A Short Tutorial for MBXASPY (Many-Body XAS with Python)

Yufeng Liang

▪ Installation

(1) Shirley_XAS

Shirley_XAS is a software package for simulating x-ray absorption spectra using the excited-state-core-hole (XCH) approach and Shirley's optimal basis functions for band structure interpolation. It is modified based Quantum Espresso simulation package and maintained by David Prendergast's group at the Molecular Foundry. The Shirley_XAS provides the Kohn-Sham eigen-energies and wavefunctions as the input for the determinant formalism of x-ray spectra.

Check out the latest commit on the overlap branch (needs):

git checkout overlap
git pull
make shirley

(2) MBXASPY

Checkout the code on github:

git clone https://github.com/yufengliang/mbxaspy ~/mbxaspy

From now on you can follow my updates by using "git pull". You will see many python source code "*.py" are cloned to your specified directory.

Python 2.7+ is required.

▪ Generate the atomic overlap matrix

This is for generating the overlap matrix used in the PAW correction, between the initial and final states:

$$\langle \phi_l | \tilde{\phi}_{l'} \rangle - \langle \phi_l^{\mathrm{PS}} | \tilde{\phi}_{l'}^{\mathrm{PS}} \rangle$$

Let's say you want to produce such an overlap matrix between a ground-state carbon atom and a core-excited one. Here are the essential steps:

(1) Generate a ground-state and a core-excited pseudo for C routinely using executables in uspp-dgp. Let's say they are C.pbe-van.UPF and C.pbe-van-1s1.UPF respectively.

(2) Generate the initial-state single-body matrix elements and name it as: C.pbe-van.pos

(3) In the pseudo output file *_ps.out, we have access to transformation waves in the "**transformed atomic waves - davegp**" block. Copy every line **with digits only** in the block into a file called "valence-gs.dat" for the ground-state, and "valence-x.dat" for the excited state.

(4) Run:

python /your/path/to/mbxaspy/sij.py valence-gs.dat valence-x.dat

(5) Then you'll see a Sij.dat file in plain text, which is the overlap matrix for the transformation waves. Rename it as "C.pbe-van-1s1.sij" and place it in the XCH pseudo library so that mbxaspy can find it when the pseudo "C.pbe-van-1s1.UPF" is provided.

(6) For sanity checks of the transformation waves, they are plot out to "**wave_gs.png**" and "**wave_x.png**". Use "eog file_name" to visualize them if on a cluster.

(7) If you run sij.py for the same atom, then you will see Sij.dat contains essentially the same values as Q_int in the atom's UPF.

▪ How to run mbxaspy

Part 1 Extend the previous shirley_xas calculations

(1) Run xas.sh, ref.sh, and ana.sh routinely. You can use the shirley_xas on the master branch without the many-body XAS development. The new many-body code is compatible with older calculations.

(2) On the overlap branch, a new script XAS_mbxas.sh is added. Cook up a script like this to run it:

```
. $SLURM_SUBMIT_DIR/Input_Block.in
 $SHIRLEY_ROOT/scripts/arvid/XAS_mbxas.sh
```

If the previous xas step has finished, this calculation is not expensive anyway and takes similar time as shirley_xas.x. So you may submit it to the debug queue in most cases when all Shirley interpolations are done.

This step generates:

*.eigval	eigenvalue file
*.eigvec	eigenvector file < B_i | nk > (transposed)
*.proj	projectors <beta | nk >
*.xmat	single-particle matrix elements <nk | r |phi_c>
overlap.dat	the overlap matrix < B_i | ~B_j >

This calculation will be done automatically for all specified excited atoms in the system and the ground state. These files will appear in the working directories for the excited atoms. overlap.dat is also produced for GS-to-GS transformation and ideally it should be an identity matrix.

You can even run mbxas.sh without any one of xas.sh, ref.sh, and ana.sh but you won't have access to *.xmat for all excited atoms (final states) which are not needed for many-body XAS, and you don't have access to energy shifts and fermi levels.

To automatically schedule all these calculations, you can alternatively checkout the xas_script on github:

git clone https://github.com/yufengliang/xas_script ~/xas_script

and run:

~/xas_script/setup_mbxas.sh	(on NERSC Edison)
~/xas_script/setup_mbxas_cori.sh	(on NERSC Cori)

to produce all 5 scripts: xas.sh, ref.sh, ana.sh, state.sh, and mbxas.sh.

Part 2 mbxaspy

(1) Cook up an input file (normally called mbxapy.in) like this:

```
# initial(ground)-state
path_i        =      'XAS/tio2/GS'  # path
mol_name_i    =      'tio2'               # file prefix
#nbnd_i       =                           # number of initial orbitals (not used now)

# final-state
path_f        =      'XAS/tio2/O5'    # path
#path_f       =      'XAS/tio2/GS'    # path
mol_name_f    =      'tio2.O05-FCH'             # file prefix
#mol_name_f   =      'tio2'               # file prefix
#nbnd_f       =                           # number of final orbitals (not used now)
```

```
xas_arg        =      5                    # number of k points along one direction
gamma_only     = True                      # Using gamma point only
nproc_per_pool =      2                    # number of procs used to process one (spin, k)
final_1p       = True                      # Need one-body final-state spectrum
xi_analysis    = True                      # print out an analysis of the xi matrix
#do_paw_correction      = False             # do PAW corrections or not
spec0_only = False   # only want one-body spectra

# spectral
maxfn = 1          # calculate up to f^(maxfn) order
I_thr = 1e-3       # throw away transitions that are below I_thr (fractional) of the
strongest one.

ELOW           = -5
EHIGH          = 35
SIGMA          = 0.6
NENER          = 1000
ESHIFT_FINAL   = 517.976                   # *** Please note that this is the final ESHIFT
```

(2) Running mbxaspy

Command line:
Non-anaconda python:
python /your/path/to/mbxaspy/main.py < mbxaspy.in > mbxaspy.out &

Anaconda distribution of python (as on NERSC):
python /your/path/to/mbxaspy/main.py mbxaspy.in > mbxaspy.out &

MPI environment (Anaconda)
srun –n #proc python /your/path/to/mbxaspy/main.py mbxaspy.in > mbxaspy.out

On NERSC, be sure to unload python 2.7 and load 3.5-anaconda, which is more stable:

module load python/3.5-anaconda

For the f^(1) term for one (spin, k) tuple, the mbxaspy calculation is actually quite fast and you can run it on command line. It will take couples of minutes at most if the system contains < 100 atoms.

Multiple (spin, k) are supported and highly parallelizable but only individual spectra are printed out at the end because we still don't know how to combine them for multiple k-points.

Only one final-state is handled for each mbxaspy calculation. We can do more using scripts in future.

(3) Check mbxaspy.out for progress and error messages

(4) If the code runs successfully, in the end you will have for each (spin,k) tuple:

spec0_i.dat one-body initial-state spectrum

spec0_f.dat one-body final-state spectrum, if xas.sh is done and final-state xmat has been output.

spec_xas.dat many-body XAS spectrum up to f^(maxfn) order

spec_xps.dat many-body XPS spectrum up to f^(maxfn – 1) order

Columns in the spec files are: energy axis, total, x, y, z

(5) To check if the PAW corrections are done correctly, check test_xi_eig.png or test_xi_eig.dat, which contains all the eigenvalues of \xi and they should not be larger than 1.0.

■