Development of EXAFS Analysis Software

Matthew Newville

Center for Advanced Radiation Sources,
The University of Chicago

EXAFS-50: 2023-Oct-30

A Personal Perspective: 1990 to present.

Thanks to: Bruce Ravel, Ed Stern, John Rehr, Yizhak Yacoby, Steve Zabinsky, Peter Livins, Daniel Haskel, Anatoly Frenkel, Yanjun Ma, Mao Xi Chen, Shelly Kelly, Julie Cross, . . .

The Problems with EXAFS Analysis

EXAFS Analysis is hard.

The normal - simplified - version of the EXAFS Equation is

$$\chi(k) = \sum_j \frac{N_j f_j(k) e^{-2R_j/\lambda(k)} e^{-2k^2\sigma_j^2}}{k{R_j}^2} \sin[2kR_j + \delta_j(k)]$$

Here $f(k)$ and $\delta(k)$ are *photo-electron scattering properties* of the neighboring atom and $\lambda(k)$ is the photo-electron mean-free-path. None of these are trivial functions.

And we want to extract R and N, and maybe σ^2 from this. Not easy.

We don't even measure $\chi(k)$! There is a messy "background subtraction" step to convert measured data $\mu(E)$ to $\chi(k)$.

To summarize the talk so far:

- getting beam time at a synchrotron: hard.
- getting good data: hard.
- getting $\chi(k)$ from measured data: hard.
- getting R N, σ^2 from $\chi(k)$: hard.

At 50 years, one still basically need a PhD and years of study to analyze EXAFS.

The State of EXAFS Analysis software in 1990-1991

I joined Ed Stern's group at U Washington in 1990. Writing software was not the main goal.

Data Processing and Analysis Software at that time used department VAX/VMS systems, with several DEC text-based terminals, and separate programs for

- pre-edge subtraction and normalization.
- EXAFS background removal.
- Fourier transform and filtering.
- Log-ratio method
- Fitting $\chi(k)$ with tables of $f(k)$, $\delta(k)$.
- Plotting to the terminals with Tektronix library.

Everything was in Fortran. Each of these used a separate configuration file to process one spectrum at a time. These were not really "interactive".

We knew this was not ideal. The analysis methods were effectively limiting the work to single-shell EXAFS. That was a real problem for high-Tc superconductors, ferroelectric materials, basically everything interesting.

There was some work on minimal interactive data processing tools to speed up workflows, and to work with results from FEFF.

There was a general expectation that FEFF was going to make things better.

Rehr & Stern, Yizhak Yacoby, Steve Zabinsky

Yizhak Yacoby was a frequent visitor to the Stern lab, and had analysis codes and ideas that greatly influenced us:

- using Fourier filtering in background subtraction
- adding Feff4 (and later Feff5) calculations to fit EXAFS $\chi(k)$ data.

To be clear, these pre-dated anything similar we had in the Stern lab.

But: they were DOS/Windows3 only, and were very hard to use.

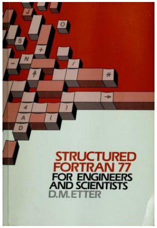
Editing Fortran code and highly-structured inputs was needed for each analysis. One student used these on one machine with Yizhak next to him. No one else could use these codes.

FEFF 4/pre-5 (Rehr, Albers, Mustre de Leon) calculations were clearly much better than tables from Lee, Pendry, etc. We knew we needed this.

Rehr's student - Steve Zabinsky - was tasked with implementing Multiple Scattering for FEFF 5. Steve was an excellent Fortran programmer. And he sat with in the lab with the Stern students and post-docs.

Zabinsky and post-doc Peter Livins taught me how to program.

Data Analysis and Programming were not clearly separated tasks.



Automating Background Subtraction

The first task in EXAFS processing is Background Removal:
Find a $\mu_0(E)$ that "almost matches" $\mu(E)$ to generate $\chi = (\mu - \mu_0)/\Delta\mu$.

Since there is no measured $\mu_0(E)$, a *cubic spline* – a piecewise polynomial with flexible points – is used to approximate a function. This leads to complications, questions:

- too many flexible points - "knots" - can match the full $\mu(E)$ spectrum
- how many knots should there be?
- how do we match the $\mu(E)$ spectrum?

These were well-known questions, and earlier work (Cook and Sayers, 1981, and others) had suggested smoothing or Fourier filtering out the high-frequency components.

Yacoby was fitting a spline to make a candidate $\chi(k)$, Fourier transforming the result and picking spline knots that minimized $\chi(R)$ below some R value.

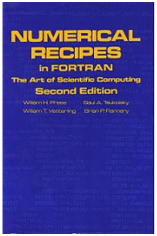
We (Stern Yacoby, Rehr, Livins, and I) refined this in a few ways:

1. Use a single $R_{\rm bkg}$ as the frequency cutoff: ignore higher R
2. Set the number of variable knots $1 + 2R_{\rm bkg}\Delta k/\pi$ (Shannon-Nyquist)
3. space the knots evenly in k
4. allow as "standard" $\chi(k)$ to allow for leakage from first shell to low-R.
5. put the spline evaluation and Fourier transform in the fitting objective function.

AUTOBK: Automating Background Subtraction

Implementing this meant understanding and using libraries for

- Evaluating splines: `pppack`
- Fast Fourier transforms `fftpack`
- Non-linear least-squares minimization `minpack`



There were descriptions of how these worked, and simple examples in *Numerical Recipes* (unofficial curriculum), but these were not really production-ready codes.

In 1990-1992 (pre-web), you could get codes from National Labs and NETLIB by sending an email message:

```
To: netlib@research.att.com
Subject: send lmdif from minpack
```

I got `AUTOBK` working and giving decent backgrounds with 1 physically meaningful parameter, $R_{\rm bkg}$, and only a few adjustable parameters. Published in Phys Rev B, 1993.

This was used in our group, and got installed on the microVax at NSLS X-11, so we could get better $\chi(k)$ for data during data collection.

In 1992(?), I presented this method at a PRT meeting for beamline X-11A at NSLS.
At that meeting, Bruce Bunker suggested writing an EXAFS fitting code using Feff5.

Feffit: Fitting EXAFS with Feff5 calculations

We knew that fitting EXAFS data with Feff calculations would be a radical development. We were not the first to do this, but we knew that people outside our group would be interested and wanted to get it right.

Feff5 made the Sum of Paths is pretty straight-forward:

$$\chi(k) = \sum_j \frac{N_j f_j(k) e^{-2R_j/\lambda(k)} e^{-2k^2\sigma_j^2}}{k{R_j}^2} \sin[2kR_j + \delta_j(k)]$$

where each path (single- or multiple-scattering) had arrays for all the Red terms. We could start with `feff5`'s subroutine to read and sum paths from its `feffNNNN.dat` files.

Minimizing "data-model" could be done after Fourier-transforming to R-space and selecting an R range to fit. Or we could allow fitting in k-space.

Some of the key challenges:

1. whether to fit in k- or R-space.
2. a sum-of-paths could quickly lead to way too many fitting variables.
3. there were real concerns about over-fitting and reporting uncertainties.

Stern, Rehr, Yacoby, Livins, Zabinsky, Ravel, and I discussed all of these topics at length. And this took about a year to get everything in place.

Feffit: Complications

To allow for variables (R, E_0, σ^2, higher order cumulants) things get a little messy.

The real EXAFS equation using Feff calculations for each Path

$$\chi(k) = \quad \mathrm{Im}\Big[\frac{f_{\rm eff}(k) N S_0^2}{k(R_{\rm eff}+dR)^2}\exp(-2p'' R_{\rm reff} - 2p^2\sigma^2 + \tfrac{2}{3}p^4 c_4) \times \exp\big\{i\big[2kR_{\rm eff} + \delta(k) + 2p(dR - 2\sigma^2/R_{\rm eff}) - \tfrac{4}{3}p^3 c_3\big]\big\}\Big]$$

where

$k = \sqrt{{k_{\rm feff}}^2 - 2m_e E_0/\hbar^2}$ is the wavenumber and index for the Feff arrays, and

$p = p' + ip'' = \sqrt{\big[p_{\rm real}(k) - i/\lambda(k)\big]^2 - i\,2m_e E_i/\hbar^2}$ is the complex wavenumber.

The terms in Blue are arrays (or sums or arrays) from the `feffNNNN.dat` file.

The terms in Red are all Path Parameters that could be varied in the fit... **for each Path**.

Some parts of the calculation use k to reconstruct the original Feff calculation, but the cumulants use complex p to modify that calculation.

The extra $-2p(2\sigma^2/R_{\rm eff})$ term in the phase is a correction to the cumulant expansion for averaging over the $1/R^2$ term in the EXAFS equation.

Feffit: Too many variables

With adjustable Path Parameters NS_0^2, E_0, dR, σ^2, c_3, c_4, and E_i available for each Path, and 3 to 10 paths, you could easily get way too many variables.

Basic information theory (Nyquist-Shannon) suggests that there are about

$$N = 1 + 2\Delta R \Delta k/\pi$$

independent measurements for data with an R range ΔR and k range Δk.

This is really just a basic property of signals, but caused many debates for years.

- there are many points in $\chi(R)$: it has been smoothed by zero-padding $\chi(k)$.
- $\mu(E)$ measurements below the edge are not measurements of first-shell coordination.
- if anything, this estimate is optimistic, ignoring entropy and noise.

After a decade, this was down to quibbling about "+1", "+2", or maybe "-10".

Practical view:

For typical data, $\Delta k = 12.5\text{Å}^{-1}$, giving about 8 independent variables per Å in R space. That's probably fine for a simple first shell, and might cause trouble at 4 or 5 Å with mixed shells and many multiple-scattering paths.

c_4 is never necessary, and fitting E_i is really rare. Usually, you would want 1 E_0 and 1 S_0^2 for all paths. Still, the number of variables can grow out of hand.

And maybe you want to model temperature dependence or something complicated. . . .

Feffit: Generalized Variables

The solution used in Feffit was to have distinct, separate fitting variables and write **all** Path Parameters as simple mathematical expressions of these fitting variables.

Example 1: one variable `del_e0` to be used for multiple E_0 Path Parameters.

Example 2: one variable `frac` could be used for competing sites. One path as NS_0^2 value of `N*frac`, while another is `N*(1-frac)`.

Example 3: we had built-in function for Einstein and Debye models so that 1 characteristic temperature could set σ^2 values for multiple paths. `eins(theta, temper)`

Also: Generalized Variables could be Fixed – not varied in the fit. There were not builtin bounds, but there were `min()` and `max()` functions.

There were no Fortran libraries for "parse-and-eval" of mathematical expressions, but I knew that converting to RPN (like an HP calculator) was needed and the "shunting yard algorithm" (E Dykstra) could help do the conversion of "N*(1-frac)" to ['N', '1', 'frac', '-', '*'].

Still, it was a bit of time to get it all to work ;).

This gave a really flexible way to set up models, if sort of complicating the input files. It allowed analyses that I think just would not have been possible any other way.

Feffit: Automating Error Analysis

At the time, EXAFS results did not always reporting uncertainties for R, N, σ^2, etc and would sometimes report just a nominal $\pm 0.01\text{Å}$ without a careful analysis.

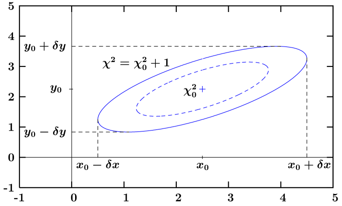
Getting 1-σ uncertainties from a non-linear least-squares fit was possible, but not always automated. It was **not** one of direct outputs from the MINPACK functions.

When done "by hand" in the very old days, some error analyses did not account for correlations between variables.

This was viewed as a very serious problem.

Getting and then inverting the *covariance matrix* after a fit is not really so hard, and that gives both the uncertainties in the variables and the correlation coefficients between them.

We did (maybe do) not have great estimates of uncertainties in the data. So we report uncertainties that increase χ^2 by reduced-χ^2 instead of 1.0



Propagating uncertainties from variables to Path Parameters is also not too hard.

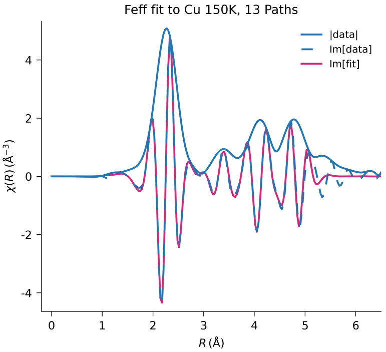
But also: Several aspects of this are improved with modern Python codes.

Feffit

By 1993, we had this all working - okay maybe a few bugs – and some nice demonstrations.

Within a year, `Feffit` could fit multiple data sets, sharing generalized variables across data sets.

With that in place, our group was able to it for several complex analyses:



- measure complex disorder and "buckling angles" in mixed salts (Frenkel, et al)
- distortions in high-Tc materias and perovskites (Ravel, Haskel, Qian, Yacoby)
- subtle temperature-dependence of coordinations around impurities (Newville)

And this became noticed and used by other EXAFS used in the US and Europe.

The UWXAFS Package

Around 1994 `Feffit` and `Autobk`, along with Bruce Ravel's `Atoms` (crystal structure to Feff.inp), Daniel Haskel's `Fluo` program, along with a few utilities became the *UWXAFS Package* that was licensed by U Washington and distributed together.

These were all Fortran 77 codes that ran as standalone programs with an input file that needed to be edited, no graphics capabilities, and output files that needed to be then used by some other program.

Basically, these programs were written by and for physics grad students of the 1990s:

edit a text file, run program, then plot results with gnuplot

counted as "User Friendly". And: no XANES analysis tools at all.

Moving towards GUIs and Interactive Processing

By 1994, we had moved from microVax/VMS to Unix (AIX then Linux), and were using Gnuplot for post-analysis graphics. Graphical operating systems (Windows, CDE) were becoming common, but text-based terminals were still common.

There was still experimenting with code and better interactivity within the group.

- Gnuplot was command-line-driven plotting and some light processing.
- Bruce Ravel had a (honestly, really good) EMACS mode for toggling between editing input files, running programs, and plotting results.
- there were at least 2 "interactive data shell" programs – neither very good – to do some light data processing (add spectra, interpolate, etc).
- there was a growing collection of Perl scripts for streamlining data processing "workflows".

So, we felt the analysis parts were pretty good, but the ease-of-use was not that great. And we knew it needed to be better for more people to use these tools.

But also: we were still grad students and needed to graduate, get post-docs, or jobs.

Ifeffit: Adding interactivity and scriptability

We all knew we needed GUIs and more interactive data analysis built on top of the UWXAFS codes. GUIs could be done in scripting languages that called C/Fortran:

- Tcl/Tk – this was dominant at the time.
- Perl/Tk – an alternative using the popular Unix glue language.
- Python/Tk – less well known but math-facing (complex numbers built in!).

Personal CV: I did a post-doc with PNNL/UW from June 1995 to June 1996, then a post-doc at LLNL in June, 1996. My first two weeks at LLNL, two things happened:

- Sasa Bajt walked into my office and told me I shoulg get a job with Steve Sutton and Mark Rivers at U. Chicago / APS. I agreed.
- I attended (listened) to a workshop on Python for scientific applications, including numerical arrays - precursor to NumPy.

I started a beamline scientist position at U. Chicago / APS in June, 1997.

During this time, I started working on an interactive program with basic commands to do processing. This extended Feffit's "constraint language" to be the language of the interactive Session all available as a simple library.

Interactive Feff Fitting $\rightarrow$ Ifeffit.   Also: completely Open Source.

Ifeffit development: 1996 to 2000

Ifeffit used more modern Unix configure/make tools (and g77 became usable). It was an interactive command-line tool - like Gnuplot - that included basic manipulation and simple plotting graphics, using the PGPLOT library.

It was also a library that could be called from C, Tcl, Perl, and Python to send in command strings and insert/extract data values and arrays.

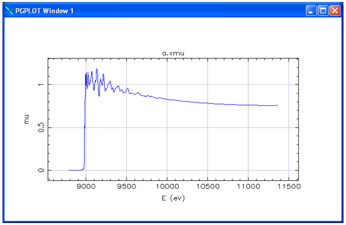
Example Ifeffit session

```
Ifeffit> read_data(file=Cu.dat,type=raw,group=cu)
Ifeffit> cu.energy=cu.1
Ifeffit> cu.xmu=ln(cu.2/cu.3)
Ifeffit> spline(energy=cu.energy, xmu=cu.xmu,
                      rbkg=1.1, kweight=1, kmin=0)
Ifeffit> plot(cu.energy, cu.xmu)
Ifeffit> plot(cu.energy, cu.bkg, xmin=8850, xmax=9300)
```

Commands could be entered at the command line for *Exploratory Data Analysis*.
Or could be sent from Perl, Python, Tcl, or C wrappers.

The main code was still in Fortran 77. We had a lot of existing code to re-use, and those Netlib libraries like MINPACK were just not available in C at the time.

Plotting worked on Unix, Linux, and Windows (and soon, Mac OS X), but the quality was not the highest quality.



Ifeffit GUIs: Athena, Artemis, SixPack 2001-2008

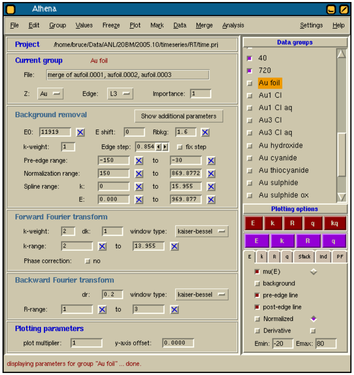
Separating the ifeffit "Session" from scripting language was a good for GUIs.
It gave a record of the "real work" that the GUI did that could be reproduced.

This also spread the work. I focused on the Ifeffit layer and Bruce Ravel focused on the GUIs Athena and Artemis with Perl/Tk. There was overlap and discussion, but this separation was helpful, especially in the pre-git era.

. . . and Sam Webb with SixPack, too.

Athena, Artemis, Hepheastus were pretty successful. ∼15,000 citations to date.

Starting in 1999, XAFS training short courses (2 to 5 days) featuring these programs started to happen, with ∼2 in the US, and often 2 more internationally each year.


Athena
File Edit Group Values Freeze Plot Mark Data Merge Analysis Settings Help
Data groups
Plotting options
E k R q kq
E k R q
E k R q Stack Ind PF
Show additional parameters

Project /home/bruce/Data/ANL/20BM/2005.10/timeseries/RT/time.prj

Current group Au foil
File: merge of aufoil.0001, aufoil.0002, aufoil.0003
Z: Au Edge: L3 Importance: 1

Background removal
E0: 11919 E shift: 0 Rbkg: 1.6
k-weight: 1 Edge step: 0.854 fix step
Pre-edge range: -150 to -30
Normalization range: 150 to 869.8772
Spline range: k: 0 to 15.955
E: 0.000 to 969.877

Forward Fourier transform
k-weight: 2 dk: 1 window type: kaiser-bessel
k-range: 2 to 13.955
Phase correction: no

Backward Fourier transform
dr: 0.2 window type: kaiser-bessel
R-range: 1 to 3

Plotting parameters
plot multiplier: 1 y-axis offset: 0.0000

Data groups: 40, 720, Au foil, Au1 Cl, Au1 Cl aq, Au3 Cl, Au3 Cl aq, Au hydroxide, Au cyanide, Au thiocyanide, Au sulphide, Au sulphide ox

mu(E), background, pre-edge line, post-edge line, Normalized, Derivative
Emin: -20 Emax: 80

displaying parameters for group "Au foil" ... done.

The *Ifeffit Mailing List*, started in February 2001 - and is still going.
800+ subscribers, discussing all aspects of XAFS, with input from many XAFS experts: Ravel, Rehr, Kelly, Frenkel, Calvin,, Marcus, G Bunker, Segre, Mosselmans, many more.

Ifeffit limitations

Athena and Artemis became very popular and helped expand the use of XAFS to more scientists, and make the existing and new XAFS beamlines more productive.

But problems with Ifeffit became apparent almost immediately.

- the command-line languages was far too limited. No “if-then-else”, no loops.
- the static memory from Fortran77 limited the number of spectra per session.
- the internal memory management was not very good.
- no linear XANES analysis (linear combinations, PCA). The GUIs added this.
- general-purpose peak-fitting was available, but not very good.
- the graphics were poor, and buggy on Windows.
- the build system was fragile
- basically no one besides me could really work on the Ifeffit code.
- basically no one besides Bruce could really work on the GUI code.

Even by 2002, we knew that these were serious problems, but there were no easy fixes.

But there was a common root cause: Fortran77.

Early attempts at using Python: 2002-2009

I had been playing around with Python since 1996 (Python 1.4).

The Ifeffit paper (*J. Synch Rad*, 2001) has Python code in a Figure to show calling from scripting language.

By 2002, I was using Python for our beamline data visualization (with Python/Tk, pre-matplotlib plotting), and had an interface to the Epics control system (later pyepics).

in 2001-2003, Tom Trainor, a post-doc in our group was interested in converting Matlab code to Python for Crystal-Truncation-Rod work, and wanted an Ifeffit-like interactive session and more complete "mini-language".

We explored parsing and language tools (Lex, Yacc, etc), but they were just not ready.

But by 2007, better tools were becoming available.

- numpy was established and well-supported.
- scipy included almost all of the NETLIB libraries we needed.
- matplotlib existed, was in active development, beautiful.
- wxPython was excellent for GUIs, and
- Python 2.6 had language-support tools for lexing/parsing AST .

And rewriting Ifeffit in Python and fixing all the mistakes became feasible, if a daunting task.

Python tools and Larch: 2009-2013

Because we had a working solution, and the goal was to improve it, I went slow, and embraced the open-source development, the “scipy community”, and GitHub.

Where possible we made separate Python libraries for different tasks.

`WXMPLOT`: wxPython-specific tools for matplotlib plotting and image display. Much higher quality graphics.

`XRAYDB`: X-ray properties and atomic scattering factors, including a web interface.

`ASTEVAL`: evaluates mathematical expression parsing as in Feffit, and the “mini-language” of Ifeffit. It includes loops, conditionals, and functions. It is a nearly complete language using Python objects and “safe-ish” replacement for “eval()”.

`LMFIT`: Levenberg-Marquardt fitting, with Feffit-like constraints using `asteval`, and named Parameters that can be fixed or bounded. Building on scipy, this allows many solving algorithms other than Levenberg-Marquardt, and has better bounds and tools for exploring and propagating uncertainties.

These are each separate Python projects (on GitHub) and are used by different, wider communities and also **with collaboration** from the wider scientific Python community.

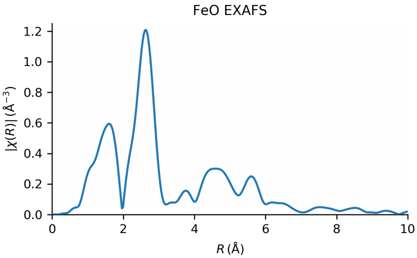
Fitting and uncertainties in Larch are much better because of the tools added to LMFIT from outside the XAFS community. The DOI for the Github Page has 1200 citations since 2016.

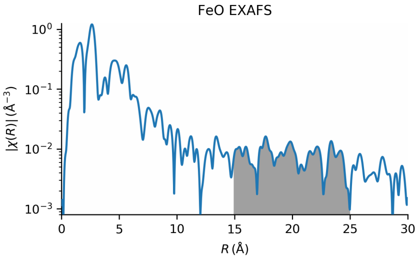
All these tools were developed to make XAFS analysis better.

Aside: Propagation of uncertainties in $\chi(k)$

Estimating uncertainties in EXAFS data has always been a challenge.

Feffit / Ifeffit estimated the uncertainty in $\chi(k)$ as *white noise* (Newville, Boyanov, and Sayers, *J Synch Rad*, 1999), using $\chi(R)$ between [15, 25] Å.





The "high-R" portion of $\chi(R)$ can estimate the "white noise" in the data pretty well. This is easy to do, but we have long-known that it misses an important component:

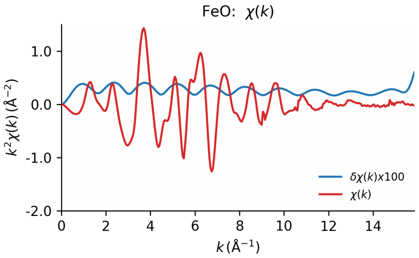
uncertainties from background subtraction

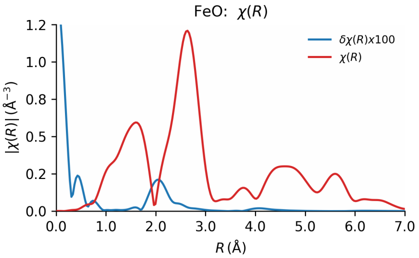
But: we were not even trying to really handle that numerically.

Uncertainties in $\chi(k)$ from background subtraction

With LMFIT we can easily propagate the uncertainties from the fit of the background spline to estimate the uncertainty in $\chi(k)$ due to the background subtraction.

As suspected, this is *not white noise*. In fact, it tends to have a peak around $2R_{\mathrm{bkg}}$!





Using this $\delta\chi(k)$ array typically reduces the fit χ^2 statistic by $\sim 3\times$ or even more.

This is now done by default.

Larch: 2012-2019

By 2012, most of the tools were in place to be able to replace Ifeffit with Larch.
This could be run:

- as a command-line function.
- with a primitive GUI – not much more than a command-line + data browser.
- with socket calls using XML-RPC from any language, or different machines – this was to allow Athena/Artemis to use it.

Features included (and with support from NSF CISE - Thanks!)

- no memory limitations.
- script in internal "Larch" language , Python, or with socket layer.
- readable code. The AUTOBK function was ∼150 lines of code.
- much better graphics.
- much better at pre-edge peak fitting.
- XANES linear analysis tools.
- includes FLUO, MBACK, DIFFKK.
- XRF analysis tools, and some XRD/CIF Processing tools too.
- often **faster** than Ifeffit at looping over datasets.

Development and maintenance for Ifeffit ended in 2013.

Larch GUIs: 2019-present

Larch-using GUIs for my beamline existed for many years:

- XRF spectral analysis
- XRF Mapping display
- Data collection
- "XAS Viewer": Athena-like viewer and pre-edge peak fitting.

By 2019, it was clear that Athena / Artemis development was stalled.
"XAS Viewer" – now "Larix" – can now replace Athena and Artemis:

- GUI browse and use 20,000+ CIF files from Am. Mineralogist.
- Convert CIFs or other XYZ / DFT files to Feff.inp.
- Principal Component Analysis, Linear Combinations, some ML-like Regression.
- Run Feff8l for EXAFS.
- Feff fit from the GUI.
- Processing history (Journal) for each dataset.
- Handling of Large data containers (HDF5/NeXuS).

There may be a few missing pieces, but many things are far better or simply unavailable (e.g., wavelets) in the Athena/Artemis.

There is active development on GitHub, outside contributions, and almost-regular Zoom meeting of interested developers.

EXAFS at 50 (and Feffit at 30): a Milestone

Pablo Casals, one of the foremost cellists of the 20th century, was asked in an interview why, at 65 years old, he still practiced cello four or fiver hours a day.

He answered, “Because I think I am making progress.”

EXAFS Analysis is still too hard. But I think we are making progress.