

IFEFFIT: interactive XAFS analysis and FEFF fitting

Matthew Newville*

Consortium for Advanced Radiation Sources, The University of Chicago, 5640 S Ellis Ave, Chicago, IL 60637, USA.
E-mail: newville@cars.uchicago.edu

IFEFFIT, an interactive program and scriptable library of XAFS algorithms is presented. The core algorithms of AUTOBK and FEFFIT have been combined with general data manipulation and interactive graphics into a single package. IFEFFIT comes with a command-line program that can be run either interactively or in batch-mode. It also provides a library of functions that can be used easily from C or Fortran, as well as high level scripting languages such as Tcl, Perl and Python. Using this library, a Graphical User Interface for rapid 'on-line' data analysis is demonstrated. IFEFFIT is freely available with an Open Source license. Outside use, development, and contributions are encouraged.

Keywords: XAFS Analysis.

1. Introduction

The availability of high-quality and easy-to-use programs for data analysis is of primary importance for beamlines and practitioners of XAFS. As Dimakis and Bunker wrote in 1999 (Dimakis & Bunker, 1999), describing their APEX program:

There has long been a need for a basic set of x-ray absorption fine-structure analysis (XAFS) programs that are freely available, including source code; that runs on all major operating systems; is readily extensible; is easy to use for beginners; and is a useful tool for experts.

This is an ambitious set of requirements, but certainly achievable, as Dimakis and Bunker have shown with their wrapper for the freely available batch-processing UW/NRL Fortran programs.

In addition to the requirements that Dimakis and Bunker set, I would also add: the use of robust EXAFS analysis procedures, and a facility for users to write macros, scripts, or even programs of varying levels of sophistication for repetitious analysis tasks. The need for form-based Graphical User Interfaces (GUIs) to otherwise complex EXAFS analysis procedures is probably obvious to the reader, as is the need for predictability, uniformity, and correctness of the analysis procedures used. With the gravity of all of these lofty analytic, social, and programming goals in mind, I describe IFEFFIT, an attempt to bring interactivity, flexibility, and scripting capabilities to the AUTOBK (Newville *et al.*, 1993) and FEFFIT (Newville *et al.*, 1995) algorithms, and demonstrate its programmability with a fledgling Graphical Interface, G.I.FEFFFIT.

2. Command Line Interface and Data Handling

IFEFFIT is implemented as a programming library that provides an interactive session which is available either from within a calling program or directly to the user through a command-line interface. The session takes commands to read, write, and manipulate data as

input as processes the data as requested. Though IFEFFIT is implemented as a program library, not as an executable program *per se*, a simple command-line-interface (CLI) is easily implemented and comes installed with the library. This CLI (simply called `ifef-fit`) serves as the most basic interface to the IFEFFIT library.

The data manipulation capabilities of IFEFFIT include XAFS analysis procedures that implement the AUTOBK background-removal scheme, XAFS Fourier transforms, and modeling and fitting of XAFS data with FEFF (Zabinsky *et al.*, 1995; Ankudinov & Rehr, 1997) calculations, based on the algorithms that make up FEFFIT. IFEFFIT also supports the plotting and simple algebraic manipulation of data (+, -, ×, /, sin, cos, exp, *etc.*). This is a very different philosophy of data analysis than in the stand-alone AUTOBK and FEFFIT programs which, when packaged as part of the UWXAFS Project (Stern *et al.*, 1995), created a rigid and fixed format of procedures and user-interaction for XAFS analysis.

IFEFFIT provides the user with the ability to create, read, and write data into named program variables in the IFEFFIT "session". Data for XAFS analysis is generally stored as one-dimensional arrays, and IFEFFIT supports and distinguishes data for *arrays*, *scalars*, and *text strings*. General mathematical manipulation of data is supported for arrays and scalars, so that it is trivial to do common tasks such as adding two arrays element-by-element or scaling every element of an array by a constant factor. Currently, input data is expected to be in simple ASCII text files with columns of data corresponding to one-dimensional arrays. IFEFFIT makes the reading and writing of data files to and from program variables very easy.

The commands that IFEFFIT supports are a mixture of fairly high-level XAFS analysis commands and basic mathematical data processing. A complete listing is available on-line (Newville, 2000b), and a partial listing of commands is given in table 1.

Table 1
Summary of principle IFEFFIT commands and their meanings.

Command	Functionality
<code>read_data</code>	read arrays from ASCII data file
<code>write_data</code>	write arrays to ASCII data file
<code>set</code>	create or set the value of a variable
<code>show</code>	show the value of a variable
<code>plot</code>	plot x-y arrays
<code>pre_edge</code>	XAFS pre-edge subtraction and normalization
<code>spline</code>	XAFS background subtraction (AUTOBK)
<code>fftf</code>	XAFS Fourier transform $k \rightarrow R$
<code>fftr</code>	XAFS Fourier transform $R \rightarrow k$
<code>path</code>	Define FEFF path and Path Parameters
<code>ff2chi</code>	combine FEFF paths to $\chi(k)$
<code>feffit</code>	combine FEFF paths to best-fit $\chi(k)$ data
<code>minimize</code>	general fitting of simple functions or combinations of data arrays.

Because many commands are repeated for analysis of XAFS data, and because it is desirable to be able to apply identical procedures for different sets of data, IFEFFIT has a simple facility for writing and saving "macros" of commands: multiple commands that are executed with a single user-defined macro name. A primitive method exists for substituting parameters in the macros, and for more sophisticated processing batch files or scripts written in high-level languages can be used.

IFEFFIT uses the PGPLOT (Pearson, 1997) graphics library for plotting data. Currently, this library works well to produce plots for X Windows, in Postscript, and in GIF. This library has been ported to native Win32 systems, and IFEFFIT has had some success

using this port. IFEFFIT can also be built without built-in graphics from PGPLOT.

3. Programmer's Interface

The IFEFFIT library is written in Fortran, and can be used from Fortran or C programs. Special attention has been given to ensure that this library can also be used from high-level scripting languages such as Tcl(Ousterhout, 1994), Perl(Wall *et al.*, 1997), and Python(von Rossum, 1995; Lutz & Ascher, 1999). All of these languages execute directly from source code without an intermediate compilation step, run on Unix, Windows, and Macintosh systems, and are freely available. Scripts written for one operating system can usually be ported to other systems with minimal work. Such scripting languages are ideal for rapidly developing small scientific programs. As will be demonstrated in the next section, they can also be used for writing cross-platform GUIs.

The application programmer's interface to the IFEFFIT library is very simple, and extensive programming experience is not required to use it. The IFEFFIT library provides 7 basic functions, which give full access to an underlying "IFEFFIT session" and its data from the calling program or script. The principle function in the library is `ifeffit()`, which executes IFEFFIT command strings as if they were typed at the command-line. The remaining 6 functions provide mechanisms to move IFEFFIT's three principle data types (scalar, array, and text string) from the "IFEFFIT session" directly to and from the calling program. These are `get_scalar()`, `get_array()`, `get_string()`, `put_scalar()`, `put_array()`, and `put_string()`. More detailed descriptions are available in the on-line documentation(Newville, 2000b).

A simple example of using the IFEFFIT library from within another program would be to add AUTOBK background capabilities for data already existing in a program, such as a beam-line data acquisition program. Such a program might look like this (this example is in Python, as it is the most like pseudo-code of all the available languages), assuming that data for $\mu(E)$ exists in the arrays energy and mu_data):

```
#!/usr/bin/python
## load the ifeffit library
from ifeffit import *

##
## process mu(E) --> energy and mu_data arrays
ret1 = put_array(energy, 'dat.energy')
ret2 = put_array(mu_data, 'dat.mu')

## execute ifeffit's spline command
ifeffit('spline(dat.energy,dat.mu, rbkg=1.1
        find_e0, kweight=1)')

## create k*chi(k) inside ifeffit
ifeffit('set dat.chik = dat.k * dat.chi')

## retrieve chi(k) and k*chi(k) from ifeffit
k = get_array('dat.k')
chi = get_array('dat.chi')
chik = get_array('dat.chik')
e0 = get_scalar('e0')
step = get_scalar('edge_step')

## print out data retrieved from ifeffit
print "e0 = ", e0, " edge step = ", step
print " chi(k) data has ", len(chi), " points."
```

Figure 1

Example script using the IFEFFIT Python extension to perform a simple AUTOBK-like background subtraction. This example shows the execution of IFEFFIT's commands `spline` and `set` and also the passing back and forth of data from the script to the underlying IFEFFIT session with the `put_array`, `get_array`, and `get_scalar` functions.

4. G.I.FEFFIT: Graphical Interactive IFEFFIT

One of the most important aspects of the IFEFFIT programming library is its ability to be called from high-level scripting languages. Such languages are excellent for rapid development of small to medium scientific applications, and often suitable for full applications, especially when used as a wrapper or "front-end" to an underlying library. Moreover, such languages typically include support for building graphical user interfaces. All three of the scripting languages (Tcl, Perl, and Python) for which IFEFFIT currently has wrappers provide GUI toolkits for Windows and Unix systems. Both Tcl and Python also support GUIs on the Macintosh.

Therefore, although IFEFFIT is not a GUI program itself, graphical form-based applications can be built fairly easily by combining it with a high-level scripting language. As a demonstration of this ability, as a teaching tool, and as a quick on-line data analysis system, G.I.FEFFIT has been written. While essentially an enhanced command-line program, G.I.FEFFIT provides graphical forms to assist common analysis tasks such as reading and writing data files, plotting arrays, and simple XAFS analysis tasks like pre-edge subtraction, normalization, post-edge background removal and Fourier transforms. G.I.FEFFIT is still in development. At this writing it does not have a complete form-based interface to IFEFFIT-type fitting of XAFS data, but it is an easy-to-use, flexible program for simple data processing, plotting, background subtraction, and Fourier transforms. G.I.FEFFIT is written in Python, and has been written and tested on linux and other Unix systems using X Windows. A port to Windows is in progress. Figs 2, 3, and 4 show example forms from G.I.FEFFIT, and Fig. 5 shows an example plot of data resulting from the background removal window.

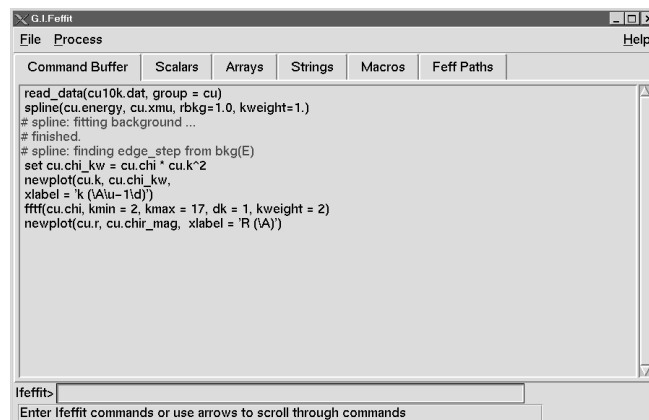


Figure 2

Main Window for G.I.FEFFIT showing the enhanced command-line interface, and notebook selections for different types of data in IFEFFIT. Commands can be typed here, or pre-defined forms can be used for data plotting and common analysis tasks such as background removal.

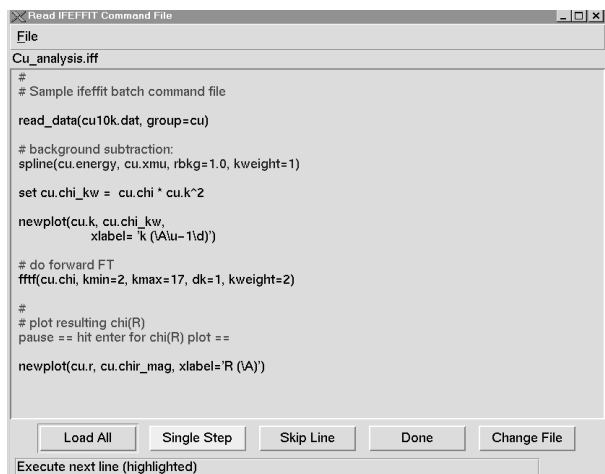


Figure 3
An enhanced batch file processor for G.I.FEFFFIT, ideal for learning XAFS analysis with IFEFFIT. A file IFEFFIT commands can be executed in full or stepped through one at a time.

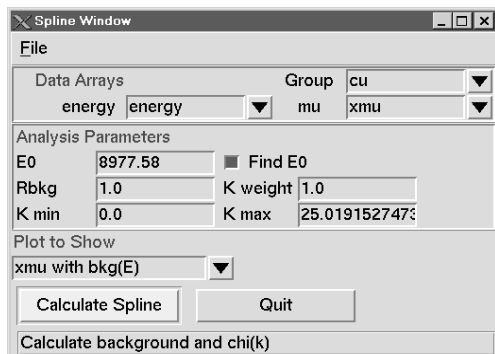


Figure 4
A simplified form for AUTOBK-like background removal. After selecting arrays for E and μ , the background $\mu_0(E)$ is calculated and one of several pre-defined plots of the data and background is drawn.

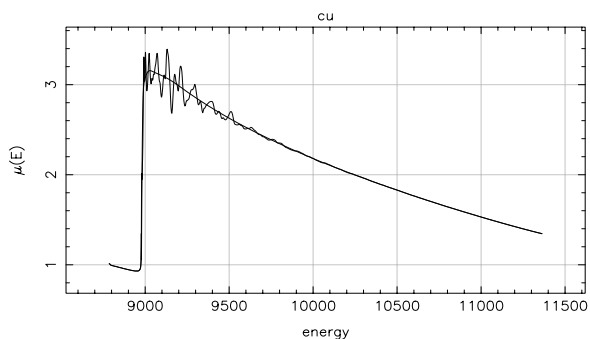


Figure 5
Sample background from G.I.FEFFFIT's AUTOBK form, showing the $\mu(E)$ and $\mu_0(E)$ data for Cu K -edge of Cu metal.

5. Conclusions

IFEFFIT is free software, distributed under an Open Source license. It is still in a relatively early stage of development, though most of the algorithms needed for XAFS analysis are supported. In comparison to the advanced fitting capabilities of FEFFIT (Newville, 2000a), IFEFFIT is missing a few important features, most notably simultaneous fitting of multiple data sets and improved background refinement. Such advanced features as well as other algorithms such as principle component analysis, and anomalous scattering factors are planned or partially implemented at this writing. IFEFFIT already has many capabilities not found in most XAFS analysis packages, and has the potential for much larger command set and more complete analysis tools.

I thank Bruce Ravel, Julie O. Cross and John Rehr for helpful discussions on the structure of IFEFFIT, and the many users of AUTOBK and FEFFIT for providing kind and encouraging feedback.

References

- Ankudinov, A. L. & Rehr, J. J. (1997). *Physical Review B*, **56**, R1712–R1715.
- Dimakis, N. & Bunker, G. (1999). *Journal of Synchrotron Radiation*, p. 274.
- Lutz, M. & Ascher, D. (1999). *Learning Python*. Sebastopol, CA: O'Reilly.
- Newville, M. (2000a). *Journal of Synchrotron Radiation*, p. these proceedings.
- Newville, M. (2000b). IFEFFIT Web Page and On-Line Documentation. <http://cars9.uchicago.edu/ifeffit/>.
- Newville, M., P. Līviņš, Yacoby, Y., Rehr, J. J. & Stern, E. A. (1993). *Physical Review B*, **47**(21), 14126–14131.
- Newville, M., Ravel, B., Haskel, D., Rehr, J. J., Stern, E. A. & Yacoby, Y. (1995). *Physica B*, **208&209**, 154–156.
- Ousterhout, J. K. (1994). *Tcl and the Tk Toolkit*. Menlo Park, CA: Addison-Wesley.
- Pearson, T. J., (1997). PGPLOT Graphics Library version 5.2.0. <http://astro.caltech.edu/~tjp/pgplot/>.
- von Rossum, G., (1995). Python Language Documentation and Website. <http://www.python.org/>.
- Stern, E. A., Newville, M., Ravel, B., Yacoby, Y. & Haskel, D. (1995). *Physica B*, **208&209**, 117–120.
- Wall, L., Christianson, T. & Schwartz, R. L. (1997). *Programming Perl*. Sebastopol, CA: O'Reilly.
- Zabinsky, S. I., Rehr, J. J., Ankudinov, A., Albers, R. C. & Eller, M. J. (1995). *Physical Review B*, **52**(4), 2995–3009.