Larch: An Analysis Package for XAFS and Related Spectroscopies

Matthew Newville
Center for Advanced Radiation Studies, University of Chicago, Building 434A, Argonne National Laboratory, Argonne, IL 60439, USA
E-mail: newville@cars.uchicago.edu

Abstract. LARCH, a package of analysis tools for XAFS and related spectroscopies is presented. A complete rewrite of the IFEFFIT package, the initial release of LARCH preserves the core XAFS analysis procedures such as normalization, background subtraction, Fourier transforms, fitting of XANES spectra, and fitting of experimental spectra to a sum of FEFF Paths, with few algorithmic changes made in comparison to IFEFFIT. LARCH is written using Python and its packages for scientific programming, which gives significant improvements over IFEFFIT in the ability to handle multi-dimensional and large data sets, write complex analysis scripts, visualize data, add new functionality, and customize existing capabilities. Like the earlier version, LARCH can run from an interactive command line or in batch-mode, but LARCH can also be run as a server and accessed from clients using standard inter-process communication techniques available in a variety of computer languages. LARCH is freely available under an open source license. Examples of using LARCH are shown, future directions for development are discussed, and collaborations for adding new capabilities are actively sought.

1. Introduction
The basic algorithms for XAFS analysis of extracting $\chi(k)$ from measured absorption spectra and determining structural information from $\chi(k)$ are well-established and mature. While there may be room for improvement in the algorithms, most existing analysis packages[1, 2, 3, 4, 5, 6, 7] vary more in details and in emphasis on particular features or user interface than on the core algorithms used. As the field matures and synchrotron beamlines become more sophisticated, XAFS data is coming at much higher rates, either as traditional spectra measured at higher rates[8], or by using fast area detectors[9], or collected in fluorescence mode using energy dispersion onto area detectors[10]. In addition, XAFS is increasingly being embedded in complimentary measurements, such 3-dimensional, full field imaging datasets[11], x-ray fluorescence mapping, elastic x-ray scattering, and inelastic resonant x-ray scattering. The existing software tools for XAFS analysis are generally found in specialized, single-purpose packages designed to work with individual $\mu(E)$ spectra, and are unable to seamlessly deal with many of the forms of XAFS data now being collected.

The IFEFFIT package[12] is a widely used XAFS analysis package, with its success due to a combination of being freely available, having high quality algorithms and a flexible interface separating core analysis engine from graphical user interface, the existence and support for easy-to-use GUI applications such as ATHENA and DEMETER[13], SIXPACK[14], and PRESTOPRONTO[15], an active mailing list, and frequent training courses. Now over a decade old,
IFEFFIT has many limitations that make it unable to meet the growing demands for quantity and type of data detailed above that a modern XAFS analysis package should have. The most severe of these limitations—only a fixed and rather small amount of computer memory could be used—resulted from IFEFFIT (like its predecessors FEFFIT, AUTOBK, and FEFF) being implemented in standard FORTRAN-77. Other limitations, such as a weak built-in scripting language and poor quality graphics, also resulted from using FORTRAN-77. Additionally, the code base of IFEFFIT was complex, making maintenance, minor modifications to existing algorithms, and the addition of new features difficult. Thus, to meet the dynamic needs of modern XAFS data, including the ability to analyze hundreds to thousands of spectra or to analyze spectra embedded in larger data sets, a complete reimplementation of IFEFFIT was undertaken. In this report, LARCH, a replacement for IFEFFIT is described. The fundamental change is that LARCH is written entirely in Python, a popular, cross-platform, and open source interpreted language known for its readability and clarity, and with a very large set of built-in libraries and efficient, well-tested libraries for scientific computing.

2. LARCH: A replacement for IFEFFIT

LARCH maintains the basic model of an interactive command-line session that was an important feature of IFEFFIT. This approach preserves data within a session, but allows for dynamic and open-ended analysis routes, so that the user can re-visit analysis steps at will and mix analysis of multiple spectra while also maintaining a history of processing steps. This interactivity does not mean one is restricted to typing at a command-line shell interface, however. Scripts can be used for batch processing, or more complex modules and programs be written.

Though IFEFFIT also featured an interactive session, weaknesses in its implementation made it difficult to work well with many sets of data. The pre-defined limits on memory mentioned above were part of this. Though allowing for thousands of separate arrays, analyzing more than a few hundred data sets in a single session was not possible. In addition, IFEFFIT did not support features necessary for moderately complex scripts such as conditionals and loops, supported only one dimensional arrays, and had no multi-dimensional or flexible data structures. Finally, IFEFFIT used a single, flat “name space” so that important quantities, such as $E_0$, could have only one value at a time. Together, these limitations made interactive sessions and scripting more challenging than they should be. Wrappers in languages such as Perl[16] or Python[17], which created and ran a single IFEFFIT session by sending commands constructed in the wrapping language, were needed for any sustained analysis session or Graphical User interface, such as ATHENA[13] or SIXPACK[14].

LARCH overcomes the limitations described above and provides a much richer scripting language. By using Python, limits on memory are not set by the program, but by the host computer and operating system. The well-tested and supported NumPy[18] and SciPy[19] modules provide efficient and high-quality numerical arrays and interfaces to basic numerical libraries for procedures such as Fourier transforms, spline interpolation, linear algebra, and non-linear least squares optimization to Python, and are used throughout LARCH.

The text of one or more lines of LARCH code goes through a minimal translation step to convert it into Python code which is then compiled with Python’s own compiler tools into an abstract syntax tree that can be evaluated in a straightforward, if recursive, way. This gives LARCH a complete language, including conditional statements (if-then-else), loops (for and while), exception handling, function definitions, multi-dimensional arrays with complex slicing operations, while sparing the need to parse and support a fully-featured language. In the translation and interpretation steps, LARCH is able to modify some of the assumptions made in Python, such as how program symbols are retrieved and how code blocks are specified, making LARCH a dialect of Python that is specially tuned for scientific computing. Compared to Python, the LARCH dialect makes these changes:
Table 1. Core XAFS analysis functions in LARCH.

| function       | description                                           |
|---------------|-------------------------------------------------------|
| pre_edge()    | Find $E_0$, Pre-edge subtraction, normalization       |
| autobk()      | post-edge background subtraction                      |
| ftwindow()    | calculate a Fourier transform window function         |
| xafsfft()     | forward ($k \rightarrow R$) Fourier transform         |
| xafshift()    | reverse ($R \rightarrow k$) Fourier transform         |
| feff_path()   | Read and define a FEFF Path, with Parameters           |
| feffit_transform() | Define Fourier transform for feffit()          |
| feffit_dataset() | Define data set, Paths for feffit()              |
| feffit()      | Perform fit of a set of Paths to a set of data sets    |
| minimize()    | General purpose least-squares fit                     |

(i) LARCH uses code blocks for if, for, etc that end with endif, endfor, etc instead of using significant indentation level.

(ii) LARCH uses a more complex namespace for program variables and built-in functions, so that names for variables and functions are organized in a hierarchy, but can be easily accessed.

(iii) one cannot create new classes of objects inside LARCH.

Being a dialect of Python means that all variables and functions in LARCH are Python objects. Thus LARCH has basic data types such as integers, floats, complex numbers, and strings, and richer objects including multi-dimensional arrays of floats, heterogeneous lists of other object, and associative arrays. This is an important improvement over iFEFFIT. In addition, two important LARCH-specific data types are also added:

Groups: An empty container to hold other objects. Many LARCH functions work with Groups, either returning them or using them to put multiple, related outputs. In this way, an entire data set (raw intensities, $\mu(E)$, $\chi(k)$, $\chi(R)$, values for $E_0$, edge jump etc) can be contained in a single Group.

Parameters Because non-linear least-squares fits are an integral part of XAFS analysis, used for XANES fitting, background subtraction autobk and fitting of FEFF paths to $\chi(k)$ spectra, fitting parameters in LARCH are defined as Parameter. These can either be variable or fixed in a fit, or constrained to a mathematical expression of other Parameters.

Extending LARCH functionality is very easy. Since LARCH is so closely based on Python, any Python module can be imported and used directly from LARCH. For functionality that interacts more closely with the LARCH infrastructure (for example, creating Groups), a plug-in mechanism exists. Finally, like Python, LARCH is cross-platform. Once the required numpy and scipy libraries are installed, LARCH runs without any compilation step or changes on Windows, Mac OS X, and Linux.

3. Using LARCH

LARCH has built-in many numerical functions from numpy and scipy (sqrt, log, and a host of others) that work efficiently on entire arrays at once, without the need for explicit looping. In addition, LARCH provides functions for high-quality graphical plots for X-Y line plots and for 2-dimensional intensity maps, using the matplotlib[20] library. Of course, LARCH also adds several XAFS-specific functions for working with XAFS data. A partial list of these functions is given in Table 1.
dat = read_ascii('raw_xafs.dat', labels='en time i0 i1')
dat.mu = -ln(dat.i1/ dat.i0)

pre_edge(dat.en, dat.mu, group=dat)

autobk(dat.en, dat.mu, rbkg=1.2, kweight=2, group=dat)

xafsft(dat.k, dat.chi, kmin=3, kmax=16, dk=5,
       kweight=2, window='kaiser', group=dat)

newplot(dat.k, dat.chi*dat.k**2, win=1,
        xlabel='$k \, (\mathrm{\AA}^{-1})$ ',
        ylabel='$k^2 \chi(k) \, (\mathrm{\AA^{-2}})$')

plot(dat.k, dat.kwin, color='black', style='dashed')

newplot(dat.r, dat.chir_mag, win=2,
        xlabel='$R \, (\mathrm{\AA}) $ ',
        ylabel='$|\chi(R)| \, (\mathrm{\AA^{-3}})$')

plot(dat.r, dat.chir_re, win=2)

Figure 1. Example LARCH session, in which data is read from an ASCII file, \(E_0\) is computed, \(E_0\) and edge jump are found with `pre_edge()`, \(\mu_0\) and \(\chi(k)\) are found with `autobk()`, \(\chi(k)\) is transform to \(\chi(R)\) with `xafsft()`, and data arrays are plotted.

With many built-in functions, LARCH can be used as a high-level command-line shell. An example session to read raw XAFS data from an ASCII file, compute \(\mu(E)\), \(\chi(k)\), and \(\chi(R)\), and plot some of the results is shown in Figure 1, with the graphical outputs shown in Figure 2. Note that a subset of the \TeX\ typesetting system is understood by the matplotlib plotting library, and used here for axis labels. Though not shown, interactive zooming, printing, copying to system clipboard, and customization of plot properties such as line colors, symbols, legends, and labels are readily available to the user from the plot window.

3.1. Fitting XAFS Data

Of course, a primary need for XAFS analysis is to compare experimental to theoretical standards from first principles calculations such as those from FEFF\cite{21}. Continuing the use of a “Sum-of-Paths” approach, LARCH provides functions for defining FEFF Paths with Path Parameters such as \(E_0\), \(S_0^2\), \(\delta R\), and \(\sigma^2\), and refining these to match data\cite{22}, usually in \(R\)-space, but with some flexibility in “fitting space”. Using Parameters mentioned in section 2, there is great flexibility in how Path Parameters can be varied, and a full range of algebraic constraints can be applied, to build elaborate structural models for the refinements. In addition, simultaneous refinement of multiple data sets can be performed without much effort.

As with IFEFFIT, fit statistics, estimates of uncertainties in fitted variables, and correlations between variables are automatically determined. Interestingly, the basic fitting algorithm is the same\cite{23}, and very little systematic difference is seen between fits using IFEFFIT and LARCH. LARCH’s ability to put a set of Parameters into a Group allows sets of fitting variables for different fitting models to be isolated and more readily compared.
Figure 2. Plots of $k^2 \chi(k)$ (top) and $\chi(R)$ (bottom) from the LARCH session shown in Figure 1 (with data from Se K edge of CoSe. The images above are extracted directly from LARCH without further processing.

3.2. Efficiency Tests

While using Python and its numerical libraries makes LARCH flexible, easy to use, and easy to implement and maintain, Python is not generally considered fast for numerical processing, especially in comparison to FORTRAN. However, because IFEFFIT contained a poor scripting language within it and tried to manage a memory pool, much of its expected efficiency was lost. Thus, when strictly limited to core, non-graphical processes, LARCH can be faster than IFEFFIT. As a benchmark appropriate for QEXAFS work, a loop that goes through a set of raw data files, reading the file, running `autobk()` and `xafsft()` to extract $\chi(R)$, is shown in Figure 3. A similar script using IFEFFIT and its Python wrapper is more complex, limited in how many spectra can be preserved, and about $\sim 2.5\times$ slower than the LARCH script.
filelist = ['xafsscan.001', 'xafsscan.002', 'xafsscan.003']
save_groups = {}
for fname in filelist:
    dat = read_ascii(fname, labels='en time i0 i1')
    dat.mu = -ln(dat.i1/ dat.i0)
    autobk(dat.en, dat.mu, rbkg=1.2, kweight=2, group=dat)
    xafsft(dat.k, dat.chi, kmin=3, kmax=16, dk=5,
            kweight=2, window='kaiser', group=dat)
    save_groups[fname] = dat
endfor

Figure 3. Example of looping over a set of files, and running autobk(), and xafsft() for each set. The use of save_groups preserves the group used for each data set, so that it can be retrieved after the loop.

3.3. Remote Server Mode
Being able to access the functionality of LARCH as a calculation engine from other applications, such as GUIs, is an important priority. To facilitate this need, LARCH can be run as a server, with a unique session per server instance. This allows client applications to connect to one or more separate server processes either on the local machine or remote machines. Since each server is a separate process, they can easily make use of multiple cores on all platforms. Server-mode also allows multiple clients to share a single LARCH session if desired. Communication between client application and LARCH server uses the standard XML-RPC protocol, which is widely supported in a variety of languages (Perl, Python, C++, Java, .Net, PHP, javascript) without the need for a LARCH-specific wrapper per language.

3.4. Additional Features
Though not discussed in detail for lack of space, LARCH is not intended to be restricted to XAFS analysis. Because LARCH can use any Python module and is already using its basic scientific libraries, adding more functionality is straightforward, as will be discussed further below. Though early in its development, LARCH already has available functionality for the following tasks:

- accessing tabulated x-ray data for emission and absorption energies[24], elastic x-ray scattering, anomalous scattering factors[25] and other data.
- preliminary analysis and visualization of x-ray fluorescence spectra and x-ray fluorescence maps from x-ray microprobe measurements.
- reading, manipulating, and displaying TIFF images, used by many diffraction cameras.
- reading data from HDF5 data files, used for data storage at many synchrotrons.
- interacting with the EPICS experimental control system, used at many synchrotrons.

Some of these require additional python modules to be installed.

4. Conclusion and Future Directions
The initial release of LARCH described here should not be considered to be a final description of its capabilities. Indeed, the main effort so far has been to reproduce the functionality of IFEFFIT, so as to be a starting place for future developments in analysis algorithms for XAFS, XANES,
and related techniques, and to provide a proof-of-concept for additional functionality outside of XAFS analysis, including more complete x-ray fluorescence analysis and experimental control. Though any list of development details will become out-dated, a few possible improvements for XAFS analysis can be mentioned. Some of these exist in other analysis programs, but may not be widely supported and available. The most obvious development would be to add a full set of linear algebra techniques, including principle component analysis and target transformation.[15] Alternative background subtraction and normalization methods[26] could also be added. Another development would be to include a complete statistical treatment of $\mu(E)$ that is carried fully through the analysis, as shown recently.[27] Finally the application of wavelet transforms[28] or other transformations beyond the simple XAFS conventions of applying a $k$-weight and Fourier window function, may be used during refinement to better isolate spectral and structural signatures. Other improvements are certainly possible, and the LARCH infrastructure is better designed to make such developments possible.

Acknowledgments

I would like to thank Bruce Ravel, Tom Trainor, Peter Eng, Joanne Stubbs, Tony Lanzilotti, Ken McIvor, and Darren Dale for discussions, and to the IFEFFIT mailing list for encouragement and continued motivation. This work was funded by through grants to GeoSoilEnviroCARS from the National Science Foundation – Earth Sciences (EAR-1128799) and Department of Energy - Geosciences (DE-FG02-94ER14466).

[1] Filipponi A, Cicco A D and Natoli C R 1995 Physical Rev. B 52 15122–15134
[2] Tomic S, Scarle B G, Wander A, Harrison N M, Dent A J, Mosселmans J F W and Ingelsfield J E 2005 CCLRC Technical Report DL-TR 001 1
[3] Ressler T 1998 Journal of Synchrotron Radiation 5 118–122
[4] George G N and Pickering I J 1995 EXAFS-PAK Tech. rep. Stanford Synchrotron Radiation Laboratory Stanford, CA
[5] Klementev K 2001 Journal of Physics D- Applied Physics 34 209–217
[6] Taguchi T, Ozawa T and Yashiro H 2005 Physica Scripta T115 205–206
[7] Michalowicz A, Moscovici J, Muller-Bouvet D and Provost K 2009 Journal of Physics: Conference Series 190 12034–12035
[8] Stötzel J, Luetzenkirchen-Hecht D and Frahm R 2011 Journal of Synchrotron Radiation 18 165–175
[9] Tanida H, Yamashige H, Okina Y, Oishi M, Takanashi Y, Fujimoto T, Sato K, Takamatsu M, Murayama H, Arai H, Matsubara E, Uchimoto Y and Ogumi Z 2011 Journal of Synchrotron Radiation 18 919–922
[10] Hayashi H, Kawata M, Takoda R, Udagawa Y, Watanabe Y, Takano T, Nanao S and Kamamura N 2004 Journal of Electron Spectroscopy and Related Phenomena 136 191–197
[11] Meier F, Cahana J, Liu Y, Mehta A, Andrews J C and Pianetta P 2011 Journal of Synchrotron Radiation 18 773–781
[12] Newville M 2001 Journal of Synchrotron Radiation 8 322–324
[13] Ravel B and Newville M 2005 Journal of Synchrotron Radiation 12 537–541
[14] Webb S M 2006 SixPack http://sixpack.sams-xrays.com/
[15] Prestipino C 2011 PrestoPronto http://code.google.com/p/prestopronto/
[16] Wall L 1997 Perl Programming Langauge http://perl.org/
[17] von Rossum G and Drake F L 2003 Python Reference Manual http://python.org/
[18] Oliphant T E 2006 Numerical Python http://numpy.scipy.org/
[19] Jones E, Oliphant T, Peterson P et al. 2001 SciPy: Open source scientific tools for Python http://scipy.org/
[20] Hunter J D 2007 matplotlib http://matplotlib.sourceforge.net/
[21] Zabinsky S I, Rehr J J, Ankudinov A, Albers R C and Eller M J 1995 Physical Rev. B 52 2995–3009
[22] Newville M 2001 Journal of Synchrotron Radiation 8 96–100
[23] Garbow B S, Hillstrom K E and Mõre J J 1980 MINPACK-1 http://www.netlib.org/minpack
[24] Elam W T, Ravel B D and Sieber J 2002 Radiation Physics and Chemistry 63 121–128
[25] Chantler C T 2000 Journal of Physical and Chemical Reference Data 29 597–1048
[26] Weng T C, Waldé G S and Penner-Hahn J E 2005 Journal of Synchrotron Radiation 12 506–510
[27] Chantler C T 2012 Journal of Synchrotron Radiation 19 145–158
[28] Munoz M, Argoul P and Farges F 2003 American Mineralogist 88 694–700