

# Manual for `multiX`

A. Uldry and B. Delley  
Paul Scherrer Institut, Switzerland.

May 2, 2012

## 1 Introduction

The program `multiX` allows you to calculate the energy levels of an atom in a crystal field defined by the charges and positions of its neighbours, and to plot the resulting XAS and RIXS spectra. The foundation of the code is explained in the following publication:

*Systematic computation of crystal field multiplets for x-ray core spectroscopies*  
Phys. Rev. B **85**, 125133 (2012).

Please cite this paper if you make use of this code.

We presume that the code has been compiled and is ready to execute. The present document describes how to run the code, the input required and the output files.

## 2 Running the code

The executable `multiX` requires the presence of a file named `INPUT` in the directory where you wish to run the code. If the executable `multiX` is in the same directory, the code is fired off by invoking “`multiX`” at the command line, and piping the results to the file “`log`”.

```
$ ./multiX | tee log
```

If the code resides in a different directory, the full path to the executable should be given. For example, if the code is in the directory `Multiplets/Source` in your home directory:

```
$ ~/Multiplets/Source/multiX | tee log
```

If the code is in a location that is included in your path, then

```
$ multiX | tee log
```

should execute the code.

It may be necessary to set the stack limit to “unlimited” in some system. In bash, this can be done by adding the following line to the `.bashrc` file:

```
ulimit -s unlimited
```

The `INPUT` file, and the various output files, are described in the next sections. Scripts that assist the running of the code for different tasks are provided and described at the end of this document.

### 3 The `INPUT` file

The `INPUT` file must be present in the execution directory. It is a plain and editable text file where the user defines which emitting atom is to be used, the core-valence ground state configuration, and what are the parameters and the task to perform. A bare minimum `INPUT` is shown in `INPUT` example 1. This input would cause `multiX` to calculate the eigenvalues and

```
new MULTO                # Task: multiplets for the
                          # ground state configuration
                          # only.

atom Mn                   # The multiplets are calculated
                          # for the open shells of a
                          # Mn atom.

ground_state 2p6 3d5     # Core-valence ground state
                          # configuration.
```

**INPUT example 1:** Minimal `INPUT`.

eigenvectors for an isolated Mn in the  $2p6\ 3d5$  ground state configuration.

The `INPUT` file contains *inline* entries, like those in the example `INPUT` 1, and *block* entries. Inline entries take the form

```
<keyword> <value> [<value> ...]
```

Block entries are typically used to list input energies for RIXS, and as one of the alternatives for giving the crystal field. Examples will be given below. Block entries take the form

```
begin_<keyword>
<value 1>
<value 2>
...
end_<keyword>
```

### 3.1 The crystal field

The crystal field is constructed from the charges and positions of the ions surrounding the emitter. The code expects the positions in Ångström and the charges in units of the electron charge (fractional charge is accepted). A general multiplier is expected at the top of the list of positions and charges:

```
<multiplier>
<coord x1> <coord y1> <coord z1> <charge of ion 1>
...
<coord xn> <coord yn> <coord zn> <charge of ion n>
```

If only a few ions are making up the crystal field, it is possible to integrate the crystal field list in the INPUT as a block entry:

```
begin_xtal
2.45
  1.000000    0.000000    0.000000   -2.000000
 -1.000000    0.000000    0.000000   -2.000000
  0.000000    1.000000    0.000000   -2.000000
  0.000000   -1.000000    0.000000   -2.000000
  0.000000    0.000000    1.000000   -2.000000
  0.000000    0.000000   -1.000000   -2.000000
end_xtal
```

Alternatively, especially if the number of ions is large, the list of charges and positions can be placed in an external file. The name of the file is declared in the INPUT:

```
xtal_file ligands.txt          # the name of the file in the same
                                # directory with the charges and
                                # position of the Mn neighbours.
```

The file `ligands.txt` must be in the same directory. An example of such a file is shown in the `ligands` file 1. The elements in grey are for clarity and

1.000000	x	y	z	q	radius		
0.186578	-1.117996	1.288676		-2.0	1.72	0	1
-1.269110	1.031996	0.776289		-2.0	1.81	0	2
-1.311288	-0.869050	-1.064677		-2.0	1.90	0	3
1.535878	1.145154	0.690135		-2.0	2.04	0	4
0.180936	1.394100	-1.490910		-2.0	2.05	0	5
1.636624	-0.982207	-0.978523		-2.0	2.14	0	6

Ligands file example 1: External ligands file.

convenience; they are optional and are in fact not read by the code (this is valid for the block structure as well). On the first line, only the multiplier is read. On subsequent lines, only the three positions and the charge are read, the rest is ignored. The program stops reading when either the end of file is met, or any other string differing from the positions-charge format. The ligands file produced by the auxiliary program `mk_lig`, which adds an “end” string below the list, can therefore be used directly, without editing.

### 3.2 Keywords for the INPUT file

The task keywords for the INPUT file are listed in Table 1. The list of possible inline multiplets keywords is given in Table 2. Furthermore, as described Sec. 3.1, the crystal field can be given within a block structure (Table 3).

It is to be noted that a warning will be issued if the given number of electrons in the chosen core shell does not correspond to that of the atomic ground state. The radial wave functions for the ground state will be calculated for the neutral atom in the fundamental state, regardless of the core level occupation given via the user input. Similarly, the radial wave functions of the core-hole state will be obtained by removing one electron from the neutral atom core, and adding one to the valence shell. So in case of mismatch between the number of core electrons declared and the neutral atom core configuration, the interactions will be calculated according to the user input, but the radial wave functions and the value of spin-orbit coupling will be wrong. Note that practical x-ray applications usually consider a fully occupied core as starting point.

The inline keywords for x-ray spectra are gathered in Table 4. The incoming photon energy(ies) for rixs must however be given in a block structure, see Table 5. Please note that if `threshold_corr` is present and non-zero, the rixs input photon energies must be given with respect to this correction.

Task keywords	Possible values and what is calculated.	
<b>new</b>	MULTIPLTS	Multiplets are calculated from scratch.
	MULTO	Eigenstates of the ground state.
	MULT	Eigenstates of the ground state and the core-hole state.
	XAS	Eigenstates+XAS for incoming polarisation <code>polar_in</code> .
	XAS_AV	Eigenstates+XAS for averaged polarisations $\perp$ to incoming beam <code>beam_in</code> .
	XAS_totAV	Eigenstates+XAS for space-averaged polarisation directions.
	RIXS	Eigenstates+RIXS for incoming polarisation <code>polar_in</code> and outgoing polarisation <code>polar_out</code> .
	RIXS_AVOUT	Eigenstates+RIXS for incoming polarisation <code>polar_in</code> and averaged polarisations $\perp$ to <code>beam_out</code> .
RIXS_totAVOUT	Eigenstates+RIXS for incoming polarisation <code>polar_in</code> and space-averaged outgoing polarisations.	
<b>continue</b>	Multiplets are read from files (from the previous run). All XAS and RIXS as above, without recalculating the eigenstates. Ex: <code>continue RIXS_AVOUT</code>	

Table 1: Task keywords for the INPUT file.

The broadening may be increased (or decreased) linearly between given energy values, provided those energy values are within the spectrum boundaries. For the core-hole state, the spectrum boundaries are 15 eV above and below the highest, respectively the lowest, core-hole state eigenvalues. For rixs, the final state broadening may also be increased (or decreased) between input values, if these are set between 15 eV above the highest, and 1 eV below the lowest, ground state eigenvalues. The keywords controlling the change in broadening are collected in Table 6.

A further set of keywords allows the user to affect the scale of the spectra (Table 7).

It is also possible to shrink the energy range of the spectrum using the keywords in Table 8. Changing `emin` and `emax` does not however change

Keywords	Description	Default
<code>atom</code>	Atomic symbol of the emitter. Ex: <code>atom Fe</code>	
<code>ground_state</code>	Core-valence ground state configuration in standard form. Ex: <code>ground_state 2p6 3d1</code>	
<code>xtal_file</code>	Name of the ligand file. Ex: <code>xtal_file myligands.txt</code>	
<code>scaler_coulomb</code>	Scaling parameter for Coulomb interaction. Ex: <code>scaler_coulomb 0.8</code>	1.0
<code>scaler_xtal_field</code>	Scaling parameter for crystal field. Ex: <code>scaler_xtal_field 1.4</code>	1.0
<code>scaler_so_coupling</code>	Scaling parameter for spin-orbit coupling. Ex: <code>scaler_so_coupling 0.97</code>	1.0
<code>scaler_so_core</code>	Spin-orbit scaling parameter for core electrons only. Ex: <code>scaler_so_core 0</code>	1.0
<code>scaler_so_val</code>	spin-orbit scaling parameter for valence electrons only. Ex: <code>scaler_so_val 0.95</code>	1.0

Table 2: Inline multiplets keywords.

Keywords	Description	Default
<code>begin_xtal</code> <code>end_xtal</code>	Wrap crystal field list, see Sec. 3.1	

Table 3: Crystal field as block entry.

the resolution, which depends on the broadening.

### 3.3 Parsing rules for the INPUT file

- The code stops with an error if the INPUT file is missing.
- The INPUT file is read line by line, and there should be only one instruction per line.
- Blank lines and lines starting with # are ignored, unless inside a block structure, where they are likely to cause an error.
- The program will stop with an error if a keyword cannot be recognised, or if the wrong type of values are given for a particular keyword.
- For each keyword, the program reads only the required values; the rest of the line is ignored. It is therefore possible to write a short comment on the same line, following the keyword and its values.

Keywords	Description	Default
<code>core_hole_broad</code>	Core-hole lifetime lorentzian broadening, in eV ( $\geq 0$ ). Ex: <code>core_hole_broad 0.6</code>	0.2
<code>threshold_corr</code>	Spectra energy shift, in eV. Ex: <code>threshold_corr 10.3</code>	0.0
<code>final_state_broad</code>	Final state lorentzian broadening for rixs, in eV ( $\geq 0$ ). Ex: <code>final_state_broad 0.05</code>	0.1
<code>polar_in</code>	Linear polarisation direction of the incoming light, in the crystal field frame. Ex: <code>polar_in -0.2 -3.1 0.65</code>	0 0 1
<code>polar_out</code>	Only read for RIXS task: linear polarisation direction of the outgoing light, in the crystal field frame. Ex: <code>polar_out -0.2 -3.1 0.65</code>	0 0 1
<code>beam_in</code>	Only read for XAS_AV or XMCD task: direction of the incoming light beam, in the crystal field frame. Ex: <code>beam_in -0.2 -3.1 0.65</code>	0 0 1
<code>beam_out</code>	Only read for RIXS_AVOU task: direction of the outgoing light beam, in the crystal field frame. Ex: <code>beam_out -0.2 -3.1 0.65</code>	0 0 1

Table 4: Inline x-ray spectra keywords.

- Entries can be listed in any order.
- All entries are read, but not all entries are necessarily used. For instance, if the task is declared as `new MULT` and the entry `polar_in 0 1 0` is also present in the `INPUT` file, then `polar_in` will be read, but not used.
- If one keyword is entered more than once, only the last value (or block) is registered; all previous entries are ignored.
- As a minimum, the atom and the ground state have to be given, as well as the task. Omission of either will cause the program to stop. For rixs tasks, the code will also stop if no input photon energies are found.
- As all lines of the `INPUT` file are read and interpreted (unless preceded by `#`), the code will stop if an `xtal_file` is declared but not present in the directory, even if `scaler_xtal_field` is set to 0.

Keywords	Description	Default
<code>begin_photon</code> <code>end_photon</code>	Wrap photon energy list. Ex: <code>begin_photon</code> 307.5 310.2 <code>end_photon</code>	

Table 5: RIXS incoming photon energy(ies) as block entry.

Keywords	Description and default
<code>deltag1</code> <code>wming1</code> <code>wmaxg1</code>	Core-hole broadening increased linearly from <code>core_hole_broad</code> to <code>core_hole_broad + deltag1</code> between <code>wming1</code> and <code>wmaxg1</code> . Ex: <code>wming1</code> 716.0 <code>wmaxg1</code> 720.0 <code>deltag1</code> 0.16 Default: <code>core_hole_broad</code> const
<code>deltagl</code> <code>wmingl</code> <code>wmaxgl</code>	Read for rixs only: final state broadening increased linearly from <code>final_state_broad</code> to <code>final_state_broad + deltagl</code> between <code>-wmingl</code> and <code>-wmaxgl</code> . Ex: <code>wmingl</code> 2.3 <code>wmaxgl</code> 4.0 <code>deltagl</code> 0.16 Default: <code>final_state_broad</code> const

Table 6: Keywords controlling linear increase in the broadening.

- CAUTION: If not stated otherwise above, everything has a default value (like the core-hole width, the incoming polarisation directions etc). So if `new RIXS_AVOUT` is declared as the task, but no `beam_out` is defined, the default (0 0 1) will be used.
- The scaling parameter for the spin-orbit coupling can be set separately for the core and the valence electrons (`scaler_so_core`, `scaler_so_val`), or for both core and electrons indistinctly (`scaler_so_coupling`). By default all those parameters are set to 1. If both the global `scaler_so_coupling` and `scaler_so_core`, respectively `scaler_so_val`, are given in INPUT, then the core, respectively valence, parameter overrides the global parameter.
- CAUTION: there are two points to keep in mind when applying `continue` to the task. The first is that the INPUT file is read again in its entirety even if the task keyword `continue` is in use, however, only the elements needed for the task continuation will apply. So for example modify-



Keywords	Description	Default
<code>spect_multip</code>	Multiplies the spectrum by a factor. Ex: <code>spect_multip 1.2</code>	1.0
<code>spect_yshift</code>	Shifts the spectrum along the y axis by an amount (arbitrary unit). Ex: <code>spect_yshift 3.0</code>	0.0

Table 7: Keywords that affect the scale of the spectra.

Keywords	Description and default
<code>spect_emin</code> <code>spect_emax</code>	Sets the lower and higher boundary for the spectrum. Ex: <code>spect_emin 270.0</code> <code>spect_emax 300.0</code> Default: for xas/xmcd, 15 eV above and below the range of core-hole state eigenvalues; for rixs, 15 eV above and 1 eV below the range of ground-state eigenvalues.

Table 8: Keywords for shrinking the energy range of the spectra.

ing the crystal field entry between a **new MULT** run and a **continue XAS** run will have no effect: you must run **new MULT** again or do **new XAS**. Modification to `core_hole_broad` would however apply, since this property is not affecting the multiplets, only the spectrum. The second is that the information on the number of eigenstates and their values is read from the `*.dat` files produced by the first run of the code. *Do not temper with these files*, as this may cause anything from an unpredictable outcome to a code crash.

## 4 The output files

Before running the code, the user has as a minimum an `INPUT` file in the working directory, and if applicable, a `ligands` file. Running the code produces an output which can be collected in a `log` file, as explained in Sec. 2. The `log` file writes out the input the code has been using, and lists the eigenvalues obtained. It is therefore important to always inspect the `log` file for potential errors and to verify that the input is what the user intended.

Several files with the `*.dat` extension will be created by `multiX`. The files ending in `*0.dat` relates to the ground state configuration, while those ending in `*1.dat` relates to the core-hole state. In principle, if produced, only the following files are of interest to the user:

- `spect-xas.dat`: produced by invocation of the `xas` module. This is a header-less file containing three columns. The first two are what the user is generally interested in:

```
<photon energies> <xas intensity>
```

The third column is generated from the real part of  $\frac{1}{\hbar\omega + E_0 - E_i + i\Gamma}$ , instead of the imaginary part that simulate the XAS intensity.

- `spect-rixs.dat`: produced by invocation of the `rixs` module. This is a header-less file containing three columns:

```
<-energy loss> <energy photon out> <rixs intensity>
```

Column 1 and 3 are the quantities usually plotted.

The spectra files can typically be read and plotted by external programs that take columns as input, such as `gnuplot`, `grace` etc.

## 5 Examples

The examples shown here can be found in the `inputs_examples` subdirectory of the bundle distribution.

### 5.1 Calculating multiplets

The example in `INPUT 2` calculates multiplets for both the ground state and the core-hole state, but does not produce any spectral file. As ligand file `ligands.txt`, we can use the somewhat unrealistic example 2, which has only one  $-2$  charge along the  $x$ -axis:

```
new MULT

atom Mn
ground_state 2p6 3d5

scaler_coulomb 0.80
scaler_xtal_field 1.4
scaler_so_coupling 0.97

xtal_file ligands.txt
```

**INPUT example 2:** Only multiplets calculation.

```
1.000000
-2.403963 -0.00000 0.0000 -2.00
End
```

**Ligands file example 2:** A somewhat unrealistic ligand file.

## 5.2 Calculating XAS spectra

We would like now to proceed with producing the XAS spectrum for a given polarisation direction. Continuing from the multiplet calculation in Sec. 5.1 (for calculation “from scratch”, just replace `continue` by `new`), we would modify the `INPUT` so that it reads like `INPUT example 3`.

Running `multiX` generates the output file `spect-xas.dat`, which we can save, plot etc. Maybe we have a XAS experiment to compare with, and we notice that the whole spectrum is shifted to the left by 6.3 eV. We can correct the calculated curve by adding `threshold_corr 6.3` to the `INPUT`. Also, the example 3 assumes that the incoming polarisation is along the bond. Maybe we would like get the spectrum for a polarisation perpendicular to that bond. Having saved the previous `spect-xas.dat` under a different name, we run the program again this time with `INPUT example 4`.

The resulting curves are plotted in Fig. 1(a).

## 5.3 Calculating RIXS spectra

After inspecting the XAS spectrum, we decide to calculate the RIXS spectra at the (corrected) energy 640.0 eV. The incoming horizontal polarisation is given with respect to the crystal field, say (0.3420 0 0.9397) in this completely fictitious example. We collect the outgoing light at 90° from the incoming beam (`INPUT example 5`). Maybe we do not believe that the final state broadening is constant over the range. We can increase the broadening between 0 and 6 eV, from 0.05 to 0.15 eV, as shown in `INPUT example 6`.

```

continue XAS

atom Mn
ground_state 2p6 3d5

scaler_coulomb 0.80
scaler_so_coupling 0.97
scaler_xtal_field 1.4

xtal_file ligands.txt

polar_in 1 0 0

core_hole_broad 0.3

```

INPUT example 3: Calculating the XAS spectrum.

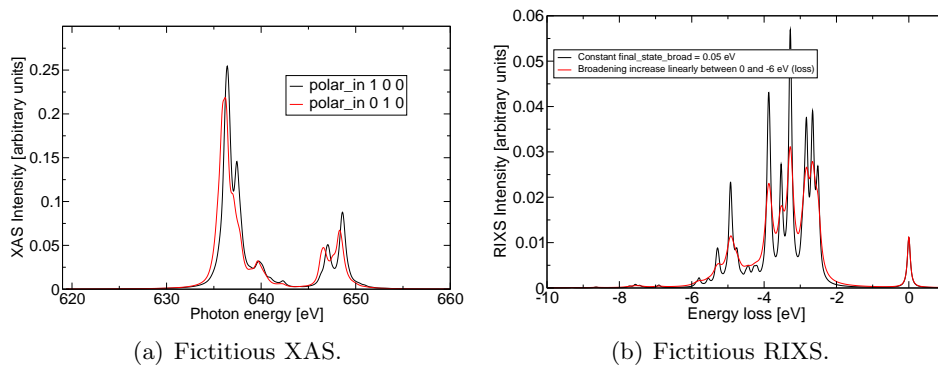


Figure 1: Spectra produced using the INPUT files in Sec. 5.

Fig. 1(b) shows the effect of the broadening on that example.

```
continue XAS

atom Mn
ground_state 2p6 3d5

scaler_coulomb 0.80
scaler_so_coupling 0.97
scaler_xtal_field 1.4

xtal_file ligands.txt

#polar_in 1 0 0 #pol 1
polar_in 0 1 0 #pol 2

core_hole_broad 0.3

threshold_corr 6.3
```

INPUT **example 4:** Another XAS calculation.

```
continue RIXS_AVOUT

atom Mn
ground_state 2p6 3d5

scaler_coulomb 0.80
scaler_so_coupling 0.97
scaler_xtal_field 1.4

xtal_file ligands.txt

#polar_in 1 0 0 #pol 1
#polar_in 0 1 0 #pol 2

polar_in 0.3420 0 0.9397 #pol LH
beam_out 0.3420 0 0.9397

core_hole_broad 0.3
final_state_broad 0.05

threshold_corr 6.3

begin_photen
640.05
end_photen
```

INPUT example 5: RIXS calculation.

```
continue RIXS_AVOU

atom Mn
ground_state 2p6 3d5

scaler_coulomb 0.80
scaler_so_coupling 0.97
scaler_xtal_field 1.4

xtal_file ligands.txt

#polar_in 1 0 0 #pol 1
#polar_in 0 1 0 #pol 2

polar_in 0.3420 0 0.9397 #pol LH
beam_out 0.3420 0 0.9397

core_hole_broad 0.3
final_state_broad 0.05

threshold_corr 6.3

begin_photen
640.05
end_photen

wmingl 0.0
wmaxgl 6
deltagl 0.1
```

INPUT **example 6**: RIXS calculation with a linear increase of the broadening through a part of the energy loss range.

## 6 Auxiliary programs

Auxiliary programs are found in the `utilities` subdirectory of the bundle distribution. It is assumed that this directory is in your path. If this is not the case, just like for `multiX`, you will have to precede the command with the full path to the program, or copy the program locally and precede the command with `./` (bash shell).

### 6.1 Generating the ligands file

A program called `mk_lig` may be used to generate ligand files for crystal structures that include neighbours up to a given distance. It uses an input file `inc_file` containing the three vector spanning the unit cell, and the name and position of the atoms in the unit cell. A cif file usually only lists the atoms in the irreducible unit cell, so it is usually necessary to use an external program to explicitly output all the atom positions, and copy them into the `inc_file`. The format of an `inc_file` is

```
$cell in angstroems of bohrs; from ICSD struc. num xxxxx
  a1 a2 a3
  b1 b2 b3
  c1 c2 c3

$coordinates
Ti x1 y1 z1
Ti x2 y2 z2
...
0 xn yn zn
$end
```

The code reads the unit cell vectors below the keyword `$cell` and the list of atom positions below `$coord`. **The atom at the centre, the emitter, must be placed at the top of this list** (Ti x1,y1,z1 in the example above). The unit for the vectors and positions is either Ångström or bohr. The output of the code, a file called `ligands.txt`, will however be in Ångström, regardless of the input unit. The unit of the input has to be declared (A or B) in the argument list when calling `mk_lig`. The cutoff for the neighbour list must be given in that same unit. The program `mk_lig` is called with the following list of arguments:

```
mk_lig <inc_file> <cut_off> <A for angs/B for bohrs>
<element> <charge> [<element> <charge> ...]
```

The first argument is the input file name. The second is the cutoff, defining the radius within which neighbours have to be included. The third



argument is the input unit statement and is a single capital letter, either A for Ångström or B for bohr. The next arguments are optional; they should appear in pairs of atom symbol (corresponding to the atoms in the input file) and the charge associated to that ion. For example, for a Ti atom in  $\text{LaTiO}_3$  with  $\text{La}^{3+}$ ,  $\text{Ti}^{+3}$  and  $\text{O}^{-2}$ , with cell vectors and positions in `inc_LaTiO3`, the command would be

```
mk_lig inc_LaTiO3 20 A Ti 3 La 3 0 -2.0
```

Neighbours up to 20 Ångström from the Ti atom will be included.

If the elements are not all declared as arguments, the code does some guessing. O is for example allocated  $-2$  by default, if not otherwise stated in the argument list. For the elements without a default defined in the code, charges are distributed among the undefined elements on the principle of charge neutrality. So in this particular case, the same result as above would have been obtained with

```
mk_lig inc_LaTiO3 20 A Ti 3
```

## 6.2 Running and plotting scripts

An elementary script named `run_multiX` automates the call to unlimited stack and the piping of the output to the log file:

```
$ run_multiX
```

An interactive facility, `plt_mult`, has been made available for plotting XAS and RIXS spectra. It takes either `xas` or `rixs` as argument:

```
$ plt_mult xas
```

or

```
$ plt_mult rixs
```

The program asks the user if modifications to the plotting defaults are required (you may often answer N) and outputs either the eps file `gxas.eps` or `grix.eps`, according to the argument given. The eps file can be viewed by programs like `ghoscript` or `ghostview`:

```
$ gv grix.eps &
```

If you suspect that `plt_mult` will have suitable defaults, you may want to have a look at the scripts `run_XAS` and `run_RIXS`, which automates the ulimit setup, changes the task in the INPUT file, calls `multiX`, then `plt_mult` accepting the defaults, and invokes the viewer `gv`. It is highly recommended that you check and edit this script if you want to run it.

`./run_XAS`

or

`./run_RIXS`

---