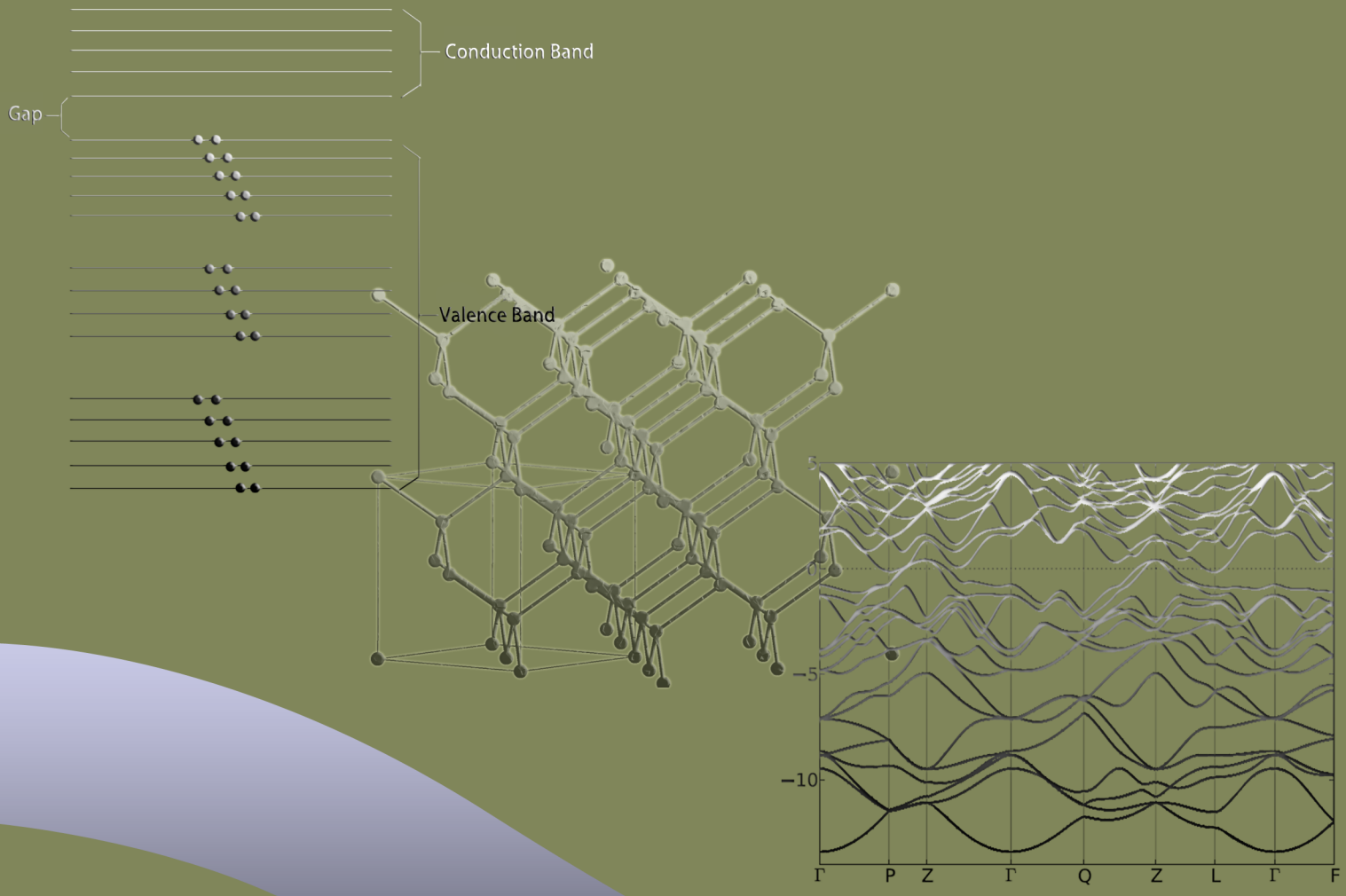
Queen Mary, University of London
School of Physics and Astronomy

Important Notes On FEFF and IFEFFIT



Notes On FEFF

1. Generally, the three modules – 'pot', 'fms' and 'genfmt' – take much time during the calculation.
   'pot' → Module for building up the potential.
   'fms' → Full multiple scattering expansion.
   'genfmt' → effective scattering amplitude calculation.
2. In FEFF, the length unit used is angstrom ($\mathring{A}$) and energy is in electron-volt (eV).
3. Briefly, the screening (or shielding) effect refers to when we calculate the final state (which is naturally outer shell) wavefunction, the electrons in the inner shell will 'shiled' the effect of nuclei to the electron on the outer shell. Therefore, when the core electron is excited to the final state, the shielding effect should change accordingly. And such change of screening effect is what we have to take into account for the absorption spectroscopy.
4. The content in 'xmu.dat' output file:

Column	Content
1	Energy (eV) – Absolute
2	Energy (eV) – Relative to Edge
3	k ($\mathring{A}^{-1}$)
4	μ
5	μ_0
6	$\chi = \frac{\mu-\mu_0}{\Delta_\mu}$

5. FEFF, by default, calculates EXAFS spectra if no spectra card was specified in the 'feff.inp' input file.
6. FEFF can be used to specify the charge transfer property of the mateirial in interest, for details refer to P_{46-47} in FEFF9.6 manual.
7. If S_0^2 (the passive amplitude factor) in the input file is set to be smaller than 0.1, then S_0^2 value is then estimated by FEFF, and the result for S_0^2 can be found in 'chi.dat' and/or 'xmu.dat'.
8. To change the energy (or k-space) grid, the 'EGRID' card can be used, for details see FEFF9.6 manual P_{88}.
9. To compare the band structure calculated by FEFF with that from other commonly used programs, there are several points to notice. First of all, the zero of the FEFF energy grid is the vacuum energy. Secondly, using a large calculation cluster (also applies for full multiple scattering (FMS) calculation) in FEFF will make the gap value better defined. For details see the online material link: Click here.
10. The total number of electrons given in the FEFF output file is that for open shell only. For example, the electronic configuration of Ge is: [Ar]3d104s24p2, then the total electron number given in FEFF output file is 14, if there is no charge transfer.

11. In FEFF output file, the positive value of charge transfer means electrons were 'transferred' out of current atom. In contrast, negative charge transfer means electrons were 'transferred' into current atom.
12. To check the band gap value of the material in interest, the l-resolved density of states given by FEFF (if specifying the 'LDOS' card in the input file) may come to help, as illustrated below:

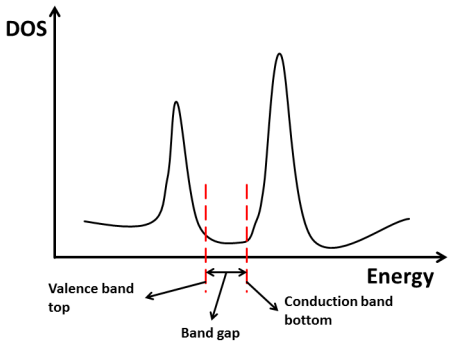


Figure 1 The illustration of band gap in FEFF.

Here it should be pointed out that the DOS of the outer shell should be used (for Ge, it is p shell) to determine the band gap.

13. When doing the FEFF LDOS calculation, if 'NOHOLE' card (see FEFF9.6 manual, P_{89}) is specified, the calculation LDOS is for the ground state. Otherwise, the calculation is excited state related and not comparable to the result from other standard DFT calculation. For details see the online material link: Click here.
14. To get the final DOS corresponding to specific material in interest (again, to compare with the result from other programs), the l-resolved LDOS given in FEFF should be summed up, taking into account of the number of atoms of each species in the unit cell. For details see the online material link: Click here.
15. Sometimes the Fermi level given by FEFF calculation is a bit higher than the standard value (usually by 3 eV). To solve this 'problem', the 'EXCHANGE' (see FEFF9.6 manual, P_{92}) card can be used in FEFF input file with the 'Vr' parameter set to be 3 eV. For details see the following link: Click here.

Notes On IFEFFIT

1. IFEFFIT doesn't allow looping over the elements of an array.
2. The 'def()' command keeps dynamic track of the formula it specifies, e.g. 'def(b=a+1)' will automatically update the value of 'b' according to the value of 'a' during the fitting or new assignment for 'a'. The function of 'def()' command is then different from that of 'set()' command, which doesn't change the value of specified quantity after the 'set' execution.
3. The system variable starts with ampersand '&', and string variable starts with '$' symbol. Therefore the system string variable starts with '&$' symbol.

4. The array is stored as groups. All arrays in a group shares the group name and each group members then has its own name. For example, the 'data' array group contains two members: 'energy' and 'xmu', and each member can be specified using this format: 'data.energy' and 'data.xmu'. This makes the following analysis and fitting process more convenient.

5. The 'show()' command is more functional and give more detailed information then 'print()' command. For example, 'show @scalars', 'show @strings', and 'show @arrays' shows the definition and values of all scalar, string and array variables, respectively. And 'show @commands' show all available commands in IFEFFIT. Furthermore, the 'show @group=GROUPNAME' command shows only the arrays belonging to the group of 'GROUPNAME'.

6. To print out simple message (which is not assigned to a string symbol), the **single** quotation mark but **NOT** the double quotation mark should be used.

7. Commands to read in arrays from data files:
   '**read_data**(file = 'FILENAME', group = 'GROUPNAME', type = 'xmu')' – The first two options 'file' and 'group' is generally applicable, which specifies the file to import data from and the group name for arrays imported. The third option 'type' then specify the way that array names are assigned, and 'xmu' here means the first two columns are assigned with the array name of 'Energy' and 'xmu', respectively. And $3, 4, \ldots$ will be used for any remaining columns.
   Another commonly used type is 'chi', which names the first two columns as 'k' and 'chi', respectively, and do the same thing as type 'xmu' for any other remaining columns.
   Or we can specify our own labels for each column in the imported file, by using the keyword 'label' in the 'read_data' command as following:
   'read_data(file = 'FILENAME', group = 'GROUPNAME', label = 'energy xmu')' – Do exactly the same thing as type 'xmu', and it should be kept in mind that there should be a space between each specified column label (i.e. 'energy', 'xmu', etc.).
   If no keyword concerning the column label is specified, IFEFFIT will try to find the label line, which comes after a line of minus sign in the imported file: '#- -'. Thus if the line following the above 'notifiction' line looks like '# Energy xmu', the result will be again exactly the same with that for 'type = 'xmu''.
   Furthermore, if IFEFFIT cannot find the 'notification' line or the label line, it will give each column the name of $1, 2, 3, \ldots$, continuously. And this can be enforced by giving the keyword of 'type = 'raw'' in the 'read_data' command.

8. To zoom in the plot in IFEFFIT, we can type 'zoom' on the command line, and then select the part we want to zoom in using cursor. Then if want to zoom out, we should replot the original figure, by using the command 'newplot' instead of 'plot' — The 'plot' command will keep the zoomed area not changed.

9. Using 'pre_edge()' on an array will create quite a few output parameters including scalars (e.g. e0, edge_step, pre_slope, etc.) and arrays (e.g. GROUPNAME.pre, GROUPNAME.norm). Then if using 'pre_edge' command on another array will create arrays corresponding to the new array (often with differ-

ent group name from the previous one), however will **overwrite** all the **scalar** variables created from the previous execution of 'pre_edge' command.

10. The post-edge background subtraction process in IFEFFIT is actually the process of removing the smoothing background absorption from the real spectra to get the wiggles contained in the spectra. The AUTOBK algorithm is used in IFEFFIT, which means the Fourier transform should be carried out to determine which part of the spectra will be taken as the background according to a parameter named '*rbkg*'. Generally, the 'spline()' function doing the post-edge background subtraction needs parameters as described following:

Variable	Description
rbkg	R_{bkg}, the highest R value to consider background
kmin	k_{min}, the starting k for the Fourier transform
kweight	w, the k-weight factor for the Fourier transform

Usually, values of $k_{min} \approx 0$ and $w = 1$ are appropriate for 'spline()' command, since at this step, we are only concerned about the background subtraction.

Here are the new arrays will be created by 'spline()' command:

Variable	Description
GROUPNAME.bkg	$\mu_0(E)$, the background function itself
GROUPNAME.k	k, the array of wavenumbers
GROUPNAME.chi	$\chi(k)$, the (unweighted) EXAFS

where the 'GROUPNAME' is the group name of which the 'spline()' command is executed on.

Finally the familia equation for the obtained $\chi(E)$ signal after background subtraction:

$$\chi(E) = \frac{\mu(E) - \mu_0(E)}{\Delta\mu(E_0)}$$

where $\Delta\mu(E_0)$ is the edge jump.

11. In the IFEFFIT manual, when talking about the Fourier transform, it says the Fourier transform command used in IFEFFIT is not general Fourier transform function, but is really XAFS Fourier transform. The reason is that in XAFS formula, the vibration term contains '2kR' term in the *sin* function, so the normal Fourier transform will give the k-2R transformation. However, what we need is the k-R transformation, which is just what the real XAFS Fourier transformation means. Also it is said that 'Aside from a scale factor, this changes the normalization constants used' (see IFEFFIT Tutorial Page12, section-5.4). Here qualitative explanation is given for the above instruction. The reason why we say 'qualitative' is that quantitative explanation needs detailed information about Fast Fourier transformation (FFT), thus the discussion is simplified here to give only an idea of the above saying. First of all, let's have a look at the normal Fourier transformation formula:

$$A_1 \int_{-\infty}^{\infty} f(k) e^{ikx} dk = F_1(x)$$

where A_1 is the normalization factor. In the on-line material Normalization of Fourier Transforms, it could be known that the A_1 factor here is for the

purpose of keeping the 'inner product', as given by:

$$\int_{-\infty}^{\infty} f(k)f^*(k)dk = \int_{-\infty}^{\infty} F_1(x)F_1^*(x)dx$$

And for normal Fourier transform, the normalization factor A_1 is $1/\sqrt{2\pi}$. Furthermore, for XAFS Fourier transform, we should have corresponding expression as:

$$A_2 \int_{-\infty}^{\infty} f(k)e^{i2kx}dk = F_2(x)$$

where the factor 2 takes into account the '2kR' term in XAFS formula and directly gives the k-R but not the k-2R transformation. Accordingly, when we again calculate the 'inner product' in k-space and R-space and make them identical, we should definitely have the **new normalization factor $\mathbf{A}_2$** different from that for **normal Fourier transformation** – $\mathbf{A}_1$. There may be problem about how the factor 2 goes into the normal Fourier transform and turn it into the so-called 'XAFS Fourier Transform'. The understanding is straightforward if imaging the original 'x' in normal Fourier transform means $2R$, and for XAFS Fourier transform, '2x' now means '2R' thus 'x' definitely means 'R'!

12. The FFT algorithn in IFEFFIT requires the input $\chi(k)$ should be an array on an even k-grid with grid spacing of $k = 0.05\,\mathring{A}$, and starting from $k = 0$. So when using the 'fftf()' command to do Fourier transformation in IFEFFIT, we don't need to specify the k array if it already meets the above requirement. Actually, the output $\chi(k)$ array from 'spline()' function automatically meets the requirement for FFT algorithm. However, if the $\chi(k)$ data was imported from other source, and it happens not to meet the above requirement, we then need to specify the k array as one of the parameters for 'fftf()' command. This will ask IFEFFIT to do the **interpolation before the FFT process** to make the k array meet the requirement. The example command is as following:
fftf(GROUPNAME.chi, k=GROUPNAME.k, kmin=2.0, kmax=17.0, dk=1.0, kweight=2)
where it should be pointed out that the 'kmin', 'kmax' and 'dk' parameters are to define the window function for the FFT transform. Also we could have another parameter named 'kwindow' to specify the window function for FFT algorithm, e.g. 'kwindow=hanning' for Hanning window, 'kwindow=kaiser' for Kaiser-Bessel window, etc.

13. As is known, the phase information is quite important when doing the Fourier transformation (briefly speaking, 'phase' means the ratio of *cos* and *sin* component in the Fourier series). However, for the raw $\chi(k)$ data obtained from processing the initial $\mu(E)$ data does not give the phase information, or in other words, the phase is artificial. In IFEFFIT, when doing the Fourier transformation for the $\chi(k)$ signal, it gives us the option to specify the $\chi(k)$ signal as real or imaginary part. That basically means IFEFFIT artificially uses either 0 (when specifying $\chi(k)$ as real) or $\pi/2$ (when specifying $\chi(k)$ as imaginary) when doing the Fourier transformation. This is just the reason for why we always have the spectra in R-space shifts to lower value if no phase information is specified. IFEFFIT provides several options to import the phase information from FEFF calculation to do the phase-corrected Fourier transformation:

- Embedded 'central atom phase shift' - 'fftf(data.chi,...,pc_edge='Z EDGE')'. Here 'Z' refers to the atom number of the central atom, which can be replaced by corresponding element symbol, and 'EDGE' specifies the absorption edge, which can be omitted (the default K-edge will be used).
- 'central atom phase shift' given by real FEFF calculation - 'fftf(data.chi,...,pc_feff_path=1,pc_caps)'. (the keyword 'pc_feff_path' is explained after next item)
- 'full phase shift' given by real FEFF calculation - 'fftf(data.chi,...,pc_feff_path=1)'.

  The path specified by 'pc_feff_path' should be read in before the phase correction process, using the command like:
  path(1,../feff/feff0001.dat,label='Path-1')
- Another method is specify the phase array by introducing 'phase_array' keyword in 'fftf()' command. The steps is described as following: read in the path from feff, use get_path() to get the scattering phase-shift, interpolate onto the data k-grid, and specify the phase array as the 'phase_array' in 'fftf()':

  read_data(file=my_chi.dat,type='chi',group='data')
  path(1,../feff/feff0001.dat,label='Path-1')
  get_path(1,do_arrays,group=f1)
  set feff.phase_shift=interp(f1.k, f1.phase, data.k)

  fftf(real=data.chi, k=data.k,.., phase_array=feff.phase_shift)

Information about the above method for phase-corrected Fourier transform can be found in: Phase Correction Using IFEFFIT (Here is the link for original webpage), fft_ phase_corrections.iff (Here is the link for original webpage) and ft_pc.iff (Here is the link for original webpage).

14. Similar to the forward FFT transform, the reverse FFT also requires the input $\chi(R)$ must be given as an array that is evenly spaced in R-space with a fixed grid spacing of $R = \pi/1024 \approx 0.03068\,\mathring{A}$ (the fixed grid spacing value should be something to do with the FFT algorithm, since at the denomenator we have $1024 = 2^{10}$), and starting at $R = 0$. The important application of reverse Fourier transform is to compare with original $\chi(k)$ signal after filtering process - the reverse Fourier transform process is actually the process of filtering since we specify the R-window ($R_{min} \rightarrow R_{max}$) during the reverse FFT process. However, the $\chi(q)$ and the original $\chi(k)$ signal cannot be compared directly to each other, since the forward Fourier was carried out on the weighted (by k-weight) and windowed (by selected window function, e.g. Hanning window) $\chi(k)$ signal. Thus the obtained $\chi(q)$ signal directly from reverse FFT is actually already windowed and weighted. To compare them, the best way is to weight and window the original $\chi(k)$ signal using exactly the same parameter as that for forward Fourier transform. In IFEFFIT tutorial, a macro was defined for this purpose:

    macro filter GROUPNAME "kweight=2,kmin=3" "dr=0"
    fftf(real=$1.chi,$2)

```
   fftr(real=$1.chir_re, imag=$1.chir_im, $3)
   set  $1.chik     =  $1.chi   * $1.k^kweight
   set  $1.chik_w   =  $1.chik * $1.win
end macro
```

Here the third and forth line in the defined macro finally gives the weighted and windowed $\chi(k)$ signal, which can then be compared to the $\chi(q)$ signal using the following commands:

```
filter data "kweight=2,kmin=3,kmax=15,dk=1" "rmin=1.6,rmax=3"
newplot(data.q, data.chiq_re)
plot(data.k, data.chik_w)
```

Here in this example, we take, by default, $\chi(k)$ as the real part for the forward Fourier transformation, so that the real part of $\chi(q)$ is the appropriate choice for comparison.

15. The experimentally measured XAFS signal is typically described as the imaginary part of a complex fine structure function $\tilde{x}$ (IFEFFIT Tutorial, Page13), therefore when combining the path information from FEFF calculation to the theoretical $\chi(k)$, using 'ff2chi()' command, the $GROUPNAME.chi array generated is taken as the imaginary part of the complex $\chi(k)$ signal (again, theoretical one). However, IFEFFIT usually assumes that the $\chi(k)$ data (when processing it, e.g. Fourier transforming it) is the real part of the complex $\tilde{x}$, which although seems a bit confusing, but actuallly is just a matter of convention.

16. In addition to the common way of using 'feffit()' command given in IFEFFIT Tutorial (Page19), several advanced way of using 'feffit()' command is given in IFEFFIT Reference Guide as following:

    - Fitting background using 'do_bkg' keyword → Page53.
    - Restraint fitting → Page53-54.
    - Multiple k-weight fitting → Page54.
    - Simultaneous fitting of multiple data sets → P54-56. When several data sets were measured at the same experimental condition, IFEFFIT proivdes the option to fit all of those data sets, simultaneously. The advantage of simultaneous fitting is that parameters like S_0^2 and e_0 can be shared among all the data sets. The most important thing to notice is that the fitting is done SIMULTANEOUSLY for the whole data sets but NOT individually on each independent data set!

17. When defining a macro, if the first line in the definition body is a plain text line, it will be regarded as the documentation line for the defined macro. And this line will be shown when executing 'show @macros' command. If we need the command-line argument as the input, we can directly type '$1', '$2', etc. to specify the first, second,..., command-line argument within the definition body. If we want to specify the default value for the command-line argument, we can then define the macro like this:
    macro NAME "ARGUMENT1=VALUE1" "ARGUMENT2=VALUE2"

18. When executing the 'write_data' command, all except the array variables will be written as comment lines (which starts with '#' symbol) before the arrays. The string variables will come at first, followed by the variables that is specified

as the keyword of 'write_data' command. Keeping the above rule, all variables will come in the order as they are in the 'write_data' command. Also order of the array output will also depend on how they are each located in the 'write_data' command.

19. The 'log(my_log_file.log,screen_echo=3)' is the commonly used command to start writing all the pop-up messages both to the screen and the specified log file (in this case, it is 'my_log_file.log'), and 'log(close)' command will close the opened log file. Starting another log file if the previous log file has not been closed will automatically close the previous one.
20. Use the 'save(my_saved_progress.sav)' and 'restore(my_saved_progress.sav)' command to save the current progress (including all the variable values) to 'my_saved_progress.sav' file and restore it later on.
21. Starting IFEFFIT using command 'ifeffit my_startup_file.iff' will start IFEFFIT and execute the 'my_startup_file.iff' file before showing the command line. And this way of starting IFEFFIT 'ifeffit -x my_startup_file.iff' will first start IFEFFIT and then execute 'my_startup_file.iff', and after that, IFEFFIT will be closed without popping up the command line environment.
22. The standard shell commands of Unix environment, including *ls, cd, pwd* and *more*, can be executed directly in IFEFFIT. To use other commands, we need type '!' sign before the corresponding command. However, the alias defined in Unix environment is not supported.