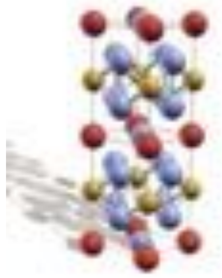
EMBEDDED DYNAMICAL MEAN FIELD THEORY FUNCTIONAL (EDMFTF)

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Kristjan Haule

G. Lucian Pascut

Heung-sik Kim

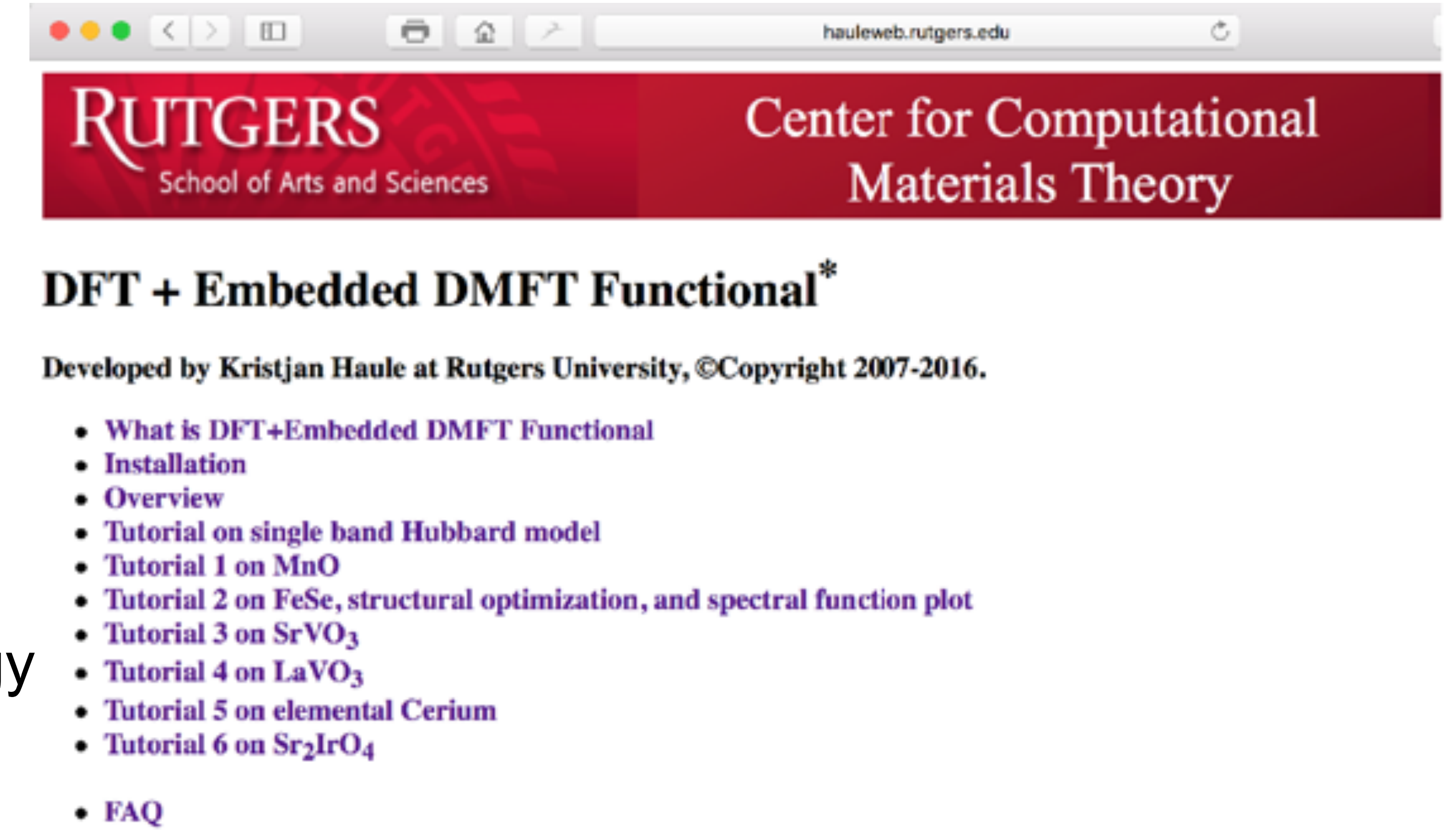


Electronic structure package: EDMFTF

Can be downloaded: http://hauleweb.rutgers.edu/tutorials

tutorials available

- Projection & Embedding instead of downloading in the full potential (APW+lo,LAPW) basis.
- Continuous time quantum Monte Carlo, OCA, NCA…
- Stationary implementation of free energy
- Forces on all atoms
- Structural optimization within DFT+DMFT


hauleweb.rutgers.edu
RUTGERS
School of Arts and Sciences
Center for Computational Materials Theory

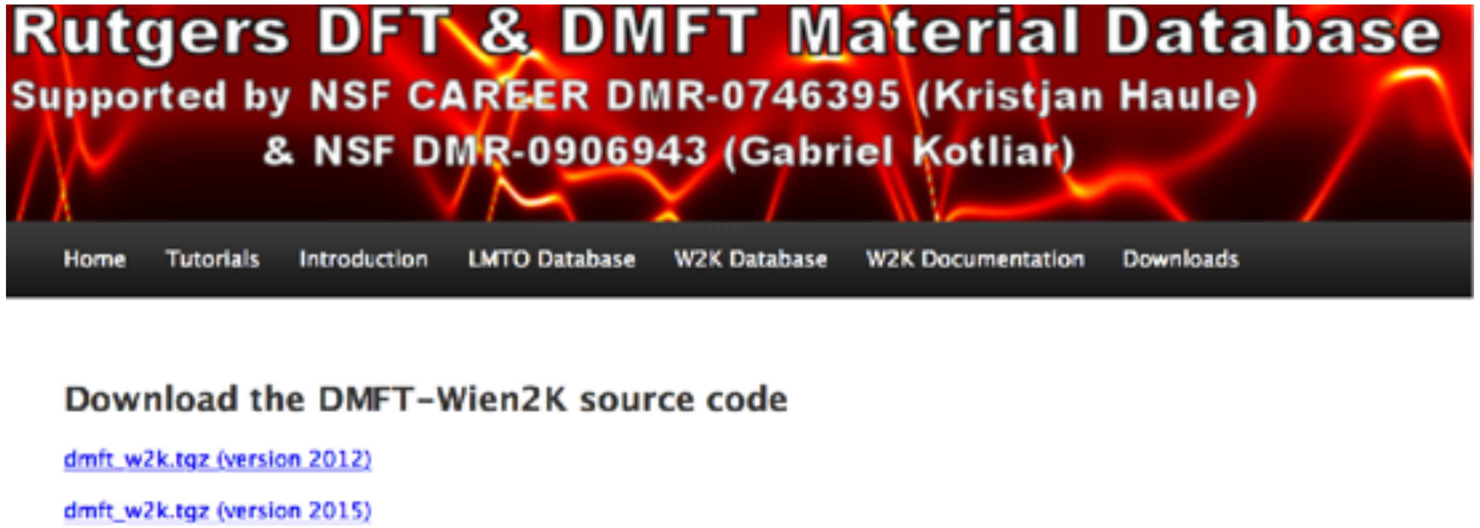
DFT + Embedded DMFT Functional*

Developed by Kristjan Haule at Rutgers University, ©Copyright 2007-2016.

- What is DFT+Embedded DMFT Functional
- Installation
- Overview
- Tutorial on single band Hubbard model
- Tutorial 1 on MnO
- Tutorial 2 on FeSe, structural optimization, and spectral function plot
- Tutorial 3 on $SrVO_3$
- Tutorial 4 on $LaVO_3$
- Tutorial 5 on elemental Cerium
- Tutorial 6 on Sr_2IrO_4
- FAQ

Database:

http://hauleweb.rutgers.edu/



Rutgers DFT & DMFT Material Database

Supported by NSF CAREER DMR-0746395 (Kristjan Haule) & NSF DMR-0906943 (Gabriel Kotliar)

Download the DMFT-Wien2K source code

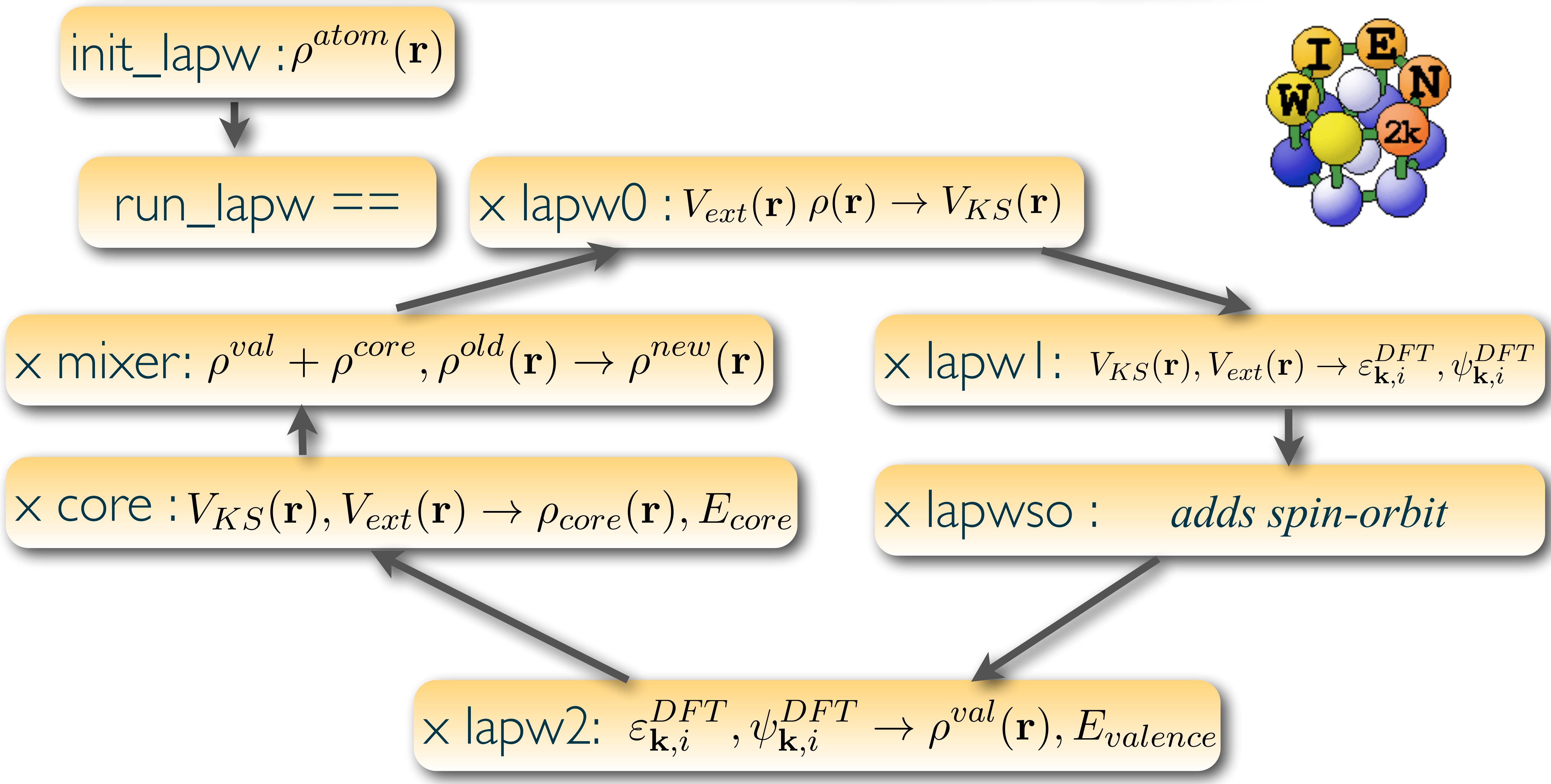
dmft_w2k.tgz (version 2012)

dmft_w2k.tgz (version 2015)

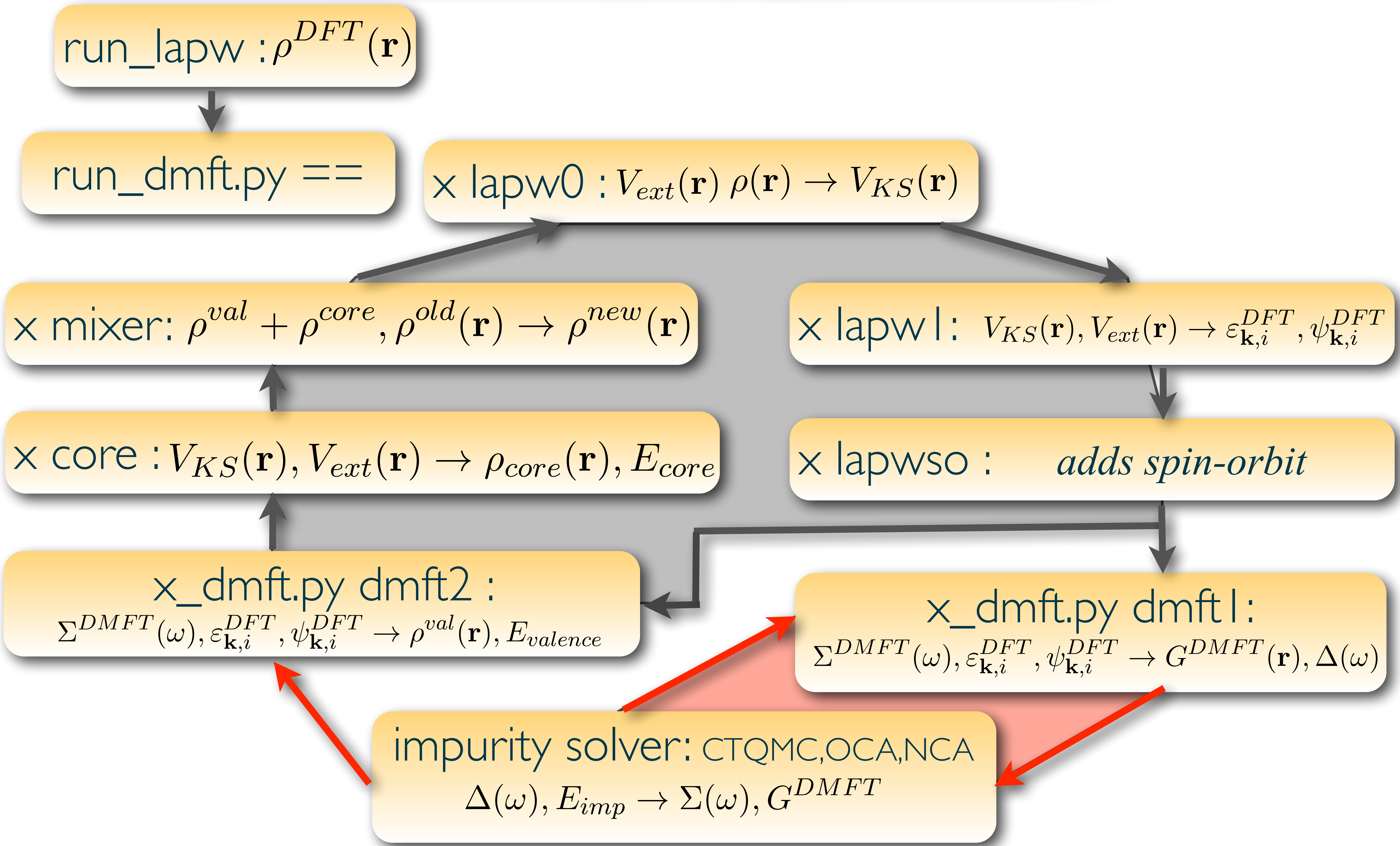
Electronic structure package: EDMFTF

- Package needs wien2k for part of the DFT calculation.
- Composed of many Python, C++, Fortran90 executables linked by Python scripts.
- Two main Python executables:
  - *init_dmft.py* (initialization)
  - *run_dmft.py* (DFT+DMFT scheduler)
- Several post processing tools, such as:
  - *maxent_run.py* *(analytic continuation)*
  - *x_dmft.py dmft1* (calculates DOS)
  - x_dmft.py dmft2 (recalculates electronic charge)
  - *x_dmft.py dmftp* (calculates A(k,w))
  - *wakplot.py* (displays A(k,w))
  - *dmftopt* (calculates optical conductivity or transport)
  - ……

EXECUTION FLOW : DFT PART


init_lapw : $\rho^{atom}(\mathbf{r})$
run_lapw ==
x lapw0 : $V_{ext}(\mathbf{r})\ \rho(\mathbf{r}) \rightarrow V_{KS}(\mathbf{r})$
x lapw1: $V_{KS}(\mathbf{r}), V_{ext}(\mathbf{r}) \rightarrow \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i}$
x lapwso : *adds spin-orbit*
x lapw2: $\varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i} \rightarrow \rho^{val}(\mathbf{r}), E_{valence}$
x core : $V_{KS}(\mathbf{r}), V_{ext}(\mathbf{r}) \rightarrow \rho_{core}(\mathbf{r}), E_{core}$
x mixer: $\rho^{val} + \rho^{core}, \rho^{old}(\mathbf{r}) \rightarrow \rho^{new}(\mathbf{r})$
WIEN 2k

FLOW: DFT+DMFT COMBINED


run_lapw : $\rho^{DFT}(\mathbf{r})$
run_dmft.py ==
x lapw0 : $V_{ext}(\mathbf{r})\ \rho(\mathbf{r}) \rightarrow V_{KS}(\mathbf{r})$
x lapw1: $V_{KS}(\mathbf{r}), V_{ext}(\mathbf{r}) \rightarrow \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i}$
x lapwso : *adds spin-orbit*
x_dmft.py dmft1: $\Sigma^{DMFT}(\omega), \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i} \rightarrow G^{DMFT}(\mathbf{r}), \Delta(\omega)$
impurity solver: CTQMC,OCA,NCA $\Delta(\omega), E_{imp} \rightarrow \Sigma(\omega), G^{DMFT}$
x_dmft.py dmft2 : $\Sigma^{DMFT}(\omega), \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i} \rightarrow \rho^{val}(\mathbf{r}), E_{valence}$
x core : $V_{KS}(\mathbf{r}), V_{ext}(\mathbf{r}) \rightarrow \rho_{core}(\mathbf{r}), E_{core}$
x mixer: $\rho^{val} + \rho^{core}, \rho^{old}(\mathbf{r}) \rightarrow \rho^{new}(\mathbf{r})$

x_dmft.py dmft1:

$$\Sigma^{DMFT}(\omega), \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i} \to G^{DMFT}(\mathbf{r}), \Delta(\omega)$$

input **output**

$$\Sigma(\omega), \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i}(\mathbf{r}) \longrightarrow G_{local}(\omega), \Delta(\omega), E_{imp}, n_{local}$$

1) Construct projector: $P(\mathbf{r}\mathbf{r}', \mathbf{R}_\mu, mm') = \langle \mathbf{r}|\phi^\mu_m\rangle \langle \phi^\mu_{m'}|\mathbf{r}'\rangle$

where $\langle \mathbf{r}|\phi^\mu_m\rangle = u_l(|\mathbf{r}-\mathbf{R}_\mu|)Y_{lm}(\widehat{\mathbf{r}-\mathbf{R}_\mu})$

2) Embed self-energy: $\bar{\Sigma}_{ij}(\mathbf{k},\omega) = \sum_{\mathbf{R}_\mu} \langle \psi^{DFT}_{\mathbf{k},i}|\phi^\mu_m\rangle (\Sigma^\mu_{mm'}(\omega) - V^\mu_{DC}) \langle \phi^\mu_{m'}|\psi^{DFT}_{\mathbf{k},j}\rangle$

3) Calculate local Green's function, hybridization, imp. levels:

$$G^\mu_{local\ mm'} = \sum_{\mathbf{k},ij} \langle \phi^\mu_m|\psi^{DFT}_{\mathbf{k},i}\rangle \left(\omega + \mu - \varepsilon_{\mathbf{k}} - \bar{\Sigma}(\mathbf{k},\omega)\right)^{-1}_{ij} \langle \psi^{DFT}_{\mathbf{k},j}|\phi^\mu_{m'}\rangle = \left(\frac{1}{\omega - E^\mu_{imp} - \Sigma^\mu(\omega) - \Delta^\mu(\omega)}\right)_{mm'}$$

symmetrization over all group operations is performed

x_dmft.py dmft2 :

$$\Sigma^{DMFT}(\omega), \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i} \rightarrow \rho^{val}(\mathbf{r}), E_{valence}$$

input **output**

$$\Sigma(\omega), \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i}(\mathbf{r}) \longrightarrow \rho^{DMFT}_{val}(\mathbf{r}), E_{valence}, F_{valence}, \mathbf{F}^{\mathbf{R}_\mu}$$

1) Construct projector: $P(\mathbf{r}\mathbf{r}', \mathbf{R}_\mu, mm') = \langle \mathbf{r}|\phi^\mu_m\rangle \langle \phi^\mu_{m'}|\mathbf{r}'\rangle$

where $\langle \mathbf{r}|\phi^\mu_m\rangle = u_l(|\mathbf{r}-\mathbf{R}_\mu|) Y_{lm}(\widehat{\mathbf{r}-\mathbf{R}_\mu})$

2) Embed self-energy: $\bar{\Sigma}(\mathbf{r},\mathbf{r}') = \sum_{\mathbf{R}_\mu} \langle \mathbf{r}|\phi^\mu_m\rangle \left(\Sigma^\mu_{mm'}(i\omega) - V^\mu_{DC}\right) \langle \phi^\mu_{m'}|\mathbf{r}'\rangle$

3) Solve the Dyson Eq.: $\left(-\nabla^2 + V_{KS} + \bar{\Sigma}\right) |\psi_{\mathbf{k},i\omega_n,i}\rangle = |\psi_{\mathbf{k},i\omega_n,i}\rangle \, \varepsilon^{DMFT}_{\mathbf{k},i\omega_n,i}$

or $(\varepsilon^{DFT}_{\mathbf{k},i_1}\delta_{i_1,i_2} + \langle \psi^{DFT}_{\mathbf{k},i_1}|\bar{\Sigma}|\psi^{DFT}_{\mathbf{k},i_2}\rangle) \langle \psi^{DFT}_{\mathbf{k},i_2}|\psi_{\mathbf{k},i\omega,i}\rangle = \langle \psi^{DFT}_{\mathbf{k},i_1}|\psi_{\mathbf{k},i\omega_n,i}\rangle \, \varepsilon^{DMFT}_{\mathbf{k},i\omega_n,i}$

4) Determine the chemical potential: $N_{val} = T \sum_{i\omega_n,\mathbf{k},i} \frac{1}{i\omega + \mu - \varepsilon_{\mathbf{k},i\omega_n,i}}$

5) Calculate DMFT electronic charge in space:

$$\rho^{DMFT}_{val}(\mathbf{r}) = \sum_{\mathbf{k},ij} \psi^{DFT}_{\mathbf{k},i}(\mathbf{r}) \, T \sum_{i\omega_n} \left[(i\omega + \mu - \varepsilon^{DFT}_{\mathbf{k}} - \bar{\Sigma}_{\mathbf{k}}(\omega))^{-1}\right]_{ij} \psi^{DFT*}_{\mathbf{k},j}(\mathbf{r})$$

6) Calculate DMFT free energy and forces on all atoms

symmetrization over group operations not performed

EDMFTF initialization: Exercise MnO

Note: In Wien2k directory name has to be equal to struct file-name ("case"). In EDMFTF this is not needed anymore, hence we can create directory with any name.

Assume Wien2k run is in directory "case" = MnO.

We will initialize EDMFTF in the same directory, because we will need energy files (for finding energy of Kohn-Sham bands).

(
```
> srun -n2 --pty bash     # start interactive shell on compute node
```
)

```
> module load edmftf
```

(loads executables, paths, etc)

Two ways to initialize EDMFTF:

1) execute init_dmft.py and answer the questions.
2) append all options in the command line as arguments to the init_dmft.py script.

We will use method 2.

```
> cd MnO
```

directory with Wien2k DFT run.

```
> init_dmft.py  -ca 1 -ot d -qs 7
```

EDMFTF initialization

What are options "-ca 1 -ot d -qs 7"?

Each option answers one of the questions in the initialization.
Options not specified use default values.

correlated atoms:
1 in MnO.struct is
treated by DMFT

orbital type:
The "d" orbital is
correlated by DMFT

"d" orbitals have cubic symmetry and
we will use real harmonics (to avoid
QMC sign problem)

All available options for orbital symmetry

```
  Qsplit  Description
------  ------------------------------------------------------------
     0  average GF, non-correlated
     1  |j,mj> basis, no symmetry, except time reversal (-jz=jz)
    -1  |j,mj> basis, no symmetry, not even time reversal (-jz=jz)
     2  real harmonics basis, no symmetry, except spin (up=dn)
    -2  real harmonics basis, no symmetry, not even spin (up=dn)
     3  t2g orbitals
    -3  eg orbitals
     4  |j,mj>, only l-1/2 and l+1/2
     5  axial symmetry in real harmonics
     6  hexagonal symmetry in real harmonics
     7  cubic symmetry in real harmonics
     8  axial symmetry in real harmonics, up different than down
     9  hexagonal symmetry in real harmonics, up different than down
    10  cubic symmetry in real harmonics, up different then down
    11  |j,mj> basis, non-zero off diagonal elements
    12  real harmonics, non-zero off diagonal elements
    13  J_eff=1/2 basis for 5d ions, non-magnetic with symmetry
    14  J_eff=1/2 basis for 5d ions, no symmetry
------  ------------------------------------------------------------
```

For more options:

```
> init_dmft.py  -h
```

```
There are 2 atoms in the unit cell:
  1 Mn
  2 O
You have chosen the following atoms to be correlated:
  1 Mn
```

Atom 1 in MnO.struct is treated by DMFT

```
For each atom, specify correlated orbital(s) (ex: d,f):
You have chosen to apply correlations to the following orbitals:
  1  Mn-1 d
```

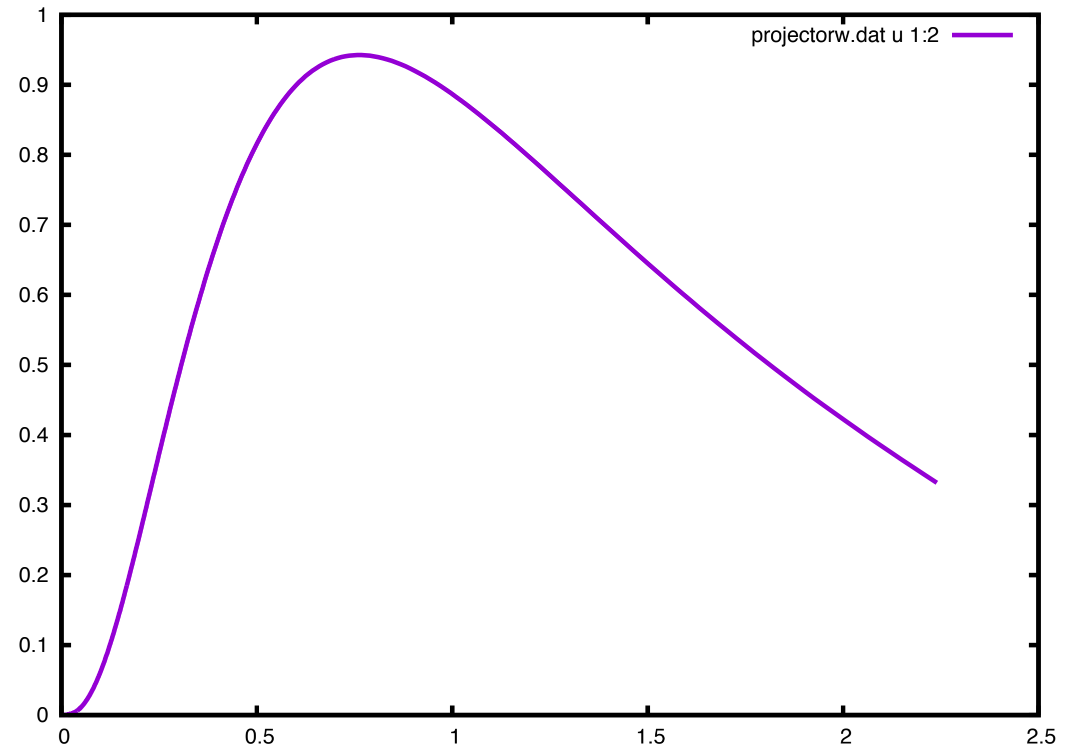
The “d” orbital is correlated by DMFT

```
Specify qsplit for each correlated orbital (default = 0):
    Qsplit  Description
------  ------------------------------------------------------------
     0  average GF, non-correlated
     1  |j,mj> basis, no symmetry, except time reversal (-jz=jz)
    -1  |j,mj> basis, no symmetry, not even time reversal (-jz=jz)
     2  real harmonics basis, no symmetry, except spin (up=dn)
    -2  real harmonics basis, no symmetry, not even spin (up=dn)
     3  t2g orbitals
    -3  eg orbitals
     4  |j,mj>, only l-1/2 and l+1/2
     5  axial symmetry in real harmonics
     6  hexagonal symmetry in real harmonics
     7  cubic symmetry in real harmonics
     8  axial symmetry in real harmonics, up different than down
     9  hexagonal symmetry in real harmonics, up different than down
    10  cubic symmetry in real harmonics, up different then down
    11  |j,mj> basis, non-zero off diagonal elements
    12  real harmonics, non-zero off diagonal elements
    13  J_eff=1/2 basis for 5d ions, non-magnetic with symmetry
    14  J_eff=1/2 basis for 5d ions, no symmetry
------  ------------------------------------------------------------
You have chosen the following qsplits:
  1  Mn-1 d: 7
```

“d” orbitals have cubic symmetry and we will use real harmonics (to avoid QMC sign problem)

```
Specify projector type (default = 5):
  Projector  Description
------  --------------------------------------------------------
     1  projection to the solution of Dirac equation (to the head)
     2  projection to the Dirac solution, its energy derivative,
          LO orbital, as described by P2 in PRB 81, 195107 (2010)
     4  similar to projector-2, but takes fixed number of bands in
          some energy range, even when chemical potential and
          MT-zero moves (follows band with certain index)
     5  fixed projector, which is written to projectorw.dat. You can
        generate projectorw.dat with the tool wavef.py
------  --------------------------------------------------------
>  5
```

Mn 3d orbital



Projector fixed to achieve stationarity of the functional.
Radial dependence of projector stored in *projectorw.dat*

```
Range (in eV) of hybridization taken into account in impurity
problems; default -10.0, 10.0:
-10.0,10.0
```

Very large 20eV window of bands is used to construct DMFT projector.
Rule of thumb : should be more than [-U,U].

```
Perform calculation on real; or imaginary axis? (i/r): (default=i)
i
```

QMC works on the imaginary axis.

EDMFTF initialization

Which files were created by this initialization?

- projectorw.dat (contains the radial dependence of the orbital)
- MnO.indmfl (connects DMFT orbitals with the Kohn-Sham states)
- MnO.indmfi (connects DMFT orbitals with the impurity solver)

.indmfl file

connects DMFT orbitals with the Kohn-Sham states

first and last band used in the projector

projector type

Green's function calculated on imaginary axis

```
5 15 1 5                        # hybridization band index nemin and nemax, renormalize for interstitials, projection type
1 0.025 0.025 200 -3.000000 1.000000  # matsubara, broadening-corr, broadening-noncorr, nomega, omega_min, omega_max (in eV)
1                               # number of correlated atoms
1     1   0                      # iatom, nL, locrot
  2   7   1                      # L, qsplit, cix
#================ # Siginds and crystal-field transformations for correlated orbitals ================
1     5   2      # Number of independent kcix blocks, max dimension, max num-independent-components
1     5   2      # cix-num, dimension, num-independent-components
#---------------- # Independent components are --------------
'eg' 't2g'
#---------------- # Sigind follows --------------------------
1 0 0 0 0
0 1 0 0 0
0 0 2 0 0
0 0 0 2 0
0 0 0 0 2
#---------------- # Transformation matrix follows -----------
 0.00000000  0.00000000    0.00000000  0.00000000    1.00000000  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000
 0.70710679  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000    0.70710679  0.00000000
 0.00000000  0.00000000    0.70710679  0.00000000    0.00000000  0.00000000   -0.70710679  0.00000000    0.00000000  0.00000000
 0.00000000  0.00000000    0.00000000  0.70710679    0.00000000  0.00000000    0.00000000  0.70710679    0.00000000  0.00000000
-0.00000000 -0.70710679    0.00000000  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000    0.00000000  0.70710679
```

Each correlated orbital has specification (which atom from structure, which L, *locrot* =for rotations, qsplit=which symmetry, cix=successive number of correlated set)

.indmfl file

```
5 15 1 5                       # hybridization band index nemin and nemax, renormalize for interstitials, projection type
1 0.025 0.025 200 -3.000000 1.000000  # matsubara, broadening-corr, broadening-noncorr, nomega, omega_min, omega_max (in eV)
1                             # number of correlated atoms
1    1   0                     # iatom, nL, locrot
  2   7   1                    # L, qsplit, cix
#================ # Siginds and crystal-field transformations for correlated orbitals ================
1     5   2       # Number of independent kcix blocks, max dimension, max num-independent-components
1     5   2       # cix-num, dimension, num-independent-components
#--------------- # Independent components are --------------
'eg' 't2g'
#--------------- # Sigind follows --------------------------
1 0 0 0 0
0 1 0 0 0
0 0 2 0 0
0 0 0 2 0
0 0 0 0 2
#--------------- # Transformation matrix follows -----------
 0.00000000  0.00000000    0.00000000  0.00000000    1.00000000  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000
 0.70710679  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000    0.70710679  0.00000000
 0.00000000  0.00000000    0.70710679  0.00000000    0.00000000  0.00000000   -0.70710679  0.00000000    0.00000000  0.00000000
 0.00000000  0.00000000    0.00000000  0.70710679    0.00000000  0.00000000    0.00000000  0.70710679    0.00000000  0.00000000
-0.00000000 -0.70710679    0.00000000  0.00000000    0.00000000  0.00000000    0.00000000  0.00000000    0.00000000  0.70710679
```

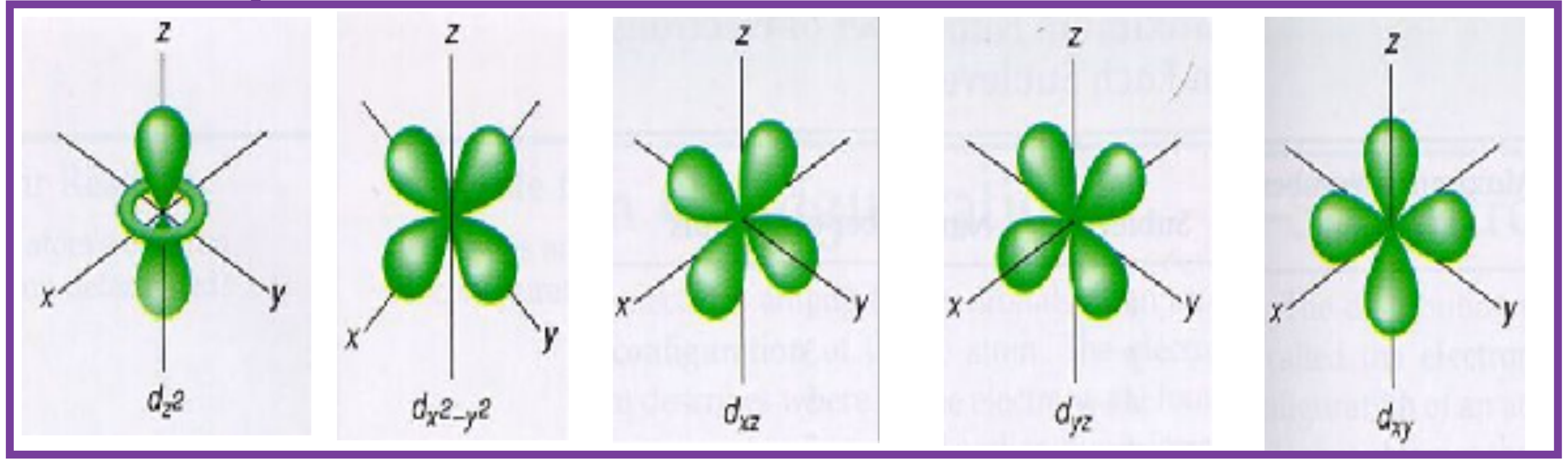
correlated set (cix) if of dimension 5 and has two self-energies, i.e., eg and t2g

How is self-energy matrix constructed from the two self-energy components

z^2
x^2-y^2
xz
yz
xy

L=2,m=-2 L=2,m=-1 L=2,m=0 L=2,m=1 L=2,m=2

Transformation matrix from spheric harmonics to the orbitals used in the DMFT.



.indmfi file

connects DMFT orbitals with the impurity solver

Only one impurity problem needs to be calculated

5 orbital problem

```
1   # number of sigind blocks
5   # dimension of this sigind block
1 0 0 0 0
0 1 0 0 0
0 0 2 0 0
0 0 0 2 0
0 0 0 0 2
```

symmetry of orbitals in the impurity problem

EDMFTF initialization

Need two more files to start EDMFTF calculation:

1) params.dat (files with parameters — se below)

params.dat is a python file, which can use python expressions.
Can be checked for syntactic correctness by "*python params.dat*"

```
> cp $RESULTS/MnO/DMFT/params.dat .
```

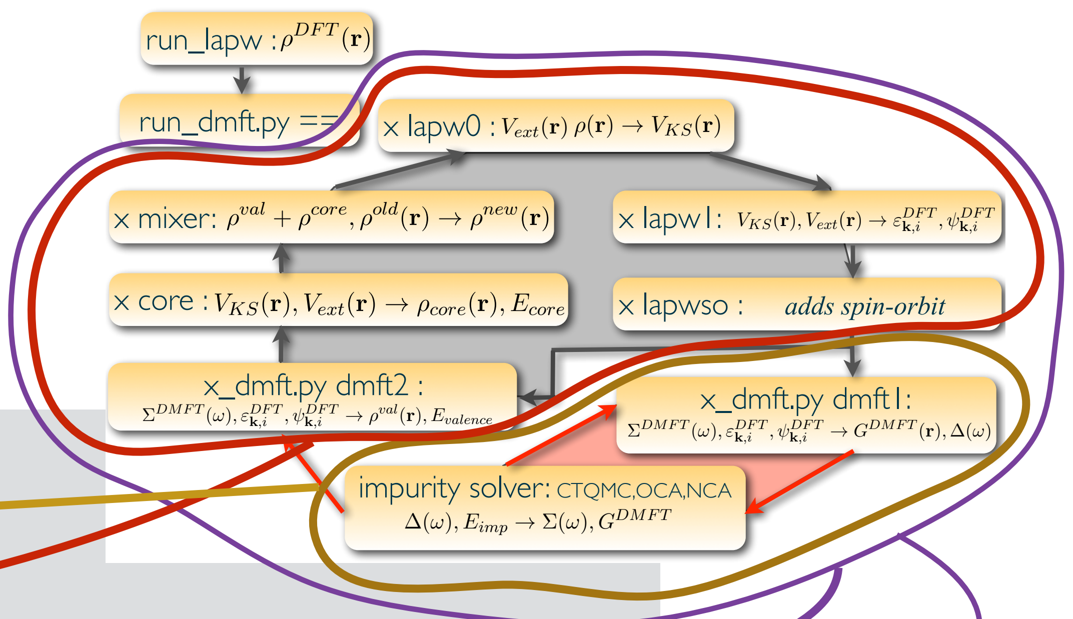
2) sig.inp (guess for initial self-energy)

text file with columns corresponding to real and imaginary part of self-energy.
first column is frequency, following by the real and imaginary part of Sigma for each orbital specified in *case.indmfi*

```
> szero.py
```

creates empty self-energy sig.inp with goods guess for Vdc and sigma(∞)

params.dat



run_lapw : $\rho^{DFT}(\mathbf{r})$
run_dmft.py ==
x lapw0 : $V_{ext}(\mathbf{r})\ \rho(\mathbf{r}) \rightarrow V_{KS}(\mathbf{r})$
x mixer: $\rho^{val} + \rho^{core}, \rho^{old}(\mathbf{r}) \rightarrow \rho^{new}(\mathbf{r})$
x lapw1: $V_{KS}(\mathbf{r}), V_{ext}(\mathbf{r}) \rightarrow \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i}$
x core : $V_{KS}(\mathbf{r}), V_{ext}(\mathbf{r}) \rightarrow \rho_{core}(\mathbf{r}), E_{core}$
x lapwso : *adds spin-orbit*
x_dmft.py dmft2 : $\Sigma^{DMFT}(\omega), \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i} \rightarrow \rho^{val}(\mathbf{r}), E_{valence}$
x_dmft.py dmft1: $\Sigma^{DMFT}(\omega), \varepsilon^{DFT}_{\mathbf{k},i}, \psi^{DFT}_{\mathbf{k},i} \rightarrow G^{DMFT}(\mathbf{r}), \Delta(\omega)$
impurity solver: CTQMC,OCA,NCA $\Delta(\omega), E_{imp} \rightarrow \Sigma(\omega), G^{DMFT}$

```
solver        =  'CTQMC'   # impurity solver
DCs           =  'exactd'    # exact double-counting with dielectric constant approx.

max_dmft_iterations = 1    # number of iteration of the dmft-loop only
max_lda_iterations  = 100  # number of iteration of the LDA-loop only
finish        =  10          # number of iterations of full charge loop (1 = no charge self-consistency)

ntail         =  300      # on imaginary axis, number of points in the tail of the logarithmic mesh

cc            =  5e-5     # the charge density precision to stop the LDA+DMFT run
ec            =  5e-5     # the energy precision to stop the LDA+DMFT run
recomputeEF   =  0        # Recompute EF in dmft2 step. If recomputeEF = 0, it fixed the chemical potential. Good for insulators

# Impurity problem number 0
iparams0={"exe"              : ["ctqmc"       , "# Name of the executable"],
          "U"                : [9.0           , "# Coulomb repulsion (F0)"],
          "J"                : [1.14          , "# Coulomb repulsion (F0)"],
          "CoulombF"         : ["'Ising'"     , "# Form of Coulomb repulsion. 'Full' allows rotational invariant form of C.I."],
          "beta"             : [38.68         , "# Inverse temperature T=116K"],
          "svd_lmax"         : [25            , "# We will use SVD basis to expand G, with this cutoff"],
          "M"                : [5e6           , "# Total number of Monte Carlo steps"],
          "mode"             : ["SH"          , "# We will use self-energy sampling, and Hubbard I tail"],
          "nom"              : [100           , "# Number of Matsubara frequency points sampled"],
          "tsample"          : [30            , "# How often to record measurements"],
          "GlobalFlip"       : [500000        , "# How often to try a global flip”],
          "warmup"           : [1e5           , "# Warmup number of QMC steps"],
          "nf0"              : [5.0           , "# Nominal occupancy nd for double-counting"],
         }
```

To stop the loop need sufficient QMC precision (many cores)

Careful: allowed only for insulators. For metals recomputeEF=1.

CTQMC options

params.dat continue

```
# Impurity problem number 0
iparams0={"exe"              : ["ctqmc"        , "# Name of the executable"],
              "U"            : [9.0            , "# Coulomb repulsion (F0)"],
              "J"            : [1.14           , "# Coulomb repulsion (F0)"],
          "CoulombF"         : ["'Ising'"      , "# Form of Coulomb repulsion. 'Full' allows rotational invariant form of C.I."],
            "beta"           : [38.68          , "# Inverse temperature T=116K"],
          "svd_lmax"         : [25             , "# We will use SVD basis to expand G, with this cutoff"],
              "M"            : [5e6            , "# Total number of Monte Carlo steps"],
            "mode"           : ["SH"           , "# We will use self-energy sampling, and Hubbard I tail"],
            "nom"            : [100            , "# Number of Matsubara frequency points sampled"],
           "tsample"         : [30             , "# How often to record measurements"],
          "GlobalFlip"       : [500000         , "# How often to try a global flip”],
          "warmup"           : [1e5            , "# Warmup number of QMC steps"],
              "nf0"          : [5.0            , "# Nominal occupancy nd for double-counting"],
              }

# Impurity problem number 1
iparams1={
   ……
   …….
}
```

- *U — Hubbard U*
- *J — Hund's interaction*
- *CoulombF — form of the Coulomb interaction : Ising==density-density Slater form, Full==rotationally invariant Slater form.* *See for details :* http://hauleweb.rutgers.edu/tutorials/CoulombUexplain.html

- *beta — 1/T[eV] or 11604/T[K]*
- *M — number of MC-steps on single core. If it runs on more cores, better results & equal run-time.*
- *mode — S==computing self-energy by Schwinger equation (rather than Dyson) H==high frequency computed by Hubbard-I form.*
- *nom — number of Matsubara points keeps in the mesh (the rest in log mesh of ntail~100) (usually 3*beta)*
- *tsample — every 30 MC steps we measure self-energy*
- *GlobalFlip — is attempted every 5e5 steps*
- *warmup — first 1e5 steps are not used to meassure*
- *nf0 — Mn valence, used for double-counting. Here only for the initial guess (from DFT), will use exactDC later.*

Final steps in initialization

Increase number of k-points for more precise DMFT frequency dependence.

```
> x kgen -f MnO
       2000
```

Optional : Rerun LDA with new k-mesh.

```
> run_lapw -NI
```

Optional : Create new directory with only necessary files for DFT+DMFT calculation

```
> mkdir LDA; mv * LDA/
> mkdir DMFT; cd DMFT
> dmft_copy.py ../LDA
```

params.dat	basic parameters	DMFT files, which we already discussed
MnO.indmfl	KS-lattice to DMFT connection	
MnO.indmfi	impurity to DMFT connection	
projectorw.dat	radial dependent projector	
sig.inp	self-energy	
MnO.struct	crystal structure	W2k files
MnO.clmsum	charge density	
MnO.in0	input to lapw0	
MnO.in1	input to lapw1	
MnO.in2	input to lapw2	
MnO.inm	input to mixer	
MnO.klist	list of k-points	
MnO.scf2	input to lapw2/dmft2	

Final steps in initialization

For parallel execution we need to create special file “mpi_prefix.dat”

```
> echo "mpirun -np 2 -x OMP_NUM_THREADS=1" > mpi_prefix.dat
```

- If this file does not exists, the run will be serial.
- np specified number of available cores (unfortunately only 2 can be afforded here)
- OMP_NUM_THREADS=1 switches off open_mp when mpi is used. If a lot of cores are available, one can use combination of open_MP and MPI.
- mpi_prefix.dat is used for the ctqmc impurity solver. If dmft1, dmft2, lapw1, lapwso require different parallelization, one can specify also mpi_prefix.dat2.

It appears that multithreading makes code slower on this machine, so please turn it off.

```
> export OMP_NUM_THREADS=1
```

Finally run EMDFTF code by executing

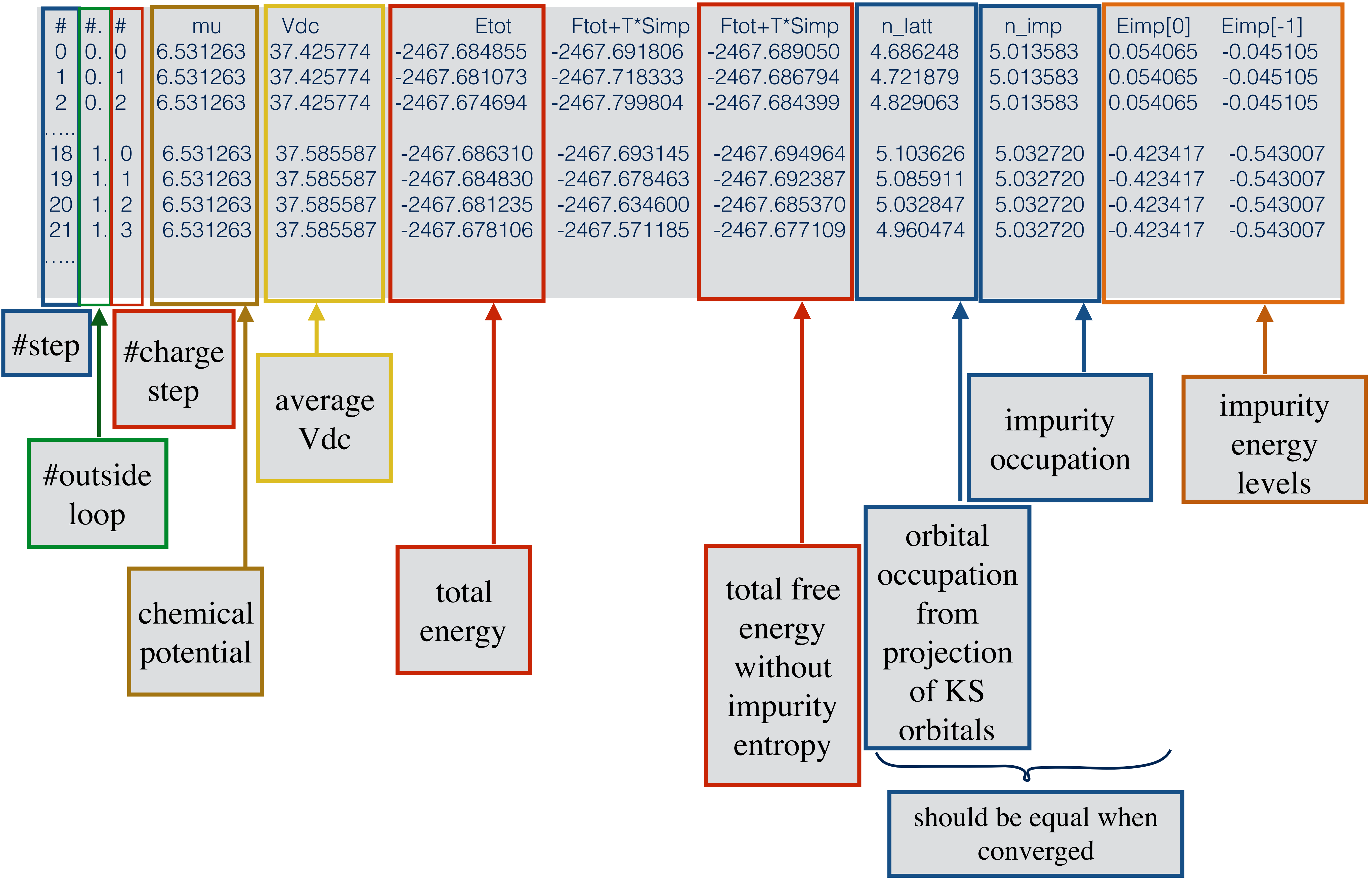
```
> run_dmft.py >& nohup.dat
```

Monitor the run

`> less info.iterate`	check the summary file
`> less  dmft_info.out`	see what/how is executed
`> less ':log'`	check execution log
`> less MnO.dayfile`	execution log+convergence
`> grep ':CHARGE' MnO.dayfile`	charge converged by iterations
`> plot -u1:3,1:5 -x:10  MnO.dlt1`	plot hybridization function
`> plot -x:20 -g -u1:3,1:5 imp.0/Gf.out.?.1`	plot the DMFT output G
`> plot -x:20 -g -u1:3,1:5 imp.0/Sig.out`	plot the impurity output Sigma
`>plot -g -u1:9,1:10 info.iterate`	plot lattice & imp. occupancy

> less info.iterate

summary file



```
#  #. #  mu        Vdc        Etot          Ftot+T*Simp   Ftot+T*Simp   n_latt    n_imp     Eimp[0]    Eimp[-1]
0  0. 0  6.531263  37.425774  -2467.684855  -2467.691806  -2467.689050  4.686248  5.013583  0.054065   -0.045105
1  0. 1  6.531263  37.425774  -2467.681073  -2467.718333  -2467.686794  4.721879  5.013583  0.054065   -0.045105
2  0. 2  6.531263  37.425774  -2467.674694  -2467.799804  -2467.684399  4.829063  5.013583  0.054065   -0.045105
.....
18 1. 0  6.531263  37.585587  -2467.686310  -2467.693145  -2467.694964  5.103626  5.032720  -0.423417  -0.543007
19 1. 1  6.531263  37.585587  -2467.684830  -2467.678463  -2467.692387  5.085911  5.032720  -0.423417  -0.543007
20 1. 2  6.531263  37.585587  -2467.681235  -2467.634600  -2467.685370  5.032847  5.032720  -0.423417  -0.543007
21 1. 3  6.531263  37.585587  -2467.678106  -2467.571185  -2467.677109  4.960474  5.032720  -0.423417  -0.543007
.....
```

#step
#charge step
#outside loop
chemical potential
average Vdc
total energy
total free energy without impurity entropy
orbital occupation from projection of KS orbitals
impurity occupation
impurity energy levels
should be equal when converged

> less dmft_info.out

details of every step and its execution (a lot of information)

```
********------ parameter change ------***********
iparams0 = {'nom': [100, '# Number of Matsubara frequency points sampled'], 'tsample': [30, '# How often to record measurements'], 'Ncout': [500000, '# How often to print out info'], 'J': [1.14, '# Coulomb J'], 'M': [5000000.0, '# Total number of Monte Carlo steps per core'], 'CoulombF': ['"Ising"', "# Density-density form. Can be changed to 'Full' "], 'beta': [38.68, '# Inverse temperature'], 'U': [9.0, '# Coulomb repulsion (F0)'], 'GlobalFlip': [500000, '# How often to try a global flip'], 'sderiv': [0.02, '# Maximum derivative mismatch accepted for tail concatenation'], 'Nmax': [500, '# Maximum perturbation order allowed'], 'exe': ['ctqmc', '# Name of the executable'], 'warmup': [500000.0, '# Warmup number of QMC steps'], 'OCA_G': [False, "# Don't compute OCA diagrams for speed"], 'aom': [3, '# Number of frequency points used to determin the value of sigma at nom']}
.......
DCs = exactd
Znucs= {1: 25}
iSiginds= {0: [[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 2, 0, 0], [0, 0, 0, 2, 0], [0, 0, 0, 0, 2]]}
......
#<lapw0>:  wien2k_14/x -p -f MnO lapw0
#<lapw1>:  x_dmft.py -p lapw1
DmftEnvironment: mpi_prefix.dat exists -- running in parallel mode.
   mpirun -np 2
..... running: mpirun -np 2  ./lapw1 lapw1.def
case=MnO, nom=100, ntail=300, insig=sig.inp, outsig=sig.inp[1-n]
s_oo= [38.22, 38.22]
Edc= [38.22, 38.22]
..... Creating logarithmic mesh
..... Going over all correlated blocks
icix= 1 colsp= [1, 2] colsm= []
icl= 1
icl= 2
Running ---- dmft1 ——
#<dmft1>:  /usr/bin/time  mpirun -np 2 ./dmft dmft1.def >> dmft1_info.out
#<sjoin>:  sjoin.py -m 1.0
Running ----- impurity solver -----
Taking care of high frequency
Eimp= [-38.16593522 -38.26510458]
Edc= [ 38.22  38.22]
ncorr= 5.01358294333
#<sgather>:  sgather.py
------- DMFT part of the step #  0 done ---------
case=MnO, nom=100, ntail=300, insig=sig.inp, outsig=sig.inp[1-n]
s_oo= [38.26987655713145, 38.35976476628431]
Edc= [37.425774429186845, 37.431910457048744]
..... Creating logarithmic mesh
--------- Preparing Charge calculation ---------
#<prep-dmft2>:  x_dmft.py -d --mode c -m 0 -x 1.0 -w 1.0 -p dmft2 >> ksum_info.out 2>&1
#<ssplit>:  ssplit.py -n 100 -t 300
#<dmft2>:  mpirun -np 2 ./dmft2 dmft2.def >> dmft2_info.out
#<lcore>:  x -f MnO lcore
#<mixer>:  x -f MnO mixer
------- LDA(charge) step 0 0 done ---------
#<lapw0>:  x -p -f MnO lapw0
#<lapw1>:  x_dmft.py -p lapw1
..... running: mpirun -np 2  ./lapw1 lapw1.def
--------- Preparing Charge calculation ---------
#<prep-dmft2>:  x_dmft.py -d --mode c -m 0 -x 1.0 -w 1.0 -p dmft2 >> ksum_info.out 2>&1
#<ssplit>:  ssplit.py -n 100 -t 300
#<dmft2>:  mpirun -np 2 ./dmft2 dmft2.def >> dmft2_info.out
#<lcore>:  x -f MnO lcore
#<mixer>:  x -f MnO mixer
------- LDA(charge) step 1 0 done ---------
```

which steps have finished when?

```
> less :log
```

```
Wed May 23 23:56:53 EDT 2018> (x) -f MnO lapw0
Wed May 23 23:56:54 EDT 2018>     lapw1
Wed May 23 23:56:55 EDT 2018>     dmft1
Wed May 23 23:56:56 EDT 2018>     impurity
Wed May 23 23:59:38 EDT 2018>     dmft2
Wed May 23 23:59:39 EDT 2018> (x) -f MnO lcore
Wed May 23 23:59:39 EDT 2018> (x) -f MnO mixer
Wed May 23 23:59:39 EDT 2018> (x) -f MnO lapw0
Wed May 23 23:59:40 EDT 2018>     lapw1
Wed May 23 23:59:41 EDT 2018>     dmft2
Wed May 23 23:59:42 EDT 2018> (x) -f MnO lcore
Wed May 23 23:59:42 EDT 2018> (x) -f MnO mixer
Wed May 23 23:59:42 EDT 2018> (x) -f MnO lapw0
Wed May 23 23:59:43 EDT 2018>     lapw1
Wed May 23 23:59:44 EDT 2018>     dmft2
Wed May 23 23:59:44 EDT 2018> (x) -f MnO lcore
Wed May 23 23:59:44 EDT 2018> (x) -f MnO mixer
Wed May 23 23:59:44 EDT 2018> (x) -f MnO lapw0
Wed May 23 23:59:46 EDT 2018>     lapw1
```

```
> less MnO.dayfile
```

```
   cycle 0      Wed May 23 23:56:53 2018 30/30 to go

>lapw0      ( 23:56:53 )
>lapw1      ( 23:56:54 )
>dmft1      ( 23:56:55 )
>impurity   ( 23:56:56 )
>dmft2      ( 23:59:38 )
>lcore      ( 23:59:39 )
>mixer      ( 23:59:39 )
:ENERGY convergence:  0.00275430000011
:CHARGE convergence:  0.269067
:EF     convergence:  0
```

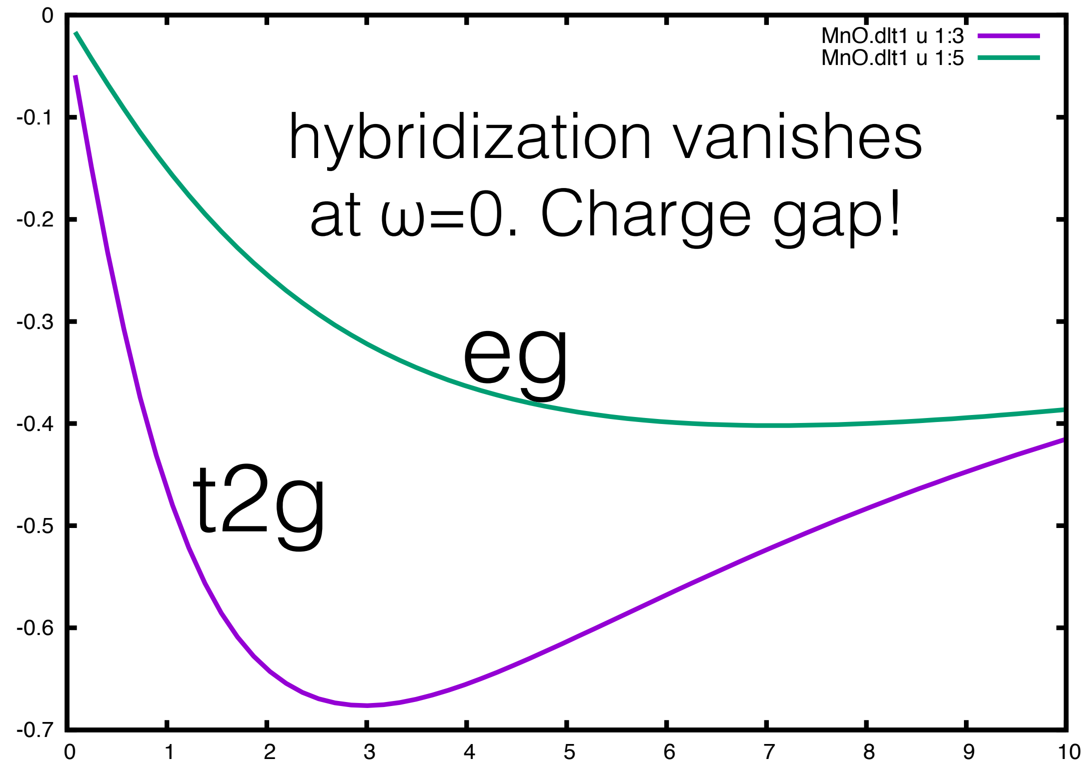
Some output files

Check current impurity hybridization function

```
> plot -u1:3,1:5 -x:10  MnO.dlt1
```

imaginary part of eg

imaginary part of t2g

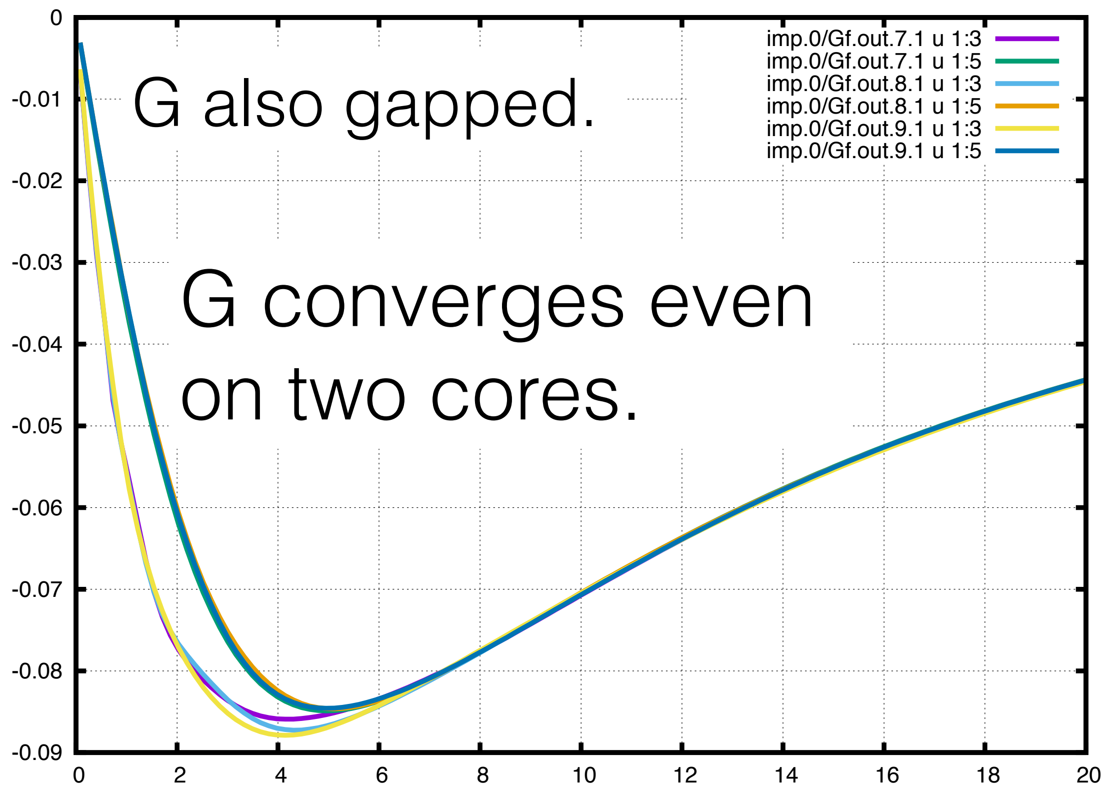


Check impurity Green's function

```
> plot -x:20 -g -u1:3,1:5 imp.0/Gf.out.?.1
```

Check self-energy

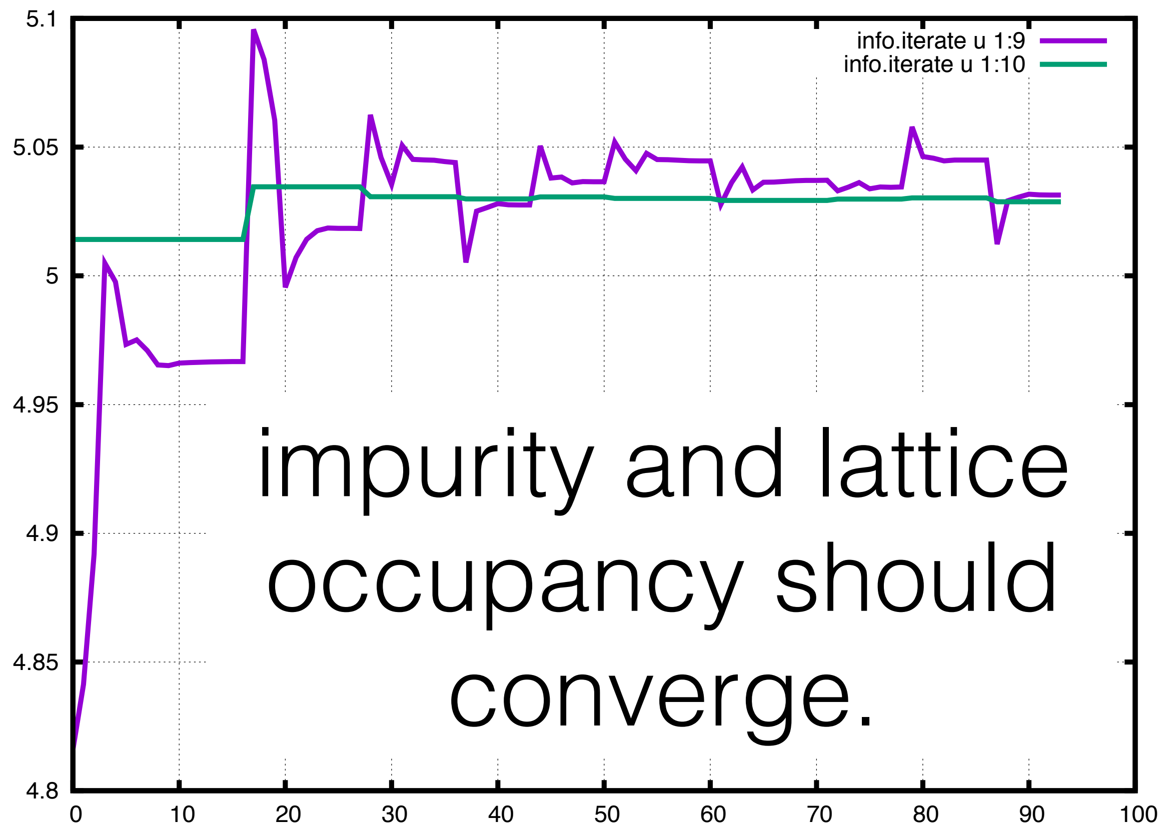
```
> plot -x:20 -g -u1:3,1:5 imp.0/Sig.out.?.1
```



Some output files

Check difference between impurity and lattice occupancy

```
> plot -g -u1:9,1:10 info.iterate
```



impurity and lattice occupancy should converge.

Check convergence of electronic charge

```
> grep ':CHARGE' MnO.dayfile
```

```
:CHARGE convergence:  1.0
:CHARGE convergence:  0.272381
:CHARGE convergence:  0.208401
:CHARGE convergence:  0.094013
:CHARGE convergence:  0.075578
:CHARGE convergence:  0.071506
:CHARGE convergence:  0.014747
:CHARGE convergence:  0.005258
:CHARGE convergence:  0.001651
:CHARGE convergence:  0.00165
:CHARGE convergence:  0.00068
:CHARGE convergence:  0.00045
:CHARGE convergence:  0.000309
:CHARGE convergence:  0.000188
:CHARGE convergence:  0.000108
:CHARGE convergence:  6.2e-05
:CHARGE convergence:  2e-05
:CHARGE convergence:  0.085869
:CHARGE convergence:  0.075714
:CHARGE convergence:  0.055726
:CHARGE convergence:  0.026821
:CHARGE convergence:  0.014785
:CHARGE convergence:  0.010197
:CHARGE convergence:  0.001906
:CHARGE convergence:  0.001064
:CHARGE convergence:  0.000376
:CHARGE convergence:  0.000176
:CHARGE convergence:  2.1e-05
```

Check what impurity solver is doing

```
>less imp.0/nohup_imp.out.000
```

Postprocessing : maxent

```
> mkdir maxent; cd maxent
```

create new directory for analytic continuation

```
> cp ../sig.inp.10.1 .
```

copy the latest self-energy

alternatively, take average over previous run:

```
saverage.py $RESULTS/MnO/sig.inp.1?.1
```

```
> cp $RESULTS/MnO/maxent/maxent_params.dat .
```

need several parameters

```
params={'statistics': 'fermi', # fermi/bose
        'Ntau'      : 300,           # Number of time points
        'L'         : 20.0,          # cutoff frequency on real axis
        'x0'        : 0.005,         # low energy cut-off
        'bwdth'     : 0.004,         # smoothing width
        'Nw'        : 300,           # number of frequency points on real axis
        'gwidth'    : 2*15.0,        # width of gaussian
        'idg'       : 1,             # error scheme: idg=1 -> sigma=deltag ; idg=0 -> sigma=deltag*G(tau)
        'deltag'    : 0.004,         # error
        'Asteps'    : 4000,          # anealing steps
        'alpha0'    : 1000,          # starting alpha
        'min_ratio' : 0.001,         # condition to finish, what should be the ratio
        'iflat'     : 1,             # iflat=0 : constant model, iflat=1 : gaussian of width gwidth, iflat=2 : input using file model.dat
        'Nitt'      : 1000,          # maximum number of outside iterations
        'Nr'        : 0,             # number of smoothing runs
        'Nf'        : 40,            # to perform inverse Fourier, high frequency limit is computed from the last Nf points
        }
```

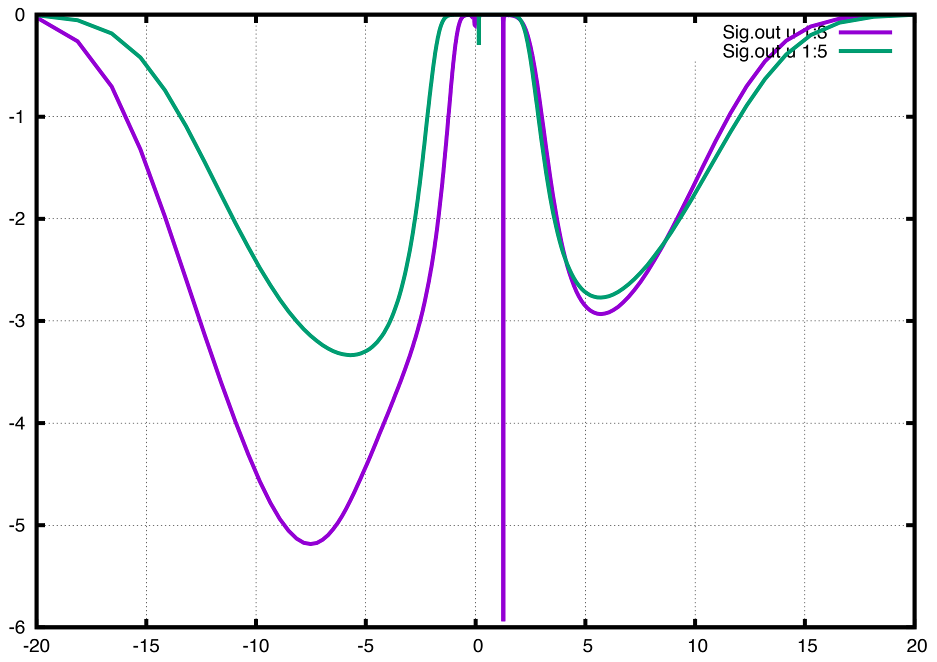
```
> maxent_run.py sig.inp.10.1
```

run maximum entropy

Postprocessing : maxent

Check the resulting self-energy on real axis

```
> plot -g -u1:3,1:5 -x-20:20 Sig.out
```



Mott gap with delta-function in the gap

Postprocessing : Density of states

```
> mkdir ../onreal; cd ../onreal
```

Create another directory for real axis DOS

```
> dmft_copy.py  ../
```

copy converged results on imaginary axis

If you do not have your own results, use:

```
dmft_copy.py  $RESULTS/MnO
```

```
>cp ../maxent/Sig.out sig.inp
```

copy maxent real-axis self-energy to this directory

*edit the second line of **MnO.indmfl** file and change the flag **matsubara** to 0*

```
5 15 1 5                        # hybridization band index nemin and nemax, renormalize for interstitials, projection type
1 0.025 0.025 200 -3.000000 1.000000  # matsubara, broadening-corr, broadening-noncorr, nomega, omega_min,
omega_max (in eV)
1                            # number of correlated atoms
1     1   0                    # iatom, nL, locrot
  2   7   1                    # L, qsplit, cix
```

changed file:

```
5 15 1 5                        # hybridization band index nemin and nemax, renormalize for interstitials, projection type
0 0.025 0.025 200 -3.000000 1.000000  # matsubara, broadening-corr, broadening-noncorr, nomega, omega_min,
omega_max (in eV)
1                            # number of correlated atoms
1     1   0                    # iatom, nL, locrot
  2   7   1                    # L, qsplit, cix
```

Postprocessing : Density of states

```
> x lapw0 -f MnO
```

Recompute potential (lapw0)

```
> x_dmft.py lapw1
```

Recompute KS bands

```
> x_dmft.py dmft1
```

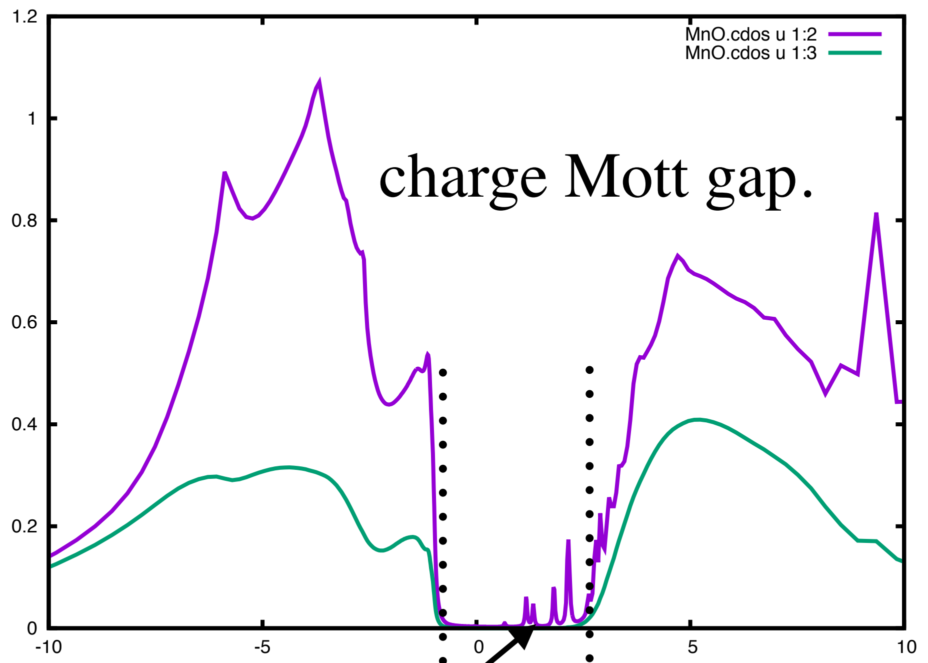
Recompute Green's function on the real axis

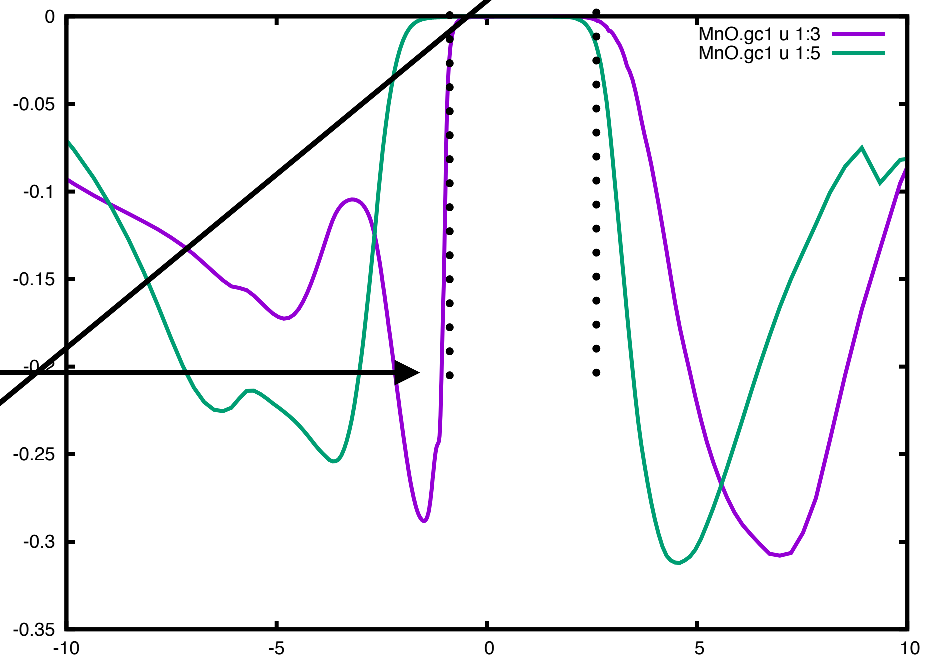
Check resulting DOS

```
> plot -x-10:10 -uall MnO.cdos
```

Check Green's t2g and eg Green's function

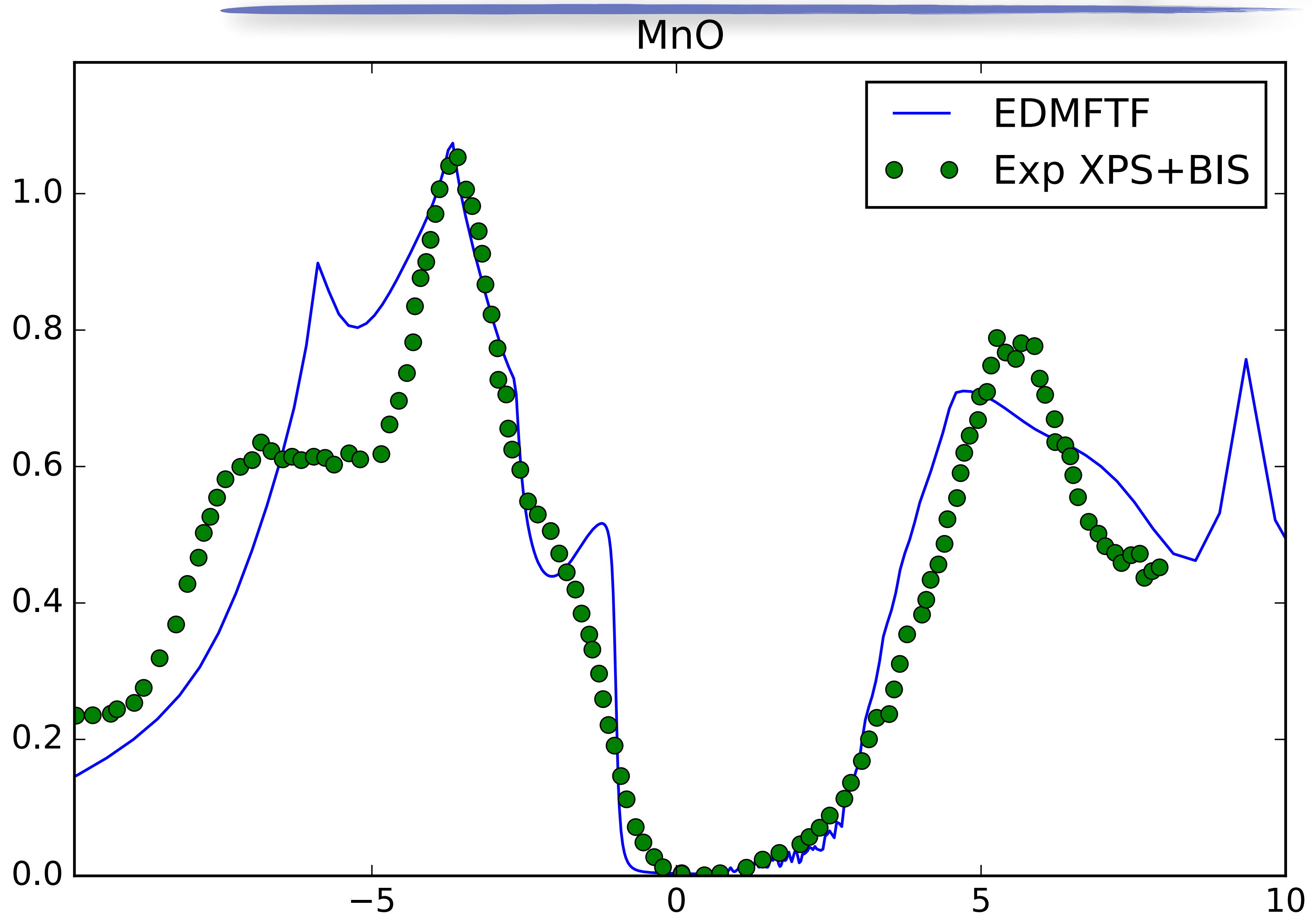
```
> plot -x-10:10 -u1:3,1:5 MnO.gc1
```





valence state is from Mn-d.
conduction state is not-Mn-d

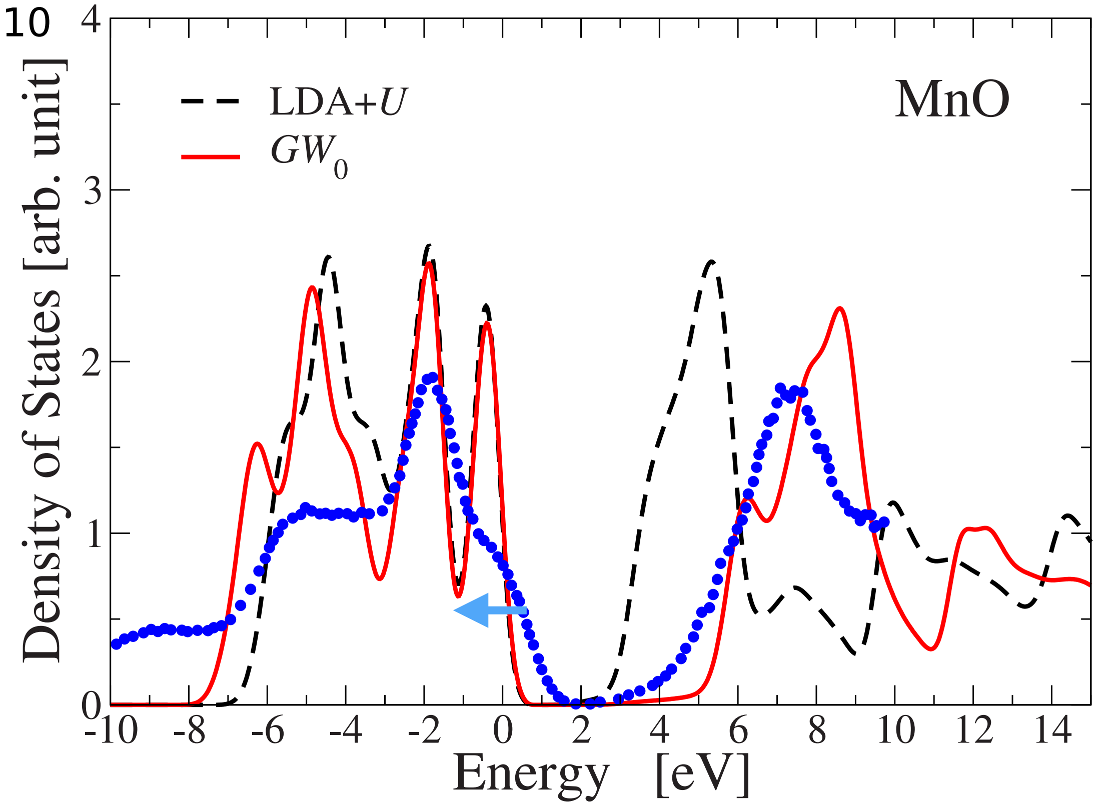
Check with Experiment



Should be careful with direct comparison(surface issues, matrix elements effects, low resolution for inverse Ph). Nevertheless, the agreement good, and far superior than LDA+U or GW.

Exp:
Sawatzky et.al., PRB 44, 1530 (1991)

Other methods (M. Scheffler):
PRB 82, 045108 (2010).



Postprocessing : A(k,w)

```
> cp $RESULTS/MnO/onreal/MnO.klist_band .
```

copy k-list path
can be created by xcrysden

```
> x_dmft.py lapw1 --band
```

recomputed KS bands on the current k-path.

*edit the second line of **MnO.indmfl** file and change **omega_min, omega_max** to -6 6 . We use 200 frequency points*

```
5 15 1 5                    # hybridization band index nemin and nemax, renormalize for interstitials, projection type
0 0.025 0.025 200 -3.000000 1.000000 # matsubara, broadening-corr, broadening-noncorr, nomega, omega_min,
omega_max (in eV)
1                         # number of correlated atoms
1    1   0                 # iatom, nL, locrot
  2   7   1                # L, qsplit, cix
```

changed file:

```
5 15 1 5                    # hybridization band index nemin and nemax, renormalize for interstitials, projection type
0 0.025 0.025 200 -6.000000 6.000000 # matsubara, broadening-corr, broadening-noncorr, nomega, omega_min,
omega_max (in eV)
1                         # number of correlated atoms
1    1   0                 # iatom, nL, locrot
  2   7   1                # L, qsplit, cix
```

```
> x_dmft.py dmftp
```

computed and store DMFT bands (eigvals.dat) on the same k-path

We did not change EF during self-consistent run, therefore “EF.dat” file was not yet created. For metals “EF.dat” should exist already. Check “mu” from **info.iterate** and copy it into “EF.dat”

```
> echo 6.530817 > EF.dat
```

```
> wakplot.py 0.02
```

adjust this number to have best contract

display the DMFT eigenvalues.

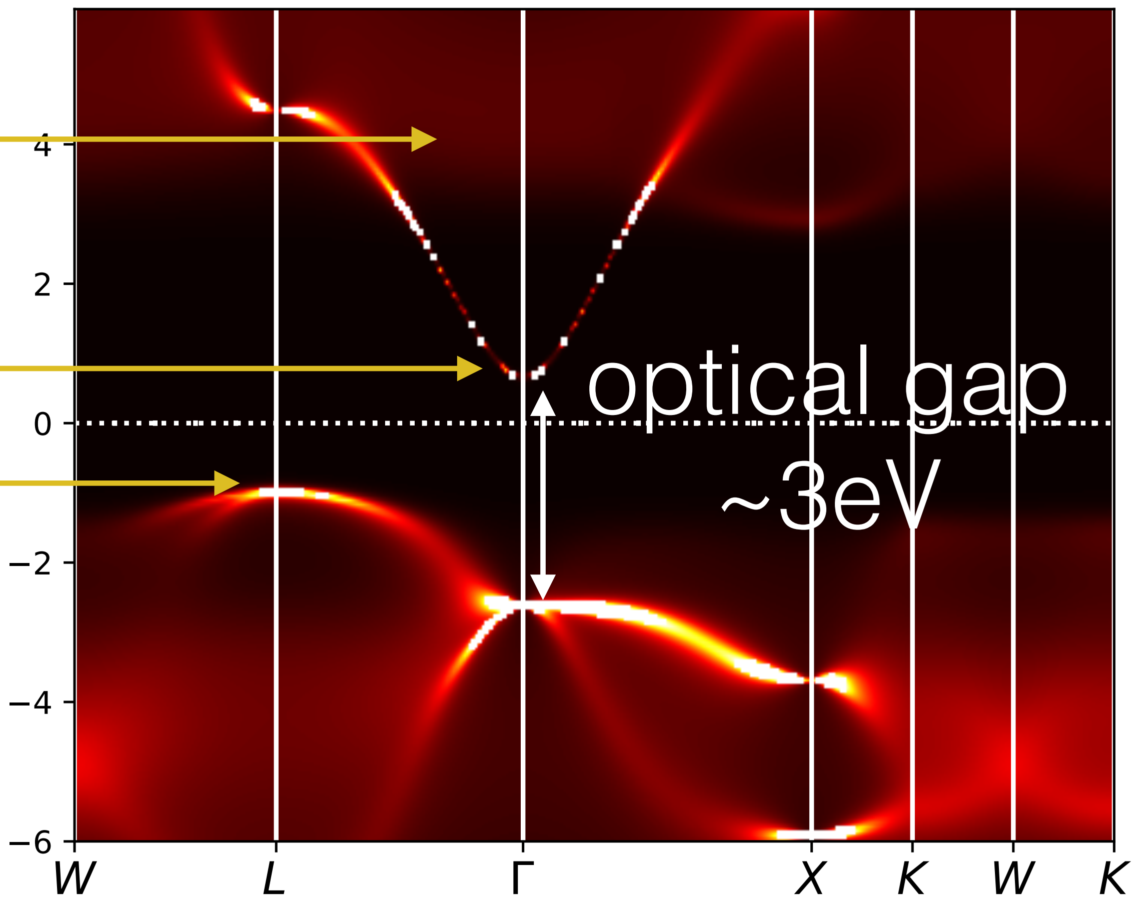
Should see picture like:

Mn-2tg state

itinerant 4s-state

hybrid Mn-eg state+
O-p state
(Zhang-Rice physics)

Exp: optical gap ~3eV,
PRB 58, 9783 (1998).



Exercise 2: FeSe example (iron superconductor)

```
> cd FeSe
```

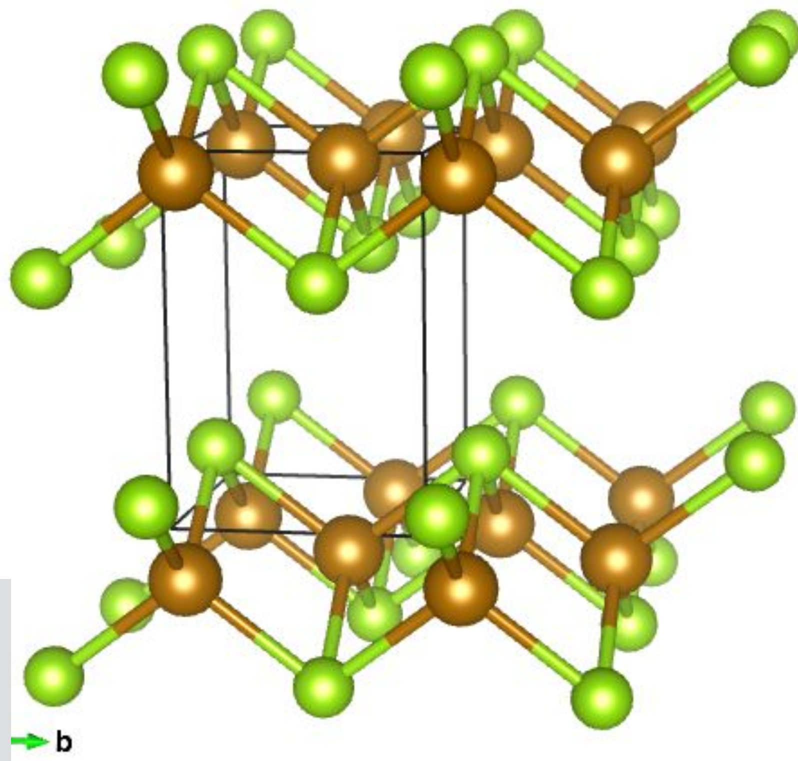
directory with Wien2k DFT run.

```
> init_dmft.py -ca 1,2 -ot d,d -qs 2,2
```

atom 1 and 2 are the two Fe in the u.c. They are correlated.

orbital type: The "d" orbital is correlated by DMFT

"d" orbitals in real harmonics with tetragonal symmetry



Add “params.dat" and “sig.inp”

```
> cp $RESULTS/FeSe/DMFT/params.dat .
```

```
solver        =  'CTQMC'   # impurity solver
DCs           =  'nominal'   # double counting scheme

max_dmft_iterations = 1      # number of iteration of the dmft-loop only
max_lda_iterations  = 100   # number of iteration of the LDA-loop only
finish         =  30              # number of iterations of full charge loop (1 = no charge self-consistency)

ntail          =  300              # on imaginary axis, number of points in the tail of the logarithmic mesh

cc             =  2e-5      # the charge density precision to stop the LDA+DMFT run
ec             =  2e-5      # the energy precision to stop the LDA+DMFT run

recomputeEF    = 1          # Recompute EF in dmft2 step. If recomputeEF = 2, it tries to find an insulating gap.

GoodGuess      = True       # We have a good guess for self-energy, and the scheduler optimizes run for good existing self-energy

# Impurity problem number 0
iparams0={"exe"      : ["ctqmc"           , "# Name of the executable"],
          "U"        : [5.0               , "# Coulomb repulsion (F0)"],
          "J"        : [0.8               , "# Coulomb repulsion (F0)"],
          "CoulombF" : ["'Full'"          , "# Can be set to 'Full'"],
          "beta"     : [50                , "# Inverse temperature"],
          "svd_lmax" : [25                , "# We will use SVD basis to expand G, with this cutoff"],
          "M"        : [10e6              , "# Total number of Monte Carlo steps"],
          "Mlist"    : [ [10e6]*5 + [20e6], "# Changing M"],
          "mode"     : ["SH"              , "# We will use self-energy sampling, and Hubbard I tail"],
          "nom"      : [200               , "# Number of Matsubara frequency points sampled"],
          "tsample"  : [100               , "# How often to record measurements"],
          "GlobalFlip" : [1000000         , "# How often to try a global flip"],
          "warmup"   : [3e5               , "# Warmup number of QMC steps"],
          "nf0"      : [6.0               , "# Nominal occupancy nd for double-counting"],
          }
```

metal requires to compute EF at each iteration

will start with a good guess for sigma

use rotationally invariant Coulomb U

number of MC steps will change. start with 10M, and later 20M

```
> cp $RESULTS/FeSe/DMFT/sig.inp .
```

copy a good guess for self-energy to achieve faster convergence.

Prepare & execute

```
> x kgen -f FeSe
      1000
      Do you want to shift k-mesh: 1
> run_lapw -NI
```

```
> mkdir LDA; mv * LDA/
> mkdir DMFT; cd DMFT
> dmft_copy.py ../LDA
```

Optional: move LDA data to directory LDA and start new calculation in directory DMFT. Copy necessary data to new directory by *dmft_copy.py* script.

```
> echo "mpirun -np 2 -x OMP_NUM_THREADS=1" > mpi_prefix.dat
```

```
> run_dmft.py >& nohup.dat
```

Monitor the run

```
> less info.iterate                          check the summary file
> less  dmft_info.out                        see what/how is executed
> less imp.0/nohup_imp.out.000               check what impurity is doing
> less FeSe.dayfile                          execution log+convergence
> plot -g -u1:9,1:10 info.iterate            impurity occupancy convergence
```

Note: due to a flag “GoodGuess=True” in params file, the calculation started with converging electronic charge on the currently provided self-energy (sig.inp), and impurity solver is run only after the charge is converged.

```
> plot -x:10 -g -u1:3,1:5,1:7,1:9,1:11 imp.0/Sig.out.?.1    plot impurity self-energy
```

Checking the amount of force on atoms

We can switch on the calculation of force on atoms by changing a variable in *FeSe.in2*

currently FeSe.in2 line1 contains:

```
# FeSe.in2, line 1 :
TOT                 (TOT,FOR,QTL,EFG,FERMI)
```

change to :

```
# FeSe.in2, line 1 :
FOR                 (TOT,FOR,QTL,EFG,FERMI)
```

After this change, *FeSe.scf* should contain a printout of the total force, for example:

```
       TOTAL FORCE IN mRy/a.u. = |F|      Fx             Fy             Fz      with/without FOR in case.in2
:FOR001:   1.ATOM       0.000000       0.000000       0.000000       0.000000 total forces
:FOR002:   2.ATOM      19.938644       0.000000       0.000000      -1.226529  total forces
       TOTAL FORCE WITH RESPECT TO GLOBAL CARTESIAN COORDINATES:
:FCA001:   1.ATOM                      0.000000       0.000000       0.000000 total forces
:FCA002:   2.ATOM                      0.000000       0.000000      -1.226529  total forces
       TOTAL FORCE WITH RESPECT TO THE GLOBAL COORDINATE SYSTEM:
:FGL001:   1.ATOM                 0.000000000     0.000000000     0.000000000 total forces
:FGL002:   2.ATOM                 0.000000000     0.000000000    -1.226529    total forces
```

This format is identical to the format in Wien2k, and it says that the Se atom has a force in z-direction of -1.22mRy/a.u. (for different RKmax, it might be somewhat different value). We can grep the force at each step by

```
grep ':FGL002' FeSe.scf
```

The DFT equilibrium value is **z=0.245**
DMFT equilibrium value : **z=0.268**
Experiment : **z=0.265**

Note: force of a few mRy/a.u. is very small and says that the structure is already optimized. This is because z-position of Se z=0.27 in this structure file is very close to its equilibrium in DMFT (and experiment).
The LDA equilibrium value is z=0.245, which would create very large force in DMFT.

If you wish to optimize the structure (which is not needed in this case), you just need to edit *FeSe.inm* file and change MSR1 in the first line to MSR1a , i.e.,

```
# FeSe.inm, line 1:
MSR1a   0.0   YES  (BROYD/PRATT, extra charge (+1 for additional e), norm)
```

Postprocessing : maxent + dos

Maximum entropy steps are identical as in previous MnO example. The same maxent_params.dat file should be good.

```
> mkdir maxent; cd maxent
> cp ../sig.inp .
> cp $RESULTS/MnO/DMFT/maxent/maxent_params.dat .
> maxent_run.py sig.inp
```

```
> mkdir ../onreal; cd ../onreal
> dmft_copy.py  ../
>cp ../maxent/Sig.out sig.inp
```

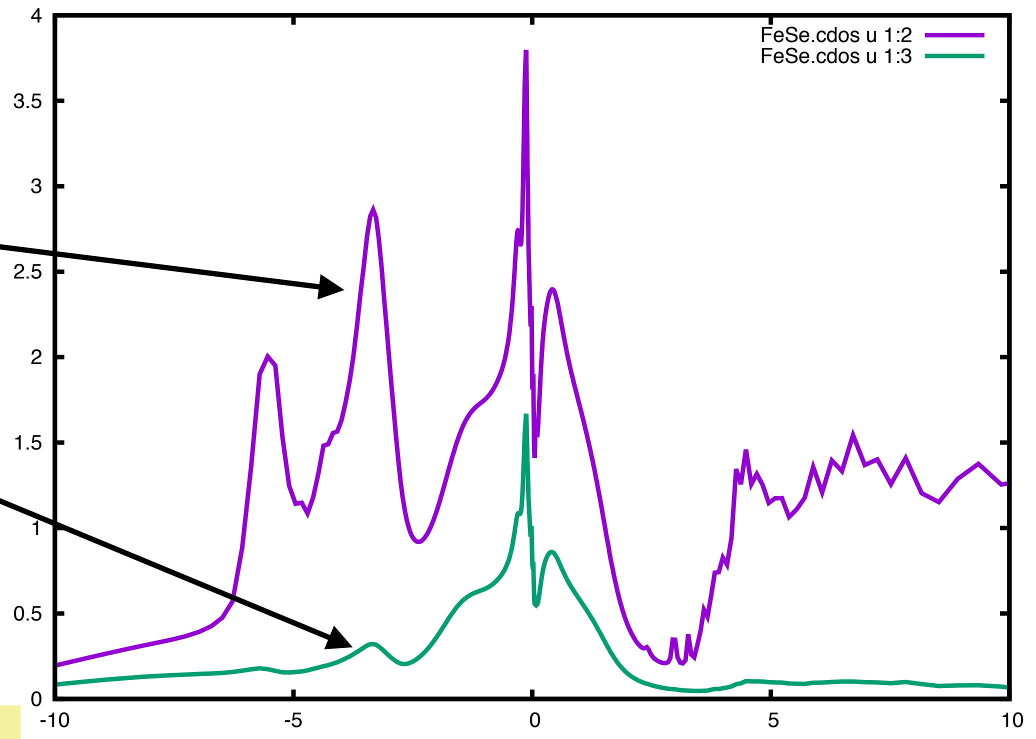
*edit the second line of **FeSe.indmfl** file and change the flag **matsubara** to 0*

```
> cp ../mpi_prefix.dat .
> x lapw0 -f FeSe
> x_dmft.py lapw1
> x_dmft.py dmft1
> plot -x-10:10 -uall FeSe.cdos
```

should give the following DOS:

total DOS

Fe:1 partial DOS



```
> cp $RESULTS/FeSe/onreal/FeSe.klist_band .
```

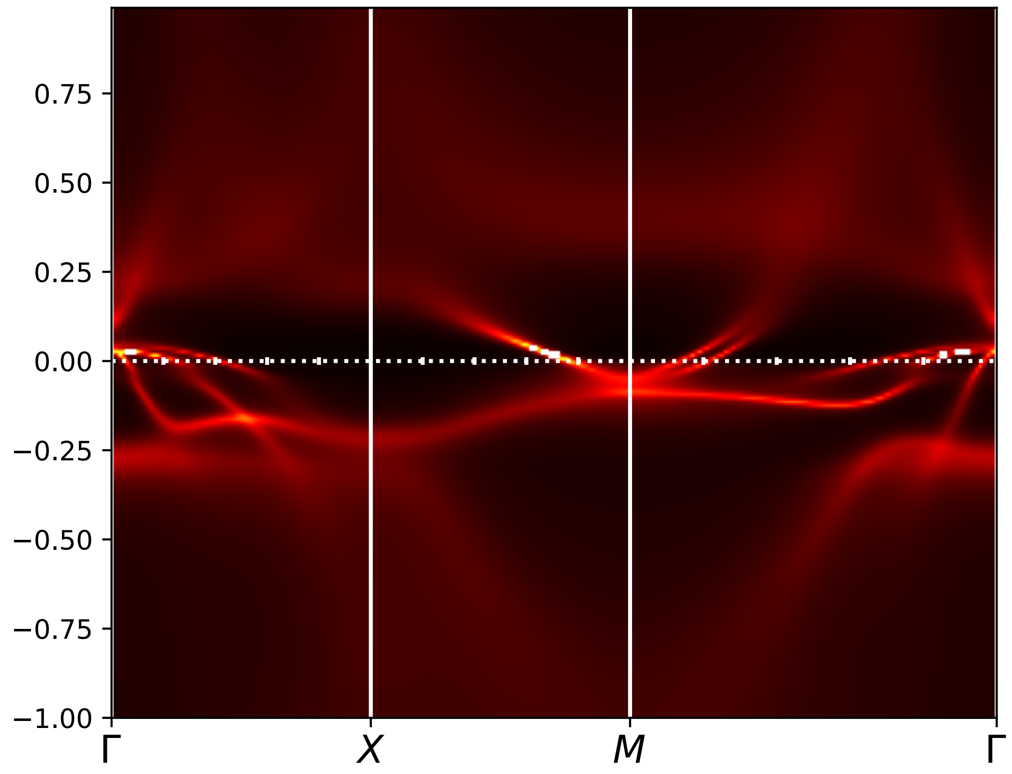
```
> x_dmft.py lapw1 --band
```

edit the second line of ***FeSe.indmfl*** *file and change* ***omega_min, omega_max*** *to -1 1 . We use 200 frequency points*

```
> x_dmft.py dmftp
```

```
> wakplot.py 0.5
```

should give the following spectra:



0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
Γ
X
M
Γ