

# Running RIXS calculations with eSPec: practical guide

Vinícius Vaz da Cruz and Freddy Fernandes Guimarães

## I. GENERAL IDEA

The eSPec package, is in a wave packet propagation program which has been applied to a wide variety of dynamical problems, from absorption spectrum to reaction dynamics. In order to apply it to RIXS cross section calculations we ought to perform two different wavepacket propagations. The procedure goes as follows:

I) first we propagate the nuclear wavefunction from the initial state of the system on the intermediate state potential surface

II) we apply the eSPec-Raman module to obtain the new starting conditions, which depend on the incoming photon frequency.

II) finally we perform a second propagation of the obtained wave packets on the final state potential surface.

The eSPec already computes the auto-correlation function and the corresponding cross-section in the .SPECTRUM mode.

## II. THE eSPec-RAMAN MODULE

The eSPec-RAMAN module was coded to provide the appropriate wave packets for the final state propagation, in other words it solves the following integral

$$|\Phi(0)\rangle = \int_0^\infty e^{-\Gamma t} e^{-i(\hat{H}_c - E_0 - \omega)t} |0\rangle dt, \quad (1)$$

now, eSPec does the propagation shifting the potentials to zero, so we should make  $\hat{H}'_c = \hat{H}_c - \omega_{c0}$ , then we get

$$|\Phi(0)\rangle = \int_0^\infty e^{-\Gamma t} e^{-i(\hat{H}'_c + \omega_{c0} - E_0 - \omega)t} |0\rangle dt, \quad (2)$$

defining

$$\phi_c(t) = e^{-i\hat{H}'_c t} |0\rangle \quad (3)$$

we can identify eq. (2) as a fourier transform (acutally here we have a half fourier transform, but it is possible to rewrite it as an ordinary transform, check Faris's file for detailed derivation)

$$|\Phi(0)\rangle = \int_0^\infty e^{-\Gamma t} e^{-i(\omega_{c0} - E_0)t} \phi_c(t) e^{i\omega t} dt, \quad (4)$$

since  $|\Phi(0)\rangle$  is function of the detuning, we can write this dependence explicitly as  $|\Phi(0)\rangle \equiv \Phi(0, \omega)$  now, if we consider the following shift property of the fourier transform:

$$\int_{-\infty}^\infty f(t) e^{i\omega t} dt = g(\omega), \quad (5)$$

$$\int_{-\infty}^\infty e^{\pm iat} f(t) e^{i\omega t} dt = g(\omega \pm a) \quad (6)$$

we can readily see that eq. (4) can be written

$$\Phi(0, \omega - \omega_{c0} + E_0) = \int_0^\infty e^{-\Gamma t} e^{-i(\omega_{c0} - E_0)t} \phi_c(t) e^{i\omega t} dt \quad (7)$$

where we can define  $\bar{\omega} = \omega - \omega_{c0} + E_0$ , which is the shifted frequency used in the calculation of the fourier transform. Now, we define the shifted resonance frequency, which is the one associated with  $\bar{\omega}$  rather than  $\omega$ . Since the original vertical resonance frequency is defined as  $\omega_{res} = V_d(R_0^g) - V_g(R_0^g) - E_0$ , where  $V_d$  and  $V_g$  are the core-excited and ground state potentials respectively, and  $R_0^g$  is the minimum position of the ground state potential, we may write  $\bar{\omega}_{res} = \omega_{res} - \omega_{c0} + E_0$ , so, summarizing, we have

$$\bar{\omega} = \omega - \omega_{c0} + E_0 \quad (8)$$

$$\omega_{res} = V_d(R_0^g) - V_g(R_0^g) - E_0 \quad (9)$$

$$\bar{\omega}_{res} = \omega_{res} - \omega_{c0} + E_0 \quad (10)$$

$$\Omega = \omega - \omega_{res} = \bar{\omega} - \bar{\omega}_{res} \quad (11)$$

this is the way the program computes different photon frequencies, by using the shift property of the Fourier transform. And it requires as input the shifted resonance frequency  $\bar{\omega}_{res}$  and the desired values of detuning  $\Omega$ .

Then we solve it using FFT techniques (such as the FFTW library). We note, however, that to take full advantage of this methodology, one should request all desired wavepackets in a single run, since the computational cost of obtaining the intermediate wavepacket  $|\Phi(0)\rangle$  for a single photon frequency, and for several ones is virtually the same. For this reason everytime you run the raman code, it will save to disk a file with the spline coefficients (fft\_spline.bcoef) of the fourier transformed wavepacket, so that on a second run, the computational time of obtaining a different  $|\Phi(0)\rangle$  will be just the one of reading the spline file and printing the desired function.

### III. THE run\_raman.sh SCRIPT

The run\_raman.sh script, was written following the ideas outline in the previous section. It runs in several modes, which are given as flags when running the script. the -all flag, runs all three steps of the calculation. -init only runs the propagation of the initial state on the intermediate potential. -cond only generates the new initial conditions (you must have run -init previously) from a given propagation

So you should run it like this

```
$ ./run_raman.sh -flag input
```

**-init** in this mode, the script solves the time independent problem to obtain the ground state wavefunction  $|0\rangle$ , then proceeds to propagate it on the core-excited potential. The wavefunction evolution is saved on disk to a series of files called ReIm\_, which are kept in the folder wf\_data for the next step

**-cond** in this mode, we run the eSPec-raman module to obtain the  $|\Phi(0)\rangle$  for the desired values of detuning (and bending states). The wavefunctions generated are saved to files, for the next step.

**-fin** in this mode, we run eSPec to propagate the generated  $|\Phi(0)\rangle$  on the final state potential, the program automatically computes the auto-correlation function, and subsequently the spectrum. The spectra for each detuning required is saved to files \*.spec and have to be shifted like  $\omega' = \omega - \bar{\omega}' - \omega_{0f} + E_0$ , here  $\bar{\omega}'$  is the abscissa in the eSPec output,  $\omega'$  is the emitted photon frequency (desired abscissa),  $\omega$  is the incoming photon frequency,  $\omega_{0f}$  is the energy difference between the ground and final state, finally  $E_0$  is the ground state vibrational energy. This happens because eSPec shifts all potentials to zero.

**-all** intuitively, runs all the option above on a single shot (This mode can be a bit messy on first runs, due to input errors and such, so I recomend running step-by-step at the beginning)

### IV. INPUT KEYWORDS

Now we'll go into details of the input file the script reads.

here is an example of some essential keywords required

```
# RIXS input
#
jobid test
dimension .2D
```

```

npoints 100 100
mass 12.00 12.00

initial_pot  test2d.init
decaying_pot test2d.decay
final_pot    test2d.final

gamma 0.05
step 5D-4
detuning -2.0 -1.0 -0.1 0.0 0.1 1.0 2.0 4.0

Vg_min  0.0000
Vd_min  90.7708
Vf_min  0.367493
Vd_vert 90.7758

fin_time 140.0

absorb_cond 0.0001
absorb_range 0.00 10.00 0.000 10.00

```

Now, we detail each one of them

**jobid** reads a string to serve as name base for all output files (necessary)

**dimension** options .1D, .2D and .2DCT, which specifies the dimension of the calculation

**npoints** number of discretization points on each dimension

**mass** mass in AMU, associated with each dimension (remember to input the three masses in .2DCT calculations)

**initial\_pot** expects the name of the file containing the ground state potential.

**initial\_wf** if you have already computed the initial wavefunction, then you should use this keyword followed by the file containing the initial wavefunction. In this case the keyword **initial\_pot** is not used. It is desirable that the wavefunction is normalized with unitary euclidian norm.

**E0** if you are using a wavefunction computed previously (**initial\_wf**) then you might need to include this keyword, followed by the energy of said wavefunction. The exception being if you already have a file named \$jobid\_init.out, from a previous -init run.

**decayin\_pot** expects the name of the file containing the intermediate state potential.

**final\_pot** expects the name of the file containing the final state potential.

**gamma** Core-excited lifetime broadening in eV.

**step** Time step to be used in the propagations ( $10^{-4}$  recommended, for test runs larger steps may be used).

**fin\_time** Propagation time (femtoseconds) on final state potential.

**Vg\_min** Potential at the minimum of the ground state potential  $V_g(R_0^g)$ , in a.u.

**Vd\_min** Potential at the minimum of the decaying state potential  $V_d(R_0^d)$ , in a.u.

**Vf\_min** Potential at the minimum of the final state potential  $V_f(R_0^f)$ , in a.u.

**Vd\_vert** Potential at the vertical transition point in the decaying potential  $V_d(R_0^g)$

All potential files must be given in a.u., and must contain the first line as a coment line starting with #. Also there must be only one empty line at the end of the file (empty in the sense that you just broke the line above, but no empty spaces). This is due to the way eSPec expects the potential files to be formatted.

**absorb\_cond** reads the strenght of the damping applyied to the wavefunction. (there is no default, if the keyword is not given, then no absorbing conditions will be applied)

**absorb\_cond** reads the range beyond which the absorbing conditions will be applied (a.u.). For instance, absorb\_cond 0.000 10.000 on a .1D calculation, applies the damping function from 10.000 a.u. until the end of the grid. For .2D/.2DCT calculations four values are expected, naturally.

When choosing absorbing conditions, some trials might be required, since large values for **absorb\_cond** and short ranges might still result in unphysical reflections. Usually a value around  $10^{-4}$  and a range of 1-2 a.u. may give good

results, note however that the kinetic energy of the wavefunction also plays a role, since for large energies we may use a stronger damping, whereas for small energies such strong damping would result in reflections. In sum, some tweaking is advised, but usually it is fairly easy to find working conditions for parameters.

## V. SPECIFIC CASE OF OUR 2D+1D MODEL

Here we take into account the bending modes, which, as we shall see, also can be obtained using the shift of the Fourier transform. In the same fashion as in the second section of this file, we will have

$$\Phi_{vc}(0, \omega - \omega_{c0} + E_0 - \epsilon_{vc} + \epsilon_0) = \int_0^\infty e^{-\Gamma t} e^{-i(\omega_{c0} - E_0 + \epsilon_{vc} - \epsilon_0)t} \phi_c(t) e^{i\omega t} dt, \quad (12)$$

so in this case we will have the shifted frequency and properties being

$$\bar{\omega} = \omega - \omega_{c0} + E_0 - \epsilon_{vc} + \epsilon_0 \quad (13)$$

$$\omega_{res} = V_d(R_0^g) - V_g(R_0^g) - E_0 \quad (14)$$

$$\bar{\omega}_{res} = \omega_{res} - \omega_{c0} + E_0 \quad (15)$$

$$\Omega = \omega - \omega_{res} = \bar{\omega} - \bar{\omega}_{res} + \epsilon_{vc} - \epsilon_0 \quad (16)$$

$$\bar{\omega} = \Omega + \bar{\omega}_{res} - \epsilon_{vc} + \epsilon_0 \quad (17)$$

so in the program we use eq (17) to obtain the functions associated with each bending state, using the shift keyword with the value  $-\epsilon_{vc} + \epsilon_0$ .

## VI. run\_raman\_water.sh SCRIPT

in addition to the flags contained in the general raman script, we now have the following additional ones

**-fc** in this mode, we run the eSPec to obtain the bending mode franck-condon factors, which are saved into files `fc_0vc.dat` and `fc_vcvf.dat` for later use.

**-cross** in this mode, the final program fcorrel is run, which puts together all the previous steps into the final cross section(see eq. (51) from WP\_water.pdf file).

## VII. SPECIFIC INPUT KEYWORDS

In addition to the previous general keywords, we need a few extra ones in the case of our 2D+1D model

```
nvc 3
nvf 5
```

```
bend_npoints 256
bend_mass 12.00
```

```
bend_init_pot bend.init
bend_decay_pot bend.decay
bend_fin_pot bend.final
```

**nvc** number of core-excited bending states to be included.  
**nvf** number of final bending states to be included.  
**bend\_npoints** number of discretization points in the bending potential.  
**bend\_mass** mass in AMU, associated with the bending mode.  
**bend\_init\_pot** name of the file with the initial bending potential.

**bend\_decay\_pot** name of the file with the intermediate beding potential.

**bend\_final\_pot** name of the file with the final bending potential.

## VIII. COMPILING THE PROGRAMS

The raman and fcorrel programs can be downloaded from <https://github.com/LTCC-UFG/eSPec-RAMAN>.

Compiling is very simple, all three programs have their own makefile, so for eSPec, you should just run

```
$ make espec
```

eSPec does not require any additional libraries, as they are included in the code.

For raman and fcorrel, you should just run make, with no specific targets.

```
$ make
```

For these programs, however you should have installed the fftw library (available at <http://www.fftw.org/>).

On a final note, we suggest that you compile the programs with ifort (intel compiler), specially eSPec, since we've had problems compiling it with the latest gfortran.