**IFEFFIT and LARCH**

Matthew Newville* and Bruce Ravel

*Center for Advanced Radiation Sources, University of Chicago, Chicago, IL 60637, USA, and \(^*\)National Institute of Standards and Technology, Gaithersburg, MD 20899, USA. \(^*\)Correspondence e-mail: newville@cars.uchicago.edu

**IFEFFIT** is a computer program and callable library for XAFS analysis. It provides an interactive session for reading, manipulating and viewing EXAFS data. **IFEFFIT** includes a full suite of tools and methods for EXAFS analysis, including functions for pre-edge removal and normalization of EXAFS data, background removal using the **AUTOBK** method and fitting EXAFS with theoretical calculations for scattering amplitudes and phase shifts from **FEFF**. The **IFEFFIT** library is used as the backend to the popular GUI programs **ATHENA**, **ARTEMIS** and **SIXPACK**. **LARCH** is a successor to **IFEFFIT**, and includes all the EXAFS functionality of **IFEFFIT**, while adding functionality for XANES and related X-ray spectroscopies and overcoming several intrinsic limitations of **IFEFFIT**. For example, **LARCH** provides better tools for general-purpose data fitting, X-ray fluorescence analysis and interaction with databases of fundamental X-ray data. Written in and extensible with Python, **LARCH** can make use of the wide range of scientific libraries available for Python.

1. **FEFFIT and the UWXAFS analysis package**

In order to make full use of the revolutionary EXAFS calculations from **FEFF** (Rehr et al., 1991; Zabinsky et al., 1995; Kas et al., 2020) and especially the newly available contributions from multiple-scattering paths, new EXAFS modelling and analysis codes were required. The **FEFFIT** program (Newville et al., 1995; Newville, 2001a) was one of many analysis codes created or modified in the early 1990s to use these newly available theoretical calculations. It used the sum-of-scattering-paths formalism from **FEFF** as the basis for the model in a nonlinear least-squares fitting to experimental EXAFS data (see Newville, 2020a). Although few of its features were completely novel or unique, **FEFFIT** used a number of ideas that came to define best practices in EXAFS analysis, including the following.

(i) The ability to fit data in **R** or **k** space, with a preference for using **R** space.

(ii) A flexible system for constraining the parameters \(N S_0^2\), \(R\), \(E_0\), \(\sigma^2\) and \(C_3\) (third cumulant) as defined below in the EXAFS equation between different scattering paths.

(iii) The ability to refine multiple data sets in one fit, sharing or constraining variable parameters between different data sets.

(iv) Automated error analysis and reporting of fit statistics and uncertainties and correlations between variable parameters using standard statistical analysis.

**FEFFIT** was bundled with **AUTOBK** (Newville et al., 1993) for background removal (see Newville, 2020b), **ATOMS** (Ravel, 2001) for generating **FEFF** inputs and a few utility programs into the **UWXAFS** (Stern et al., 1995) suite of programs for EXAFS analysis.

The exponential increase in the number of scattering paths with distance presents a fundamental challenge for the
inherently limited amount of information available in an experimental EXAFS signal (Stern, 1993). Although one might want to allow all parameters in the EXAFS equation to be varied to best match experimental data, it was recognized that this is not possible and that using prior knowledge and placing constraints on the EXAFS parameters are necessary for an analysis to give robust and physically meaningful results. FEFIT employed a flexible system for placing such constraints. The parameters in the EXAFS equations for each path, \( N^i_o, R, E_p, \sigma^2 \) and \( C_o \), were not varied directly in the fit but were instead written as mathematical expressions of generalized variables that could be used to determine the path parameters for many paths, or even for many data sets. Of course, one could simply have \( N \) and \( R \) vary for a particular path, but one could also set \( N \) for two different paths to vary while constraining them to add up to some value in order to allow two different ligands to contribute to a shell and fitting the contribution from each type of ligand. One could also define and vary a single expansion parameter that would increase \( R \) for all scattering paths, or use a characteristic Einstein temperature to set values for \( \sigma^2 \) for a number of paths in a simple material (Sevillano et al., 1979).

The automated error analysis in FEFIT was the result of a community-wide effort to improve the reporting of reliable statistical results for EXAFS analysis (Lytle et al., 1989) including uncertainties and parameter correlations, and led to a much clearer presentation of the strengths and limits of results of EXAFS analysis in the literature. The FEFIT program was used to solve a number of complex EXAFS analysis problems, including multiple scattering and polarization-dependent EXAFS, and has been used by many researchers and for hundreds of published EXAFS articles.

2. IFEFFIT

While the FEFIT program and UWXTAFS package successfully met the analysis needs for many researchers, it was not particularly user-friendly. It had a primitive user interface using a structured text input file for each fit, defining the scattering paths and fitting parameters to use. In particular, it had no graphics capabilities for visualization of data and fit quality, and thus relied on external programs to read and visualize results from output files. Because of its use of formatted input text files and its strict approach of running one fit at a time, it could not easily be incorporated into interactive analysis workflows. In addition, the UWXTAFS package had a restrictive licence and nominal licensing fee which prevented redistribution, sharing or modifying the code. This combination of factors made it difficult for novices to learn these tools, which limited the use of the UWXTAFS programs to expert users.

The IFEFFIT program and library (Newville, 2001b) was created in an effort to overcome these limitations, and to provide a more interactive environment for exploratory data analysis. The essential XAFS algorithms of FEFIT and AUTOBK were left essentially unchanged, but made more easily separated and repeated. The main data-reduction and analysis steps, such as

(i) reading arrays from data files,
(ii) pre-edge subtraction and normalization,
(iii) post-edge background subtraction,
(iv) XAFS Fourier transform,
(v) defining the FEFF path,
(vi) fitting \( \chi(k) \) data to the sum of paths

and so forth became functions in a primitive domain-specific language with an interactive shell that included some plotting functions for the one-dimensional data used in EXAFS analysis. This provided a limited but scriptable interface with a flexible workflow that allowed the user to re-process and visualize the data and fit results as the analysis progressed.

Because it relied on open-source software including MINPACK (Moàe et al., 1980), FITPACK (de Boor, 1978) and FFTPACK (Swartztrauber, 1984) and aimed at making XAFS analysis more accessible for non-expert users, IFEFFIT used an open-source (BSD) licence that encouraged free use and redistribution of the source code. In addition to an interactive shell program, IFEFFIT provided a programming library so that an interactive session could be run from outside programs and included wrappers for C, Fortran, Perl and Python so analysis sessions could be created and used from these languages. This design and licence allowed user-friendly GUI applications such as ATHENA and ARTEMIS (Ravel & Newville, 2005; Ravel & Newville, 2020), SIXPACK (Webb, 2005, 2020) and PRESTROPONTO (Figuerola & Prestipino, 2016; Prestipino, 2020) to use the IFEFFIT library.

The GUI applications built with IFEFFIT have gained a wide user base and have been used in hundreds of publications per year for the past decade. While the quality of the analysis algorithms may be partly responsible this usage, the difference in adoption of the IFEFFIT-based GUIs compared with the UWXTAFS package demonstrates that nontechnical factors also played a role. These factors include (i) an active and open-source development model, (ii) an active, public mailing list with discussions from many expert users on all topics related to XAFS and (iii) many XAFS tutorials and training sessions using ATHENA, ARTEMIS and SIXPACK as XAFS training tools. We suspect that all these factors, as well as a good deal of luck and the so-called bandwagon effect, account for the high adoption rate of these tools.

3. LARCH: IFEFFIT and more

Although IFEFFIT and the GUIs built with it have been used successfully by many researchers, it has many limitations that ultimately arise from its implementation in Fortran77 and use of static memory. Analysis sessions could easily approach the compiled-in memory limits, and the internal data model was limited to one-dimensional arrays, scalars and strings, which gives poor support for data from complex sources or with complex formats, including modern multidimensional X-ray detectors. The IFEFFIT session was limited enough that any serious scripted analysis required an external language. Finally, the implementation made adding new functionality
or modify existing functionality very difficult for anyone but the original authors.

Fortran was used for both FEFFIT and IFEFFIT because it was the language used for FEFF and most numerical scientific codes through the 1990s. As the interoperability of computer languages and the ability to use high-level interpreted languages for numerical work improved, this choice was no longer necessary after about 2005 or so. In particular, Python, an object-oriented interpreted language with clean and simple syntax, became widely adopted for scientific programming. Python libraries such as NumPy and SciPy (Oliphant, 2007) wrap standard numerical libraries, including those used by IFEFFIT, and many advanced analysis and statistical tools into a comprehensive set of open-source libraries, including access to complex data sources.

The LARCH software package (Newville, 2013) implements all of the XAFS analysis algorithms of IFEFFIT in Python, building on the wealth of scientific Python libraries. Like IFEFFIT and most scientific Python libraries, LARCH uses a liberal BSD-like licence and an open development process. This use of a high-level scripting language greatly simplifies implementation and debugging, and eliminates issues with memory allocation common in C or Fortran. Using Python’s data model immediately enables support for large, complex, data structures including multidimensional arrays. Supported data sources include plain ASCII files, standard image formats, HDF5 files, relational databases or even direct reading from control systems used at synchrotron sources. As is appropriate for EXAFS and XANES analysis, LARCH puts special emphasis on methods for fitting scientific data. By using the LMFIT package (Newville et al., 2019) that was developed in conjunction with LARCH, the user can easily switch between several fitting algorithms, place robust bounds on all fitting parameters (James & Roos, 1975), explore parameter uncertainties and confidence levels, and propagate uncertainties. Although designed with XAFS as the main emphasis, LARCH intentionally aims to be useful for other X-ray spectroscopies and methods, and so includes X-ray data cross sections and resources (Elam, 2002; Chantler, 2000), tools for X-ray fluorescence (XRF), X-ray fluorescence mapping and related spectroscopies, as well as some tools for data acquisition. LARCH also provides publication-quality interactive graphics for both one- and two-dimensional data, and can easily make use of the growing set of Python libraries for the analysis of scientific data.

Like IFEFFIT, LARCH provides an interactive session with a command language. For LARCH, that domain-specific language is derived from Python itself, and includes programming constructs such as conditionals, loops, user-defined functions, string handling, data structures and clean syntax for selecting sub-arrays and object attributes. LARCH keeps this command language well separated from the scientific functions, and all functionality can be accessed, modified and tested from Python directly and without fear of adversely affecting unrelated functionality.

By using Python and the scientific Python libraries, the LARCH implementations of the XAFS algorithms are simplified and easier to comprehend than their Fortran versions. For example, the AUTOBK algorithm was originally a standalone program in the UWXAFS package with its own input/output routines, became a built-in function in IFEFFIT and ultimately was reduced to about 250 lines of Python code, even while adding the calculation of the variance in $\chi(k)$ so

### Table 1

Values for the path parameters for the 13 paths used in the fit to EXAFS data for room-temperature Cu metal.

The values refined for $S_0^2$ (S02) and $\Delta E_0$ (e0) were 0.925 (0.025) and 5.31 (0.41), respectively, and these were used for all paths. Note that the values of $R$ did not vary independently but were constrained for all paths by the linear scale value $\alpha$ (alpha), which was refined in the fit to be 0.028 (0.013).

<table>
<thead>
<tr>
<th>Path</th>
<th>N_{legs}</th>
<th>$N$</th>
<th>$R$ (Å)</th>
<th>$\sigma^2$ (Å$^2$)</th>
<th>$C_2$ (Å$^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>12</td>
<td>2.562 (0.003)</td>
<td>0.0089 (0.0002)</td>
<td>1.7 $\times$ 10^{-4} (0.3)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3.618 (0.005)</td>
<td>0.0141 (0.0008)</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>48</td>
<td>3.836 (0.005)</td>
<td>0.0097 (0.0028)</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>48</td>
<td>4.364 (0.006)</td>
<td>0.0110 (0.0003)</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>24</td>
<td>4.427 (0.006)</td>
<td>0.0110 (0.0003)</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>48</td>
<td>4.769 (0.006)</td>
<td>0.0066 (0.0012)</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>96</td>
<td>4.769 (0.006)</td>
<td>0.0066 (0.0012)</td>
<td>0</td>
</tr>
<tr>
<td>8–14</td>
<td>2, 3, 4</td>
<td>12,24</td>
<td>5.110 (0.006)</td>
<td>0.0110 (0.0003)</td>
<td>0</td>
</tr>
</tbody>
</table>

### Figure 1

EXAFS $k^2\chi(k)$ of an example fit (dashed) to Cu K-edge EXAFS (solid) of copper metal (room temperature) using four single-scattering paths and nine multiple-scattering paths.

### Figure 2

EXAFS $\chi(R)$ of an example fit (dashed) to Cu K-edge EXAFS (solid) of copper metal (room temperature) using four single-scattering paths and nine multiple-scattering paths. Imaginary parts of $\chi(R)$ for the data and fit are shown, as well as $\chi(R)$ for the data, and the difference in the imaginary parts of $\chi(R)$ (offset) over the fitted $R$ range (shaded).

3 of 5  Newville and Ravel  •  IFEFFIT and LARCH

that uncertainties from data-processing steps can be better propagated through the full analysis.  
LARCH and all of its functionality can be used directly as a Python library, from a dedicated interactive session or using scripts for the interactive session. To be useful from languages other than Python, and to be compatible with the existing Python library, from a dedicated interactive session or using remote procedure calls. In addition, LARCH also comes with a few graphical user applications of its own. One such application is XAS Viewer, which is meant to be a refreshed version of ATHENA designed for EXAFS data reduction and XANES processing including pre-edge peak fitting, principal component analysis, linear combination fitting and advanced regression methods that is well supported for all common operating systems. LARCH also includes graphical user applications for displaying XRF and X-ray diffraction maps from X-ray microprobe beamlines, and for visualizing and quantitative analysis of XRF data.  

As an example EXAFS analysis, Figs. 1 and 2 show the fit to Cu K-edge EXAFS of room-temperature copper metal through the fourth shell of the face-centred-cubic structure, using four single-scattering and nine multiple-scattering paths from FEFF. The fit was performed between \( R = 1.3 \) and 4.8 Å (shaded in Fig. 2), after Fourier-transforming the \( k^2 \chi(k) \) between \( k = 3 \) and 16 Å\(^{-1}\) using a Kaiser–Bessel window. The fit parameters included an overall amplitude factor \( S_0^2 \), an energy shift \( \Delta E_0 \) and a single scaling parameter \( \alpha \) for all distances (that is, multiplied by the nominal half-path length), five independent values of \( \sigma^2 \) and a third cumulant term for the near-neighbour path. The complete text of the LARCH script used to perform this fit is shown in Fig. 3, and numerical results for the fitting variables and path parameters are given in Table 1.  

A consistent theme in the development from FEFFIT, through IFEFFIT and to LARCH, has been the goal to provide robust analysis tools to X-ray spectroscopy community that reflect the best practices for correctness while also being accessible for novice users. By using high-level scripting languages and building on the wealth of open-source scientific tools, we aim to not only enable good analysis practices but also to encourage study and further developments of these tools, with each using fewer than 300 lines of code. This synthesis allows these methods to be integrated with other methods so that, for example, LARCH can now use the wavelet transforms to fit XAFS data in \( k \), \( R \) or ‘wavelet’ space. In addition, well supported third-party libraries for machine-learning methods can readily be used for XANES analysis. As mentioned above, the nonlinear least-squares fitting for XANES pre-edge peaks and FEFF fitting relies on the LMFIT package because a larger community of scientists has added functionality for better exploration and propagation of uncertainties that can be used in these XAFS analysis tasks. With the improved ability to develop and improve

### Larch Script for Fitting Cu XAFS

```python
# read data, perform background subtraction
cu_data = read_xdi('cu_metal.rt.xdi',
    label='s=' energy i d i i')
cu_data.mu = -log(cu_data.i/0/cu_data.i)
autobk(cu_data, rbgk=1.2, kw=2)

# define fitting parameter group
fit_params = group(s02 = guess(1),
    e0 = guess(4),
    al pha = guess(0),
    c3_p1 = guess(0),
    s2_p1 = guess(0.002),
    s2_p2 = guess(0.002),
    s2_p3 = guess(0.002),
    s2_p4 = guess(0.002),
    s2_p6 = guess(0.002))

# define a list of Feff Paths and Path Parameters
paths = [feffpath('feff0001.dat', \( \text{si gm 2} = \text{ss2_p1} \),
    \text{th i r d} = \text{c3_p1}, \text{s o2} = \text{s02}',
    \text{e o} = \text{e0}', \text{de l tar} = \text{al ph a+re f f}')]

# set FT parameters, window function, fit ranges
trans = feffit_transform(km n=3, km x=16, kw=2, d k=5,
    \text{w n d o w} = \text{ka i ser}', \text{r m n} = 1.3, \text{r m x} = 4.8)

# define dataset with data, transform path list
dset = feffit_dataset(data=cu_data,
    \text{t r a n s} = \text{trans}, \text{path list} = \text{paths})

# perform fit, show results
result = feffit(fit_params, dset)

print(feffit_report(result))
plot_chifit(dset, show_image=True, r m x=6.5)
```

---

**Figure 3**

The LARCH script used to perform the fit to the Cu K-edge EXAFS of copper metal shown in Figs. 1 and 2. After reading raw data and extracting \( \chi(k) \), fitting parameters are defined using the guess() function. Several scattering paths are created with the feffpath() function, which takes the name of a feffNNNN.dat file and arguments to define expressions for the path parameters, some of which are common to all paths. Fourier transform and fit range are defined and used with the list of paths to model the data. The fit is performed with the feffit() function, which optimizes the fitting parameters so that the sum of paths matches the data. Results are then printed out, and plots similar to those in Figs. 1 and 2 are created.
XAFS analysis methods and access to the rapid developments of analysis tools in Python from other scientific disciplines, we are confident that things should start to get interesting right about now (Dylan, 1997).

References