Platypus bootcamp

Dr Andrew Nelson

August 5, 2010

Contents

1 Important checks before use

Before using the instrument the following things should be checked and understood.

1. Go through the Platypus SWMS and understand it.
2. Make sure you have the requisite training to use the instrument. The training includes a knowledge of the safety systems and any radiological issues that we need to consider. If you don't know these, ask an instrument scientist.
3. Before the tertiary shutter is opened make sure that the slits are closed (with a visual check) and that the chopper is operating with the correct frequency and phase. If you are unsure that this is the case ask an instrument scientist for help. If these two points are not followed then you WILL DAMAGE the detector.
4. You will probably start off using FIZZY (7), in this case you don't need to read most of the other sections first off, but you should familiarise yourself with them at some stage.

2 Chopper system

The chopper system pulses the neutron, at around 20Hz. There are many neutron wavelengths (energies) contained in each pulse. As the neutron pulse is allowed through a stopwatch is started on the data acquisition computer, via a signal from the choppers. When each neutron is detected (at the detector) it is possible to work out the time-of-flight for that neutron to travel the set distance from the chopper to the detector. Because we know distance and time we can work out velocity, and therefore the wavelength of the neutron from de-Broglies relationship.

2.1 Startup

The ASTRIUM rack should be powered on first, only an instrument scientist is allowed to do this.

Cooling water The cooling water should be being supplied at ~16 degrees at a rate of ~4l/min. Check that the flow rate is high enough on the dc-1 computer in the instrument cabin. Check that there are no leaks and that there is enough water in the reservoir.

Vacuum The chopper housing should be at a pressure <2e-2 mBar, and be constantly being pumped by the turbo pump.

1. Close scroll-collimation tank valve and turbo-chopper valve. Scroll-Chopper valve is open.
2. Pump for ~30 minutes with scroll pump.
3. Turn turbo pump on (turbo-chopper valve still closed).
4. When the pressure reading on the turbo pump gauge matches that in the chopper housing *slowly* open the turbo-chopper valve.
5. Let both pumps work together for ~15 minutes, slowly opening turbo valve as you go.
6. Now close chopper - scroll pump valve. If the rotation speed of the turbo drops, use scroll pump again aid turbo pump.
7. Check that the turbo pump reaches 1500Hz before walking away, otherwise turbo will shut down.

Other error warnings If there are other problems with the choppers then red warning LEDS will appear on the ASTRIUM rack. The choppers will not start until these are all extinguished. Investigate the other warning LED's further.

2.2 Operation

Once the steps in startup have been followed then one can run the NCS013 chopper control program. Do not start this program up before the chopper rack is switched on. If the program starts up without errors then:

1. check the following
   (a) the water flow rate is >4l/min
   (b) the water temp is less than 20 degrees
   (c) the vacuum is low enough (in the green region)
   (d) the vibration level is not high.
   (e) Any other chopper status notices
2. if 1) is ok then press RESET CONTROLLERS button
3. press CALIBRATE, the chopper status should turn from inactive to commutation.

The choppers are now ready to use. Chopper speed is entered in rpm, not Hz. Further details are available in the operating manual.

2.2.1 Chopper phasing and speed

The choppers are designed to be run in synchronous mode, i.e. with chopper x slaved to chopper 1. This is easy to achieve on the disc chopper controller computer. Chopper x will need to be set as a) synchronous and b) masterwise (spins in same direction as chopper). If the setting is correct then you will see chopper 1 spin with a speed of e.g. 1380 rpm, chopper 2 with a speed of -1380 rpm, chopper 3 with a speed of -1380 rpm and chopper 4 with a speed of 1380 rpm. (you will only use one of 2, 3 or 4. The reason why the speed is negative for discs 2 and 3 is due to the chopper motors being mounted in the opposite direction.

For a typical reflectometry measurement the choppers are normally phased such that the window of the second chopper used just opens as the window of the first chopper closes. Thus if chopper 1 has a window of 60 degrees and chopper 2 has a window of 10 degrees, then a nominal phase angle of 35 degrees needs to be used (angles are taken from the middle of the window).

However, because of the way the choppers are mounted one needs to make sure the correct polarity is used. As of 24/03/2009 the calibrated window openings are:

chopper 2: 35.02
chopper 3: 42.17
chopper 4: -59.79

IMPORTANT: AFTER YOU HAVE CHANGED/RESET THE CHOPPER SETTINGS YOU NEED TO ISSUE THE SICS

```
::chopper::ready?
```

COMMAND. This is because SICS needs to have the updated values of the chopper speeds and phasings, otherwise they won't be saved in the datafiles correctly. Consequently all data reduction based on those files will be wrong.

3 SICS

3.1 What is SICS?

SICS (SWISS INSTRUMENT CONTROL SOFTWARE/SYSTEM), is a computer program running on a LINUX box. SICS is a command line program that is responsible for keeping track of all motors, moving the motors, starting data acquisitions, etc. There are specific commands that SICS accepts, which will be referred to as SICS commands in this manual.

3.2 server address

```
ics1-platypus.nbi.ansto.gov.au
```

3.3 starting SICS

ssh to the server from Cygwin/Putty:

```
ssh platypus@ics1-platypus
```

navigate to SICS directory:

```
cd /usr/local/sics/server
```

start the server, with the Platypus configuration file. The configuration file says what motors are attached, etc.

```
runsics stop
runsics start
```

To gain access to the SICS command line:

```
socat READLINE,history=history tcp4:127.0.0.1:server-platypus,crlf
```

You then have to give a username and password, please ask the instrument scientist for these if you require SICS to be rebooted.

3.4 Important SICS commands

3.4.1 SICSLIST - getting list of SICS components

List all the SICS commands:

```
sicslist
```

To get a list of the available motors type:

```
sicslist type motor
```

3.4.2 SICSLIST TYPE MOTOR - List the motors available to be driven

Here is a list of the motors and their explanations. The frame of reference is looking down the instrument from the reactor towards the detector.

SICS axis	real name	details
SS1L	slit1 left blade	
SS1R	slit1 right blade	
SS1U	slit1 upper blade	
SS1D	slit1 down blade	
SS1VG	slit1 vertical gap	the space between SS1U and SS1D (~2mm)
SS1VO	slit1 vertical offset	the vertical offset from the middle of SS1VG to the beam centre
SS1HG	slit1 horizontal gap	the space between SS1L and SS1R (~50mm)
SS1HO	slit1 horizonal offset	the vertical offset from the middle of SS1HG to the beam centre
SS2<X>		slit 2 has the same submotors as ss1
C1HT	collimation tank translation	translates different collimation elements into beam
M1RO	mirror1 rotation	rotates deflection mirror to deflect neutron beam downwards
SS3<X>		slit 3 has the same submotors as ss1
ST3VT	slit3 vertical tower	moves the entire slit 3 package vertically
SZ	sample z	changes the height of the sample. Watch out for collisions of sample stage and slit4.
SX	sample x	translates different samples into the beam
STH	sample theta	changes the pitch of the sample stage with respect to incident neutron beam
SPHI	sample phi	changes the roll of the sample stage with respect to the incident neutron beam
SS4<X>		slit 4 has the same submotors as ss1
ST4VT	slit4 vertical tower	moves the entire slit 4 package vertically
DY	detector y	moves the detector longitudinally (horizontally)
DZ	detector z	moves the detector transversely (vertically)
BZ	beamshade z	moves the beamshade in front of the detector

3.4.3 RUN / DRIVE - Moving a motor

There are two ways of moving a motor. One command, DRIVE, doesn't let you do anything else in SICS until the motor has finished moving. The second, RUN, allows you to move many motors at the same time.

Change the vertical gap of slit 3 to 10mm, don't allow any other SICS commands be entered while you are moving:

```
drive ss3vg 10
```

Change the vertical gap of slit 3 to 10mm, allow any other SICS commands be entered while you are moving:

```
run ss3vg 10
```

To get an update of the motor position, whenever it changes type:

```
ss3vg interest
```

To deregister that interest type:

```
ss3vg uninterest
```

You can also get updates by using:

```
hnotify / 1
```

Which specifies that you want to be informed about anything happening on the HIPADABA path / (which is the root path for all axes). The hipadaba system is a way of containing the instrument information, it is akin to the filesystem on a computer. For example the hipadaba path for ss2l is:

```
/instrument/slits/second/horizontal/left
```

To get the hipadaba path for a specific motor type:

```
sicslist <motorname> hdb_path
```

3.4.4 MOTORNAME - get the current motor position

type the motor name, the current motor position will be printed to screen

```
motorname
```

3.4.5 LIST - Parameters associated with an axis

To get a list of the parameters associated with the motor, such as position, softzero, softlimits, hardlimits, etc type:

```
motorname list
```

To change any of these parameters you would type:

```
motorname parameter newvalue
```

For example, the following code sets the softlowerlimit of sphi to -5 degrees:

```
sphi softlowerlim -5
```

3.4.6 fixed motors

Some of the motors you use may be fixed, probably with good reason. So don't bother with this section unless you are an instrument scientist.

Fixing the motors prevents them from being moved. You can check this with:

```
motorname fixed
```

It will return the current fixed status. To free the motor again type:

```
motorname fixed -1
```

3.4.7 Limits

Don't change these unless you know what you are doing, not even then, speak to Andy / Ferdi /Doug.

Hardlimits Hardlimits are the axis positions at which the motor will drive into the upper and lower limit switches.

Softlimits Softlimits are the software limits on how far each axis may be driven. This is to prevent the motor from ever driving onto the HARDLIMITS. The RUN/DRIVE commands can only issue orders if the new position is within the softlimits

3.4.8 MOTORNAME HOMERUN 1 - Homing slits 1,2,3,4

slit systems 1,2,3 and 4 (not st3vt and st4vt) are not on absolute encoders. This means that after a power outage the slit positions are lost. This is bad because SICS may say the slit gaps are small, when in fact they are large. This may let too many neutrons through and overwhelm the detector. The slits must be homed before use, enter *all* the following commands:

```
ss1u homerun 0
ss1u homerun 1
drive ss1vg -1
drive ss2vg -1
ss3u homerun 0
ss3u homerun 1
drive ss3vg -1
drive ss4vg -1
```

You are now in a position to consider what slit positions you need for the experiment.

3.4.9 SETPOS / SOFTZERO - changing offsets for a motor

When calibrating the instrument we will need offsets. Each motor axis has an absolute scale. Each axis has a corresponding SOFTZERO, which is where all DRIVE and RUN commands are offset from.

For example ST3VT may have an absolute range of -20 to +20 (mm) on the motor hardware. If the softzero is changed to +20, then you may enter RUN/DRIVE commands between -40 and +0. If you changed the softzero to -100 then you could enter drive commands between -120 and -80.

SOFTZERO The command for changing the softzero is:

```
motorname softzero 20
```

so for the example given above you would type:

```
st3vt softzero 20
```

SETPOS The SETPOS command can also change the SOFTZERO. This is often more useful than the SOFTZERO command.

It has two forms. The first form takes a position OLD, and makes it NEW.

```
motorname setpos old new
```

For example:

```
st3vt setpos 20 40
```

changes the SOFTZERO such that the position of 20 would now be 40 (in effect this is the same as setting the softzero lower by 20).

The second form takes the current position and sets it to NEW:

```
motorname setpos new
```

For example:

```
st3vt setpos 100
```

Would change the SOFTZERO such that the current position of st3vt is now 100.

3.4.10 *bat* - using and troubleshooting the beam attenuator

The beam attenuator rasters the beam over the detector to ensure that the detector is not burnt. This should be used for measuring the direct beam + the critical angle.

To park the beam attenuator in the beam:

```
bat send oscd=0
```

The SEND command is actually a motor controller command. To remove the beam attenuator:

```
bat send oscd=-1
```

To oscillate the beam attenuator (which is its intended use):

```
bat send oscd=1
```

Sometimes thread0 has stopped, which is the computer process that drives the oscillation. To start this you have to reset the motor controller:

```
bat send rs
```

To see if thread0 is functioning type:

```
bat thread0
```

If the number changes on several uses of this command, then thread0 is probably running (number refers to line in code). If thread0=-1 the thread has stopped.

To list motor controller variables type:

```
bat send lv
```

To see if the bat axis is on either of the limit switches type:

```
bat send mg _lr‘,lf‘
```

If it is not on the limit switches both values are 1.

3.5 Troubleshooting motors

Sometimes one experiences problems with motors, the following SICS commands can help. We'll concentrate on *sth*, although they apply equally to all other motors

3.5.1 Listing the variables associated with a motor

(speed, direction, encoder counts, etc

```
sth send lv
```

3.5.2 Seeing if the motor controller has seized

```
sth thread0
```

Is the value returned==-1? If so you have to restart the motor controller.

3.5.3 restarting the motor controller for a specific motor

```
sth send xq
```

3.5.4 Getting the encoder counts on a motor

```
sth send tp‘
```

4 Histogram Server

The Histogram server, das1-platypus, is the computer that does all the data acquisition. In otherwords, it interfaces to the chopper system, SICS and the detector and possibly your sample environment. The program that does all this work is actually a webserver and can be viewed in a web browser. The program that you use to acquire data, FIZZY, downloads data from the webserver, and displays the detector pattern in a nice image.

4.1 Server address

```
das1-platypus.nbi.ansto.gov.au
```

4.2 Viewing histogram server configuration on a webpage

The histogram can be fully configured and run from a web browser (I recommend Firefox). You need to go to the address:

```
http://137.157.202.140:8080/admin/viewdata.egi
```

Ask the instrument scientist for a user name and password to view this.

4.3 Getting data using CURL (instrument scientists)

The data on the histogram server can be programatically downloaded using a command line client such as CURL, or by using the easyHttp XOP in IGOR. You will need to work out the correct HTTP request to give to those clients. This is fairly simple to do, but some features are hidden/not documented:

1) */admin/textstatus.egi* returns the text status of the instrument (including count rate, total counts, etc).

2) when downloading the data be sure not to use the gui version, as this affects the webpage, e.g.

```
http://137.157.202.140:8080/admin/saveviewdata.egi
```

instead of

```
http://137.157.202.140:8080/admin/saveviewdatagui.egi
```

It is also easy to download the histogram config file, change it with an XML editor, then re-upload it. One can also start and stop the histogram server using this type of approach.

4.4 Starting histogram server/beam monitor servers (instrument scientists)

Using Cygwin/PUTTY ssh to the server:

```
ssh root@das1-platypus
```

To stop and start the server type the following command:

```
/etc/init.d/ansto_histogram_server stop
/etc/init.d/ansto_histogram_server start
```

To (restart) the beam monitor type:

```
/usr/local/beam_monitor/start_beam_monitor.sh
```

4.5 Connection to chopper signal (instrument scientists)

Make sure that a BNC cable runs from the CS1 connector to the INPUT A1 PORT on the ORTEC DUAL SUM AND INVERT CARD. This carries the TTL pulse from the chopper opening to the histogrammer, triggering the start of a new frame. If this isn't done your data will be crap.

If this is done correctly a red LED will flash with a frequency equal to the chopper frequency (choose a frequency of 1 Hz to check).

4.6 Configuring the histogram server (instrument scientist)

Once you have setup the instrument and started the histogram server you are in a position to do scans, runs, etc. The instrument must be configured correctly, otherwise the data is gibberish. This is especially important for things like the chopper frequency.

4.6.1 Configuring the OAT_TABLE

The OAT_TABLE configures the pixellation on the histogram server, i.e. the bin boundaries in the x, y, t dimensions. The larger the number of bins, the greater the file size, so beware. The OAT_TABLE is typically configured with the following SICS command:

```
oat_table -set Y {-110.5 -109.5} NYC 221
oat_table -set T {0 43} NTC 1000
```

To set X and T bins use NXC and NYC respectively

This would configure 221 bins in the Y direction, with the leftmost bin position starting at -110.5 and having a width of 1. The OAT_TABLE command then generates the rest of the bins based on this bin width. Do not set

negative times for the time bins, that doesn't make sense. The *times are entered in microseconds*. The 1000 time bins requested means that the last bin ends at 42957 microseconds, which is 0.043 seconds. This time channel setting might be appropriate for a chopper operating at 23 Hz (period = 0.0434sec). ***Do not set the oat_table up to have a longer time period than the chopper period.***

The OAT_TABLE configuration then needs to be uploaded from SICS to the histogram server. This can be accomplished by the following SICS command:

```
histmem loadconf
```

4.6.2 Making sure the histogram server / SICS are using the right chopper frequency

If the chopper frequency/phasing is changed then SICS and the histogram server must be updated. To check what SICS thinks is the frequency type:

```
histmem freq
```

This should be the same as the chopper frequency as listed on the chopper computer. You can use the CONFIG tab of the histogram server webpage to check it's value. If the two aren't the same as the chopper type:

```
histmem freq 23
histmem loadconf
```

If the chopper frequency is 23 Hz.

This will update the values contained in SICS, and will cause the histogram servers version to be updated. It's worth checking again after this. The

```
::chopper::ready?
```

command should also be issued after any chopper velocities/phasings have changed.

5 More useful SICS commands

These commands can ONLY be issued from SICS.

5.0.3 Drive a motor

In blocking mode (no other motors are allowed to run):

```
drive sth 0.1
```

In non-blocking mode (other motors are allowed to run):

```
run sth 0.1
```

5.0.4 Entering offsets

e.g. sets the current position of sth to be 0.5

```
setpos sth 0.5
```

e.g. sets sth such that a value of 1.2 becomes 0.5

```
setpos sth 1.2 0.5
```

5.0.5 Getting into the SICS command line from an SSH terminal

```
sicsclient
```

5.0.6 Starting the sics server (issued from a terminal, not SICS itself)

```
runsics start
```

5.0.7 stopping the sics server (issued from a terminal, not SICS itself)

```
runsics stop
```

5.0.8 Leaving the SICS command line

```
logoff
```

5.0.9 Listing the variable associated with a motor (speed, softzero's, etc)

```
sth list
```

5.0.10 Changing the histogram frequency

```
histmem freq 23
```

5.0.11 Updating SICS after changing the chopper speed/phasing

When you alter the chopper speed or phasing you need to get SICS to update it's information. To get SICS to query the chopper server issue the following command:

```
chopperController status
```

5.0.12 Changing the histogram OAT_TABLE

```
oat_table -set Y {-110.5 -109.5} NYC 221
histmem loadconf
```

5.0.13 Setting user metadata e.g. sample name, phone number, email

This metadata is saved with every NeXUS file that is created. At a minimum the sample data should be entered, otherwise no one will know what the data is.

To enter a sample name type:

```
samplename Cobaltate-Iron mix. Prepared on 12/3/03.
```

The sample name is everything after the SAMPLE command. One can also store the TITLE, USER, EMAIL, PHONE.

5.0.14 SAVING A NEW FILE

```
newfile HISTOGRAM_XYT
```

NEWFILE tells SICS to start a new NeXUS datafile to write to. If this command is not issued then consecutive SAVE commands will overwrite (or append to) the data contained in the last file.

5.0.15 SAVING DATA

```
save <number>
```

this saves the data currently in the histogram memory to the latest NeXUS file. The data will be inserted into the NUMBER'th spectrum in the file. This has the possibility of overwriting previous data, unless NUMBER is incremented by one each time (such as in a temperature run), or unless the NEWFILE command has been issued. One can save data, even if the data stopped acquiring hours ago (as long as a new acquisition wasn't started).

6 IGOR Pro

6.1 Quick Introduction to IGOR

IGOR is a data analysis package which can be used for many purposes. It handles data in many different ways, you will use it to acquire data (with FIZZY), reduce data (with SLIM) and analyse data (Motofit).

Data is encapsulated in IGOR in different forms, listed below. To get a full handle of what IGOR can do please use the "Getting started" tutorial, which is accessed through the "Help->Getting Started" menu.

6.1.1 Waves

Waves can be from 1 to 4 dimensional. Each dimension can have as many points as you like. You can think of Waves as matrices or arrays (one can do many different matrix operations on them). Waves can hold either numeric data or

text information. The numeric data can be real or complex, and it can be integer, float or double. IGOR has a rich suite of inbuilt routines for operating on waves, such as image processing, matrix math, statistics, etc. If there is a bit of analysis that's not builtin you can write it yourself in what is called a procedure.

6.1.2 Procedures

Type CTRL-M, the IGOR procedure window pops up. You can type code in here to write your own analysis routines, or control the instrument.

An example routine could be:

```
Function test(a,b,words)
variables a, b
string words
variable c
c = a+b
print words,c
End
```

This function is called TEST. The function has three ARGUMENTS called A, B and WORDS. A and B are variables (or numbers), whilst WORDS is a STRING. A string contains text, such as "abc123". This function adds a and b, then prints the sum to the command window, along with the string that you sent to the function. IGOR procedures can be written to do any analysis you like, using all the inbuilt functionality of IGOR makes things easier. Please note that we could've used a wave as an argument as well. A collection of these IGOR functions is good enough to run Platypus.

6.1.3 Command line

The results from all the analysis you do often turns up in the HISTORY section of the COMMAND WINDOW. The history contains a record of what you've asked IGOR to do. In the command line you can issue your own commands. For example you could write the following command:

```
test(1,10,"the sum was: ")
```

If you had created the test function this would've printed out;

THE SUM WAS: 11

in the history window.

To get to the command window type CTRL-J.

6.1.4 Strings

Strings are variables that contain text. e,g,

string test = "Hello world!"

String variables can be passed to functions.

6.1.5 Variables

Variables contain real or complex numbers.
e.g. variable foo = 10
Variables can be passed to functions.

7 FIZZY - operation of Platypus via a GUI, in IGOR Pro.

FIZZY is a set of code written in the IGOR Pro language that is designed to operate an instrument running SICS. Within the FIZZY program there is a terminal window in which you can type SICS commands. However, IGOR also has a command window, in which you can enter IGOR commands. Please note that IGOR COMMANDS AND SICS COMMANDS ARE TOTALLY DIFFERENT AND MUST NOT BE MIXED UP. Please see 7.2 for further details.

7.1 Starting FIZZY up

FIZZY must be started after SICS starts. Select the menus PLATYPUS -> FIZZY. You are then prompted for a SICS username and password, the instrument scientist will give these to you.

7.1.1 FIZZY console

Figure 1 is the main screen of FIZZY.

SICS terminal

On the left hand side there is a terminal window where you can enter SICS (not IGOR) commands. Previous commands are highlighted in blue. Output from SICS is in black. Here we have asked about the vertical gap for slits 1 and two, and changed the title of the experiment.

Current motor positions

On the right side is a list of motors, their current positions, and the lower and upper limits for each of the motors. One can move a motor by a) issuing a SICS command in the terminal or b) by clicking in the current position box for a motor, entering the desired value and pressing return.

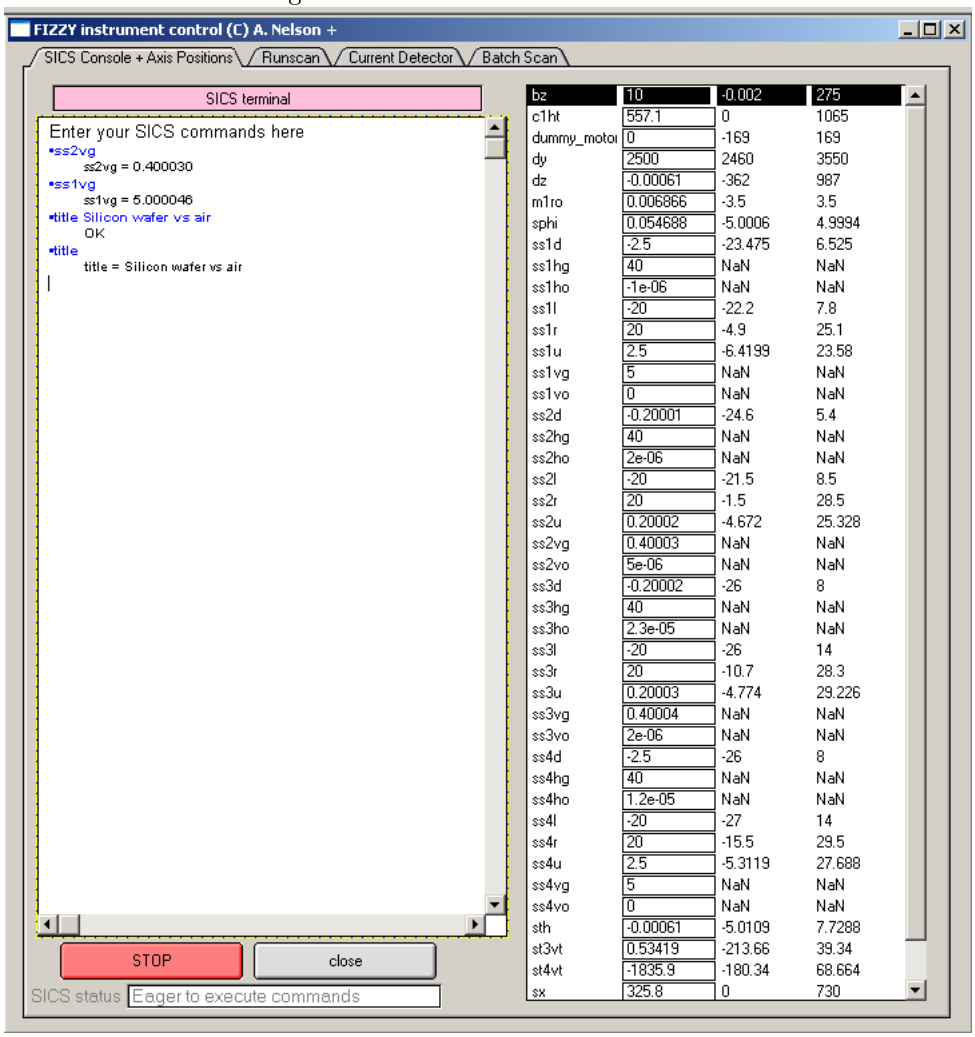
Stop and close buttons

If you experience any difficulties, such as the motion positioning crashing into something, you can press the STOP button. This stops all the motors from moving, shuts all the slits and puts the attenuator in. The close button shuts FIZZY down.

SICS status

This box tells you what SICS is doing, e.g. Driving, Counting, etc. The only way you can be sure that SICS will do what you want is if it says "Eager to execute commands".

Figure 1: FIZZY terminal window



7.1.2 Run a scan

It's easy to run a scan. Select the motor you wish to investigate (select "_none_" if you just want to acquire data without moving anything). Select the range over which you would like to scan, and enter the number of points, with a run time for each point. The scan will examine the region:

currentpos-range/2 <= motor <= currentpos+range/2. Enter a sample name for the run (so you know what the datafile is at a later date) and press the Go button. The run will be saved if you choose to, the run name will be updated when you press go (it will be something like PLP0001120.nx.hdf, note this down in the instrument logbook). When the run has started you can stop and pause it using the STOP and PAUSE buttons that appear.

The graph show the number of counts versus the motor position.

Once the scan finishes FIZZY will attempt to fit a gaussian to the curve, and calculate the centroid. You are then asked if you would like to move to the centroid/gaussian centre or a graph cursor position. Then a prompt appears asking whether you would like to make an offset to the motor position. This is a key feature for sample alignment.

7.1.3 Detector view

Figure (3) illustrates how to visualise the data coming off the instrument (although there isn't any data on there). A PLATYPUS detector pixel has three components: *x_bin, y_bin* and *time_of_flight*. In FIZZY you can only visualise any two of those axes at a time. The displayorder popup allows you to select those axes. Since PLATYPUS normally operates with a single *x_bin* the *time_of_flight:y_bin* is the most useful. The top image shows a 2D plot of the requested axes. For a specular beam this image will usually show a stripe at constant *y_bin*. The two graphs below are the projection of the 2D image along the ordinate (left axis) and abscissa (bottom axis) of the 2D image. For the *time_of_flight:y_bin* option these would normally show a single peak (specular neutrons come out at the same height on the detector) and the time of flight spectrum respectively. You can get the plot to update automatically, and also view a logarithmic plot of the 2D data.

7.1.4 Batch

This tab is used to schedule a large list of experimental commands, such as those required for multiple samples. The commands you enter here are IGOR commands!!!! However, you can send a command to SICS by using the SICS(cmdStr) IGOR command. Create the batch file in a normal text editor (e.g. Notepad), then load them in using the LOAD BATCH FILE button. You select the commands you want to run using the checkboxes. Before running the batch file press the CHECK BATCH FILE button, if there are any problems with the commands you will discover that here. When you are ready press GO. Once the batch is running a STOP and PAUSE button will appear. You could use PAUSE if you need to go into the Platypus enclosure to check your sample.

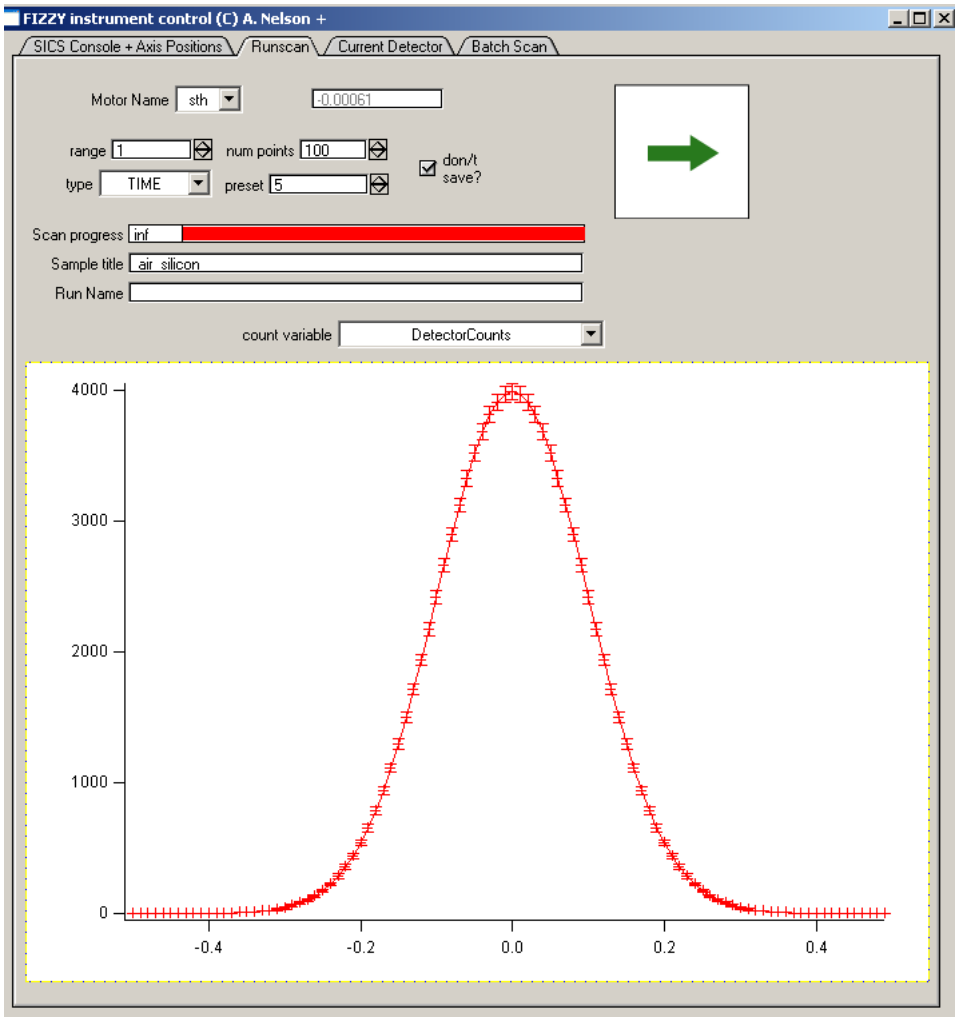


Figure 2: FIZZY scan window

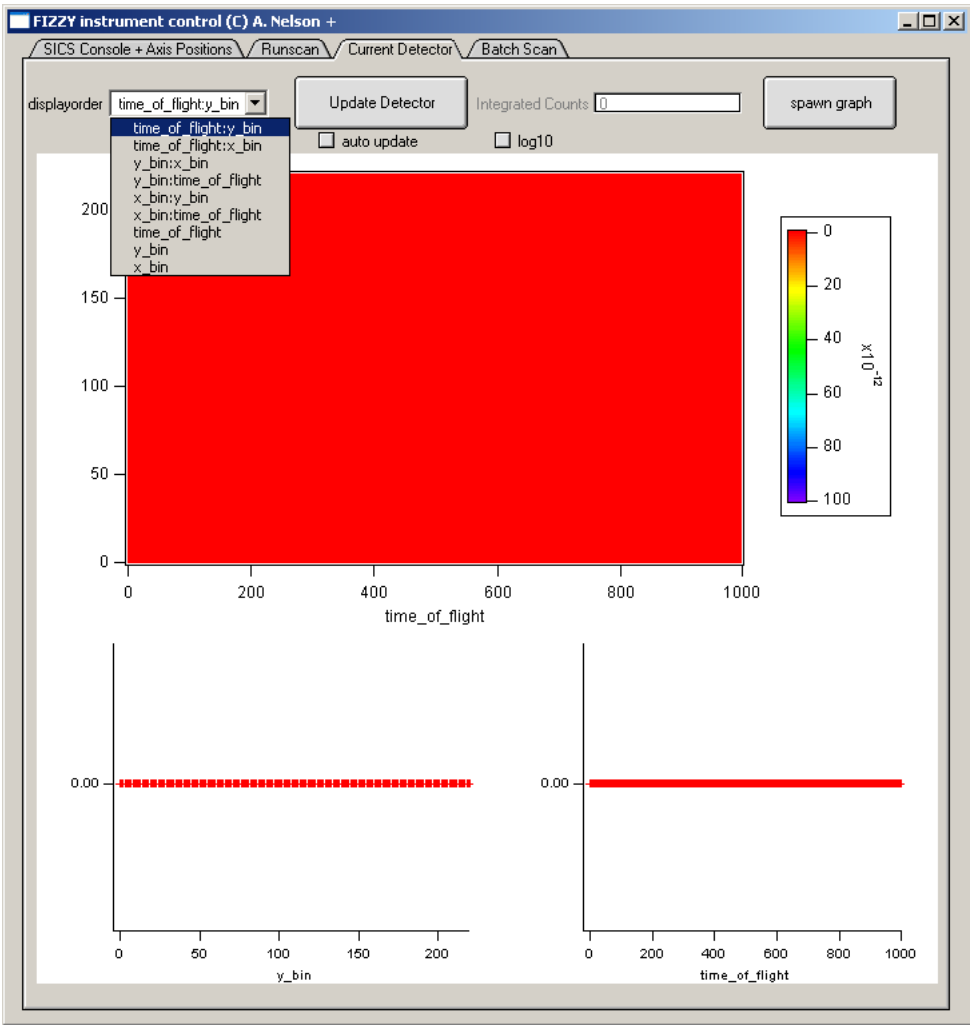


Figure 3: The detector view

As the instrument finishes each line the window will show a "Finished message" in the second column. The active line is indicated by an "executing" message. Even if the batch mode has started you can still change the commands that appear after the currently executing line.

7.1.5 Instrument layout

In this window you can check on several important parameters at once. Such as the current count rate (don't allow the count rate to go above 20000), the positions of the more important motors, the reactor power, cold source temp, shutter status, SICS status, etc. omega is the nominal angle of incidence of the beam onto the sample, two theta is the nominal scattered angle of the reflected beam.

7.2 SICS cmd line and IGOR command line

You can enter SICS commands on the first tab of the FIZZY window (Figure (7.1.1)). However, the SICS terminal and the IGOR command line are different beasts. You cannot *directly* issue SICS commands from the IGOR command line (although it is possible). You definitely can't send IGOR commands from the SICS console window. Figure 6 shows some commands in the IGOR command window. Note the last one, it is a command that is sent to SICS, through the *sics(cmdStr)* command that is within FIZZY.

So what use is either? Well the SICS terminal is fine for when you just want to talk to SICS, but that's only really useful if you know all the SICS commands.

The IGOR command line is useful. One can develop IGOR scripts to control the instrument, then start them running from here. The IGOR command line can be brought to the front using CTRL-J or WINDOWS->COMMAND LINE. Similarly, the batch (7.1.4) view uses IGOR commands. The whole set of IGOR commands is quite large (just look at the IGOR help file), and it's not within the scope of this manual to describe how they all work. However, I've written several convenience commands that help make running the instrument easier. The following commands can all be issued from the IGOR command line, or from the batch mode. Batch mode is obviously useful for running several samples at once. However, not all of the commands will be useful in batch mode, as successive commands are only executed when the previous one has finished.

7.3 Viewing instrument status via the web

The status of Platypus can also be viewed from the web, `http://dav1-platypus.nbi.ansto.gov.au/status.html`. This webpage has some reactor status, detector count rate, acquisition status, recent detector images, motor positions, etc. This is useful if you wish to check on your experiment remotely.

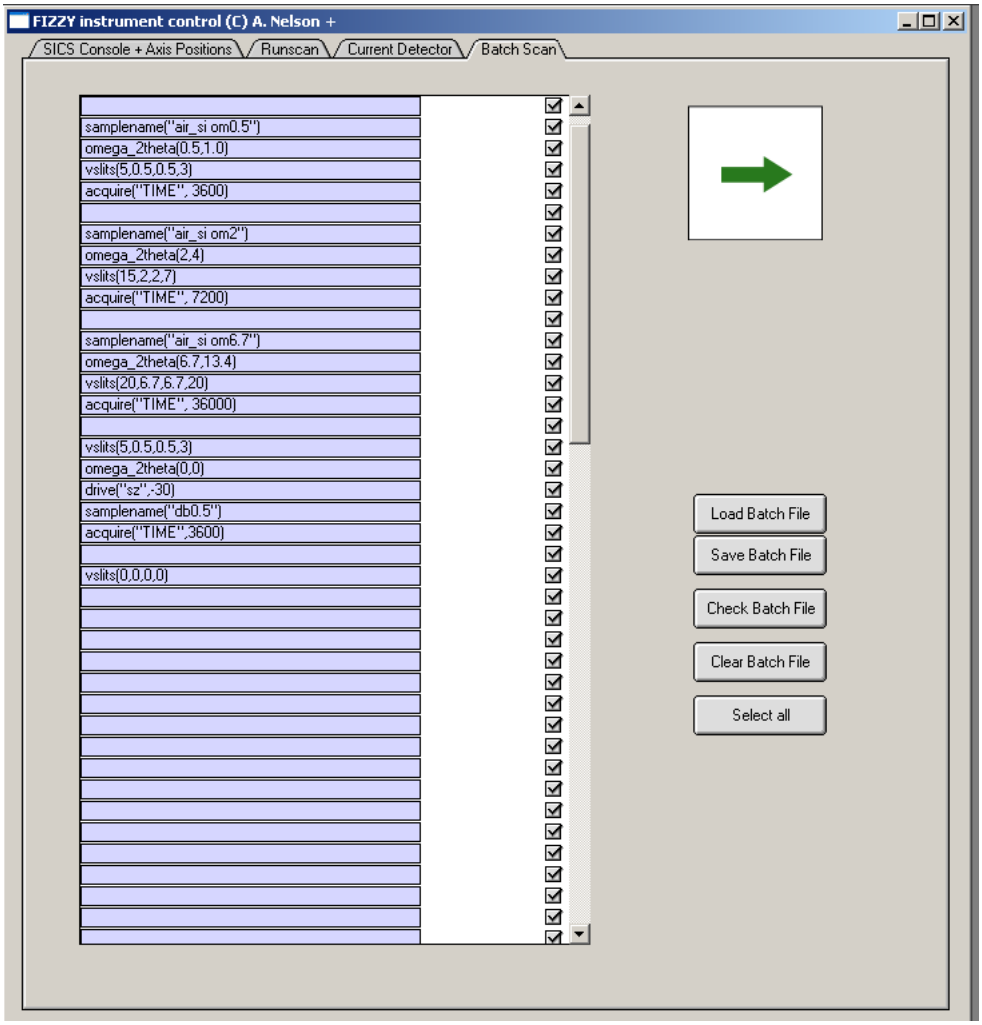


Figure 4: FIZZY batch file window

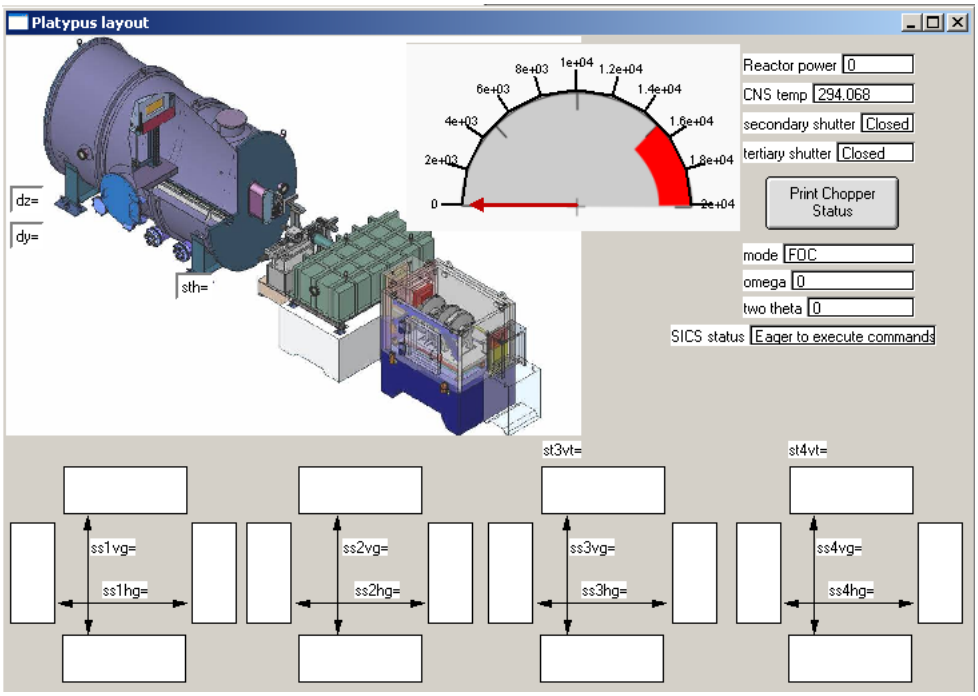


Figure 5: FIZZY instrument layout

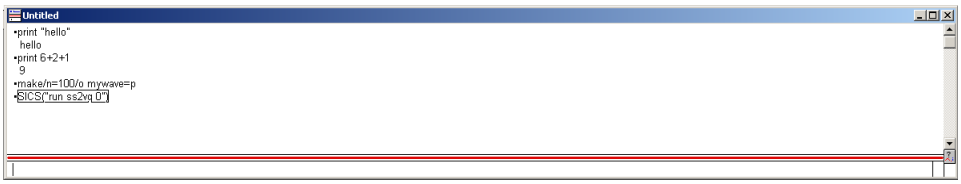


Figure 6: The IGOR command line

7.4 Relevant Igor commands to running Platypus from FIZZY

7.4.1 sics(cmdStr)

this IGOR function send the string *cmdStr* to SICS. For example the command:

```
sics("run ss3vg 1.5")
```

tells SICS to open the slit3 vertical gap to 1.5mm.

7.4.2 attenuate(posVar)

sets the attenuator. posVar is a variable and has one of three values:
-1 removes the attenuator altogether
0 puts the attenuator in the beam, blocking it almost altogether
1 puts the attenuator in the beam in an oscillating fashion. The attenuation factor is around 100. Do not leave the attenuator running for a large amount of time.

```
attenuate(0)
```

7.4.3 drive(motorStr, posVar)

drives motorStr to posVar, if possible. e.g.

```
drive("ss3vg", 1.5)
```

This motion is blocking, you cannot drive anything else whilst this motor is moving. To move several motors at the same time use RUN.

7.4.4 run(motorStr, posVar)

moves motorStr to posVar, if possible. e.g.

```
run("ss3vg", 1.5)
```

Unlike DRIVE this motion is not blocking. You can run several motors at the same time,saving time.

7.4.5 rel(motorStr, posVar)

moves motorStr by an incremement posVar. e.g.

```
rel("sth", 1.5)
```

takes the current position of sth and moves it by 1.5. i.e. it is not an absolute motion.

7.4.6 checkdrive(motorStr, posVar)

checks to see if one can drive a motor to a position without contravening limits. It returns 0 if the motion is possible, or 1 if it is forbidden. e.g.

```
print checkdrive("ss3u", 2)
//can we drive ss3u to 2?
```

7.4.7 getpos(motorStr)

Gets the physical position of a motor, e.g.

```
print getPos("ss3vg")
```

7.4.8 setpos(motorStr, newpos[, oldpos])

setpos changes the offset for a motor axis. It *doesn*'t move a motor.

MOTORSTR is a string containing the name of a motor

NEWPOS is a variable specifying a new position

OPTIONAL

OLDPOS is a variable specifying an old position.

Note that when you use optional parameters you have to specify the parameter name, this isn't necessary for required parameters.

e.g.

```
setpos("ss3u", 0.2)
//sets the current position of ss3u to 0.2
```

e.g.

```
setpos("ss3u", 0.2, oldpos=0.5)
//adjusts the offset such that what was a position of 0.5 for ss3u is now 0.2.
```

7.4.9 vslits(slit1var, slit2var, slit3var, slit4var)

sets all the vertical slit gaps. Slit1, or SS1VG, is set to SLIT1VAR, etc. e.g.

```
vslits(2, 0.6, 0.6, 2)
```

is the same as using:

```
run("ss1vg, 2)
run("ss2vg,0.6)
run("ss3vg,0.6)
run("ss4vg,2)
```

7.4.10 hslits(slit1var, slit2var, slit3var, slit4var)

sets all the horizontal slit gaps (see 7.4.9), i.e. SS1HG, SS2HG, SS3HG, SS4HG.

7.4.11 acquire(presettypeStr, presetVar, samplename, [points])

Acquires data on the detector and saves it in a NeXUS file.
PRESETTYPESTR is either "TIME" or "MONITOR"
PRESET is linked to the presettype (e.g. 5 seconds).
SAMPLENAME is a string containing the name of the sample that is recorded in the datafile
OPTIONAL:
POINTS - variable specifying how many acquisitions do you want to do?
e.g.

```
acquire("TIME", 10, "Si wafer", points=3)
```

Tells IGOR to do 3 runs, each 10 seconds long, with the samplename "Si wafer". Note that when you use the optional parameter, *points*, you have to specify the parameter name, this isn't necessary for required parameters.

7.4.12 acquireStop()

Stops an acquisition started with acquire (7.4.11).

7.4.13 pauseAcquire(pauseOrRestart)

Pauses an acquisition started with acquire (7.4.11)pauseOrRestart is a variable, if set to 1 will pause the run, if set to 0 will restart. e.g.

```
pauseAcquire(1) //pause
pauseAcquire(0) //restart
```

7.4.14 acquireStatus()

returns the status of an acquisition started with acquire (7.4.11).
return values:
0 - not running
1 - running
2 - paused
e.g.

```
print acquireStatus()
```

7.4.15 samplename(sampleNameStr)

Sets the sample name to SAMPLENAMESTR, e.g.

```
samplename("Si wafer")
```

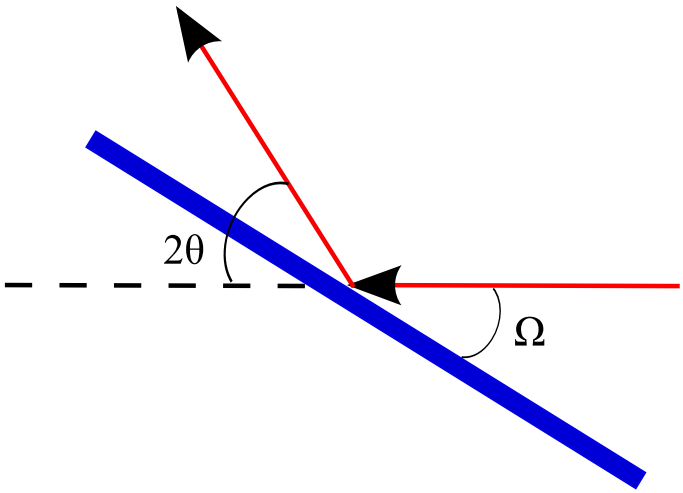


Figure 7: definition of omega and two theta

7.4.16 user(userNameStr)

changes the user name to USERNAMESTR, e.g.

```
user("Fred Bloggs")
```

7.4.17 setexperimentalmode(modeStr)

Used to set the type of experiment you would like to run, which then determines combined motions of motors on the instrument. Most of the modes correspond to the type of collimation element desired. It moves the guide element into place. This SHOULD NOT BE USED without prior instruction from the instrument scientist.

allowable modes:
"MT" - for normal air-solid experiments
"FOC" - for inserting the focussing guide for solid- liquid experiments.
"SB" - for air-liquid experiments with the single bounce deflection mirror
"DB" - for air-liquid experiments with the double bounce deflection mirror.
e.g.

```
setexperimentalmode("FOC")
```

7.4.18 omega_2theta(omegaVar, twothetaVar)

Moves the instrument to a given angle of incidence (omega) and scattered angle (twotheta). For example, for an air-solid experiment it tilts the sample stage to omega degrees, then moves the detector and slit4 to track the reflected beam. This motion is different for each experimental mode you are in (7.4.17).

With an air-solid experiment the sample stage tilts, slit tower 4 moves up, and the detector moves up to track the reflected beam.

For an air-liquid experiment the single bounce mirror rotates, slit tower 3 moves down, the sample stage moves down. Slit tower 4 and the detector are moved, depending on the value of omega and twotheta.

IMPORTANT: For higher angles of incidence the slits (vslits) are normally opened wider. Before going to lower angles of incidence you must make sure you close the slits before returning to low angle of incidence - otherwise you can damage the detector.

e.g.

```
setexperimentalmode("FOC")
omega_2theta(1.2, 2.4)
```

7.4.19 stopAndPrimeDetector(presettypeStr, preset)

Sets the detector up to perform a "freestyle" detector acquisition, i.e. you don't care about saving the data, etc.

e.g.

```
stopAndPrimeDetector("TIME", 5)
```

stops the detector and sets it up to measure for 5secs, but doesn't actually start the detector. Use acquire() - 7.4.11, when you wish to start the detector in a batch run.

7.4.20 stopDetector()

stops the detector acquiring. It speaks directly to the histogram server. This has the effect of stopping any other scans you might be doing.

7.4.21 pauseDetector(pauseOrRestart)

see 7.4.13.

7.4.22 fpx(motorStr, rangeVal, points, [presettype, preset, saveOrNot,sampleTitle,auto]) - scanning of a motor

Performs a scan of a single motor (motorStr), over a range (rangeVal), with several points (points). The scan will normally return MOTORSTR to its starting position when it finishes, unless AUTO is specified.

REQUIRED:

MOTORSTR - the name of the motor of interest

RANGEVAL - the range of interest

POINTS - the number of points in the scan

OPTIONAL:

PRESETTYPE - "TIME" or "MONITOR" (default = "TIME")

PRESET - how much of the presettype do you want to acquire at each point (default = 1).

SAVEORNOT - whether you want to save the scan. Set to 0 to save, or 1 to discard (default = 0).

SAMPLETITLE - the name of the sample.

AUTO - when the scan finishes a curvefit + centroid analysis is performed. If auto=1, then the motor is moved to the peak position. If auto = 0 (default), then the motor returns to the start position.

e.g.

```
fpx("ss3u", 0.3, 11, presettype = "TIME", preset = 10, saveOrNot=1, auto = 1)
//performs a scan of ss3u, over a range of 0.3mm, with 11 points.  Each point is run fo
//when the run finishes ss3u is moved to the centroid of the peak.
```

7.4.23 fpxStop()

Stops an fpx scan (7.4.22)

7.4.24 pausefpx(pauseOrRestart)

pauses an fpx scan. See (7.4.22) and (7.4.13) for more details for what PAUSEORRESTART means.

7.4.25 fpxStatus()

returns the status of the fpx scan.

0 = not running
1 = running
2 = paused

e.g.

```
print fpxStatus()
```

7.4.26 batchScanStop()

Stops a batch scan

7.4.27 batchScanPause(pauseOrRestart)

Pauses a batch scan (see 7.4.13 for more details on pauseOrAcquire).

7.4.28 batchScanStatus()

returns the status of the batch scan.

0 = not running
1 = running
2 = paused

e.g.

```
print batchScanStatus()
```

7.4.29 catalogue(pathStr [,start, stop])

goes through all the NeXUS datafiles in the folder pointed to by pathStr. One can control where the cataloging starts and finishes using the optional start and finish variables. It puts the catalogue into an XML file in the same path and also produces an IGOR table, which can be printed out and put in the log book.

e.g.

```
//catalogues all the NeXUS files on my desktop
catalogue("//Users/andrew/Desktop")
//catalogues all the files on the desktop, starting from PLP0000010.nx.hdf
//to PLP0000100.nx.hdf
catalogue("//Users/andrew/Desktop", start = 10, finish = 99)
```

7.4.30 wait(timeout)

can be used in a batch scan to wait for timeout seconds before proceeding to the next batch point.

7.4.31 stopwait()

issued from the command line to stop the wait (7.4.30)

7.4.32 waitstatus()

issued from the command line, it returns the status of the wait (7.4.30).

7.4.33 labels(labelsStr)

see 7.4.34

7.4.34 goto(labelsStr, repeats)

In conjunction with the LABELS command GOTO can be used in the batch file to create a loop, allowing acquisitions to be repeated, etc.

A typical batch file may look like:

```
labels("a")
acquire("TIME", 3600, "mysamplename")
goto("a", 5)
```

This code sets up the LABEL called "a". The batchfile scheduler will first of all do an acquisition of 3600 seconds. When it gets to the GOTO command it will scan the batchfile from the start for a line containing the text *labels("a").* It is important that there are no extra spaces in the command (e.g. labels("a")), or any comments after. If it finds the line containing *labels("a")* it will run all

the commands following it, until it gets to the GOTO line again. It will repeat that loop REPEATS times.

8 How to align a sample (air-solid, solid-liquid)

This is a quick and *rough* overview of how to perform a sample alignment on a air-solid or solid-liquid sample. If you don't really understand what these steps mean ask an instrument scientist.

1. Close the beam
2. Use the following IGOR commands:
   (a) vslits(0,0,0,0)
   (b) setexperimentalmode("FOC")
   (c) omega_2theta(0,0)
3. Put the sample(s) on the sample stage. Make sure that the interface of interest is 570mm above the main sample stage, use the labjack to do this. Make sure that the beam will be centred on the sample (change the sx motor to achieve this).
4. Make sure that the samples are lower than the beam, so that the beam passes directly to the detector, e.g.
   (a) run("sz", -30)
5. Exit the enclosure
6. Open the beam
7. Use the following IGOR commands:
   (a) vslits(5,0.5,0.5,3).
8. Perform a quick acquire on the detector, for ~100 seconds, 1 point, scanning no motor, don't save the data. Spawn a 2D graph of *time_of_flight:y_bin* from the detector view tab, once enough counts have been acquired. Using the line profile button in the spawned window create a vertical line profile using the *low* time of flight data (~40 timebins wide). This should be a single peak corresponding to the direct beam. Right click on the line profile, select "quick fit" and fit to a gaussian. Note the peak position of the gaussian. Stop the acquisition when you've got this far.
9. Use the following IGOR commands:
   (a) omega_2theta(0.5, 1)
   (b) run("dz", 0)

(c) run("ss4vg", 25)

10. Perform a quick acquire on the detector, for ~1000 seconds, 1 point, scanning no motor, don't save the data. As you are acquiring the data slowly raise the sample back into the beam, step by step (a). As the beam starts to intersect the sample you should notice a second stripe appearing. The lower stripe is the direct beam, the upper stripe should be the reflected beam. The reflected beam will probably be most intense towards the large time bin end. As you raise the sample the direct beam will become less intense (more neutrons are reflected). You will want to stop before the direct beam disappears. Stop the acquisition when you've got this far. If you don't see a second stripe appear then try increasing/decreasing the angle of incidence onto the sample (b). Note that in step 9) we set the angle of incidence to 0.5, so you could investigate the range $0 < \text{sth} < 1$.

    (a) run("sz",-1)

    (b) run("sth", 0.6).

11. Work out the pixel displacement between the direct and reflected beams. To do this perform a quick acquire on the detector, for ~100 seconds, 1 point, scanning no motor, don't save the data. Spawn a 2D graph of *time_of_flight:y_bin* from the detector view tab, once enough counts have been acquired. Using the line profile button in the spawned window create a vertical line profile, with enough timebins to get reasonable statistics (~100 timebins wide). There should be two peaks, corresponding to the direct beam (low y pixel) and the reflected beam (higher y pixel). Try and maximise the reflected beam signal by getting the line profile from the high time bin end of the spectrum. Right click on the line profile, select "quick fit" and fit to a gaussian. Note the peak position of the reflected beam. Stop the acquisition when you've got this far.

12. Now we need to work out the reflected angle. Steps 8) and 11) allow us to work out the deflection, in detector pixels, between the direct beam and the reflected beam. Because we know the sample-detector distance (dy) and how many mm a detector pixel represents, we can work out two_theta (Figure (7)). To do this type:

    (a) wott(reflectedPixel, directPixel).

    (b) The actual angle of incidence is printed out in the IGOR command window.

$$\Omega = 0.5 \times \arctan\left(\frac{(ref_{px} - db_{px}) \times 1.168}{dy}\right)$$

13. Now we have to enter an offset for sth, because it was probably slightly wrong. So type:

(a) sics("sth setpos 0.5 newpos")

(b) where newpos was the angle that you calculated in step 12).

14. Now that we have aligned the angle of incidence somewhat we need to align the sample height more accurately, with a rocking curve scan. First of all we will close down the slit 4 slits, to give more accuracy

    (a) run("ss4vg",1)

    (b) then run a scan for "sz", with a range of ~1->2mm, with ~21 points. You should see a peak in this scan. If you don't increase the range & use more points.

    (c) When the scan finishes move to the peak centre or centroid (your choice), and when FIZZY asks you if you want to set an offset, say yes, and set the new position to be 0.

15. At this point we should have a better alignment, but because we changed the sample height, and sample tilt, either one may have changed slightly. Therefore repeat steps 9) to 14), several times. You can stop when step 14) has a peak at sz=0, and when step 12) prints out an angle of incidence=0.5 (or whatever value you chose in step 9). Note that in step 10) you probably won't need to change the "sz" position after the first time, just work out the reflected beam position.

16. For good measure perform a "sphi" scan (sphi is the roll of the sample). This scan is quite broad, use a range of ~3deg with 31 steps. The scan should have a peak centred at 0. Enter an offset to make this so if it isn't the case. You will need to repeat steps 9) to 15) after this.

17. Your sample is aligned, perform a measurement.

9 Access to your data

You have access to your data whilst you are at ANSTO. You can take your data in many ways, such as emailing it to yourself, burning a CD, memory stick, etc. However, data is also available for downloading via a webpage: `http://dav1-platypus.nbi.ansto.gov.au/cgi-bin/getData.cgi`. This webpage requires authentication. If you would like a login for this method please see an instrument scientist. You can also download data using SLIM, which is the PLATYPUS data reduction program. This uses the same web based method for downloading the data, so you will need a login for this method as well. The SLIM is the easiest way. The only thing you need to remember are the run numbers for your data files.

10 SLIM - reducing data in IGOR

SLIM is the data viewer and reduction module for PLATYPUS. It also works in IGOR. This means that you can acquire data (FIZZY), reduce it (SLIM) and analyse it (MOTOFIT) in a fluid, logical way. SLIM can reduce data in real time (i.e. as the data comes off the instrument), from any computer that has web access, anywhere in the world.

SLIM is part of more recent Motofit distributions, which can either be obtained from `http://motofit.sourceforge.net` or `http://dav1-platypus.nbi.ansto.gov.au` (this is the more up to date version).

The first thing to do is set the directory where your data is, or where you would like it to be saved. This can be done by pressing the change button at the top right hand corner of the SLIM window.

10.1 Data format

Platypus data is saved in NeXUS format, inside an HDF file, which is binary, not ASCII. One can download HDFview (`http://www.hdfgroup.org/hdf-java-html/hdfview/`) to examine the data within the file, or there is an HDF browser within IGOR.

10.2 Downloading data if you don't already have it

Simply press the *Download Platypus data button*. You are then prompted for a login password (see an instrument scientist for a login). If the login succeeds you enter the run numbers you require, and SLIM will download, unzip and place the required files in the data directory you specified.

10.3 Controlling how SLIM performs the reduction (background, rebinning, etc)

Check the *background sub* checkbox to perform background subtractions during the reduction.

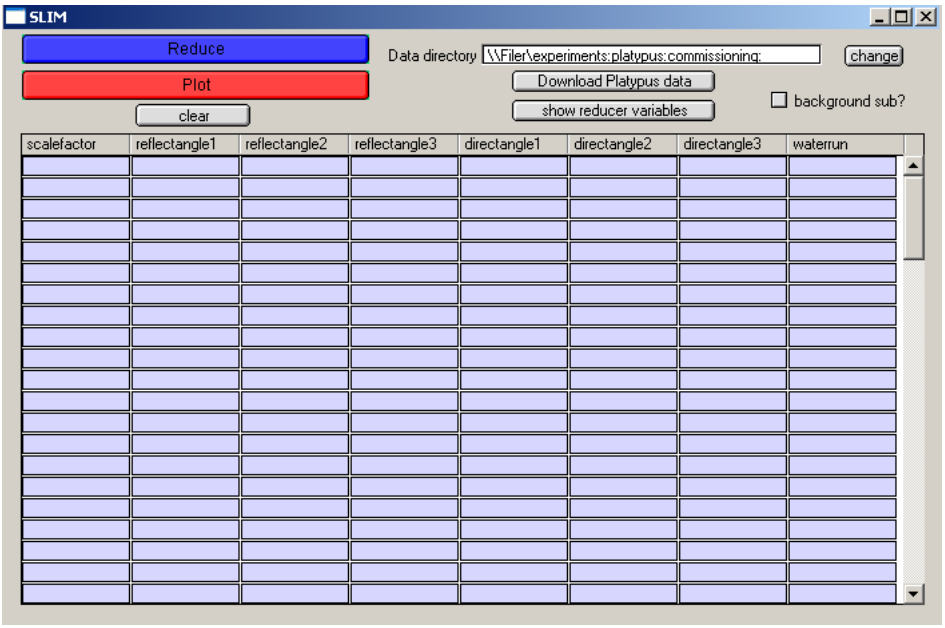


Figure 8: The main SLIM window

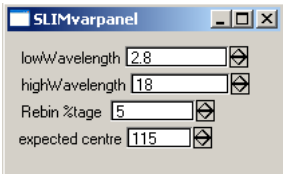


Figure 9: Controlling the parameters of a data reduction

If you wish to control other options press the *show reducer variables* button (Figure (9)). Here you can change the wavelength region that you are interested in, how coarsely you want the data to be rebinned, and at what y pixel you expect the specular beam to appear.

10.4 Visualising Platypus files (wavelength spectra, etc)

Click on *Plot* in the main SLIM window (Figure (10)). Select the files you wish to open and press *Get Files*. If you select the *.hdf* file type you will view the raw data, with wavelength spectra, etc. If you select the *.xml* file type you will visualise reduced data. If you select the *.itx* file type you will visualise scans resulting from alignment scans (i.e. counts vs. motor position). If one of the run numbers is the current sample then pressing the *refresh* button will cause that spectrum to be updated in the plot.

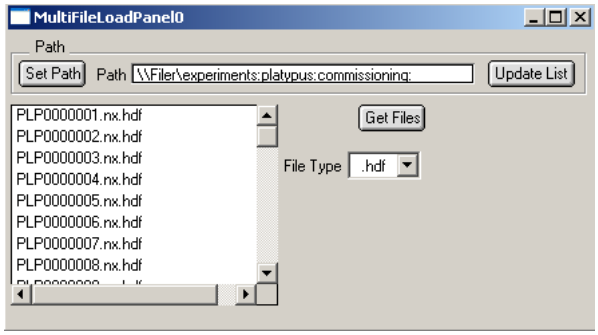


Figure 10: Loading many files at once.

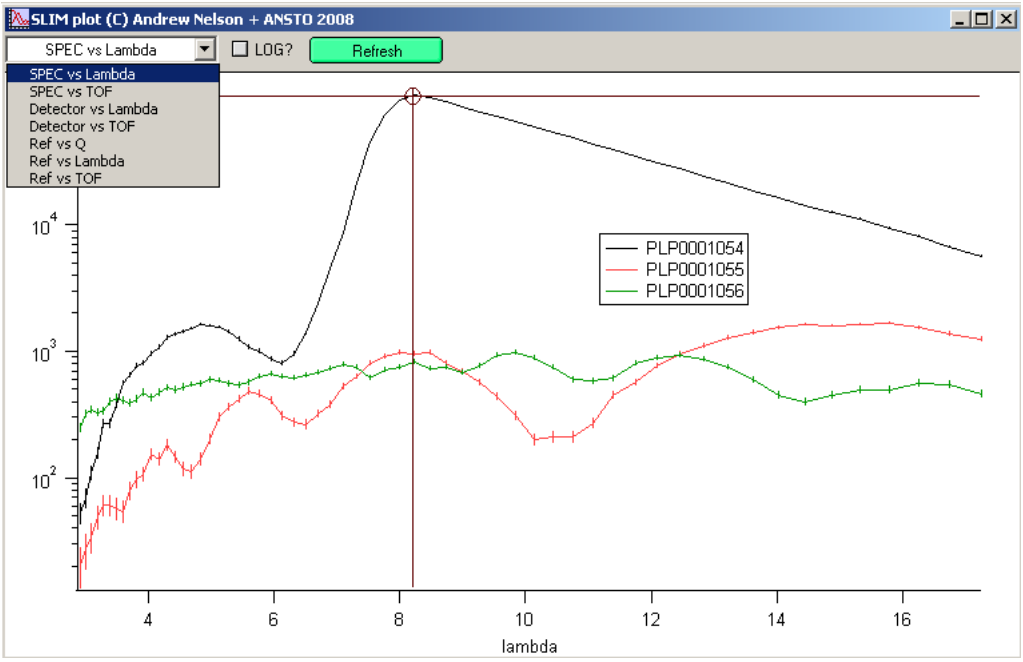


Figure 11: Using SLIM to view wavelength spectra

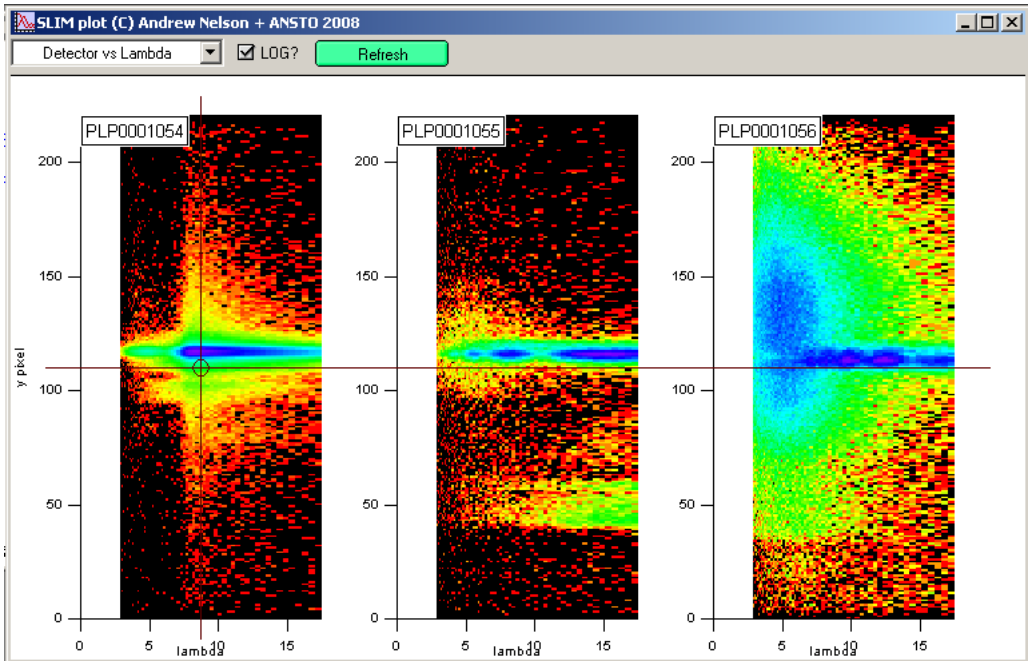


Figure 12: Using SLIM to view detector patterns, showing data from 3 different angles of incidence.

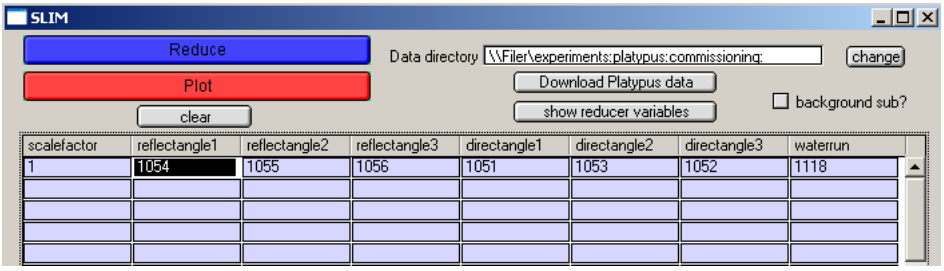


Figure 13: reducing

10.5 Reducing data

To reduce data set up the options for the reduction (Figure (9)). Then enter the scale factor, and the run numbers for the reflected and direct beam measurements. The scalefactor is a number that the final reflectivity curve is divided by, to make the reflectivity of the critical edge equal to 1. If you have a detector efficiency scan, a *waterrun*, enter that as well. You can reduce all your files at the same time if you wish.

Sometimes the automated specular beam finding algorithm cannot find the specular beam. If this happens SLIM will display a graph, asking you to help define the specular region. This is done by asking you to specify a region of interest. As you change the ROI SLIM will integrate over the time bins present, and try to fit a specular peak. Once you are happy that SLIM has found the specular beam press continue. *Hint*: the ROI should be located at the high time bin end, as this region usually has less background than low wavelength data.

Once you have reduced your data you can analyse them using Motofit, or you can view the files in SLIM.

10.6 Data format from the reduction

There are several files produced during the data reduction process. They are of two file type *.dat* and *.xml*. Both these can be opened by recent versions of Motofit (*xml* is more recent). The XML file is a way of including more metadata (such as sample name) in the reduced file, it is also extensible to offspecular data files.

Each individual reflection run has a corresponding *xml* and *dat* file. In addition, if several runs are reduced together there is a combined file which contains a single dataset that consists of all the reflectivity data spliced together. The example shown in Figure (13) will produce the following files as output:

Individual runs: PLP1054.dat, PLP1055.dat, PLP1056.dat, PLP1054.xml, PLP1055.xml, PLP1056.xml

Combined runs: c_plp1054.dat, c_plp1054.xml

There are 4 arrays/columns of data in a specular reflectivity file which are of interest. Each of these arrays has an equivalent number of points and correspond

to (listed in order that they appear in the *.dat* file):

Q

The Q value for a reflectivity point, units $Å^{-1}$.

Ref

The reflectivity value, which is unitless.

dRef

The experimental uncertainty in a reflectivity value, expressed as a standard deviation.

dQ

The uncertainty in the given Q value, expressed in FWHM terms, units $Å^{-1}$. This value represents the resolution with which the experiment was done, i.e. a combination of the angular resolution of the beam (which is dependent on the slits), and the wavelength resolution of the measurement. Full details on this calculation may be obtained from two papers:

1. *A.A. van Well, H. Fredrikze*, "On the resolution and intensity of a time-of-flight neutron reflectometer", *Physica B*, **357**(2005) 204-207.
2. *V-O. de Haan, J. de Blois, P. van der Ende, H. Frederikze, A. van der Graaf, M.N. Schipper, A.A. van Well, J. van der Zanden*, "ROG, the neutron reflectometer at IRI, Delft", *Nuclear Instruments and Methods in Physics Research A*, **362**(1995), 434-453.