Flexible Atomic Code and Integration with Prism Softwares

Ming Gu
Prism Computational Sciences
and
Space Science Laboratory, UC Berkeley

Download and Installation

- https://github.com/flexible-atomic-code/fac
- git clone https://github.com/flexible-atomic-code/fac
- git pull
- Requires python, c, f77 compilers (gcc and gfortran work fine)
- ./configure --with-mpi=omp --prefix=install_dir
- make
- make install
- make pfac
- make install-pfac
- python setup.py install --prefix=install_dir
- Tutorial examples: https://github.com/flexible-atomic-code/fac_data/lle2023

Electronic configurations & Angular coupling

- Non-relativistic subshells, 1s, 2s, 2p, 3d, 4f, etc
- Relativisitic subshells, 1[s+], 2[p-], 3[d+], “-”: j=l-1/2, “+”: j=l+1/2
- For large orbital angular momentum, 30[26], n=30, l=26
- Configurations, ‘1s2 2s2 2p4’, occupations directly follow subshells
- “*” can denote any orbital angular momenta. ‘3*1’ expands to 3s1, 3p1, 3d1 configurations.
- Restriction on electron occupation, ‘3*10;3d<3’, all possible ways of distributing 10 electrons in 3s, 3p, 3d subshells, with no more than 3 electrons in 3d. Multiple conditions separated by “;” with logical and. ‘3*10;3d>1;3d<4’, 3d must have 2 or 3 electrons.
- Angular momentum coupling, 1s+(1)1.2s+(1)2.2p+(3)3.3s+2(0)3

Atomic processes implemented in FAC

- Structure: solving Dirac equation –> energy levels (QED corrections)
- Radiative transition.
- Electron impact excitation.
- Electron impact ionization.
- Photoionization, radiative recombination.
- Autoionization, dielectronic capture.
- A simple collisional radiative model for spectral modeling
- Excitation and ionization between magnetic sublevels, linear polarization in EBITs.
- External E&B fields.
- Screening potential of plasma electrons.
- 2^{nd} order MBPT + CI
- Dirac R-Matrix for excitation
- Electron impact Stark broadening.

An example, He-like Fe.

```
from pfac.fac import *

WallTime('Start')
SetAtom('Fe’)

Config('g1', '1s2')
Config('g2', '1s1 2*1')
Config('g3', '1s1 3*1')
ListConfig()
ListConfig('ex1a.cfg')

WallTime('Opt')
ConfigEnergy(0)
OptimizeRadial('g1')
ConfigEnergy(1)

WallTime('EN')
Structure('ex1b.en', ['g1','g2'])
Structure('ex1b.en', ['g3'])

MemENTable('ex1b.en')
PrintTable('ex1b.en', 'ex1a.en')
```

```
FAC 1.1.5
Endian  = 0
TSess   = 1665109512
Type    = 1
Verbose = 1
Fe Z    =  26.0
NBlocks = 2
E0      = 0, -1.81043476E+04

NELE    = 2
NLEV    = 7
 ILEV  IBASE    ENERGY       P   VNL   2J
    0    -1  0.00000000E+00 0   100    0 1*2          1s2          1s+2(0)0
    1    -1  6.63537106E+03 0   200    2 1*1.2*1      1s1.2s1      1s+1(1)1.2s+1(1)2
    2    -1  6.66438962E+03 1   201    0 1*1.2*1      1s1.2p1      1s+1(1)1.2p-1(1)0
    3    -1  6.66649019E+03 1   201    2 1*1.2*1      1s1.2p1      1s+1(1)1.2p-1(1)2
    4    -1  6.66764932E+03 0   200    0 1*1.2*1      1s1.2s1      1s+1(1)1.2s+1(1)0
    5    -1  6.68118301E+03 1   201    4 1*1.2*1      1s1.2p1      1s+1(1)1.2p+1(3)4
    6    -1  6.69988981E+03 1   201    2 1*1.2*1      1s1.2p1      1s+1(1)1.2p+1(3)2

NELE    = 2
NLEV    = 10
 ILEV  IBASE    ENERGY       P   VNL   2J
    7    -1  7.86199121E+03 0   300    2 1*1.3*1      1s1.3s1      1s+1(1)1.3s+1(1)2
    8    -1  7.87001039E+03 1   301    0 1*1.3*1      1s1.3p1      1s+1(1)1.3p-1(1)0
    9    -1  7.87045470E+03 0   300    0 1*1.3*1      1s1.3s1      1s+1(1)1.3s+1(1)0
   10    -1  7.87060953E+03 1   301    2 1*1.3*1      1s1.3p1      1s+1(1)1.3p-1(1)2
```

He-like Fe cont…

```
WallTime('TR')
TRTable('ex1b.tr', ['g1'], ['g2'])
TRTable('ex1b.tr', ['g1'], ['g3'])
TRTable('ex1b.tr', ['g2'], ['g2'])
TRTable('ex1b.tr', ['g2'], ['g3'])
TRTable('ex1b.tr', ['g3'], ['g3'])
PrintTable('ex1b.tr', 'ex1a.tr')
```

```
FAC 1.1.5
Endian	= 0
TSess	= 1665111809
Type	= 2
Verbose	= 1
Fe Z	=  26.0
NBlocks	= 5

NELE	= 2
NTRANS	= 4
MULTIP	= 0
GAUGE	= 2
MODE	= 1
   1  2     0  0  6.635371E+03  3.088469E-07  1.966808E+08  3.088469E-07
   3  2     0  0  6.666490E+03  6.558891E-02  4.216122E+13  6.558891E-02
   5  4     0  0  6.681183E+03  1.683081E-05  6.520057E+09  1.683081E-05
   6  2     0  0  6.699890E+03  7.243883E-01  4.703217E+14  7.243883E-01

NELE	= 2
NTRANS	= 7
MULTIP	= 0
GAUGE	= 2
MODE	= 1
   7  2     0  0  7.861991E+03  9.581767E-08  8.566413E+07  9.581767E-08
  10  2     0  0  7.870610E+03  1.377064E-02  1.233841E+13  1.377064E-02
  11  4     0  0  7.875002E+03  4.593787E-06  2.472360E+09  4.593787E-06
  12  4     0  0  7.879353E+03  1.793314E-04  9.662221E+10  1.793314E-04
  13  2     0  0  7.879422E+03  1.057020E-09  9.492052E+05  1.057020E-09
  14  2     0  0  7.880018E+03  1.399493E-01  1.256937E+14  1.399493E-01
  16  4     0  0  7.881321E+03  3.579022E-04  1.929310E+11  3.579022E-04
```

He-like Fe Cont…

```
WallTime('CE')
CETable('ex1b.ce', ['g1'], ['g2'])
CETable('ex1b.ce', ['g1'], ['g3'])
CETable('ex1b.ce', ['g2'], ['g2'])
CETable('ex1b.ce', ['g2'], ['g3'])
PrintTable('ex1b.ce', 'ex1a.ce')

WallTime('Done')
```

```
FAC 1.1.5
Endian        = 0
TSess         = 1665111809
Type          = 3
Verbose       = 1
Fe Z          =  26.0
NBlocks       = 8

NELE          = 2
NTRANS        = 6
QKMODE        = 0
NPARAMS       = 0
MSUB          = 0
PWTYPE        = 0
NTEGRID       = 2
                 6.63537106E+03
                 6.69988981E+03
TE0           =  6.66763038E+03
ETYPE         = 1
NEGRID        = 6
                 3.33381522E+02
                 7.54265046E+03
                 1.71362181E+04
                 2.83443511E+04
                 4.06131653E+04
                 5.35991185E+04
UTYPE         = 1
NUSR          = 6
                 3.33381522E+02
                 7.54265046E+03
                 1.71362181E+04
                 2.83443511E+04
                 4.06131653E+04
                 5.35991185E+04
```

```
  0  0     1  2  6.6354E+03 1
-1.0000E+00  0.0000E+00  0.0000E+00
 3.3338E+02  3.3610E-04  5.7337E-03
 7.5427E+03  1.4461E-04  1.2041E-03
 1.7136E+04  6.9284E-05  3.4093E-04
 2.8344E+04  3.8145E-05  1.2621E-04
 4.0613E+04  2.3517E-05  5.6943E-05
 5.3599E+04  1.5772E-05  2.9597E-05
    0  0     2  0  6.6644E+03 1
-1.0000E+00  0.0000E+00  0.0000E+00
 3.3338E+02  2.0883E-04  3.5477E-03
 7.5427E+03  6.5067E-05  5.4067E-04
 1.7136E+04  2.4767E-05  1.2172E-04
 2.8344E+04  1.1622E-05  3.8421E-05
 4.0613E+04  6.3954E-06  1.5476E-05
 5.3599E+04  3.9500E-06  7.4086E-06
    0  0     3  2  6.6665E+03 1
 5.3544E-04  1.1245E-04  6.6999E+05
 3.3338E+02  7.9381E-04  1.3481E-02
 7.5427E+03  7.3149E-04  6.0774E-03
 1.7136E+04  9.2055E-04  4.5237E-03
 2.8344E+04  1.1411E-03  3.7719E-03
 4.0613E+04  1.3576E-03  3.2849E-03
 5.3599E+04  1.5690E-03  2.9426E-03
```

Save script as ex1.py

Run with python: python ex1.py

Or run with the sfac program: sfac ex1.py

Output files:

ex1a.cfg, ex1b.en, ex1a.en, ex1b.tr, ex1a.tr, ex1b.ce, ex1a.ce

Examining energy file with LevelInfo()

Examining transition file with TRBranch()

Interpolating excitation cross sections with InterpCross()

Thermal excitation rate coefficients with MaxwellRate()

Use pfac.rfac module (Keisuke Fujii)

Radial wavefunctions and atomic states

```
from pfac.fac import *

WallTime('Start')
SetAtom('Fe')

Config('g1', '1s2')
Config('g2', '1s1 2*1')
WallTime('Opt')
OptimizeRadial('g1')
```

```
WallTime('Wavefunctions’)
#bound orbitals
WaveFuncTable('w1s.txt', 1, -1, 0)
WaveFuncTable('w2p-.txt', 2, 1, 0)
WaveFuncTable('w3d+.txt', 3, -2, 0)

#free orbitals
WaveFuncTable('ws.txt', 0, -1, 2e3)
WaveFuncTable('wp-.txt', 0, 1, 2e3)
```

```
WallTime('EN')
Structure('ex2b.en', ['g1','g2'])
MemENTable('ex2b.en')
PrintTable('ex2b.en', 'ex2a.en')

BasisTable('ex2a.bs')
BasisTable('ex2a', 10)

WallTime('Done')
```

rfac.read_wfun()

jj2lsj from GRASP.

Collisional radiative model

```
from pfac.crm import *

AddIon(2, 0.0, 'ex1b')
SetBlocks(-1)
SetEleDist(0, 3e3, -1, -1)
SetTRRates(0)
SetCERates(1)
SetAbund(2, 1.0)
SetEleDensity(1.0)
InitBlocks()
LevelPopulation()
```

```
DumpRates('ex1a.r0', 2, 0, -1, 1)
DumpRates('ex1a.r1', 2, 1, -1, 1)
DumpRates('ex1a.r2', 2, 2, -1, 1)
DumpRates('ex1a.r3', 2, 3, -1, 1)

SpecTable('ex1b.sp', 0)
PrintTable('ex1b.sp', 'ex1a.sp')

SelectLines('ex1b.sp', 'ex1a.ln', 2, 0, 6e3, 10e3, 1e-5)
```

Multi-ion models with recombination & ionization

- Radial potential needs to be optimized for individual ions.
- FAC requires orthogonality of radial wavefunctions of same symmetry.
- One has a choice of using the potential optimized for the ionized or recombined ion.
- Let’s build a spectral model of H-like, He-like and Li-like Fe ions, with ionization and recombination coupling.

python ex4.py 26 2 output: Fe02b.en .tr, .ce, .ci, .rr, .ai

```
from pfac.fac import *
import sys, os

z = int(sys.argv[1])
k = int(sys.argv[2])

InitializeMPI(4)

a = ATOMICSYMBOL[z]
p = '%s%02d'%(a,k)
pb = p+'b'
pa = p+'a'

WallTime('Beg '+p)
SetAtom(a)
```

```
if k == 1:
   Config('g1', '1s1')
   Config('g2', '2*1')
   Config('i1', ' ')
   gcs = ['g1', 'g2']
   dcs = []
   ics = ['i1']
elif k == 2:
   Config('g1', '1s2')
   Config('g2', '1s1 2*1')
   Config('d2', '2*2')
   Config('i1', '1s1')
   gcs = ['g1','g2']
   dcs = ['d2']
   ics = ['i1']
elif k == 3:
   Config('g1', '1s2 2*1')
   Config('i1', '1s2')
   Config('i2', '1s1 2*1')
   gcs = ['g1']
   dcs = []
   ics = ['i1','i2']

ListConfig(pa+'.cfg')
```

```
WallTime('OPT')

ConfigEnergy(0)
OptimizeRadial('g1')
ConfigEnergy(1)

WallTime('EN')
Structure(pb+'.en', gcs)
Structure(pb+'.en', ics)
if len(dcs) > 0:
   Structure(pb+'.en', dcs)
MemENTable(pb+'.en')
PrintTable(pb+'.en', pa+'.en')

WallTime('TR')
for i in range(len(gcs)):
   for j in range(i,len(gcs)):
      TRTable(pb+'.tr', [gcs[i]], [gcs[j]])
   for j in range(len(dcs)):
      TRTable(pb+'.tr', [gcs[i]], [dcs[j]])
PrintTable(pb+'.tr', pa+'.tr')
```

```
WallTime('CE')
for i in range(len(gcs)):
   for j in range(i,len(gcs)):
      CETable(pb+'.ce', [gcs[i]],
[gcs[j]])
PrintTable(pb+'.ce', pa+'.ce')

WallTime('CI')
for i in range(len(gcs)):
   for j in range(len(ics)):
      CITable(pb+'.ci', [gcs[i]], [ics[j]])
PrintTable(pb+'.ci', pa+'.ci')

WallTime('RR')
for i in range(len(gcs)):
   for j in range(len(ics)):
      RRTable(pb+'.rr', [gcs[i]], [ics[j]])
PrintTable(pb+'.rr', pa+'.rr')

WallTime('AI')
for i in range(len(dcs)):
   for j in range(len(ics)):
      AITable(pb+'.ai', [dcs[i]], [ics[j]])
PrintTable(pb+'.ai', pa+'.ai')

WallTime('Done')

FinalizeMPI()
```

python ex5.py 26 2 1e3 1e2 0.1 0.1

```
from pfac.crm import *
from pfac import fac
import sys, os

z = int(sys.argv[1])
k = int(sys.argv[2])
t = float(sys.argv[3])
d = float(sys.argv[4])
ai = float(sys.argv[5])
ar = float(sys.argv[6])

a = fac.ATOMICSYMBOL[z]

p = '%s%02d'%(a,k)
pbi = '%s%02db'%(a,k-1)
pbr = '%s%02db'%(a,k+1)
pb = p+'b'
pa = p+'a'

WallTime('AddIons')
AddIon(k-1, 0.0, pbi)
AddIon(k, 0.0, pb)
AddIon(k+1, 0.0, pbr)

SetBlocks(0)

SetEleDist(0, t, -1, -1)

WallTime('SetRates')
SetTRRates(0)
SetCERates(1)
SetCIRates(1)
SetRRRates(0)
SetAIRates(1)

SetAbund(k-1, ai)
SetAbund(k, 1.0)
SetAbund(k+1, ar)

SetEleDensity(d)

InitBlocks()
LevelPopulation()
DumpRates(pa+'.r0', -1, 0, -1, 1)
DumpRates(pa+'.r1', -1, 1, -1, 1)
DumpRates(pa+'.r3', -1, 3, -1, 1)

SpecTable(pb+'.sp', 0)
PrintTable(pb+'.sp', pa+'.sp')

if os.path.exists(pa+'.ln'):
    os.system('rm '+pa+'.ln’)

SelectLines(pb+'.sp', pa+'.ln', 1, 201, 0, 1e5, 0)
SelectLines(pb+'.sp', pa+'.ln', 2, 201, 0, 1e5, 0)
SelectLines(pb+'.sp', pa+'.ln', 2, 20201, 0, 1e5, 0)

WallTime('Done')
```

Plasma Screening Potential

Effects of plasma electrons and ions on the atomic structure.

PlasmaScreen(zp, ne, te, mode, zs)

zp = free electrons per ion

ne = electron density in 10^24

te = electron temperature in eV

mode = 0 Ion –sphere model (if te=0, uniform ion sphere)

= 1 Debye-Huckel screening

= 2 Stward-Pyatt potential

zs = an extra parameter for SP mode

define screening electrons from a plasma mixture, zs=<z^2>/<z>

Handling FAC output files (ascii)

MemENTable(enbfile)

PrintTable(bfile, afile)

Converts FAC binary output to ascii format.

pfac.rfac module provides functions to read ascii files.

read_en(fname), read_tr(fname), read_ce(fname), read_ci(fname)

read_rr(fname), read_ai(fname), read_wfun(fname), read_pot(fname)

read_sp(fname), read_bst(fname), read_mix(fname)

load_fac(fname).

d = FLEV(fname) returns a python obj for the level structure.

d1 = FLEV(f1), d2=FLEV(f2)

d1.match(d2) attempts to match the level structure of f1 to that of f2.

Handling FAC output files (binary)

MemENTable(ebfile)

InterpCross(bfile, ofile, i0, i1, elist, mp)

Interpolate the cross sections for a specified list of energies.

Works on binary files from CE, CI, RR.

i0 is the lower (bound) level index

i1 is the upper (ionized) level index

elist a list of energies in eV

mp = 0, elist is given as the incident electron, mp=1 for outgoing.

MaxwellRate(bfile, ofile, i0, i1, tlist)

Integrate cross section over thermal distribution to obtain rate coef.

Handling FAC output files (binary cont.)

TotalRRCross(bfile, ofile, i0, elist)

Sum RR cross sections from level i0 to all possible final states.

TotalCICross(bfile, ofile, i0, elist)

Sum CI cross sections from initial bound state i0 to all ionized states.

TotalPICross(bfile, ofile, i0, elist)

Sum PI cross sections from initial bound state i0 to all ionized states.

JoinTable(f0, f1, fc)

Join two binary files f0 and f1 to a combined file fc.

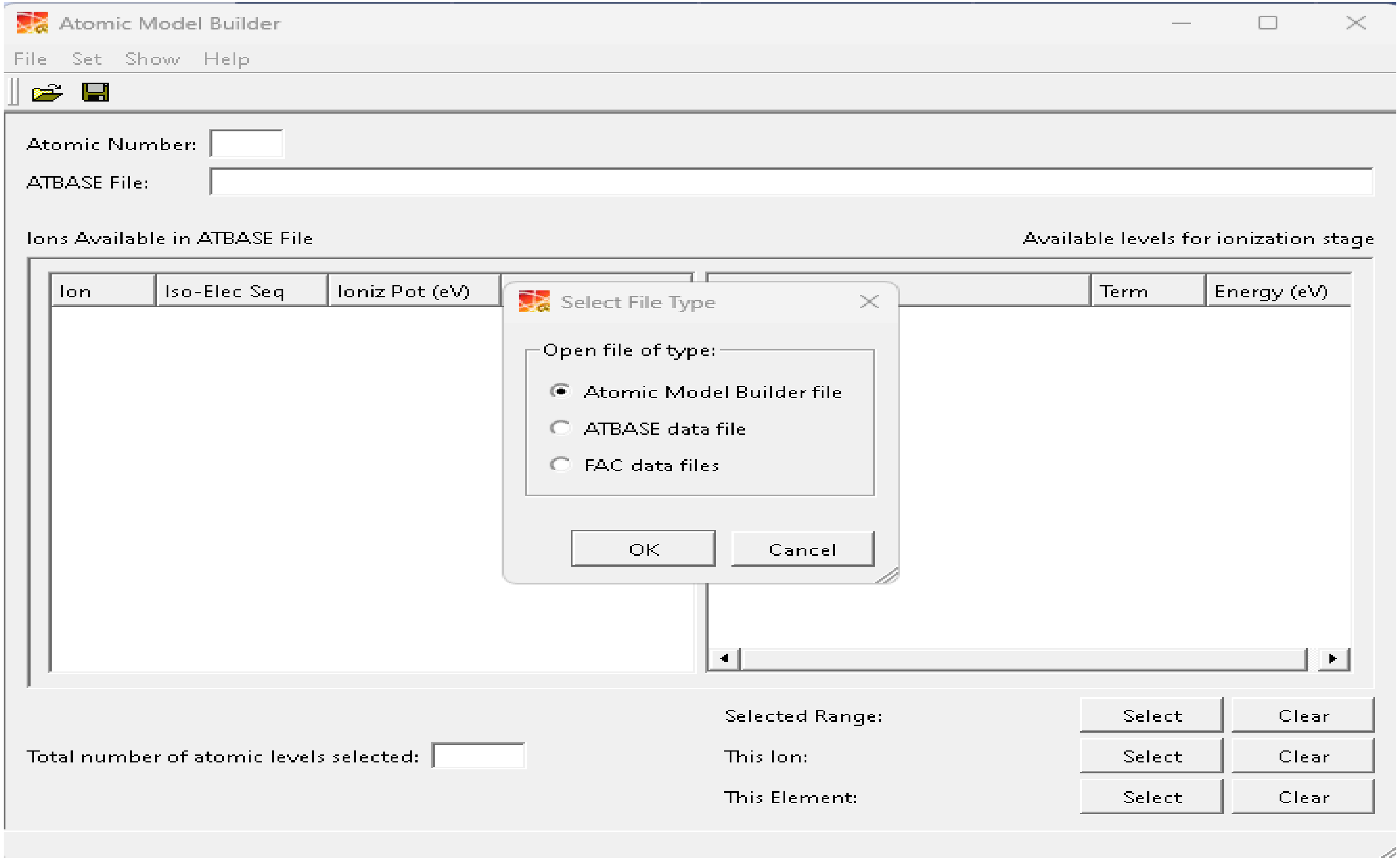
Additional Functionalities

- 2^{nd} order MBPT
- Rmatrix
- Electron impact Stark width and shift
- Effects of external E & B fields
- Transitions between magnetic sublevels, polarization
- Unresolved transition arrays (UTA)
- Charge exchange in Landau-Zener approximation
- Muonic atom (or any other charged particles instead of electron)
- Average atom models

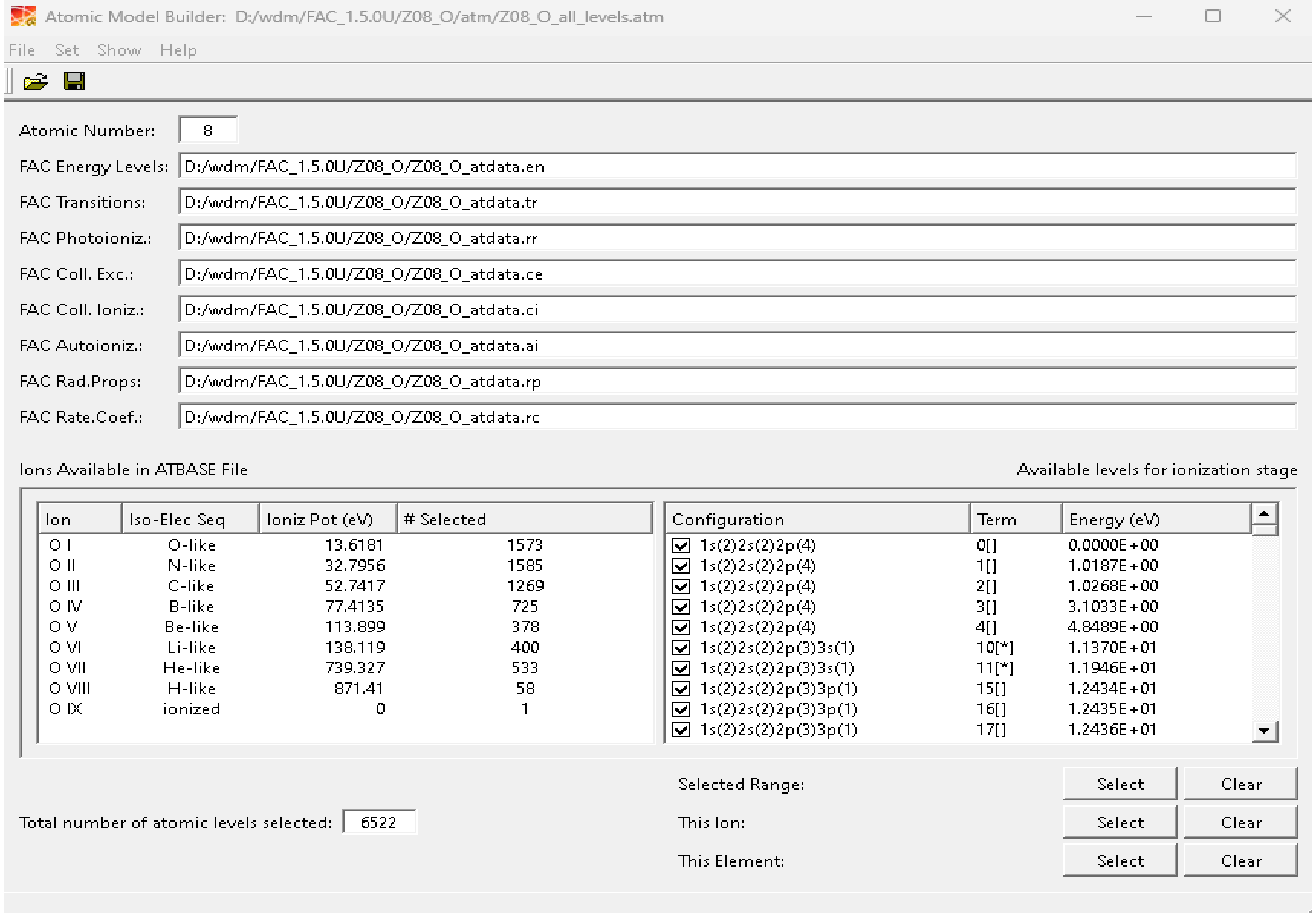
Integration with Prism Softwares

- Prism codes use the ATOMIC MODEL BUILDER to create the atm file as an interface to the underlying atomic database
- A new option was added to read the FAC binary output files. Its use in Prism applications is encapsulated in the atm file, and largely transparent to users.
- Several utility functions are created in FAC to make this possible.
- CombineDBase() to combine atomic data from different ions of the same element into a single file, with correct level index mapping
- RateCoefficients() to tabulate recombination, ionization and excitation rate coefficients due to intermediate autoionizing levels not included in the databse explicitly.
- Output radial moments and binding energies for single electron wavefunctions needed for continuum lowering models.
- A new Stark broadening model based on the Quasi-contiguous approximation of Stambulchick et al.

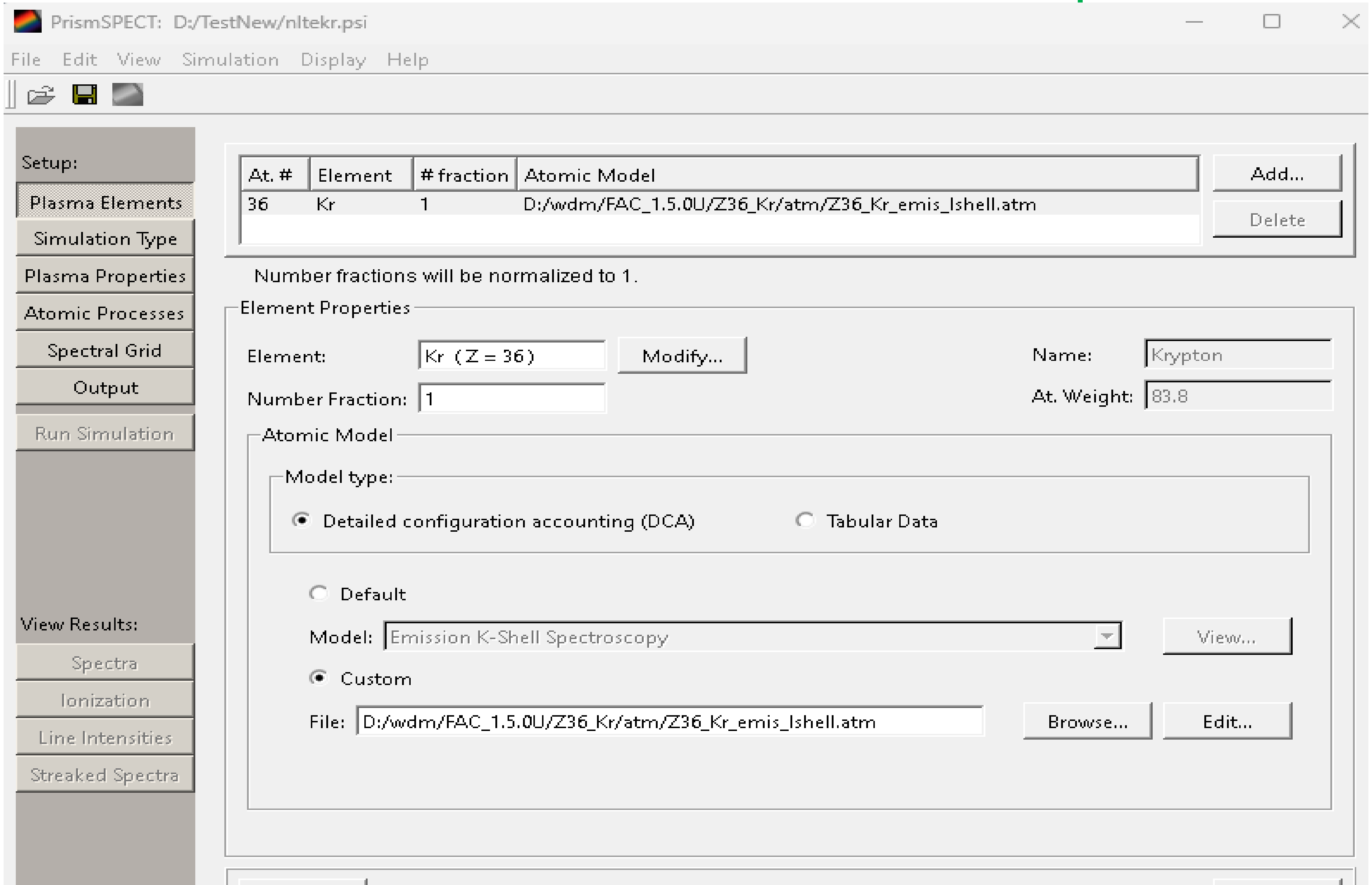
New option in Prism Atomic Model Builder



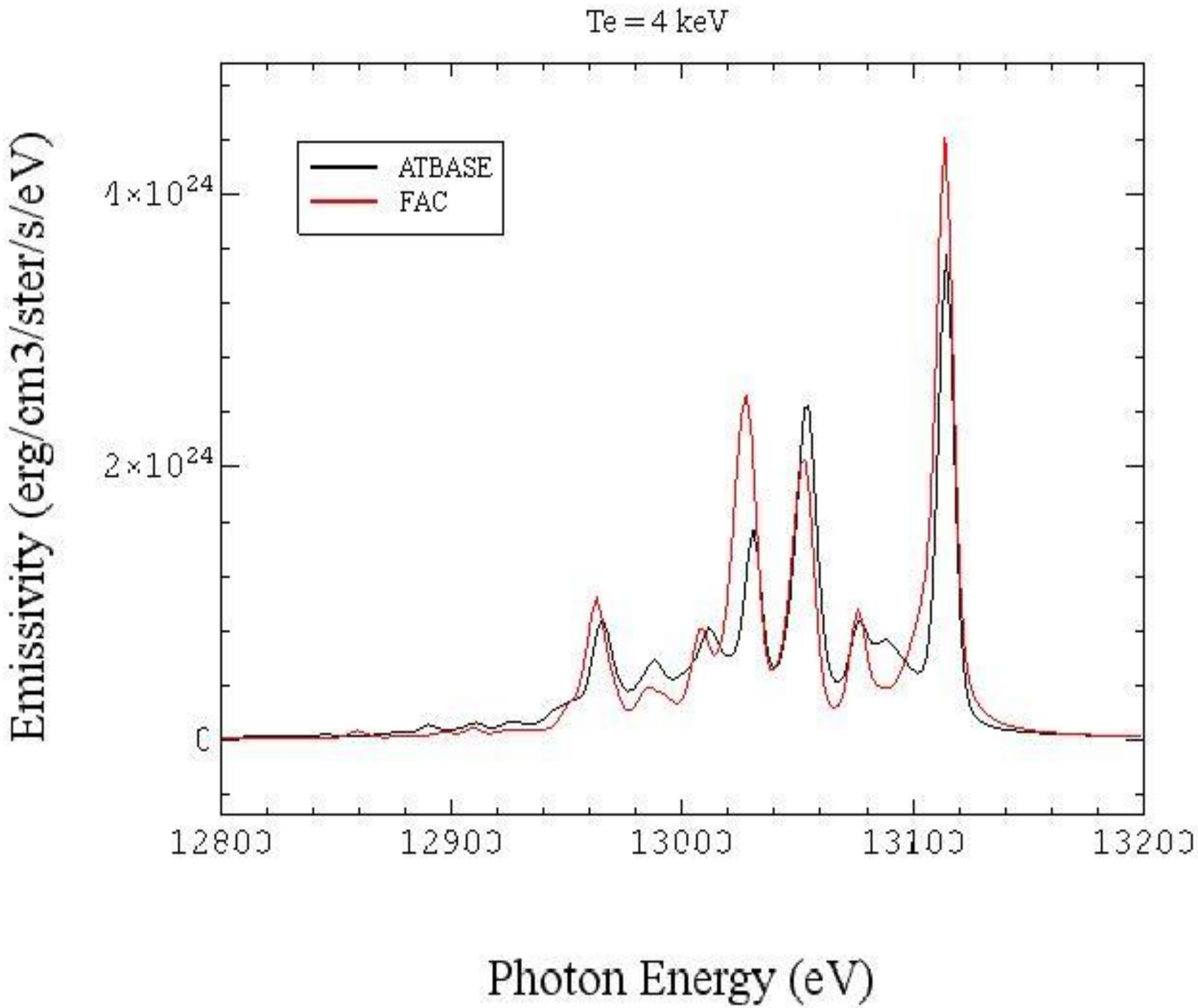
FAC data files needed for AMB



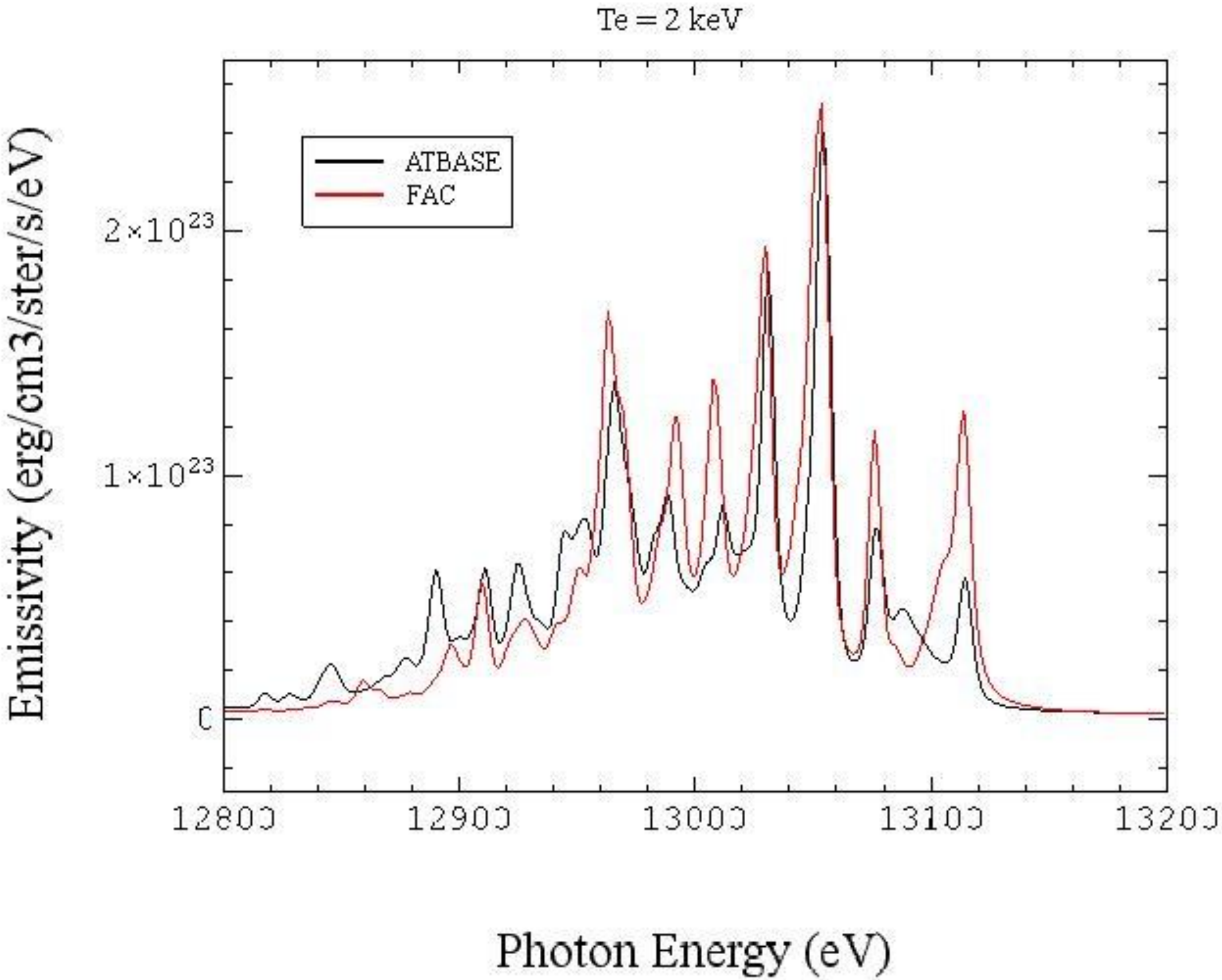
Use FAC data in PrismSPECT or Spect3D



Comparison of FAC and ATBASE. Kr at Te=4keV



Comparison of FAC and ATBASE, Kr at Te=2keV



Comparison of ATBASE and FAC, Mean Z