Programmer's guide : calculating EELS with feff85.

implementation by Kevin Jorissen
this document by Kevin Jorissen
this version : 05-15-2006
contact me at kevin.jorissen@ua.ac.be

If you are not on the FEFF development team, it is highly unlikely that you will ever need the information contained in this document. Please save a tree and don't print this!

It is assumed throughout this document that the reader is familiar with the contents and notations of the accompanying User's guide and the FEFF User's guide.

1. General comments.

A central quantity in FEFF is the polarization tensor (ptz). In regular FEFF calculations, it is calculated from the polarization vector e (evec in the code) and the beam direction k (xivec in the code), and then the spectrum corresponding to that physical situation is calculated. Ptz is processed internally.
For EELS calculations, we want access to all 9 polarizations (6 of which are independent), giving us 9 partial spectra that we can sum, with weights depending on experimental conditions, to very quickly (as sample information is precomputed into those 9 components) assemble a physical EELS spectrum. This means that all elements of the code where a specific value for ptz, evec or xivec is used need to be looped over (ptz) or circumvented (evec and xivec).

The modifications to the code fall into three categories :
- minor things, such as adding new cards, according to FEFF tradition ;
- modifying existing routines that handle the polarization tensor so that the full tensor is preserved ;
- adding a new module, called *eels*, for calculating the spectrum out of the 9 partial spectra.

It is important to realize that ***all modules except pot, ldos, and screen have been modified*** and will adapt their behaviour in the presence of active EELS input!! This is because "inner loops" in the code (with substantial changes) allow for much faster calculations than "outer loops" (with no substantial changes).
It is the intention of the author to mention ***every*** modified routine explicitly in this document.

2. Module 0 (RDINP or ffmod0).

* This module now accomodates new cards and variables and a new output file. Changing the coordinate system has been disabled in EELS calculations.
* New cards ELNES, EXELFS and MAGIC have been added, with internal codes 54, 56 and 55, respectively. To this purpose, relevant blocks of code were added to *rdinp* and *itoken* (in src/COMMON). Since the ELNES and EXELFS card takes several lines of input from the feff.inp file, a new 'reading mode' mode=4 was added in *rdinp*.
A new common block /elnes/ defined in allinp.h contains the new variables of the ELNES/EXELFS and MAGIC card.
* ELNES is very similar to XANES; the parameters on the first line of the card are exactly the same. Similarly, EXELFS incorporates the EXAFS card. I have 'absorbed' the relevant variables because it seems confusing to have to use both ELNES and XANES cards in one file – quod non, now. On the other hand, the calculation of ELNES invokes essentially the same

routines as that of XANES, so that I have not attributed a new ispec number to elnes and exelfs calculations. Apart from a loop or two, the difference is mainly cosmetical until we reach the *eels* module.
* *Iniall* sets the elnes variables to defaults that essentially mean : don't calculate eels, do traditional x-ray feff.
Rdinp has new card consistency checks : it tries to avoid combinations of ELNES and POLARIZATION or ELLIPTICITY, and it ignores MAGIC if ELNES is not present. It also checks consistency of the control switches aver, cross and elnes, and sets the important parameters ipmin,ipmax,ipstep depending on the control switches. The ip parameters determine which components of the sigma tensor are calculated.
Ffsort has a new input argument : a logical switch doptz which disables the call to mkptz if eels is active. This is because 1/ mkptz is not necessary : the polarization tensor will be set later ; and 2/ because rotation of the atom coordinates in mkptz may interfere with the existence of off-diagonal sigma tensor components and the desired interpretation of the spectrum in terms of its components in a particular basis.
Wrtall writes the eels parameters to the file eels.inp. All other files retain their exact traditional format. Note that global.dat will contain some unset variables (in particular the ptz matrix) in case eels is active.

3. Module POT.
No changes at all.

4. Module SCREEN.
No changes at all.

5. Module XSPH.
* Minor change : explicit setting of the polarization tensor is necessary for EELS calculations for strictly practical reasons.
* *Rexsph* checks for the presence of a eels.inp file with meels=1 (i.e., an active eels.inp file). If it finds this, it sets the polarization matrix ptz – which was read earlier from global.dat – to 1/3. This is necessary, since in case of eels rdinp has not set the polarization matrix, and okay, since for xsph only the trace of ptz matters. A message is written to the log file.

6. Module LDOS.
No changes at all.

7. Module FMS.
* This module has been changed significantly. Instead of summing over the components of the polarization tensor internally, it now calculates a spectrum for each component separately, and outputs all those (9) spectra to its output file fms.bin.
* *Reafms* looks for an active eels.inp file (eels=1). If it finds one, it takes the parameters ipmin, ipstep, ipmax from that file. If not, it sets them all to 1. The variables elnes,ipmin,ipstep,ipmax are returned as output parameters.
Ffmod3 passes them on to fmstot.
Fmstot calls bcoef several times to set up the bmat matrix for every polarization considered (do ip=ipmin,ipmax,ipstep). If (eels=1), each call to bcoef is preceded by a call to *iniptz*, a new routine that sets the polarization matrix and can work either in carthesian or in spherical coordinates (current settings use a carthesian representation) ; if (eels=0), the value read from global.dat in reafms is used. The bmat's are stored in a supermatrix bmat0, which is one dimension larger. Then inside the big loop over energy, the matrix inversion is done only once

(as for xas) by calling fms. Now there is a loop over the section where gg (inverted matrix from fms) is formed into gtr (the output of ffmod3) by applying bmat. This loop over polarization tensor components selects the right bmat from bmat0, calculates, and puts the result in gtr – which has an extra dimension in the new code : gtr(ip, E). The MPI instructions for gathering pieces of gtr have been modified (but not tested!) to accommodate this extra dimension. The full gtr matrix is written to fms.bin in a consistent way (i.e., if ipmin=ipmax=1 it has the exact same format and content as in a xas calculation).

8. Module 4 (PATH or ffmod4).

* Most eels calculations require the absorption tensor for every polarization component and for every sample to beam orientation. I am not sure how this influences the selection of paths. I fear this may prohibit use of any symmetry and it may be necessary to use icase=7 for orientation sensitive calculations in *mpprmp* until further testing has been done.
* *Repath* looks for an active eels.inp-file (eels=1). It passes eels on to *ffmod4*, then to *pathsd*, then to *timrep*. If eels=1, timrep initializes a variable icase to 7 ; if eels=0, it puts it to -1. This variable is a new input parameter to *mpprmp*, which uses it to override its own decision procedure for icase if between 1 and 7.

9. Module GENFMT.

* *Regenf* looks for an active eels.inp file (eels=1). If it finds one, it takes the parameters ipmin, ipstep, ipmax from that file. If not, it sets them all to 1. The variables elnes,ipmin,ipstep,ipmax are returned as output parameters.
Ffmod5 passes them on to genfmt.
Practically all of *genfmt* (except for the reading of phase.bin in rdxsph) is contained within a loop over the components of the polarization tensor (do ip=ipmin,ipmax,ipstep). For every ip-value, a file feff{ip}.bin and list{ip}.dat is written – except for ip=1, where the filenames are simply feff.bin and list.dat for compatibility reasons. If (eels=1), iniptz is called (see module 3) to set the polarization tensor ptz to the desired value determined by ip ; if not, the value read from global.dat by regenf is used. The polarization tensor is passed on to mmtr, which will eventually call bcoef to create a bmat matrix.

10. Module FF2X.

* This module has been looped over the components of the sigma tensor ; it needs to read input and write output for every component separately.
* *Reff2x* looks for an active eels.inp file (eels=1). If it finds one, it takes the parameters ipmin, ipstep, ipmax from that file. If not, it sets them all to 1. The variables elnes,ipmin,ipstep,ipmax are returned as output parameters.
Ffmod6 passes them on to ff2xmu (called for ispec=1 (XANES/ELNES) and ispec=2 (XES)) or ff2chi (called for ispec=0 (EXAFS/EXE/FS)) or ff2afs (called for ispec=3 (DANES) and ispec=4 (FPRIME)).
In *ff2xmu*, a new variable gtrful (and temporary buffer gtrtemp) has been created to enable reading the enlarged fms.bin file written by module3. The code will work fine for old fms.bin files. Much of ff2xmu (after reading xsect.bin) has been looped over the components of the sigma tensor (do iip=ipmin,ipmax,ipstep – the variable ip has another meaning in this routine!). The component of gtrful corresponding to iip is copied into gtr, and the corresponding feff{iip}.bin is read. Because direct components (eg. xx) should have the atomic background, but off-diagonal components (eg. xy) should not, instead of xsec (read from xsect.bin) I use the variable kxsec which is equal to xsec for diagonal components and is 0 for offdiagonal components. The resulting spectrum – a x-ray spectrum! – is written to xmu{iip}.dat (for iip=1, we use 'xmu.dat' for compatibility reasons).

In *ff2chi*, a similar approach is followed. Since fms.bin is not read in this routine, we do not need new gtr-related variables. Again, most of the code is looped over. We now need to read list{iip}.dat in addition to the files mentioned earlier. As the variable omega (read from xsect.bin) is recycled in the code (why keep things clear?), I read it again in each cycle. The same trick with kxsec and xsec is performed. An additional output file chi{iip}.dat is written. In *ff2afs*, again the same thing is done.

10. Module SO2CONV.
* Checks for active eels.inp file. If present, the whole code loops over the polarization tensor (do ip=ipmin,ipstep,ipmax). Each xmu.dat file is read and overwritten individually.

11. Module EELS.
* Completely new module, separate from existing code.
* Description to be provided later.

12. Other source directories.
COMMON : function itoken is adapted to accommodate the new cards ELNES, EXELFS and MAGIC.
ATOM, DEBYE, EXCH, FOVRG, HEADERS, LIB, MATH, PAR, SCREEN, TDLDA, Utility : No changes at all.

13. Programming conventions.
* In existing routines, my aim has been to make as little conceptual modifications as possible. This means that sometimes a routine will not be the shortest possible, or there may be some redundancy in variables ; but its 'spirit' will be conserved as much as possible, so that someone who knew the old version of the routine won't have too much trouble getting used to the new version.
* In existing routines, every line that I change bears my initials in a comment !KJ , sometimes followed by an explanation. When a block of new code is inserted, it will be preceded and followed by !KJ. If significant changes are made to an instruction, the old version is often preserved as a comment.
* In new routines, the implicit none statement is always used. In existing routines, all new variables (including the index i etc.) are declared explicitly. Always.
* No old-fashioned coding! This means avoiding line numbers, goto statements, and other encryption techniques.
* In the new module *eels*, some f90 is used. We also have a f77 version.

situation	eels	aver	cross	ipmin	ipstep	ipmax	ip	ptz
XAS	0	/	/	1	1	1	1	as given by POLA and ELLI cards
EELS, orientation averaged	1	1	0	10	1	10	10	1/3 * I
EELS, orientation averaged, symmetric coordinates	1	0	0	1	4	9	1, 5, 9	xx, yy, zz
EELS, orientation	1	0	1	1	1	9	1, 2, 3, 4, 5,	xx, yy, zz,

averaged, general coordinates							6, 7, 8, 9	xy, yx, xz, zx, yz, zy

Table 1 : values of some important EELS variables for several physical situations.

ip	Carth. name	polarization tensor
1	xx	1 0 -1 0 0 0 * 1/2 -1 0 1
2	xy	-1 0 -1 0 0 0 * i/2 1 0 1
3	xz	0 1 0 0 0 0 * 1/sqrt(2) 0 -1 0
4	yx	1 0 -1 0 0 0 * i/2 1 0 -1
5	yy	1 0 1 0 0 0 * 1/2 1 0 1
6	yz	0 -1 0 0 0 0 * i/sqrt(2) 0 -1 0
7	zx	0 0 0 1 0 -1 * 1/sqrt(2) 0 0 0
8	zy	0 0 0 1 0 1 * i/sqrt(2) 0 0 0
9	zz	0 0 0 0 1 0 0 0 0
10	average	1 0 0 0 1 0 * 1/3 0 0 1

Table 2 : Carthesian components of the polarization tensor expressed in FEFF's spherical basis, as implemented in routine *iniptz*.