WHY NeXus?

RAYMOND OSBORN
Neutron & X-ray Scattering Group
Materials Science Division

Work supported by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division.

ORSO – June 14, 2021

STATUS OF NeXus

https://www.nexusformat.org

- The NeXus data format is now well established as an international standard for the storage of data at neutron and synchrotron x-ray facilities.
- It is the official archive format for the entire instrument suite of a number of facilities.
  - Particularly spallation neutron sources, such as SNS, ISIS, and, in the near future, ESS.
- There is active participation in the NeXus International Advisory Committee by nearly 20 facilities in Asia, Europe, and North America.
  - Official NIAC meetings take place every two years with code camps nearly every year.
  - Monthly online meetings (even before the pandemic) deal with maintenance issues.
- Dectris has worked with NIAC to adopt NeXus for detector storage.

NeXus INTERNATIONAL ADVISORY COMMITTEE

Chair: Ben Watts (Swiss Light Source)

- Advanced Light Source, USA
- Advanced Photon Source, USA
- Bragg Institute, Australia
- Canadian Light Source, Canada
- Diamond/ISIS, UK
- European Synchrotron Radiation Facility, France
- European XFEL, Germany
- Extreme Light Infrastructure, Eastern Europe
- Helmholtz Zentrum Berlin, Germany
- J-PARC, Japan
- Los Alamos National Laboratory, USA
- NSLS-II, USA
- Spallation Neutron Source/HFIR, USA
- Spring8, Japan
- Swiss Light Source/SINQ, Switzerland
- Synchrotron Soleil, France

BASIS OF NeXus Data Format

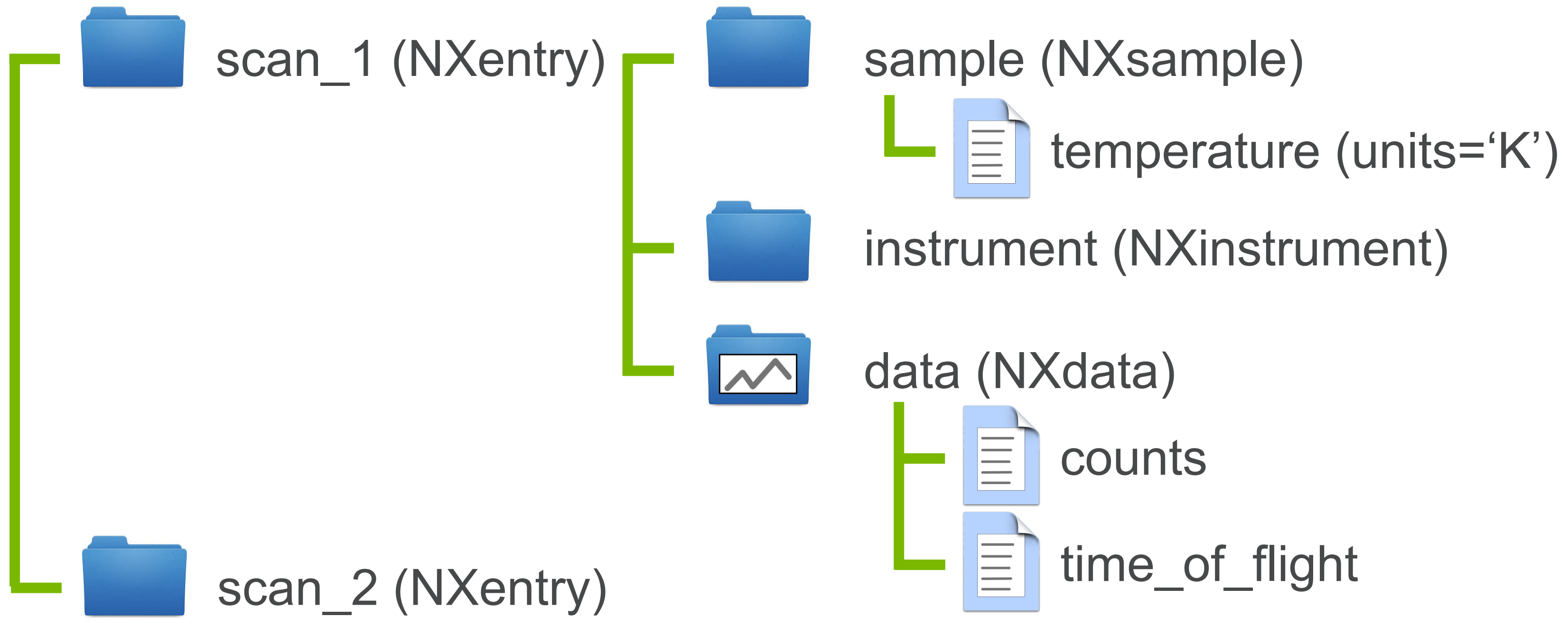
Semantic HDF5 files

- NeXus files are HDF5 files with the addition of some semantics.
  - Simple design rules to make the files easy to navigate.
  - A list of definitions that cover most experimental metadata.
- The purpose of these rules is to make them self-describing.
  - It is usually possible to understand their contents without referring to any documentation.
- NeXus uses a hierarchical design similar to a file system
  - Hierarchy allows complex data to be stored in a readily accessible form
    - Important data at a high level
    - Arcane details at a low level
- Base classes provide a glossary of terms required for most experiments.

DESIGN RULES OF NeXus Data Format

Design Rules

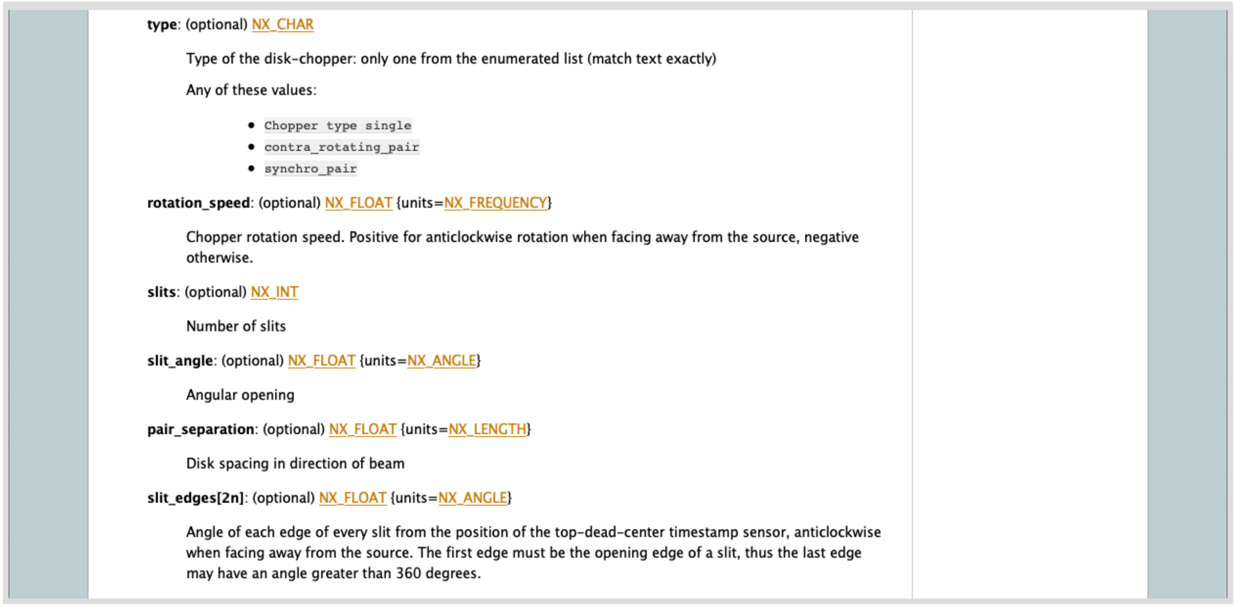
- NeXus files contain three types of object.
  - Groups
  - Fields
  - Attributes



NeXus Base Classes

https://manual.nexusformat.org/classes/base_classes/

- NeXus groups have classes that define what they contain.
  - *e.g.*, NXsample groups contain sample information, *etc*.



type: (optional) NX_CHAR

Type of the disk-chopper: only one from the enumerated list (match text exactly)

Any of these values:

- `Chopper type single`
- `contra_rotating_pair`
- `synchro_pair`

rotation_speed: (optional) NX_FLOAT {units=NX_FREQUENCY}

Chopper rotation speed. Positive for anticlockwise rotation when facing away from the source, negative otherwise.

slits: (optional) NX_INT

Number of slits

slit_angle: (optional) NX_FLOAT {units=NX_ANGLE}

Angular opening

pair_separation: (optional) NX_FLOAT {units=NX_LENGTH}

Disk spacing in direction of beam

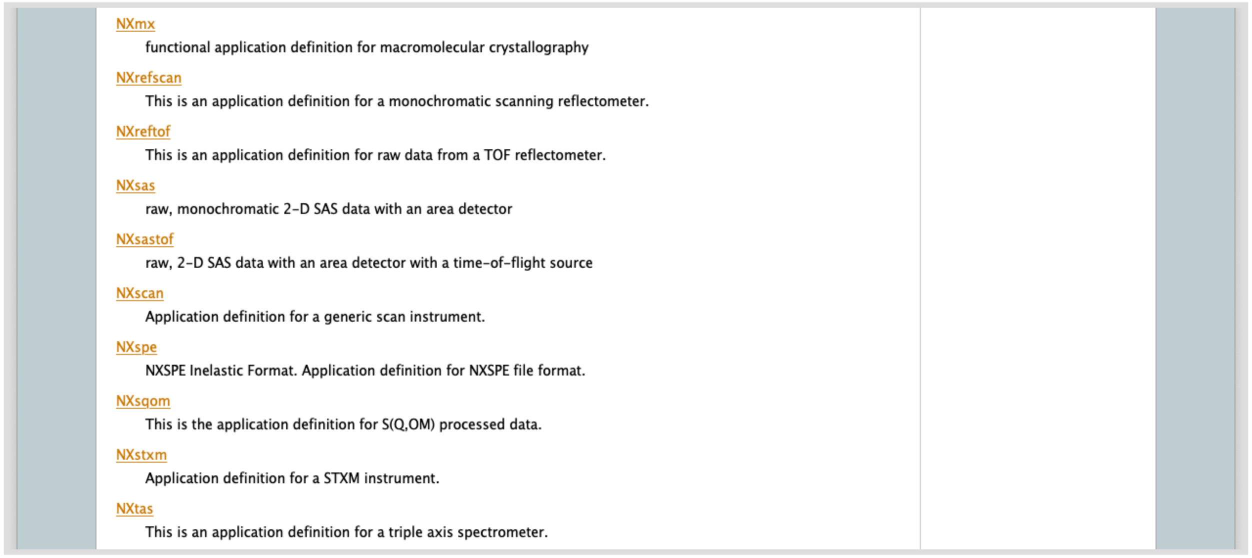
slit_edges[2n]: (optional) NX_FLOAT {units=NX_ANGLE}

Angle of each edge of every slit from the position of the top-dead-center timestamp sensor, anticlockwise when facing away from the source. The first edge must be the opening edge of a slit, thus the last edge may have an angle greater than 360 degrees.

NeXus Application Definitions

https://manual.nexusformat.org/classes/applications/

- NeXus Application Definitions describes which groups and data items have to be present in a file in order to properly describe an application of NeXus.
  - *i.e.*, the minimum required information necessary to satisfy data analysis requirements.



NXmx
: functional application definition for macromolecular crystallography

NXrefscan
: This is an application definition for a monochromatic scanning reflectometer.

NXreftof
: This is an application definition for raw data from a TOF reflectometer.

NXsas
: raw, monochromatic 2-D SAS data with an area detector

NXsastof
: raw, 2-D SAS data with an area detector with a time-of-flight source

NXscan
: Application definition for a generic scan instrument.

NXspe
: NXSPE Inelastic Format. Application definition for NXSPE file format.

NXsqom
: This is the application definition for S(Q,OM) processed data.

NXstxm
: Application definition for a STXM instrument.

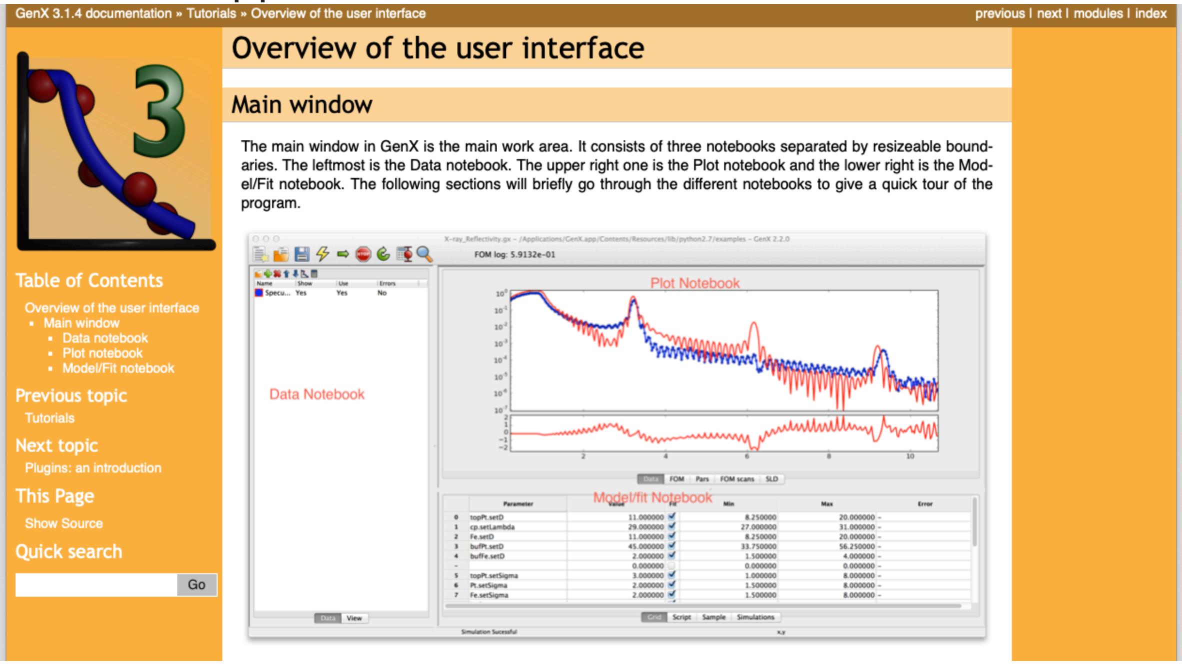
NXtas
: This is an application definition for a triple axis spectrometer.

NeXus Application Definitions

NeXus

Original Motivation

- Application Definitions are intended to provide an interface to analysis software.
  - *e.g.*, a program like GenX could define the Application Definition.
  - Facilities would then export the data in a NeXus file conforming to this definition.
- In practice, such interfaces depend on sociological structures.
  - ORSO?



GenX 3.1.4 documentation » Tutorials » Overview of the user interface   previous | next | modules | index

Overview of the user interface

Main window

The main window in GenX is the main work area. It consists of three notebooks separated by resizeable boundaries. The leftmost is the Data notebook. The upper right one is the Plot notebook and the lower right is the Model/Fit notebook. The following sections will briefly go through the different notebooks to give a quick tour of the program.

X-ray_Reflectivity.gx – /Applications/GenX.app/Contents/Resources/lib/python2.7/examples – GenX 2.2.0

FOM log: 5.9132e-01

Name	Show	Use	Errors
Specu...	Yes	Yes	No

	Parameter	Value	Fit	Min	Max	Error
0	topPt.setD	11.000000	✓	8.250000	20.000000	-
1	cp.setLambda	29.000000	✓	27.000000	31.000000	-
2	Fe.setD	11.000000	✓	8.250000	20.000000	-
3	bufPt.setD	45.000000	✓	33.750000	56.250000	-
4	bufFe.setD	2.000000	✓	1.500000	4.000000	-
-		0.000000		0.000000	0.000000	-
5	topPt.setSigma	3.000000	✓	1.000000	8.000000	-
6	Pt.setSigma	2.000000	✓	1.500000	8.000000	-
7	Fe.setSigma	2.000000	✓	1.500000	8.000000	-

Simulation Sucessful   x,y

ORIGINS OF NeXus

A Format for Data Exchange

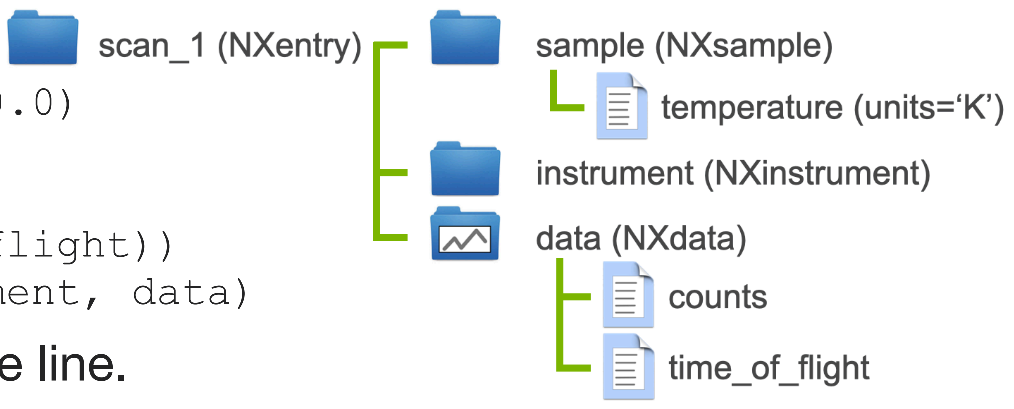
- The origin of NeXus goes back to a series of workshops held in the US, the *SoftNeSS* workshops, nearly 20 years ago.
- The original purpose of NeXus was to empower neutron scattering scientists.
  - Those who perform experiments at multiple facilities.
  - Those who want to use their own software for data analysis.
- The popularity of Python now makes it possible to achieve that original goal.
  - By making it easy to read and plot data from any facility without reading the documentation.
  - By providing an intuitive language to process the data and save the results.
  - By encouraging rapid experimentation and innovation.

ACCESSING NeXus FILES

https://nexpy.github.io/nexpy/

- NeXus has a C-API but it is no longer maintained so the HDF5 library must be used.
- There is, however, a Python API (built on *h5py*).
- The API is designed to be concise and intuitive.
  - *e.g.*, to generate the example on the right.

```
>>> sample = NXsample(temperature=40.0)
>>> sample['temperature'].units='K'
>>> instrument = NXinstrument()
>>> data = NXdata(counts, (time_of_flight))
>>> scan_1 = NXentry(sample, instrument, data)
```



- Existing NeXus files can be read in a single line.
  -

```
>>> pycco = nxload('pycco_120K.nxs', 'rw')
>>> print(pycco['entry/sample/temperature'])
120.0
```

NeXus Python API

https://nexpy.github.io/nexpy/

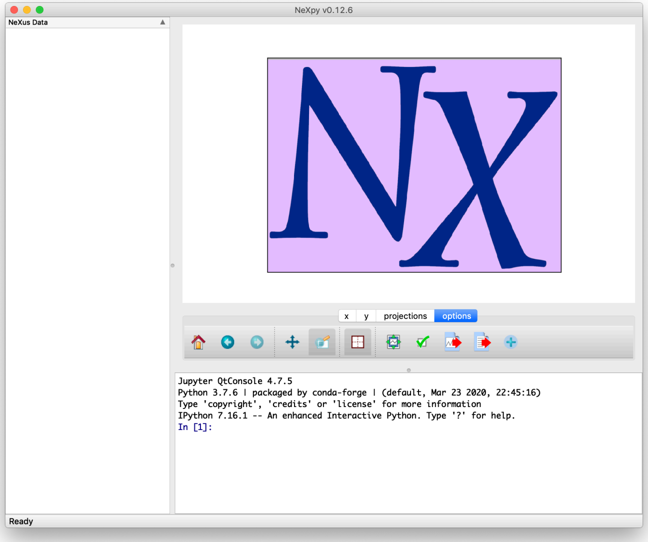
- The API has two advantages over using h5py directly.
  1. It ensures compliance with NeXus design rules without requiring any prior knowledge.
  2. It is much more concise in reading and writing NeXus data.
- As an example, the following code uses 'nexusformat':
  - `>>> NXdata(z, (y,x)).save('data.h5')`
- Here is the same code written in h5py:

```
>>> NXdata(z, (y,x)).save('data.h5')
>>> f = h5py.File('data.h5','r+')
>>> f.create_group('entry')
>>> f['entry'].attrs['NX_class'] = 'NXentry'
>>> f['entry'].create_group('data')
>>> f['entry/data'].attrs['NX_class'] = 'NXdata'
>>> f['entry/data'].attrs['signal'] = 'z'
>>> f['entry/data'].attrs['axes'] = ['y', 'x']
>>> f['entry/data'].create_dataset('x', data=x)
>>> f['entry/data'].create_dataset('y', data=y)
>>> f['entry/data'].create_dataset('z', data=z)
>>> f.close()
```

NeXpy - a GUI interface to NeXus

Restoring control over our own data

- At several major neutron and x-ray facilities, powerful computational frameworks have been developed to handle data reduction.
  - *e.g.*, Mantid, DIALS.



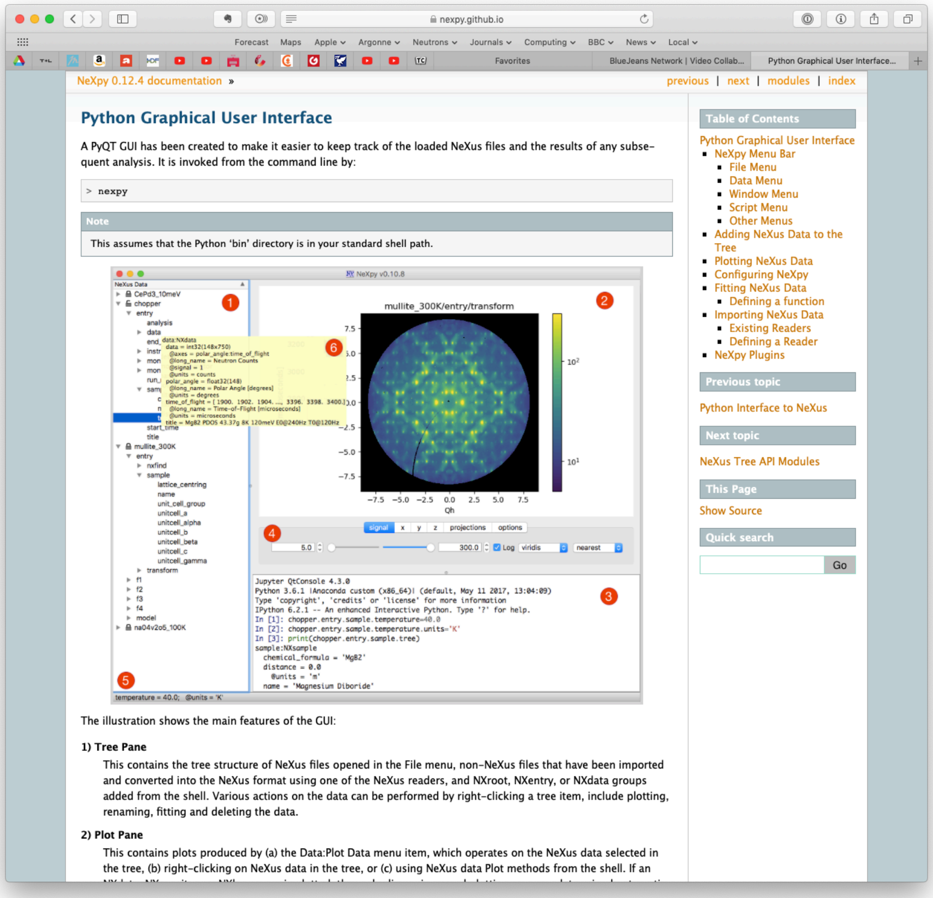
```
Jupyter QtConsole 4.7.5
Python 3.7.6 | packaged by conda-forge | (default, Mar 23 2020, 22:45:16)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.
In [1]:
```

- They are important for data reduction, but may be too complex for data analysis.
  - They can have a steep learning curve.
  - There is a danger in becoming too dependent on facility expertise.
- NeXpy is designed to make it easy to 'play' with the data (even 'Big Data').
  - Easy to inspect, visualize, manipulate, and fit the data.
  - Easy to compare data from multiple experiments and techniques.
  - Easy to develop new algorithms and modes of analysis.

INSTALLING NeXpy

https://nexpy.github.io/nexpy/

- NeXpy is a pure Python package.
- There are multiple ways to install it.
  - conda install -c conda-forge nexpy
  - pip install nexpy
  - git clone https://github.com/nexpy/nexpy.git
- Dependencies:
  - PyQt (PyQt5 or PySide2)
  - IPython
  - Matplotlib
  - h5py
  - nexusformat


nexpy.github.io
Forecast
Maps
Apple
Argonne
Neutrons
Journals
Computing
BBC
News
Local
Favorites
BlueJeans Network | Video Collab...
Python Graphical User Interface...
NeXpy 0.12.4 documentation »
previous | next | modules | index

Python Graphical User Interface

A PyQT GUI has been created to make it easier to keep track of the loaded NeXus files and the results of any subsequent analysis. It is invoked from the command line by:

```
> nexpy
```

Note

This assumes that the Python 'bin' directory is in your standard shell path.

```
Jupyter QtConsole 4.3.0
Python 3.6.1 |Anaconda custom (x86_64)| (default, May 11 2017, 13:04:09)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.
In [1]: chopper.entry.sample.temperature=40.0
In [2]: chopper.entry.sample.temperature.units='K'
In [3]: print(chopper.entry.sample.tree)
sample:NXsample
  chemical_formula = 'MgB2'
  distance = 0.0
    @units = 'm'
  name = 'Magnesium Diboride'
```

temperature = 40.0;   @units = 'K'

The illustration shows the main features of the GUI:

1) Tree Pane

This contains the tree structure of NeXus files opened in the File menu, non-NeXus files that have been imported and converted into the NeXus format using one of the NeXus readers, and NXroot, NXentry, or NXdata groups added from the shell. Various actions on the data can be performed by right-clicking a tree item, include plotting, renaming, fitting and deleting the data.

2) Plot Pane

This contains plots produced by (a) the Data:Plot Data menu item, which operates on the NeXus data selected in the tree, (b) right-clicking on NeXus data in the tree, or (c) using NeXus data Plot methods from the shell. If an

Previous topic

Python Interface to NeXus

Next topic

NeXus Tree API Modules

This Page

Show Source

Quick search

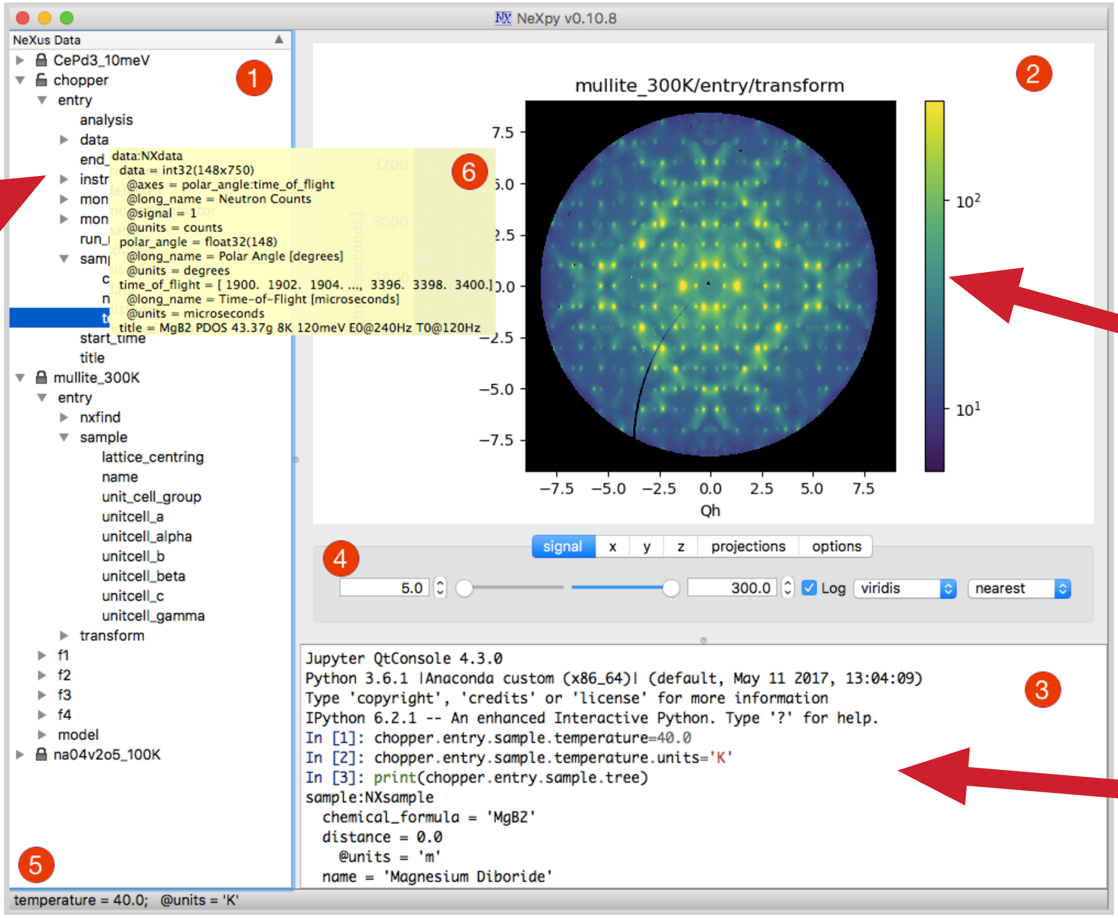
Go

ANATOMY OF NeXpy

1) Tree Pane

2) Plot Pane

3) Shell Pane



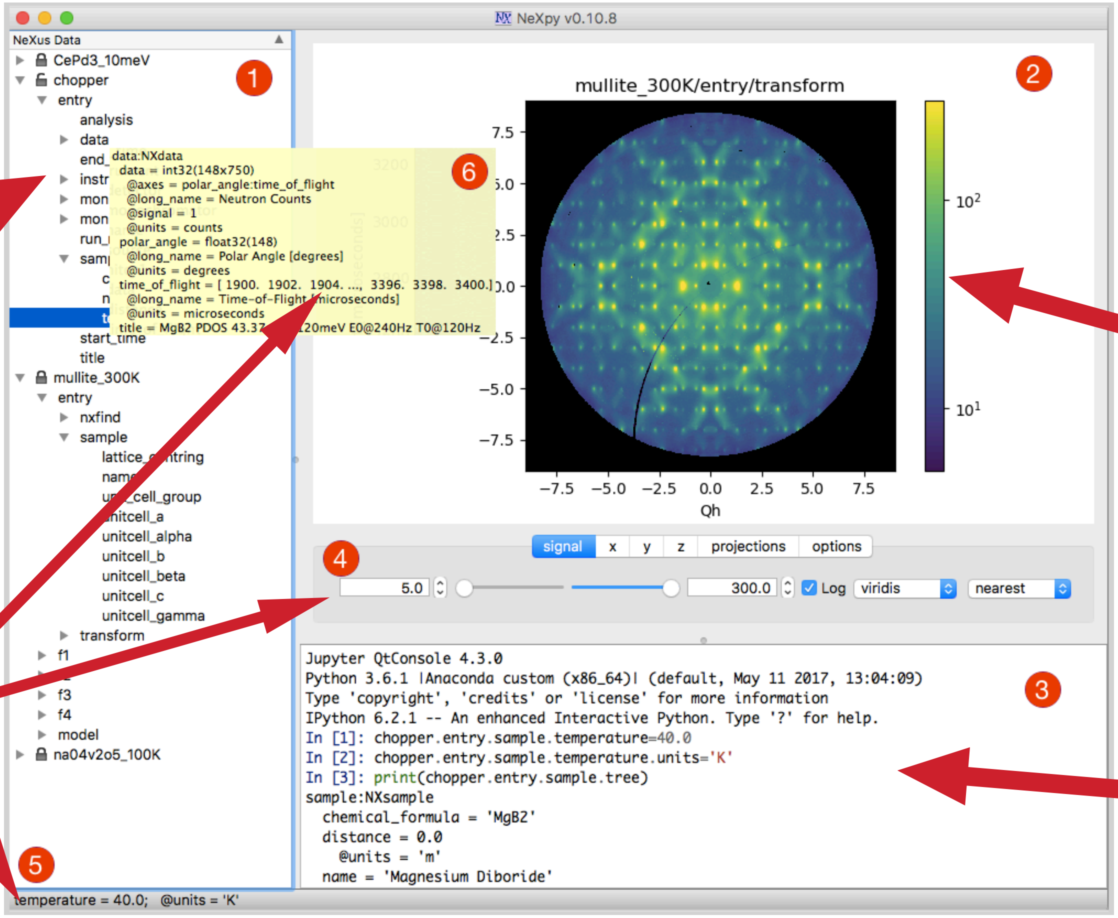
```
▶ CePd3_10meV
▼ chopper
  ▼ entry
      analysis
    ▶ data
      end_
    ▶ instr
    ▶ mon
    ▶ mon
      run_
    ▼ samp
        c
        n
        t
      start_time
      title
▼ mullite_300K
  ▼ entry
    ▶ nxfind
    ▼ sample
        lattice_centring
        name
        unit_cell_group
        unitcell_a
        unitcell_alpha
        unitcell_b
        unitcell_beta
        unitcell_c
        unitcell_gamma
    ▶ transform
  ▶ f1
  ▶ f2
  ▶ f3
  ▶ f4
  ▶ model
▶ na04v2o5_100K
```

```
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
```

```
Jupyter QtConsole 4.3.0
Python 3.6.1 |Anaconda custom (x86_64)| (default, May 11 2017, 13:04:09)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.
In [1]: chopper.entry.sample.temperature=40.0
In [2]: chopper.entry.sample.temperature.units='K'
In [3]: print(chopper.entry.sample.tree)
sample:NXsample
  chemical_formula = 'MgB2'
  distance = 0.0
    @units = 'm'
  name = 'Magnesium Diboride'
```

```
temperature = 40.0;   @units = 'K'
```

ANATOMY OF NeXpy



```
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 43.37 120meV E0@240Hz T0@120Hz
```

```
Jupyter QtConsole 4.3.0
Python 3.6.1 |Anaconda custom (x86_64)| (default, May 11 2017, 13:04:09)
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.
In [1]: chopper.entry.sample.temperature=40.0
In [2]: chopper.entry.sample.temperature.units='K'
In [3]: print(chopper.entry.sample.tree)
sample:NXsample
  chemical_formula = 'MgB2'
  distance = 0.0
    @units = 'm'
  name = 'Magnesium Diboride'
```

```
temperature = 40.0;   @units = 'K'
```

1) Tree Pane

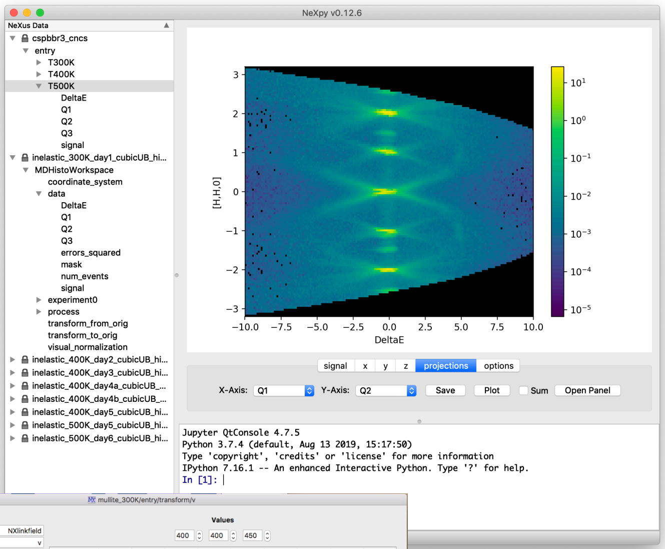
2) Plot Pane

3) Shell Pane

4) Axis Panels
5) Status Bar
6) Tooltips

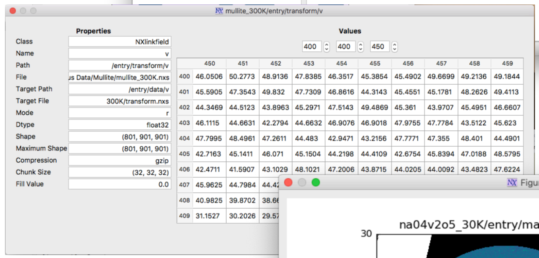
FEATURES OF THE NeXpy GUI

- NeXus data can be directly loaded into the tree using an Open File dialog.
- The GUI allows data to be viewed and manipulated:
  - *e.g*., plotted, viewed in a table, created, deleted, renamed, copied, and pasted.
- New NeXus data can be created, copied, and saved to a file.
- All groups and fields in the tree are accessible from the command line of the IPython shell, with all changes updated in the tree.
- Panels facilitate comparisons of data from multiple files.
  - Projection, Limits, Scan, and Fit Panels
- Specialized functionality can be implemented using a plugin architecture.
- As a bonus, NeXpy provides convenient GUI access to special Matplotlib features.
  - Skewed axes
  - Symmetric color plots
  - Smoothing in 1D and 2D
  - Reordering legends



```
cspbbr3_cncs
  entry
    T300K
    T400K
    T500K
      DeltaE
      Q1
      Q2
      Q3
      signal
inelastic_300K_day1_cubicUB_hi...
  MDHistoWorkspace
    coordinate_system
    data
      DeltaE
      Q1
      Q2
      Q3
      errors_squared
      mask
      num_events
      signal
    experiment0
    process
    transform_from_orig
    transform_to_orig
    visual_normalization
inelastic_400K_day2_cubicUB_hi...
inelastic_400K_day3_cubicUB_hi...
inelastic_400K_day4a_cubicUB_...
inelastic_400K_day4b_cubicUB_...
inelastic_400K_day5_cubicUB_hi...
inelastic_500K_day5_cubicUB_hi...
inelastic_500K_day6_cubicUB_hi...
```

```
Jupyter QtConsole 4.7.5
Python 3.7.4 (default, Aug 13 2019, 15:17:50)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.
In [1]: 
```

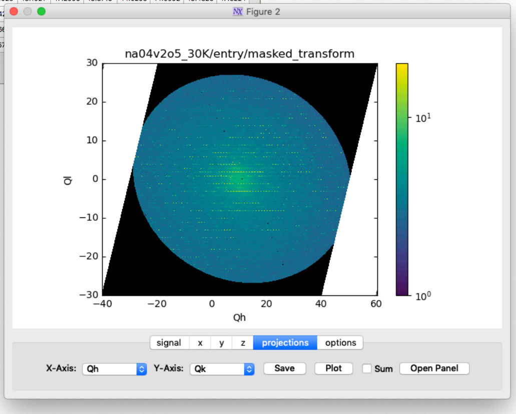

mullite_300K/entry/transform/v

Properties

Class	NXlinkfield
Name	v
Path	/entry/transform/v
File	us Data/Mullite/mullite_300K.nxs
Target Path	/entry/data/v
Target File	300K/transform.nxs
Mode	r
Dtype	float32
Shape	(801, 901, 901)
Maximum Shape	(801, 901, 901)
Compression	gzip
Chunk Size	(32, 32, 32)
Fill Value	0.0

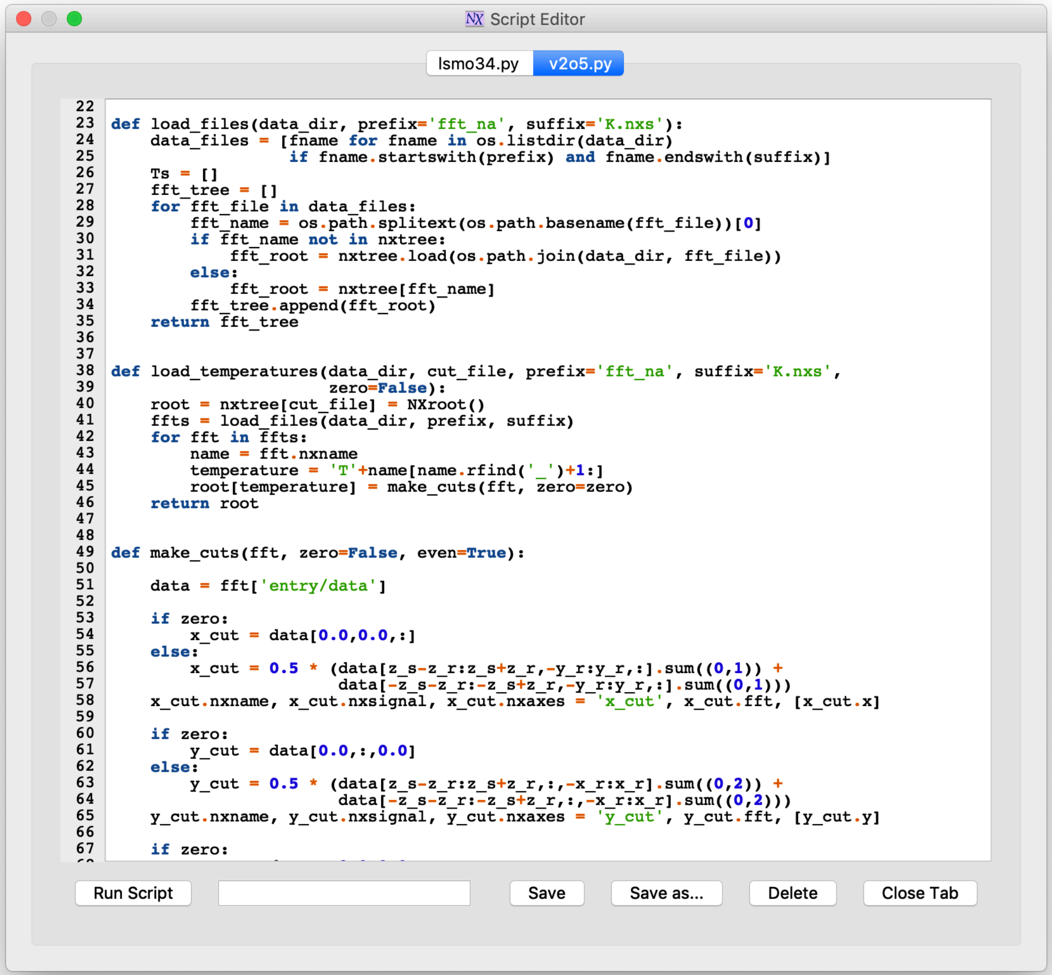
Values 400 400 450

	450	451	452	453	454	455	456	457	458	459
400	46.0506	50.2773	48.9136	47.8385	46.3517	45.3854	45.4902	49.6699	49.2136	49.1844
401	45.5905	47.3543	49.832	47.7309	46.8616	44.3143	45.4551	45.1781	48.2626	49.4113
402	44.3469	44.5123	43.8963	45.2971	47.5143	49.4869	45.361	43.9707	45.4951	46.6607
403	46.1115	44.6631	42.2794	44.6632	46.9076	46.9018	47.9755	47.7784	43.5122	45.623
404	47.7995	48.4961	47.2611	44.483	42.9471	43.2156	47.7771	47.355	48.401	44.4901
405	42.7163	45.1411	46.071	45.1504	44.2198	44.4109	42.6754	45.8394	47.0188	48.5795
406	42.4711	41.5907	43.1029	48.1021	47.2006	43.8715	44.0205	44.0092	43.4823	47.6224
407	45.9625	44.7984	44.4							
408	40.9825	39.8702	38.6							
409	31.1527	30.2026	29.5							



SCRIPT EDITOR

- NeXpy has a built-in editor for developing Python scripts.
- The code can be run immediately within the IPython shell.
  - For performing repetitive operations.
  - For developing complex algorithms.
- The script editor can be used to prototype new modes of data analysis.
  - *e.g.*, 3D-ΔPDF

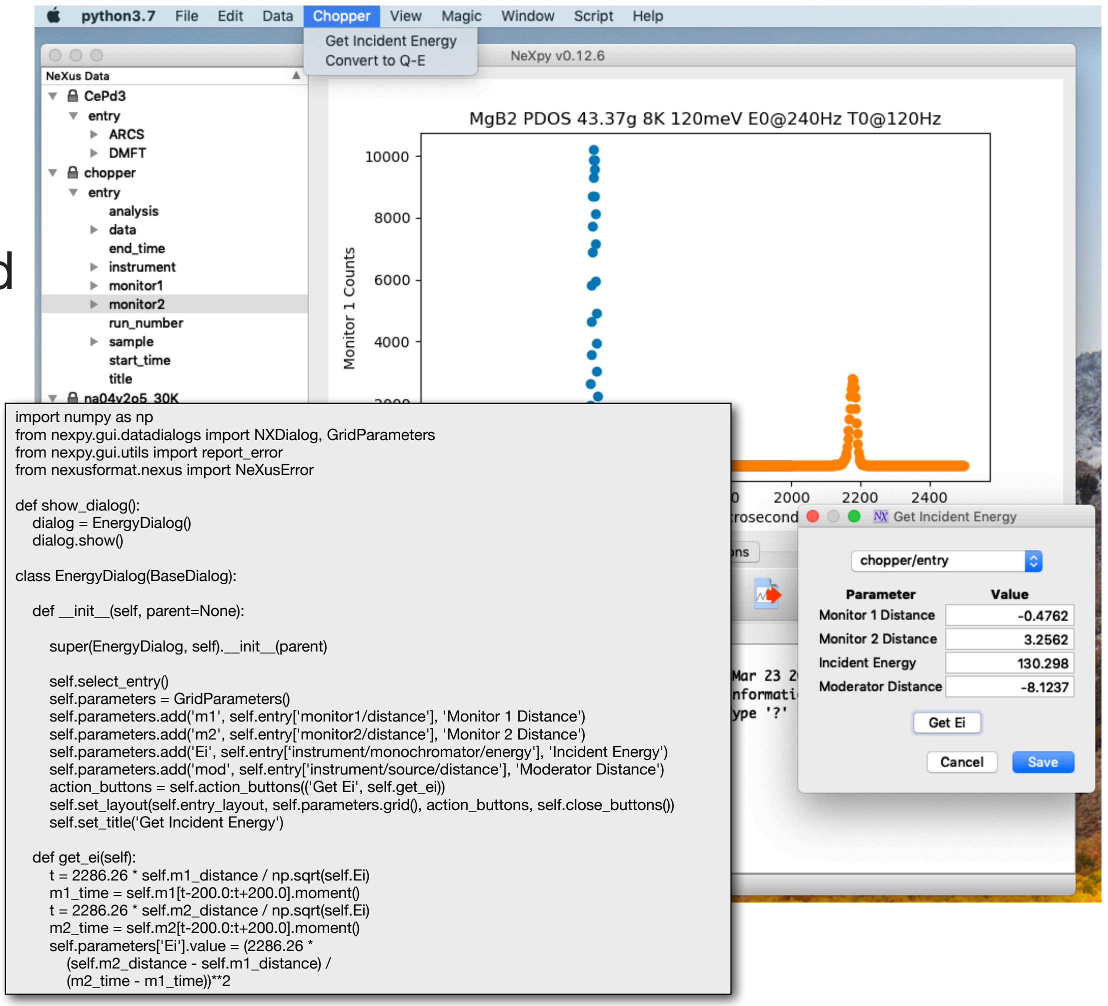


```python

def load_files(data_dir, prefix='fft_na', suffix='K.nxs'):
    data_files = [fname for fname in os.listdir(data_dir)
                  if fname.startswith(prefix) and fname.endswith(suffix)]
    Ts = []
    fft_tree = []
    for fft_file in data_files:
        fft_name = os.path.splitext(os.path.basename(fft_file))[0]
        if fft_name not in nxtree:
            fft_root = nxtree.load(os.path.join(data_dir, fft_file))
        else:
            fft_root = nxtree[fft_name]
        fft_tree.append(fft_root)
    return fft_tree


def load_temperatures(data_dir, cut_file, prefix='fft_na', suffix='K.nxs',
                      zero=False):
    root = nxtree[cut_file] = NXroot()
    ffts = load_files(data_dir, prefix, suffix)
    for fft in ffts:
        name = fft.nxname
        temperature = 'T'+name[name.rfind('_')+1:]
        root[temperature] = make_cuts(fft, zero=zero)
    return root


def make_cuts(fft, zero=False, even=True):

    data = fft['entry/data']

    if zero:
        x_cut = data[0.0,0.0,:]
    else:
        x_cut = 0.5 * (data[z_s-z_r:z_s+z_r,-y_r:y_r,:].sum((0,1)) +
                       data[-z_s-z_r:-z_s+z_r,-y_r:y_r,:].sum((0,1)))
    x_cut.nxname, x_cut.nxsignal, x_cut.nxaxes = 'x_cut', x_cut.fft, [x_cut.x]

    if zero:
        y_cut = data[0.0,:,0.0]
    else:
        y_cut = 0.5 * (data[z_s-z_r:z_s+z_r,:,-x_r:x_r].sum((0,2)) +
                       data[-z_s-z_r:-z_s+z_r,:,-x_r:x_r].sum((0,2)))
    y_cut.nxname, y_cut.nxsignal, y_cut.nxaxes = 'y_cut', y_cut.fft, [y_cut.y]

    if zero:
```

EXTENDING NeXpy

Plugin Architecture

- Additional menu items can be added to extend NeXpy functionality for specialist applications.
- A simplified widget library allows sophisticated GUIs to be developed without expert knowledge of PyQt.



```
import numpy as np
from nexpy.gui.datadialogs import NXDialog, GridParameters
from nexpy.gui.utils import report_error
from nexusformat.nexus import NeXusError

def show_dialog():
    dialog = EnergyDialog()
    dialog.show()

class EnergyDialog(BaseDialog):

    def __init__(self, parent=None):

        super(EnergyDialog, self).__init__(parent)

        self.select_entry()
        self.parameters = GridParameters()
        self.parameters.add('m1', self.entry['monitor1/distance'], 'Monitor 1 Distance')
        self.parameters.add('m2', self.entry['monitor2/distance'], 'Monitor 2 Distance')
        self.parameters.add('Ei', self.entry['instrument/monochromator/energy'], 'Incident Energy')
        self.parameters.add('mod', self.entry['instrument/source/distance'], 'Moderator Distance')
        action_buttons = self.action_buttons(('Get Ei', self.get_ei))
        self.set_layout(self.entry_layout, self.parameters.grid(), action_buttons, self.close_buttons())
        self.set_title('Get Incident Energy')

    def get_ei(self):
        t = 2286.26 * self.m1_distance / np.sqrt(self.Ei)
        m1_time = self.m1[t-200.0:t+200.0].moment()
        t = 2286.26 * self.m2_distance / np.sqrt(self.Ei)
        m2_time = self.m2[t-200.0:t+200.0].moment()
        self.parameters['Ei'].value = (2286.26 *
            (self.m2_distance - self.m1_distance) /
            (m2_time - m1_time))**2
```

Parameter	Value
Monitor 1 Distance	-0.4762
Monitor 2 Distance	3.2562
Incident Energy	130.298
Moderator Distance	-8.1237

SUMMARY

https://www.nexusformat.org

- NeXus is the only data format that is widely utilized in neutron and x-ray scattering facilities.
- It is maintained by a well-organized (though part-time) international team (NIAC).
- NeXus could, I believe, impact scientific users much more than it now does.
- A Python API provides a convenient interface to NeXus files
  - Reducing the dependency on facility computing support for data analysis.
  - Providing the means to test-drive new ideas.
- If ORSO discovers any issues with NeXus, please raise them with NIAC.