

File Formats for X-ray Absorption Spectroscopy and X-ray Microprobe Data

Matthew Newville

Consortium for Advanced Radiation Sciences
University of Chicago

January 12, 2010

Acknowledgments:

Bruce Ravel NIST, NSLS
Ken McIvor, Carlo Segre IIT, APS

Funding from NSF and DOE

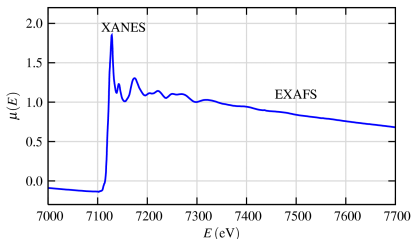
What Is XAFS?

X-ray Absorption Fine-Structure (XAFS) is the modulation of the x-ray absorption coefficient at energies near and above the binding energy for a core-level electron. XAFS is broken into 2 regimes:

XANES X-ray Absorption Near-Edge Spectroscopy
EXAFS Extended X-ray Absorption Fine-Structure

XAFS contains information about an element's chemical state (XANES) and local atomic coordination (EXAFS).

Fe K-edge XAFS for FeO:

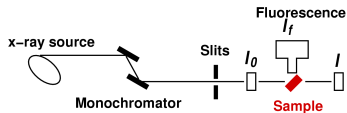


XAFS Characteristics:

- local atomic coordination
- chemical / oxidation state
- applies to any element
- works at low concentrations
- minimal sample requirements

XAFS: x-ray absorption fine-structure spectroscopy

XAFS is a precise measurement of the x-ray absorption coefficient $\mu(E)$:



Transmission what is absorbed by the sample:

$$I = I_0 e^{-\mu(E)t}$$

Fluorescence what is emitted from the sample because of absorption:

$$\mu(E) \propto I_f / I_0$$

- Energy is needed at a resolution of $\sim 10^{-4}$.
- Energy usually involves a mechanical motion: Slow.
- Spectra consist of 100 to 1000 Energy values.
- $\mu(E)$ is typically needed to $\sim 10^{-3}$, sometimes better.
- Spectra can take 1 to 10,000 seconds to acquire.
- Can take much longer to analyze data.

Data rates are slow (Mb/hour) and data files are small.

XAFS Data Files

XAFS data files are almost always some sort of ASCII column file:

Example XAFS Data File:

```
#sample:    Cu foil Room Temperature
#notes:     data from cu_foil.001
#detectors: IO=N2 10cm; I1=N2 10cm
#beamline:  APS 13ID, vert slits = 0.3mm
#mono:      Si(111) focussed, detuned 50
#date:      Tue Jun 20 14:27:31 2006
#npts:      418
#-----
#  energy      xmu      i0
8879.0000      -1.3276930    117383.70
8889.0000      -1.3312944    117185.70
8899.0000      -1.3336289    117058.70
8909.0000      -1.3305114    117276.70
8919.0000      -1.3385381    117332.70
8929.0000      -1.3403222    117332.70
8939.0000      -1.3419374    117756.70
8949.0000      -1.3353518    117199.70
8959.0000      -1.3394162    117458.70
8959.5000      -1.3390900    118720.70
8960.0000      -1.3386739    119008.70
8960.5000      -1.3382262    120520.70
```

Some sort of Header – sometimes readable, sometimes parsable.
crude metadata

Columns of numbers in ASCII text: Energy is typically one of the first few columns, usually in eV or keV

μ may be stored or just raw values for I_0 , I_1 , and/or I_f from measurement chain.

There are **many** variations for ASCII column files: $\gtrsim 1$ per beamline!
Client applications are expected to understand the columns.

OK for working with 1 spectra at a time, Not so good otherwise.

The Case for Simple ASCII Data Files

ASCII Column Files have some clear advantages for such small data sets:

- 1 Readable by humans.
- 2 Editable by humans.
- 3 Will be readable (and editable) in 25 years.
- 4 Readable by all standard apps: (Excel, Origin, Gnuplot, etc).
- 5 Readable by current XAFS analysis programs.
- 6 Readable by any program environment.
- 7 History: Lots of existing data, some of it still useful!

For small, 1-D data sets that are analyzed one at a time, ASCII files are fine. When asked, most XAFS users and beamline scientists **prefer** ASCII data files.

What problems could a standard file format solve for the XAFS community?

Data Sharing and Organization.

Why Do We Want a Standard Format?

A standard data format allows easier, more automated data sharing between beamlines, and analysis applications.

For XAFS alone, this could be done with *Tagged ASCII Files*.
Such a format specification exists:

Proposed Standard XAFS File:

```
# IXASIF/1.0 MX/2.0
# Crystal: Si 111
# Beamline: APS 10ID
# Mirrors: single harmonic rejection mirror
# Start-time 2005-03-08 20:08:57
# Edge-energy: 7112.00
# Mu-transmission: ln($2/$3)
# Mu-reference: ln($3/$5)
# MX-Offsets: 11408.00 11328.00 13200.00 10774.00
# MX-Gains: 8.00 7.00 7.00 9.00
#---
# Fe K-edge, Lepidocrocite powder on kapton tape, RT
# 4 layers of tape
# exafs, 20 invang
#---
# energy      mcs3      mcs4      mcs6      mcs5
6899.9609    48120    19430    2250    54540
6900.1421    48390    19540    2260    54860
6900.5449    48520    19610    2250    55110
6900.9678    48930    19780    2280    55650
6901.3806    48460    19590    2250    55110
```

Header Values specified as
KEY: VALUE \n

Parsable (standard grammar).
Better metadata.

Numerical Data still text in columns.

Not too bad, but some issues unresolved.
energy units?

This could standardize data files, and
make it easier to share data.

It does not help organize spectra.

Standard Data Formats and APIs

Even using this simple ASCII Format requires an API that has to be

- 1 available in multiple languages
- 2 written, tested, documented.
- 3 used by applications.

But if we need an API, the format doesn't matter too much.

Might as well use HDF5 (or something else ...).

Would HDF5 be the first choice for storing XAFS data?

Frankly, no. But it doesn't get in the way.

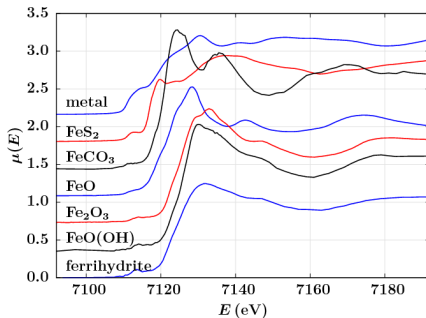
If HDF5 brings together x-ray microprobe, diffraction, and allows hyperspectral data to be viewed as XAFS and/or MCA maps, it's worth it.

Does HDF5 address any issues XAFS users do care about?

XAFS Data Libraries

XAFS Users often collect and use sets of related spectra:

Fe XAFS for a few compounds:



Used for Linear Algebra Analysis (PCA, NNMA, etc)

Spectra may be from different sources.

Spectra need to be aligned in Energy.

Libraries of Spectra are very valuable and should be preserved. Users are always looking for standard spectra.

Can XAFS Apps using a richer File Format help create, use, and manage these spectral libraries?

Can existing web archives of XAFS data allow such libraries to be shared?

I hope so.

Metadata: Sample Information

To organize spectra into Libraries, we need better Metadata:

- sample provenance – materials, preparation.
- description of related measurements (XRD data? photos?)
- measurement conditions – beamline information and settings.

All x-ray measurements would benefit from such information!

Do we need to think about an entire experiment as a single set of data, or must each scan and map stand on it's own?

How do we relate measurements made at different conditions?

Fe XAFS scan 1 was made at pixel X1,Y1 of XRF Map #3 on Sample A.
Fe XAFS scan 2 was made at pixel X2,Y2 of XRF Map #1 on Sample B.

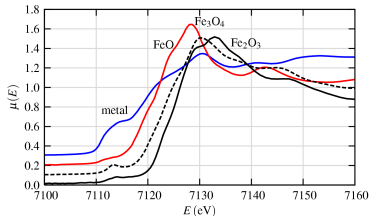
It's starting to look like we need a database.

Should we barcode all samples?

Metadata Issue #2: Energy Calibration

Libraries of XAFS spectra have a sticky (illustrative) issue: *Energy Calibration*.

More Fe XAFS!



Energy has high precision (10^{-5} or better)

Energy is rarely calibrated that accurately. Shifts of several eV are common.

We need energy to high relative accuracy.

Energy is non-linear in Mono Angle!

$$hc/E = 2d\sin(\theta)$$

Most calibration offsets are in θ_0 (or in d).

Recalibrating with a constant Energy shift is OK if the shift is small enough (few eV), but larger shifts can introduce errors.

Recalibrating data properly after the fact requires knowing what d (and hc !) were used, and is still a non-trivial procedure.

Can your metadata do this?

Metadata Issue #3: Fluorescence Detector Deadtime

For energy-analyzed XRF detectors, there is a finite deadtime due to multiple photons that cannot be separately analyzed:

$$\text{OCR} = \text{ICR}e^{-\text{ICR}\tau}$$

ICR Input Count Rate: photons that enter detector

OCR Output Count Rate: photons energy analyzed.

τ detector dead time, typically a few μs .

Fluorescence signals for XAFS (or maps) need to be corrected by the factor ICR/OCR. Typically, ICR and OCR are reported by the detector.

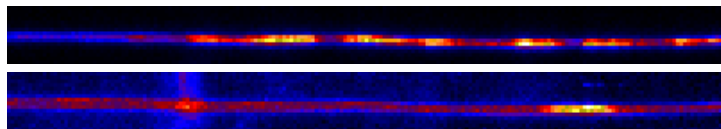
τ (determined separately) could be stored as metadata.

Or ICR and OCR can be stored as data!

Or LiveTime and RealTime can be stored!

My Experience with HDF5, Python, and IDL

Converting GSE microprobe data to HDF5



P K_{α}

Fe K_{α}

In October (2009), A GSECARS microprobe user wanted to use Stefan Vogt's MAPS program. MAPS can't read the XRF maps from GSECARS ...

Hey, Let's use HDF5 to exchange data! How hard can it be?

OK, I will tell you.

Convert maps (quad Vortex detector) in ASCII data for ~ 60 maps and ~ 300 1-d scans. Scans included ROI data and Full XRF spectra.

- Write data to HDF5 with Python and h5py. 1 HDF5 file per Map / Scan.
- Test reading with Python and IDL.
- Play with HDF5 Schema for x-ray microprobe data.

I looked at existing HDF5 schema from the MAHID web page, but made my own to get a feel for how data can be organized with HDF5.

A simple microprobe schema for HDF5

data: Root group. **Attributes:** Title, Collection Time, Version, Beamline.

scan: Scan data, including ROI data.

Attributes: dimension, start_time, stop_time, scan_prefix

Datasets: x, y, pos, pos_desc, pos_addr, dt_factor, det_desc, det_addr, det, det_corr, sums_names, sums_list, sums, sums_corr, user_titles

environ: Environmental data (expected to not change during scan).

Datasets: addr, desc, val (all string arrays)

full_xrf: full XRF spectra per pixel from individual detectors.

Attributes: ndetectors, dimension, nmca

Datasets: data, data_corr, energies, roi_hi_limit, roi_lo_limit, roi_labels

merged_xrf: XRF spectra merged to a single spectra per pixel.

Attributes: dimension, nmca

Datasets: data, data_corr, energies

Favors redundant data over non-trivial processing (merging, summing ROIs, saving deadtime corrected data) in the client.

I'm not at all committed to this schema.

Lesson #1: Deadtime

I discussed detector deadtime earlier ... OK, how **should** we store this?

$$\text{OCR} = \text{ICR}e^{-\text{ICR}\tau}$$

The most efficient option would be to store τ : one value per detector.

That would require the client library to do the correction.

So of course, I didn't do that.

I could have saved ICR and OCR arrays for **each pixel in the map!**

Instead, I saved ICR/OCR at each pixel: the deadtime correction factor ($\gtrsim 1$).

And I saved deadtime-corrected maps.

Space inefficient, but much easier to use.

Lesson #2 (Metadata Issue #4): x-ray intensities

We don't actually measure x-ray intensity. We use an ion chamber in which:

- 1 an inert gas absorbs some (energy dependent!) fraction of the x-rays.
- 2 the current generated from the ionized gas is amplified.
- 3 the amplified signal is then integrated for some time to give **counts**.

So: how can we compare measurements from different times or beamlines?

Either: Know all of the following (and how to use them!):

- 1 composition of gas in ion chamber.
- 2 Active length of ion chamber
- 3 Gain settings on amplifier
- 4 Dark currents: were they already subtracted?
- 5 integration factor (how many nA gives 1 count).
- 6 integration time

We also need the absorption coefficients for the absorbing gas(es).

Tables for these vary by 10% (especially at high energies), so the absorption coefficient values used should be given.

Or: Don't compare intensities between different times and beamlines.

Lesson #3: HDF5 v. HDF5

HDF5 \neq HDF5: : v 1.6.* in IDL 6 and 7.0 is not compatible with the latest “current version” v1.8.*.

Solutions: use IDL 7.1, use “compatibility mode” with v 1.6.*

Comments: HDFGroup says they support v1.6.* **and** v1.8.*. Many 3rd party apps use v1.6.* (Matlab, Octave, Ubuntu, etc.)

HDFView: : The **current, supported** HDFView program from HDFGroup alters *internal version info* in HDF5 files. IDL 7.0 and earlier **crash** on h5f_open!

Datasets, attributes are **not changed**, but checksums are changed. Yikes.

Solutions: Use IDL 7.1. Don't use HDFView?? Don't rely on checksums??

Comments: HDFView is built against v 1.8.2, and this is a “known bug”. For 6 months. HDFView has still not been updated.

There is NO WAY to investigate version information from the HDF5 API.

Mixing data from different HDF5 versions (1.6, 1.8, 1.10) in a single file is a *feature*: It is easy to do, and there is no way to detect this!

Lesson #3: HDF5 versions

Applications using the HDF5 API **cannot detect** if a dataset from a newer version has been added, and this may break applications.

IDL with v 1.6.* crashes with HDF some files written with v 1.8.*.

```
from: Quincey Koziol <koziol@hdfgroup.org>  
date: Tue, Dec 8, 2009 at 2:35 PM  
  
...if your version of h5py still linked  
against 1.8.x after the 1.10.x releases  
were out and you received a file with format  
changes from that release, you will have  
similar problems to the IDL situation.
```

In fairness, **Forward Compatibility** is very hard.

For long term archiving and exchange of data between multiple client applications, this concerns me.

Describing and Saving an Entire Experiment

There are many types of data for an x-ray beamline: XRF spectra, XRD detector images, Video images, sample information, and “External Environmental” Data (ring current, room temperature).

I want my users to walk away with all this data, tied together.

Can HDF5 do this?

Perhaps.

A relational database can definitely do this.

Relational Database for Metadata and Data Collection

I want to see a relational database at the heart of data acquisition.

Relational Databases (MySQL, Postgres, Oracle) are robust data storage and retrieval systems, and solve many issues for modern data acquisition:

- Multi-client access to database during experiment. Remote Access.
- Security can be built in.
- Automated logging.
- After experiment, DB can be converted to single file database (SQLite).

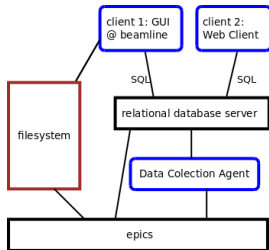
What would go into the database for acquisition?

- Configuration of detectors, monochromators, motors, other instruments.
- Definitions for macro code (Python?) that users are allowed to run.
- Queue of Macro commands to be run.
- State information about what is being done, what has been done.

Data could go to database, or to file system with references in the database.

RDBMS for Data Acquisition

RDBMS For DAQ



Client sends commands (macros) to DB via SQL over the network. Commands are Queued.

Collection Agent sees next command, launches an (Epics) process or thread, may wait for it to finish, looks for next command.

Clients can read status, data from DB or file system.

Epics processes can write to file system or database: HDF5, TIFF, JPEG, ASCII,...

User takes home SQLite version of the DB: the whole experiment.

Data Analysis Programs use SQL to query this, can add analysis information.

Data sets could be extracted to other databases or applications.

RDBMS or HDF5?

Metadata may map better to a relational model than a hierarchical one:

- Where on Map 1 was XAFS Scan A taken?
- Was there a sudden change in Room Temperature (Ring Current? Mirror Temperature?) during this map? If so, where?
- When was XAFS Scan for FeO collected?
- Interpolate XAFS Scan #3 of FeSO₄ (from another beamline, last year) onto the same energy grid as the FeO scan.

Can we use a RDBMS *in addition to* HDF5?