EASE Tutorial

Learning the EXAFS Analysis System for Emacs
Version 0.1 for EASE 0.6.0

Bruce Ravel

The author of EASE, Bruce Ravel, can be reached at:

`<ravel@phys.washington.edu>`

The latest version of EASE can always be found at
`http://feff.phys.washington.edu/~ravel/ease/`

1 Introduction

In this tutorial, I will lead you step by step through the analysis of some copper data. Along the way, we will explore many of the features of EASE . In this tutorial I assume that you, the reader, are familiar with the physics of EXAFS and the principles of EXAFS analysis. I also assume that you are familiar with FEFF and FEFFIT, although you certainly do not have to be an expert in their use. In fact, working through this tutorial is a good way to explore some of the features of those programs. I make some effort in this tutorial to accomodate the Emacs novice. It is possible that a reader may have been drawn to EASE despite not being a convert to the religion of Emacs.[1] For that reader, I try to offer enough guidance that she can get through the entire tutorial without having to know too many magic Emacs key sequences.

1.1 Using Emacs

As an aid to the Emacs-non-expert, I will explain some of the notation that I will use in the pages to come. Many tasks in Emacs (using EASE or performing any other editing chore) involce the use of modifier keys. When you read something like `C-c`, that means to hold down the ⟨ctrl⟩ key while hitting the `c` character. `C-c C-r r` then means to hit ⟨ctrl⟩ and ⟨c⟩ followed by ⟨ctrl⟩ and ⟨r⟩ followed by ⟨r⟩. In Emacs jargon `M-` is read as `Meta` and, on most keyboards, refers either to the ⟨alt⟩ or ⟨esc⟩ key. These two can be used interchangably, however these two keys typically behave somewhat differently. When you read `M-n` that means *either* to hit the ⟨alt⟩ and ⟨n⟩ keys at the same time *or* to hit the ⟨esc⟩ key followed by the ⟨n⟩ key. `S-` is interpretted the same as `C-` except that it means to hold the ⟨shift⟩ while hitting the following character. In a few cases, a key sequence may involve more than one modifier. `M-C-‘` means to hit ⟨‘⟩ (that is the backtick or open quote character) while holding down both ⟨alt⟩ and ⟨ctrl⟩.

I will assume that you are using a three-button mouse or a mouse capable of three-button emulation. Throughout this document I will refer to the mouse buttons as `mouse-1`, `mouse-2`, and `mouse-3`. typically `mouse-1` is the left button, `mouse-2` is the middle, and `mouse-3` is the right. Sometimes this arrangement is altered to suit the preferences of the user.

Most of the interactivity of Emacs occurs either via pop-up windows or through the minibuffer. Pop-up windows usually have text, buttons, and menus which are fairly self-explanatory. The minibuffer is that strip at the bottom of the Emacs frame below the mode line. That space is reserved for displaying messages and for interaction with the user. Many of the functions in EASE use this space for one of those purposes. The phrase *minibuffer* is used to refer to that space when Emacs is interacting with the user and the phrase *echo area* is used to describe it when it is displaying a message non-interactively. I will use both terms in this document.

There are four ways of making Emacs execute a command or function.

[1] It is not a ridiculous exageration to refer to Emacs as a religion. Many of its adamant users, your humble author included, use it for almost everything they do on a computer. Well, now the faithful can even analyze their EXAFS data. Nirvana!

- **Explicit invocation**: This is done by hitting *M-x* then typing the entire name of the function or command at the prompt which appears in the minibuffer. This method is guaranteed to work in all but the most perverse situations, but is not how the user typically interacts with Emacs.
- **Key sequences**: This is the fastest way of executing commands while editing text because it does not require moving your hands from the keyboard. Virtually every command in EASE has a key sequence. Key sequences are the keystrokes discussed above involving the ⟨ctrl⟩, ⟨alt⟩, and normal character keys.
- **Menu selection**: At the top of the Emacs frame is a menubar with pull down menus. These behave in the same manner as pull down menus in other programs and on other systems. Click `mouse-1` on the menu label and a menu will open. Click on an item in the menu and that function will execute or click on a submenu, usually denoted by a right-pointing triangle, and that submenu will open. Sometimes a menu item will be displayed in grey stipple. This means that option is not currently selectable. In Emacs, the key sequence bound to same function as a menu item is usually written in the right column of the pull down menu. The menus are great for searching for a certain function and also serve to teach you the key sequences for the functions you use often.
- **Toolbar selection**: One of the ways XEmacs differs from the Emacs written by the Free Software Foundation is that it can display images within the frame. It therefor offers a iconic toolbar. EASE binds several of the most commonly used functions to a toolbar that typically is displayed on the left side of the frame. Click `mouse-1` on a toolbar icon to execute the function bound to it. A short description of the function appears in the echo area whenever the mouse is over the toolbar icon.

If you have a Emacs open in front of you, click `mouse-1` on the `Input` menu. The contents of this menu are shown in Figure 1. This menu and its submenus will be referred to throughout this tutorial. If you click any mouse button anywhere else on your screen, the menu will disappear without anything selected.

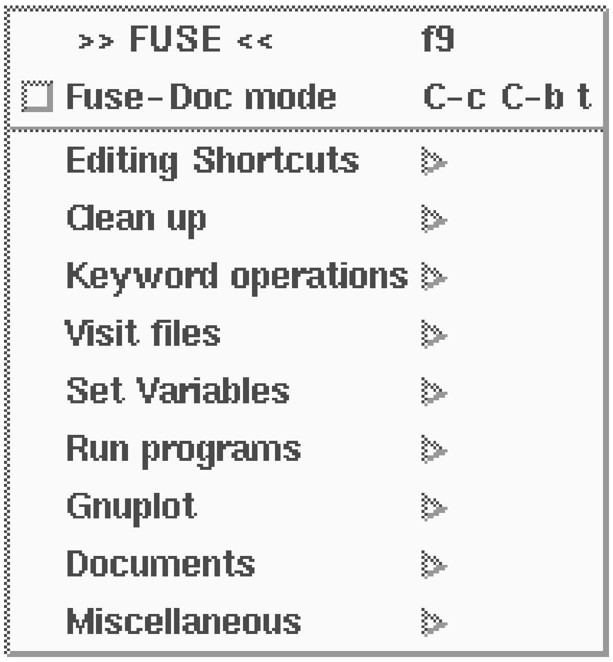
1.2 Starting an Analysis Problem with EASE

In the `example` subdirectory of the EASE distribution are two files related to this tutorial. (1) '`cu10k.dat`' contains copper K edge transmission data on a pure copper foil. (2) '`copper`' contains crystallographic data for metallic copper.

Both analysis of data and the use of EASE are facilitated by practical and consistent use of th directory structure and disk space where you do you work. Throughout this example I will refer to files foudn in certain locations. These examples reflect my own preference for how to organize the many files generated by the various programs used to analyze the data. Although I encourage you to come up with practices for managing files with which you are comfortable, I suggest that you follow my lead during this example and experiment once you are more familiar with EASE .

Remember that the the `Buffers` menu or the key sequences *C-x b* or *C-x C-b* can be used to switch between different buffers. A file is saved by hitting *C-x C-s* or selecting the save function from the `Files` menu. A new file is opened by hitting *C-x f* or selecting `Open...` from the `Files` menu. Finally, you can quit Emacs by hitting *C-x C-c* or `Exit Emacs` from the `Files` menu.

Figure 1: Input mode pull-down menu



Before launching into using EASE , let's set up some workspace. I am writing this assuming you are parked at a command line in a terminal emulator, but these disk management chores could just as easily be done with a file manager such as The Midnight Commander or tkDesk or with the `dired` package in Emacs. On my computer, I have a directory where I do all of my analysis which is called, appropriately enough, '`analysis/`'. Find a place where you want to do the analysis chore in this tutorial and *cd* to that directory. Now make a directory to hold your analysis of the copper data by issuing the command

```
> mkdir Cu
```

Now *cd* to the '`Cu/`' directory. This will be our base of operations for the remainder of this tutorial.

I like to be tidy and place common kinds of files in their own subdirectories. in the '`Cu/`' directory, issue this command

```
> mkdir data feff fits
```

This will make three subdirectories for holding the differnt kinds of files that will be generated throughout this tutorial.

Now, using the *cp* command, copy the file '`cu10k.dat`' from the EASE distribution into the '`data/`' subdirectory and copy the '`copper`' file into the '`feff/`' subdirectory.

Ready for some analysis?

1.3 A Few Notes About this Document

This document was prepared using the GNU Texinfo system. This allows me to generate documentation in info, printed, or html format from a single source. The info, postscript, and html versions of this document can be found in the '`docs/`' subdirectory of the EASE distribution.

The images in this document are screen shots of an XEmacs session on my computer. I use Linux 2.0.32, XEmacs 20.4, and the FVWM2 window manager. Some of the text in shots of actual input files is a bit hard to read in the printed manual. This is because of the greyscale representation of the syntax colorization of the text. It may be fuzzy in the printed manual, but it looks great on the screen!

Although all of my screen shots are of an XEmacs session, EASE works just as well with FSF Emacs as with XEmacs. The only difference significant to this tutorial is the presence of the toolbar in the XEmacs frame. Since all functionality bound to the XEmacs toolbar is also bound to pull-down menus, this tutorial proceeds identically for both flavors of Emacs.

2 Running Atoms

2.1 Starting with ATOMS

Ultimately we want to use FEFFIT analyze the fine structure $\chi(k)$ from the copper data. To get ready for that we need to run FEFF to generate a set of fitting standards and run AUTOBK to isolate $\chi(k)$ from $\mu(E)$. But first, we need to run ATOMS to generate the appropriate input file for FEFF.

Our first chore, then, is to edit an ATOMS input file. It is most convenient to run ATOMS in the '`feff`' subdirectory, so begin editing this input file by typing

```
> emacs feff/atoms.inp
```

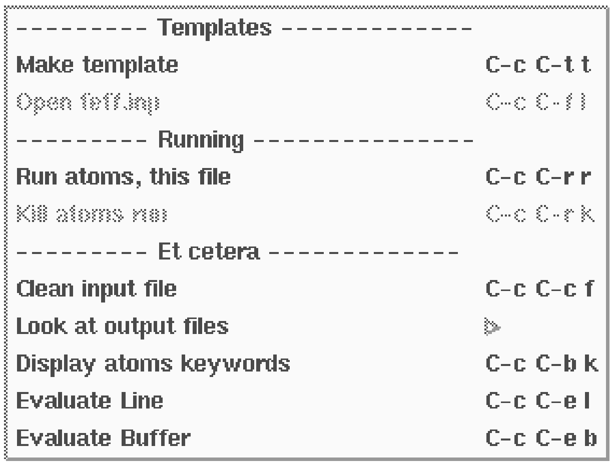
at the command line. I am assuming that you are currently in the '`Cu/`' directory. Since it takes some time to start up Emacs and since Emacs can hold many files at the same time, EASE is best used by firing up Emacs only once and loading successive input files as you want to edit them. We will see how that is done as the tutorial progresses.

Once Emacs fires up, you will be presented with a blank screen. You should see two important things on the mode line at the bottom of the Emacs window. Near the left side of the mode line it should say `atoms.inp` to indicate that you are currently editing that file. Somewhere to the right of center you should see the words `Input` and `Atoms`. These indicate that you are in `Input` major mode and, since you are editing an ATOMS input file, in `Atoms` minor mode. At the top of the Emacs window in the menu bar, you should see menus labeled `Input` and `Atoms`. Because the file was named '`atoms.inp`', EASE assumed that the file is intended to run ATOMS and so placed the buffer containing the file into `Atoms` minor mode. As the tutorial progresses and we edit other input files, you will see that the minor mode changes appropriately from buffer to buffer.

In Emacs, a *major mode* is a state of the program which is customized for a particular editing chore. A *minor mode* is a state which modifies certain features and behaviors of Emacs. In this case the major mode is a state customized for editing input files to FEFF and the other programs. Then each program has a minor mode associated with it which modifies the behavior of input major mode appropriate to the program. The modes in EASE interacts well with most minor modes which offer specific editing features which are offered by emacs.

Now that you have a buffer open in front of you, click `mouse-1` on the `Atoms` menu. The contents of this menu are shown in Figure 2. If you click any mouse button anywhere else on your screen, the menu will disappear without anything selected. You can use this menu to perform the rest of the tasks in this section.

Figure 2: ATOMS mode pull-down menu


--------- Templates -------------
Make template C-c C-t t
Open feff.inp C-c C-f i
--------- Running ---------------
Run atoms, this file C-c C-r r
Kill atoms run C-c C-r k
--------- Et cetera -------------
Clean input file C-c C-c f
Look at output files
Display atoms keywords C-c C-b k
Evaluate Line C-c C-e l
Evaluate Buffer C-c C-e b

2.2 Making the ATOMS template

The first thing to do is to make a template for this input file. This is done by typing *C-c C-t t*, choosing `Make template` from the `Atoms` menu, or clicking on the top-most icon in the toolbar. When you do this, you will be asked a question in the minibuffer. At this point, EASE is asking which crystal class your material belongs to. For copper, the correct answer is `c` which is short for *cubic*. Before answering you can hit the ⟨tab⟩ key to see a list of all possible answers. This function (and many others in EASE) operates with word completion. Since cubic is the only crystal class beginning with `c`, it suffices to just enter `c` in this case. After this, a template appears in the buffer. You will notice that this template has a space for the a lattice constant, but not for b or c or for any of the angles. Had you chosen a crystal class of lower symmetry, appropriate keywords would have been placed in the template.

Notice that the template has several salmon colored markings. These are immediately before hotspots in the template. A hotspot is a place in the template where you are expected to insert a value. The markings provide a visual cue for where to place the values. You can move between the hotspots in all the standard ways of moving the screen cursor, but there are two special shortcut key sequences. *M-n* and *M-p* jump to the next and previous hotspots. When you are done filling in the keyword values, you can leave the markings as they are, or erase them with *C-c C-t c*.

The a lattice constant of copper is 3.61 and the space group is `F m 3 m`, although ATOMS also understands `fcc` as a shorthand for this space group. There is a copper atom at the coordinates (0,0,0) and a cluster radius of 6.5 Angstroms is appropriate. The central atoms is, obviously, Cu. Save this file by striking *C-x C-s*. The buffer should now look like Figure 3.

Notice that when you saved the file, some new lines were written to the end of the file. Thie first line looks like this:

```
!!&& Local Variables:
```

EASE uses the Local Variables list as a way of configuring itself. It saves information about the input file in these lines so that the next time you edit it, you will be in the same state as the last time. One of the most important uses of these lines is to identify the program associated with the input file. For example, you might wish to rename this input file '`cu.inp`'. The next time you edit '`cu.inp`', EASE can read from the Local Variables list that '`cu.inp`' is an input file for ATOMS. Thus you can name your input files anything, not just by the name of the program.

2.3 Other Features of ATOMS mode

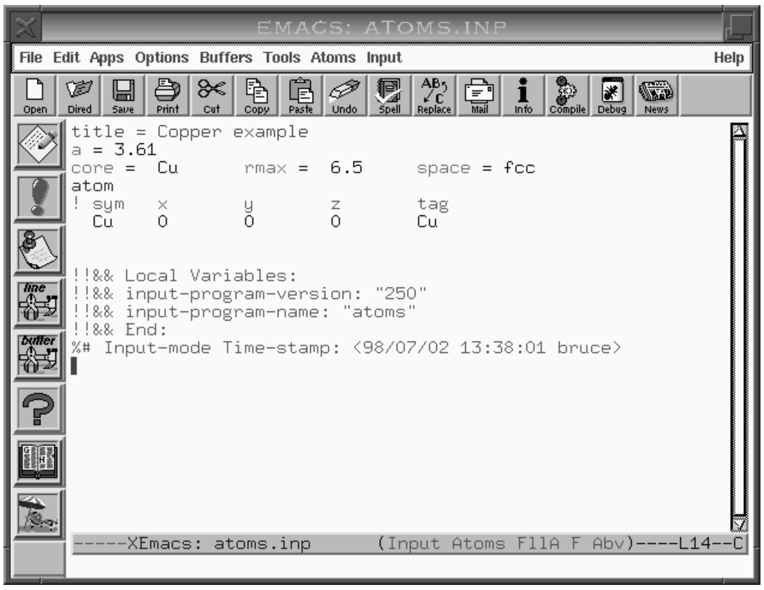
With the input file complete and saved to disk, it is time to run ATOMS. Do this by hitting *`C-c C-r r`*, by selecting `Run atoms, this file` in the `Atoms` menu, or by clicking on the exclamation point icon in the toolbar. If you are running under X-windows, EASE will open a small frame in which to display the run-time messages from the program. When ATOMS is done, EASE will beep and you will see a message in the echo area.

ATOMS Let's look at a few things before finishing with the ATOMS input file. Type *`C-c C-b k`*, select `Display atoms keywords` from the `Atoms` menu, or click on the question mark icon in the toolbar. The frame will split into two windows, and a list of keywords recognized by the ATOMS program will be displayed in the bottom window. When that key sequence is hit in any of the minor modes, you will be presented with a list of keywords appropriate to that program. Each minor mode menu has a `Display keywords` item and each minor mode toolbar has a question mark icon that is bound to this function. Note that the screen cursor in positioned in the keyword window. After examining the list of keywords, type *`C-x 0`* to close the keyword window and return to the input file window. (If you get lost among your buffers, remember to use *`C-x b`* or the `Buffers` menu.)

Now hit *`C-c C-b d`* and then hit ⟨ret⟩ or click on the book icon in the toolbar. A second frame will open displaying the info version of the ATOMS document. There are online documents for several of the programs covered by EASE and for EASE itself. (The notable exceptions are AUTOBK and FEFFIT.) You can peruse the document or hit *`C-x 5 0`* to make the document frame disappear.

Now it is time to run FEFF.

Figure 3: Completed ATOMS input file


EMACS: ATOMS.INP
File Edit Apps Options Buffers Tools Atoms Input Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News

```
title = Copper example
a = 3.61
core =  Cu       rmax =  6.5      space = fcc
atom
! sym    x        y        z        tag
  Cu     0        0        0        Cu


!!&& Local Variables:
!!&& input-program-version: "250"
!!&& input-program-name: "atoms"
!!&& End:
%# Input-mode Time-stamp: <98/07/02 13:38:01 bruce>
```

```
-----XEmacs: atoms.inp      (Input Atoms F11A F Abv)----L14--C
```

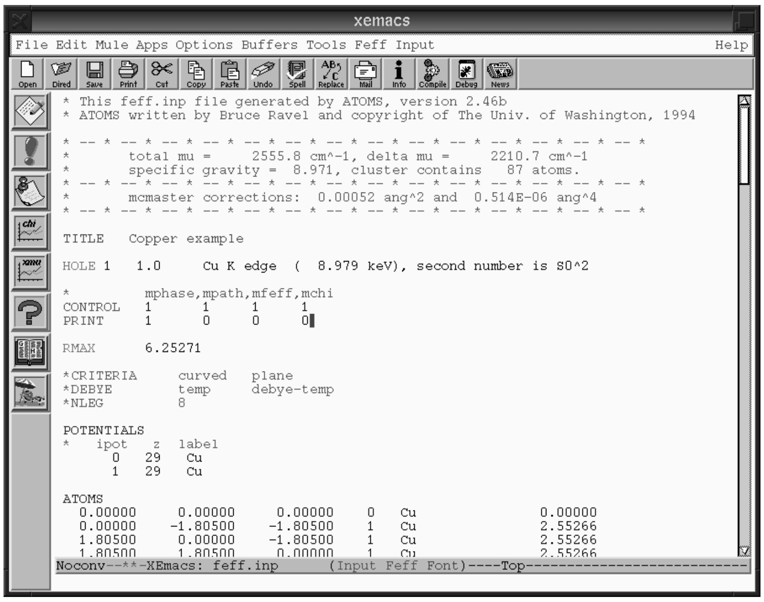
3 Running Feff

3.1 The FEFF Input File

EASE has a quick way of jumping to the FEFF input file generated by an ATOMS run. Type *C-c C-f l*, select `Open feff.inp` from the `Atoms` menu, or click `mouse-1` on the third icon from the top of the toolbar. The '`feff.inp`' file will be displayed in place of the '`atoms.inp`' file. Notice that the mode line and the menu bar have both changed to reflect the fact that you are now editing a feff input file.

For the sake of this tutorial, you will need to alter '`feff.inp`' in only one way. Change the value of the '`PRINT`' keyword to '`1 0 0 0`'. This is done to be sure that the '`misc.dat`' file is written to disk by FEFF. Of course FEFF has many more options for altering its behavior, but exploring them is beyond the scope of this tutorial. We will now run FEFF. This is done is exactly the same way as we ran ATOMS. In fact, the manner in which a program can be run is independent of which program will be run. The key sequence *C-c C-r r* always runs the appropriate program for the current file. There is always an option in the program specific menu labeled '`Run program, this file`'. Also the `Input` menu has an option labeled '`Run this program, this file`' which also works. Finally, there is always a red exclamation point toolbar icon in XEmacs.

Figure 4: The first few lines of the FEFF input file for copper


xemacs
File Edit Mule Apps Options Buffers Tools Feff Input Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News

```
* This feff.inp file generated by ATOMS, version 2.46b
* ATOMS written by Bruce Ravel and copyright of The Univ. of Washington, 1994

* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *
*       total mu =     2555.8 cm^-1, delta mu =     2210.7 cm^-1
*       specific gravity =  8.971, cluster contains   87 atoms.
* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *
*       mcmaster corrections:  0.00052 ang^2 and  0.514E-06 ang^4
* -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- * -- *

TITLE   Copper example

HOLE 1   1.0     Cu K edge  (  8.979 keV), second number is S0^2

*         mphase,mpath,mfeff,mchi
CONTROL   1      1      1      1
PRINT     1      0      0      0

RMAX      6.25271

*CRITERIA     curved   plane
*DEBYE        temp     debye-temp
*NLEG         8

POTENTIALS
*   ipot   z  label
      0   29   Cu
      1   29   Cu

ATOMS
   0.00000     0.00000     0.00000    0    Cu               0.00000
   0.00000    -1.80500    -1.80500    1    Cu               2.55266
   1.80500     0.00000    -1.80500    1    Cu               2.55266
   1.80500     1.80500     0.00000    1    Cu               2.55266
Noconv--**-XEmacs: feff.inp      (Input Feff Font)----Top--------------------------
```

Notice that the run-time messages from FEFF are appearing in the run-time buffer after those from ATOMS. During your analysis session, this buffer serves as a log of your work.

When you are finished with analysis and exit Emacs, the contents of this buffer[1] will be automatically saved in a file called '`~/.ease-run.log`'.

Running FEFF takes a few minutes, so this is a good time to stretch your legs.

3.2 Other Features of FEFF mode

If you do a directory listing on the '`feff`' subdirectory, you will see that FEFF has written out many files. EASE has several tools for examining the output of your '`feff`' run. In the '`Feff`' menu is a sub-menu labeled '`Look at output files`'. In that sub-menu you can choose to examine any of '`misc.dat`', '`paths.dat`', '`files.dat`', or '`list.dat`'. The key sequences for these are *`C-c C-f`* followed by *`m`*, *`p`*, *`f`*, or *`s`* respectively. Choosing any of these will cause that file to be visited and displayed in a read-only buffer. EASE assumes that files jumped-to in this manner are are log files and thus should not be casually altered. That is why they are placed in read-only buffers. If you want to edit one of these jumped-to files, type *`M-x toggle-read-only`*.

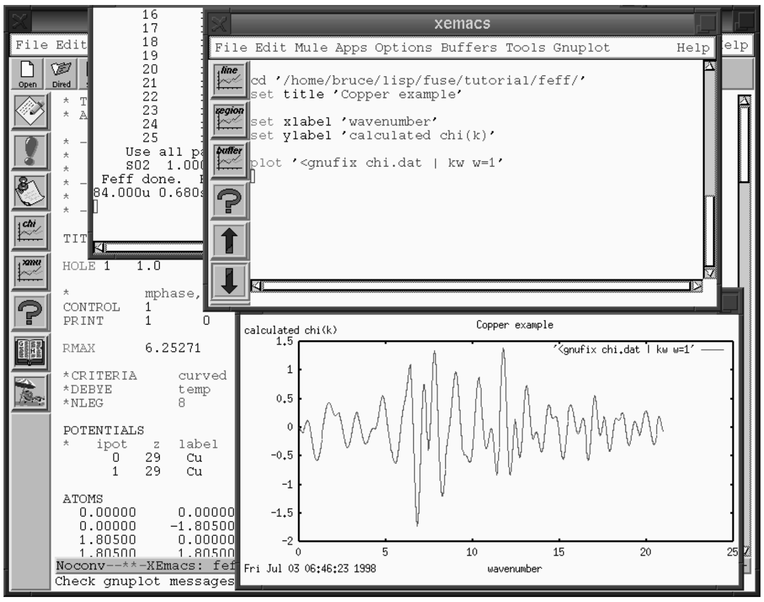
Because these files are all rather cryptic, EASE comes with a handy perl script called INTRP. INTRP reads '`feff.inp`', '`paths.dat`', and '`files.dat`' and writes a summary of all of the scattering paths from the FEFF calculation. INTRP can be run from the '`Look at output files`' submenu or by hitting *`C-c C-f i`*. The INTRP buffer is very useful for setting up a FEFFIT input file.

EASE will display FEFF results graphically by constructing scripts for the GNUPLOT program then sending those scripts to GNUPLOT. These scripts are generated automatically, usually using information contained ni the input file itself. We can examine the total $\chi(k)$ from the FEFF calculation by selecting '`Plot chi`' from the '`Feff`' menu, typing *`C-c C-p c`*, or clicking on the graph icon labeled `chi`. When you do this, a new frame will appear on the screen containing the GNUPLOT script. The buffer containing the script will be in a major mode specifically designed for GNUPLOT scripts (see Chapter 6 [Plotting with Gnuplot], page 21). After a short time, a window displaying the GNUPLOT plot will appear. This is shown in Figure 5.

Notice that, in the plot, $\chi(k)$ is weighted by k. Examinnig the script, you see that the '`chi.dat`' file was filtered through two scripts. GNUFIX is a SED script which comments out the header material in '`chi.dat`' so that it will plot properly in GNUPLOT. KW is an AWK script that actually applies the k-weighting. Both of these scripts are found in the '`scripts/`' subdirectory of the EASE distribution and are available for your use outside of EASE .

[1] Actually only the most recent 1000 lines.

Figure 5: Plotting the calculation



```
     16
     17
     18
     19
     20
     21
     22
     23
     24
     25
 Use all pa
  S02  1.000
Feff done.  F
84.000u 0.680s
```

```
* T
* A

* _
*
*
* _
*
* _

TIT

HOLE 1   1.0

*            mphase,
CONTROL      1
PRINT        1        0

RMAX         6.25271

*CRITERIA        curved
*DEBYE           temp
*NLEG            8

POTENTIALS
*    ipot   z   label
       0   29    Cu
       1   29    Cu

ATOMS
   0.00000       0.00000
   0.00000      -1.80500
   1.80500       0.00000
   1.80500       1.80500
Noconv--**-XEmacs: fef
Check gnuplot messages
```

```
cd '/home/bruce/lisp/fuse/tutorial/feff/'
set title 'Copper example'

set xlabel 'wavenumber'
set ylabel 'calculated chi(k)'

plot '<gnufix chi.dat | kw w=1'
```

You can change the k-weighting easily by selecting '`Set k-weight`' from the '`Input`' menu or typing ***`C-c C-d k`***. You will asked for a k-weight value in the minibuffer. Answer with 2 or any other number then replot chi with the new k-weight.

The other plotting option in FEFF mode is to plot $\mu(E)$ together with the background function. This can only be done is the **`XANES`** keyword is set in the input file and a version of FEFF is used that calculates XANES. This was not done in this tutorial, but feel free to play with this later.

4 Running Autobk

4.1 The AUTOBK Input File

To begin editing an AUTOBK input file, type *C-x C-f* or select `Open File` from the `Files` menu. You will be asked for the name of a file in the '`feff`' subdirectory. Before responding with a name, delete the characters '`feff/`' so that you will open a file in the '`Cu`' subdirectory. Then type out '`autobk.inp`'. You will be presented with a blank window. Notice that you are now in `Autobk` minor mode, the menu bar has the `Autobk` label, and there is an `Autobk` menu in the menu bar.

Before writing a template, let's set several variables that will be useful in the next few steps of the tutorial. In the `Input` menu you will notice options for `Set path to input data`, `Set path to feff files`, and `Set output path`. The key sequences for these are *C-c C-d* followed by *d*, *f*, and *o* respectively. These are used to tell EASE where to find different sorts of files. Run each of these functions using '`data/`' as the data path, '`feff/`' as the feff path, and '`data/`' as the output path.

Now insert a template by typing *C-c C-t t*, selecting '`Autobk template`' from the '`Autobk`' menu, or clicking `mouse-1` on the top-most toolbar icon. Note that the information about file paths that you just entered has appeared in appropriate places in the template. Note that, just as with the ATOMS template, hotspot markings have been placed in the template as a visual cue for where keyword values must be inserted. If you enter reasonable data at each hotspot, it is quite likely that your input file will run to completion.

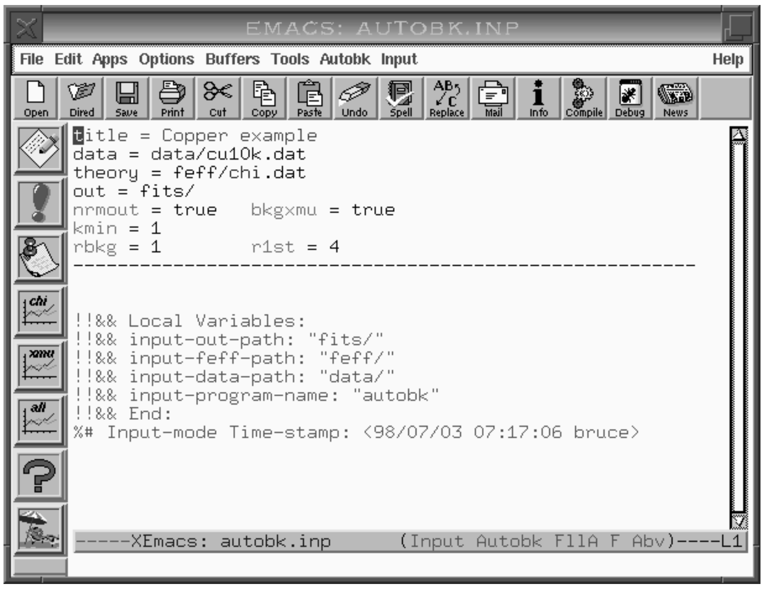
Now fill in appropriate values at each hotpot. Use values of *1*, *1*, and *4* for `kmin`, `rmin`, and `rmax` respectively. Write '`cu10k.dat`' after '`data/`' for the `data` keyword and '`chi.dat`' after '`feff/`' for the `theory` keyword. Pick a good basename for the output files, '`cu10k`' should do just fine. You can delete the line with `e0` as we will let AUTOBK find it. Now place the screen cursor on one of the keywords, `kmin` for example. Hit *M-?*. This command causes EASE to display a description of the keyword under the screen cursor in the echo area. Now hit *M-⟨ret⟩*. If you put a number in as the value for *kmin*, you will see a message in the echo area saying that this value is ok. Try changing the value of `kmin` to *q* and hitting *M-⟨ret⟩* again. Don't forget to change it back! Now place the screen cursor on the `data` keyword and try *M-?* and *M-⟨ret⟩* again. Notice that the keyword checking facility knows about and checks for readable files. If you like the help given by *M-?*, you can have it happen automatically by hiting *C-c C-b t* or selecting `Ease-doc mode` from the top of the `Input` menu. With ease-doc mode turned on, a keyword description will automatically be displayed in the echo area whenever the screen cursor comes to rest on a keyword. This works in any program minor mode. The messages displayed are the same as those displayed by *C-c C-b k* or the question mark icon in the toolbar.

Now finish filling in the template and save it. It should look like Figure 6.

Several interesting things happened when you saved this file. The Local Variables list was written to this files just as it was to '`atoms.inp`' and '`feff.inp`'. Notice that the values for file paths that you entered earlier were written to the list. The next time that you edit this file, it will not be necessary to re-enter that information. EASE will read it from the Local Variables list. Also note that a time stamp was placed after the Local Variables list.

This is updated every time you save the file so you will always know when you last edited the file. Now move the mouse cursor so it is over the line that specifies the data file. See how the background flashes orange as soon as the cursor is over either the keyword `data` or the filename? This orange flashing is a visual cue in EASE that the mouse cursor is in a position where hitting `S-mouse-3` will do something. That something depends on the kind of input file you are editing and on the location of the mouse cursor. In this case, hitting S-MOUSE-3 will open and display the file under the cursor. These orange areas are set when the input file is first visited and whenever it is saved. You can turn off the orange flashing for a given buffer in the `Input - Miscellaneous` submenu.

Figure 6: Completed AUTOBK input file


EMACS: AUTOBK.INP
File Edit Apps Options Buffers Tools Autobk Input Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News

```
title = Copper example
data = data/cu10k.dat
theory = feff/chi.dat
out = fits/
nrmout = true   bkgxmu = true
kmin = 1
rbkg = 1        r1st = 4
----------------------------------------------------------


!!&& Local Variables:
!!&& input-out-path: "fits/"
!!&& input-feff-path: "feff/"
!!&& input-data-path: "data/"
!!&& input-program-name: "autobk"
!!&& End:
%# Input-mode Time-stamp: <98/07/03 07:17:06 bruce>

-----XEmacs: autobk.inp      (Input Autobk FllA F Abv)----L1
```

4.2 Running and Plotting in AUTOBK Mode

Now run AUTOBK. Note that the run-time messages from AUTOBK will be added to those already in the run-time messages buffer. Also note that the elapsed time of the run is written to the buffer in the run-time frame. This is printed out in the form of your computers '`time`' shell command. On my Linux system using bash, it looks like this:

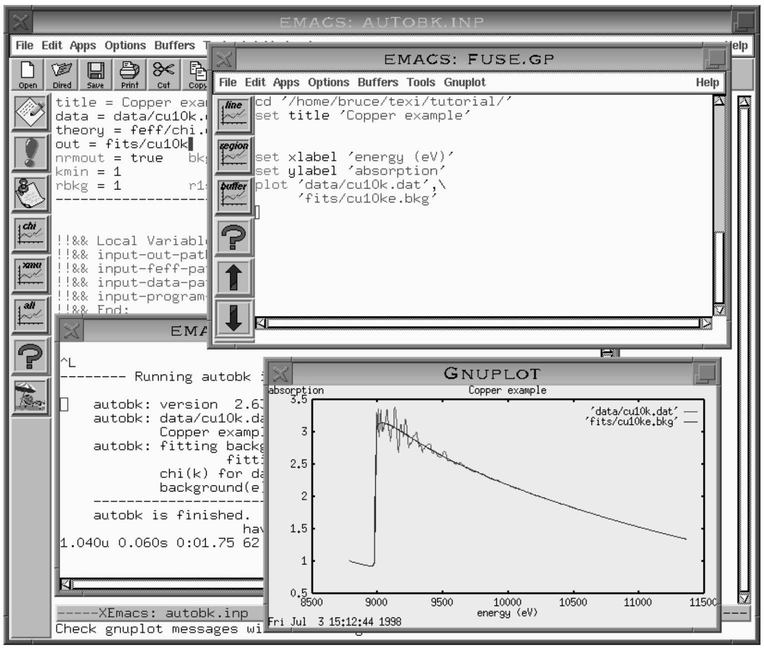
```
193.560u 0.160s 0:09.25 97.2% 0+0k 0+0io 171pf+0w
```

The relevant number is the third one. This says that that particular AUTOBK run took 0 minutes and 9.25 seconds. Check the man pages for your shell for complete details about the '`time`' command.

Once the run is finished (it should not take long), there are several things that can be plotted. In the `Autobk` menu there are several entries under the heading `Plotting`. The things that can be displayed automatically from this input file are the $\mu(E)$ data along with the background (`C-c C-p b` and shown in Figure 7), $\chi(k)$ (`C-c C-p k`), and $\chi(k)$ along with the theoretical $\chi(k)$ from the FEFF run (`C-c C-p t`). The $\mu(E)$ and $\chi(k)$ plots are also bound to toolbar icons.

Although we will not be using this feature of AUTOBK during this tutorial, it has the ability to batch process $\mu(E)$ data into $\chi(k)$. The instructions then look the same for each data set and are separated in the input file by lines of dashes. In EASE , a line-of-dashes separated set of instructions is called a *stanza*. AUTOBK mode in EASE has several functions for operating on stanzas. In the **Autobk** menu there are options for running AUTOBK on the entire file or on the stanza currently occupied by the screen cursor. The plotting functions described in the last paragraph all work on the current stanza. Additionally there is a plotting function for displaying all $\chi(k)$ from the input file (`C-c C-p a` and bound to the toolbar).

Figure 7: Plotting the data and background



```
title = Copper exa
data = data/cu10k.
theory = feff/chi.
out = fits/cu10k
nrmout = true    bk
kmin = 1
rbkg = 1          r1
-------------------


!!&& Local Variabl
!!&& input-out-pat
!!&& input-feff-pa
!!&& input-data-pa
!!&& input-program
!!&& End:
```

```
cd '/home/bruce/texi/tutorial/'
set title 'Copper example'


set xlabel 'energy (eV)'
set ylabel 'absorption'
plot 'data/cu10k.dat',\
     'fits/cu10ke.bkg'
```

```
^L
-------- Running autobk i

   autobk: version  2.6
   autobk: data/cu10k.da
           Copper exampl
   autobk: fitting backg
                  fitti
           chi(k) for da
           background(e
   --------------------
   autobk is finished.
                  hav
1.040u 0.060s 0:01.75 62
```

```
-----XEmacs: autobk.inp
Check gnuplot messages wi
```

4.3 Examining the output of AUTOBK

Plotting the output files from AUTOBK is an important part of evaluating and understanding the result, but there is more to the interpretation of data than making pretty pictures. For example, from an AUTOBK run, it is important to know where E_0 is. Hit *`C-c C-s e`*. A line with `e0f = 8976.886` is inserted in the stanza. Cool! Where did that come from?

Now hit *`C-c C-f l`*, select `Look at log file` from either the `Autobk` or `Input - Visit files` menu, or hit the third toolbar icon (the one with the thumbtack). Doing so displays the log file from your autobk run. Somewhere around line 15 in the log file is the final value of E_0 found by AUTOBK. The *`C-c C-f l`* key sequence and its related menu and toolbar bindings are the common way of examining log files in EASE . That key sequence and the thumbtack icon will always display the log file appropriate to the current input file.[1] You can easily return to the input file by hitting *`M-C-‘`*.

EASE is clever enough to search the log file for the final value of E_0 and insert it into the input file. The `e0f` keyword tells AUTOBK to fix E_0 to that value. If you want to play around with other background removal parameter while keeping E_0

fixed, `e0f` is the right keyword. In the next chapter you will learn of other ways that EASE is able to glean information from log and other files for use in input files.

[1] In some modes the definition of a *log file* might be a little different. For example, in ATOMS mode, the log file is the '`feff.inp`' file.

5 Running Feffit

FEFFIT is the program covered by EASE which has the most complicated input files. Consequently, the most interesting and powerful features of EASE are designed explicitly for the FEFFIT input file.

Before starting in on creating the input file for FEFFIT, I want to define an important term – the *path paragraph*. In FEFFIT each scattering path is described by one or more *path parameters*. The path parameters are a set of keyword that tell FEFFIT how to calculate the contribution from a given scattering path. They have a common syntax. Each path parameter must be on its own line and the parameter itself must be the first word on the line. The second word on the line is an integer index identifying the path. The rest of the line is devoted to the value of the parameter. In FEFFIT there are no further syntactic requirements[1]. Specifically, they can appear in any order in the input file and FEFFIT will happily process them.

EASE adds an additional syntax requirement to the FEFFIT input file. EASE requires that all path parameters used to describe a given scattering path be contiguous in the file and that the `path` (or `feff`, they are synonyms) path parameter be the first first listed in the contiguous group. These groupings of path parameetrs are separated by one or more lines containing only white space or comment characters. White space in FEFFIT is defined as spaces or tabs and comment characters are any of `%`, `!`, `#`, or (at the beginning of a line) `*`. This grouping beginning with the `path` path parameter and ending with a line of white space or comment characters is called a *path paragraph*. EASE has numerous functions for operating on path paragraphs. If you prefer to organize your input file by parameters rather than by paragraphs, the you will probably not find EASE very helpful.

5.1 Creating the FEFFIT Input File

Writing a new input file for FEFFIT can be very tedious, time consuming, and error prone. To my mind, a task that is as inherently repititious as constructing a brand new FEFFIT input file begs for automation. EASE to the rescue.

Start by creating a new file. Use `C-x f` to tell Emacs to create a new file. When it asks for the file name, answer with '`feffit.inp`'. Note that the mode line now reads `Feffit` and there is a `Feffit` menu in the menu bar. The first thing you need to do is set the paths to the data, FEFF, and output files by hitting *`C-c C-d`* `a` or selecting `>> Set all three paths` from the `Input - Set variables` menu. Set them to '`data/`', '`feff/`', and '`fits/`' resprectively. Don't skip this step! Correctly setting these file paths will save you an *enormous* amount of typing in a few minutes.

Now type *`C-c C-f f`* or select `files.dat` from the `Feffit - Examine output from Feff` menu. This will display '`files.dat`', one of the output files from the FEFF run. Notice that this file actually bears a resemblence to a '`feffit.inp`' file. It contains a list of the file names that are needed as the values of the `path` path parameters and it shows some relevant information about each of those path files. Now type *`M-C-'`* to return to the '`feffit.inp`' file.

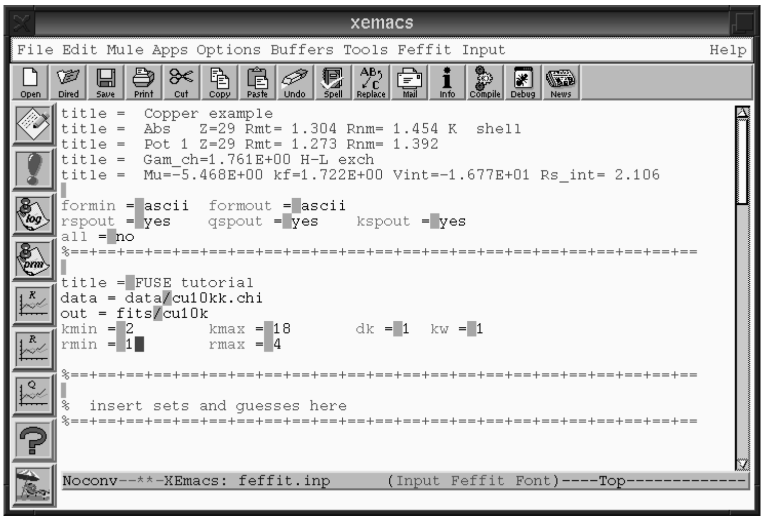
[1] Well, the math expressions that are the value of several of the keywords have certain syntactic requirements.

Now comes some wizardry. Type *C-c C-t f* or select `Make feffit.inp from files.dat` from the `Feffit - Templates` menu. (You can also hit the templates icon in the toolbar, but the behavior of that button may not make sense to you yet.) You will be asked in the minibuffer for the name of the '`files.dat`' file. The initial guess will be correct since you set the path to the FEFF files. Just hit ⟨ret⟩. EASE will display the contents of '`files.dat`' file and ask you a slightly cryptic question about the `minimum amplitude.` Just hit `!` and ⟨ret⟩. After working for a few second, the '`files.dat`' file is transformed into a skeleton for a FEFFIT input file. Wow!

The first several lines are title lines containing the header information from the '`files.dat`' file. Then templates for global and local variables were inserted into the input file. Note that the hotspot markers have been placed in the templates. The remaining lines are path paragraphs formed from the lines in '`files.dat`'. Note that the path to the '`feffnnnn.dat`' files was inserted in each path paragraph. The small chore of setting that path a few minutes ago has saved you a huge amount of dull typing. Yay! You actually have some control over how many lines from '`files.dat`' are written out as path paragraphs. The minimum amplitude question was giving you the option of discarding small paths by specifying an amplitude cutoff. Other questions would have given you options to discard paths beyond a certain length or containing too many legs. By hitting `!`, you told EASE to use defaults for these three cutoffs, that is not to discard any files. Later on, you may want to play with this function to see how these filters behave.

Now fill out the header templates so that they look like Figure 8.

Figure 8: The headers of the FEFFIT input file


xemacs
File Edit Mule Apps Options Buffers Tools Feffit Input Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Info Compile Debug News

```
title =  Copper example
title =  Abs   Z=29 Rmt= 1.304 Rnm= 1.454 K  shell
title =  Pot 1 Z=29 Rmt= 1.273 Rnm= 1.392
title =  Gam_ch=1.761E+00 H-L exch
title =  Mu=-5.468E+00 kf=1.722E+00 Vint=-1.677E+01 Rs_int= 2.106

formin = ascii  formout = ascii
rspout = yes    qspout = yes    kspout = yes
all = no
%==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==

title = FUSE tutorial
data = data/cu10kk.chi
out = fits/cu10k
kmin = 2         kmax = 18        dk = 1   kw = 1
rmin = 1         rmax = 4

%==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==

%  insert sets and guesses here
%==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==+==

Noconv--**-XEmacs: feffit.inp      (Input Feffit Font)----Top------------
```

5.2 Finishing the FEFFIT Input File

Copper metal is a fairly simple fitting problem. It is cubic, so all of the changes bond lengths can be parameterized in terms of a volume lattice expansion constant, α . A single Debye temperature, Θ_D , will be a good enough approximation for the σ^2 values. We will also need an E_0 and S_0^2 . Delete the line that says `% insert sets and guesses here` and type in the following

```
guess   amp     0.9
guess   e0      0.0
guess   thetad  300
guess   alpha   0.0
```

There are two ways we could procede from here. Using the fitting model I described in the last paragraph, it would suffice to set the `s02`, `e0`, `delr`, and `sigma2` path parameters as parameters in a zeroth path paragraph. Alternatively, we could explicitly set each of those parameters in each paragraph. Since this is a tutorial, let's do two of them as zeroth path parameters and two of them in each paragraph.

Type *`C-c C-t z`* or select `Zeroth path template` from the `Feffit - templates` menu. Fill this template in with the appropriate guessed parameters. It should look like this:

```
s02     0       amp
e0      0       e0
```

This will set make the best fit values of the variables `amp` and `e0` the S_0^2 and E_0 values for each path.

Now we need to add `delr` and `sigma2` path parameters in each path. EASE provides an elegant shortcut for this onerous editing task. Hit *`C-c C-v a`* or select `Add parameter to all paragraphs` from the `Feffit - Paragraph manipulation` menu. When asked '`Which parameter?`', answer with '`delr`'. Then when asked '`Default value for delr?`', answer with '`alpha*reff`'. This information is filled into every paragraph with the correct indexing. Now execute that function again again, this time answering the questions with '`sigma2`' and '`debye(temp, thetad)`'. Beats editing by hand!

We are almost done. The `debye` function require the sample temperature. For the data in this tutorial the temperature is 10 K, so put

```
set temp 10
```

after the guess parameters. To get the correct temperature dependence of the σ^2 terms it is necessary to consider the so-called McMaster correction which is necessary since $\chi(k)$ was normalized to the edge step in AUTOBK. Hit *`C-c C-s m`* or select `Insert McMaster corrections` from the `Feffit` menu. Answer yes to the question about setting `sigmm`. This function reads the value for the McMaster σ^2 correction from the '`feff.inp`' file and inserts its value in the '`feffit.inp`' file. It also adds this to the σ^2 for each path.

When you are finished, each path paragraph will look like rather like this

```
path    1       feff/feff0001.dat
id      1       amp=100.000, deg=12.000, nleg=2, r_eff=2.5527
delr    1       alpha*reff
sigma2  1       debye(temp, thetad) + sigmm
```

There should be 25 paragraphs. Now save the input file and run FEFFIT.

5.3 Examining the Output of FEFFIT

You can plot the results of the fit in k-space, R-space, or back-transformed k-space using *`C-c C-p k`*, *`C-c C-p r`*, and *`C-c C-p q`*, respectively. There are toolbar icons and menu entries for each of these plot options as well. You can examine the log and prm files using `C-c C-f l` and `C-c C-f r`.

EASE has a convenient mechanism for updating the guess parameters with their best fit values. Move the screen cursor to the line containing one of the guesses and hit `C-c C-s g`. EASE fetches the best fit value and the error bar from the log file and replaces the initial guess. `C-c C-s b` replaces all of the initial guesses in the file with their best fit values from the log file.

6 Plotting with Gnuplot

Several time throughout this tutorial, we plotted graphical output using the specialize and built-in plotting functions in EASE . In this chapter, I will describe how these functions work and how you can further interact with GNUPLOT from within Emacs.

One of the files that ships with EASE is '`gnuplot.el`' which defines '`gnuplot mode`' for Emacs. '`gnuplot mode`' is an emacs major mode for creating gnuplot scripts and sending them to gnuplot. In the '`docs`' directory of the EASE distribution there is a reference card for '`gnuplot mode`' called '`gpelcard.ps`'. This is a two page summary of the features of '`gnuplot mode`' in PostScript format. You should refer to that for complete details about '`gnuplot mode`'.

The basic idea behind '`gnuplot mode`' is that you can create a script for GNUPLOT in the '`gnuplot mode`' buffer and use functions to send parts or all of the script to a GNUPLOT process. The various plotting functions in EASE work by reading the input file to determine the names of the files to plot[1] and writes a script in the '`gnuplot mode`' buffer. It then uses the function for sending the entire buffer to GNUPLOT to display the plot.

For many common plotting chores during your analysis of EXAFS data, it quite suffices to simply use the functions offered by EASE . For looking at such common things as fit results and background removals, these functions work well. The nice feature of the manner in which EASE creates its graphical output is that it leaves you with a script that you can modify. Often it is simple and quicker to modify one of the scripts written by EASE than to create an entire script from scratch.

'`gnuplot mode`' provides four functions for sending parts of scripts to `gnuplot`. There are functions for sending the line currently occupied by the screen cursor (*`C-c C-l`*), a marked region (*`C-c C-r`*), or an entire buffer (*`C-c C-b`*). There is also a function for sending an external file (*`C-c C-f`*) to GNUPLOT. All four of these can be found in the '`Gnuplot`' menu. The line, region, and buffer functions are also bound to the toolbar in XEmacs.

EASE maintains a history of scripts sent to GNUPLOT. If you want to revisit an earlier plot, either one written by EASE or one that you wrote and sent to GNUPLOT, you can navigate the script history. *`C-c C-p`* steps backwards to previous scripts and *`C-c C-n`* steps forward to more recent scripts. These twop commands are bound to the up and down arrows in the XEmacs toolbar for '`gnuplot mode`'.

'`gnuplot mode`' also offers keyword completion, interaction with the GNUPLOT document, syntax colorization of GNUPLOT keywords, and direct interaction with the GNUPLOT process. See the reference card for more details.

[1] And a few other pieces of information, such as the title.

7 Closing Remarks

There are many features of EASE that I have not covered in this tutorial. This tutorial is only intended as an introduction to EASE . Once you have worked through the tutorial, I encourage you play around with EASE . Try out the various functions in the '`Input`' and program menus. Click on things and see what happens. I guarantee that you will find nifty little features that you will end up using on a regular basis.

Here is an incomplete and unordered list of things to explore:

Customization
Versions of Emacs and XEmacs numbered in the 20's come with a hypertextual customization package. With it, you can point and click to customize Emacs packages. Hit *`C-c C-b c`* or choose '`Customize EASE`' from the '`Input - Miscellaneous`' menu. This will open up a customization buffer for variables in EASE . Customizing these variables will allow you to control dozens of features regarding the appearence and behavior of EASE and it interaction with other packages and programs.

The EASE command wrapper
Although FEFF and the other programs require that input files be named according to their program, EASE does not require this. You can call you input files anything you want. For example, you might have called the FEFFIT input file from this tutorial '`cu.inp`'. Before executing the program, EASE will rename the input to the name exected by the program. When the program is finished, EASE renames it back. Care is taken not to overwrite any existing files. EASE also renames the log files[1] written by the programs. In the case of '`cu.inp`', EASE would rename the log and prm files '`cu.log`' and '`cu.prm`'.

Include files
FEFFIT and AUTOBK allow the use of include files. EASE has many features for working with include files. Most importantly, it allows you to assign a master file to each include file using *`C-c C-d m`* or '`Set master file`' from the '`Imput - Set variables`' menu. Once the master file is assigned, functions for running programs, looking at log files, and others do *the right thing.*

Tag files
When using include files in FEFFIT, it can be easy to forget in which file variables are defined. In '`feffit mode`' you can create a tag file which cross references variables with their location of definition. Once the tag file is created with *`C-c C-f t`*, you can place the screen cursor on a variable name in a math expression and type *`M-.`*. The display will switch to the place where that variable is defined.

Paragraph manipulation
In the chapter on FEFFIT (see Chapter 5 [Running Feffit], page 17) we used the paragraph manipulation function for adding a parameter to each paragraph. There are also functions for renumbering paragraphs, deleting parameters, and commenting out and uncommenting parameters. These functions work on every

[1] Note that EASE does *not* currently rename the output files from FEFF.

paragraph in the file. If you preceed the key sequence or menu selection with `C-u` then the paragraph manipulation function will start working from the current screen position rather than on the entire file.

Cleaning EASE files by indenting lines and separating columns as specified by the appropriate variables.

Editing shortcuts
Lots of repetitive tasks are automated by EASE . In AUTOBK and FEFFIT there are functions for filling in keyword values by snagging values from similar keywords in surrounding stanzas and paragraphs. There are also functions for swapping the words '`set`' and '`guess`' in a FEFFIT input file, for swapping boolean values, and for commenting and uncommenting single lines or blocks of text.

HEY! Have fun.

Index

Table of Contents