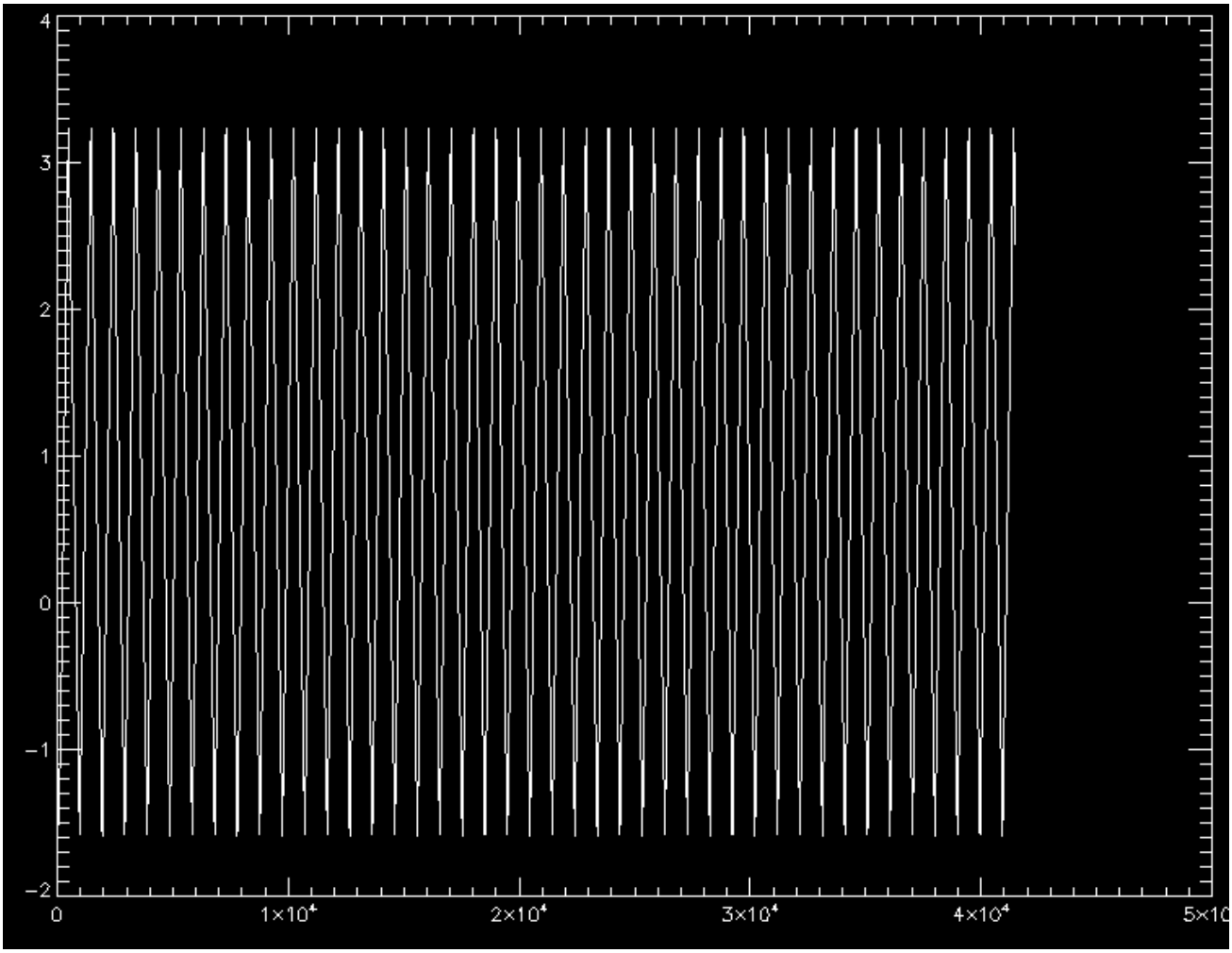
AS MEX H5 processing in GeoPIXE

Debug contents of MEX hdf5 file

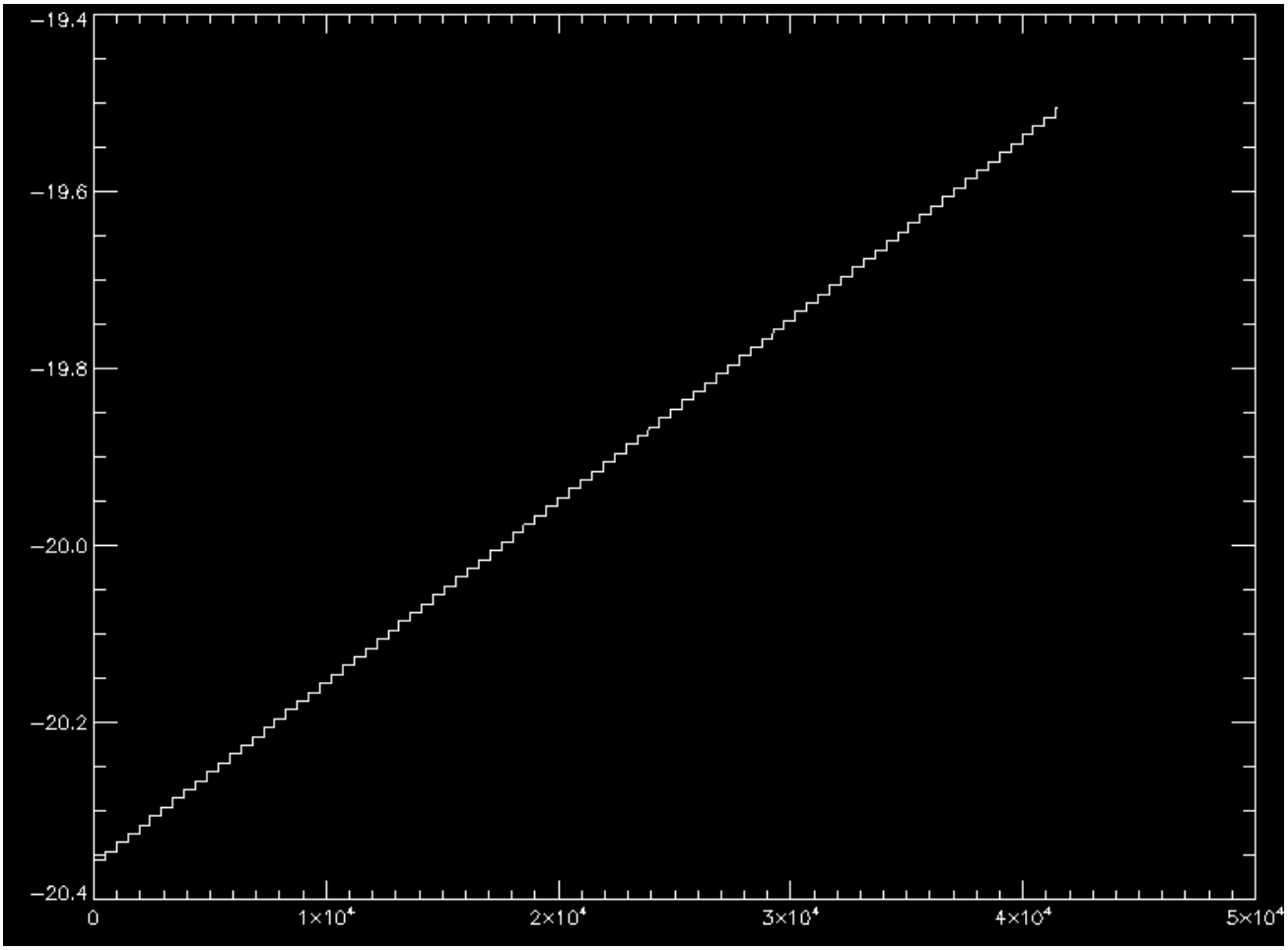
Notes about what we find …

1. There is no metadata for ‘nx’, ‘ny’, nominal dwell, beam energy and detector energy cal.
2. There is an ‘abs_x’, which is absolute ‘x’ position (float) and an ‘x’, which is relative ‘x’ (but first item seems corrupted). Need to determine pixel x,y from these and an ‘effective nx,ny?
3. Records ‘x_ts’ are dwell time (ns?), which seem all about equal, same for ‘y_ts’, ‘i0_ts’, etc.
4. ‘position’ is a sequence number index (starts at 0).
5. ‘spectrum’ is the spectra from 4 detectors over pixel count/sequence index (1,41496,4,4096). Not sure what the first redundant index is about.
6. ‘i0’ is a flux, ‘i1’ is a flux, ‘i0_ts’, ‘i1_ts’ are (same) dwell.
7. ‘dcm_energy_ev’, energy (eV) by sequence. ‘mex1_dcm_bragg’ by Bragg angle.
8. ‘Accelerator_ring_current’, ring mA versus sequence.

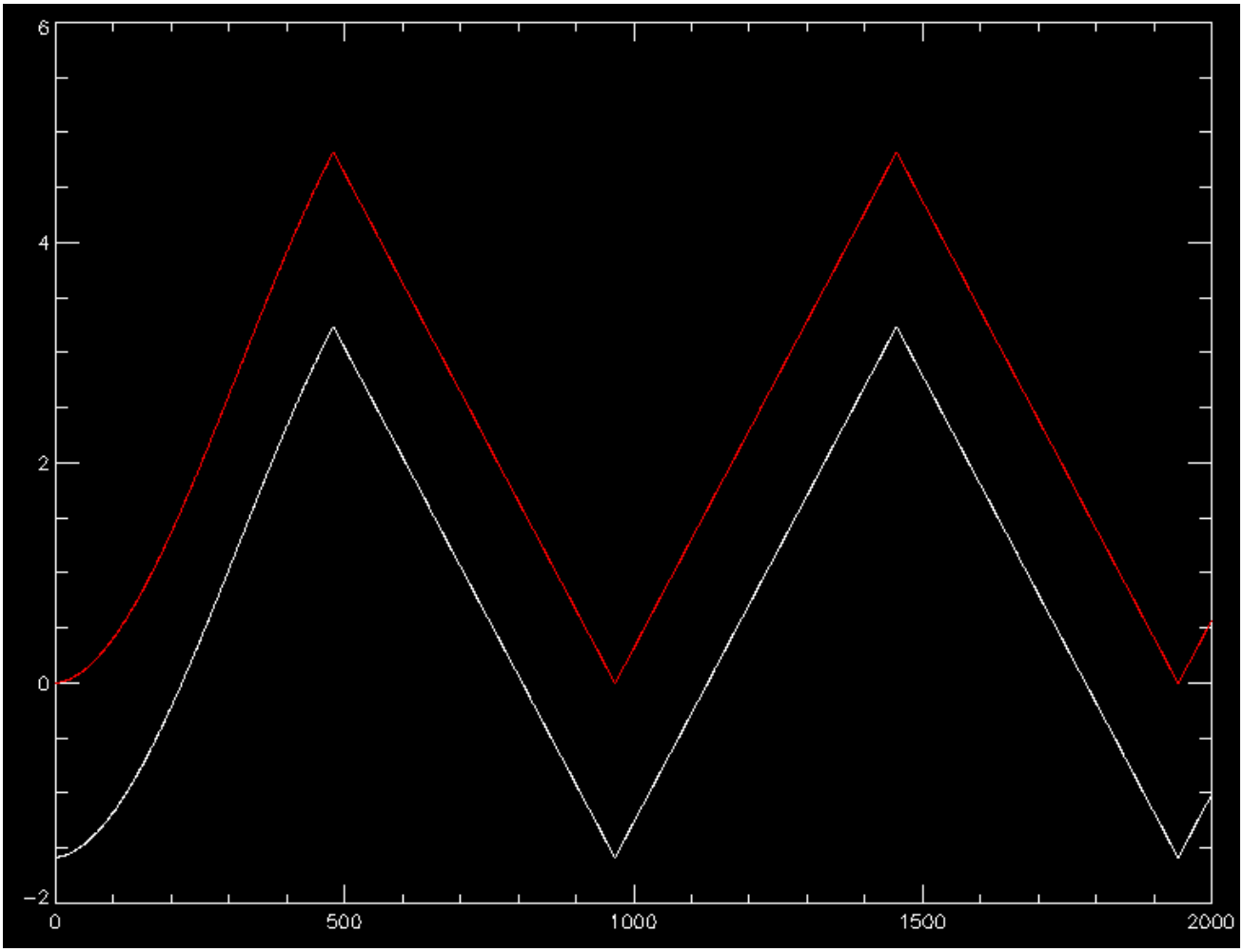
Dataset



Abs_x



Abs_y



Abs_x[0:2000] (white) versus x[0:2000] (red) – red is just starting at zero

Seems to have about 43 swings in 'abs_x'. Does not step evenly. Used this code to get an average step size (assumed equal in x and y) … Total number of sequence steps (41496) is less than the inferred nx (488) * ny (85).

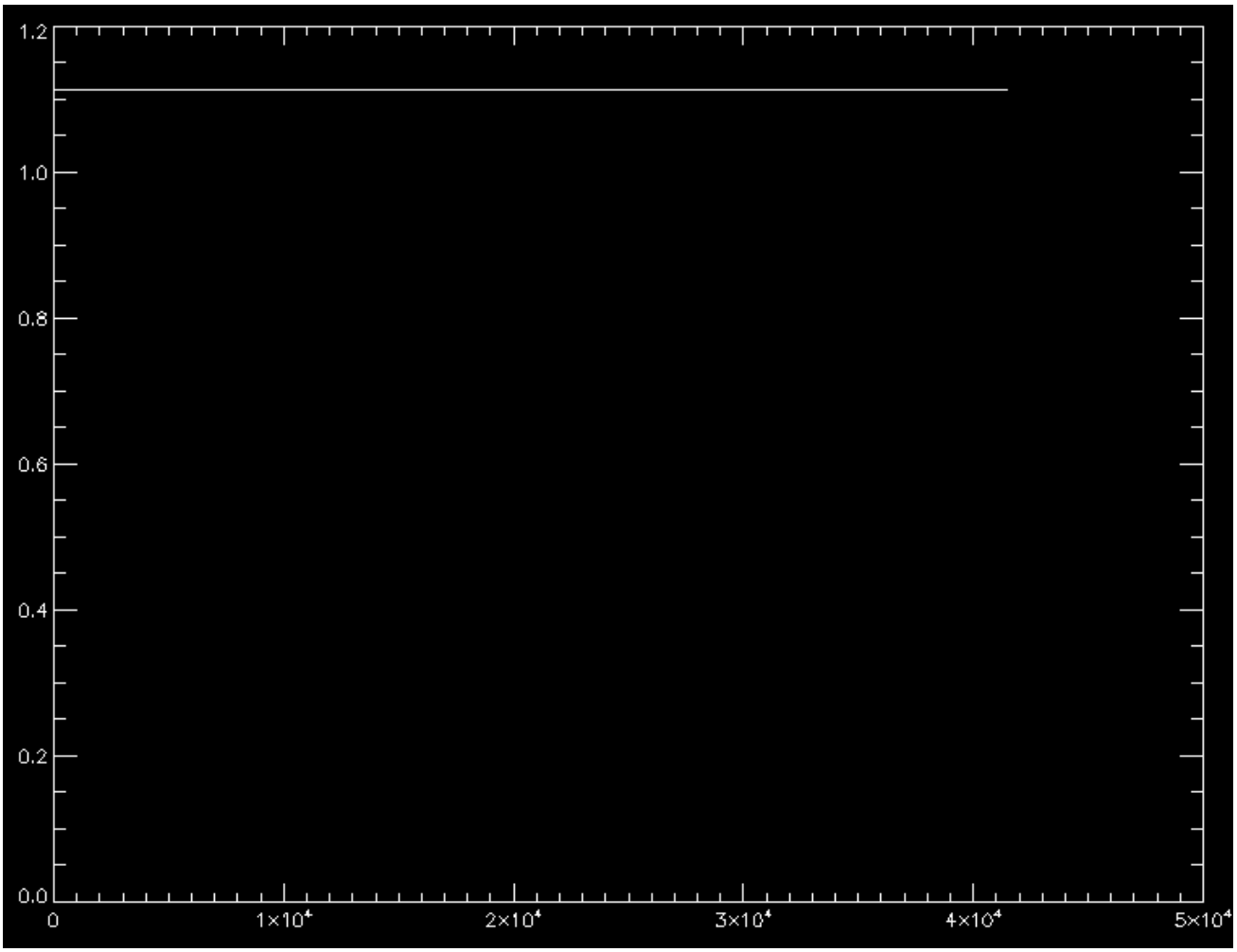
```
step_x = x - shift(x,1)
step_x[0] = step_x[1]                                   ; fix a bug in first value?
step_x = mean( median( abs(step_x[nseq/10:*]),5))       ; best shot at ave. step in X
step_y = step_x                                         ; assume step in Y is the same as X

pixel_x = round( (x - min(x)) / step_x )                ; pixel addresses
pixel_y = round( (y - min(y)) / step_y )
min_x = min( pixel_x)
min_y = min( pixel_y)
nx = max(pixel_x) + 1                                   ; effective nx, ny
ny = max(pixel_y) + 1
```

Since we work out the pixel x,y from the abs_x, abs_y for each sequence step, easier was to read all sequence steps for one detector channel and assign x,y this way … Then step through detector channels (4). Progress was then simply by detector channel.

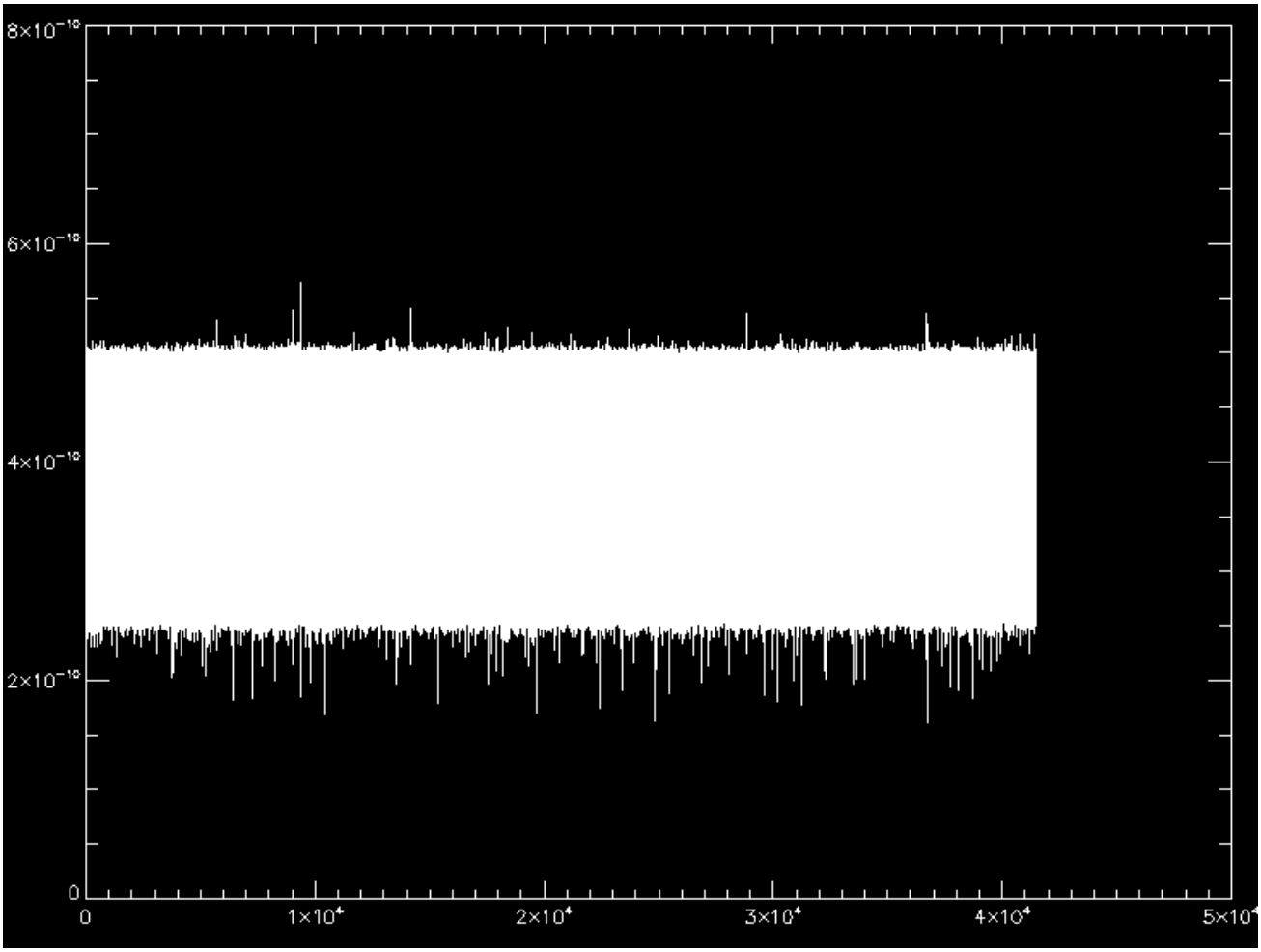
```
for j=0L,nseq-1 do begin
        pnc_hdf_x1[*,j] = x[j]
```

```
        pnc_hdf_y1[*,j] = y[j]
        pnc_hdf_e[*,j] = ramp
endfor
```

Looking at the time-stamp ‘x_ts’ values, we see this … and the differences should be dwell …



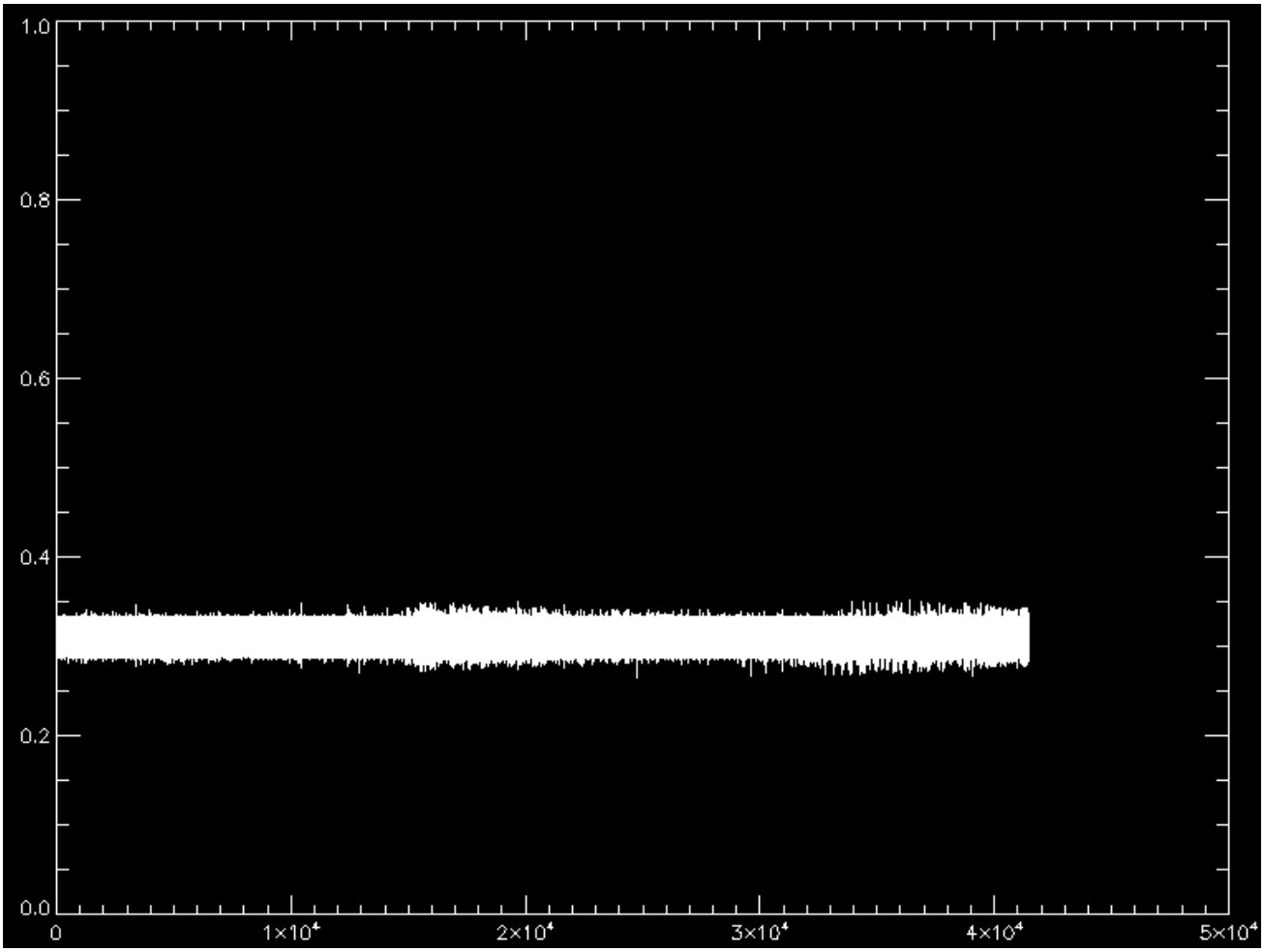
X_ts



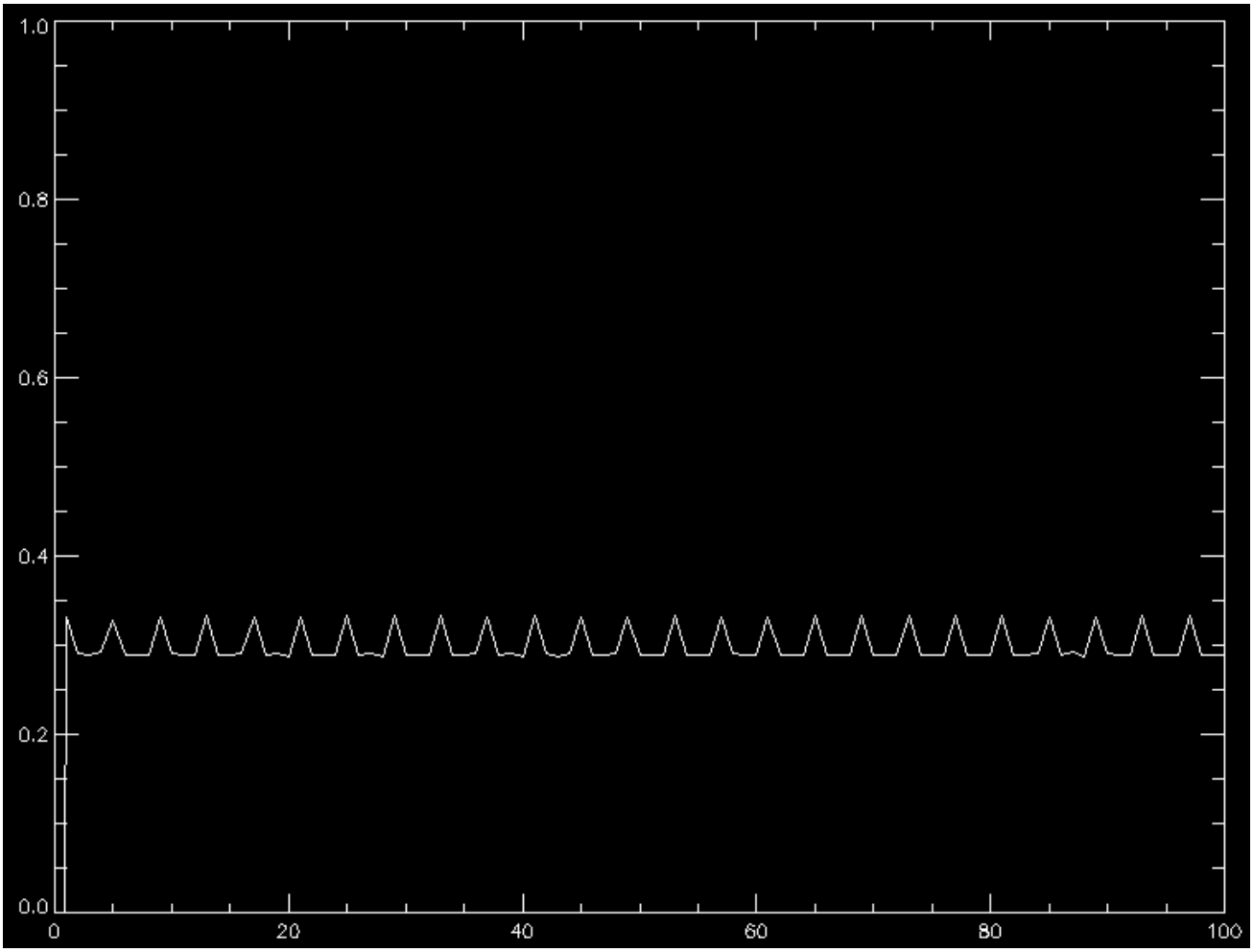
x_ts - **shift**(x_ts,**1**), yrange=[**0**,0.**7**] – mean around 0.4 seconds?

First 100 values

The dwell (as differences between time-stamps) averages about 0.39 seconds. But jumps over about 0.2 seconds. This is probably not real, some artefact of the logging software?



I0_ts - **shift**(i0_ts,**1**)



I0_ts - **shift**(i0_ts,**1**), first 100 values

These time-stamps vary less. But they average around 0.3 seconds with jumps of 0.05 s.

Overview of pipeline

The two main methods for access to the data are (i) reading "header" metadata to extract parameters like the pixel size of an image (nx,ny), energy, sample name, etc. and also pull out data for the stage encoders, dwell (exposure time) as a function of sequence number, and (ii) reading the HDF file to get spectral data and associating that with pixel address x,y to form vectors of photon events (E,x,y, …).

More detail

The methods in the device objects generally include (i) a few that handle "options", (ii) 'get_header_info', which calls the 'read_as_mex_h5_header' routine, (iii) 'update_header_info', which copies the read header info into the 'self' structure of the object (can avoid unnecessary re-reading of long header files), (iv) 'flux_scan', which is usually used to look for the PV names and settings for flux (it also calls 'read_as_mex_h5_header'), (v) 'read_setup', setup photon event vectors for each spectral data read from the HDF file, (vi) 'read_buffer' read next buffer of data from the HDF file, (vii) a few that handle "import" approaches and (viii) 'init', which is the object initialize method, which is only called when a new instance of the object is created.

"Option" GUI elements

"Options" are GUI elements that can appear across GeoPIXE (e.g. "device" parameters in the *Import spectra* and *Sort EVT* windows) for control of internal "device" parameters. These are not really used at this time for

this P06 NXS device object. They are place holders for later features, and currently just save a default 'version' value.

Import spectra

"Import" methods provide parameters to use for (i) the *Import Spectra* call (Menu: "Import→Spectra" in the *spectrum display* window), and (ii) custom local spectral data reads. In this device, we are only using the first, which is handled via the *Import Spectra* approach (see routines "import_select" and "spectrum_load" called from *spectrum display* window), which scans all raw spectra/event data in Xspress3 NXS files to accumulate spectra for 'E' and the projection of all events onto X,Y axes. The definition for this is …

```
opt_39 = define(/import)                                ; MEX new HDF5 file read as a list-mode
        opt_39.name =       'as_mex_h5_evt'             ; unique name of import
        opt_39.title =      'Extract E,X,Y from MEX map HDF5 file as list-mode'
        opt_39.in_ext =     '.hdf5'                     ; input file extension
        opt_39.request =    'Select MEX HDF file to scan for all spectra, X,Y'
        opt_39.preview =    0                           ; allow spectrum preview
        opt_39.raw =        1                           ; flags use of separate Raw data path
        opt_39.multifile =  0                           ; denotes multi-file data series
        opt_39.separate =   ''                          ; char between file and run #
        opt_39.spec_evt =   1                           ; uses call to 'spec_evt' to extract events
        opt_39.use_IC =     1                           ; pop-up the 'flux_select' PV selection panel
        opt_39.IC_mode =    1                           ; default to using PV for IC
```

This tells GeoPIXE that HDF data is found in one file (multifile=0) and a file extension of "hdf5" (in_ext=".hdf5"). 'Title' is for the *Import Spectra* popup title, 'request' is a title for the *File-Requester* popup, 'raw' flags a separate path for raw data and analyzed data, 'use_IC' flags using a popup 'flux_select' to choose the PV to use for flux values, 'IC_mode'=1 means a default mode of using a PV selection for flux with gain settings.

Header read ("read_as_mex_h5_header.pro")

Reading "header" metadata is needed to extract parameters like the effective pixel size of an image (nx,ny), energy, sample name, etc. and also pull out data for dwell and flux as a function of sequence number.

It reads 'x' and 'y' data per sequence number. This gives pixel position as a function of sequence number. From these we calculate pixel address "pixel_x, pixel_y" versus sequence number using these lines.

```
step_x = x - shift(x,1)
step_x[0] = step_x[1]
step_x = mean( median( abs(step_x[nxy/10:*]),5))
step_y = step_x

pixel_x = round( (x - min(x)) / step_x )
pixel_y = round( (y - min(y)) / step_y )
min_x = min( pixel_x)
min_y = min( pixel_y)
nx = max(pixel_x) + 1
ny = max(pixel_y) + 1
```

We also pull out dwell time per pixel from "i0_ts" time-stamp differences as a function of sequence number. We then populate a 2D array of dwell time ("maia_dwell") making use of the pixel address and sequence

number. The array “maia_dwell” is stored crudely in a common block for now, so it can be accessed when scanning data later in ‘read_setup’ and ‘read_buffer’.

```
rec_id = H5D_OPEN(file_id, 'x_ts')
dw = H5D_read(rec_id)
t = dw - shift(dw,1)
t[0] = t[1]                                          ; fix wrap
dwell_array = t                                      ; s
maia_dwell = fltarr(nx,ny)
maia_dwell[pixel_x,pixel_y] = dwell_array * 1000.    ; ms dwell
H5D_close, rec_id
```

A common/representative dwell “common_dwell” is also estimated.

Processing spectra/image data

Reading spectral data is done in the methods “read_setup” and “read_buffer”. ‘read_setup’ is called for each new HDF5 file and then ‘read_buffer’ is called, normally in a loop until all data is read. In this device object, this looping is used access all detector channel data found in the HDF file (indexed using ‘i_petra_channel’).

On entry into ‘read_setup’, the HDF5 data file is already open and ‘fstat’ is used to get its filename so we can open that using HDF5 routine ‘H5F_OPEN’. We first read sequence number vector for this file and determine the pixel address vectors ‘x,y’ for this sequence number vector (‘sequence’) using the saved ‘pixel_x, pixel_y’. Note that HDF5 sequence number start at 0.

‘read_setup’ then determines how many detectors are present from ‘dims’and sets ‘n_petra_channel’. It then creates some arrays that will be used in ‘read_buffer’ to assemble vectors for the events stream for pixel address (x,y), “station” or channel number (ste), and photon energy (e).

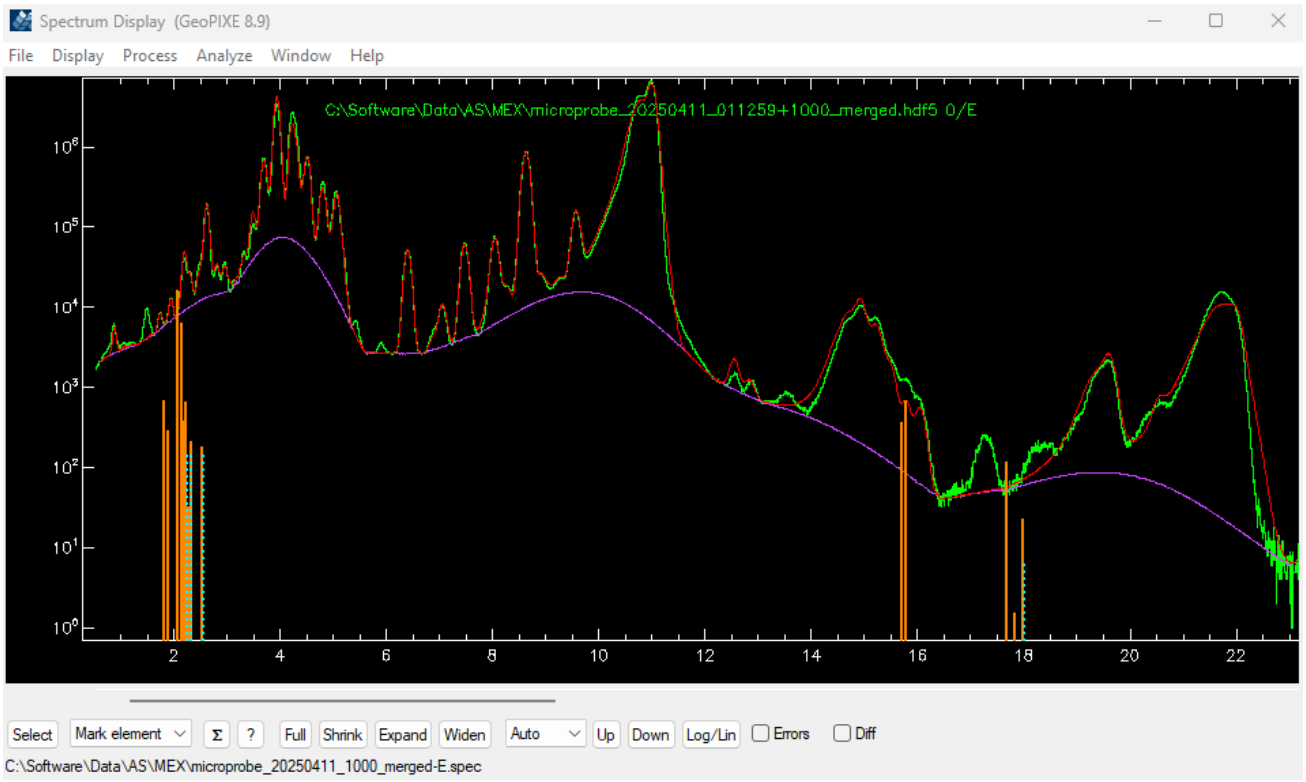
‘read_setup’ then looks for the *IC PV* to find the vector of flux values as a function of sequence number. The chosen PV is given by ‘flux_ic.pv’. Its gain units are set in ‘flux_ic.val’ and ‘flux_ic.unit’, which for now are set to unity (1 nA/V, i.e. nsls_flux_scale=1.0). It then sets the corresponding pixel flux values in an array ‘flux’ across the image. Here we distinguish between imaging mode, where we need to maintain a 2D flux array and spectrum mode, where we just accumulate a total flux.

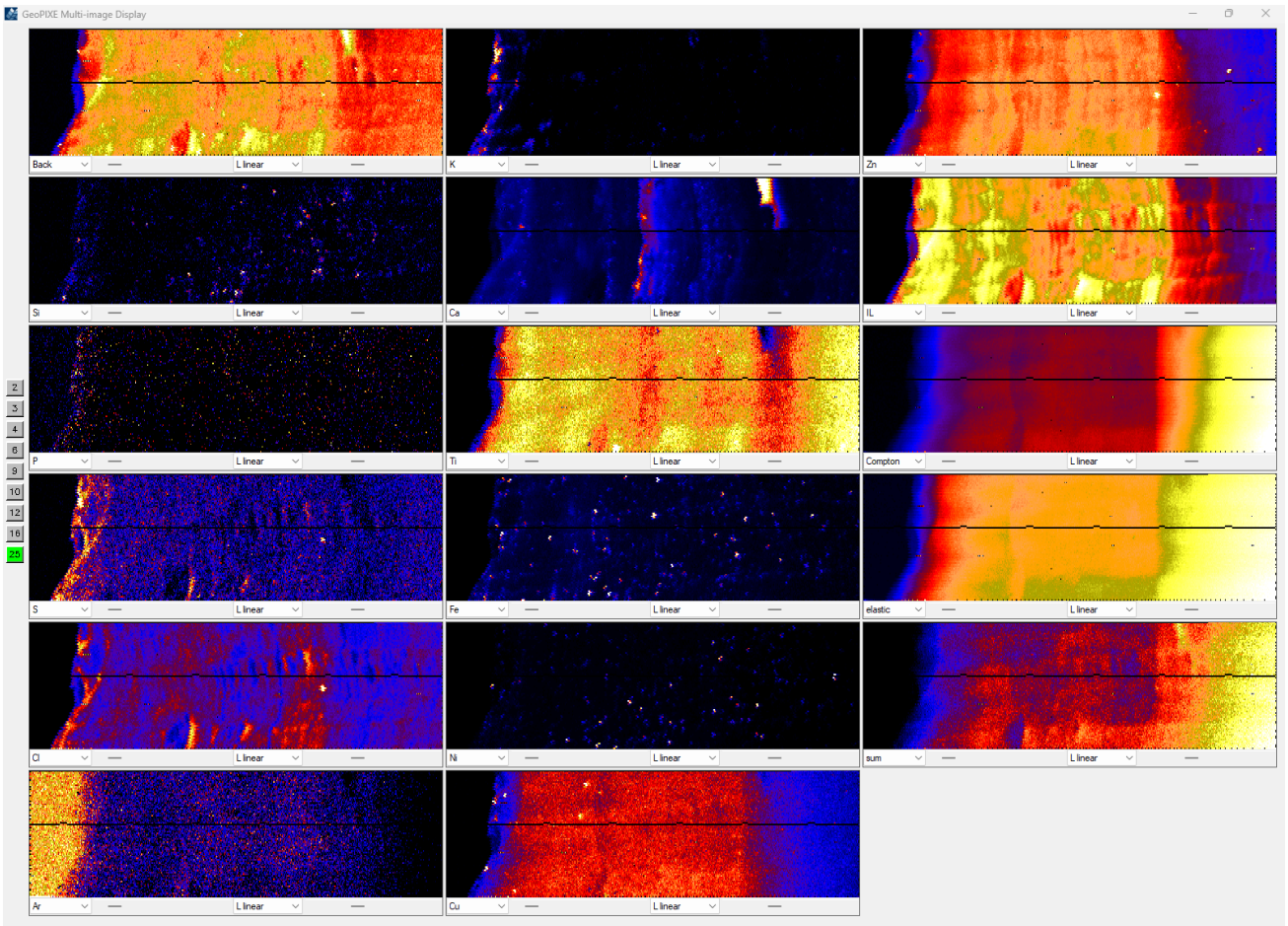
```
nsls_flux_scale = flux_ic.val * flux_ic.unit
i0 = H5D_read(rec_id)
flux[x,y] = nsls_flux_scale * i0
```

‘read_buffer’ reads the spectra data from the HDF5 “spectra” for each channel ‘i_petra_channel’. It then builds vectors of photon events ‘e,ste,x1,y1,multiple for return to GeoPIXE (e.g. to ‘da_evt’ or ‘spec_evt’ routines). The vector ‘multiple’ makes it simple to convert a spectrum into a pseudo photon event stream by setting ‘multiple’ to the counts in each bin of the spectrum histogram.

Test data

Data file: microprobe_20250411_011259+1000_merged.hdf5 (note “+” in filename causes problems). Energy 11 keV, major peaks due to IL, Fe, Ni, Cu, Zn, minor Cl, Ca, Ti. Fitted using a garnet skarn setup for ME3 at XFM, for 11 keV.





Black (missing) pixels across images may be an artefact of determining pixel address from supplied x,y positions. Seems to be some “jaggies” shearing, evidence for back-lash or logging lags.