A Python-based approach to interactive x-ray and neutron data analysis

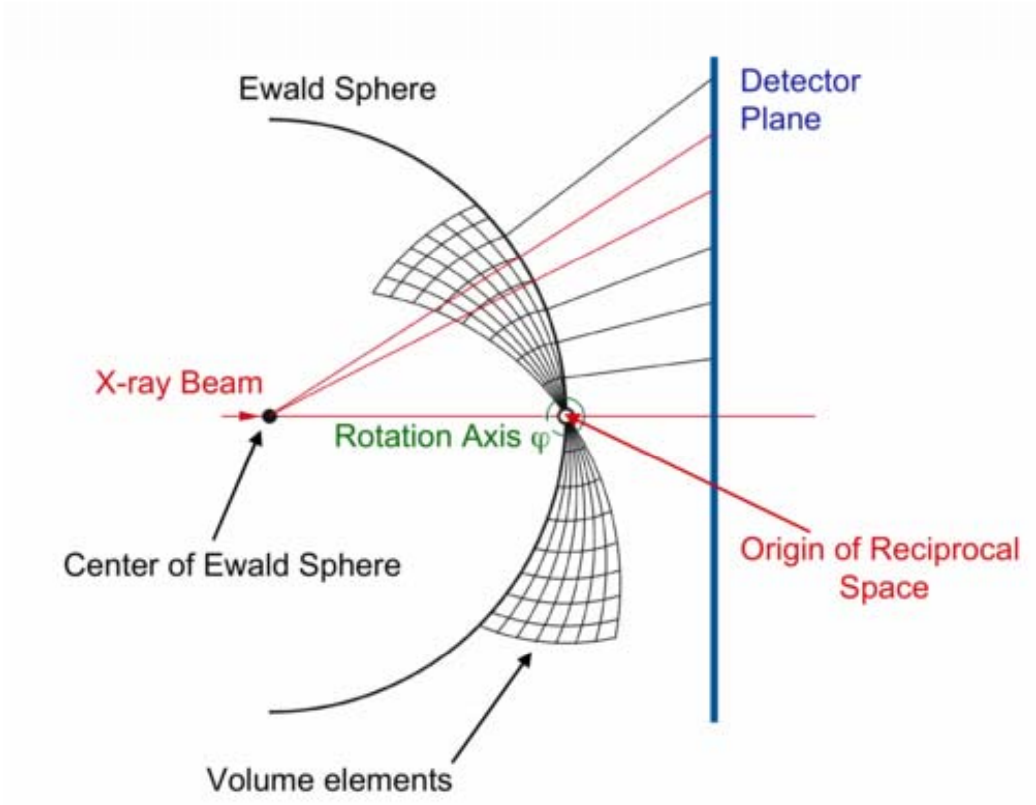
How an ex-programming scientist recovered his mojo

Ray Osborn, Stephan Rosenkranz, John-Paul Castellan
Materials Science Division, Argonne National Laboratory, Argonne, IL

Boyana Norris, Jason Sarich, Dmitry Karpeev
Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL

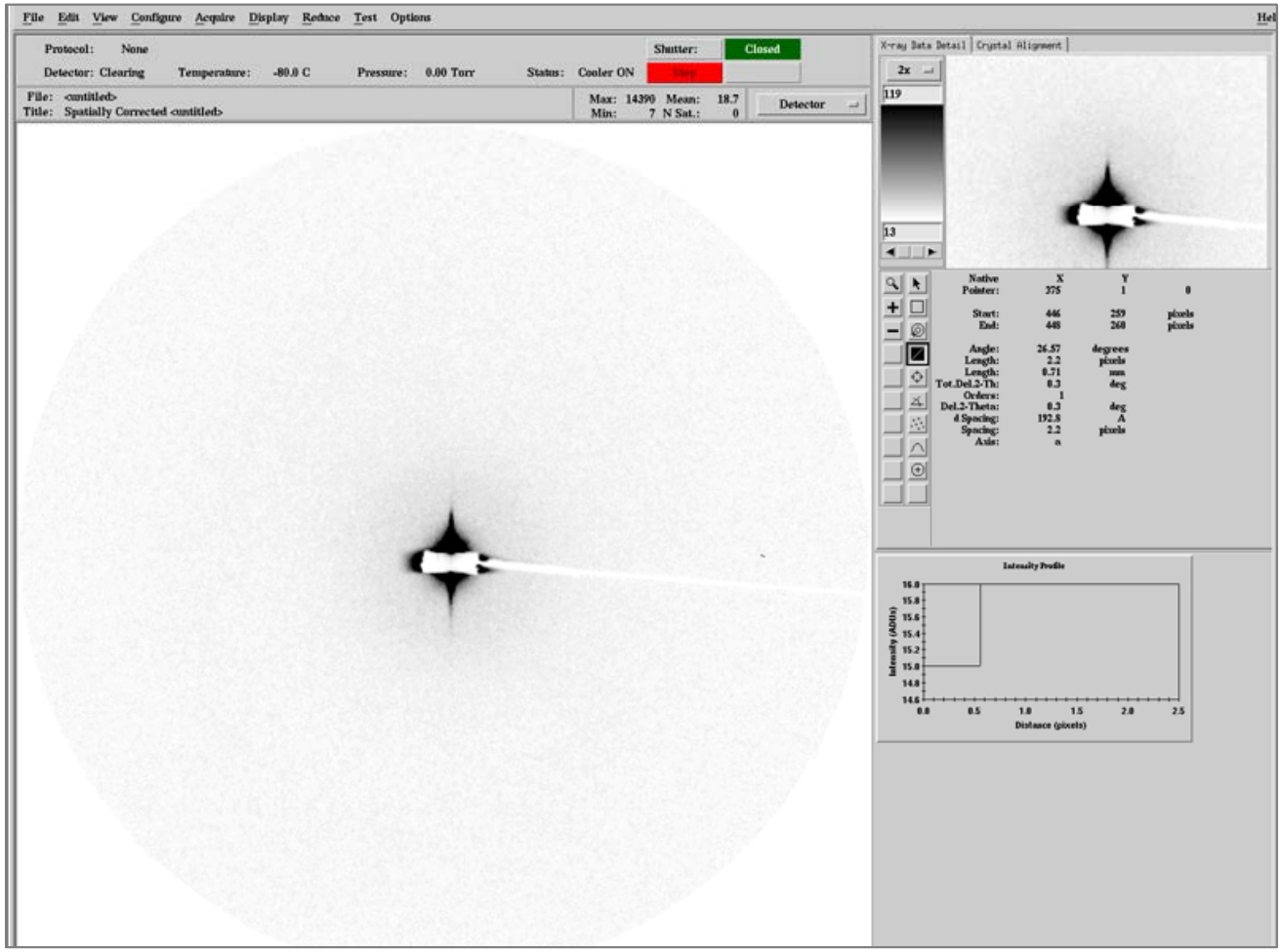
with special thanks to
Paul Kienzle
NIST Center for Neutron Research, NIST, Gaithersburg, MD

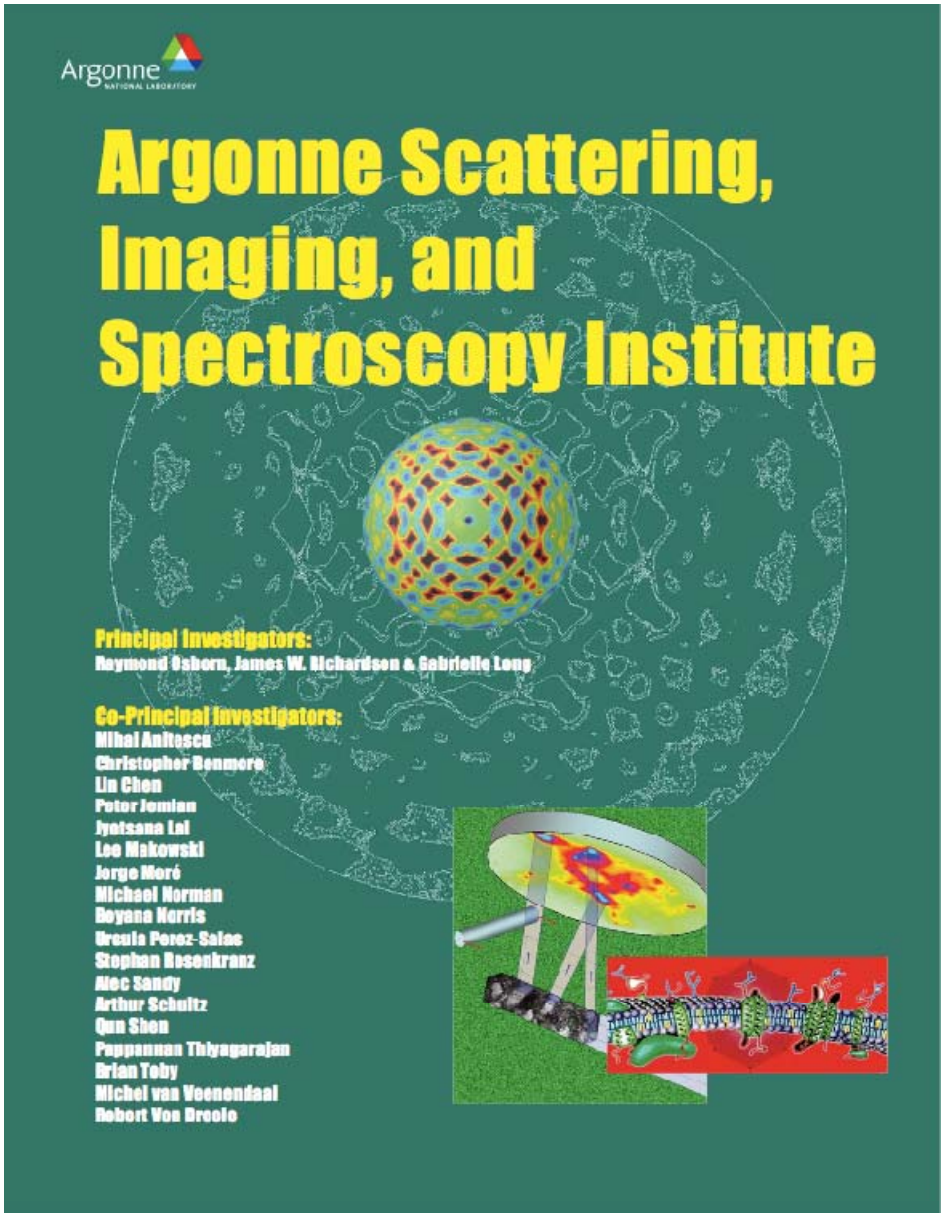
Where we want to be



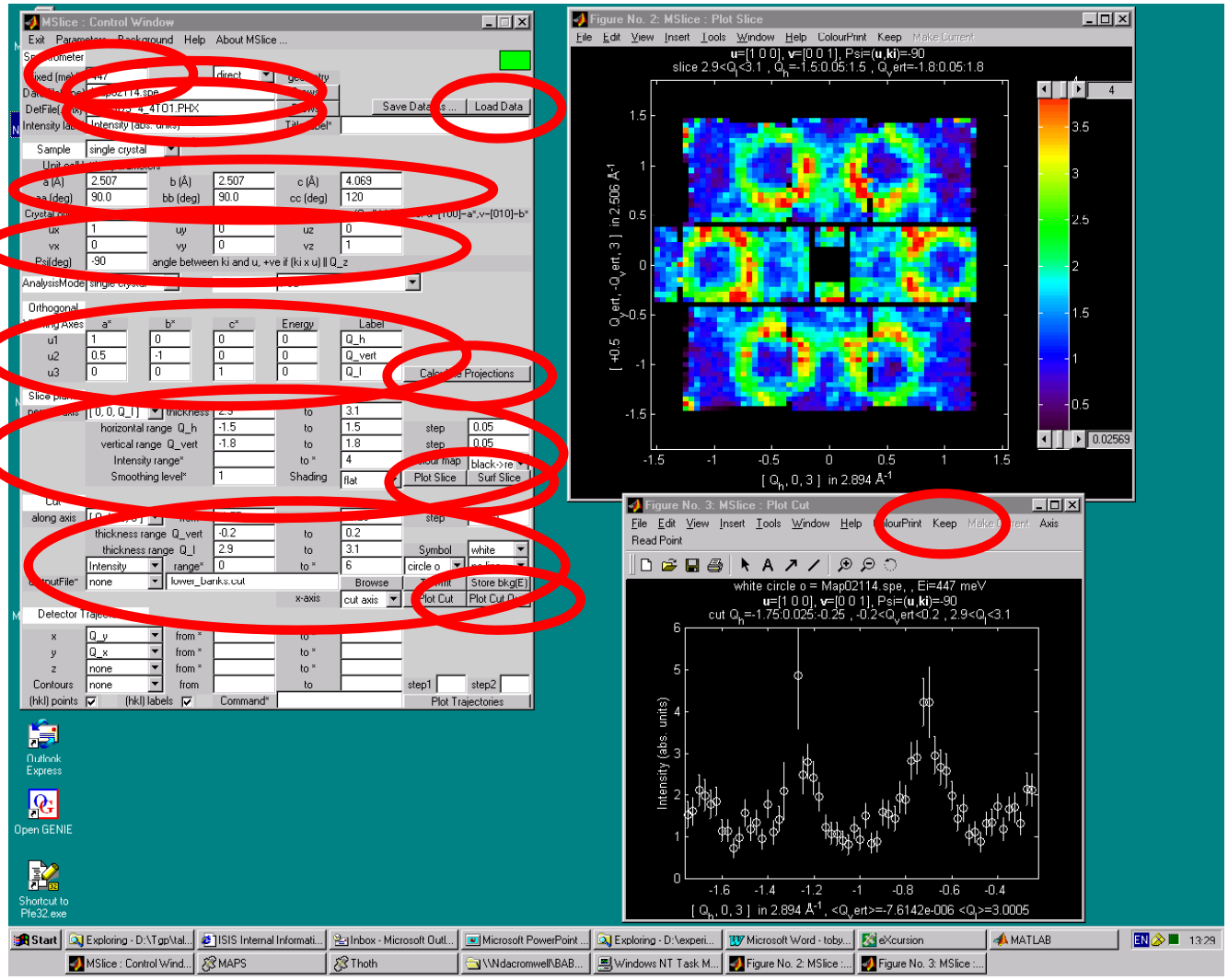
QuickTime™ and a
Microsoft Video 1 decompressor
are needed to see this picture.

Where we are

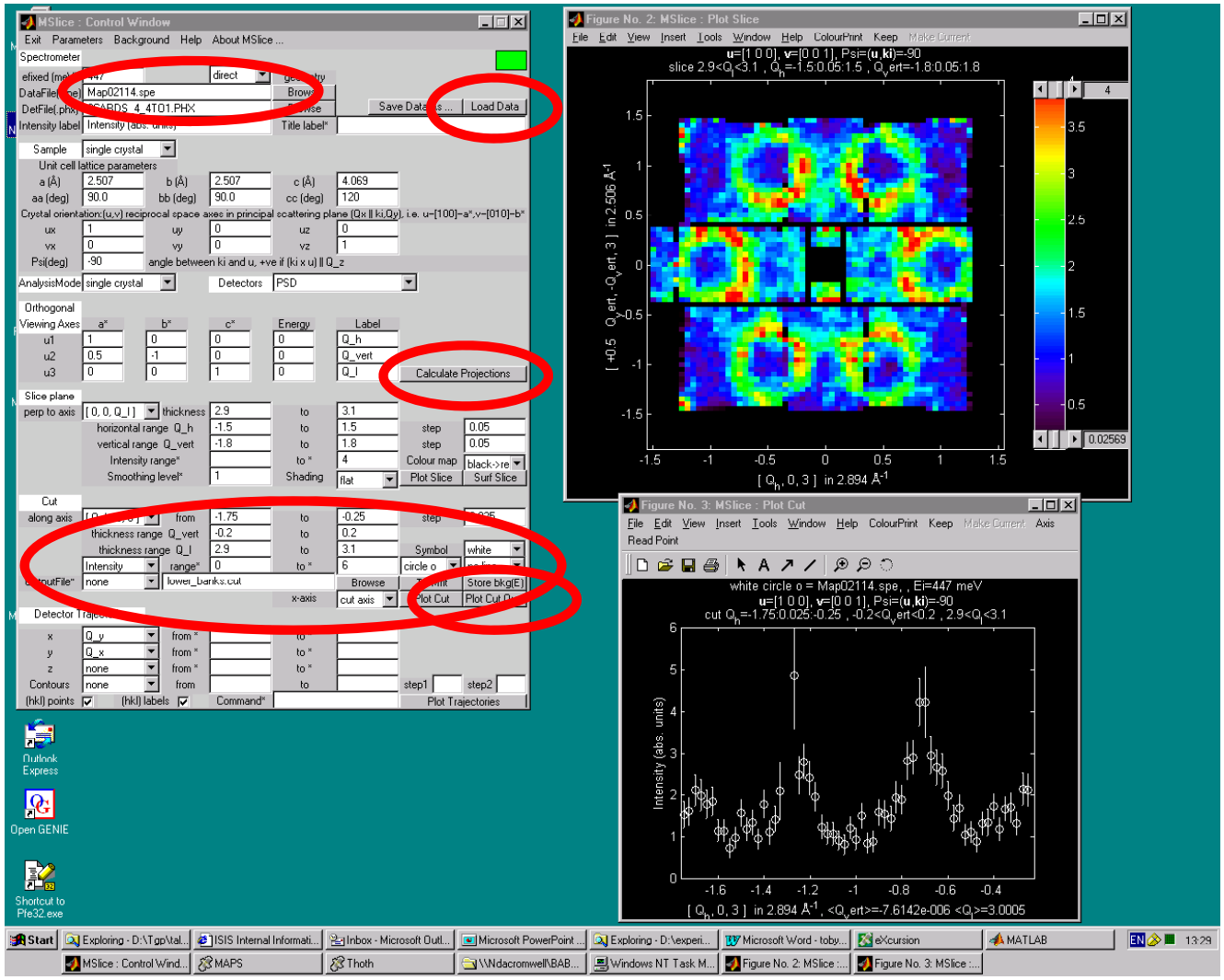

File Edit View Configure Acquire Display Reduce Test Options
Protocol: None
Shutter: Closed
Detector: Clearing Temperature: -80.0 C Pressure: 0.00 Torr Status: Cooler ON
File: <untitled>
Title: Spatially Corrected <untitled>
Max: 14390 Mean: 18.7
Min: 7 N Sat.: 0
Detector
X-ray Data Detail
Crystal Alignment
2x
119
13
Intensity Profile
Intensity (ADUs)
Distance (pixels)


Argonne
NATIONAL LABORATORY
Argonne Scattering, Imaging, and Spectroscopy Institute
Principal Investigators:
Raymond Osborn, James W. Richardson & Gabrielle Long
Co-Principal Investigators:
Mihai Anitescu
Christopher Benmore
Lin Chen
Peter Jemian
Jyotsana Lal
Lee Makowski
Jorge Moré
Michael Norman
Boyana Norris
Ursula Perez-Salas
Stephan Rosenkranz
Alec Sandy
Arthur Schultz
Qun Shen
Pappannan Thiyagarajan
Brian Toby
Michel van Veenendaal
Robert Von Dreele

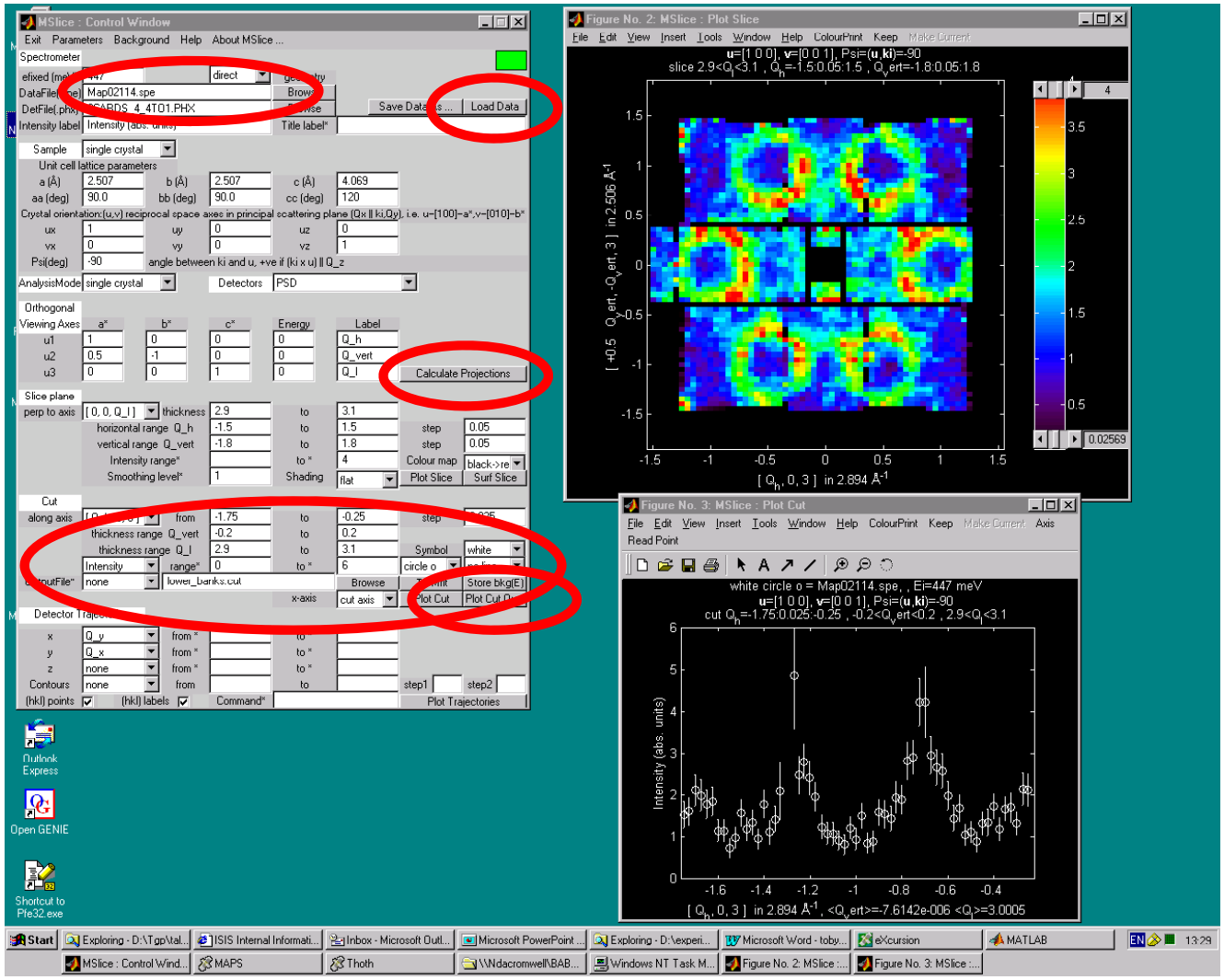
Interactive Friction



Interactive Friction (Take II)



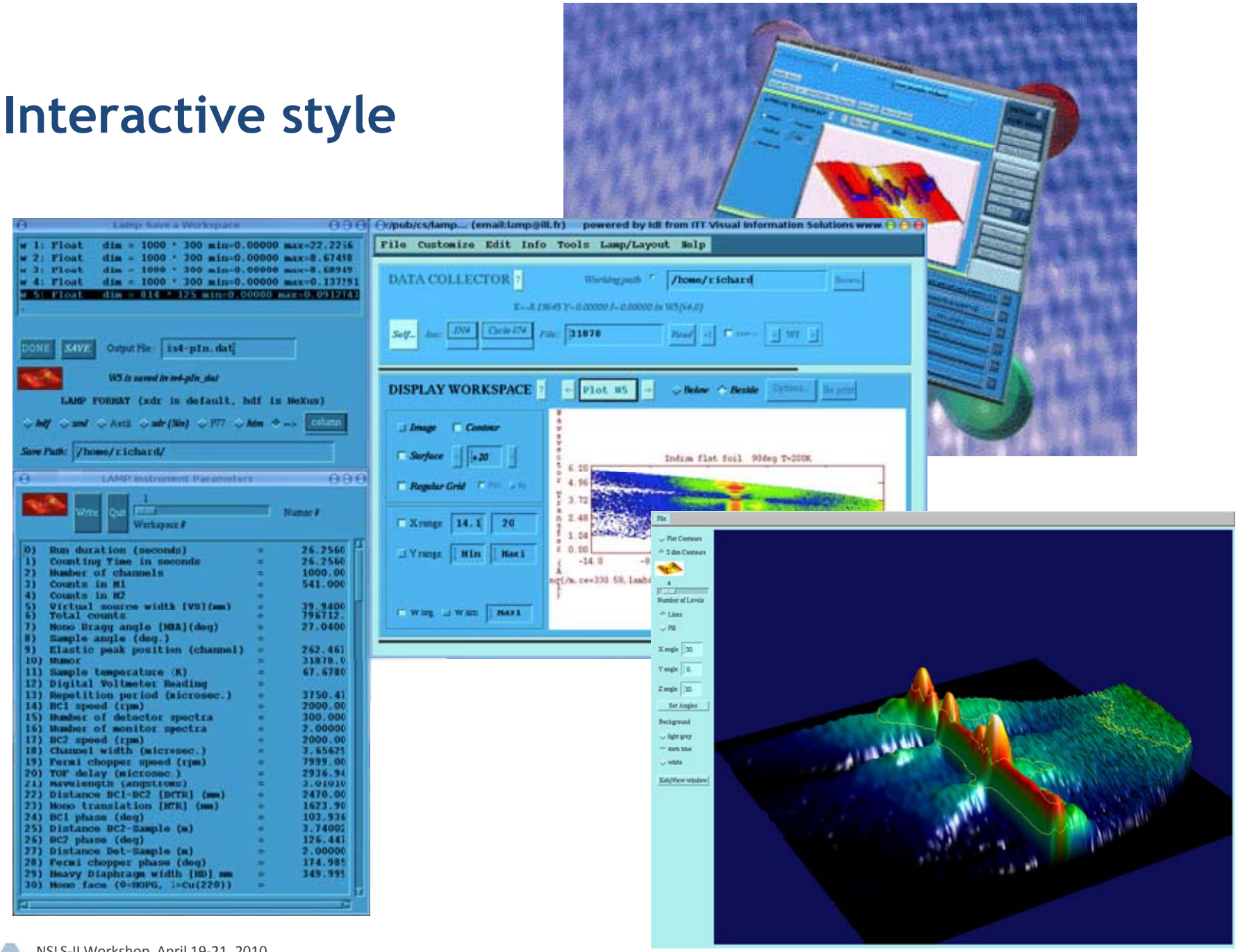
Interactive Friction (Take III)



Issues

- Repetitive workflow
- Insufficient data persistence
- Poor integration with previous analysis (or operating system)
- Lack of automation (scripting)
- No memory management
- Use of proprietary language

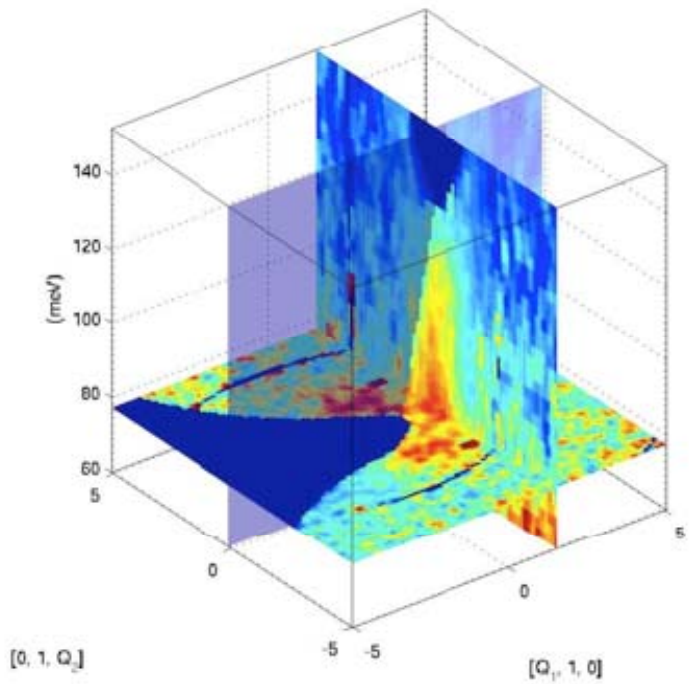
Interactive style



Issues

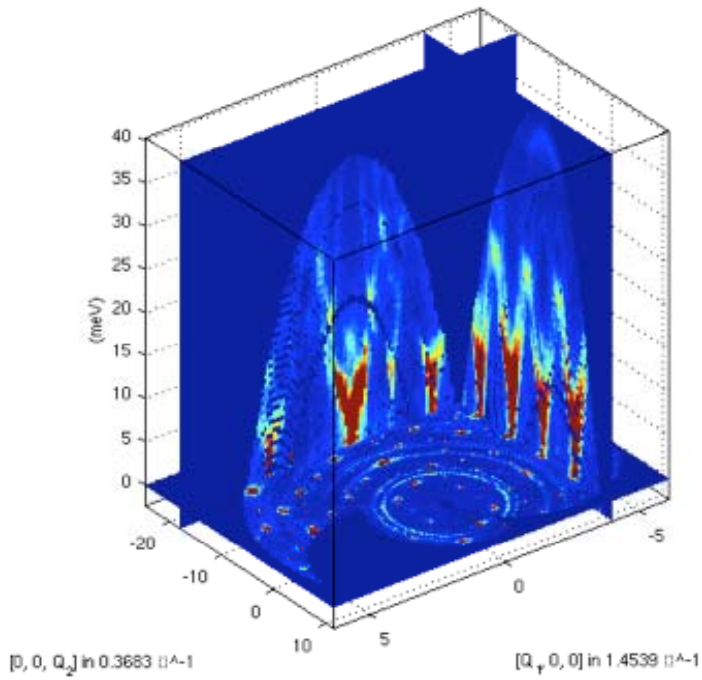
- Inadequate data model
- Insufficient and inconsistent access to instrumental metadata
- Use of proprietary language

Measurements of S(Q,ω) at ISIS (Horace-mode)



Magnetic Fluctuations in MnSi

R.A.Ewings, T.G.Perring (ISIS) unpublished



Lattice vibrations in calcite ($CaCO_3$)

Beth Cope, Martin Dove (Cambridge) unpublished

Issues

- Never, ever, define a “temporary” file format.

Problems to address

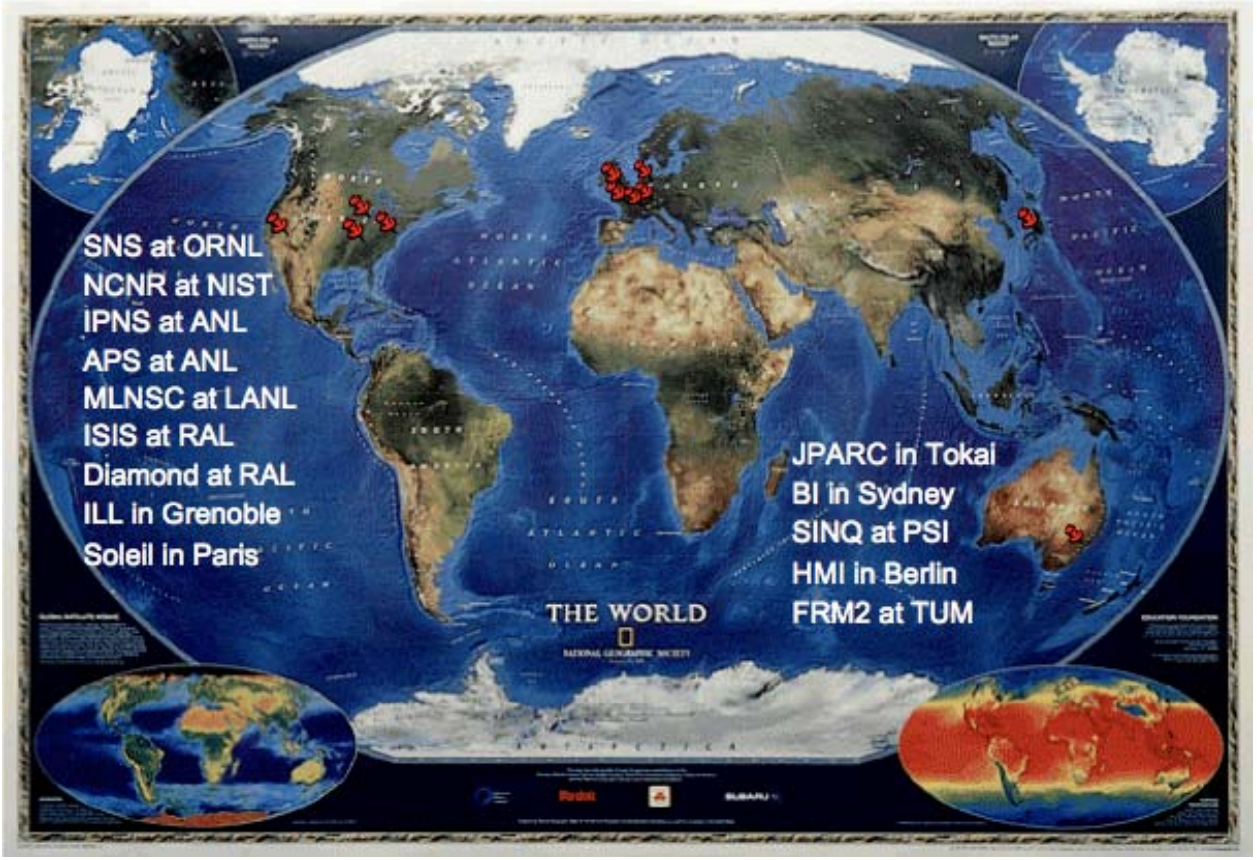
- Ensure that the workflow does not actively discourage data analysis
- Store the results of data analysis as long as needed (not limited by memory)
- Ensure that all relevant instrumental metadata is carried with the analysis
- Allow all repetitive actions to be scripted by the user
- Do not use proprietary languages and packages unless the functionality really demands it

NeXpy - a Python-based approach

- What NeXpy is not:
  - a comprehensive solution to all the issues so far raised
  - particularly sophisticated or novel
- What NeXpy is:
  - a toolbox for manipulating and visualizing arbitrary NeXus data
  - a possible scripting engine for GUI applications
  - a demonstration of the value of combining:
    - a flexible data model
    - a powerful scripting language

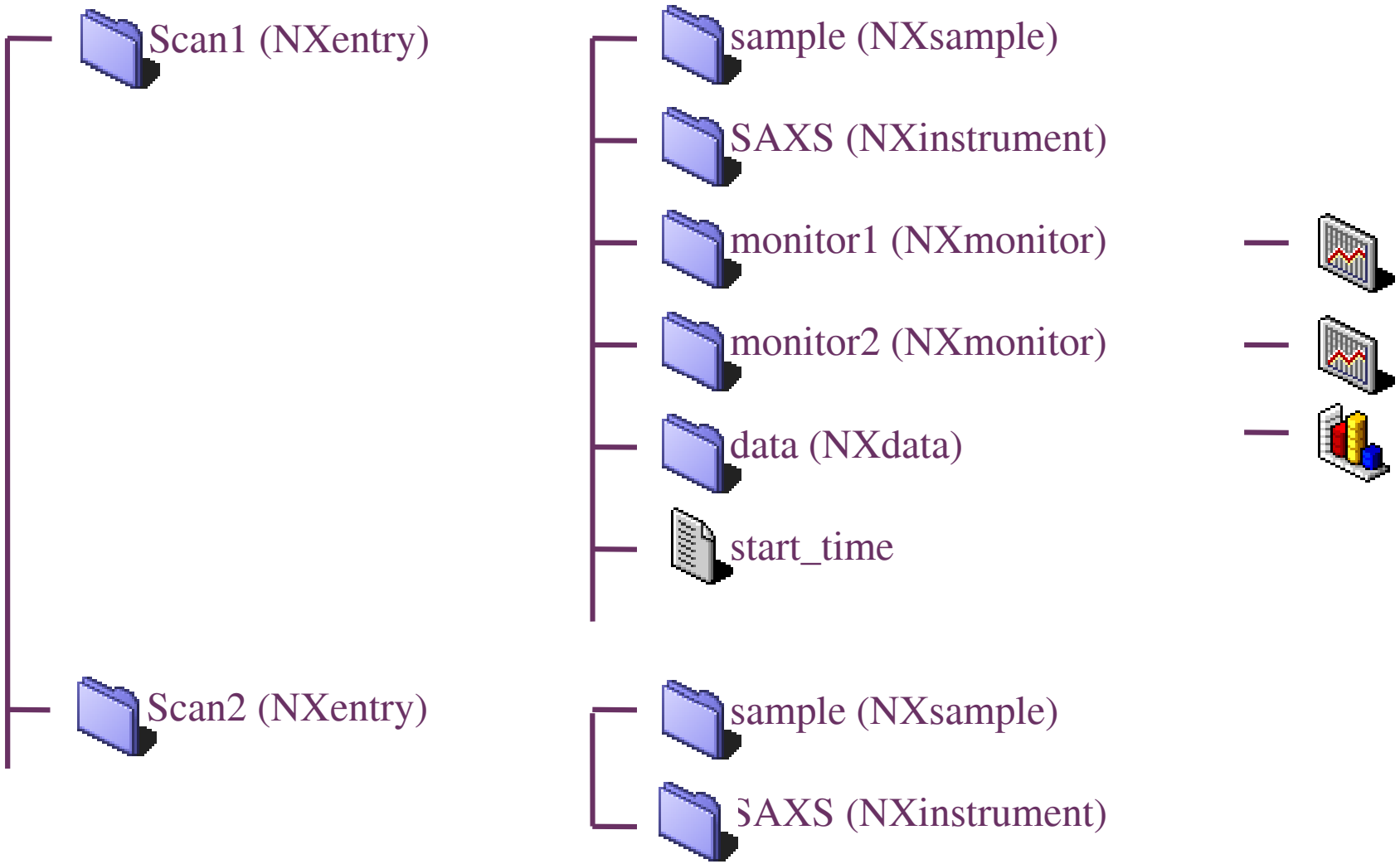
NeXus + NumPy

NeXus



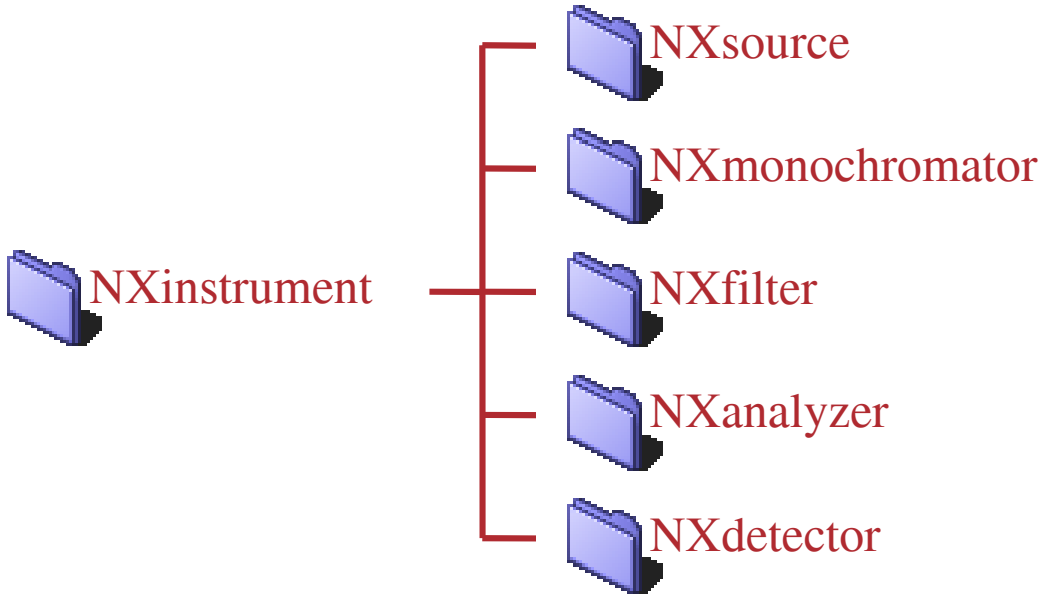
- The NeXus International Advisory Committee represents 16 international facilities and institutions
- If you want to use HDF5, there is no reason not to use the NeXus design principles
- If there are deficiencies in NeXus, join the NeXus mailing list and make a nuisance of yourself

NeXus Hierarchy

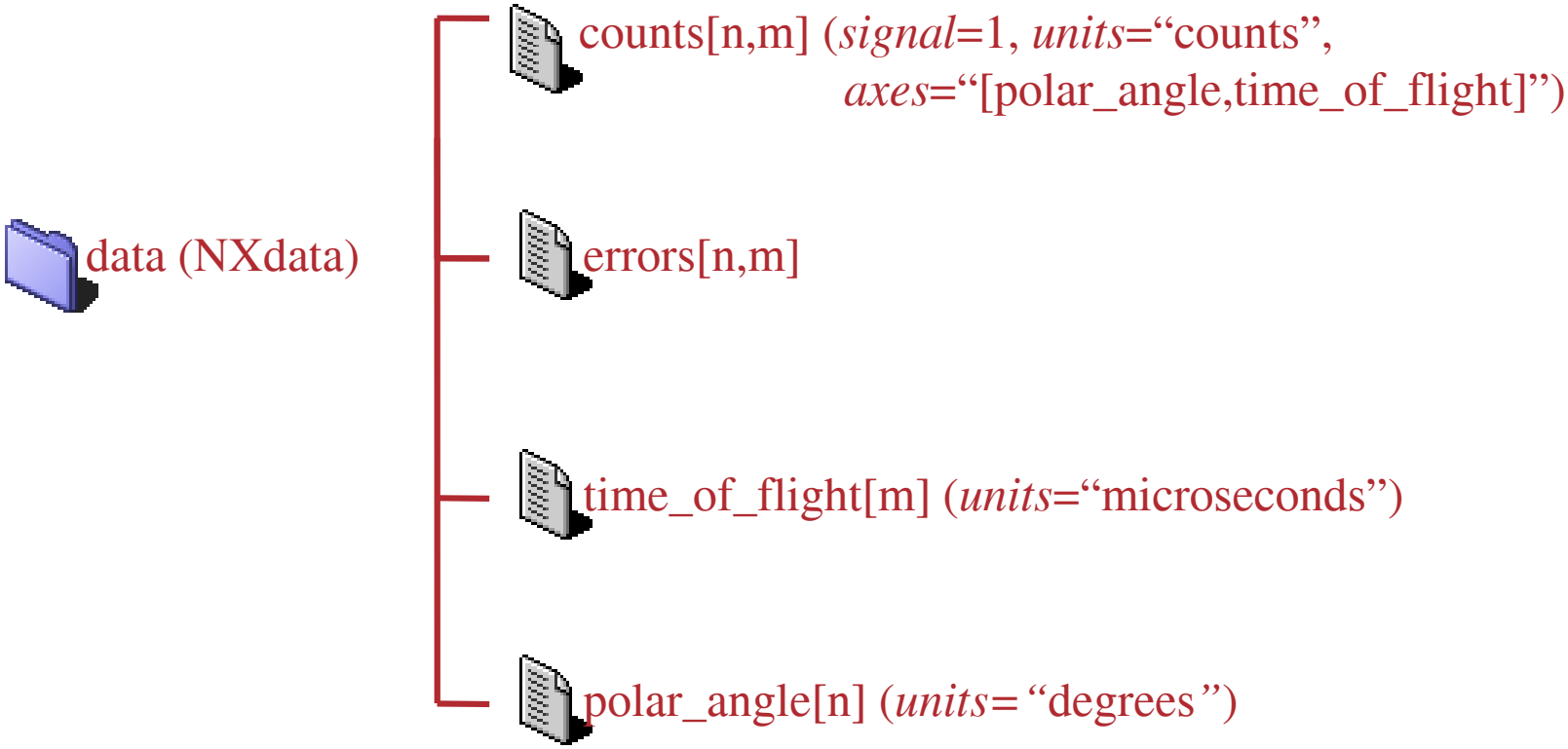


Simplest NeXus File

NXinstrument



NXdata Structures



Numpy

Array manipulation routines — NumPy v1.5.dev8106 Manual (DRAFT)

http://docs.scipy.org/doc/numpy/reference/routines.array-manipulation.html

numpy

Numpy and Scipy Documentation » NumPy Reference » Routines »

previous | next | modules | index

Array manipulation routines

Changing array shape

reshape(a, newshape[, order])	Gives a new shape to an array without changing its data.
ravel(a[, order])	Return a flattened array.
ndarray.flat	A 1-D iterator over the array.
ndarray.flatten([order])	Return a copy of the array collapsed into one dimension.

Transpose-like operations

rollaxis(a, axis[, start])	Roll the specified axis backwards, until it lies in a given position.
swapaxes(a, axis1, axis2)	Interchange two axes of an array.
ndarray.T	Same as self.transpose(), except that self is returned if self.ndim < 2.
transpose(a[, axes])	Permute the dimensions of an array.

Changing number of dimensions

atleast_1d(*arys)	Convert inputs to arrays with at least one dimension.
atleast_2d(*arys)	View inputs as arrays with at least two dimensions.
atleast_3d(*arys)	View inputs as arrays with at least three dimensions.
broadcast	Produce an object that mimics broadcasting.
broadcast_arrays(*args)	Broadcast any number of arrays against each other.
expand_dims(a, axis)	Expand the shape of an array.
squeeze(a)	Remove single-dimensional entries from the shape of an array.

Changing kind of array

asarray(a[, dtype, order])	Convert the input to an array.
asanyarray(a[, dtype, order])	Convert the input to a ndarray, but pass ndarray subclasses through.
asmatrix(data[, dtype])	Interpret the input as a matrix.
asfarray(a[, dtype])	Return an array converted to a float type.
asfortranarray(a[, dtype])	Return an array laid out in Fortran order in memory.
asscalar(a)	Convert an array of size 1 to its scalar equivalent.
require(a[, dtype, requirements])	Return an ndarray of the provided type that satisfies requirements.

Table Of Contents

Array manipulation routines
- Changing array shape
- Transpose-like operations
- Changing number of dimensions
- Changing kind of array
- Joining arrays
- Splitting arrays
- Tiling arrays
- Adding and removing elements
- Rearranging elements

Previous topic

numpy.bmat

Next topic

numpy.reshape

This Page

Show Source

Edit page

Quick search

Go

Enter search terms or a module, class or function name.

```
!
!Determine Ei from monitors 2 and 3
!
    subroutine get_Ei (Ei, a_M, s_M, t_M, b_M, d_M)

    include 'GET_SOURCES:COMCON.INC'
    include 'GET_SOURCES:COMFOR.INC'
    include 'GET_SOURCES:COMRUN.INC'
    include 'GET_SOURCES:COMINS.INC'
    include 'GET_SOURCES:COMTCB.INC'
    include 'GET_SOURCES:COMRAW.INC'
    real*4 y_t(MAX_CHAN), e_t(MAX_CHAN), a_M(3), s_M(2:3), t_M(2:3), b_M(2:3), d_M(2:3)
    real*4 t_min, t_max, fwhm
    parameter (c = 2286.26, pi = 3.1415927)

!Initialize peak search range
    if (Ei .gt. 0.0)) then
        t_M(2) = c * d_M(2) / sqrt (Ei)
        t_min = 0.9 * t_M(2)
        t_max = 1.1 * t_M(2)
    else
        t_min = 400
        t_max = 12000
    end if
!Loop over two monitors
    do i_m = 2,3
!Input Monitor data
        call get_tcb_time (timr(m_det(i_m)))
        call get_rawdata (1, spec(m_det(i_m)), 1)
        do i = 1,n_tc
            dt = tc(i+1) - tc(i)
            y_t(i) = rawdata(i) / dt
            e_t(i) = rawdata(i) / dt**2
        end do
!Determine peak centres from a moment analysis
        call get_moments (n_tc, tc, y_t, e_t, t_min, t_max, a_M(i_m)
b_M(i_m))
        if (i_m .eq. 2) then
            t_M(3) = t_M(2) * d_M(3) / d_M(2)
            t_min = 0.95 * t_M(3)
            t_max = 1.05 * t_M(3)
        end if
    end do
!Perform energy calculation
    Ei = ((c * (d_M(3)-d_M(2))) / (t_M(3) - t_M(2))) ** 2

    return
    end
```

```
    subroutine get_moments (n_tc, t, y_t, e_t, t_min, t_max, area, centre, sigma, fwhm, bkgd)
!
!  Modified:
!   10/11/94  T.G.Perring
!
!
    include 'GET_SOURCES:COMCON.INC'

    real*4 t(MAX_CHAN), y_t(MAX_CHAN), e_t(MAX_CHAN), t_min, t_max, centre, sigma, area, bkgd
    real*4 h, xp, xm, deriv, error, dtp, dtm, dt, xmin, x, dx, y, a0, a1, a2, a3
    integer*4 n_tc, i, imax, ip, im, nbkgd

!Determine peak position - assume it lies between t_min and t_max
    i = 1
    do while (i .lt. n_tc .and. t(i) .lt. t_min)
        i = i + 1
    end do
    h = 0.0
    do while (i .lt. n_tc .and. t(i) .lt. t_max)
        if (y_t(i) .gt. h) then
            h = y_t(i)
            imax = i
        end if
        i = i + 1
    end do
!Determine width of peak
    ip = imax
    do while (ip .lt. n_tc-2 .and. y_t(ip) .gt. h/2)
        ip = ip + 1
    end do
    if (y_t(ip-1) .ne. y_t(ip)) then
        xp = t(ip) - (((h/2) - y_t(ip)) * (t(ip) - t(ip-1)) / (y_t(ip-1) - y_t(ip)))
    else
        xp = 0.5 * (t(ip-1) + t(ip))
    end if
    im = imax
    do while (im .gt. 2 .and. y_t(im) .gt. h/2)
        im = im - 1
    end do
    if (y_t(im+1) .ne. y_t(im)) then
        xm = t(im) + (((h/2) - y_t(im)) * (t(im+1) - t(im)) / (y_t(im+1) - y_t(im)))
    else
        xm = 0.5 * (t(im) + t(im+1))
    end if
    fwhm = xp - xm
!Determine time range for moment analysis - use derivatives to define peaks
    deriv = -1000
    error = 0.0
    do while (ip .le. n_tc-2 .and. deriv .lt. -error)
        dtp = 0.5 * (t(ip+2) - t(ip))
        dtm = 0.5 * (t(ip+1) - t(ip-1))
        deriv = 0.5*(((y_t(ip+1) - y_t(ip)) / dtp) + ((y_t(ip) - y_t(ip-1)) / dtm))
```

Python API

- There are two levels to the Python API
  - A one-to-one mapping of the C-API returning Numpy arrays (napi.py)
  - A one-to-one mapping of the NeXus data objects into Python classes (tree.py)

```
>>> f=nexus.open("data/chopper.nxs")>>> f.opengroup("entry","NXentry")>>>
f.opendata("title")>>> f.getdata()'MgB2 PDOS 43.37g 8K 120meV E0@240Hz
T0@120Hz'
>>> f.closedata()
>>> f.closegroup()
>>> f.close()
```

```
>>> a=nexus.load("data/chopper.nxs")>>> a.entry.data.nxtree()data:NXdata  data =
int32(148x750)    @axes = polar_angle:time_of_flight    @long_name = Neutron
Counts    @signal = 1    @units = counts  polar_angle = float32(148)
@long_name = Polar Angle [degrees]    @units = degrees  time_of_flight = [ 1900.
1902.  1904. ...,  3396.  3398.  3400.]    @long_name = Time-of-Flight
[microseconds]    @units = microseconds  title = MgB2 PDOS 43.37g 8K 120meV
E0@240Hz T0@120Hz>>> print a.entry.data.titleMgB2 PDOS 43.37g 8K 120meV
E0@240Hz T0@120Hz
```

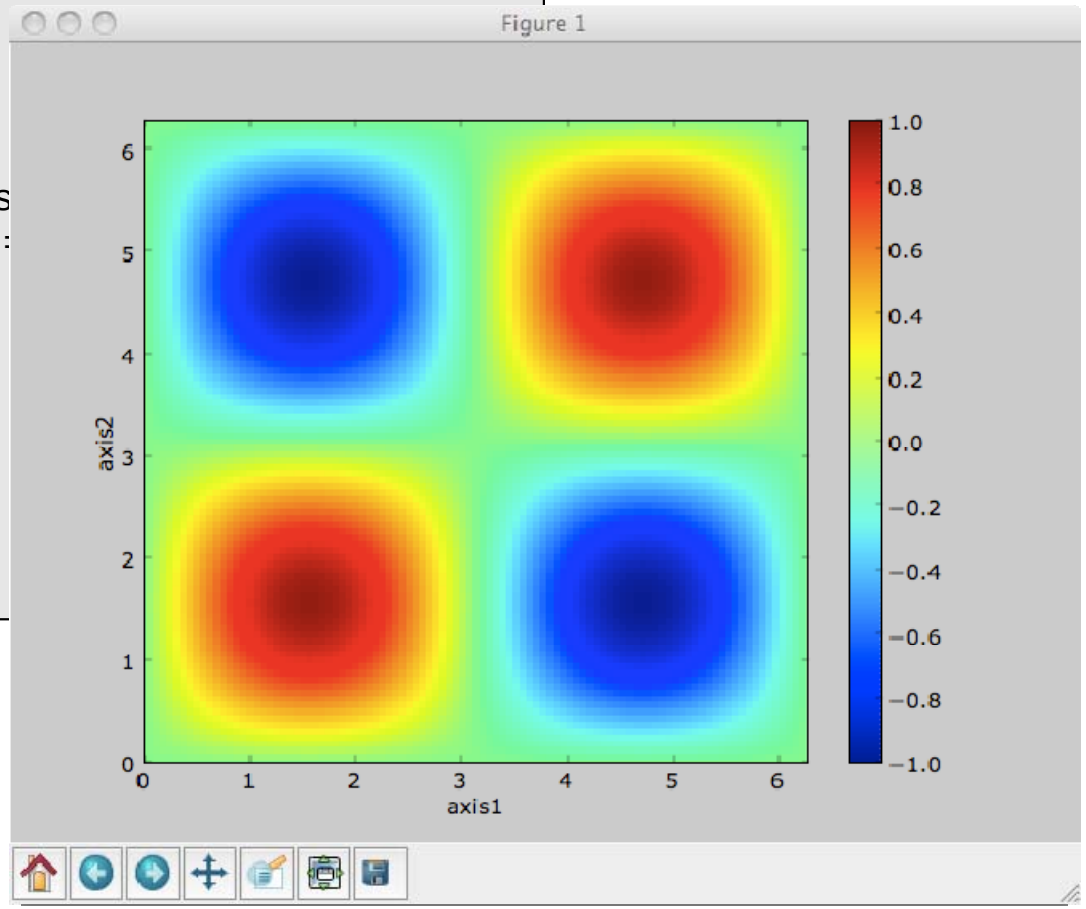
Features of NeXus tree interface

- The entire tree structure of a NeXus file can be loaded with a single command
- The data values are not read unless directly referenced
- Each NeXus object is a sub-class of the class NXnode
  - NeXus data have class SDS
  - NeXus groups have class NXgroup, with subclasses for each group class
    - e.g., NXentry, NXdata, NXsample ...
- All objects of the same class can easily be listed
  - e.g., entry.NXdata[0], entry.NXdata[1], ...
- All data attributes have a class NXattr
  - e.g., entry.data.polar_angle.units="K"
- Command-line assignments automatically convert Numpy data into NeXus objects
  - e.g., typing "entry.sample.temperature=40.0" converts the temperature into a valid SDS object
  - this is achieved by customizing __setattr__

Interactive manipulation of NeXus data

- The syntax makes the creation of standard-conforming NeXus data structures simple

```
>>> entry=NXentry()>>> x=np.linspace(0,2.*np.pi,101)
>>> y=x
>>> X,Y=np.meshgrid(x,y)
>>> z=np.sin(X)*np.sin(Y)
>>> entry.data=NXdata(z,(x,y))
>>> entry.nxtree()entry:NXentry  data:NXdata   axis
float64(101)   signal = float64(101x101)     @axes =
>>> entry.nxplot()
```



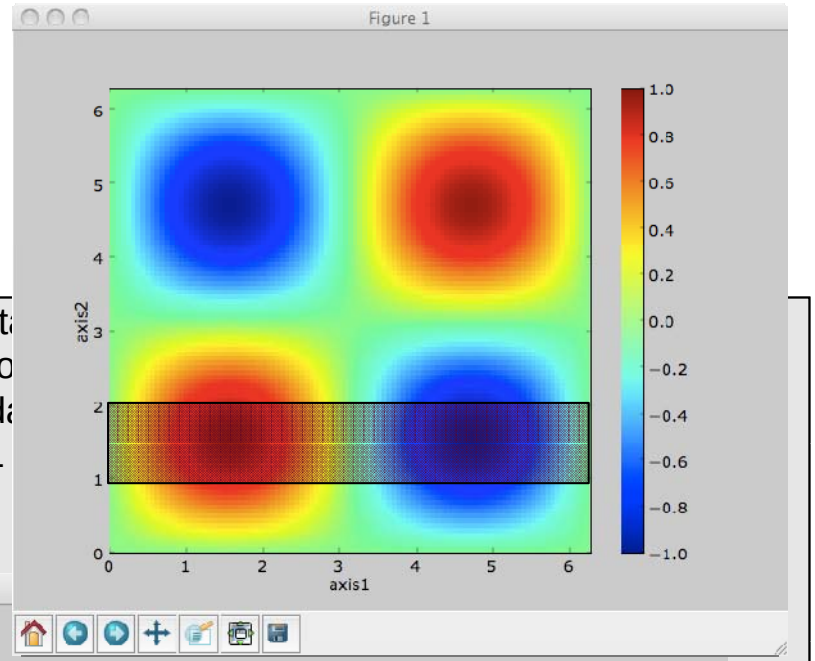
NXdata group manipulations

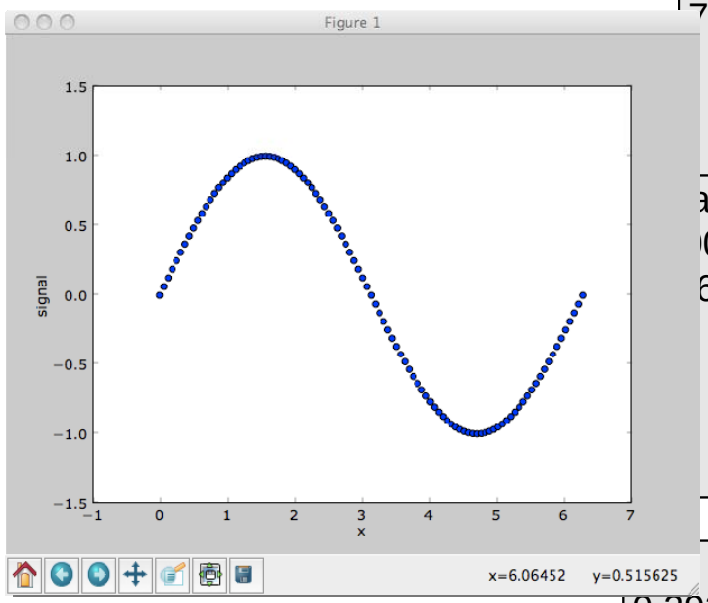
- The NXdata objects are designed to be manipulated:
  - Diced and sliced
  - Scaled
  - Added and subtracted

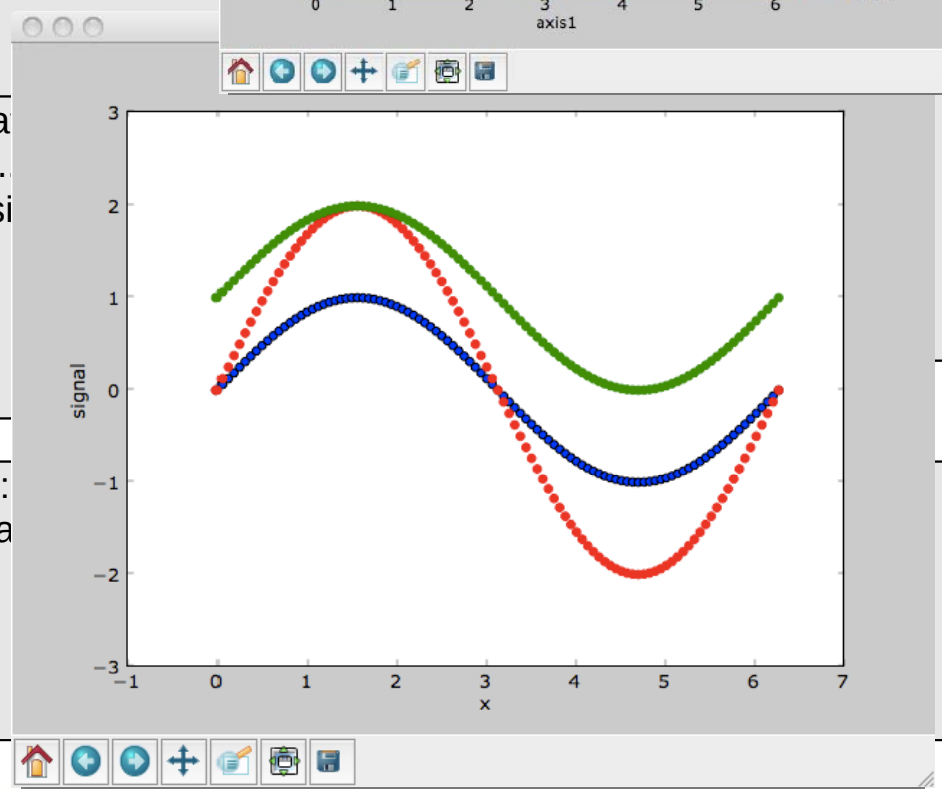
```
>>> data[1:2.].nxsum(0).nxtree()dat
float64(101)    @axes = axis2    @lo
2.0106192983    @signal = 1>>> da
0.00000000e+00   7.07106781e-01
7.07106781e-01  -2.44929360e-16
```

```
ata).nxtree()data:NXda
0   2.00000000e+00 ..
6 ]    @axes = x    @si
```

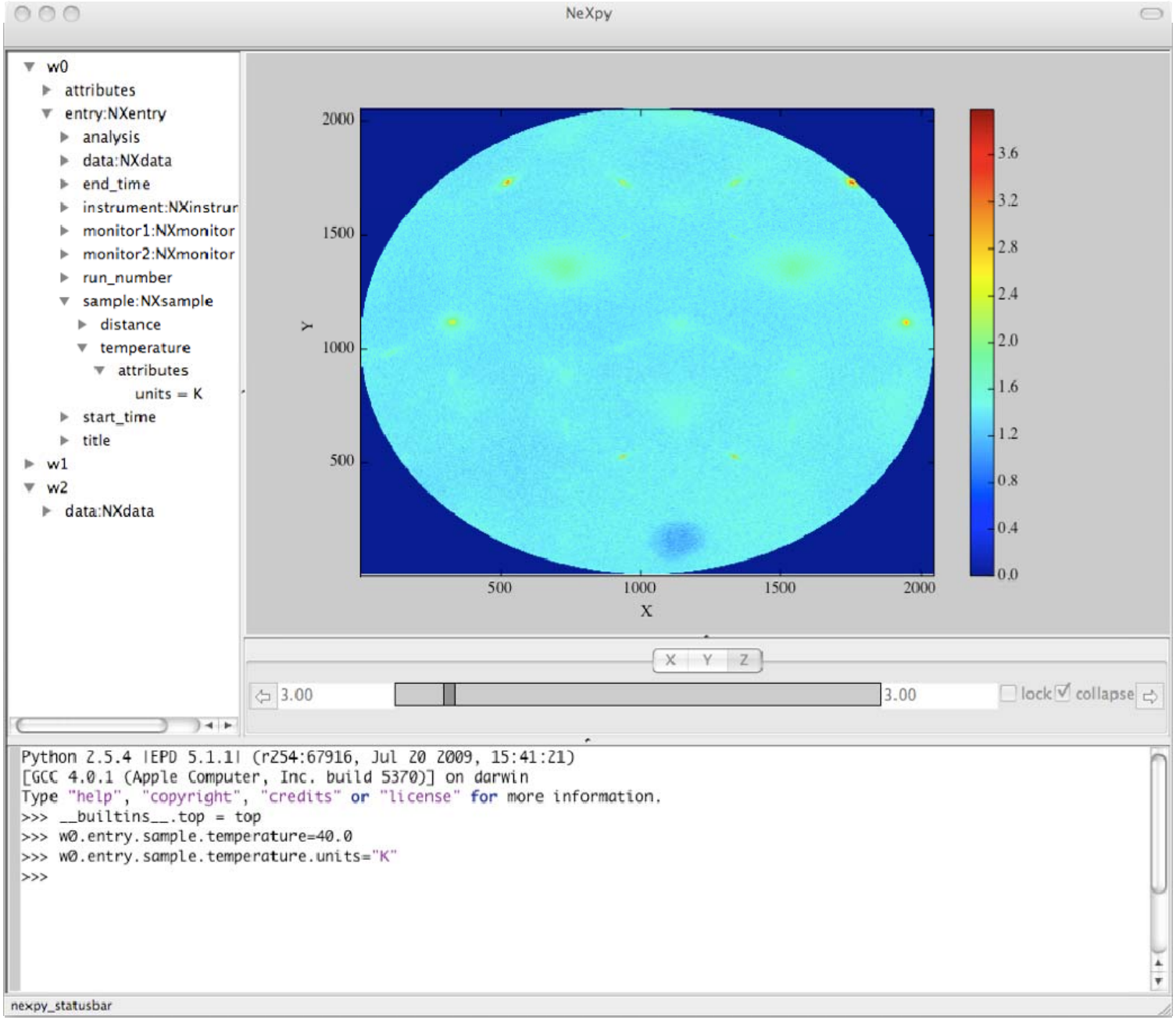
```
(data+1).nxtree()data:
0.29289322  1.        ]    @a
```







NeXpy GUI


NeXpy
w0
attributes
entry:NXentry
analysis
data:NXdata
end_time
instrument:NXinstrur
monitor1:NXmonitor
monitor2:NXmonitor
run_number
sample:NXsample
distance
temperature
attributes
units = K
start_time
title
w1
w2
data:NXdata
2000
1500
1000
500
Y
500
1000
1500
2000
X
3.6
3.2
2.8
2.4
2.0
1.6
1.2
0.8
0.4
0.0
X Y Z
3.00
3.00
lock
collapse
Python 2.5.4 |EPD 5.1.1| (r254:67916, Jul 20 2009, 15:41:21)
[GCC 4.0.1 (Apple Computer, Inc. build 5370)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> __builtins__.top = top
>>> w0.entry.sample.temperature=40.0
>>> w0.entry.sample.temperature.units="K"
>>>
nexpy_statusbar

On the To-Do List (near-term)

- Adding a data editor
- Incorporating generalized coordinate transformations
- Incorporating Open GL modules to improve over current Matplotlib speeds
- Adding full 3D plotting
  - Using Mayavi
- Adding a fitting pane
  - Using Paul Kienzle's PARK framework
- Installing it as a standard part of the NeXus distribution