAGE]:calibration
Calibration sub-menu: ENTER COMMAND [FIND PEAKS]:size
NUMBER OF PIXELS (Range: 50 to 5000) [200]:400
Calibration sub-menu: ENTER COMMAND [FIND PEAKS]:
MAXIMUM PEAK SEARCH DISTANCE (PIXELS) (Range: 1 to 100) [3]:
PEAK DETECTION RATIO (Range: 0.0 to .99000) [.10000]:
PEAK STANDARD DEVIATION WIDTH (Range: 0.0 to 1.000E+05)
 [1.30000]:
NUMBER OF SUB-PIXELS (Range: 1 to 100) [9]:5
INFO: Starting peaks found O.K.
INFO: Starting peak is at position    492.29999   469.69998
INFO: Next first grid axis peak is at position    519.30005   470.29999
INFO: Next second grid axis peak is at position    492.50000   496.29999
PROGRESS REPORT FREQUENCY (PEAKS) (Range: 1 to 100000) [400]:100

INFO: Starting peak search, this takes some time for big grids
```

```
...

INFO: Found   1000  peaks
INFO: Number of peaks found =   1001
INFO: Number of grid peaks (X/Y) =     34    34 (    1156)
INFO: Time taken =    .21E+03 seconds
GRID SPACING (CENTRE TO CENTRE IN microns)
 (Range: 1.00000 to 1.000E+06) [2000.00]:5000
INFO: Average pixel size in X-direction =     162.6968 +-   2.7524 microns
INFO: Average pixel size in Y-direction =     162.6369 +-   2.6746 microns
INFO: Overall average pixel size =     162.6669 +-     2.6770 microns
INFO: RMS pixel size in X-direction =     158.8293
INFO: RMS pixel size in Y-direction =     159.0563
INFO: Average rotation based on rows =      .473566 +-   .1525 degrees
INFO: Average rotation based on columns =      .583813 +-   .1607 degrees
INFO: Average rotation (both rows and columns) =
INFO:      .528689 +-      .1643 degrees
IDEAL X-PIXEL SIZE (MICRONS) (Range: 1.000E-04 to 1.000E+06)
 [158.829]:
IDEAL Y-PIXEL SIZE (MICRONS) (Range: 1.000E-04 to 1.000E+06)
 [159.056]:
GRID ROTATION ANGLE (DEGREES)
 (Range: -360.0000000 to 360.0000000) [.528689305]:
X-NUMBER OF IDEAL HOLE (Range: 1 to 34) [17]:
Y-NUMBER OF IDEAL HOLE (Range: 1 to 34) [17]:
Calibration sub-menu: ENTER COMMAND [DISPLAY DISTORTION]:fit
AVERAGE FIT DISCREPANCY (PIXELS)
 (Range: 1.000E-05 to 1000.00) [.10000]:.2
INFO: Fitting X-Distortion
INFO: X-Distortion Spline
INFO: Number of horizontal/ vertical true knot points =     6    5
INFO: Average position discrepancy =  1.99996E-01
INFO: Fitting Y-Distortion
INFO: Y-Distortion Spline
INFO: Number of horizontal/ vertical true knot points =     5    6
INFO: Average position discrepancy =  1.99989E-01
INFO: Worst case fit discrepancies  in pixels (spline value - measured)
INFO: Lowest X-distortion =   -.82921E+00 at grid position (X/Y)   32   6
INFO: Highest X-distortion =   .73063E+00 at grid position (X/Y)   27   2
INFO: Lowest Y-distortion =   -.11955E+01 at grid position (X/Y)   17  15
INFO: Highest Y-distortion =   .11901E+01 at grid position (X/Y)   16  18
Calibration sub-menu: ENTER COMMAND
 [OUTPUT SPATIAL FUNCTION]:
Name of file for spatial distortion interpolation function
FILE NAME [spatial.dat]:gridi20.d2i
```

The spatial distortion function from the distorted grid to an ideal grid has now been defined and saved, so it remains to change the definition of the peak positions and distortion values with the `INVERSE DISTORTED/IDEAL` command and to fit a new spline function.

```
Calibration sub-menu: ENTER COMMAND [EXIT]:inverse
INFO: Distortion now defined from ideal grid to distorted grid
Calibration sub-menu: ENTER COMMAND [FIT GRID PEAKS]:
AVERAGE FIT DISCREPANCY (PIXELS)
 (Range: 1.000E-05 to 1000.00) [.20000]:
INFO: Fitting X-Distortion
INFO: X-Distortion Spline
INFO: Number of horizontal/ vertical true knot points =     7    5
INFO: Average position discrepancy =  2.00015E-01
INFO: Fitting Y-Distortion
INFO: Y-Distortion Spline
INFO: Number of horizontal/ vertical true knot points =     7    8
INFO: Average position discrepancy =  2.00044E-01
INFO: Worst case fit discrepancies  in pixels (spline value - measured)
INFO: Lowest X-distortion =   -.88624E+00 at grid position (X/Y)   33  22
INFO: Highest X-distortion =   .10403E+01 at grid position (X/Y)   33   9
INFO: Lowest Y-distortion =   -.87807E+00 at grid position (X/Y)   16  18
INFO: Highest Y-distortion =   .95938E+00 at grid position (X/Y)   17  15
Calibration sub-menu: ENTER COMMAND
 [OUTPUT SPATIAL FUNCTION]:
Name of file for spatial distortion interpolation function
FILE NAME [gridi20.d2i]:gridi20.i2d
Calibration sub-menu: ENTER COMMAND [EXIT]:
```

17 Fit Sub-Menu

The FIT sub-menu contains commands for model fitting to 2-D data, and also a number of other commands not particularly related to fitting. The common feature of the other commands is the possibility to "mask-off" areas of the data. Thus, commands related to powder diffraction are included in the FIT sub-menu.

Model fitting can range from a relatively simple to a very complicated process, owing to the problems of local minima, poor initialisation, saddle points on the fit surface, etc. This sub-menu contains commands which allow initialisation of a fit model, minimisation, inspection of results, and the setting of constraints of model parameters. Novice users should seek advice from users more experienced with fitting if attempting more than the simplest operations [5].

The basic commands for fitting are:

`INPUT PARAMETERS`: Initialisation of a fit model by graphical and keyboard input.

`MINIMISATION`: Optimisation of fit model by the maximum descent method

`RESULTS`: View fitted parameters

The memory is used to calculate the fit model, so at the end of `MINIMISATION` it is possible to exit the `FIT` sub-menu and use `EXCHANGE` so that fit model can be examined. (`EXCHANGE` is the default command after exiting.) Similarly `SUBTRACT` can be used to create the residuals.

Additionally, there are many other commands which support the fitting process and allow other forms of fitting. The commands are:

`CHANGE SCALE`: Change the typical parameter variation sizes

`CLEAR MASK`: Eliminate all masked-off "bad" data points

`CONSTRAIN`: Set constraints, change parameters values

`COVARIANCE`: Output covariance matrix

`DEFINE MASK`: Mask-off "bad" data prior to fitting

`DISPLAY MASK`: Display masked-off data as a two level image

`EXIT`: Exit fit sub-menu

`INPUT PARAMETERS`: Initialisation of fit model

`MASK STATISTICS`: Statistics of mask

`MINIMISE`: Search for better fit by maximum descent method

`MODEL`: Create fit model (in memory) from current parameter values

`NORMALISATION`: Uni-directional flat-field normalisation

`OUTPUT PARAMETERS`: Output fit parameters to a file

`POWDER DIFFRACTION`: Convert 2-D data into a 1-D "scan"[55]

`QUIT`: Exit fit sub-menu

`R/THETA RE-BINNING`: Radial/Angular (polar) re-binning of ADR

`RADIAL PROFILE`: Calculate 1-D radial profile in memory

`RESULTS`: View results of fitting

`SET MASK COLOUR` User choice of colour to draw masked pixels

`SET UP`: Alter fit control parameters

`SURFACE POLYNOMIAL`: Fit data using 2-D Chebyshev polynomial

`THRESHOLD MASKING` Mask elements depending on ADR data values

`TILT/BEAM CENTRE` Determine tilt and beam centre from powder rings

`TRANSFER MASK TO MEMORY` Set masked elements to 1.0 in memory

All the available commands are described in detail in the following sub-sections, but first some very important general concepts are introduced.

17.1 Powder Diffraction Ring Integration

Included in the `FIT` sub-menu are a number of commands concerned with integrating 2-D images of powder diffraction rings to 1-D "scans". These commands have been placed within the `FIT` sub-menu since they all allow arbitrary regions of the data to be "masked-off" and therefore not be included in the integration. Thus all the commands relating to defining or un-defining the "mask" are relevant to powder diffraction, as are the commands `TILT/BEAM CENTRE` and `POWDER DIFFRACTION`.

`TILT/BEAM CENTRE` allows any non-orthogonality of the detector with respect to the beam to be fitted from the shape of one or more powder rings. Similarly the beam centre on the detector can be refined.

From the tilt and beam centre found by `TILT/BEAM CENTRE` or input by the user, the ADR may be re-binned to a 1-D equal angle or equal radial distance bin scan using the `POWDER DIFFRACTION` command. This allows the option or simultaneously correcting spatial distortion.

Users of these commands are kindly asked to acknowledge and cite:

> A P Hammersley, S O Svensson, M Hanfland, A N Fitch, and D Häusermann, "Two-Dimensional Detector Software: From Real Detector to Idealised Image or Two-Theta Scan", *High Pressure Research*, **14**, pp235-248, (1996)

[55]This option is under development.

(Note: Other options to allow more flexible integration as a function of azimuth are under development. At present `R/THETA RE-BINNING` performs a polar transformation to the ADR, with user control of the number and size of radial and azimuthal pixels.)

17.1.1 Correction for Polarisation Effects

The commands `POWDER DIFFRACTION` and `R/THETA RE-BINNING` re-bin 2-D powder rings to one or more 1-D spectra. In order that the relative intensities are correct it is important to consider both the effect of polarisation and the Lorentz correction factor.

In general the effect of polarisation is dependent on both the 2θ angle and the azimuthal angle of diffracted radiation. Thus, it is important to correct for polarisation when the data is being converted from a 2-D image to one or more 1-D spectra. If the polarisation correction is to be applied the user will be prompted for the polarisation degree of the X-ray beam and the geometry of the experiment. The beam polarisation is defined as:

$$P = \frac{(I_h - I_v)}{(I_h + I_v)} \tag{5}$$

where: I_h is the horizontal component of the intensity, and I_v is the vertical component.

At a synchrotron the X-rays are usually highly horizontally polarised, so $P \approx 1.0$, but the beam-line scientists should be able to supply an appropriate number for their beam-lines.

The correction applied is based on the formula of Kahn [16].

Having corrected for polarisation it is important that subsequent software does not also apply a polarisation correction !

17.2 Advice on Model Fitting

Finding a optimum fit between a parameterised model and measured data can be a difficult task owing to the problem of local minima, or simply owing to the size of the computational problem being undertaken. To avoid local minima and to reach quickly the global minimum good initialisation is vital. Great care should be taken with the the initialisation process. Many of the commands in the main menu may be useful to obtain good estimates of model parameters. The `CONSTRAIN` command may be used to set parameter values individually.

The fewer variable parameters, the faster will be the fitting. Thus if certain parameters can be given a good initialisation, it will be useful to constrain these parameters whilst optimum values are found for the remaining parameters.

If the model makes the fit much more sensitive to variation in one parameter than the others, it may completely dominate the minimisation process, such that the other parameters are not optimised. The `CHANGE SCALE` command allows this problem to be reduced (See Section 17.4, Page 211).

17.3 The Data Mask

Allowing elements to be arbitrarily included or excluded from fitting and other operations is a very powerful facility which allows much flexibility in the data analysis process. "Bad" data points can be excluded from operations. Optimum non-rectangular data regions can be selected reducing calculations for fitting and other calculation intensive operations.

The "mask" is a 2-D area of the same size as the current data and memory arrays. Each data element may be "masked" or not. By default no elements are masked. The command `DEFINE MASK` (See Section 17.8, Page 212) allows masked-off regions to be defined, and the command `INPUT PARAMETERS` (See Section 17.11, Page 213) also calls the `DEFINE MASK` routine. The command `CLEAR MASK` (See Section 17.5, Page 211) resets all elements to "unmasked".

Alternatively the "masked-off" areas can be defined from the data values themselves by using the `THRESHOLD MASKING` command. This allows a threshold value to be defined and all data elements with values above, or below (as required), this value are defined as "masked-off" (see Section 17.25, Page 219). The mask can be "transferred" to the memory for further use and possible storage using the `TRANSFER MASK TO MEMORY` command (see Section 17.27, Page 223).

17.4 `CHANGE SCALE`

The minimisation method needs to calculate numerical derivatives. To estimate the descent direction, small variations are tried in each of the variable parameters. If the fit is far more sensitive to a small change in one parameter than the others, then it will only really search for the best fit for that parameter, whilst the others effectively remain constrained. To avoid this problem, each parameter has a scale size of variation. Thus, sensitive parameters have a small scale size and so are only varied by a small amount. When the parameters are input, default scale sizes are set. These should be suitable for most purposes, but sometimes better minimisation will be obtained by altering the default values. This can be seen at the start of the `MINIMISATION` process. The first $n+1$ fit (reduced chi-squared) values correspond to the n variable parameters of the fit model. If one shows much greater variation than the others, its scale size is too large.

The `CHANGE SCALE` command allows the fit parameter scale sizes to be examined and values to be changed. This command itself is a sub-menu with the following commands:

`EXIT`: Return to `FIT` sub-menu

`MODIFY`: Make a change to one or more parameters. The number of the parameter to alter, followed by its new value.

`VIEW`: Display one or more parameter values, together with their name and scale size.

17.5 `CLEAR MASK`

Resets all mask elements so that the data points are "good".

17.6 CONSTRAIN PARAMETERS

The `CONSTRAIN` command allows the fit parameters to be examined, values changed, and constraint of parameters. This command itself is a sub-menu with the following commands:

`EXIT`: Return to `FIT` sub-menu

`MODIFY`: Make a change to one or more parameters. The number of the first parameter to alter and the number of the last parameter to alter are to be specified (0 means no parameter), followed by one of the following: `CONSTRAIN, SET, UNCONSTRAIN`. `SET` asks for the new value of the parameter.

`VIEW`: Display one or more parameter values, together with their name and constrained or unconstrained state.

17.7 COVARIANCE

(This option has not been re-installed since the move to Unix, but may be installed quickly if a need arises).

Displays covariance matrix as calculated by `MINIMISATION` fitting option.

17.8 DEFINE MASK

(For a general description of the "data mask" concept see Section 17.3, Page 211.)

This command is the same as the GUI interface **MASK** command. See Section 5.7, Page 56 for further details.

17.9 DISPLAY MASK

Displays the data as a pixel image, but with all the masked elements being displayed in a uniform colour. The colour used to displayed the masked-off pixels may be changed using the `SET MASK COLOUR` command (see Section 17.22, Page 218).

(As the mask is displayed as an image, all the normal attributes for image display will apply.)

17.10 EXIT

Leave fit sub-menu and return to main menu. (No information is lost so it it possible to return to the fit menu at a later stage.)

17.11 INPUT PARAMETERS

INPUT PARAMETERS allows an initial fit model to be defined. The fit model can be defined GRAPHICALLY or from a FILE, which has previously been saved using OUTPUT PARAMETERS. Selecting INPUT PARAMETERS re-initialises the fit model so any previously defined fit model parameters are lost (prior to this operation values may be saved with OUTPUT PARAMETERS).

17.11.1 GRAPHICAL INPUT

This is the method to define an initial fit model. First any masked-off elements are defined (See Section 17.8, Page 212).

The fit model can be made up of a general polynomial background plus one or more 2-D peaks. The order of the polynomial in the X and Y-directions is entered by the keyboard, followed by the a number of points used to define initial values for the polynomial. The initial polynomial may be defined from a lower order, which is more stable. The higher coefficients of the polynomial are then initialised to zero. To define the initial coefficients it is necessary to click on a given number of data points e.g. $(n + 1) \times (m + 1)$ where n and m are the orders of the initialisation polynomial. It is recommended to use only lower order polynomials, since higher order polynomials are not very stable. The user should click on the given number of data points spread throughout the data region, in an approximate grid. (If the points are all co-linear it will not be possible to set initial coefficient values.)

A choice of three peak types is available. A menu appears presenting the three types. the user should click on the required peak type, or on exit to finish this stage of the model definition. The three peak types are:

GAUSSIAN: 2-D Gaussian profile peak

POLAR GAUSSIAN 2-D Gaussian profile peak defined in a polar geometry i.e. Gaussian profile in a radial sense and in an angular arc

TWIN POLAR GAUSSIAN 2 polar Gaussians defined symmetrically about a centre of symmetry

For each peak a number of peak parameters need to be defined. These are all entered graphically by clicking on the centre of the peak (position and maximum intensity), and two half height positions. Care should be taken with the initialisation or greater danger of falling into local minima will be present.

Finally the user may specify a "row-line" as part of the model. A "row-line" is a line of polar Gaussians which all share the same radial and angular standard deviations. The peaks are placed at equal angles along a line at an angle from the vector from the symmetry centre to the "row-line" centre. The "row-line" is reflected in the vector from the symmetry centre to the row-line centre, but with a change in intensity. All the parameters which define a "row-line" are entered with the keyboard. **(It is planned to define a tri-clinic unit cell diffraction pattern peak model to replace the "row-line" model which is an approximation.)**

17.12 MASK STATISTICS

Outputs to the screen the statistics of masked and un-masked elements within the current *ADR*.

17.13 MINIMISE

Once a fit model has been defined it may be optimised (minimised) using `MINIMISE`. This calculates numerical derivatives of the least squares surface defined by the fit model, and tries to find a minimum using the maximum descent vector. The NAG routine E04FCF is used to perform the minimisation so NAG documentation may be used for further explanation.

The user may enter an estimate of the total "distance" from the initialisation model to the best fit model. (The default values seem to be fine.)

The fitting is weighted by the inverse of the estimated variances of the data points if variances exist and weighted fitting has not been turned off. If variances have not been defined the fitting is un-weighted.

The program outputs the fit value for each iteration. This is a reduced chi-squared value for weighted fitting, and an average squared residual for un-weighted fitting.

The NAG routine is likely to end will one of several "error" messages, unless it thinks the maximum has been obtained. Depending on the "error" message more iterations may be necessary to find the minimum point.

The program uses the memory to create the fit model at each stage, so any previous memory data is lost. At the end of `MINIMISATION` it is possible to return to the main menu, use `EXCHANGE` (the default option), and examine the fitted model.

17.14 MODEL

Creates fit model, based on currently set parameter values, in the memory. This may be useful when parameters values have been changed using the `CONSTRAIN` command.

17.15 NORMALISATION

`NORMALISATION` allows non-uniform sensitivity response to be corrected for detector systems which suffer from non-uniform response in a manner which is equal (or at least similar) for all rows or all columns.

The method is to mask-off all data except for the background data which are (ideally) assumed to be approximately flat. The program adds up all the values in each column or row, and produces a 1-D normalisation array. This array is then used to correct all the data in each column or row. The output is in the memory. The user can select the direction of the correction.

(Clearly this is only an approximate correction, and if available a calibration image should be used to apply the flat-field correction.)

17.16 OUTPUT PARAMETERS

Once defined, and at any stage of the fitting process, fit model parameters may be saved to file.

17.17 QUIT

Same as `EXIT`.

17.18 POWDER DIFFRACTION

`POWDER DIFFRACTION` converts 2-D powder rings from a flat detector to 1-D profiles whilst taking account of spatial distortion and detector tilt [10]. The command `TILT/CENTRE BEAM` (see Section 17.26, Page 220) can be used to determine the tilt and optionally the beam centre. If a semi-transparent beam-stop is used the beam centre can be found be fitting a suitable function e.g. 2-D Gaussian (see Section 17, Page 208). Other commands may also be useful e.g. the `PIXEL X/Y` option within the `DISPLAY` sub-menu of the graphics menu of the `IMAGE` command (see Section 3.5, Page 25).

A choice of equal orthogonal detector pixel distance or equal diffraction angle bins is available.

The re-binned 1-D profile can be output into a "Powder Diffraction Standard" (PDS) format file[56] within this command, or will be saved in the memory for further manipulation or output in a alternative form[57].

Below is the log file from **FIT2D** of the present user interaction. The documentation will be fully written when the algorithm and user inputs are stable:

```
Main menu: ENTER COMMAND [INPUT DATA]:fit
Fit sub-menu: ENTER COMMAND [INPUT PARAMETERS]:powd
INFO: The current pixel coordinates for the beam centre = 256.500 256.500
INFO: The current pixel sizes (microns) =    100.000   100.000
INFO: The current sample to detector distance (millimetres) =    315.000
INFO: The rotation angle of the tilt plane =     .000
INFO: The tilt angle of the detector =     .000
CHANGE BEAM CENTRE AND/OR TILT VALUES [NO]:
CORRECT FOR X-RAY BEAM POLARISATION [YES]:
BEAM POLARISATION (AT SAMPLE) (Range: -1.00000 to 1.00000)
 [.99000]:?
```

[56]The PDS format is a simple ASCII format devised by Andy Fitch and Andrew Murry for the storage of angular scan data for input to Rietveld refinement programs.

[57]It is necessary to output the PDS format data here, as the number of pixels averaged to create one angular bin is not available after this option is finished

```
Enter the "polarisation" of the main beam on the sample. This is defined as
(I_h - I_v) / (I_h + I_v),  where I_h is  the horizontal component  of  the
intensity and I_v is  the vertical.  (The horizontal should correspond with
the X-direction on  an image.  Normally for  a synchrotron the polarisation
is positive and approaches 1.0.  e.g. Station 9.6 at  the SRS Daresbury has
been measured to  have  a polarisation of 0.86.  This  is  a value which is
dependent on the X-ray source, beam-line mirrors, and on the monochromator.
The beam-line scientist should  be  able  to  give  a good estimate of this
number.
BEAM POLARISATION (AT SAMPLE) (Range: -1.00000 to 1.00000)
 [.99000]:.95
TYPE OF LORENTZIAN CORRECTION TO APPLY [NONE]:?
Enter the type of "Lorentz" correction which you want to apply to the
output intensities. At present the following choices are available:

   NONE: No correction factors applied
   PARTIAL POWDER (2-THETA SCAN): Correct intensities to be equivalent to
      a 2-theta scan with a single counter. This allows standard powder
      diffraction software to apply their own Lorentz corrections.
TYPE OF LORENTZIAN CORRECTION TO APPLY [NONE]:part
PRODUCE EQUAL ANGLE PIXEL SCAN [YES]:?
Enter: "YES" if you want to produce a 1-D scan as a function of equal
       angle pixels.
       "NO" if you want to produce a 1-D scan as a function of equal
        radial distance pixels (on an orthogonal detector).
(The two are not the same.)
The best choice will depend on the capabilities on any further
processing software. e.g. If a Rietveld program requires equal angle
pixels then this is clearly the choice. If either type of scan may be
treated then the equal radial re-binning is probably preferable.
PRODUCE EQUAL ANGLE PIXEL SCAN [YES]:
2 THETA SCAN ANGULAR PIXEL STEP (DEGREES)
 (Range: 1.000E-03 to 10000.0) [1.800E-02]:?
Enter required angle step between bins for the calculation of the 1-D
2 theta scan. (The default value corresponds to the angular size of
the average pixel size at the beam centre.)
2 THETA SCAN ANGULAR PIXEL STEP (DEGREES)
 (Range: 1.000E-03 to 10000.0) [1.800E-02]:
TAKE ACCOUNT OF SPATIAL DISTORTION [NO]:?
YES: if a the detector spatial distortion has been characterised and is
to be taken into account. If you answer "YES" you will be required to
input the name of a valid spatial distortion interpolation file.
TAKE ACCOUNT OF SPATIAL DISTORTION [NO]:
INFO: Starting to re-bin 2-D data to a 1-D profile, this can take some
      time for large arrays.
INFO: Number of rows treated =    300 ( 58%)

INFO: Minimum fractional intensity decrease owing to polarisation = .96
INFO: (The reciprocal value is applied to the data.)
```

```
SAVE DATA IN "POWDER DIFFRACTION STANDARD" FORMAT [YES]:
Enter name of output file
FILE NAME [scan.pds]:
```

Note:

- The default pixel size for equal angular pixel size scans is an exact multiple of one thousandth of a degree (V7.37). This is because the PDS format saves the angular step to this precision. Since thousands of scan bins may be output, a small error can build up to be relatively large angle. It is recommended to only use pixel bin sizes which are exact multiples of a thousandth of a degree when outputting data to PDS files.
- For the Polarisation correction to be correct, the image must have been input in the correct orientation. i.e. The horizontal on the displayed image should correspond with the horizontal on an upright detector in line with the direct beam.
- The "Lorentz" correction allows data from a flat 2-D detector, which has been re-binned into a 1-D "2θ scan" to be corrected for geometrical differences between this scan and a true 2θ scan (V7.31). Following this correction, the normal Lorentz corrections present in Rietveld refinement and other refinement programs should be appropriate.

17.19 R/THETA RE-BINNING

`R/THETA RE-BINNING` transforms a 2-D Cartesian coordinate system pixel image into a polar coordinate system pixel image. There is arbitrary choice of the number of radial and angular pixels.

Spatial distortion, beam centre, and detector tilt are taken into account. The command `TILT/CENTRE BEAM` (see Section 17.26, Page 220) can be used to determine the tilt and optionally the beam centre. If a semi-transparent beam-stop is used the beam centre can be found be fitting a suitable function e.g. 2-D Gaussian (see Section 17, Page 208). Other commands may also be useful e.g. the `PIXEL X/Y` option within the `DISPLAY` sub-menu of the graphics menu of the `IMAGE` command (see Section 3.5, Page 25).

By varying the number of radial and angular bins this command allows anything from a 1-D radial profile, to a 1-D azimuthal profile of a diffraction ring, to a number of radial profiles as a function of angle, to be produced.

17.20 RADIAL PROFILE

Creates a 1-D radial profile from un-masked data in the current *ADR*; output is in the memory. The centre of symmetry is specified, as is the required size of profile pixels. The average values of pixels at each distance range from the centre are calculated.

If variance arrays exist, estimates of the variance for each radial pixel position will be calculated. This is based on the variation of values which contribute to each value. When more than 20 pixels contribute to the radial average the variance (σ^2) is estimated from the normal equation

$$\sigma = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - nm^2\right) \tag{6}$$

where m is the mean value, n is the number of measured values and x_i are the measured values.

(The best estimate for smaller numbers of pixels is under review.)

17.21 RESULTS

Once a fit has been `MINIMISE`'d the results may be displayed. This converts certain parameters to more convenient units e.g. degrees for angles. For each parameter a short character string shows the type of the parameter.

(At present the error estimates for the fitted parameters are not being calculated. If this if required it can be re-installed relatively quickly.)

Note: By using the `OPEN LOG` command in the main menu prior to entering the `FIT` sub-menu the output of `RESULTS` may be saved in an ASCII file (See Section 15.63, Page 145).

17.22 SET MASK COLOUR

This commands allows the user to change the colour used to represent the "masked-off" elements in the `DEFINE MASK` and `DISPLAY MASK` options. The user is prompted for the colour to be used. e.g.

```
COLOUR FOR MASKED OFF ELEMENTS [RED]:?
Enter one of the following available colours:
BLACK BLUE BROWN CYAN GREEN GREY MAGENTA ORANGE RED VIOLET WHITE YELLOW
(0 INDEX) (1 INDEX)
COLOUR FOR MASKED OFF ELEMENTS [RED]:green
```

When the data and mask are next displayed the chosen colour will be used to display the "masked-off" elements. (In fact the colour will be as close as is possible, given the colours presently stored in the colour table.)

17.23 SET UP

Various aspects of the minimisation may be controlled from this sub-menu. Most importantly are the number of iterations `MINIMISE` will perform before stopping, and the aimed accuracy of the fit. (Many of the options are not presently relevant since they refer to multiple 1-D fitting which is not presently implemented.)

The available options are:

`ACCURACY`: Set aimed accuracy of the minimisation process. This is the aimed for reduced chi-squared for a weighted fit, or the average square residual for an unweighted fit.

`CHANGE FIT ORDER`: Not relevant

`DISPLAY FREQUENCY`: Not relevant

`EXIT`: Exit from `SET UP` sub-menu to return to `FIT` sub-menu

`FAST DISPLAY`: Not relevant

`HALT CRITERION`: Not relevant

`MAXIMUM LIKELIHOOD`: Not relevant

`MODEL EVOLUTION`: Not relevant

`NUMBER ITERATIONS`: Set number of iterations per variable parameter

`POISSONIAN STATISTICS`: Not relevant

`REQUEST CONTINUATION`: Not relevant

`VIEW`: Display current state of all set-up variables

`WEIGHTED FIT`: Toggle between weighted, or unweighted fit (only relevant if variance arrays are present, otherwise all fits are unweighted)

17.24 SURFACE POLYNOMIAL

The *ADR*, except for masked-off data is fitted by a 2-D polynomial of user input order in X and Y. The chosen surface is the least squares solution to the data. Generally the order of the polynomial should be much lower than the number of data points in each direction to avoid that the surface fits noise in the data.

The RMS residual of the fit is output. This figure may be compared to the error estimates of the data to see if the fit is to loose or fits the errors too much. If variance estimates are present, the fit will be weighted, and the RMS residual output will be a reduced chi-squared figure.

The fitted surface is created in the memory, so may be recovered after leaving the `FIT` sub-menu.

17.25 THRESHOLD MASKING

The `THRESHOLD MASKING` command allows masked and un-masked elements within the *ADR* to be defined automatically based on a user set logical criterion. The user selects one of two logical tests: i. greater than, ii. less than, and a threshold value. All data elements in the *ADR* which obey the test are set to be masked, and all those which fail are set to un-masked.

e.g. If we want all elements with a data value below 0.5 to be set to be masked, then the following program dialogue would be produced:

```
Fit sub-menu: ENTER COMMAND [INPUT PARAMETERS]:threshold
INFO: This option allows masked-off elements with the ADR to be defined
      automatically depending on the data values. The user can set a
      threshold operation (<GREATER THAN> or <LESS THAN>) and a
      threshold level. All data values which CONFORM WITH the defined
      test will be set to be masked-off. All others will be set to be
      un-masked.
LESS THAN COMPARISON [YES]:
DECISION THRESHOLD DATA VALUE [1.0000]:0.5
```

17.26 TILT/BEAM CENTRE

TILT/BEAM CENTRE allows any non-orthogonality of the detector to the direct beam (tilt), and the direct beam centre on the detector to be determined [10]. This is achieved by least squares fitting of powder rings (or rings from other randomly orientated samples e.g. wax).

Below is the log file from **FIT2D** of the present user interaction. The documentation will be fully written when the algorithm and user inputs are stable:

```
Fit sub-menu: ENTER COMMAND [INPUT PARAMETERS]:tilt

WARNING: This is a development routine, user prompts are likely to change

SAMPLE TO DETECTOR DISTANCE (MILLIMETRES)
 (Range: .10000 to 1.000E+05) [200.000]:
NUMBER OF ANGULAR SECTIONS (Range: 10 to 360) [90]:?
The powder ring is divided into a number of equal angle sections for
calculating the radial centre of each section. From the radial centre
and the average angle two Cartesian coordinates are calculated. These
coordinates are used to fit optimum beam centre and detector plane tilt
angles. Enter required number of sections for calculations.
NUMBER OF ANGULAR SECTIONS (Range: 10 to 360) [90]:
WEIGHTED FITTING [NO]:?
The fitting of tilt angle and beam centre to the estimated ring centres
may be performed using weighted fitting or unweighted fitting. If
weighted fitting is chosen then the average intensity (3 pixels) around
the calculated centre of each radial profile is used to weight the fit.
This means that strong rings and strong angular regions of rings will
have more influence than weaker ones. This should make the fitting more
robust when the data has weak rings and noisy background.
WEIGHTED FITTING [NO]:
REJECT OUTLYING COORDINATES [YES]:?
Outlying coordinate positions which are more than an input number of
standard deviations radially from the fitted ring positions may be
rejected from the coordinate lists. The beam centre/tilt can then be
re-fitted without these coordinates. If erroneous coordinate positions
are influencing the fit, this option may allow them to be removed.
```

```
have more influence than weaker ones. This should make the fitting more
robust when the data has weak rings and noisy background.
REJECT OUTLYING COORDINATES [YES]:
REJECT LIMIT (NUMBER OF STANDARD DEVIATIONS)
 (Range: 1.00000 to 10.0000) [3.00000]:?
Enter the limit of number of standard deviations after which coordinate
positions are to be rejected from the coordinate lists. A three sigma
limit should be reasonable for less than about 200 coordinates (assuming
they are normally distributed.)
REJECT LIMIT (NUMBER OF STANDARD DEVIATIONS)
 (Range: 1.00000 to 10.0000) [3.00000]:
FULL ALGORITHM INFORMATION [YES]:?
YES: if you want step by step diagnostics information
FULL ALGORITHM INFORMATION [YES]:
INPUT METHOD FOR BEAM CENTRE [KEYBOARD]:?
AVERAGED GRAPHICAL: Average of average symmetry centres
CIRCLE COORDINATES: Least squares fit on >=3 coordinates
ELLIPSE COORDINATES: Least squares fit on >=5 coordinates
GRAPHICAL COORDINATE: Single graphical input coordinate
KEYBOARD: Single keyboard entered X/Y coordinate
INPUT METHOD FOR BEAM CENTRE [KEYBOARD]:
X-PIXEL COORDINATE OF BEAM CENTRE [256.5]:256
Y-PIXEL COORDINATE OF BEAM CENTRE [256.5]:255

INFO: The search distance either side of the powder rings (mm) =   1.511
INFO: The search distance either side of the powder rings (X-pixels) = 15.11
GRAPHICAL INPUT: 4.3077289E+02 2.6403860E+02

INFO: Radius of powder ring  1 (mm) =      6.894
INFO: Radius of powder ring  2 (mm) =     17.501
INFO: Radius of powder ring  1 (X-pixels) =     68.940
INFO: Radius of powder ring  2 (X-pixels) =    175.006
REFINE BEAM CENTRE [YES]:?
The entered beam centre position may be kept fixed or may be refined
along with the tilt and powder ring opening angles. If the beam centre
is well known e.g. by using a semi-transparent beam-stop then it is
probably better not to refine the position.
   Enter "YES" for the beam centre position to be refined, and "NO" if
the present values are to be kept fixed.
REFINE BEAM CENTRE [YES]:
REFINE SAMPLE DISTANCE [NO]:?
Enter "YES" to refine the sample to detector distance. Normally it is
probably best initially to keep this fixed, so answer "NO". Having
refined all other parameters, on a second iteration the distance can
also be simultaneously refined.
REFINE SAMPLE DISTANCE [NO]:

INFO: Calculating centre of gravity coordinates on 1 powder ring
INFO: Number of acceptable coordinates on first ring =  90
```

```
INFO: Fitting ellipse to centre of gravity coordinates
INFO: Number of coordinates =  90
INFO: Best fit ellipse centre (X/Y mm) =    25.61649       25.61244
INFO: Best fit ellipse centre (X/Y pixels) =    256.1649       256.1244
INFO: Best fit radius 1, radius 2 (mm) =    7.004938       6.983946
INFO: Best fit radius 1 (X pixels) =    70.04938
INFO: Best fit radius 2 (Y pixels) =    69.83945
INFO: Best fit angle of axis 1 (degrees) =    36.94184
INFO: Estimated coordinate radial position error (mm) =    .1446273E-01
INFO: Estimated coordinate radial position error (X pixels) =    .1446273

INFO: Calculating centre of gravity coordinates on powder rings

INFO: Fitting powder rings to centre of gravity coordinates
INFO: Number of iterations =            6
INFO: Number of function calls =           64
INFO: Sum of squares =     .3310081043632615E-07
INFO: Number of rejected coordinates =     0
INFO: Number of iterations =            4
INFO: Number of function calls =          125
INFO: Sum of squares =     .3310080011801160E-07
INFO: Fit of powder ring to inclined detector
INFO: Best fit beam centre (X/Y mm) =    25.59796       25.60419
INFO: Best fit beam centre (X/Y pixels) =    255.9796       256.0420
INFO: Cone  1 best fit 2 theta angle (degrees) =    1.999999
INFO: Cone  2 best fit 2 theta angle (degrees) =    4.999608
INFO: Best fit angle of tilt plane rotation (degrees) =    28.59482
INFO: Best fit angle of tilt (degrees) =    4.826739
INFO: Estimated coordinate radial position error (mm) =    .1379255E-01
INFO: Estimated coordinate radial position error (X pixels) =    .1379255

INFO: Alternative fit of powder rings to centre of gravity coordinates
INFO: Number of iterations =            5
INFO: Number of function calls =           46
INFO: Sum of squares =     .3310209888343639E-07
INFO: Number of rejected coordinates =     0
INFO: Number of iterations =            1
INFO: Number of function calls =           18
INFO: Sum of squares =     .3310145134466964E-07
INFO: Fit of powder ring to inclined detector
INFO: Best fit beam centre (X/Y mm) =    25.59794       25.60424
INFO: Best fit beam centre (X/Y pixels) =    255.9794       256.0424
INFO: Cone  1 best fit 2 theta angle (degrees) =    2.000001
INFO: Cone  2 best fit 2 theta angle (degrees) =    4.999619
INFO: Best fit angle of tilt plane rotation (degrees) =    28.54871
INFO: Best fit angle of tilt (degrees) =    4.825097
INFO: Estimated coordinate radial position error (mm) =    .1379269E-01
INFO: Estimated coordinate radial position error (X pixels) =    .1379269
```

```
INFO: In the absence of any other information, there are two equally
      valid solutions to the tilt angle and beam centre which could
      have formed a powder ring on an inclined detector. Theoretically
      these should both give the same goodness of fit, so both solutions
      are output. If the position of the beam-stop is known, then the
      "correct" solution may be selected.

INFO: SOLUTION 1
INFO: Best fit beam centre (X/Y mm) =    25.59796       25.60419
INFO: Best fit beam centre (X/Y pixels) =    255.9796       256.0420
INFO: Cone  1 best fit 2 theta angle (degrees) =    1.999999
INFO: Cone  2 best fit 2 theta angle (degrees) =    4.999608
INFO: Best fit angle of tilt plane rotation (degrees) =    28.59482
INFO: Best fit angle of tilt (degrees) =    4.826739
INFO: Estimated coordinate radial position error (mm) =    .1379255E-01
INFO: Estimated coordinate radial position error (X pixels) =    .1379255

INFO: SOLUTION 2
INFO: Best fit beam centre (X/Y mm) =    25.59794       25.60424
INFO: Best fit beam centre (X/Y pixels) =    255.9794       256.0424
INFO: Cone  1 best fit 2 theta angle (degrees) =    2.000001
INFO: Cone  2 best fit 2 theta angle (degrees) =    4.999619
INFO: Best fit angle of tilt plane rotation (degrees) =    28.54871
INFO: Best fit angle of tilt (degrees) =    4.825097
INFO: Estimated coordinate radial position error (mm) =    .1379269E-01
INFO: Estimated coordinate radial position error (X pixels) =    .1379269
WHICH SOLUTION (Range: 1 to 2) [1]:
```

17.27 TRANSFER MASK TO MEMORY

This “transfers” the data mask to the memory. All elements which are masked-off in the data mask will be set to 1.0 in the memory; un-masked elements will be set to 0.0.

This option can be used together with the `THRESHOLD MASKING` command to save and re-use masks. Operations such as adding two masks may also be used.

18 Publication Quality Graphs

The keyboard interface command `PUBLICATION QUALITY` changes all line widths (except those of the grid) to double thickness, and sets all text fonts to the best available font; previous line widths or font settings are replaced[58].

Thicker lines are usually desirable when graphics are photo-reduced for publication.

[58]At present only one font family is used by the graphics system so the fonts do not change, but the font family used, “Times-Roman” should be suitable for most purposes.

19 Creation and Usage of Simple Macros

Often the same sequence of data analysis operations will need to be repeated for many different data-sets. The instructions here show how a simple "macro" may be created for this purpose and automatically run on the sequence of files. More versatile and sophisticated macros may also be developed, their creation is described in Appendix B (Section 29, Page 251).

The simplest way in which you create a macro file is to perform the normal data analysis of one file, but to enter the **MACROS / LOG FILE** interface before and click on the **CREATE MACRO** button, and afterwards to return and click on the **STOP MACRO** button. When you click on the **CREATE MACRO** button, you will be asked for the name of the macro file to create.

FIT2D will save this macro to an ASCII file. This contains the instructions for the data analysis sequence, but will only run on the file which has just been treated. Using your favourite editor the file name(s) can be found and replaced by "variable names" which will be automatically generated by **FIT2D**.

e.g. A standard basic macro has one input file and one output file, so the input file name would be replaced by the "variable" `#IN` and the output file name by the "variable" `#OUT`.

Having edited the macro it is ready to use with **RUN SEQUENCE** button. See Section 8, Page 74 for further information. This prompts for the names of the first and last input files (corresponding to `#IN`). From these files the changing numerical part is deduced. The user is prompted for the increment between files. By default this is 1 (or -1), i.e. every file in the series is treated, but by entering larger integers every secong, third, etc. file can be used and the others ignored. The output files (corresponding to `#OUT`) are given the same name as the input files except for the file extension. The user is prompted for the file extension which replaces that of the input files. This will then automatically generate the sequence of file names and run the macro on each set of input and output files.

It is also possible to define and run macros from the "KEYBOARD" interface using the `START MACRO`, `STOP MACRO`, and `SEQUENCE` commands. This is much more flexible (and complicated): more than one input and output files are possible, and other variables can be generated to make macros more versatile. See Section 29, Page 251 for further details.

20 Command Line Options

To make **FIT2D** more versatile and for "batch" mode applications a number of command line options are available. This allows a Unix script to be written to provide batch mode processing of a sequence of files (see Section 29.5, Page 259). (Note: Now the `SEQUENCE` command can also be used to process a sequence of files with interactive control of the files to be treated, but for special tasks a Unix script may still be useful.) Program "variables" may be defined (See Section 29.1, Page 251), a macro file may be specified to run **FIT2D** in macro mode (See Section 19, Page 225), the program array dimensions may be specified, and **FIT2D** may be started with "landscape" graphics instead of the default "portrait" graphics. The graphics system may also be "turned-off". The manner in which the command line options work is best illustrated by an example. The following is an example command line which may be used:

```
> fit2d -dim1152x1482 -land -key -svar#IN=im.gel -svar#OUT=im.ps -macplot.mac
```

The command line would start **FIT2D** in the following manner:

`-dim1152x1482` Starts **FIT2D** with programs arrays of 1152 pixels in the X-direction and 1482 pixels in the Y-direction

`-land` Starts the graphics in "landscape" mode i.e. the graphics window appears wider than its height and PostScript output is printed in this orientation.

`-key` Starts **FIT2D** in "keyboard" mode.

`-svar#IN=im.gel` Defines a program variable of data type "character string" whose name is `#IN` and whose value is `im.gel`

`-svar#OUT=im.ps` Defines another "character string" program variable whose name is `#OUT` and whose value is `im.ps`

`-macplot.mac` Starts **FIT2D** in macro input mode, taking input from the macro file `plot.mac`.

Here the order of the options has been given in a "logically sensible order", but the order is not important except if options are repeated. This example only makes sense if the macro file `plot.mac` exists which would normally input a file defined by the variable `#IN`, plot the data or some region of the data using "landscape" mode, and output the graphics into a PostScript file defined by `#OUT`. (Complicated command line calls may be made simpler and shorter through the use of "aliases".)

The different options are described in greater detail below:

20.1 Macro File

FIT2D may be started automatically from the operating system command line to run a macro file. (The macro file may or may not close **FIT2D**.) The option to start-up in macro mode is

`-mac` and must be followed by the name (or a path and name) of a valid macro file. (`-MAC`, `+mac`, and `+MAC` may also be used.) For creation of macro files see Section 19, Page 225.

When **FIT2D** is started in macro mode it will not prompt for the size of the program arrays or if variance arrays are to be created. By default the program arrays will be set to 1000×1000 pixels and no variance arrays will be created. If this is not suitable for the macro, the program arrays may be created a different size by using the `-dim` option and variance arrays may be created by using the `-var` option (see below). Alternatively the `DIMENSIONS` command may be used within a macro to over-ride the defaults values.

To make complicated macros versatile "variables" may be used within the macros and their "values" may be changed to change the behaviour of the macro i.e. to parameterise the macro. This is most useful to allow the macro to work on different input and output files, but may also be used for other purposes. Thus, to make the macros versatile the `-ivar, -lvar, -rvar, -svar` options exist (see below).

Only one macro file may be opened so it only makes sense to specify one macro file on the command line. If more than one is specified, then only the first one will be used.

20.2 Variable Definition

The options `-ivar, lvar, -rvar, -svar` exist to enable program variables to be defined from the operating system command line. (`-bvar, -cvar, -fvar` are alternative names of the options.)

The different options define variables of different data types. i.e.:

`-bvar` Logical (boolean) variable

`-cvar` Character string variable

`-fvar` Floating point real variable

`-ivar` Integer variable

`-lvar` Logical (boolean) variable

`-rvar` Floating point real variable

`-svar` Character string variable

(`+bvar, -BVAR, +BVAR` are equivalent to `-bvar`, as are the corresponding variations for the other options.) Immediately following the option is the name and the value of the variable separated by an equals sign. The first blank ends the variable definition (It is recommended to always use names that cannot be mistaken for other entry e.g. start every variable name with a hash (`#`).) At present lines of text cannot be defined as values for variables from the command-line.

Variable definition makes most sense when used within a macro (See Section 8.1, Page 75).

Multiple variable definitions may be used to define a number of different variables. If the same variable name is used more than once the later versions will over-write the previous versions.

(`-sym, -SYM, +sym, +SYMBOL,` and `+SYM` was the manner in which to define program "symbols" before V10.0. These may still be used and defined a variable of data type "unknown", however, it is recommended to replace these with the new options which allow the data type to be defined. Internally, these are stored as strings, and can usually be converted to the required integer, logical, real, or character string value, but this may cause problems.)

20.3 Program Dimensions

In macro mode **FIT2D** will not prompt the user for the size of program arrays. By default they will be 1000×1000 pixels. If this is not suitable then their size may be specified on the command line by using the `-dim` or `-dimensions` option. (`-DIM, -DIMENSIONS, +dim, +dimensions, +DIM,` and `+DIMENSIONS` are equivalent.) The `-dim` option should be immediately followed by the size of the program arrays to create in the X-direction, and the size of the arrays to create in the Y-direction separated by an `x` or an `X`[59]. The `-dim` option may also be used to avoid being prompted at start-up for the program dimensions.

e.g. The following operating system command line would start **FIT2D** with program arrays of 500 elements in the X-direction and 700 elements in the Y-direction:

```
> fit2d -dim500x700
```

(The `DIMENSIONS` command may be used within a macro as an alternative method of changing the size of program arrays from the default values.)

20.4 Variance Arrays

In macro mode **FIT2D** will not prompt the user as to where or not variance arrays should be created. By default they will **not** be created. If this is not suitable then their creation may be specified on the command line by using the `-var` or `-variances` option. (`-VAR, -VARIANCES, +var, +variances, +VAR,` and `+VARIANCES` are equivalent.)

(The `DIMENSIONS` command may be used within a macro as an alternative method of creating variance arrays.)

20.5 No Graphics

Sometimes it is required to run **FIT2D** without a graphics output window. This may be required because the output window cannot work properly across the network. It may also be for batch processing, which doesn't require any display. The `-nographics` command line option allows **FIT2D** to start-up and stop without the graphics display window being opened. Clearly,

[59]The multiplication sign cannot be used as this has a special purpose for the Unix operating system.

this makes many commands invalid[60]. (The `-nographics` option can be shortened to `-nogr`, and alternative forms are equally permissible: `-NOGRAPHICS, +nographics, +NOGRAPHICS`).

20.6 "Landscape" Output

By default **FIT2D** always produces vertical A4 ("portrait") output on the screen and on paper. The page format can be changed to horizontal ("landscape") output by using the option `-landscape` or `-land` on the command line when running the program (`-LANDSCAPE, -LAND, +landscape, +land, +LANDSCAPE,` and `+LAND` are equivalent.) i.e. To start up **FIT2D** in "landscape" mode enter:

```
fit2d -landscape
```

Since the page layout has been designed for "portrait" mode, some re-arrangement of the page layout may be necessary.

20.7 Colour Table Control

The number of colour indices used by **FIT2D** may be control by the `-col` option followed by the number of colour levels to be used. (`-colours, -COLOURS, -COL, +COL, +COLOURS,` etc are alternative accepted options.) **FIT2D** uses eight colours for the primary and secondary colour, and for black and white, and a variable number of colours for continuous false colour scale display of image data. The minimum number of colours is 16, whilst the maximum is presently 210. Values outside this range are ignored. Clearly colour tables with only eight colours (the minimum allowed) are not smooth.

This option can be useful to allow other programs which are not written to create their own colour tables to run after **FIT2D** has been started.

e.g. to start-up **FIT2D** using only 72 colour levels (64 used for the false colour image display) enter:

```
> fit2d -col72
```

20.8 Graphical Form and Menu PostScript Output

For documentation purposes e.g. producing this manual, the `-dform` and `-dmenu` options enables optional saving of form and menu graphics to PostScript files. This will not normally be used.

When `-dform` or `-dmenu` has been specified it is possible to click within a form or a menu, but outside button and other active areas. **FIT2D** then prompts for the name of a file to save the

[60]At present use of some graphics command, whilst **FIT2D** is in this mode will result in a core dump. Gradually, these commands will be protected, and warning messages will be issued instead.

PostScript graphics within. This file will be written with the instructions to draw the form or the menu.

Since forms also continue menus, it is necessary to use the `-dform` option without the `-dmenu` option to be able to save the graphics of the whole form.

21 Coordinate Systems

For full use of **FIT2D**'s display possibilities it is necessary to understand the coordinate systems used for different user controllable items. Interactive control of the position and size of false colour images and contour plots is available as are the position of extra graphical items.

The positions of all graphical items are specified in X/Y coordinates, where the X-axis is horizontal and a larger X-value is further to the right, and the Y-axis is vertical and a larger Y-value is higher on the page.

There are two different types of coordinate system:

- Data Coordinates
- Page Coordinates

'Data coordinates' are the coordinate system of the data, and the range of displayed data coordinates is either automatically determined or is set by the user (`REGION`, `PIXEL REGION`). This region is the *ADR*, but is also referred to as the 'Data Display Region' (DDR) by the graphics system.

'Page Coordinates' are the coordinates of the 'page' and the position of all the graphical items may be altered by altering their page coordinate. The position of the graph on the page is referred to as the 'Graph Page Position' (GPP). This is the position of DDR on the page. There is a mapping, or transformation, between data coordinates and page coordinates.

The page coordinate system goes from 0.0 to 1.0 in the longest edge, and from 0.0 to the page aspect ratio in the the other direction. e.g. For A4 portrait, X may be between 0.0 and 0.7071, and Y may be between 0.0 and 1.0.

Almost all items whose layout position may be changed are in page coordinates. The exception is annotation labels and annotation arrows. These may be defined in either page coordinates or in data coordinates. This allows the possibility to link the position of annotation labels the arrows to parts of the graph.

22 Including Graphics in LaTeX 2_εDocuments

The PostScript files produced by **FIT2D** are suitable for including in LaTeX 2_εdocuments using the `graphicx` package. To use the `graphicx` package include the following line near the start of your LaTeX file (before the \begin{ document} line):

```
\usepackage{graphicx}
```

The \includegraphics command is used within the LaTeX document to specify the inclusion of a PostScript file. e.g. If a file called `fit2d.ps` has been created and we want include it in a document taking up 150mm vertically we can used the command:

```
\centerline{\includegraphics[height=150mm]{fit2d.ps}}
```

The **FIT2D** graphics are designed to occupy a page of A4 so the `height` option is necessary to tell LaTeX 2_ε to reduce the size of the graphics.

In the GUI mode **FIT2D** always outputs one graphics page per PostScript file. In the "keyboard" mode **FIT2D** will output several graphics pages into a single PostScript file, unless the `END GRAPHICS FILE` command is used in between `PRINT GRAPHICS` commands.

To use only part of a diagram it is necessary to specify the region as part of the \includegraphics command.

See the "LaTeX User's Guide and Reference Manual" [19], or "Using EPS Graphics in LaTeX 2_ε Documents" [28], for further details.

23 The Graphics System

A simple efficient graphics system (LG: the “Light Graphics” system) has been written to allow graphical output to X-11 displays and PostScript files. This replaces the previous commercial system.

LG is similar in concept to, but not the same as GKS (“The Graphics Kernel System”), an international standard for 2-D computer graphics. LG allows a device independent interface, but has extra functionality for more efficient use of the underlying hardware.

The system tries to be close to WYSIWYG (“What You See Is What You Get”), but the limitations of X-11 or the PostScript printer will lead to some differences.

The new system allows many advantages and allows similar appearance of the graphical output, but is quite different in some important aspects.

23.1 Fonts for Text Output

Hardware (bitmap) fonts are now used, which means faster solid text, but the choice of sizes are limited and slight differences may occur between the screen font and the hard-copy font[61].

If available the “Times-Roman” font family is used. Depending on the size of the window and the required size of characters, the nearest font size will be used. However, very big and very small fonts are not available in X-11 so the range of sizes is limited. If the required font is not found, a standard X-11 font will be used instead (6x10 or 9X15), but this font will be far from the ideal size. It has been seen that some X-servers have fonts with different aspect ratios to those for which **FIT2D** was designed. This may lead to text extending outside the normal region. For PostScript output the exact font size (as originally requested) is produced, so slight differences in the size of characters on the screen and on paper will occur.

X-11 (generally) does not provide for rotated characters (although PostScript does), so rotated text is simulated. In the future other fonts e.g. Greek characters, may be provided.

23.2 Colour Map Usage

FIT2D is presently optimised for use on colour displays (“X-displays”) with a maximum of 256 available displayable colours at any one time (“8-bit plane pseudo-color”). These are still the vast majority of available displays, although “24-bit plane” “true colour” displays are starting to become more common.

An individual window e.g. the **FIT2D** graphics window, is allowed to define up to 256 individual colours, which may be different to those which are defined by another window. When the user clicks in a new window[62], then the colours defined by the new window replace the colours

[61]In fact X11R6 allows arbitrary sized fonts to be generated, but this is only available on a smaller percentage of current X-servers, so this possibility has deliberately not been exploited.

[62]Or in some cases by simply moving the cursor into a new window, depending on how the “window manager”

previously used. This is termed “colour flash”, and previously readable text may become unreadable.

“Colour flash” may be avoided if all windows share the same colours, but this not suitable for efficient false colour image display e.g. grey scale, of large images. **FIT2D** defines its own colours. If there are enough remaining undefined colours in the default colour table, then this will be used and “colour flash” will not occur. Unfortunately, window managers and other programs are using more and more colours simply for button display and shading, so often very few colours levels are undefined. In this case, a new individual colour table is created and used by **FIT2D**. To minimise the problem cased by “colour flash” the first 40 colours in the existing colour table are copied and are left unchanged. On some X-displays this keeps text readable, but unfortunately other X-displays use other colour levels for displaying text so still suffer from “colour flash”.

FIT2D needs a large number of colour levels to be able display false colour (or grey-scale) images efficiently with a smooth colour scale. To minimise the problem of colour “flash” **FIT2D** will use the default colour map if a minimum number of contiguous colour levels are free (presently 108). It will then use the remaining colours to produce the smooth false colour scale and 8 basic colours. Otherwise a new colour map, unique to **FIT2D** will be created. The first 40 levels will be copied from the window managers colour map, and the remaining levels will be used for the smooth false colour scale and 8 basic colours. If this happens colour “flash” will probably be noticed, but depending on which levels are used for which windows the effect may be minimised[63]. It is worth noting that some programs, have not been written to create their own colour tables, so will not work after **FIT2D** has been started. However, if such programs are started first, and **FIT2D** afterwards, the two can be used simultaneously.

Within the colours defined by **FIT2D** eight colours are always fixed, and are used for menu display and user prompts (white, black, red, green, blue, yellow, cyan, and magenta). The remaining colour levels are filled with the required false colour table e.g. geographical or inverse grey-scale. Different colour tables contain different colours so graphical objects which are not coloured in one of the eight constant colours will change colour when the colour table is changed. (This is why the “FIT2D” logo background sometimes changes colour just after **FIT2D** is started. The background is orange, which is defined in the default colour table, but not for example in a grey-scale.)

The number of colour indices used may be varied by a command-line argument `-col` (see Section 20, Page 226).

Note: some programs e.g. the DENZO display program **Xdisplayf**, do not work properly if they cannot obtain enough colour levels from the window managers colour map. With such a program it may be better to start the program first and **FIT2D** afterwards, if both are required simultaneously.

has been defined.

[63]On HP booted X-servers xterm uses colour levels within the first 40, so the terminal window remains readable, but on Sun booted X-servers this is not the case.

23.3 Graphical Input

Graphical coordinate input activates a "cross-hair" cursor, so that horizontal and vertical alignment is easier. During input of a series of connected coordinates a "rubber-band" may be drawn from a previous coordinate, and a line may be used to join the input coordinates. Similarly, if a rectangular region is to be defined by two coordinates a "rubber-band" box will be drawn from the first input coordinate whilst inputing the second. Certain coordinate inputs, e.g. the "`ZOOM IN`" button command, produce a special "spy-glass" or "looking glass" window showing in greater detail the region around the graphics cursor (V7.32). As the cursor moves, so this display region will change.

Text input is now available within the graphics system (V7.22). This supports "command recall" and command editing, using the arrow keys. Text may edited, by using the left and right arrow key to move to the required character. Newly typed text is automatically inserted at the cursor position, and the `<DELETE>` key (or the `<BACKSPACE>`) may be used to delete characters before the cursor. Often a default value will be presented. This may be edited, or the down arrow key may be used to remove it.

At present all graphical input and menus appear within the same window as the image display. Eventually a separate graphical menu window is desirable.

23.4 X-11 Restrictions

At present the **LG** system has only been written for X-displays where the default colour mapping method, or "visual" (in X-11 terms), is an 8-bit plane "PseudoColor visual"[64] or 24-bit plane "TrueColor" (V9.136). Unfortunately 24-bit "TrueColor" graphics is handled in a very complicated manner in X-11 and there are many possible combinations of bit and byte ordering. The output has been tested on eXceed and LAN Workplace Pro X-servers in 24-bit mode, and a number of Unix workstations, but it may not work properly on some other systems. Other types of colour mapping can now be added with reasonable ease, since they are related to one of the above choices e.g. a 16-bit plane "TrueColor" system could be added, but still this involves a variety of changes to the source code, and needs testing on such a system. For this to be viable I need access to such systems at the ESRF.

If the default is not suitable a detailed error message will be output explaining the characteristics of the default window.

[64] "Visual" and "PseudoColor", with the American spelling, are X-11 terminology.

24 Limitations, Bugs, and Unclear Documentation

24.1 Known Bugs and Limitations

At present there are no known bugs, but there are most likely to be severe limitations of the present graphics system. It is unlikely to work on monochrome systems.

The graphics system tries to use the "Times-Roman" font family, which is also available for PostScript output. If it does not find the required font an alternative X-11 font will be tried. This will generally work, but will not be the intended size, so text may be much smaller or larger.

FIT2D now works on Windows system, but this is new and there are many differences, so bugs may well be present on these systems.

24.2 Limitations

24.2.1 "Legal" Paper Hard-copy Output

The graphics system supports the "A" series of paper formats, where the two dimensions are related by the square root of two. Other paper formats such as the "Legal" size often used in North America are not supported explicitly.

Since, by default the graphics are designed for A4 (portrait or landscape orientations), they will be too wide for the legal paper format. By using the keyboard "PAGE POSITION" command or the GUI OPTIONS menu "POSITION" command you can change the area of the size and hence the paper which is used to print the graphics. Thus, the whole of the graphics can be made to appear on the paper.

24.3 Reporting Bugs, Bad Features, and Bad Documentation

Users are invited and encouraged to provide feed-back on **FIT2D** including comments on the on-line and paper documentation. User feed-back has already led to new features being added, and modification to old ones to make them more understandable and easier to use.

If you think that you have found a bug then please follow the following reporting procedure:

1. Check the **FIT2D** Reference Manual section describing the command to make sure that the observed behaviour is not as described.
2. Check the **FIT2D** Reference Manual "Known Bugs" section (Section 24.1, Page 236) to see if the bug has already been identified, and whether a work around has been suggested.
3. If the bug is "new", note the version number of **FIT2D** being used.

4. Use the `OPEN LOG` and `CLOSE LOG` commands to create a log-file record of the operations (minimum) which produce the bug.

5. Send by E-mail, or other means, a description of the bug (if this is not obvious from the log-file), the version number, any terminal or file traceback information. If the bug appears to be data specific, a means of obtaining (e.g. aftp) the data file which causes the bug.

As an encouragement to follow this procedure, a free beer is offered to anyone who reports a *serious* unknown bug[65].

Users are also encouraged to provide feed-back on the documentation. Please report mistakes and unclear documentation. Also report prompts which give insufficient information for the user to know how to respond. (Note: Giving documentation to users and getting them to try to use it is the only way to test it properly !)

The index is a very important part of the **FIT2D** Reference Manual. Users are not expected to read more than small sections of the manual. The index should be able to guide users to relevant sections of the manual and show how the commands may be used to perform particular operations. It you find that a useful index item is missing, for an operation which can be performed please inform me.

[65]The definition of whether or not a bug is serious or not rests with the author. Clearly, mis-use of the program, and annoying features do not count, and mistakes in documentation are unlikely to count. The offer does not cover bugs that occur within source code not written by the author e.g. "readline" code (but this seems unlikely). New (experimental) features are similarly not covered. The beer will be a standard "demi pression" and not a Palais de la bière tourist sized top price beer. Anyone who "wins" a free beer is responsible for paying their own transport to and from the centre of Grenoble. Users who find features which work unusually well are invited to offer a free beer in return . . .

25 Trouble-Shooting

25.1 Command not found

If when you enter **FIT2D** the system responds:

```
fit2d: Command not found.
```

this means either that **FIT2D** has not been installed (properly), or that your `PATH` variable is not properly defined. Normally, **FIT2D** and other EXPG programs should be found in the `/usr/local/bin` directory.

Thus, your environment variable `PATH` should include this directory.

e.g. To add the `/usr/local/bin` directory to the search "path" for executable programs you can add the following instruction to your `.cshrc` file[66]:

```
setenv PATH /usr/local/bin:$PATH
```

See your system administrator, if you cannot rectify the problem.

25.2 Failure in FIT2D Start-up

If **FIT2D** fails to start-up properly, this can be due to one of a number of reasons, which are outside the control of **FIT2D**.

If the following message appears, or one similar, it means that the **present** available system virtual memory size is too small.

```
Not enough memory
```

This problem may appear and go away simply owing to other users starting and finishing processes. If this is a persistent problem see your system administrator.

If the welcome text appears, but afterwards a message similar to the following appears, it is necessary to authorise another workstation (client) to communicate with your workstation (you have a remote login).

```
Xlib: connection to ``expgf:0.0'' refused by server
Xlib: Client is not authorized to connect to Server
Failed to open X-11 display
```

[66]Great care must be taken when re-defining `PATH` as if a mistake is made it is possible to loss all system commands.

```
ERROR: X-11 workstation not opened successfully (EXPG_LG_OPEN_WK)

STATUS: The error was identified in module:
EXPG_LG: ``Light Graphics''

STATUS: Position where error condition was identified
Subroutine EXPG_LG_X11_OPEN V0.3

STATUS: The error condition has been classified as:
Failed to open X-11 window (use ``xhost+'' on X-server ?)

WARNING: The graphics window could not be opened, therefore the graphics
         has been turned-off. You will need to change something to be
         able to output the graphics e.g. xhost+ on an X-server. Or
         maybe the ``DISPLAY'' environment variable is not set correctly.
         You can type:

         printenv DISPLAY

         at the command line (C-shell and T-shell) to check.
         Following this you need to re-start FIT2D.
```

To add the remote workstation to the list of allowed communicating workstations use `xhost` e.g. if the remote workstation is called `expga`, it is necessary to type the following command on **your** workstation:

```
xhost +expga
```

25.3 No Image Output

If the image display appears to be wrong, or nothing appears to be displayed it is likely that you have previously used the `Z-SCALE` command to set the scaling mode to semi-automatic, or fixed scale. If the present image is outside the range then the image will appear totally uniform (black or white for most of the colour tables).

To overcome this problem, simply use the `Z-SCALE` command to set the scaling mode back to fully automatic, or set a reasonable range (See Section 15.129, Page 175).

25.4 LaTeX 2_ε Instructions Do Not Work

If the instructions given in Section 22, Page 232 do not work it is probably because you are using an old installation of LaTeX which does not include the `graphics` or `graphicx` packages.

The instructions are for the 2_ε version of LaTeX and not for the older 2.09 version. You are recommended to update to the newer version which has much better and consistent support for including PostScript graphics within LaTeX documents.

(With older versions of LaTeX and the `psfig` macros you may overcome problems by removing the `showpage` instruction from the PostScript file produced by **FIT2D**.)

25.5 Graphics Output File Does Not Print

A number of users have occasionally encountered problems printing the graphics output file. The symptom is that the file goes to the printer, takes some time being processed, but finally no output is produced.

This is almost certainly due to `<CONTROL-C>` being used to exit **FIT2D**. The Unix[67] file system does not necessarily flush file buffers to disk unless forced to do so. When **FIT2D** is exited normally there is a call to properly close any open files including the graphics output file. This ensures that all output is properly written to the file. With PostScript an image is only drawn on "paper" when the `showpage` command is encountered. This is normally the last line of the file, so if the file is not properly closed this line can easily be lost.

The simple manner to avoid this problem is not to use `<CONTROL-C>`. Using `<CONTROL-C>` may well cause other problems. It is possible that **FIT2D** also exits abnormally owing to circumstances beyond the users control e.g. bugs, or system crashes. Thus, it may be useful to be sure that a graphics output file is properly closed, without exiting the program. This can be done using the `END GRAPHICS FILE` command (see Section 15.31, Page 123). After the `END GRAPHICS FILE` has been issued and the prompt returns, the user can be sure that the graphics output file is properly closed and written to disk. In this manner it is possible to send the file to a printer without first having to exit **FIT2D**.

25.6 Missing Font Message at Start-up

At start-up a message explaining that the a font is missing may occur. i.e.

```
WARNING: Cannot open required font
```

This message can in principle appear later on, but this is at least rare.

The missing font message is relatively normal, and means that a standard X-11 font will be used instead of a times roman font. (The times roman font family is available in PostScript, so is used to give a close match between the screen and hard copy output.)

Ideally the graphics system uses one of the following fonts to provide text in a wide variety of scaled sizes:

```
clR4x6, timR08, timR10, timR12, timR14, timR18, timR24
```

[67] This may well be a problem limited to Unix systems.

If the nearest smaller times roman font is not available then, the above warning message is output, and one of the following list of X-11 fonts will be used if available:

```
5x7, 5x8, 6x9, 6x10, 6x12, 7x13, 7x14, 9x15, 8x16, 10x20, 12x24
```

If the requested font is not found then **FIT2D** will try to use either the `6x10` or the `9x15` font. If neither of these is found, then the font selection has really failed, no text will be output for the required size, and the following message will be output:

```
ERROR: Cannot open  6x10  nor  9x15 fonts,  text  cannot  be
       output to the X-11 screen.  You may be able to 'load'
       the necessary fonts,  or  change  the font path.  See
       your system administrator or local X-11 guru. Ideally
       the TIMES-ROMAN  series of fonts should be available.
```

It may be possible to alter the font path, or to add extra fonts to the server to overcome these problems. This is however a job for the system administrator.

25.7 The Graphics is not Re-drawn when the Window is Uncovered

Normally the graphics window is re-drawn whenever the window is uncovered after being covered by another window. This generally works, but does not work by default with Silicon Graphics X-servers. (This may also be a problem with other X-servers, but at present this has not occurred.)

The problem is that Silicon Graphics X-servers do not provide "backing store" by default. **FIT2D** requests "backing store" when the graphics window is created, and this works fine on all other known X-servers. When "backing store" is enabled it is the window manager which automatically re-draws the window when it is uncovered.

To make this work for Silicon Graphics X-servers:

The file `/usr/lib/X11/xdm/Xservers` (which is used when launching `Xsgi`) should NOT contain the option `-bs`. (see `man Xsgi`).

This will also avoid the same problem for **IDL** and other programs.

Given the cheapness and availability of RAM there is no good reason for not providing "backing store".

(Dominique Bourgeois can be thanked for investigating and finding the solution to this annoying problem.)

26 Acknowledgement and Citation of Use of FIT2D

For general usage of **FIT2D** beyond basic graphical display and very simple mathematical operations (see Appendix E, Page 278, for further precision), users are kindly requested to add acknowledgements to published work and to cite :

A P Hammersley, *ESRF Internal Report*, **ESRF97HA02T**, "FIT2D: An Introduction and Overview", (1997)

and/or

A P Hammersley, *ESRF Internal Report*, **ESRF98HA01T**, FIT2D V9.129 Reference Manual V3.1 (1998)

For detector distortion calibration and correction, users are kindly requested to additionally cite one or more of the following papers:

A P Hammersley, S O Svensson, and A Thompson, "Calibration and correction of spatial distortions in 2D detector systems", *Nucl. Instr. Meth.*, **A346**, 312-321, (1994)

A P Hammersley, S O Svensson, and A Thompson, H Graafsma, Å Kvick, and J P Moy, "Calibration and correction of distortions in 2D detector systems", *Rev. Sci. Instr.*, (SRI-94), **66**, 2729-2733 (1995)

For non-uniformity of intensity response calibration using fluorescence from doped lithium borate glasses users should acknowledge J P Moy, and cite:

J-P Moy, A P Hammersley, S O Svensson, A Thompson, K Brown, L Claustre, A Gonzalez, and S McSweeney, "A Novel Technique for Accurate Intensity Calibration of Area X-ray Detectors at Almost Arbitrary Energy", *J Syn Rad*, January (1996)

For data obtained using an X-ray image intensifier and corrected for spatial distortion and/or non-uniformity of response by **FIT2D**, users are kindly requested to consider additionally citing:

A P Hammersley, K Brown, W Burmeister, L Claustre, A Gonzalez, S McSweeney, E Mitchell, J-P Moy, S O Svensson, A Thompson, "Monochromatic Protein Crystallography Data Collection Using an X-ray Image Intensifier/ CCD Detector", *J. Syn. Rad.*, **4**, 67-77, (1997)

For integration of 2-D data to 1-D "2θ scans" users are kindly requested to additionally cite:

A P Hammersley, S O Svensson, M Hanfland, A N Fitch, and D Häusermann, "Two-Dimensional Detector Software: From Real Detector to Idealised Image or Two-Theta Scan", *High Pressure Research*, **14**, pp235-248, (1996)

For 1-D fitting with the **MFIT (MULTIPLE FITTING)** interface users should cite:

A P Hammersley, and C Riekel, "MFIT: Multiple Spectra Fitting Program", *Syn. Rad. News*, **2**, pp24-26, (1989)

27 Acknowledgements

FIT2D uses the FITPACK library of Paul Dierckx, Dept. of Computing Science, Katholieke Universiteit, Leuven, Belgium, for surface fitting with spline functions.

FIT2D uses the "readline" library from the "Free Software Foundation" and has benefitted greatly from the use of `gmake` and `emacs`.

FIT2D has benefitted much (and hopefully vice versa) from the involvement of ESRF staff and Ph.D. students, and external users. In particular the following have provided much input and constructive criticism: Peter Bösecke, Andy Fitch, Michael Hanfland, Daniel Häusermann, Martin Kunz, Christian Riekel, Olof Svensson, Karl Syassen, Andy Thompson, Thomas Ursby, Christine Wassilew Reul, Tim and Linda Wess, and Fredrico Zontone. Similarly Ron Ghosh (ILL, Grenoble) has provided much helpful advice.

The author would like to thank Tim and Linda Wess for permission to show the collagen data, which was collected at the SRS, Daresbury, England; Andy Fitch and Phil Pattison for permission to show the olivine data, which was collected on the Swiss-Norwegian Beam-line at the ESRF, France.

References

[1] Adobe Systems Incorporated, "PostScript Language Reference Manual, Second Edition", *Addison-Wesley*, ISBN: 0-201-18127-4, (1990)

[2] Y Amemiya, T Matsushita, A Nakagawa, Y Satow, J Miyahara, and J Chikawa, *Nucl. Inst. Methods*, **A266**, 645-653, (1988)

[3] P R Bevington and D K Robinson, "Data Reduction and Error Analysis for the Physical Sciences", (Second Edition), *McGraw-Hill, Inc*, ISBN 0-07-911243-9, (1992)

[4] I Fujii, Y Morimoto, Y Higuchi, N Yasoka, C Katayama, and K Miki, "Evaluation of X-ray Diffraction Data for Protein Crystals by Use of an Imaging Plate", *Acta Cryst.*, **B47**, 137-144, (1991)

[5] P E Gill, W Murray, and M H Wright, "Practical Optimization", *Academic Press*, ISBN: 0-12-283952-8, (1981)

[6] C J Hall, R A Lewis, B Parker, and T Worgan, "2D detectors for synchrotron X-ray sources, some comparative tests", *Nucl. Instr. Meth.*, **A310**, 215-219, (1991)

[7] A P Hammersley and C Riekel, "MFIT: Multiple Spectra Fitting Program", *Syn. Rad. News*, **2**, 24-26, (1989)

[8] A P Hammersley, S O Svensson, and A Thompson, "Calibration and correction of spatial distortions in 2D detector systems", *Nucl. Instr. Meth.*, **A346**, 312-321, (1994)

[9] A P Hammersley, S O Svensson, and A Thompson, H Graafsma, Å Kvick, and J P Moy, "Calibration and correction of distortions in 2D detector systems", *Rev. Sci. Instr.*, (SRI-94), **66**, 2729-2733 (1995)

[10] A P Hammersley, S O Svensson, M Hanfland, A N Fitch, and D Häusermann, "Two-Dimensional Detector Software: From Real Detector to Idealised Image or Two-Theta Scan", *High Pressure Research*, **14**, 235-248 1996

[11] A P Hammersley, A Thompson, S O Svensson, K Brown, L Claustre, A Freund, A Gonzalez, S McSweeney, and J-P Moy, *ESRF Internal Report*, **ESRF95HA05T**, "Results and Conclusions from Beam-line 10 Calibration and Test Experiments of the ESRF X-Ray Image Intensifier/ CCD Detector (October-December 1994)", (1995)

[12] A P Hammersley, K Brown, W Burmeister, L Claustre, A Gonzalez, S McSweeney, E Mitchell, J-P Moy, S O Svensson, A Thompson, "Monochromatic Protein Crystallography Data Collection Using an X-ray Image Intensifier/ CCD Detector", *J. Syn. Rad.*, **4**, 67-77, (1997)

[13] Hammersley, A. P. and Antoniadis A. (1997). "Reducing Bias in the Analysis of counting statistics data", *Nucl. Instr. Meth.*, **A394**, 219-224

[14] A P Hammersley, *ESRF Internal Report*, **ESRF97HA02T**, "FIT2D: An Introduction and Overview", (1997)

[15] A C Larson, and R B Von Dreele, "General Structure Analysis System (GSAS) Manual", **LANSCE, MS-H805**, Los Alamos National Laboratory, Los Alamos, NM 87545, (1994)

[16] R Kahn, and R Fourme, "Macromolecular Crystallography with Synchrotron Radiation: Photographic Data Collection and Polarization Correction", *J. Apl. Cryst.*, **15**, 330-337, (1982)

[17] P E Kinahan, and J S Karp, *Nucl. Inst. Met.*, **A299**, 484-489, (1990)

[18] G F Knoll, "Radiation Detection and Measurement", *John Wiley and Sons*, ISBN: 0-471-49545-X, 95-103 (1979)

[19] Leslie Lamport, "A Document Preparation System LaTeX User's Guide and Reference Manual", Second Edition, Addison-Wesley Publishing Company, ISBN 0-201-52983-1, 1994

[20] A Messerschmidt and J W Pflugrath, "Crystal Orientation and X-ray Pattern Prediction Routines for Area-Detector Diffractometer Systems in Macromolecular Crystallography", *J. Appl. Cryst.*, **20**, 306-315, (1987)

[21] J P Moy, "A 200 mm input field, 5-80 keV detector based on an X-ray image intensifier and CCD camera", *Nucl. Instr. Meth.*, **A348**, 641-644, (1994)

[22] J P Moy and S Gibney, *SPIE* **1982** (Photoelectric Detection and Imaging 1993), 17-23, (1993)

[23] J-P Moy, A P Hammersley, S O Svensson, A Thompson, K Brown, L Claustre, A Gonzalez, and S McSweeney, "A Novel Technique for Accurate Intensity Calibration of Area X-ray Detectors at Almost Arbitrary Energy", *J Syn Rad*, **3**, 1-5 (1996)

[24] A D Murry and A Fitch, Powder Diffraction Program Library (PDPL), University College London, UK

[25] F Né, D Gazeau, J Lambard, P Lesieur, T Zemb, and A Gabriel, "Characterization of an Image-Plate Detector used for Quantitative Small-Angle-Scattering Studies", *J. Appl. Cryst.*, **26**, 763-773, (1993)

[26] Z Otwinowski, "Oscillation Data Reduction Program", in *Proceeding of the CCP4 Study Weekend: "Data Collection and Processing", 29-30 January 1993, Compiled by L Sawyer, N Isaacs, and S Bailey, SERC Daresbury Laboratory, England*, 56-62, (1993)

[27] R O Piltz, M I McMahon, J Crain, P D Hatton, R J Nelmes, R J Cernik, and G Bushwell-Wye, "An imaging plate system for high-pressure powder diffraction: The data processing side", *Rev. Sci. Instr.*, **63**, 700-703, (1992)

[28] Keith Reckdahl, "Using EPS Graphics in LaTeX 2_ε Documents", Version 1.9, Available as epslatex.ps from ftp://ftp.tex.ac.uk/tex-archive/info/ directory and other CTAN sites, 1997

[29] C C Shaw, J M Herron, and D Gur, "Signal fading, erasure, and re-scan in storage phosphor imaging", *SPIE*, **1651**, 156-162, (1992)

[30] M Stanton, W C Philips, Y. Li, and K. Kalata, "Correcting Spatial Distortions and Nonuniform Response in Area Detectors", *J. Appl. Cryst.*, **25**, 549-558, (1992)

[31] S O Svensson, A P Hammersley, A Thompson, Gonzalez A, and T Ursby, "Measurements and corrections of spatial distortions in imaging plate systems", *CCP4/EPS-EACBM Newsletter*, **29**, (1993)

[32] M W Tate, E F Eikenberry, S L Barna, M E Wall, J L Lowrance, and S M Gruner, "A Large Format, High Resolution Area X-ray Detector Based on a Directly Coupled CCD", *J. Appl. Cryst.*, (In press, 1994)

[33] R H Templer, "Measurements on some characteristics of BaFBr:Eu^{2+}, relevant to its use as a storage phosphor for X-ray diffraction imaging", *Nucl. Instr. Meth.*, **A300**, 357-366, (1991)

[34] D J Thomas, "Calibrating an area-detector diffractometer: imaging geometry", *Proc. R. Soc. Lond.*, **A 425**, 129-167, (1989)

[35] D J Thomas, "Calibrating an area-detector diffractometer: integral response", *Proc. R. Soc. Lond.*, **A 428**, 181-214, (1990)

[36] T J Wess, A P Hammersley, L Wess, and A Miller, "Molecular Packing of Type 1 Collagen in Tendon", *J. Mol. Biol.*, **275**, 255-267, (1998)

28 Appendix A: FIT2D Availability

(Appendix A may be photocopied and freely distributed separately from the rest of the manual.)

28.1 Program Description

FIT2D is a 2-D data analysis program, specialising in model fitting. **FIT2D** is a program which allows difficult data analysis problems to be tackled using 2-D dimensional fitting of user specified models. To enable model fitting to be performed on a wide variety of input data, many other more basic data analysis operations are also available. Calibration and correction of detector distortion has become an important part of **FIT2D**.

FIT2D is still under development, but already a wide range of basic data analysis operations and a number of more sophisticated methods are available. Normal graphics display methods are available.

28.2 Availability

FIT2D is freely available to the academic community and to all ESRF users in return for the normal considerations e.g.

1. Work using these programs should acknowledge accordingly, and in the case of publications suitable citations should be made. (Having ones name occasionally added to the end of a list of authors is clearly not to be discouraged.)
2. There is no warranty that the code and/or methods are correct. (Users should always be very suspicious of any program including **FIT2D**. You should always try testing code with examples which you (think you) know what results should be given).
3. Generally source code will not be willingly circulated, be in some circumstances it will need to be. Users are not allowed to modify code (remove names and add their own). In the case of bug fixes these MUST (for the good of everyone) be send back to me. (Parallel diverging versions need to be avoided.)
4. Comments and other feedback are most welcome, but no guarantee can be given of the possibility to add particular features to the programs. I'll just have to see what I can and can't do. Clearly small changes are much more likely to be incorporated than features which do not fall within the design of the programs.

Non-academic companies will need to make special arrangements.

28.3 Operating Systems and Computer Languages

FIT2D was originally developed on a VAX running VMS, but was later been ported Sun and HP versions of Unix[68], and now runs equally on all common versions of Unix (Dec-OSF1/Dec-Unix, Dec-Ultrix, HPUX-9, HPUX-10, IBM AIX4, Linux, Silicon Graphics IRIX5.3, Silicon Graphics IRIX6.2 SunOS4, SunOS5.4/Solaris).

The vast majority of the code is written in Fortran-77 (plus the MIL-STD-1753 extensions), and is highly portable from one system to another. (As its written in Fortran the frequency of bugs is smaller than a correspondingly sized "C" program as pointers are much less necessary, and run-time array bound checking allows most of the worst bugs to be trapped before they do any harm.) A small amount of code is written in ANSI-C for interfacing to certain POSIX system calls and the X-11 library.

FIT2D is not in any sense Unix dependent (as a good program should not be operating system dependent), however they are to a limited extent POSIX 1003.1 dependent (POSIX 1003.1 being an international standard). As such running the programs on any system which is POSIX compliant should not be a problem. An OpenVMS version would be possible, but so far there has not been sufficient interest. A port to Windows NT will start soon, but this is a more involved task since, whilst POSIX 1003.1 is supported, the X-11 graphics library is not so a graphics driver will need to be written for OpenGL.

28.4 Documentation

An introductory guide: "FIT2D: An Intorduction and Overview" is available as a Reference Manual which covers in detail use of **FIT2D**. The reference manual is presently ≈ 300 pages long and contains an extensive index. Users are not expected to read the manual completely, but to use it as a reference, using the index to find information on the necessary commands needs to perform a particular task. (The programs are hopefully self-evident and contain much on-line help information, but for more detailed explanation the reference manual will need to be consulted occasionally.)

A number of articles are available which cover in more detail the algorithms and methods employed within the program:

- A P Hammersley and C Riekel, "MFIT: Multiple Spectra Fitting Program", *Syn. Rad. News*, **2**, 24-26, (1989)
- S O Svensson, A P Hammersley, A Thompson, A Gonzalez and T Ursby, "Measurements and corrections of spatial distortions in imaging plate systems", *CCP4/EPS-EACBM Newsletter*, **29**, (1993)
- A P Hammersley, S O Svensson, and A Thompson, "Calibration and correction of spatial distortions in 2D detector systems", *Nucl. Instr. Meth. Section A*, **346**, pp312-321 (1994)

[68]Thanks are due to the "Free Software Foundation" (FSF/GNU), without whom a sensible working environment would not have been possible.

- A P Hammersley, S O Svensson, A Thompson, H Graafsma, A Kvick, and J P Moy, "Calibration and correction of distortions in 2D detector systems", *Rev. Sci. Instr.*, (SRI-94), **66**, 2729-2733 (1995)
- A P Hammersley, S O Svensson, M Hanfland, A N Fitch, and D Häusermann, "Two-Dimensional Detector Software: From Real Detector to Idealised Image or Two-Theta Scan", *High Pressure Research*, **14**, 235-248, (1996)
- J-P Moy, A P Hammersley, S O Svensson, A Thompson, K Brown, L Claustre, A Gonzalez, and S McSweeney, "A Novel Technique for Accurate Intensity Calibration of Area X-ray Detectors at Almost Arbitrary Energy", *J Syn Rad*, **3**, 1-5 (1996)
- A P Hammersley, K Brown, W Burmeister, L Claustre, A Gonzalez, S McSweeney, E Mitchell, J-P Moy, S O Svensson, A Thompson, "Monochromatic Protein Crystallography Data Collection Using an X-ray Image Intensifier/ CCD Detector", *J. Syn. Rad.*, **4**, 67-77, (1997)

28.5 Further Questions

This text has been written in order to provide a first general response to the majority of questions concerning **FIT2D**. If having read this document you have further (or still unanswered) questions please feel free to contact me again.

29 Appendix B: FIT2D Macro Language

Macros can be defined within the GUI **MACROS / LOG FILE** interface of within the "KEYBOARD" interface using the `START MACRO` and `STOP MACRO` commands. However, a macro defined in one, cannot be used from the the other. A previously defined "KEYBOARD" macro can be run using the `RUN MACRO` or `MACRO` commands. A previously defined GUI macro can be run using the GUI **RUN MACRO** command.

Once defined appropriately a "KEYBOARD" macro can be run on a whole sequence of files using the `SEQUENCE` command (See Section 15.90, Page 160). A GUI macro can similarly be run on a file series by using the **RUN SEQUENCE** command.

Whilst defining a macro it is possible to call another macro. The contents of the old macro are included within the new macro. By previously defining variables, and using them within macros, macros can be "parameterised" in a primitive manner. Great care must be taken within macros, with commands which can change their input demands e.g. `PRINT GRAPHICS` may or may not prompt for the name of an output file, depending on whether or not an output file is already open[69].

The macro files are ASCII files, so it is possible to modify the macros, but great care is necessary.

(At present **FIT2D** is under development, and modifications to the user interface are likely owing to user suggestions. If new user inputs are demanded old macro files may need to be modified. Contact a member of the Experiments Division Programming Group if you encounter problems.)

29.1 Variables

Macros may be made much more flexible through the use of "variables". Internal program variables may be defined using the `DEFINE VARIABLE` (see Section 15.26, Page 121). and the `VARIABLE` commands. Variables are also automatically defined by certain commands e.g. `STATISTICS`. Variables can be removed by using the `UN-DEFINE VARIABLE` command (see Section 15.119, Page 172).

This is similar to the Unix `alias` command, or the VMS `DEFINE` or `ASSIGN` commands. A "token" can be defined as a variable and given a value. Once defined, whenever the variable[70] is encountered it will be substituted by its value[71].

This is very powerful, but also very dangerous, and potentially very confusing. Users are recommended to only use unlikely character strings for variables e.g. Start every variable with a hash sign (#), but **do not** create variables starting with double hash marks (##) as these are reserved for program generated variables. These may be used inside suitably defined macros, but should not be defined by the user.

[69]`END GRAPHICS FILE` may be used to make sure that the file is closed

[70]The variable must be separated from other characters by one or more spaces, a comma, or a <TAB>.

[71]Before Version 5.8 variables only work for character input e.g. file names and commands, at Version 5.8 treatment of variables for all input, including integer and real constants was added.

Certain commands which produce scalar results also define program variables, so that the values may be used interactively and in macros. All such variables start with a double hash sign (##) e.g. the command statistics produces the following variables `##MINIMUM, ##MAXIMUM, ##MEAN, ##TOTAL, ##RMS, ##SIGMA, ##SKEWNESS` corresponding to the information output on the screen. By entering the variable as a response to a prompt the program will automatically take the "value" of the variable. e.g. The following example will divide each pixel in the *ADR* by the mean value (assuming `##MEAN` has already been defined):

```
Main menu: ENTER COMMAND [INPUT DATA]:cdivid
DIVISION CONSTANT [1.00000]:##MEAN
```

This allows much more flexible macros to be written.

Re-defining an existing variable will overwrite its previous value.

A list of the currently defined program variables and their corresponding values (translations) can be obtained with the `LIST VARIABLES` command.

The creation and usage of simple "KEYBOARD" macros is covered in Section 19, Page 225. Creation and usage of simple GUI macros is covered in Section 8, Page 74.

29.2 More Complicated Macros

More complicated, and powerful macros can be produced. Probably, the best manner in which to produce such macros, is to produce a normal macro, by carrying out the series of data analysis operations once, and then editting the macro file to add, for example, a loop to repeat the operation for a whole series of files.

"Do" loops may be added to the macro to allow counted looping controlled by integer variables. This is best illustrated with an example. The following macro will input and display a series of files `FILE_1010.DAT, FILE_1020.DAT,` to `FILE_1100.DAT`, in binary format.

```
%!*\ BEGINNING OF EXPG_IO MACRO FILE
%!*\
%!*\ This is a comment line
%!*\
DO #COUNT = 10, 100, 10
   I2C
   #COUNT
   no
   3
   #CVALUE
   CONCATENATION
   FILE_1
   #CVALUE
   #CVALUE
   CONCATENATION
   #CVALUE
.DAT
   #CVALUE
   INPUT DATA
   BINARY (UNFORMATTED)
   #CVALUE
   768
   512
   INTEGER (2-BYTE)
   yes
   no
   PLOT DATA
END DO
%!*\ END OF EXPG_IO MACRO FILE
```

The looping is controlled by the `DO` loop, which works like the Fortran-90 `Do` loop. The loop variable, in this case `#COUNT` starting with the first value, and will be incremented up until the second value, in steps defined by the third value. The loop ends when the `END DO` statement is found. Here the commands within the loop have been indented, with the exception of an input string, where we don't want any preceeding blanks.

To form the file name the commands `I2C` and `CONCATENATE` have been used. `I2C` converts the loop integer counter to a character string with a fixed number of characters, so zeros will preceed low numbers. `CONCATENATE` is used to build up the file names from a fixed prefix, the changing numerical part, and the fixed file exetension. The variable `#CVALUE` is set with the file name and used within the `INPUT` command.

Macros can be made interactive, and given a "Graphical User Interface", using the `QUESTION` command. Thus, the simple example above can be made much more sophisticated and versatile, by getting the user to click on the first and last file in a file series, and by using the `DEDUCE FILE SEQUENCE` command to define values of variables which define the loop, and the parts of the file name. The following example shows such a macro:

```
%!*\ BEGINNING OF EXPG_IO MACRO FILE
%!*\
%!*\ fit2d_display.mac : Inputs a series of images and displays together
%!*\
QUESTION
%!*\
%!*\ Definition of interactive input of a value
%!*\ Data value type
INPUT FILE
%!*\ User prompt
CLICK ON FIRST FILE OF SEQUENCE
%!*\ Give default
y
n
%!*\ Help text (up to 100 lines)
This is the first file in a sequence
```

```
of files to be displayed.
\\
%!*\ Program variable
#START_FILE
%!*\
%!*\ End of Definition
%!*\
QUESTION
%!*\
%!*\ Definition of interactive input of a value
%!*\ Data value type
INPUT FILE
%!*\ User prompt
CLICK ON LAST FILE OF SEQUENCE
%!*\ Give default
y
y
%!*\ Default value
G93001.bsl
%!*\ Help text (up to 100 lines)
Click on last file in sequence to be
treated.
\\
%!*\ Program variable
#END_FILE
MESSAGE
FILE SEQUENCE DEFINED
\\
%!*\
%!*\ End of Definition
%!*\
DEDUCE FILE SEQUENCE
#START_FILE
#END_FILE
%!*\
%!*\ Loop through files, input and display them
%!*\
DO #COUNT = ##START, ##END, 10
%!*\
%!*\ Form current file name
%!*\
%!*\
   I2C
   #COUNT
   ##VARIABLE
   ##NUM_CHARS
   #SCOUNT
   CONCAT
   ##PREFIX
   #SCOUNT
   #FILE_IN
   CONCAT
   #FILE_IN
   ##POSTFIX
   #FILE_IN
   CONCAT
   #FILE_IN
   ##EXTENSION
   #FILE_IN
%!*\
%!*\ Input image
%!*\
   INPUT DATA
   BINARY
   #FILE_IN
   768
   512
   INTEGER (2-BYTE)
   YES
   NO
%!*\ Display image
   PLOT
END DO
%!*\ END OF EXPG_IO MACRO FILE
```

Here `##START, ##END, ##VARIABLE, ##NUM_CHARS` are variables which have been defined automatically by the command `DEDUCE FILE SEQUENCE`; see Section 15.25, Page 120. It may be noticed that the `MESSAGE` command has been used to provide some user feed-back on the progress of the macro. Note: The step of 10 has been left in the macro, as the files exist, only in step of 10.

The above example can be made even more flexible with further user input with the `QUESTION` command (see Section 15.77, Page 156) e.g. to define the size of the image to input.

29.3 "KEYBOARD" Interface Commands for Macros

There are a number of "KEYBOARD" commands which are especially useful within macros to help make them versatile and easy to use. Here is list of the commands and their purpose:

`CALCULATOR`: The `CALCULATOR` menu can be used to preform mathematical operations or variable values, and set new variable values with the `VARIABLE` command.

`CONCATENATION`: Concatenate two input strings e.g. stored in two variables, and save in a named variable.

`DEDUCE FILE SEQUENCE`: Deduce components of file sequence and define standard internal variables with the components.

`I2C`: Convert integer to character representation, with control on the number of characters used

`LIST VARIABLES`: Outputs a table of currently defined variables and values. This is very useful for debugging a macro.

`MESSAGE`: Define text message for output to the user during a macro.

`PAUSE`: Wait for user return. This is a method of getting **FIT2D** to pause during a macro. (Useful for debugging and demostrations.)

`QUESTION`: Ask an interactive question during a macro. This is a very powerful command, and allows many different types of input to be defined, with default values, help text, and range checking.

`SLEEP`: Pause for a defined number of seconds. (Useful for slowing down **FIT2D** for demonstrations.)

`VARIABLE`: Define a variable and its corresponding value.

`UN-DEFINE VARIABLE`: Remove variable from the variable translation table.

`WAIT`: Wait for a user return (same as `PAUSE`).

Of course many other commands are also very useful, but these are the ones especially concerned with macros.

In addition to the main menu commands, a few very special instructions may be added to the macro, at any point:

`DO`: Start of a counted loop. See above for further information.

`END DO`: End of a counted loop.

`%SLEEP`: Special pause, which may be placed anywhere within the macro, unlike the `SLEEP` command which only works in the main menu of the "KEYBOARD" interface. The `%SLEEP` instruction is followed by the number of seconds to pause. e.g. `%SLEEP 3` will pause for 3 seconds. A fraction of a second may be specified, but this will only work on certain versions of Unix.

29.4 A Macro Example

The example shows how variables and macros may be defined to allow a sequence of operations to be defined for one file and easily adapted to apply to other files. The example shows how regions of input files may be defined and corrected for spatial distortion, provided of course that the spatial distortion has been previously defined (See Section 16, Page 176). This example should be easy to modify for other sequences of operations which need to be repeated.

Two variables are defined, `#IN` for the input file and `#OUT` for the output file to contain the corrected image. The `START MACRO` command is used to open a macro, and within the macro

the variables #IN and #OUT are used instead of the real files names. The STOP MACRO command closes the macro. Prior to using the macro, the variables are re-defined to different file names. (The text has been created by **FIT2D** using the OPEN LOG command to create a listing of terminal I/O, see Section 15.63, Page 145).
First the variables for the input and output files are defined:

```
Main menu: ENTER COMMAND [VARIABLE]:variable
ENTER VARIABLE NAME [#IN]:#IN
ENTER VARIABLE VALUE [test.dat]:lyso1.gel
Main menu: ENTER COMMAND [INPUT DATA]:variable
ENTER VARIABLE NAME [#IN]:#OUT
ENTER VARIABLE VALUE [lyso1.gel]:lyso1.cor
```

Now the macro is created using the variables rather than the direct file names:

```
Main menu: ENTER COMMAND [INPUT DATA]:start macro
FILE NAME [fit2d.mac]:lyso.mac
Main menu: ENTER COMMAND [SET CURVE STYLES]:input
FILE FORMAT [IMAGEQUANT]:imagequant
DATA FILE NAME [test.dat]:#IN
LEFT-HAND PIXEL OF IMAGE REGION [1]:
LOWER PIXEL OF IMAGE REGION [1]:
RIGHT-HAND PIXEL OF IMAGE REGION (Range: 1 to 1152) [1152]:
UPPER PIXEL OF IMAGE REGION (Range: 1 to 1482) [1482]:
INFO: Full image size =      1152 *      1482 pixels
Main menu: ENTER COMMAND [IMAGE]:reg
X-MINIMUM VALUE (Range: 0.0 to 1152.00) [.50000]:10
Y-MINIMUM VALUE (Range: 0.0 to 1482.00) [.50000]:90
X-MAXIMUM VALUE (Range: 10.0000 to 1152.00) [1151.50]:1120
Y-MAXIMUM VALUE (Range: 90.0000 to 1482.00) [1481.50]:1410
Main menu: ENTER COMMAND [IMAGE]:calibrat
Calibration sub-menu: ENTER COMMAND [SPATIAL CORRECTION]:input
Name of file for spatial distortion interpolation function
FILE NAME [176.spline]:176.spline
Calibration sub-menu: ENTER COMMAND [SPATIAL CORRECTION]:
INFO: The corrected pixel dimension in X is     176.4269 microns
INFO: The corrected pixel dimension in Y is     175.8364 microns
INFO: Number of rows re-binned =    100 (  7%)
```

(Here some program information output has been missed out.)

```
INFO: Number of rows re-binned =   1300 ( 98%)
INFO: Time for re-binning =        57.68 seconds
Calibration sub-menu: ENTER COMMAND [EXIT]:
DESTROY DYNAMIC ARRAYS [NO]:
Main menu: ENTER COMMAND [EXCHANGE]:
Main menu: ENTER COMMAND [IMAGE]:out
```

```
FILE FORMAT [FIT2D STANDARD FORMAT]:dump
DUMP FILE NAME [fit2d.dump]:#OUT
INFO: Data region is   1152 *   1482 pixels
RECORD LENGTH (BYTES) (Range: 1 to 6000) [2304]:
FIRST RECORD FOR OUTPUT (Range: 1 to 100000) [1]:
BIG ENDIAN FORMAT INTEGERS [NO]:y
INFO: Data minimum value =    .0000000E+00
INFO: Data maximum value =    .5378928E+09
LOWER LIMIT OF RANGE (Range: -1.700E+38 to 5.379E+08) [0.0]:
UPPER LIMIT OF RANGE (Range: 0.0 to 1.700E+38) [5.379E+08]:
Main menu: ENTER COMMAND [INPUT DATA]:stop
```

The first image has been, input corrected, and output. In doing this, we have saved the entered operations as a macro. Now the variables can be re-defined to refer to new input and output files:

```
Main menu: ENTER COMMAND [EXIT]:variable
ENTER VARIABLE NAME [#OUT]:
ENTER DATA TYPE OF VARIABLE:string
ENTER VARIABLE VALUE [lyso1.cor]:lyso2.cor
Main menu: ENTER COMMAND [INPUT DATA]:varia
ENTER VARIABLE NAME [#OUT]:#IN
ENTER DATA TYPE OF VARIABLE:s
ENTER VARIABLE VALUE [lyso2.cor]:lyso2.gel
```

Now the command macro is used to run the previously created macro file. The same variables are used, but now they refer to different files:

```
Main menu: ENTER COMMAND [INPUT DATA]:macro
FILE NAME [lyso.mac]:lyso.mac
Main menu: ENTER COMMAND [IMAGE]:INPUT DATA
FILE FORMAT [IMAGEQUANT]:IMAGEQUANT
DATA FILE NAME [grid1.gel]:#IN
LEFT-HAND PIXEL OF IMAGE REGION [1]:
LOWER PIXEL OF IMAGE REGION [1]:
RIGHT-HAND PIXEL OF IMAGE REGION (Range: 1 to 1152) [1152]:
UPPER PIXEL OF IMAGE REGION (Range: 1 to 1482) [1482]:
INFO: Full image size =      1152 *      1482 pixels
Main menu: ENTER COMMAND [IMAGE]:REGION
X-MINIMUM VALUE (Range: 0.0 to 1152.00) [.50000]:1.0000000E+01
Y-MINIMUM VALUE (Range: 0.0 to 1482.00) [.50000]:9.0000000E+01
X-MAXIMUM VALUE (Range: 10.0000 to 1152.00) [1151.50]:1.1200000E+03
Y-MAXIMUM VALUE (Range: 90.0000 to 1482.00) [1481.50]:1.4100000E+03
Main menu: ENTER COMMAND [IMAGE]:CALIBRATION
Calibration sub-menu: ENTER COMMAND [SPATIAL CORRECTION]:INPUT SPATIAL
Name of file for spatial distortion interpolation function
FILE NAME [176.spline]:176.spline
```

```
Calibration sub-menu: ENTER COMMAND [SPATIAL CORRECTION]:SPATIAL CORR
INFO: The corrected pixel dimension in X is     176.4269 microns
INFO: The corrected pixel dimension in Y is     175.8364 microns
INFO: Number of rows re-binned =    100 (  7%)
```

(Here some program information output has been missed out.)

```
INFO: Number of rows re-binned =   1300 ( 98%)
INFO: Time for re-binning =        56.05 seconds
Calibration sub-menu: ENTER COMMAND [EXIT]:EXIT
DESTROY DYNAMIC ARRAYS [NO]:NO
Main menu: ENTER COMMAND [EXCHANGE]:EXCHANGE
Main menu: ENTER COMMAND [IMAGE]:OUTPUT DATA
FILE FORMAT [FIT2D STANDARD FORMAT]:DUMP
DUMP FILE NAME [fit2d.dump]:#OUT
INFO: Data region is   1152 *   1482 pixels
RECORD LENGTH (BYTES) (Range: 1 to 6000) [2304]:
FIRST RECORD FOR OUTPUT (Range: 1 to 100000) [1]:
BIG ENDIAN FORMAT INTEGERS [NO]:YES
INFO: Data minimum value =    .0000000E+00
INFO: Data maximum value =    .1259027E+10
LOWER LIMIT OF RANGE (Range: -1.700E+38 to 1.259E+09) [0.0]:
UPPER LIMIT OF RANGE (Range: 0.0 to 1.700E+38) [1.259E+09]:
```

29.5 Command Files (Scripts) for Batch Mode Processing

Note: The `SEQUENCE` command now allows a macro file to be run automatically on a sequence of input and output files, with much control over the names of the input and output files. For most purposes this will be much easier to use than operating system command files or scripts (See Section 15.90, Page 160). However, under special circumstances the use of a command file or script may allow flexibility which is difficult to achieve with the `SEQUENCE` command.

The ability to define variable names and values on the operating system command line allows flexible control over processing sequences of files through the use of a Unix "script" i.e. an operating system command file which automatically defines calls to **FIT2D** with different variable values

The Unix "C"-shell is designed for operating system programming using a language similar to the "C" programming language. Thus, loops and conditional branches are available. The following is an example of a "C"-shell script written by Thomas Ursby (ESRF Diffraction Group) which automatically processes all files in a directory ending with the same extension.

The following is a genuine example produced by Thomas Ursby including **FIT2D** macro files and Unix script files, allowing flexible and easy correction of spatial distortion. These files show how **FIT2D** macros together with Unix scripts may be used in a very flexible manner. However, clearly this example is only valid for a particular directory structure and users must understand how to edit the file names and programming structures for their own needs. Users who are unsure of "C"-shell programming should consult a specialist or a programming guide or reference book.

(Very minor changes have been made to these files, simply for display purposes.)

29.5.1 Explanation of Files

```
fit2d.batch : corrects all images with a certain extension in a directory.

What I finally used were:

scorr.batch : which reads a file in which the first row contains the name
  of the spline-file, the second row the name of the directory where to
  put the corrected images and then one file name per row.

gen_files : would generate this file if one would prefer not to do it
  manually.

scorr_imagequant.mac, etc : are the Fit2D macros. I guess it would work
  with one macro giving in-file-type as a variable on the command line.

        Thomas
----------
```

29.5.2 fit2d.batch

```
#!/usr/local/bin/tcsh -f
#
#
set datapath = /users/a/ursby/test/
set splinefile = ${datapath}sbgrid1.spline
set macroname = /users/a/ursby/etc/scorr.mac

foreach i (${datapath}*.spe)
  fit2d -dim1242x1152 -svar\#file_in=$i -svar\#file_out=${i:r}.scorr   \
    -svar\#file_spline=$splinefile -mac${macroname}
end
```

29.5.3 scorr.batch

```
#!/usr/local/bin/tcsh -f
#
# NAME: scorr.batch
#
# DESCRIPTION:
# Script for running Fit2D in batch mode correcting images
# for spatial distortion (or whatever the macro named below
# does!). The corrected images end up in a specified directory
# with the extension changed to ".scorr".
#
# CALLING SEQUENCE:
#
# scorr.batch [-type] < files.txt > scorr.log
#
# where
#
#  [-type] is one of
#    -pr    for Princeton CCD camera
#    -ph    for Photometrics CCD camera
#    -md88  for Molecular Dynamics 88um scan (20x25cm plates)
#    -md176 for Molecular Dynamics 176um scan (20x25cm plates)
#  If not given then "-md176" is taken as default.
#
#  files.txt is a file containing (one line per parameter):
#    <name of file containing the spline function>
#    <name of output directory>
#    <name of first file to be corrected>
#    <name of second file to be corrected>
#    etc
#
#  scorr.log is the name of the file where you want the
#    output text to be stored.
#
####
# COMMENTS:
#
# The "files.txt" can be generated by:
#   ls -1 grid.spline > files.txt
#   echo output-directory   >> files.txt
#   ls -1 *.spe >> files.txt
# in the case of a spline function file called "grid.spline"
# and if we want to correct all the images that have the file name
# extension ".spe" in the current directory (images from the
# Princeton CCD camera) and put the corrected images in
# "output-directory".
# Or use "gen_files"
#
# NOTE: The file-names can be preceded by their path.
#
# T.Ursby 12/10/94

# To avoid opening of graphics window
unsetenv DISPLAY

switch ($1)
```

```
case -pr:
  set xdim = 1242
  set ydim = 1152
  set macroname = /users/a/ursby/etc/scorr_princeton.mac
  breaksw
case -ph:
  set xdim = 1024
  set ydim = 1024
  set macroname = /users/a/ursby/etc/scorr_photometrics.mac
  breaksw
case -md88:
  set xdim = 2304
  set ydim = 2964
  set macroname = /users/a/ursby/etc/scorr_imagequant.mac
  breaksw
case -md176:
  set xdim = 1152
  set ydim = 1482
  set macroname = /users/a/ursby/etc/scorr_imagequant.mac
  breaksw
default:
  set xdim = 1152
  set ydim = 1482
  set macroname = /users/a/ursby/etc/scorr_imagequant.mac
  breaksw
endsw

# Read from input file
set splinefile = $<
set outdir = $<
# Temporary file
set tmp_file = `logname`.$$

set i=$<
echo $i > /tmp/$tmp_file
# Loop for each image (given in the input file)
while ($i != '')
  set base = `basename $i`
  echo "************************************************************"
  echo "Image: " $base
  echo
  # Run fit2d for one image
  fit2d -dim${xdim}x${ydim} -svar\#file_in=$i                        \
    -svar\#file_out=${outdir}/${base:r}.scorr                        \
    -svar\#file_spline=$splinefile -mac${macroname}
  set i=$<
  echo $i >> /tmp/$tmp_file
end
```

```
# Make list at end of the log-file of the corrected images
echo "**************************************************************"
echo 'Corrected images:'
echo '(Spline file: ' $splinefile ')'
cat /tmp/$tmp_file
echo "The corrected images were put in the directory:"
echo $outdir
echo "**************************************************************"

# Clean up
/bin/rm /tmp/$tmp_file
----------
```

29.5.4 gen_files

```
#!/bin/csh -f
#
# NAME: gen_files
#
# DESCRIPTION:
# Generate input file to scorr.batch.
#
# CALLING SEQUENCE:
#
# gen_files <grid-file> <base> <image-extension> <output-directory>
#
# where
#   <grid-file> is the name of the grid file.
#   <base> is the path name (if present) and the beginning of the
#     names of the image files to be listed in the output file.
#   <image-extension> is the file extension of the files to be
#     listed in the output file.
#   <output-directory> is the directory where the corrected images
#     are to be put.
#
###
# COMMENTS:
# See scorr.batch.
#
# T.Ursby 12/10/94
#
echo $1
echo $4
ls -1 ${2}*.$3
```

29.5.5 scorr_imagequant.mac

```
%!*\ BEGINNING OF EXPG_IO MACRO FILE
%!*\
%!*\ This is a comment line
%!*\
INPUT DATA
IMAGEQUANT
#file_in
CALIBRATION
INPUT SPATIAL FUNCTION
#file_spline
SPATIAL CORRECTION

EXIT
NO
EXCHANGE
OUTPUT DATA
DUMP
#file_out


YES



EXIT
YES
%!*\ END OF EXPG_IO MACRO FILE
```

29.5.6 scorr_princeton.mac

```
%!*\ BEGINNING OF EXPG_IO MACRO FILE
%!*\
%!*\ This is a comment line
%!*\
INPUT DATA
PRINCETON CCD FORMAT
#file_in
CALIBRATION
INPUT SPATIAL FUNCTION
#file_spline
SPATIAL CORRECTION

EXIT
NO
EXCHANGE
OUTPUT DATA
DUMP
```

```
#file_out


YES


EXIT
YES
%!*\ END OF EXPG_IO MACRO FILE
```

29.5.7 scorr_photometrics.mac

```
%!*\ BEGINNING OF EXPG_IO MACRO FILE
%!*\
%!*\ This is a comment line
%!*\
INPUT DATA
PHOTOMETRICS CCD FORMAT
#file_in
CALIBRATION
INPUT SPATIAL FUNCTION
#file_spline
SPATIAL CORRECTION

EXIT
NO
EXCHANGE
OUTPUT DATA
BINARY
#file_out


YES


EXIT
YES
%!*\ END OF EXPG_IO MACRO FILE
```

30 Appendix C: Menu Commands and Short Description

30.1 Main Menu Commands

? List of available commands and functions

ADD Add active data region of memory to current data

ANNOTATION LABEL Define annotation labels for graphics

BLUR Blur data by a top hat convolution

BRAGGS' EQUATION Variety of uses of Braggs equation

CADD Add constant to active data region

CALCULATOR Reverse Polish Notation calculator

CALIBRATION Calculate or apply calibration functions

CDIVIDE Divide by a constant value the active data region

CLEAR DATA Set data region (and variances) to zero

CLOSE LOG Close log file

CMULTIPLY Multiply active data region by a constant

COLOUR TABLE Choice of colour table and index range

CONTOUR PLOT Graphic display of active data region

CREATE DATA Define blank data region (for simulation)

CURVE STYLES Set attributes for curve representation

DEFINE VARIABLE Define variable and its corresponding value

DIFFRACTION PATTERN Predict part of diffraction pattern

DIMENSIONS Change dynamic array dimensions

DISPLAY LIMITS Set maximum number of displayable pixels

DIVIDE Divide *ADR* of current data by the memory

END GRAPHICS FILE Close previously opened PostScript file

EXCHANGE Exchange current data with the memory contents

EXIT Exit from program

FAST IMAGE Fast output of an image to PostScript

FILTER Fourier Filtering of data with sharp cut-off filter

FIT Least squares fitting with Gaussian functions

FLIP Reflect data through horizontal or vertical middle

FONT Select text font to be used for all Roman text

FUJI LINEARISATION Convert intensities to linear scale

FULL REGION Set active data region to full data region

GAUSSIAN Add 2-D Gaussian to data

GEOMETRY (EXPERIMENT) Define experimental geometry

GRID Define graphics grid requirements

HELP User help and information

IMAGE Interactive control of image display

INFORMATION Information on the internal state of FIT2D

INPUT DATA Input data from data file

LINEARISE FILM Non-linearity corrections to film data

LIST VARIABLES Table of currently defined variables and values

LOGARITHM Take logarithm base 10 of all elements in *ADR*

MACRO Run previously saved macro definition file

MEDIAN FILTER Filter by taking median value within window

MOVE/ROTATE Move and or rotate image, output in memory

MULTIPLY Multiply *ADR* of memory to the current data

NORMALISE Normalise ADR; divide by maximum value within ADR

OFFSET/SCALE Calculate offset and scale between two images

OPEN LOG Open log file for record of input and output

OUTPUT DATA Output data to named data file

PAGE POSITION Set position of data display region on page

PEEP Look at pixel coordinates and value

PIXEL REGION Change active data region using pixel limits

PLOT DATA Plot data as 2-D image (or X-Y graph if 1-D)

POISSONIAN NOISE Add Poissonian noise to data

POWER SPECTRUM Calculate power spectrum of *ADR*

PRINT GRAPHICS Output graphics to a file for printing

PUBLICATION QUALITY Set attributes for high quality output

QUIT Exit from program

RAISE TO A POWER Raise elements in *ADR* to specified power

REBIN Re-bin data, output in memory

RECALL Recall data set from internal memory

REFLECT Reflect data about input line, output in memory

REGION Change active data region

ROI Defined Region Of Interest (Active Data Region) (pixels)

ROTATE LUT Interactive rotation of the colour table

RUN MACRO Run previously saved macro definition file

SELECT PIXEL OPERATION Operation on defined pixel value range

SEQUENCE Run macro for a sequence of files (or not)

SET ANNOTATION STYLE Set style of annotation label text

SET ARROW STYLE Set style of an arrow; style, colour, etc.

SET AXES STYLE Set style of axes, line width, colour, etc.

SET BACKGROUND STYLE Define background colour

SET COLOUR Set colour of graph lines, text, and markers

SET CURVE STYLES Set attributes for curve representation

SET ENUMERATION STYLE Set style for axis numbering

SET FONT Select text font to be used for all Roman text

SET GRID STYLE line type, colour, and width for grid

SET TICK POSITIONS Number, interval of large tick marks

SMOOTH Top-hat convolution smoothing of user input size

SPATIAL FILTERING Filtering in Spatial domain

START MACRO Save commands in macro definition file

STATISTICS Calculate parameters of a region of the data

STOP MACRO Close previously opened macro definition file

STORE Store present data set in internal memory

SUBTRACT Subtract active data region of memory from data

SURFACE INTERPOLATION User defined surface in memory

SYMBOL Same as `DEFINE VARIABLE`

SYMMETRIC FUNCTION Add circularly symmetric function to data

TITLE Input new text for title

THRESHOLD Set minimum and/or maximum values in *ADR*

TRANSPOSE Transpose data arrays and variance arrays

UN-DEFINE VARIABLE Remove variable from translation table

UNIT CELL PARAMETERS Convert between real and reciprocal

V2C Variances to current and vice versa (special command)

VARIABLE Define variable and its corresponding value

VARIANCES DEFINITION Define variances from current data

WEIGHTED AVERAGE data and memory weighted by variances

X-AXIS LABEL Input new text for X-axis label

Y-AXIS LABEL Input new text for Y-axis label

Z-AXIS LABEL Input new text for Z-axis (intensity) label

1-D INTERPOLATION Correct 1-D values by interpolation

3-D SURFACE PLOT Graphics view of active data region

Z-SCALE Image scaling mode, automatic, minimum/maximum

30.2 `CALIBRATION` Sub-Menu Commands

? List of available commands

CALCULATE FITTED DISTORTION output in memory (for X or Y)

DECAY CORRECTION Correct intensity decay (Molecular Dynamics)

DESTROY GRID PEAKS Set "found" peaks to missing peaks

DISPLAY DISTORTION Create 2-D image in memory and display

EXIT Exit fit sub-menu

FIND PEAKS Measure centres of grid mask peaks

FIT GRID PEAKS Fit interpolation function to distortion

FLAT-FIELD CORRECTION Correct for source distribution, etc

HELP User help text

INPUT SPATIAL FUNCTION Recover from file distortion

INVERSE DISTORTED/IDEAL definition of distortion mapping

LEARN HOLE PROFILE Average sub-pixel hole profile

LINEARISE INTENSITIES Correct intensities for non-linearity

OUTPUT SPATIAL FUNCTION Save current distortion function

PLATYPUS CORRECTION FILE Create PLATYPUS calibration file

QUIT Exit fit sub-menu

RE-CALCULATE DISTORTION More user choice on distortion

RESIDUALS OF FIT Fit minus measured distortion (in memory)

SAVE PEAKS Output peak position to ASCII file

SIZE (IMAGE DISPLAY) Change region size for "FIND PEAKS"

SPATIAL CORRECTION Apply spatial correction function

TRANSFER DISTORTION Save distortion values in memory

VIEW PEAKS Examine peak positions and distortions

XRII FLAT-FIELD Theoretical flat-field correction

30.3 `FIT` Sub-Menu Commands

CHANGE SCALE Change the typical parameter variation sizes

CLEAR MASK Eliminate all masked-off "bad" data points

CONSTRAIN Set constraints, change parameters values

COVARIANCE Output covariance matrix

DEFINE MASK Mask-off "bad" data prior to fitting

DISPLAY MASK Display masked-off data as a two level image

EXIT Exit fit sub-menu

INPUT PARAMETERS Initialisation of fit model

MASK STATISTICS Statistics of mask

MINIMISE Search for better fit by maximum descent method

MODEL Create fit model (in memory) from current parameter values

NORMALISATION Uni-directional flat-field normalisation

OUTPUT PARAMETERS Output fit parameters to a file

POWDER DIFFRACTION Fit beam centre and tilt to powder ring

QUIT Exit fit sub-menu

R/THETA RE-BINNING Radial/Angular (polar) re-binning of ADR

RADIAL PROFILE Calculate 1-D radial profile in memory

RESULTS View results of fitting

SET MASK COLOUR User choice of colour to draw masked pixels

SET UP Alter fit control parameters

SURFACE POLYNOMIAL Fit data using 2-D Chebyshev polynomial

THRESHOLD MASKING Mask elements depending on ADR data values

TILT/BEAM CENTRE Determine from a powder ring

TRANSFER MASK TO MEMORY Set masked elements to 1.0 in memory

31 Appendix D: Command Output and Effects

It is sometimes difficult to know where the output of a particular command is placed: in the current data arrays, or in the "memory" arrays. Internally **FIT2D** follows a simple logic, but externally it may be difficult to know the effect of this logic. Where possible the output is in the current data arrays. All operations which involve only single pixel by pixel operations are "in-place" e.g `CADD, CMULTIPLY, LOGARITHM, FUJI LINEARISATION, RAISE TO A POWER`. Similarly all binary pixel by pixel operations produce output in the current data arrays e.g. `ADD, SUBTRACT, DIVIDE, MULTIPLY`. Operations which can swap pairs of pixels in a simple well defined manner e.g. `TRANSPOSE, FLIP` are also "in-place". Operations which in general can involve several input pixels contributing to one or more output pixels use the "memory" for output e.g. `SPATIAL FILTER, REBIN, REFLECT, MOVE/ROTATE`.

Whilst hopefully the logic is clear, it may still be difficult to know where the output appears as the programs internal needs may be unclear. Thus the following tables explain where the output appears. Additionally, the **"Memory" Required** column explains which commands need the "memory" data to be defined before they can be used. The **Current Data Affected** and the **"Memory" Affected** columns explain whether an operation over-writes the contents of either of the sets of program arrays.

The following variables are used in the tables.

BOTH Output in a general sense is in both the current data array and the "memory" array

CURRENT Output in current data array, the previous contents of the current data arrays are over-written, the "memory" may be changed (see "Memory" Affected entry)

FILE Output is in a named file, neither the current data nor the "memory" are affected

MEMORY Output in memory

NONE No output affecting either the current data arrays or the "memory" arrays, however minor changes such as the data title and axis labels may take place

NONE(*), NO(*) Commands which presently have no output nor effect on neither set of arrays, as they are not presently implemented

EXIT Commands which may lead to the destruction of all program arrays through the exiting of the program. User confirmation is demanded

MACRO Commands which run previously saved macros. The effect of such a macro depends on the contains of the macro

MENU Commands which enter a sub-menu which contains commands which produce a variety of different effects. See sub-menu table for details of which commands affect which arrays, and for position of output

Table 1: Main Menu Commands and Effects

Command	"Memory" Required	Output Position	Current Data Affected	"Memory" Affected
`?`	NO	NONE	NO	NO
`ADD`	YES	CURRENT	YES	NO
`ANNOTATION LABEL`	NO	NONE	NO	NO
`BLUR`	NO	MEMORY	NO	YES
`BRAGGS' EQUATION`	NO	NONE	NO	NO
`CADD`	NO	CURRENT	YES	NO
`CALCULATOR`	NO	NONE	NO	NO
`CALIBRATION`	*MENU*	*MENU*	*MENU*	*MENU*
`CDIVIDE`	NO	CURRENT	YES	NO
`CLEAR DATA`	NO	CURRENT	YES	NO
`CLOSE LOG`	NO	NONE	NO	NO
`CMULTIPLY`	NO	CURRENT	YES	NO
`COLOUR TABLE`	NO	NONE	NO	NO
`CONTOUR PLOT`	NO	NONE	NO	NO
`CREATE DATA`	NO	CURRENT	YES	NO
`CURVE STYLES`	NO	NONE	NO	NO
`DEFINE VARIABLE`	NO	NONE	NO	NO
`DIFFRACTION PATTERN`	NO	NONE	NO	NO
`DIMENSIONS`	NO	NONE	YES	YES
`DISPLAY LIMITS`	NO	NONE	NO	NO
`DIVIDE`	YES	CURRENT	YES	NO
`END GRAPHICS FILE`	NO	NONE	NO	NO
`EXCHANGE`	YES	BOTH	YES	YES
`EXIT`	NO	*EXIT*	*EXIT*	*EXIT*
`FAST IMAGE`	NO	FILE	NO	NO
`FILTER`	NO	NONE(*)	NO	NO(*)
`FIT`	NO	*MENU*	*MENU*	*MENU*
`FLIP`	NO	CURRENT	YES	NO
`FONT`	NO	NONE	NO	NO
`FUJI LINEARISATION`	NO	CURRENT	YES	NO
`FULL REGION`	NO	NONE	NO	NO
`GAUSSIAN`	NO	CURRENT	YES	NO
`GEOMETRY (EXPERIMENT)`	NO	NONE	NO	NO
`GRID`	NO	NONE	NO	NO
`HELP`	NO	NONE	NO	NO

Table 2: Main Menu Commands and Effects (Continued)

Command	"Memory" Required	Output Position	Current Data Affected	"Memory" Affected
`IMAGE`	NO	NONE	NO	NO
`INFORMATION`	NO	NONE	NO	NO
`INPUT DATA`	NO	CURRENT	YES	NO
`LINEARISE FILM`	NO	CURRENT	YES	NO
`LIST VARIABLES`	NO	NONE	NO	NO
`LOGARITHM`	NO	CURRENT	YES	NO
`MACRO`	*MACRO*	*MACRO*	*MACRO*	*MACRO*
`MEDIAN FILTER`	NO	MEMORY	NO	YES
`MOVE/ROTATE`	NO	MEMORY	NO	YES
`MULTIPLY`	YES	CURRENT	YES	NO
`NORMALISE`	NO	CURRENT	YES	NO
`OFFSET/SCALE`	NO	NONE	NO	NO
`OPEN LOG`	NO	NONE	NO	NO
`OUTPUT DATA`	NO	FILE	NO	NO
`PAGE POSITION`	NO	NONE	NO	NO
`PEEP`	NO	NONE	NO	NO
`PIXEL REGION`	NO	NONE	NO	NO
`PLOT DATA`	NO	NONE	NO	NO
`POISSONIAN NOISE`	NO	CURRENT	YES	NO
`POWER SPECTRUM`	NO	NONE(*)	NO	NO(*)
`PRINT GRAPHICS`	NO	FILE	NO	NO
`PUBLICATION QUALITY`	NO	NONE	NO	NO
`QUIT`	NO	*EXIT*	*EXIT*	*EXIT*
`RAISE TO A POWER`	NO	CURRENT	YES	NO
`REBIN`	NO	MEMORY	NO	YES
`RECALL`	YES	CURRENT	YES	NO
`REFLECT`	NO	MEMORY	NO	YES
`REGION`	NO	NONE	NO	NO
`ROI`	NO	NONE	NO	NO
`ROTATE LUT`	NO	NONE	NO	NO
`RUN MACRO`	*MACRO	*MACRO*	*MACRO*	*MACRO*
`SELECT PIXEL OPERATION`	NO	CURRENT	YES	NO
`SEQUENCE`	*MACRO	*MACRO*	*MACRO*	*MACRO*

Table 3: Main Menu Commands and Effects (Continued)

Command	"Memory" Required	Output Position	Current Data Affected	"Memory" Affected
`SET ANNOTATION STYLE`	NO	NONE	NO	NO
`SET ARROW STYLE`	NO	NONE	NO	NO
`SET AXES STYLE`	NO	NONE	NO	NO
`SET BACKGROUND STYLE`	NO	NONE	NO	NO
`SET COLOUR`	NO	NONE	NO	NO
`SET CURVE STYLES`	NO	NONE	NO	NO
`SET ENUMERATION STYLE`	NO	NONE	NO	NO
`SET FONT`	NO	NONE	NO	NO
`SET GRID STYLE`	NO	NONE	NO	NO
`SET TICK POSITIONS`	NO	NONE	NO	NO
`SMOOTH`	NO	MEMORY	NO	YES
`SPATIAL FILTERING`	NO	MEMORY	NO	YES
`START MACRO`	NO	FILE	NO	NO
`STATISTICS`	NO	NONE	NO	NO
`STOP MACRO`	NO	NONE	NO	NO
`STORE`	NO	MEMORY	NO	YES
`SUBTRACT`	YES	CURRENT	YES	NO
`SURFACE INTERPOLATION`	NO	MEMORY	NO	YES
`SYMBOL`	NO	NONE	NO	NO
`SYMMETRIC FUNCTION`	YES	CURRENT	YES	NO
`TITLE`	NO	NONE	NO	NO
`THRESHOLD`	NO	CURRENT	YES	NO
`TRANSPOSE`	NO	MEMORY	NO	YES
`UN-DEFINE VARIABLE`	NO	NONE	NO	NO
`UNIT CELL PARAMETERS`	NO	NONE	NO	NO
`V2C`	NO	CURRENT	YES	NO
`VARIABLE`	NO	NONE	NO	NO
`VARIANCES DEFINITION`	NO	CURRENT	YES	NO
`WEIGHTED AVERAGE`	YES	CURRENT	YES	NO
`X-AXIS LABEL`	NO	NONE	NO	NO
`Y-AXIS LABEL`	NO	NONE	NO	NO
`Z-AXIS LABEL`	NO	NONE	NO	NO
`Z-SCALE`	NO	NONE	NO	NO
`1-D INTERPOLATION`	NO	CURRENT	YES	NO
`3-D SURFACE PLOT`	NO	NONE	NO	NO

Table 4: Calibration Sub-Menu Commands and Effects

Command	"Memory" Required	Output Position	Current Data Affected	"Memory" Affected
`?`	NO	NONE	NO	NO
`CALCULATE FITTED DISTORTION`	NO	MEMORY	NO	YES
`DECAY CORRECTION`	NO	CURRENT	YES	NO
`DESTROY GRID PEAKS`	NO	NONE	NO	NO
`DISPLAY DISTORTION`	NO	NONE	NO	NO
`EXIT`	NO	NONE	NO	NO
`FIND PEAKS`	NO	NONE	NO	NO
`FIT GRID PEAKS`	NO	NONE	NO	NO
`FLAT-FIELD CORRECTION`	NO	CURRENT	YES	NO
`HELP`	NO	NONE	NO	NO
`INPUT SPATIAL FUNCTION`	NO	NONE	NO	NO
`INVERSE DISTORTED/IDEAL`	NO	NONE	NO	NO
`LEARN HOLE PROFILE`	NO	MEMORY	NO	YES
`LINEARISE INTENSITIES`	NO	CURRENT	YES	NO
`OUTPUT SPATIAL FUNCTION`	NO	FILE	NO	NO
`PLATYPUS CORRECTION FILE`	NO	FILE	NO	NO
`QUIT`	NO	NONE	NO	NO
`RE-CALCULATE DISTORTION`	NO	NONE	NO	NO
`RESIDUALS OF FIT`	NO	MEMORY	NO	YES
`SAVE PEAKS`	NO	FILE	NO	NO
`SIZE (IMAGE DISPLAY)`	NO	NONE	NO	NO
`SPATIAL CORRECTION`	NO	MEMORY	NO	YES
`TRANSFER DISTORTION`	NO	MEMORY	NO	YES
`VIEW PEAKS`	NO	NONE	NO	NO
`XRII FLAT-FIELD`	NO	MEMORY	NO	YES

Table 5: Fit Sub-Menu Commands and Effects

Command	"Memory" Required	Output Position	Current Data Affected	"Memory" Affected
`CHANGE SCALE`	NO	NONE	NO	NO
`CLEAR MASK`	NO	NONE	NO	NO
`CONSTRAIN`	NO	NONE	NO	NO
`COVARIANCE`	NO	NONE	NO	NO
`DEFINE MASK`	NO	NONE	NO	NO
`EXIT`	NO	NONE	NO	NO
`INPUT PARAMETERS`	NO	NONE	NO	NO
`MASK STATISTICS`	NO	NONE	NO	NO
`MINIMISE`	NO	MEMORY	NO	YES
`MODEL`	NO	MEMORY	NO	YES
`NORMALISATION`	NO	MEMORY	NO	YES
`OUTPUT PARAMETERS`	NO	FILE	NO	NO
`POWDER DIFFRACTION`	NO	MEMORY	NO	YES
`QUIT`	NO	NONE	NO	NO
`R/THETA RE-BINNING`	NO	MEMORY	NO	YES
`RADIAL PROFILE`	NO	MEMORY	NO	YES
`RESULTS`	NO	NONE	NO	NO
`SET MASK COLOUR`	NO	NONE	NO	NO
`SET UP`	NO	NONE	NO	NO
`SURFACE POLYNOMIAL`	NO	MEMORY	NO	YES
`THRESHOLD MASKING`	NO	NONE	NO	NO
`TILT/BEAM CENTRE`	NO	NONE	NO	NO
`TRANSFER MASK TO MEMORY`	NO	MEMORY	NO	YES

32 Appendix E: Conditions of Use

Users of **FIT2D**, exterior to the ESRF, are required to sign the following conditions of use. (Internally users are not presently required to sign the form, but they expected to abide by the conditions.) This form is intended to allow freedom of use for personal academic research provided suitable acknowledgement of the program and methods is given. (For a copy of any of the source code this form and the following source code disclosure form must be signed and returned.)

I .... (print name)

working at .... (name of institute)
agree to abide by all of the following conditions.

1. I will only use **FIT2D** for non-profit making, non-military, academic research.
2. I accept that no guarantee of correctness of methods is given by the author.
3. I undertake to acknowledge both the author and the home institute of **FIT2D** for any "major" use of the program. This means acknowledging A P Hammersley and the ESRF for use of **FIT2D**. The following use should be considered "major" i.e. requires acknowledgement.

   - Detector distortion calibration and correction
   - Use of any data analysis operation not in the list below

   The following use does not require any acknowledgement:

   - Graphical display and output (without "major" data analysis)
   - Trivial data analysis i.e. addition, subtraction, multiplication, division pixel by pixel or by scalars. Basic mathematical functions applied pixel by pixel (Log, exponention, square root)
   - File format conversion, including the following image transformations: reflection, rotation, transposition

4. I agree to cite appropriate references (as described within the **FIT2D** Reference Guide) for publications which result from "major" data analysis using **FIT2D**.

.... (sign here)

Date: ....

Completed forms may be posted to:

A P Hammersley
ESRF
BP 220
GRENOBLE 38043
CEDEX
FRANCE

The form may be initially faxed, in order to obtain the software "key" more quickly:

Fax: (+33)-4-76-88-25-42

Further enquires may be made by e-mail:

E-mail: hammersley@esrf.fr

33 Appendix F: Conditions of Disclosure of Source Code

See next page.

Source code will be disclosed only when absolutely necessary, and in return for signed copies of both the "Conditions of Use" form and this "Disclosure of Source Code" form. The conditions are required: to protect the copyright of the author(s); to prevent overly high expectations by programmers working with the code; to ensure that beneficial modifications are generally available; and to prevent divergent versions.

I .... (print name)

working at .... (name of institute)

using the following computer systems: ....

.... (give make, type, and where available Internet address of computer system(s))

agree to abide by all of the following conditions.

1. I accept that all source code is the copyright of the author(s) and I will not remove the author(s) name from any of the source code nor from the program banner, nor use the code for any other reason (without additional approval).
2. I will not allow access to the source code to any third party, nor copy it to any computer system other than the one(s) listed above.
3. I will not allow use of the program by any user who has not signed the "Conditions of Use" form.
4. I will ensure that users of the program are given adequate access to the available documentation. (The documentation may be reproduced and distributed to users who sign the "Conditions of Use" form.)
5. I undertake to answer users questions and will try to solve their problems of usage to the maximum of my ability, without recourse to the author(s), except when absolutely necessary.
6. I accept that both the form (subroutine/procedure names and arguments) and the functionality of the source code may change in future versions.
7. I accept that even after any "minor" modification(s) the source code or documentation remains the complete intellectual copyright of the original author(s), and I undertake to send to the author(s) (preferably by e-mail) an example of all "minor" modifications. "Minor" modifications which may be made are:
   - Changes required to overcome compilation errors or warnings
   - Changes required to overcome errors in the data processing
   - Changes to the default choices offered by **FIT2D**.
   - Correction of inaccurate documentation.
8. I will not make any "major" modifications. (Except under the circumstances of an additional agreement.) "Major" modification means any modification not defined as "minor".
9. I will allow executable version(s) to be copied to the ESRF for general availability to the academic community under the "Conditions of Use" conditions.

.... (sign here)

Date: ....

34 Appendix G: Collaborative Development

Any programmers interested in adding new or improved features and functionality into **FIT2D** are welcome to contact me. Extra manpower is always useful. **FIT2D** may well provide a good framework for your data analysis developments e.g. An environment of file input/output and data display already exists together with facilities to allow definition and execution of macros. Such added code would remain the copyright of the author(s), but the author(s) would have to accept the distribution of the code under the same conditions as given in Appendices E and F.

Such code would need to follow a number of general guidelines which are necessary in order to maintain the integrity of **FIT2D** i.e. prevent one part of the program corrupting another part:

- In order to make memory corruption as difficult as possible, all code (or almost all code) should be in FORTRAN-77 with only a few generally available extensions (or hopefully soon Fortran-90) and be able to be compiled with array bound checking i.e. with the `-C` option for Unix Fortran compilers or the `CHECKS=ALL` option for the VMS Fortran compiler. All code should be tested and run with array bound checking enabled.
- Use of unconstrained pointers should be non-existent or almost non-existent. At present unconstrained pointers are necessary to allow dynamic memory allocation, but with Fortran-90 this need will disappear. (When this exception there is only one point in the whole of **FIT2D** where unconstrained pointers are useful.)
- All code should include "defensive programming" code to protect as much as possible against mis-use of the code. The last argument of every routine must be the `status` return variable. On input this variable must be checked to correspond to the "good status" value. If not routines must call the subroutine `EXPG_ST_SAVE` to report the apparent bad status, and then immediately return. All possible input arguments should be checked for allowed values. If values are not appropriate the error condition should be reported through the `status` value and a call to `EXPG_ST_SAVE`. After an error condition has been found in one of the input arguments all routines must return without any further action. Very low level repeatedly calls routines may be allowed not to include this "defensive programming" code when appropriate, provided higher level calls can filter all possible mis-use. (Examples and templates of use of the system can be supplied.)
- When Fortran-90 becomes generally available for the main target machines it will be additionally required to provide an "interface" for each subroutine and function.
- All code should be well documented, preferably in the standard style of **FIT2D**. Templates can be supplied. (An automatic documentation extraction program exists for code following a well defined structure.)
- Programmers are strongly encouraged to follow the following conventions for typing the names of variables, constants, arrays, functions and subroutine calls:
  - **Scalar constants:** First letter upper case, followed by lower case only
  - **Scalar variables:** Lower case only
  - **Arrays:** Upper case only

  - **Function calls:** First letter upper case, followed by lower case
  - **Subroutine calls:** Upper case only
  - **Fortran function keywords:** First letter upper case, followed by lower case
  - **Pointers:** Lower case `p` followed immediately by all upper case

  Use of these conventions makes understanding the program and program structure much easier, and may allow automatic identification and processing of particular elements e.g. pointers.

- Interfacing to POSIX 1003.1 operating system functions may use "C", but this should be made immediately callable by Fortran, taking into account both Fortran compilers which add underscores to the end of external symbols, and those which do not. Similarly the different handling of characters strings between Fortran and "C" must be foreseen.

Index