

User's Guide, DFTWIEN2k_14.1+DMFT- Wien2K_2012(2014)

Kristjan Haule^α, Ang Liu^β, Peng Zhang^β and Ronald Cohen^β

^α: Rutgers University, Piscataway, New Jersey

^β: Carnegie Institution of Washington, Washington, D.C.

October 1, 2014

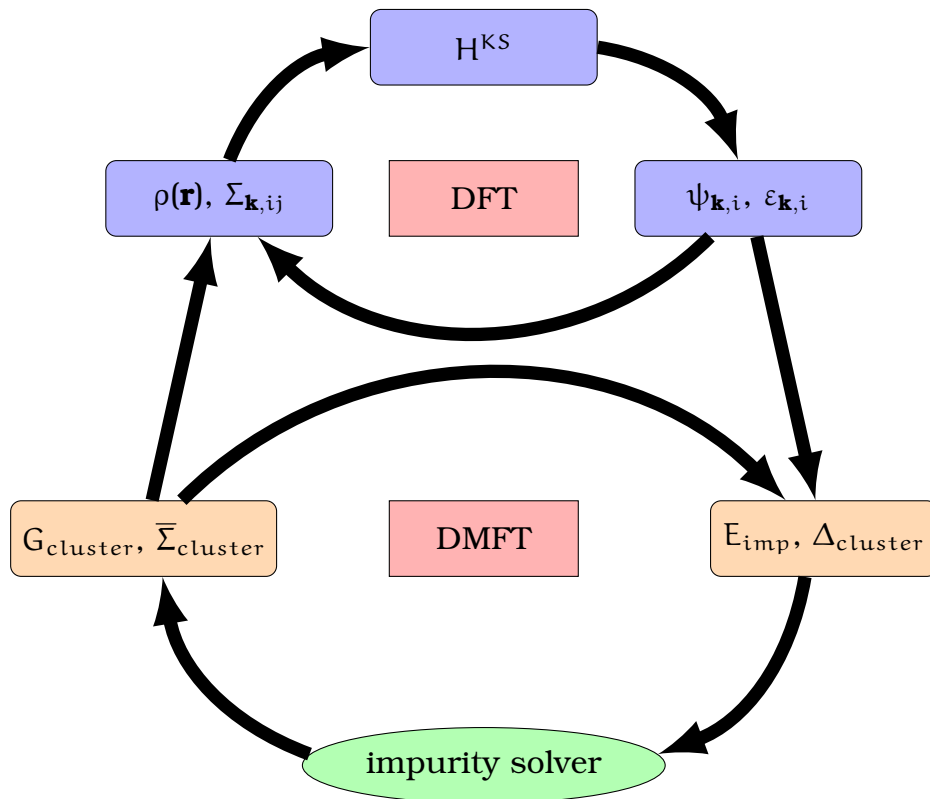


Table of Contents

I	Introduction	1
I.1	Electronic structure of crystal and Density Functional Theory (DFT)	4
I.2	Strongly correlated systems and Dynamical Mean-Field Theory (DMFT)	7
I.2.1	Strongly correlated systems	7
I.2.2	DMFT formulism	8
I.3	DFT + DMFT	11
I.3.1	Basics of DFT + DMFT	11
I.3.2	Flow chart (Figure 1)	14
II	Installation	16
III	General steps for running self-consistent cycles in DFT + DMFT	23
III.1	DFT part	23
III.1.1	Prepare struct-files	23
III.1.1.1	Gather all the required information	23
III.1.1.2	Generate struct-files	24
III.1.2	Initialize DFT calculations using init_lapw	35
III.1.3	Run the self-consistency cycles (SCF) of DFT calculations .	37

III.2	DMFT part	40
III.2.1	Initialize DMFT calculations	40
III.2.2	Prepare all files	42
III.2.2.1	Prepare input files (case.indmfl, case.indmfi) and output files	42
III.2.2.2	Prepare <i>params.dat</i> and <i>sig.inp</i> files	42
III.3	Run DFT + DMFT codes	47
III.3.1	Submit the job and run DFT + DMFT iterations	47
III.3.2	Monitor running of DFT + DMFT	50
III.3.3	Check convergence of DFT + DMFT cycles and plot the results	51
IV	Test cases	61
IV.1	FCC Fe at 1500 K 0 GPa	61
IV.1.1	Run DFT+DMFT codes	61
IV.1.1.1	Prepare struct-files	61
IV.1.1.2	Initialize calculations using <i>init_lapw</i>	63
IV.1.1.3	Run the self-consistency cycles (SCF) of DFT cal- culations	65
IV.1.1.4	Initialize DMFT calculations	66
IV.1.1.5	Prepare all files	68
IV.1.1.6	Submit the job and run DFT + DMFT iterations . .	70
IV.1.1.7	Monitor running of DFT + DMFT	72
IV.1.2	Properties based on DMFT calculations	73

IV.1.2.1	<i>histogram.dat</i> and <i>Probability.dat</i>	73
IV.1.2.2	Density of States (DOS) and Partial Density of States (Partial DOS)	77
IV.1.2.3	Spectral functions	85
IV.1.2.4	Optics including conductivity	88
IV.1.2.5	Fermi Surface	93

References	94
-------------------	-----------

List of Figures

1	Flow chart of DFT+DMFT	15
2	Main window of <i>w2web</i>	29
3	<i>StructGen</i> screen before clicking “Save Structure”	30
4	Choosing options for Automatic determination of Radius of Muffin-Tin (RMT)	31
5	<i>StructGen</i> screen after saving	32
6	<i>StructGen</i> screen after cleaning up	33
7	Chemical Potential v.s. number of DMFT loops in <i>info.iterate</i> 0 to 90	52
8	Chemical Potential v.s. number of DMFT loops in <i>info.iterate</i> 20 to 60	53
9	Chemical Potential v.s. number of DMFT loops in <i>info.iterate</i> 20 to 90	53
10	Impurity level v.s. number of DMFT loops in <i>info.iterate</i> 0 to 90 . .	54
11	Self-energy v.s. number of DMFT loops in <i>sig.inp.*.1</i> for e_g orbitals	55
12	Self-energy v.s. number of DMFT loops in <i>sig.inp.*.1</i> for t_{2g} orbitals	56
13	Conditioned self-energy v.s. number of DMFT loops in <i>sig.inp</i> before and after augmenting Monte Carlo measurements for e_g orbitals . .	57
14	Unconditioned self-energy v.s. number of DMFT loops in <i>Sig.outb</i> before and after augmenting Monte Carlo measurements for e_g orbitals	58
15	Imaginary part of self-energy v.s. real frequency in <i>Sig.out</i> for e_g and t_{2g} orbitals	60
16	Plot <i>histogram.dat</i> file	73

17	Plot <i>imp.0/Probability.dat</i> file	74
18	Plot <i>imp.0/Probability.dat</i> file 0-600	75
19	Real part of self-energy v.s. real frequency in Sig.out for e_g and t_{2g} orbitals	79
20	Total Density of States on the real axis for FCC Fe	81
21	Partial Density of States (e_g and t_{2g}) on the real axis for FCC Fe . .	83
22	Partial Density of States and Total Density of States on the real axis for FCC Fe	84
23	First Brillouin Zone of FCC Fe lattice	87
24	Spectral functions of FCC Fe	87
25	Optical conductivity of FCC Fe	91
26	Density of States based on optics calculations and comparison with Total DOS for FCC Fe	92

List of Tables

1	Variables for DFT	5
2	Formulas for DFT	6
3	Variables for DMFT	9
4	Formulas for DMFT	10
5	Variables for DFT + DMFT	12
6	Formulas for DFT + DMFT	13
7	Example <i>.txt</i> file to be converted into <i>.struct</i> file	26
8	Head of example struct-file generated for FCC Fe	34
9	Columns of <i>info.iterate</i> file	51
10	Lattice parameters in <i>FCC_Fe</i> text file	62

Part I

Introduction

One of the most challenging topics in condensed matter physics is the simulation and prediction of strongly correlated materials. The density functional theory plus dynamical mean-field theory (DFT+DMFT) method introduced in this userguide has been proven successful in fulfilling this target. The DFT+DMFT method is a combination of the traditional well-tested DFT method, in which the crystal problem is treated as the functional of electron density, and the DMFT method, where the local correlated physics is included in local self-energy. The DFT+DMFT method has been used in realistic simulations of strongly correlated materials, such as transition metals, lanthanides, actinides, as well as their compounds. We will give a detailed step-to-step introduction to the DFT+DMFT code in this userguide, from preparing the input files, checking whether the codes have arrived at converged and high-quality outputs, to how to extract realistic physical properties that can be compared with experiments.

This userguide is separated into four chapters: in the first chapter we give an introduction of the DFT+DMFT formalism; in the second chapter we present how to install the DFT+DMFT code; in the third chapter we present how to run the DFT+DMFT code, how to monitor the running, how to check if the outputs are converged, and how to improve the quality of your data; in the fourth chapter we use realistic examples to show the analysis of the DFT+DMFT output data, e.g.

derivation of physical properties, such as density of states (DOS), momentum resolved spectra, optical conductivity, etc.

The DFT+DMFT code has two parts: the DFT code and the DMFT code. In the DFT code, the inputs are atomic species and unit cell parameters. Its outputs are converged DFT Hamiltonian matrix at each k point and other parameters of the DFT Hamiltonian which are needed to construct local Green's function. The second part is the DMFT code. Inputs of DMFT are a choice of correlated orbitals and double counting, converged DFT Hamiltonian matrix and self-energy of previous iteration. After solving the DMFT equations with impurity solver, the local Green's function and new self-energy are fed back to the DFT iterations. There are three cycles in the DFT+DMFT codes, the DFT iterations, the DMFT iterations, and the DFT+DMFT charge self-consistent iterations. The final convergence is reached after the convergence of these three iterations.

1. The websites for downloading the source codes are shown below,

- Where to download the *WIEN2k* source code for DFT calculations (version WIEN2k_14.1):

<http://www.wien2k.at/>

- Where to download the *DMFT-Wien2K* source code for DMFT calculations (version 2012):

<http://hauleweb.rutgers.edu/downloads/>

2. The major references for WIEN2k and DFT+DMFT are listed below,

- The major reference for WIEN2k is shown here,

[Haule et al., 2010]

- The major references for DFT+DMFT are shown here,

[Blaha et al., 2001; Kotliar and Vollhardt, 2004; Kotliar et al., 2006)

I.1 Electronic structure of crystal and Density Functional Theory (DFT)

Density Functional Theory (DFT) is an exact way to solve the many-body problem for weakly interacting electronic systems with nuclei at fixed positions, for instance, atoms, molecules, nanosystems and crystals. Even though DFT is formally exact, but the exact functional is still unknown, and presumably the functional has a non-closed form and would be extremely non-local. Generally, DFT is good for vibrational properties, structural stability, elasticity and equations of states. Many systems are treated well using DFT, whereas DFT does not work well for other systems which have uncontrolled approximations. Below summarizes the variables and equations to construct exact functional for time-independent standard DFT. (see Tables 1, 2)

Solving Kohn Sham equations is one of the primary computational tasks, and it requires a self-consistent iterative process since the external potentials are dependent on rounds of calculating spin electron densities and orbitals. In summary, DFT is the exact theory to determine the ground state properties of electronic systems by spin electron densities. (Kotliar et al., 2006)

Table 1: Variables for DFT

$\rho(\mathbf{r})$	local electronic charge density of the solid
Ψ	exact many body wavefunction
V	external potential
V_{KS}	Kohn Sham potential
$T_s[\rho(\mathbf{r})]$	kinetic energy of noninteracting electron gas
E_{xc}	local exchange correlation energies
Ψ_i	Kohn-Sham wave functions for single particle orbitals

Table 2: Formulas for DFT

$\rho(\mathbf{r}) = \langle \Psi^* \Psi \rangle$	Ground state spin electron density
$E_{\text{tot}} = T_s[\rho(\mathbf{r})] + E_{\text{xc}}[\rho(\mathbf{r})] + V$	Total energy equation
$H = \sum_{i=1}^N (-\nabla^2) + \sum_i (V(\mathbf{r}_i)) + \frac{1}{2} \sum_{i \neq j} \left(\frac{e^2}{ \mathbf{r}_i - \mathbf{r}_j } \right)$	Exact Hamiltonian
$[-\nabla^2 + V_{\text{KS}}(\mathbf{r})]\Psi_i = \varepsilon_i \Psi_i$	Kohn-Sham equations
$V_{\text{KS}}(\mathbf{r}) = v_{\text{ext}}(\mathbf{r}) + v_{\text{H}}(\mathbf{r}) + v_{\text{xc}}(\mathbf{r})$	Kohn-Sham potential
$v_{\text{ext}}(\mathbf{r}) = \frac{\delta E_{\text{ext}}[n(\mathbf{r})]}{\delta n(\mathbf{r})}$	External potential
$v_{\text{H}}(\mathbf{r}) = \int \frac{n(\mathbf{r}')}{ \mathbf{r} - \mathbf{r}' }$	Hartree potential
$v_{\text{xc}}(\mathbf{r}) = \frac{\delta E_{\text{xc}}[n(\mathbf{r})]}{\delta n(\mathbf{r})}$	Exchange correlation potential
$\rho(\mathbf{r}) = \sum_{i=1}^N \Psi_i^*(\mathbf{r}) \Psi_i(\mathbf{r})$	Kohn Sham interacting density

I.2 Strongly correlated systems and Dynamical Mean Field Theory (DMFT)

I.2.1 Strongly correlated systems

Materials could be categorized into conductors, semiconductors and insulators based on their easiness of conducting electricity. Almost all the metals are conductors, but the physical properties of the metals and their oxides could be different if they belong to different groups in the periodic table.

Main-group metals, such as aluminum and silicon, have weakly correlated electrons, and we can assume their electrons are delocalized and independent. In this case, DFT allows us to compute the total energy of the materials with high accuracy.

On the other hand, we can not ignore the interactions between the electrons in the d and f shells in transition metals and their oxides, e.g., iron, vanadium and their oxides, because these electrons which occupy the partially filled shells are too close to each other in the narrow orbitals, and Coulomb repulsion among d and f electrons are too strong. In this case, the strongly correlated electrons cannot be viewed as being immersed in the static mean field generated by other electrons. Consequently, the strongly correlated materials are more sensitive to change of external parameters, for example, temperature, pressure or doping. While it is hard to exactly calculate the full many-body model Hamiltonians for strongly correlated systems, various analytical and numerical methods

have been developed to handle those challenges. Dynamical Mean-Field Theory (DMFT) is one of them.

I.2.2 DMFT formulism

To simplify the many-body lattice problem, theorists mapped the single-site lattice model (e.g., Hubbard model) onto a self-consistent quantum impurity model (e.g., Anderson impurity model), which forms the basis of DMFT. They apply the analytic and numerical techniques such as quantum Monte Carlo to solve the impurity models.

DMFT simplifies the spatial dependence of the correlations among electrons and yet accounts fully for their dynamics - that is, the local quantum fluctuations missed in static mean-field treatments. One can extract the density of states of the strongly correlated systems from the impurity model. One key point is DMFT reproduces the local part of the Green's function of the system. A more system-specific modeling of strongly correlated materials requires both band theory and many-body theory. Below summarizes the variables and formulas to derive the exact functionals, (see Tables 3, 4) (Kotliar et al., 2006)

Table 3: Variables for DMFT

i, j	lattice site
τ, τ'	time
t_{ij}	matrix element, describes hopping of electrons
σ	electron spin (\uparrow or \downarrow)
U	local Coulomb interaction
$n_{i\sigma}$	density of electrons at site i with spin σ
h.c.	Hermitian conjugate
H_{AIM}	Anderson impurity model Hamiltonians
H_{atom}	a general Hamiltonian, a lattice site's atomic degrees of freedom
$\varepsilon_{\nu}^{\text{bath}}$	a bath of electrons with energy levels
V_{ν}	hybridization
$c_{0,\sigma}$	atomic electrons
$a_{\nu,\sigma}^{\text{bath}}$	bath electrons
$\Delta(\omega)$	mean field
t_k	Fourier transform of the hopping matrix t_{ij}

Table 4: Formulas for DMFT

$G_{i\sigma}(\tau - \tau') \equiv - < c_{i\sigma}(\tau) c_{i\sigma}^\dagger(\tau') > .$	Green function specifies the probability
$H = \sum_{ij,\sigma} t_{ij} c_{i\sigma}^\dagger c_{j\sigma} + U \sum_i n_{i\uparrow} n_{i\downarrow}$	Hubbard model
$H_{AIM} = H_{atom} + \sum_{\nu,\sigma} \epsilon_\nu^{bath} n_{\nu,\sigma}^{bath} + \sum_{\nu,\sigma} (V_\nu c_{0,\sigma}^\dagger a_{\nu,\sigma}^{bath} + h.c)$	Anderson impurity model
$\Delta(\omega) = \sum_\nu \frac{(V_\nu)^2}{\omega - \epsilon_\nu^{bath}}$	hybridization function
$G[\Delta(\omega)] = \sum_k \omega - \sum [\Delta(\omega)] - t_k^{-1}$	self-consistency condition
$E_{tot} = T_s[\rho(\mathbf{r})] + E_{xc}[\rho(\mathbf{r}), G] + V$	total energy equation

I.3 DFT + DMFT

I.3.1 Basics of DFT + DMFT

To take into account the strong correlation effects in a more accurate way, a combined DFT + DMFT method has been proposed for studying the extended strongly correlated systems. In this approach, the first step is to do DFT approximations on system geometry and electron band structure of 'non-correlated' quasi-particles. In the DFT + DMFT approach for extended systems, the central quantity is the local Green's function, which is a particular case of the general time-ordered Green's function. In practice, DFT calculations give the optimized structure, then DMFT shows its Green's functions, includes the dynamical effects, implements self-consistency conditions and consequently physical properties of finite systems where electron-electron correlation effects play an important role.

Below summarizes the variables and formulas for DFT + DMFT, (see Tables 5, 6) (Kotliar et al., 2006; Kotliar and Vollhardt, 2004)

Table 5: Variables for DFT + DMFT

$c_{i\sigma l}^\dagger$	creation operator of the fermion at site i with spin σ
$c_{j\sigma m}$	annihilation operator of the fermion at site j with spin σ
$n_{i\sigma l} = c_{i\sigma l}^\dagger c_{i\sigma l}$	corresponding particle number operator
$t_{il;jm}$	matrix elements, hopping parameters
$U_{\sigma\bar{\sigma}}^{lm} n_{i\sigma l}$	Coulomb repulsion matrix
$\omega - \varepsilon_{\text{DFT}}(\mathbf{k})$	free quasi-particle spectrum
μ	chemical potential
$\mu - \Sigma(\omega)$	quasi-particle self-energy
$G_{\sigma l; \sigma' m}^{-1}(\omega)$	effective dynamical mean field
ψ, ψ^*	Grassmann variables

Table 6: Formulas for DFT + DMFT

$H = - \sum_{i,j,\sigma,l,m} t_{il;jm} c_{i\sigma l}^\dagger + \sum_{i,\sigma,\sigma',l,m} U_{\sigma\bar{\sigma}}^{lm} n_{i\sigma l} n_{i\bar{\sigma}m}$	typical DMFT Hamiltonian
$G_{i\sigma l;j\sigma m}(t, t') = -i \langle T c_{i\sigma l}(t) c_{j\sigma m}^\dagger(t') \rangle$	local Green's function
$G_{\sigma l;\sigma' m}(\omega) = \int \frac{d\mathbf{k}}{(2\pi)^d} \left(\frac{1}{\omega - \varepsilon_{\text{DFT}}(\mathbf{k}) + \mu - \Sigma(\omega)} \right)$	local Green's function
$n_{\sigma l} = - \int \frac{d\omega}{2\pi} \int \frac{d\mathbf{k}}{(2\pi)^d} \text{Im} G_{\sigma l;\sigma l}(\omega)$	orbital spin density
$G_{\sigma l;\sigma' m}^{-1}(\omega) = (G^{-1}(\omega) + \Sigma(\omega))_{\sigma l;\sigma' m}^{-1}$	effective dynamical mean field
$G_{\sigma l;\sigma' m}(\omega) = \int D[\psi] D[\psi^*] \psi_{\sigma l} \psi_{\sigma' m}^* e^{A[\psi, \psi^*, G^{-1}]}$	quantum impurity problem

I.3.2 Flow chart (Figure 1)

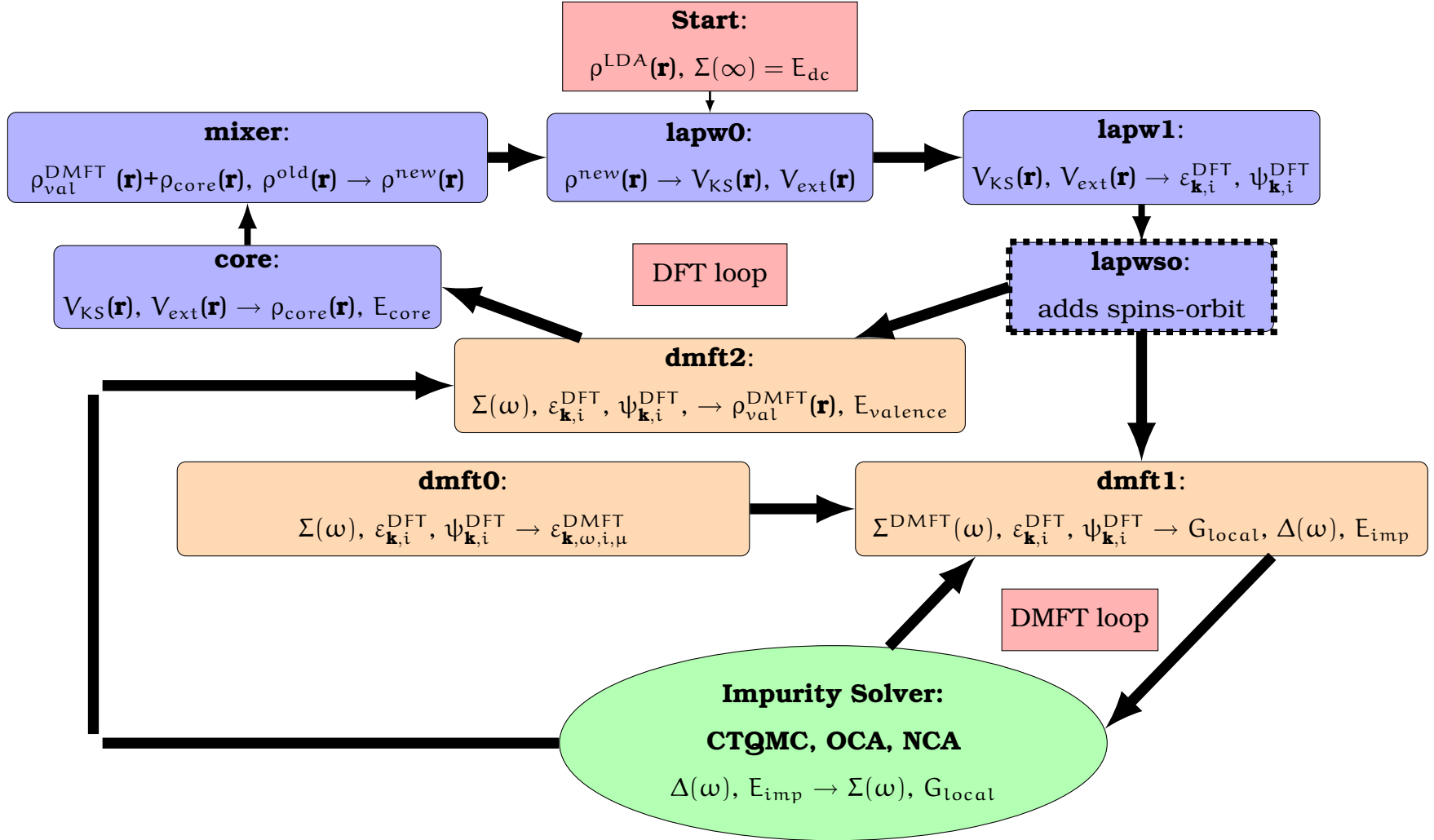
The following flow chart for DFT + DMFT code shows the two sections of the code, DFT code and DMFT code.

DFT code has the following inputs, atomic species and unit cell parameters. Its outputs are local, orthogonalized, converged DFT Hamiltonian matrix on a local axis at each k point, parameters of the unperturbed Hamiltonian which are needed to construct local Green's function.

DMFT code has the following inputs, choices of correlated orbitals and double counting, converged Hamiltonian matrix and self-energy. Its outputs are the bath function and a new self-energy. The output of the recomputed density will be used as input for the DFT calculations.

These two self-consistency processes (DFT self-consistency process and DMFT self-consistency process) will continue until both the total density and the local Green's function have converged.

Figure 1: Flow chart of DFT+DMFT



Part II

Installation

For detailed information about how to download, properly install and configure the *WIEN2k* source code for DFT calculations (version WIEN2k_14.1, please refer to *WIEN2k User's Guide, WIEN2k_14.1*.

<http://www.wien2k.at>

Furthermore, below illustrates how to properly install and configure the *DMFT-Wien2K* source code for DMFT calculations (version 2012).

The program consists of many independent programs and executables, which are written in C++, fortran 90 and Python. Before you start the installation, you must make sure that the following packages are installed in the system.

- [intel mkl library](#)
- [gnu C++ compiler](#)
- [intel fortran compiler or gnu fortran compiler](#)
- [gsl library: gnu scientific library](#)
- [Python with numpy and scipy](#)
- [Python CXX package](#)

Except for some Python packages, these are quite standard codes, which are likely already installed.

The compilation consists of six simple steps:

- download the code here:

<http://hauleweb.rutgers.edu/downloads/>,

unpack it (**tar xzvf dmft_w2k.tgz**) and cd to directory main.

- Edit the file *Makefile.in* under the main directory to set the path to compilers on the system, their options, and libraries (see description below)
- set the environment variable *WIEN_DMFT_ROOT* (in */.bashrc*) to point to the directory you plan to install the binaries code
- type **make**
- type **make install**
- for execution, also make sure that the following variables are properly set in *.bashrc*:
 - *WIENROOT* : should be set during *WIEN2k* installation to *WIEN2k* executables.
 - *PATH=\$WIENROOT:\$WIEN_DMFT_ROOT:\$PATH* : add both dmft and wien executables to your path.
 - *PYTHONPATH= :\$WIEN_DMFT_ROOT: :\$PYTHONPATH* : add dmft binary dir to *PYTHONPATH*.
 - *OMP_NUM_THREADS=< n >* : you might want to set number of cores for multithreading. Note that dmft1 and dmft2 steps support multi-

threading, ctqmc uses only mpi (which is very efficient in Monte Carlo). For ctqmc, it is best to have OMP_NUM_THREADS=1 to avoid collision between independent Monte Carlo walkers parallelized by MPI. For dmft1 and dmft2 steps, one can use double parallelization: k-point parallelization with MPI and extra parallelization with open_mpi, setting OMP_NUM_THREADS to number of cores on a node. For non-experts, set OMP_NUM_THREADS=1.

- EDITOR : environment variable EDITOR should be set (during WIEN2k installation) to preferred editor, for example “emacs -nw” or “vi”.
- SCRATCH : environment variable SCRATCH should be set to current dir, i.e., *export SCRATCH=.*

Here we will briefly describe the options which are set in *Makefile.in* and might need to be changed. Note that this file *Makefile.in* is included in all other makefiles located in subdirectories (there are around 20 makefiles in total), hence no other makefile needs to be edited.

Makefile.in contains:

- DESTDIR = \$ (WIEN_DMFT_ROOT) : Here we expect that environment variable \$WIEN_DMFT_ROOT is set on the system (likely in .bashrc).
- F90 = ifort : non parallel version of fortran compiler, which is used to compile smaller modules.
- F77 = ifort : some older modules need non-parallel fortran77 compiler.

- `preproc = cpp` : preprocessor, which can preprocess source code (in this case fortran files).
- `WFOPT = -xHOST -O3 -ipo -no-prec-div -traceback -FR -override-limits -openmp -I$(MKLROOT)/include/intel64/ lp64 -I$(MKLROOT)/include` : fortran options for both `dmft1 (ksum)` and `dmft2 (chargesc)` codes. More detailed explanation of some options.
 - `-O3 -override-limits -pad -mavx` : These options are for maximal optimization, regardless of memory required, and using intel's avx technology, etc... These options most likely need some changes on different platforms.
 - `-pre_div -mpl -pc80` : controls floating point precision. Might need to be changed.
 - `-FR` : is needed to tell compiler that fortran code is written in free format even when extension of files is “.f”.
 - `-openmp` : the code is using multi-threading techniques, hence it is wise to enable them.
 - `-DINTEL_VML` defines to use Intel's vector math library (if available).
- `FFLAGS = -O3 -fPIC` : similar fortran options than above. These are used in some fortran codes which are packed into python modules (.so) and they need extra option “-fPIC”. This option ensures position independent code, which can be packed into shared libraries.

- C++ = icpc : both g++ and intel icpc should work here. It is used for smaller impurity modules.
- OFLAGS = -O3 : options for C++ compiler for these small impurity modules.
- GFLAGS = -g -C : options for C++ compiler for these small impurity modules.
- CMP = f2py -fcompiler=intelem : name of the fortran to python converter. The option “-fcompiler=intelem” might need to be changed, because it depends on the platform.
- WLDFLAGS = -i-static \$(FOPT) : linking flags for many fortran codes, such as dmft1 step.
- WLIBS = -L\$(MKLROOT)/lib/intel64 \$(MKLROOT)/lib/intel64/
libmkl_blas95_lp64.a \$(MKLROOT)/lib/intel64/ libmkl_lapack95_lp64.a
-lmkl_cdft_core -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core
-lmkl_blacs_intelmpi_lp64 -liomp5 -lpthread -lm
: linking options for dmft2 steps, which does not use WLDFLAGS, and also requires intel math library.
- PFLAGS = -D_MPI -DMPICH_IGNORE_CXX_SEEK -I/home/user/gsl/include
: compilation flags for ctqmc code compiled with PC++. Here “-D_MPI” enables parallel version, “-DMPICH_IGNORE_CXX_SEEK” avoids some name-conflict between mpich and standard libraries. The rest of the options are for optimization.

- PC++ = mpicxx : parallel C++ compiler used for compiling ctqmc code.

- pcc = mpicc : parallel c-compiler

- PLIBS = -L\$(MKLROOT)/lib/intel64 \$(MKLROOT)/lib/intel64/

libmkl_blas95_lp64.a \$(MKLROOT)/lib/intel64/libmkl_lapack95_lp64.a

-lmkl_cdft_core -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core

-lmkl_blacs_intelmpi_lp64 -liomp5 -lpthread -lm -L/home/rcohen/lib

-L/share/apps/intel/impi/4.0.1.007/intel64/lib -impich -lgsl

: linking options for ctqmc. We need gsl library for random number generator.

- LLIBS = -L\$(MKLROOT)/lib/intel64 \$(MKLROOT)/lib/intel64/

libmkl_blas95_lp64.a \$(MKLROOT)/lib/intel64/libmkl_lapack95_lp64.a

-lmkl_cdft_core -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core

-lmkl_blacs_intelmpi_lp64 -liomp5 -lpthread -lm

-L/share/apps/intel/impi/4.0.1.007/intel64/lib -impich

: linking options for other C++ utility modules.

- CMPLIBS = -opt='-fast' -link-lapack.opt : options for f2py compiler. We need to link lapack and blas to several f2py modules.

- CMPLIBS2 = -f90flags='-openmp' -opt='-fast' -link-lapack.opt : some python modules written in fortran can support multithreading, hence we allow openmp.

Note: When installing numpy on a linux distribution, make sure to use gnu c++ compiler (as opposed to intel c++ compiler). Before installing numpy and scipy, you should set environment variables CC and CXX to CC=gcc and CXX=g++. Otherwise, f2py might use intel c++ compiler (icc and icpc), which does NOT work properly in some versions of python. (Haule et al., 2010)

Part III

General steps for running self-consistent cycles in DFT + DMFT

III.1 DFT part

III.1.1 Prepare struct-files

III.1.1.1 Gather all the required information

- The lattice parameters,

lengths a , b , c in Bohr or Ångstrom and angles α , β , γ .
- The space group according to the “International Tables for Crystallography”,

<http://it.iucr.org/>.
- The positions of all equivalent atoms (number them 1, 2, 3, ..., e.g., Fe 1, Fe 2, if there are many) and the positions of all inequivalent atoms in fractions of the unit cell. Alternatively you can utilize *StructGen* in WIEN2k to generate all the equivalent positions automatically by providing only the inequivalent positions and the space group.

III.1.1.2 Generate struct-files

A new struct-file should be generated on which we can run DFT and DMFT calculations. There are many approaches, such as by hand (using a simple command, converting other files into struct-files and manipulating example struct-files) and by using *w2web*.

- Use a simple command, *makestruct_lapw*

In the newest version of WIEN2k, i.e., WIEN2k_14.1, we can easily create struct-files by a simple command shown below and then interactively type in all the required input information.

“makestruct_lapw”

Note that this command can only generate a struct-file named *init.struct*. Copy it to your desired name *case.struct* by the command,

“cp init.struct case.struct”

Also, please be reminded that *makestruct_lapw* command is only suitable for small to medium sized cells. For large sized cells, *cif2struct* is recommended which will be introduced next.

- Convert other files into struct-files using *cif2struct*

Besides the *makestruct_lapw* command, converting established *.cif*, *.txt* files into *.struct* files is sometimes more convenient.

Store structural data in *case.cif* or *case.txt* files, then convert them into the *case.struct* files using the commands,

“cif2struct case.cif” or

“cif2struct case.txt”

Some *.cif* files are available for download from Crystallographic databases (e.g., the Inorganic Crystal Structure DataBase).

<http://it.iucr.org/>)

.txt files can be created by hand. Type in the following command to make a new *.txt* file.

“vi case”

Make sure that the *.txt* file contains all the data in the correct format, and one example of the contents of the *.txt* file is given below, (see Table 7)

Table 7: Example *.txt* file to be converted into *.struct* file

b						“a..Ang, b..Bohr”
0.0	0.0	0.0				“shift of origin”
8.1450	8.1450	8.1450	90	90	90	“a, b, c, angles α, β, γ ”
Fm-3m						“space-group symbol”
Fe						“atom- name”
0.0000000	0.0000000	0.0000000				“atomic position”

- Manipulate existing example struct-files

When the elements in the compounds are very similar and presumably the unit cell structures are quite similar, it is also acceptable to make some minor modifications on the parameters in the existing example struct-files. You can find and download some example struct-files in the installed WIEN2k folder, i.e., *example_struct_files* folder, or from the *Input Files/struct* column on Haule's website,

http://hauleweb.rutgers.edu/database_w2k/

- Use *w2web* to set up struct-files

The new interface *w2web* provides a more user-friendly environment to make struct-files and derive some properties thereafter.

For the first-time user, set up your username and password by the command,

“w2web -p xxxx”

in which “xxxx” stands for the default port number (7890) or a different port number between 1024 and 65536.

If you forget your portnumber, simply use the command,

“ps -ef | grep w2web”

If the remote server machine, e.g., *mw*, does not have the browser installed, you can establish a tunnel between *mw* server and your local server by the command,

“ssh -X -L 5890:localhost:7890 mw”

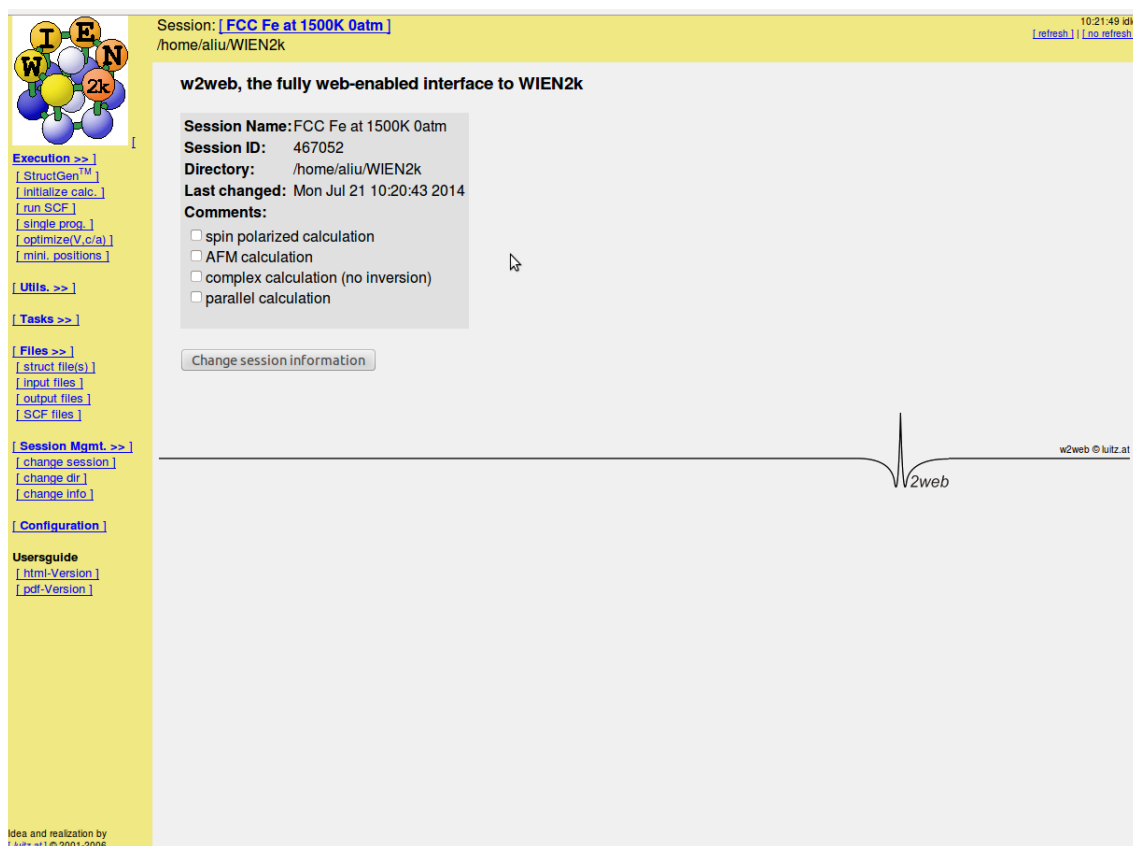
In the above case, 7890 and 5890 are considered as port 1 on the *mw* side and port 2 on the local side (any random number you want) respectively. Once the tunnel is established, you can start using *w2web* interface by opening your local browser, e.g., *firefox*, and typing in the following address and your username and password,

“username@localhost:5890”

In the *w2web* interface, there are three steps to set up the session part. The first step is to *Create new session* if you don't have one, choose your session

name (*case*) and click *Create*. The second step is to change into or create a new directory in which all your output files will be stored, and click *Select current directory*. The last step is to click *Click to restart session*, and the main window of *w2web* will show up. (see Figure 2)

Figure 2: Main window of *w2web*



On the left side of the *w2web* main window, start the struct-file generator by clicking on StructGenTM under the directory *Execution >>*. It will ask for the necessary information to create *case.struct*. The questions asked and some actions taken are listed below,

- Please specify the number of independent atoms of your initial structure

- Click *Generate template*
- Title
- Lattice Type
- Lattice parameters (lengths: a, b, c in Å or bohr) (angles: α , β , γ)
- Inequivalent Atoms:

Atom n: Atom-name

Pos n: Atomic position of atom n (see Figure 3)

Figure 3: *StructGen* screen before clicking “Save Structure”

The screenshot shows the StructGen web interface. At the top, the session is identified as "FCC Fe at 1500K 0atm" with the path "/home/aliu/WIEN2k". The main content area is a green box with the following fields:

- Title:** FCC Fe at 1500K 0atm
- Lattice:** Type: P. A dropdown menu shows options: P, F, B, CXY, CYZ, CXZ, R, H, 1_P1. A link "[Spacegroups from Bilbao Cryst Server]" is provided.
- Lattice parameters in bohr:**
 - a= 8.1450, b= 8.1450, c= 8.1450
 - α = 90.000000, β = 90.000000, γ = 90.000000
- Inequivalent Atoms: 1**
 - Atom 1: Fe, Z=, RMT=, [remove atom]
 - Pos 1: x= 0.00000000, y= 0.00000000, z= 0.00000000, [remove] [split]
 - [add position]
 - [add an atom]
- Number of symmetry operations: generate

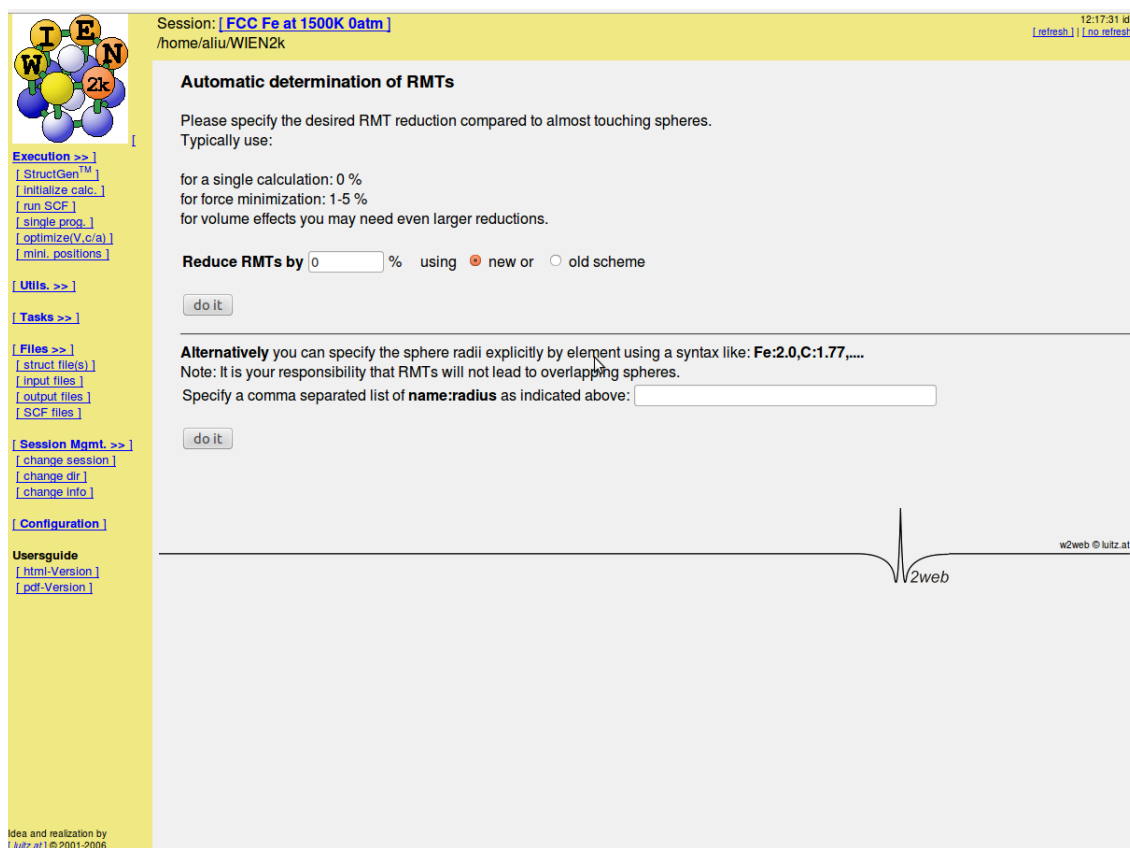
At the bottom of the green box, a message states: "You have to click 'Save Structure' for changes to take effect!" with a "Save Structure" button.

The left sidebar contains navigation links: Execution >>, [StructGen™], [Initialize calc.], [run SCF], [single prog.], [optimize(V,c/a)], [mini_positions], [Utils. >>], [Tasks >>], [Files >>], [struct file(s)], [input files], [output files], [SCF files], [Session Mgmt. >>], [change session], [change dir], [change info], [Configuration], [Usersguide], [html-Version], [pdf-Version].

The footer includes "Idea and realization by [kutz.at] © 2001-2006" and "w2web © kutz.at".

- Click *Save Structure*
- Click *set automatically RMT and continue editing (do it at least once!)* (RMT: muffin-tin radius, atomic sphere radius) (see Figure 4)

Figure 4: Choosing options for Automatic determination of Radius of Muffin-Tin (RMT)



Session: [\[FCC Fe at 1500K 0atm \]](#) 12:17:31 idle
/home/aliu/WIEN2k [\[refresh \]](#) [\[no refresh \]](#)

Automatic determination of RMTs

Please specify the desired RMT reduction compared to almost touching spheres.
Typically use:

for a single calculation: 0 %
for force minimization: 1-5 %
for volume effects you may need even larger reductions.

Reduce RMTs by % using ☒ new or ☐ old scheme

Alternatively you can specify the sphere radii explicitly by element using a syntax like: **Fe:2.0,C:1.77,....**
Note: It is your responsibility that RMTs will not lead to overlapping spheres.

Specify a comma separated list of **name:radius** as indicated above:

W2web © tuitz.at

Idea and realization by [\[tuitz.at \]](#) © 2001-2006

- Click *do it!* (Z and RMT values will be calculated automatically for each atom) (figure 5)

Figure 5: *StructGen* screen after saving

Session: [\[FCC Fe at 1500K 0atm\]](#) 12:18:01 idle
/home/aliu/WIEN2k [\[refresh\]](#) [\[no refresh\]](#)

StructGen™

You have to click "Save Structure" for changes to take effect!

[Save Structure](#)

Title:

Lattice:
Type: F

[\[Spacegroups from Bilbao Cryst Server\]](#)

P
F
B
CXY
CYZ
CXZ
R
H
1_P1

Lattice parameters in

a= b= c=
 α = β = γ =

Inequivalent Atoms: 1

Atom 1: Fe Z= RMT= [\[remove atom\]](#)
 Pos 1: x= y= z= [\[remove.\]](#) [\[split\]](#)
[\[add position\]](#)
[\[add an atom\]](#)

Number of symmetry operations: generate

You have to click "Save Structure" for changes to take effect!

[Save Structure](#)

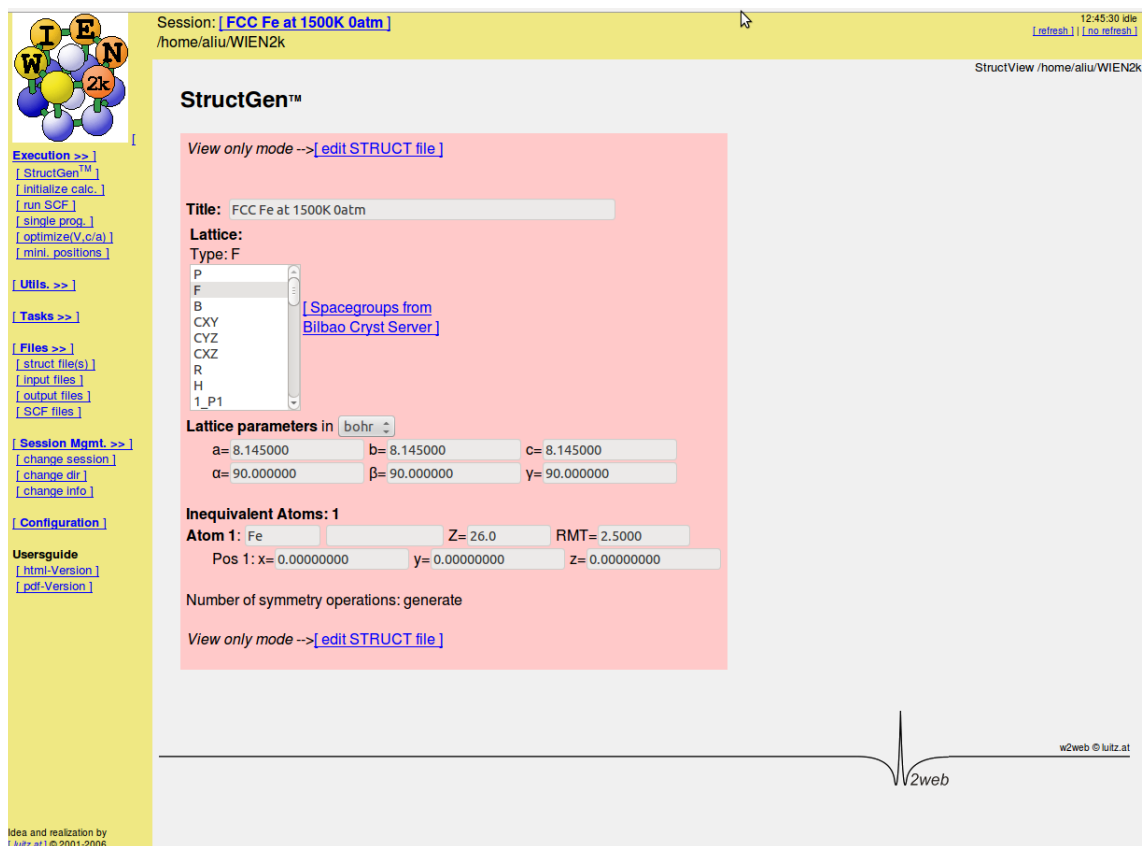
Idea and realization by [\[kutz.at\]](#) © 2001-2006

StructEdit /home/aliu/WIEN2k

w2web © kutz.at

- Click *Save Structure*
- Click *save file and clean up (when you are done)* (View only mode of StructGen file will show up) (see Figure 6)

Figure 6: *StructGen* screen after cleaning up



To double check if your *case.struct* is in your specified directory, go to *show all files* under the directory *Files >>*.

- One example of *.struct* file

Below illustrates the head part of one example of the generated struct-file for FCC Fe. (see Table 8)

Table 8: Head of example struct-file generated for FCC Fe

```
FCC Fe
F LATTICE,NONEQUIV.ATOMS          1          225          Fm-3m
MODE OF CALC=RELA unit=bohr
8.145000          8.145000  8.145000  90.00          90.00  90.00
ATOM 1:          X=0.0000  Y=0.0000  Z=0.0000
MULT= 1  ISPLIT= 2
Fe          NPT= 781  R0=.000050  RMT= 2.5000  Z: 26.000
LOCAL ROT MATRIX:          1.00000  0.00000  0.00000
          0.00000  1.00000  0.00000
          0.00000  0.00000  1.00000
48 NUMBER OF SYMMETRY OPERATIONS
1          0          0          0.000
0          -1          0          0.000
0          0          -1          0.000
          1
```

III.1.2 Initialize DFT calculations using `init lapw`

`case.struct` file contains all the information needed for the DFT calculations of the crystal structures. Run the following command,

“`init lapw`”

The following shows all the subsequent questions and corresponding answer choices.

1. The screen first shows:

next is `setrmt`.

Automatic determination of RMTs. Please specify the desired RMT reduction compared to almost touching spheres.

Typically, for a single calculation just hit enter, for force minimization use 1-5; for volume effects you may need even larger reductions.

2. Answers choices to a series of questions,

(a) Use old or new scheme (o/N)

(b) accept these radii; discard them; or rerun `setRmt` (a/d/r)

(c) nn-bondlength factor (usually=2)

(d) continue with `sgroup` or edit the `case.struct` file (c/e)

(e) continue with symmetry (old `case.struct`) or use/edit `case.struct.sgroup`
(c/e)

(f) continue with `lstart` or edit the `case.struct.st` file (c/e/x)

- (g) Eventually specify switches for instgen.lapw (or press ENTER): -up (default) -dn -nm (non-magnetic) -ask
- (h) SELECT XCPOT, 13:PBE-GGA (Perdew_Burke-Ernzerhof 96) 5:LSDA 11:WC-GGA (Wu-Cohen 2006) 19:PBESol-GGA (Perdew et al.2008)
- (i) SELECT ENERGY to separate core and valence states: -6.0 Ry (check how much core charge leaks out of MT-sphere)
- (j) continue with kgen or edit the case.inst file and rerun lstart (c/e)
- (k) number of k-points in whole cell, usually 200-500
- (l) continue with dstart or execute kgen again or exit (c/e/x)
- (m) do you want to perform a spinpolarized calculation (n/y)

III.1.3 Run the self-consistency cycles (SCF) of DFT calculations

After *init_lapw*, there are many new files generated in the *case* folder. The next step is to run self-consistency cycles (SCF) of DFT calculations. It is strongly suggested to submit the DFT calculations job via *submit_dft.scr* script utilizing services provided by a parallel computing machine (e.g., *mw* server). An example of *submit_dft.scr* file is shown below,

```
#!/bin/bash

#$ -N /test

#$-S /bin/bash

#$-cwd

#$-j y

#$-r no

#$-m abes

#$-M user@carnegiescience.edu

#$-pe mpi 8

#$-P GL

#$-l vf=2G

#$-l h_vmem=2G

ulimit -l unlimited

. /home/user/.bashrc

pwd
```

```

###echo "env mpirun -verbose -n 4 " > mpi_prefix.dat

echo $jobdir

export HOME=/san2/user

###export TMPDIR=/san3/smandal/main_July_2012

export OMP_NUM_THREADS=2

set -x

pwd

echo $jobdir

export OMP_NUM_THREADS=2

env

cat $PE_HOSTFILE

let n='mkmachinefile.sh $PE_HOSTFILE'

cat .machines

echo n=$n

###export MY_NSLOTS=$n

echo "env OMP_NUM_THREADS=1 mpirun -n 8 " > mpi_prefix.dat

echo "env OMP_NUM_THREADS=1 mpirun -n $n " > mpi_prefix.dat2

#which mpirun

/share/apps/intel13/composer_xe.2013.0.079/bin/compilervars.sh intel64

#export INTEL_LICENSE_FILE=27000@troy.tacc.utexas.edu

#echo ${INTEL_LICENSE_FILE}

#ls -l /etc/redhat -release

$WIENROOT/run_lapw -p

```

The command to submit the DFT calculation job to *mw* server is as follows,

“qsub submit_dft.scr”

Another way to run SCF is to use the command **“run lapw”**, but it is not recommended.

The self-consistency cycles (SCF) consist of the following parts:

LAPW0 (POTENTIAL) generates potential from density

LAPW1 (BANDS) calculates valence bands (eigenvalues and eigenvectors)

LAPW2 (RHO) computes valence densities from eigenvectors

LCORE computes core states and densities

MIXER mixes input and output densities

After SCF of DFT calculations, good charge densities have been converged.

You can save your results by the command **“save lapw”**.

III.2 DMFT part

III.2.1 Initialize DMFT calculations

Following *init_lapw* and SCF of DFT, proceed to the command,

“init_dmft.py”

to invoke DMFT calculations in the current directory.

Answer choices thereafter are shown below,

- Specify correlated atoms (e.g. 1-4, 7, 8):
- Do you want to continue; or edit again? (c/e):
- For each atom, specify correlated orbital(s) (e.g. d, f):
- Do you want to continue; or edit again? (c/e):
- Specify qsplit for each correlated orbital (default = 0):
- Do you want to continue; or edit again? (c/e):
- Do you want to group any of these orbitals into cluster-DMFT problems?
(y/n):
- Enter the correlated problems forming each unique correlated problem, separated by spaces (e.g. 1, 3 2, 4 5-8):
- Do you want to continue; or edit again? (c/e):
- Broken symmetry run? (y/n):

- Range (in eV) of hybridization taken into account in impurity problems;
default -6.0, 6.0: -10, 10:
- Perform calculations on real; or imaginary axis? (r/i):
- Is this a spin-orbit run? (y/n):
- Generate blank sig.inp? (y/n):

III.2.2 Prepare all files

III.2.2.1 Prepare input files (*case.indmfi*, *case.indmfl*) and output files

The initialization of DMFT calculations by *init_dmft.py* will generate two input files, *case.indmfi* and *case.indmfl*. These two files will connect the solid and impurity with DMFT equations.

Create a new folder that is in the parent directory,

“`mkdir case_backup`”

then move all the files from the *case* folder into the *case_backup* folder by the command,

“`mv ./* ../case_backup/`”

then type in the following command in the current directory (*case/*),

“`dmft.copy.py <parent directory>/case_backup/ -a`”

-a: files for both DFT and DMFT will be copied.

This command makes a copy of all the output files of DFT calculation results and DMFT initialization results into the folder *case*. These files are needed for the following DMFT calculations. Examples of copied files are listed below,

struct-file, WIEN2k files (*.in0*, *.in1*, *.in2*, *.inm*, *.inso*, *.inc*), charge density file (*.clmsum*), input files (*.indmfi*, *.indmfl*) and other files (*.klist*, *.kgen*, *.scf2*).

III.2.2.2 Prepare *params.dat* and *sig.inp* files

We need to prepare two more files needed for DMFT calculations in the folder, *params.dat* and *sig.inp*.

- *params.dat*

Construct *params.dat* file. Below illustrates one example of the *params.dat* file.

```
solver = 'CTQMC'

finish=20          # how many charge self-consistent iterations

max_dmft_iterations=1  # how many DMFT iterations inside charge loop

max_lda_iterations=1  # how many DMFT iterations inside charge loop

UpdateAtom=0        # the impurity cix file needs to be recomputed?

DCs='fixn'          # Double counting scheme

mix_delta=1.0

wbroad=0.005        # Broadening of the hybridization function

kbroad=0.1           # Broadening of the hybridization function

ntail=300

recomputeEF=2

saver=1.0

# Impurity problem number 0

iparams0= {

    "exe"    : ["ctqmc" , "# Name of the executable"],

    "U"      : [8.0 , "# Coulomb repulsion (F0)"],

    "beta"   : [38.68333333 , "# Inverse temperature"],

    "cx"     : [0.0 , "# Spin-orbit"],

    "M"      : [10000000 , "# Total number of Monte Carlo steps"],
```

```

"Ncout"           : [1000000 , "# How often to print out info"],
"Naver"           : [100000000 , "# How often to print out debug info"],
"nf0"             : [6.0 , "# Double counting parameter"],
"nc"              : [[4,5,6,7,8] , "# Impurity occupancies"],
"nom"             : [40 , "# Number of Matsubara frequency points"],
"aom"             : [5 , "# Number of frequency points"],
"sderiv"          : [0.02 , "# Maximum derivation mismatch accepted"],
"Ntau"            : [1000 , "# Number of imaginary time points"],
"SamplpeGtau"     : [1000 , "# How often to update G(tau)"],
"GlobalFlip"      : [1000 , "# How often to try a global flip"],
"tsample"         : [10 , "# How often to record measurements"],
"warmup"          : [1000000 , "# Warmup number of QMC steps"],
"CleanUpdate"     : [100000 , "# How often to make clean update"],
"minM"            : [1e-10 , "# The smallest atomic trace"],
"minD"            : [1e-10 , "# The smallest determinant"],
"Ncorrect"        : [-1 , "# Which baths should not be corrected"],
"PChangeOrder"    : [0.9 , "# Ratio between trial steps"],
"ChooseRandom"    : [0.8 , "# How often to choose kink"],
"CoulombF"        : ["Georges" , "# Full Coulomb repulsion"],
"OCA_G"           : [False , "# No OCA diagrams"],
}

```

Or you can download example *params.dat* files from Haule's W2K Database website,

http://hauleweb.rutgers.edu/database_w2k/

Move it from your local server to *mw* work directory and use the command,

"scp <local_directory> <mw_work_directory>/FCC_Fe/"

params.dat file contains information about the program flow and impurity solver. Leave the *params.dat* file name as it is. In the *params.dat* file, you can change the following parameters,

– finish=

The number of charge self-consistent iterations in *params.dat* file, e.g.,

finish=60 means there are 60 iterations in DFT+DMFT calculations.

– iparams0={"beta"}

"beta" is the inverse temperature. It is derived from the absolute temperature in Kelvin, $\beta = \frac{1(\text{eV})}{T(\text{K})}$ in which $1\text{eV} = 11604\text{K}$. For example,

if absolute temperature T is 300 K, then $\beta = \frac{11604\text{K}}{300\text{K}} = 38.68$. **"beta":**

38.68.

– iparams0={"M"}

"M" is the Total number of Monte Carlo steps. Typical M is 5 million (5e6), i.e., **"M": 5e6**. Total Monte Carlo measurements (N) is calculated

by the equation, $N = (\text{Number of CPU}) * M$. (*Number of CPU*) could be changed in *mpi* value in the *submit_dmft.scr* file.

- *sig.inp*

sig.inp is a blank self-energy file, and it is generated by the command,

“szero.py -e 35.7 -T 0.026 -n 3000”

Make sure that the T value here (0.026) is the same as the *beta* value in *params.dat* file. The meanings of the characters (e , T , n) are as follows,

– e

Double counting energy, calculated by the equation:

$$E_{DC} = U(n - \frac{1}{2}) - J/2(n - 1)$$

– T

Effective temperature in DMFT which should be consistent with *beta* in the *params.dat* file. For example, if we have *beta*=38.68333, then

$T = \frac{1}{\text{beta}} = \frac{1}{38.68333} = 0.026$. Please notice that T here is the effective temperature in DMFT derived from *beta* which is different from absolute temperature.

– n

The number of Matsubara frequency points

III.3 Run DFT + DMFT codes

III.3.1 Submit the job and run DFT + DMFT iterations

Copy *submit_dmft.scr* script file into the current directory (*case*). An example of *submit_dmft.scr* file is shown below.

```
#!/bin/bash

#$ -N test

#$-S /bin/bash

#$-cwd

#$-j y

#$-r no

#$-m abes

#$-M user@carnegiescience.edu

#$-pe mpi 8

#$-P GL

#$-l vf=2G

#$-l h.vmem=2G

ulimit -l unlimited

. /home/user/.bashrc

pwd

###echo "env mpirun -verbose -n 4 " > mpi_prefix.dat

echo $jobdir

export HOME=/san2/user
```

```

###export TMPDIR=/san3/smandal/main_July_2012

export OMP_NUM_THREADS=2

set -x

pwd

echo $jobdir

export OMP_NUM_THREADS=2

env

cat $PE_HOSTFILE

let n='mkmachinefile.sh $PE_HOSTFILE'

cat .machines

echo n=$n

###export MY_NSLOTS=$n

echo "env OMP_NUM_THREADS=1 mpirun -n 8 " > mpi_prefix.dat

echo "env OMP_NUM_THREADS=1 mpirun -n $n " > mpi_prefix.dat2

#which mpirun

/share/apps/intel13/composer_xe.2013.0.0.79/bin/compilervars.sh intel64

#export INTEL_LICENSE_FILE=27000@troy.tacc.utexas.edu

#echo ${INTEL_LICENSE_FILE}

#ls -l /etc/redhat-release

$WIEN_DMFT_ROOT/run_dmft.py > wnohup.dat 2>&1

```

The final step is to type in the command,

“qsub submit_dmft.scr”

Using remote powerful machine (*mu*) is very efficient when computing large

number of Monte Carlo cycles due to the large number of CPU cores it has. The command above is going to create a *.dat* file for mpi parallel execution, *mpi_prefix.dat*.

III.3.2 Monitor running of DFT + DMFT

During the run, we can monitor the information file and update the parameters, if necessary. The files we can monitor are listed below,

- *info.iterate*

It contains information about chemical potential, impurity levels, exchange energies and total fillings, etc.

- *:log*

It helps monitor the modules that are running.

- *dmft_info.out, case.dayfile*

To monitor the DMFT calculations and convergence levels.

- *imp.0/nohup_imp.out, imp.0/Sig.out, imp.0/Gf.out*

To monitor the impurity levels.

- *dmft1_info.out, case.outputdmf1.0*

To monitor the dmft1 step.

- *dmft2_info.out, case.outputdmf2.0*

To monitor the dmft2 step.

III.3.3 Check convergence of DFT + DMFT cycles and plot the results

To check convergence of DFT + DMFT cycles, you need to plot several output files, for example, *info.iterate*, *sig.inp.*.1*, *sig.inp* and *imp.0/Sig.outh*. Here we use the *Test Case 0: FCC Fe* as an example.

- Plot *info.iterate* v.s. iterations

There are seven columns in the *info.iterate* file representing the following information respectively, (see Table 9)

Table 9: Columns of *info.iterate* file

μ	chemical potential
E_{imp}	impurity level
$E_{\text{imp}}[-1]$	impurity level
E_{dc}	exchange energy
nf	total filling

Note that if you change the parameters in the *params.dat* file and resubmit the DMFT calculations, the *info.iterate* file will recount the number of DMFT loops finished.

Use gnuplot to plot the chemical potential, impurity level as a function of the number of DMFT loops.

- Plot chemical potential v.s. number of DMFT loops in *info.iterate* (see

Figure 7). In this graph, from iteration 0 to iteration 10, fluctuations of chemical potentials are noticeable. From iteration 10 to iteration 20, it starts flattening out. From iteration 20 to iteration 90, it becomes quite stable.

Figure 7: Chemical Potential v.s. number of DMFT loops in *info.iterate* 0 to 90

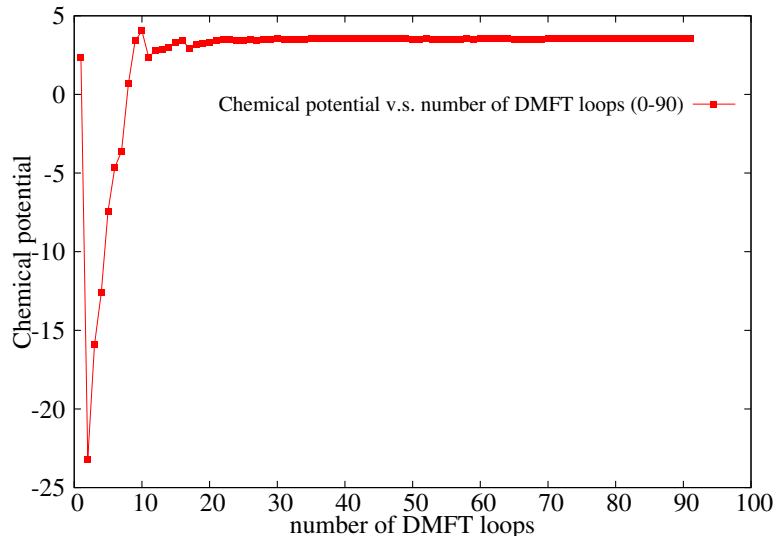


Figure 8 and Figure 9 show the zoomed-in significant improvement feature of convergence by increasing the Monte Carlo number “M” from 12e6 (iteration 20-60) to 120e6 (iteration 60-90). M number could be changed in *params.dat* file. Also, you can keep M number the same, but increase number of CPU cores by 10 times instead.

Figure 8: Chemical Potential v.s. number of DMFT loops in *info.iterate* 20
to 60

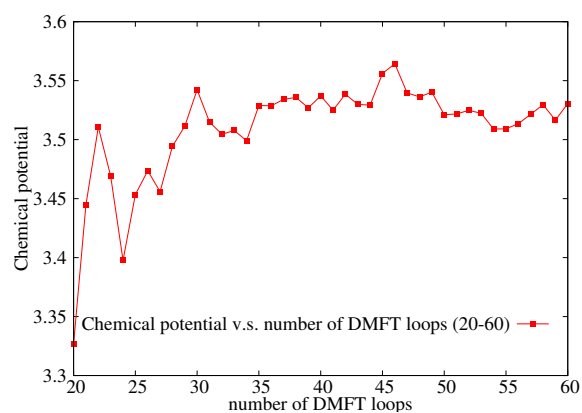
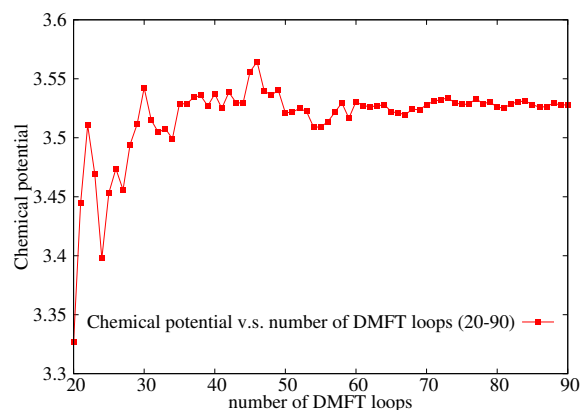
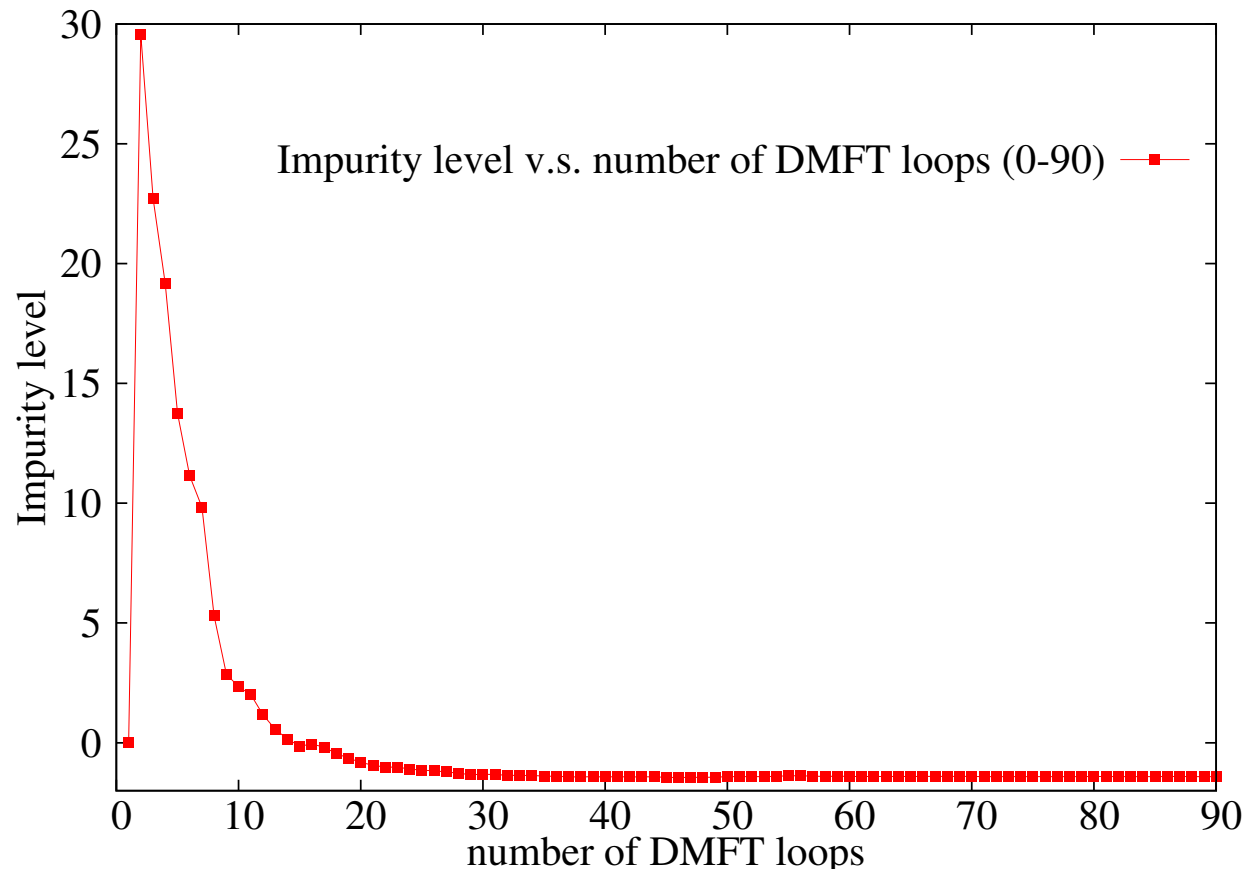


Figure 9: Chemical Potential v.s. number of DMFT loops in *info.iterate* 20
to 90



- Plot impurity level v.s. number of DMFT loops in *info.iterate* (see Figure 10)

Figure 10: Impurity level v.s. number of DMFT loops in *info.iterate* 0 to 90



- Plot self-energies in *sig.inp.*.1*

We can check the convergence of DFT + DMFT iterations from self-energy files.

Suppose we have finished 90 iterations, then we plot the last 5 self-energy files, *sig.inp.86.1*, *sig.inp.87.1*, *sig.inp.88.1*, *sig.inp.89.1* *sig.inp.90.1*.

The self-energies of e_g orbitals in *sig.inp.*.1* files are shown in Figure 11.

The self-energies of t_{2g} orbitals in *sig.inp.*.1* files are shown in Figure 12.

Figure 11: Self-energy v.s. number of DMFT loops in *sig.inp.*.1* for e_g orbitals

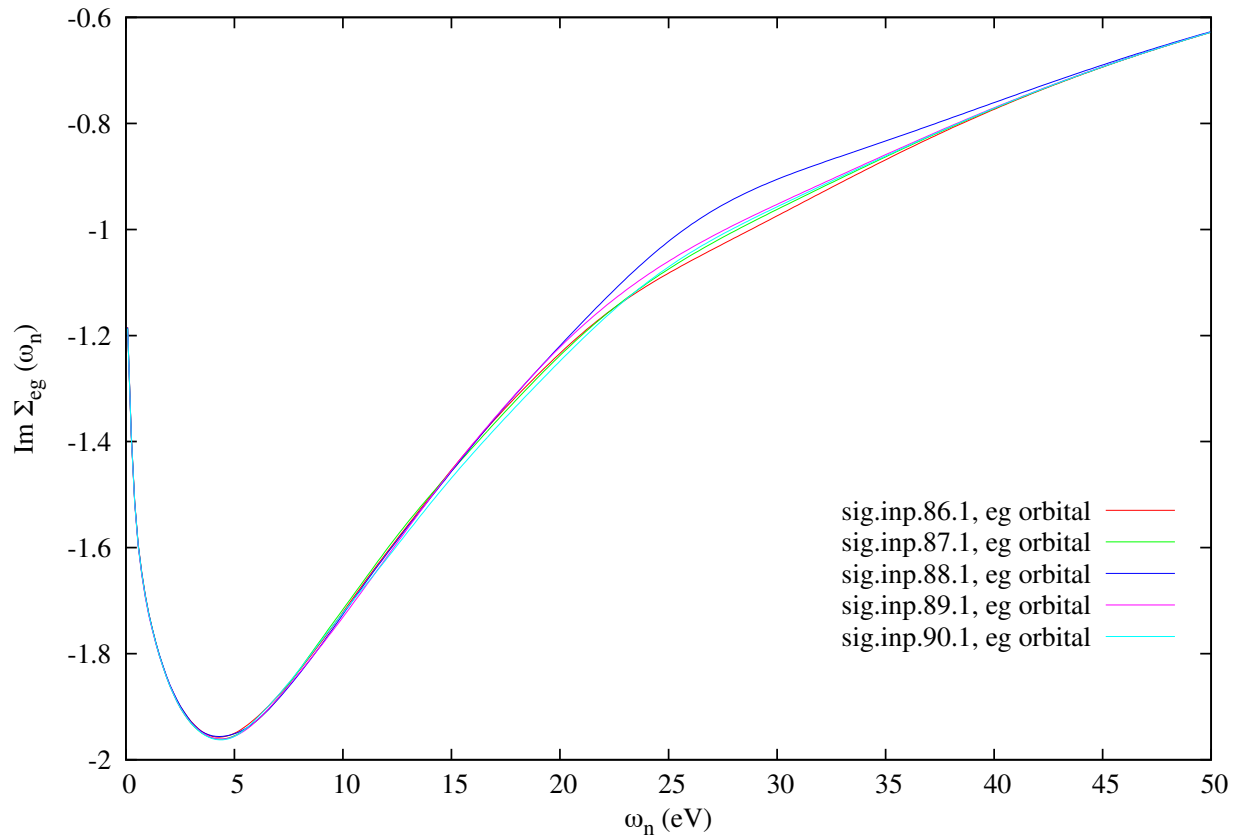
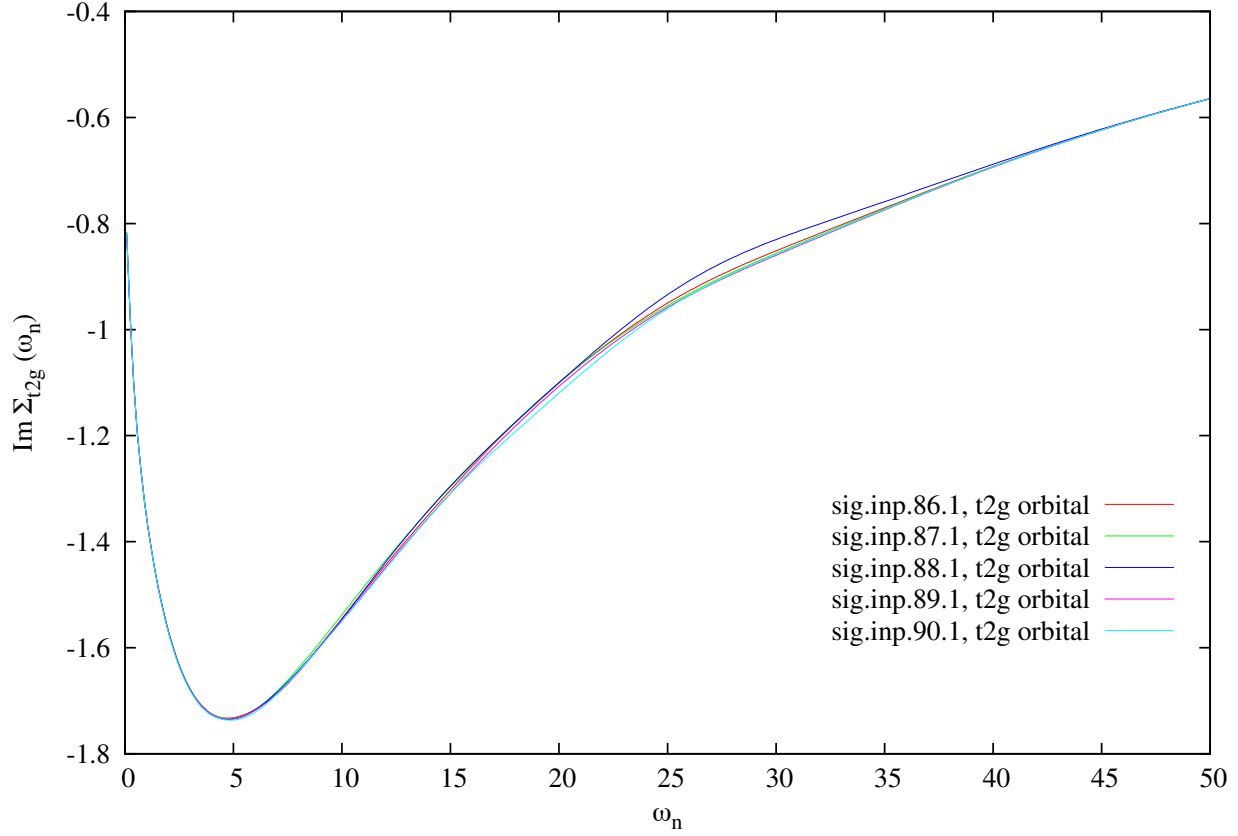


Figure 12: Self-energy v.s. number of DMFT loops in *sig.inp.*.1* for t_{2g} orbitals



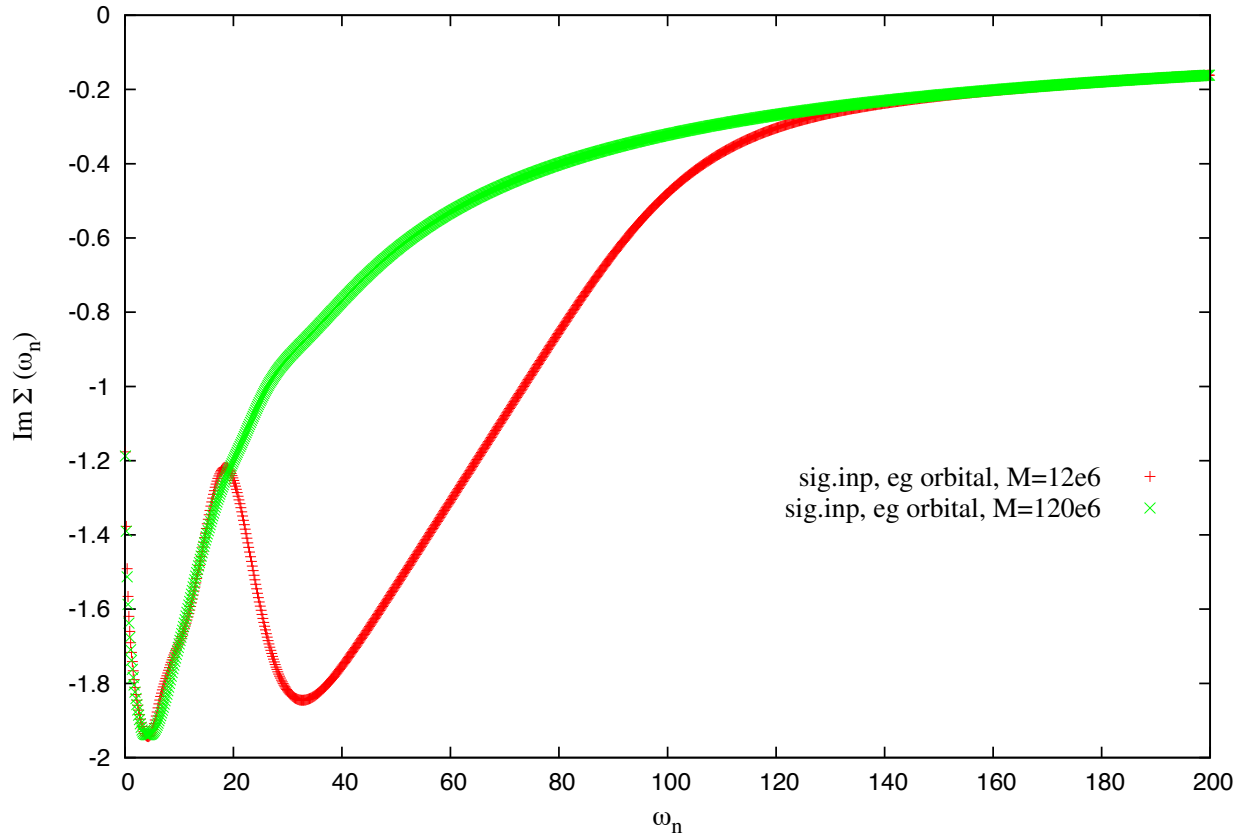
- Plot self-energies in *sig.inp* and *Sig.outb*

Alternatively, you can check the quality of self-energy convergence by plotting the self-energy results in the following files, *sig.inp* and *imp.0/Sig.outb*. As can be seen, unconditioned self-energy pattern in *Sig.outb* is quite scattered. There are two solutions to solve this problem, 1. Increase number of Monte Carlo cycle measurements by simply changing “M” value in the *params.dat* file. 2. Increase number of CPU cores by modifying *mpi* value in DMFT submit script.

- Plot conditioned self-energy in *sig.inp* before and after augmenting Monte Carlo numbers

Here illustrates conditioned self-energies before we increase the total Monte Carlo measurements (red line) and after the increase (green line) in *sig.inp* (see Figure 13)

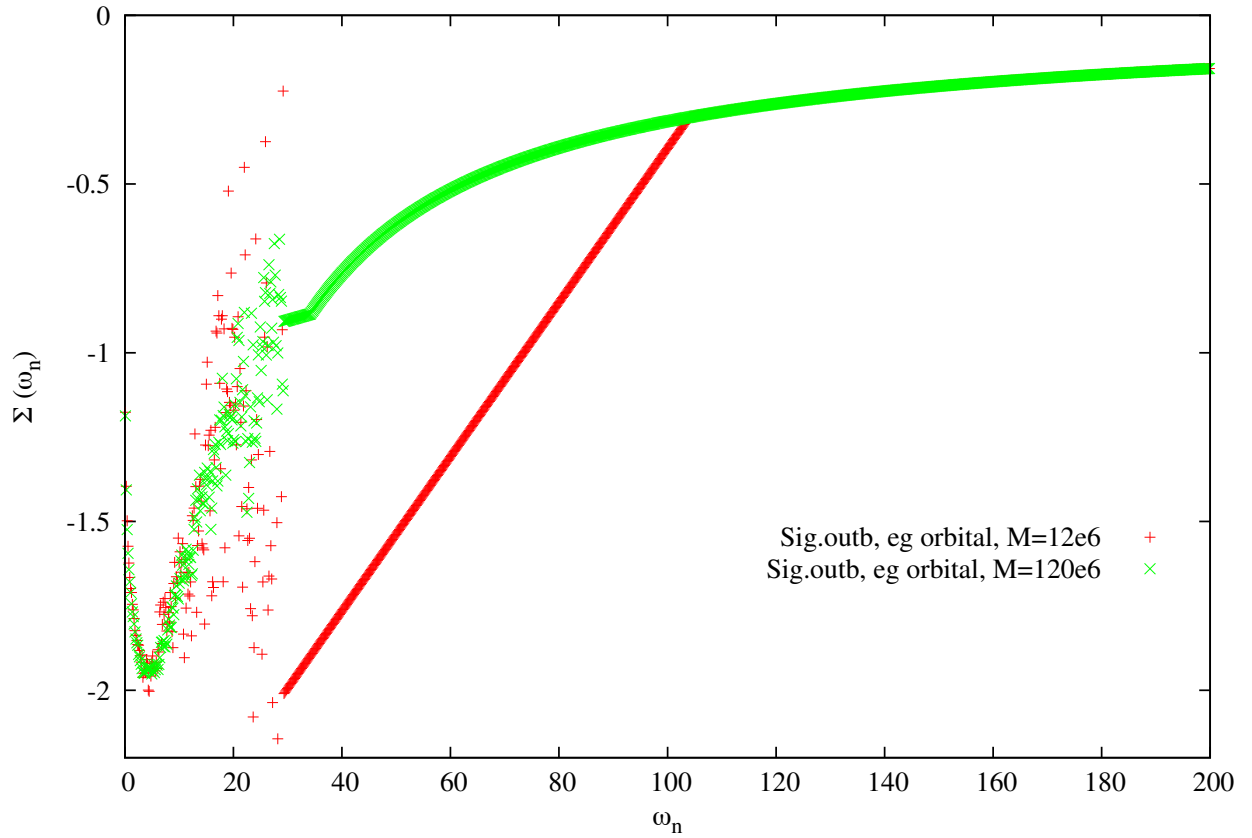
Figure 13: Conditioned self-energy v.s. number of DMFT loops in *sig.inp* before and after augmenting Monte Carlo measurements for e_g orbitals



- Plot unconditioned self-energy in *imp.0/Sig.outb* before and after augmenting Monte Carlo numbers

Here shows unconditioned self-energies before we increase the total Monte Carlo measurements (red line) and after the increase (green line) in *Sig.outb* (see Figure 14). It is obvious that the quality of unconditioned self-energy has been improved tremendously.

Figure 14: Unconditioned self-energy v.s. number of DMFT loops in *Sig.outb* before and after augmenting Monte Carlo measurements for e_g orbitals



– Conditioning of self-energy in *Sig.out*

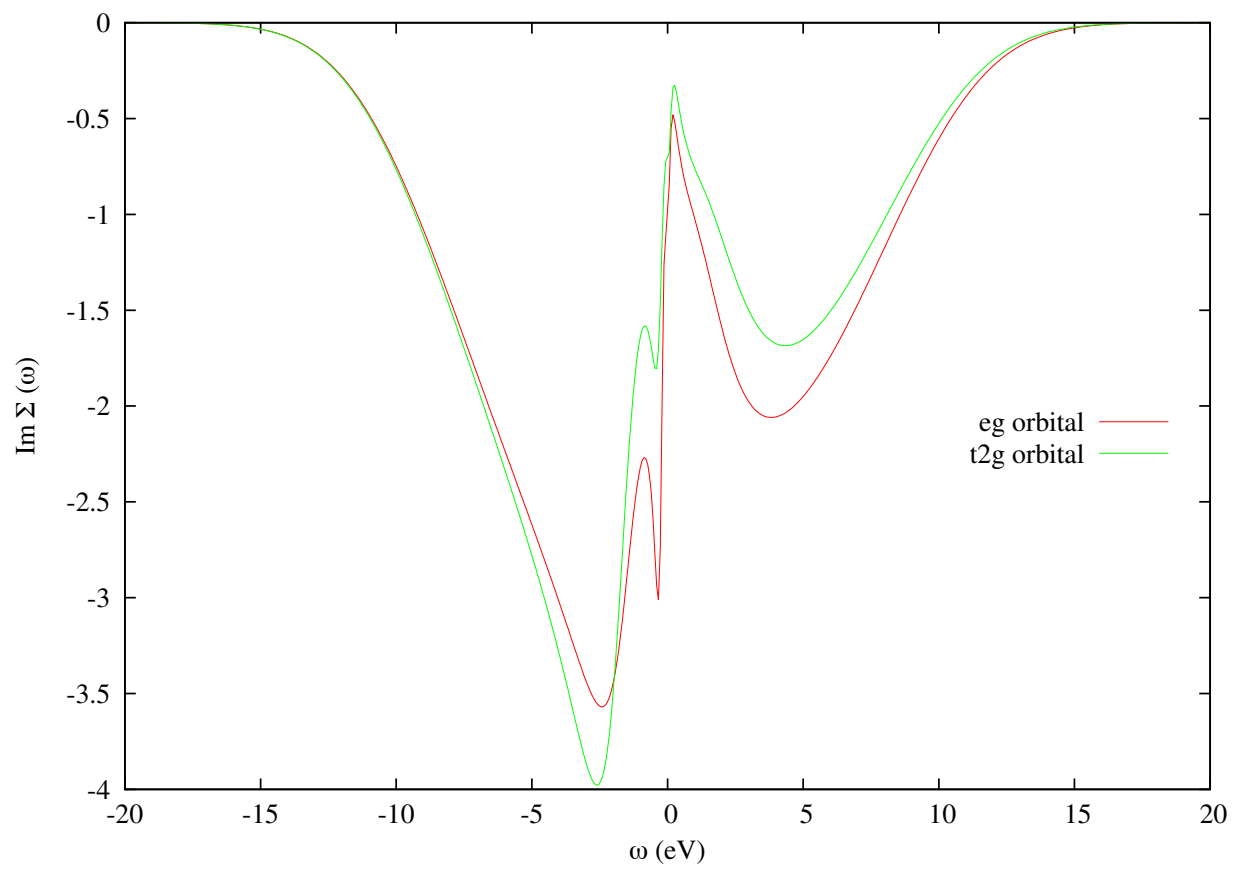
In DFT + DMFT, only the first “nom” Matsubara frequency points are directly from Quantum Monte Carlo (QMC) outputs using the Dysons’ calculated equations.

$$\Sigma(i\omega_n) = G^{o-1}(i\omega_n) - G^{-1}(i\omega_n)$$

Green’s function has a $\frac{1}{i\omega_n}$ asymptotic behavior and at the high frequency/energy region, $G(i\omega_n)$ and $G^o(i\omega_n)$ will be close to zero value. Due to the systematic error in QMC measurements, $\Sigma(i\omega_n)$ at high frequency will have a very large scattering, as can be seen in *imp.0/Sig.outb* at frequencies near $\omega_n = 65$. (see Figure 15).

Since we know the high frequency asymptotic behavior of self-energy, then the high frequency tail of self-energy at number of $\omega_n > \text{“nom”}$ is given by the high frequency asymptotic functional; and the self-energy points from [non-nom, nom] is used to fix the parameters of asymptotic functional.

Figure 15: Imaginary part of self-energy v.s. real frequency in Sig.out for e_g and t_{2g} orbitals



Part IV

Test cases

IV.1 FCC Fe at 1500 K 0 GPa

First, we want to test the DFT + DMFT calculations on a simple test case, FCC Fe at 1500 K 0 GPa, which has 1 atom per primitive cell and paramagnetic.

IV.1.1 Run DFT+DMFT codes

IV.1.1.1 Prepare struct-files

- Gather all the required information
 - The lattice parameters,
lengths a , b , c in Bohr or Ångstrom and angles α , β , γ .
 - The space group according to the “International Tables for Crystallography”,
<http://it.iucr.org/>
 - The positions of all inequivalent atoms in fractions of the unit cell.

- Manually set up the struct-files

Create a new *.txt* file by the command

“vi FCC_Fe”

in your new folder *FCC_Fe*. Next, type in all the prepared lattice information, (see Table 10)

Table 10: Lattice parameters in *FCC_Fe* text file

b						“a..Ang, b..Bohr”
0.0	0.0	0.0				“shift of origin”
8.1450	8.1450	8.1450	90	90	90	“a, b, c, angles α, β, γ ”
Fm-3m						“space-group symbol”
Fe						“atom- name”
0.0000000	0.0000000	0.0000000				“atomic position”

Exit editing and save the text file, then the *FCC_Fe* text file has been made.

Next task is to create your struct-file, *FCC_Fe.struct*, by the command

“cif2struct FCC_Fe”

in your current directory. You can further modify the title and the automatically calculated values in the established struct-file.

IV.1.1.2 Initialize calculations using `init_lapw`

FCC_Fe.struct file contains all the information needed for the crystal structure calculations that WIEN2k uses. Run the following command,

“init_lapw”

The following shows all the subsequent questions and corresponding default answers.

1. The screen first shows:

next is setrmt.

Automatic determination of RMTs. Please specify the desired RMT reduction compared to almost touching spheres.

Typically, for a single calculation just hit enter, for force minimization use 1-5; for volume effects you may need even larger reductions.

Press **“Enter”**

2. You will choose answers to a series of questions, typically you will enter the default setting values that WIEN2k uses as follows,

“o, a, 2, c, c, c, -up, 13, -6, c, 200, c, n”

which stands for the following meanings respectively:

o: use old scheme;

a: accept these radii;

2: nn-bondlength factor;

c: continue with sgroup;

c: continue with symmetry;

c: continue with lstart;

-up: switches for instgen.lapw -up(default);

13: SELECT XCPOT, 13:PBE-GGA (Perdew_Burke-Ernzerhof 96);

-6: SELECT ENERGY to separate core and valence states: -6.0 Ry (check how much core charge leaks out of MT-sphere);

c: continue with kgen;

200: number of k-points in whole cell, usually 200-500;

c: continue with dstart;

n: not perform a spinpolarized calculation.

IV.1.1.3 Run the self-consistency cycles (SCF) of DFT calculations

After *init_lapw*, there are many new files generated in the *FCC_Fe* folder. The next step is to run self-consistency cycles (SCF) of DFT calculations. It is strongly suggested to submit the DFT calculations job via the *submit_dft.scr* script utilizing services provided by a parallel computing machine (e.g., *mw* server). An example of *submit_dft.scr* is shown in the Part III. The command to submit the DMFT calculation job to *mw* server is as follows,

“qsub submit_dft.scr”

Another way to run SCF is to use the command **“run_lapw”**, but it is not recommended.

The self-consistency cycles (SCF) consist of the following parts:

LAPW0 (POTENTIAL) generates potential from density

LAPW1 (BANDS) calculates valence bands (eigenvalues and eigenvectors)

LAPW2 (RHO) computes valence densities from eigenvectors

LCORE computes core states and densities

MIXER mixes input and output densities

After SCF of DFT calculations, good charge densities have been converged.

You can save your results by the command **“save_lapw”**.

IV.1.1.4 Initialize DMFT calculations

Following *init_lapw* and SCF of DFT, proceed to the command

“init_dmft.py”

to invoke DMFT calculations in the current directory.

Suggestions on the choices thereafter are shown below,

- Specify correlated atoms (e.g. 1-4, 7, 8): **1**
- Type **“c”**
- For each atom, specify correlated orbital(s) (e.g. d, f): **d**
- Type **“c”**
- Specify qsplit for each correlated orbital (default = 0): **7** (cubic symmetry in real harmonics)
- Type **“c”**
- Do you want to group any of these orbitals into cluster-DMFT problems?
(y/n): **n**
- Enter the correlated problems forming each unique correlated problem, separated by spaces (e.g. 1, 3 2, 4 5-8) **1**
- Type **“c”**
- Broken symmetry run? (y/n): **n**

- Range (in eV) of hybridization taken into account in impurity problems;
default -6.0, 6.0: -10, 10: **-10, 10**
- Perform calculations on real; or imaginary axis? (r/i): **i**
- Is this a spin-orbit run? (y/n): **n**
- Generate blank sig.inp? (y/n): **y**

IV.1.1.5 Prepare all files

- Prepare input files (*FCC_Fe.indmfl*, *FCC_Fe.indmfi*) and output files

The initialization of DMFT calculations by *init_dmft.py* will generate two input files, *FCC_Fe.indmfi* and *FCC_Fe.indmfl*. These two files will connect the solid and impurity with DMFT equations.

Create a new folder that is in the parent directory,

“mkdir FCC_Fe_backup”

then move all the files from *FCC_Fe* folder into the *FCC_Fe.backup* folder by the command,

“mv ./* ../FCC_Fe_backup/”

then type in the following command in the current directory (*FCC_Fe/*),

“dmft_copy.py <parent directory>/FCC_Fe_backup/ -a”

-a: files for both DFT and DMFT will be copied.

This command makes a copy of all the output files of DFT calculation results and DMFT initialization results into the folder *FCC_Fe*. These files are needed for the following DMFT calculations. Examples of copied files are listed below, struct-file, WIEN2k files (*.in0*, *.in1*, *.in2*, *.inm*, *.inso*, *.inc*), charge density file (*.clmsum*), input files (*.indmfi*, *.indmfl*) and other files (*.klist*, *.kgen*, *.scf2*)

- Prepare *params.dat* and *sig.inp* files

We need to prepare two more files needed for DMFT calculations in the *FCC_Fe* folder, *params.dat* and *sig.inp*.

– *params.dat*

Manually set up the *params.dat* file or download the *params.dat* file from Haule’s W2K Database website,

http://hauleweb.rutgers.edu/database_w2k/

The *params.dat* file contains information about the program flow and impurity solver. Leave the *params.dat* file name as it is. In the *params.dat* file, you can change the following parameters,

* finish=

The number of charge self-consistent iterations in *params.dat* file, e.g., **finish=60** means there are 60 iterations in DFT+DMFT calculations.

* iparams0={"beta"}

“beta” is the inverse temperature. It is derived from the absolute temperature in Kelvin, $\beta = \frac{1(\text{eV})}{T(\text{K})}$ in which $1\text{eV} = 11604\text{K}$. For example, if absolute temperature T is 300 K, then $\beta = \frac{11604\text{K}}{300\text{K}} = 38.68$. **“beta”:**
38.68.

* iparams0={"M"}

“M” is the Total number of Monte Carlo steps. Typical M is 5 million (5e6), i.e., **“M”:** **5e6**. Total Monte Carlo measurements (N) is calculated by the equation, $N = (\text{Number of CPU}) * M$. (*Number of CPU*) could be changed in *mpi* value in the *submit_dmft.scr* file.

– *sig.inp*

sig.inp is a blank self-energy file, it is generated by the command

“szero.py -e 35.7 -T 0.026 -n 3000”

Make sure that the T value here (0.026) is the same as the *beta* value in *params.dat* file. The meanings of the characters (e , T , n) are as follows,

* e

Double counting energy, calculated by the equation

$$E_{DC} = U(n - \frac{1}{2}) - J/2(n - 1)$$

in which $U = 6$ eV; $J = 0.8$ eV

* T

Effective temperature in DMFT which should be consistent with *beta* in the *params.dat* file. For example, if we have *beta*=38.68333, then

$T = \frac{1}{\text{beta}} = \frac{1}{38.68333} = 0.026$. Please notice that T here is the effective temperature in DMFT derived from *beta* which is different from the absolute temperature.

* n

The number of Matsubara frequency points

IV.1.1.6 Submit the job and run DFT + DMFT iterations

Copy *submit_dmft.scr* script file into the current directory (*FCC.Fe*). An example of *submit_dmft.scr* file is shown in Part III. The final step is to type in the command,

“qsub submit_dmft.scr”

Using the remote powerful machine (mw) is handy in terms of computing a large number of Monte Carlo cycles due to a large number of CPU cores it has. The command above is going to create a *.dat* file for mpi parallel execution, *mpi_prefix.dat*.

IV.1.1.7 Monitor running of DFT + DMFT

During the run, we can monitor the information file and update parameters, if necessary. The files we can monitor are listed below,

- *info.iterate*

It contains information about chemical potential, impurity levels, exchange energies and total fillings, etc.

- *:log*

It helps monitor the modules that are running.

- *dmft_info.out, FCC_Fe.dayfile*

To monitor the DMFT calculations and convergence levels.

- *imp.0/nohup_imp.out, imp.0/Sig.out, imp.0/Gf.out*

To monitor the impurity levels.

- *dmft1_info.out, FCC_Fe.outputdmf1.0*

To monitor the dmft1 step.

- *dmft2_info.out, FCC_Fe.outputdmf2.0*

To monitor the dmft2 step.

IV.1.2 Properties based on DMFT calculations

Once the SCF cycle has converged one can calculate various properties, such as Probabilities, Density of States (DOS), Partial Density of States (Partial DOS), spectral functions, optics (including conductivity) and Fermi surface.

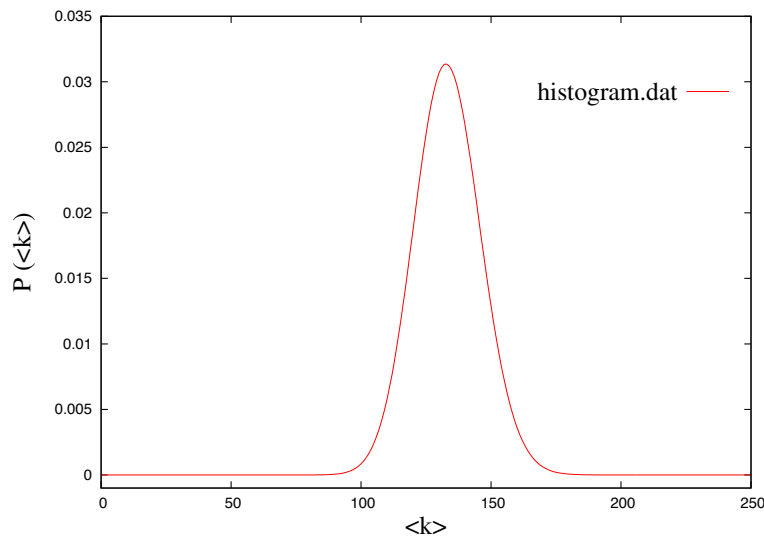
Let's continue with our simplest test case, *FCC Fe at 1500 K 0 GPa*, and calculate properties based on it.

IV.1.2.1 *histogram.dat* and *Probability.dat*

- *histogram.dat*

To check whether the parameter “Nmax” in the *params.dat* file is large enough, we should plot “imp.0/histogram.dat”, which should be gaussian, and should look like figure 16.

Figure 16: Plot *histogram.dat* file



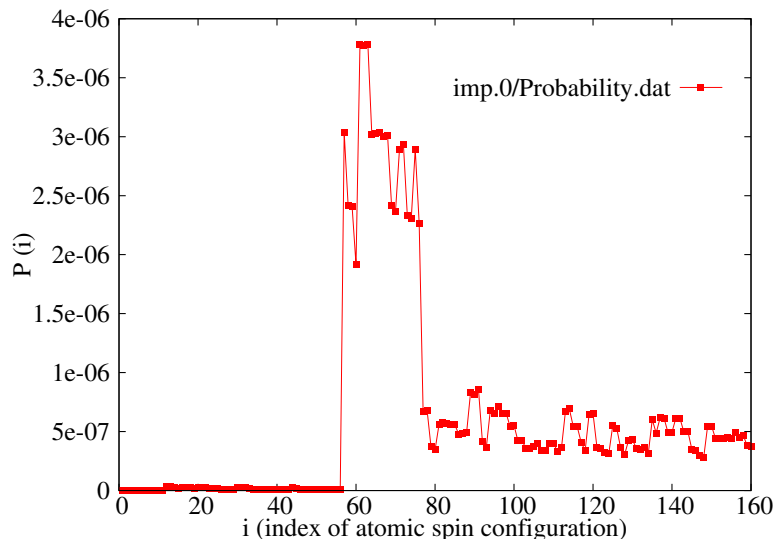
This shows the distribution of the perturbation order. The perturbation

order around 200 has almost no probability, hence $N_{\text{max}}=400$ (two kinks for each perturbation order) would be sufficient. We used substantially larger $N_{\text{max}}=600$, so that we could reduce the temperature for factor of 3. Notice that larger N_{max} only slightly slows down the execution of the code, and uses only a bit more memory, hence it is a good idea to use large N_{max} .

- *imp.0/Probability.dat*

We can also check what the probability for each atomic state is, i.e., probability that a Fe-d electron is in any of the atomic states. This information is written in “*imp.0/Probability.dat*”. The first and the second column correspond to the index of each spin configuration, as defined in “*actqmc.cix*” (which enumerates all atomic states). The third column is the probability for a state. Here is the plot of probability for first 170 states, see figure 17,

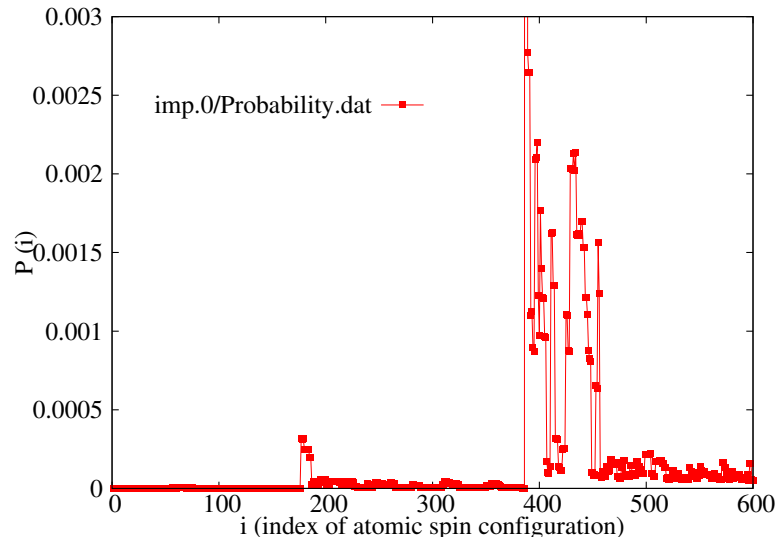
Figure 17: Plot *imp.0/Probability.dat* file



The first state corresponds to the empty 4f ion. The next 80 states (from

1...80) correspond to nominal occupancy $N=1$, and the values between 80 and 170 correspond to occupancy $N=2$. At $N=1$, there are two sets of probabilities, 6 take the value around $4e-6$, and 8 take the value of $2e-6$. The first and the second set corresponds to $j=5/2$ and $j=7/2$ multiplet, respectively. The next 90 states, which correspond to occupancy $N=2$, have probability around $5e-7$. They can be grouped into multiplets, the lowest being $J=4$, followed by $J=2$ and $J=5$. Notice that the lowest multiplet satisfies Hunds rules (largest $S=1$, largest $L=5$, and $J=L-S=4$). Notice also that although FCC Fe is a very itinerant metal, the projection to the atomic states still satisfies atomic rules, such as Hunds rules. The cumulative probability for $N=2$ states is around 0.086. Compare this to $N=1$ probability of 0.836, and $N=0$ probability of 0.077. The next 430 states correspond to $N=3$ occupancy, and their histogram looks like figure 18.

Figure 18: Plot *imp.0/Probability.dat* file 0-600



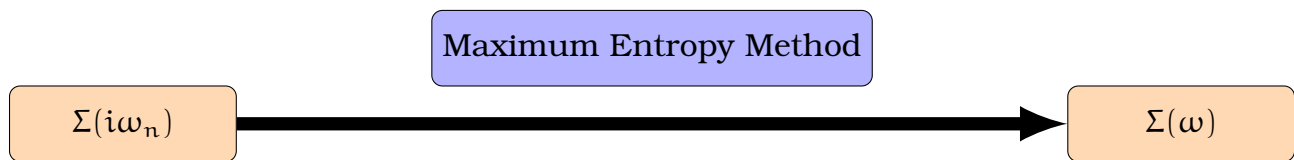
Typical probability of $N=3$ state is $1e-6$, and cumulative probability is 0.0012.

Clearly, the probability falls off very rapidly with particle number N , hence we can safely restrict occupancy in `params.dat` file to `nc=[0, 1, 2, 3]`.

IV.1.2.2 Density of States (DOS) and Partial Density of States (Partial DOS)

- Analytical continuation of self energies

We are going to do analytical continuation to calculate the self energy on the real axis after acquiring our results on the imaginary axis.



1. Preparation: make several new directories under the parent directory (*FCC_Fe*) before we do any calculations on DOS, **DOS/FCC_Fe**, **mem/Sig_real** and **mem/Gf_real**

2. Average the self-energies from the last few iterations to minimize the noise,

**saverage.py sig.inp.86.1 sig.inp.87.1 sig.inp.88.1 sig.inp.89.1
sig.inp.90.1 sig.inpx**

Here it averages the last five *sig.inp.*.1* files and creates an output file *sig.inpx* in the current directory *FCC_Fe*.

3. Copy the averaged self energy file *sig.inpx* into the new folder *mem/Sig_real*.

4. Download and modify *maxent_params.dat* file from Haule's website, <http://hauleweb.rutgers.edu/downloads/>.

Make sure 'Ntau' value in *maxent_params.dat* file matches *beta* value in *params.dat* file. Copy modified *maxent_params.dat* file into the same direc-

tory *mem/Sig_real*. This file contains the parameters needed for analytical continuation process.

5. Run the maximum entropy method (mem).

Now there are two files in *mem/Sig_real* folder, *sig.inpx* and *maxent_params.dat*. Next execute the command,

maxent_run.py sig.inpx

to produce the self energy file *Sig.out* on the real axis. Results in *Sig.out* file are plotted in Figure 15, 19.

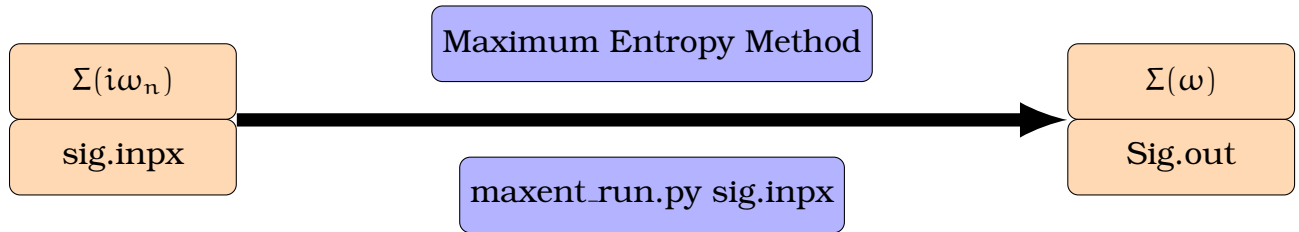
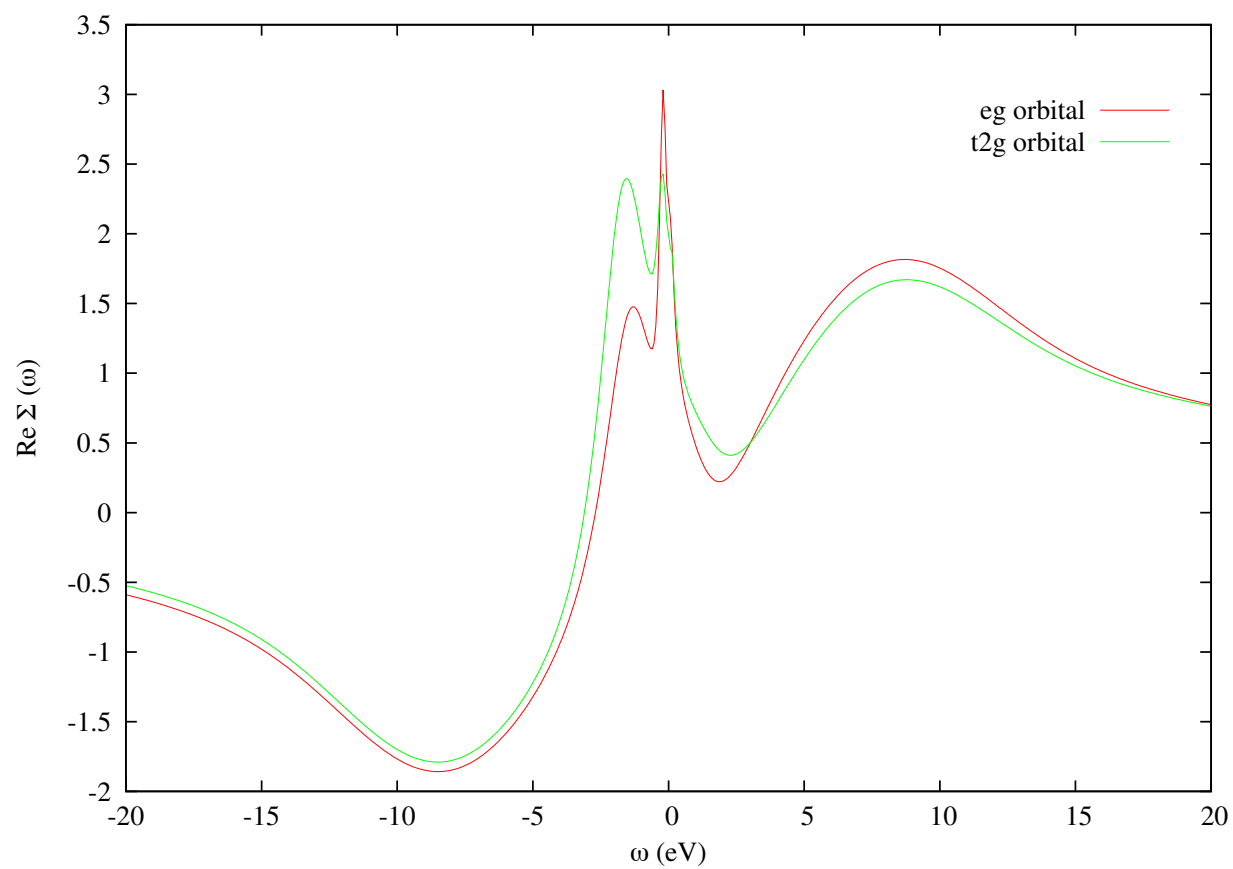


Figure 19: Real part of self-energy v.s. real frequency in Sig.out for e_g and t_{2g} orbitals



- Plot Total DOS

To plot Total Density of States (DOS), we need to run one more dmft1 calculation to obtain Green's function and DOS on the real axis.

1. Now we are going to work in the directory */FCC_Fe/DOS/FCC_Fe/*, first, copy all the DMFT output files,

```
dmft_copy.py ../../ -a
```

2. Copy the generated *Sig.out* file to *sig.inp* in current directory (*/DOS/FCC_Fe/*)

```
cp ../../mem/Sig_real/Sig.out sig.inp
```

3. Modify *FCC_Fe.indmfl* file by changing 'Matsubara' value on the second line from imaginary axis "1" to real axis "0".

4. Run **x kgen** and specify number of k-points in whole cell, **8000**

5. Run dmft1 calculations once,

```
x lapw0
```

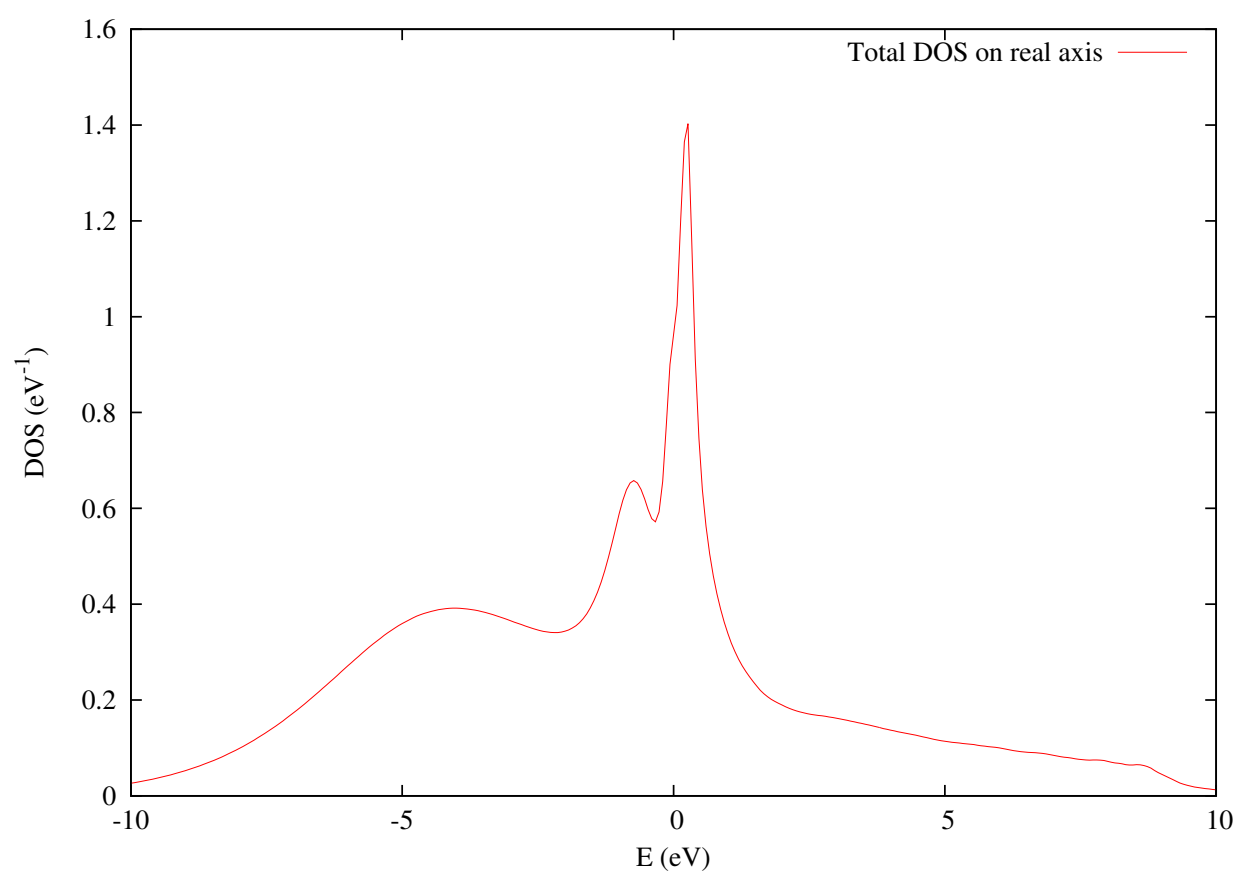
```
x lapw1
```

```
x dmft.py dmft1
```

6. Plot the Total Density of States (DOS) which is stored in the *FCC_Fe.cdos* file.

Results of Total DOS are shown in Figure 20.

Figure 20: Total Density of States on the real axis for FCC Fe



- Plot Partial DOS

Now we move on to calculate and plot the partial (d-orbital) Density of States (DOS).

- We are going to work in the new directory *FCC_Fe/mem/Gf_real*
- Copy *maxent_params.dat* and *Gf.out* files into *Gf_real* directory from directories, */FCC_Fe/mem/Sig_real/maxent_params.dat* and */FCC_Fe/imp.0/Gf.out* respectively.
- Run **maxent.S.py Gf.out** in the current *mem/Gf_real* directory.
- Plot 'Sig.out' file generated. Note that partial DOS is given by dividing the imaginary part of the Green's function by $-\pi$.

$$A(\omega) = -1/\pi * (\text{Im}G(\omega))$$

So the commands should resemble the following,

plot 'Sig.out' u 1:(-\$3/3.14) w l ":DOS of e_g d orbital"

plot 'Sig.out' u 1:(-\$5/3.14) w l ":DOS of t_{2g} d orbital"

- The results of all partial d orbital DOS are shown in Figure 21. The comparison between the partial DOS and total DOS is shown in Figure 22.

Figure 21: Partial Density of States (e_g and t_{2g}) on the real axis for FCC Fe

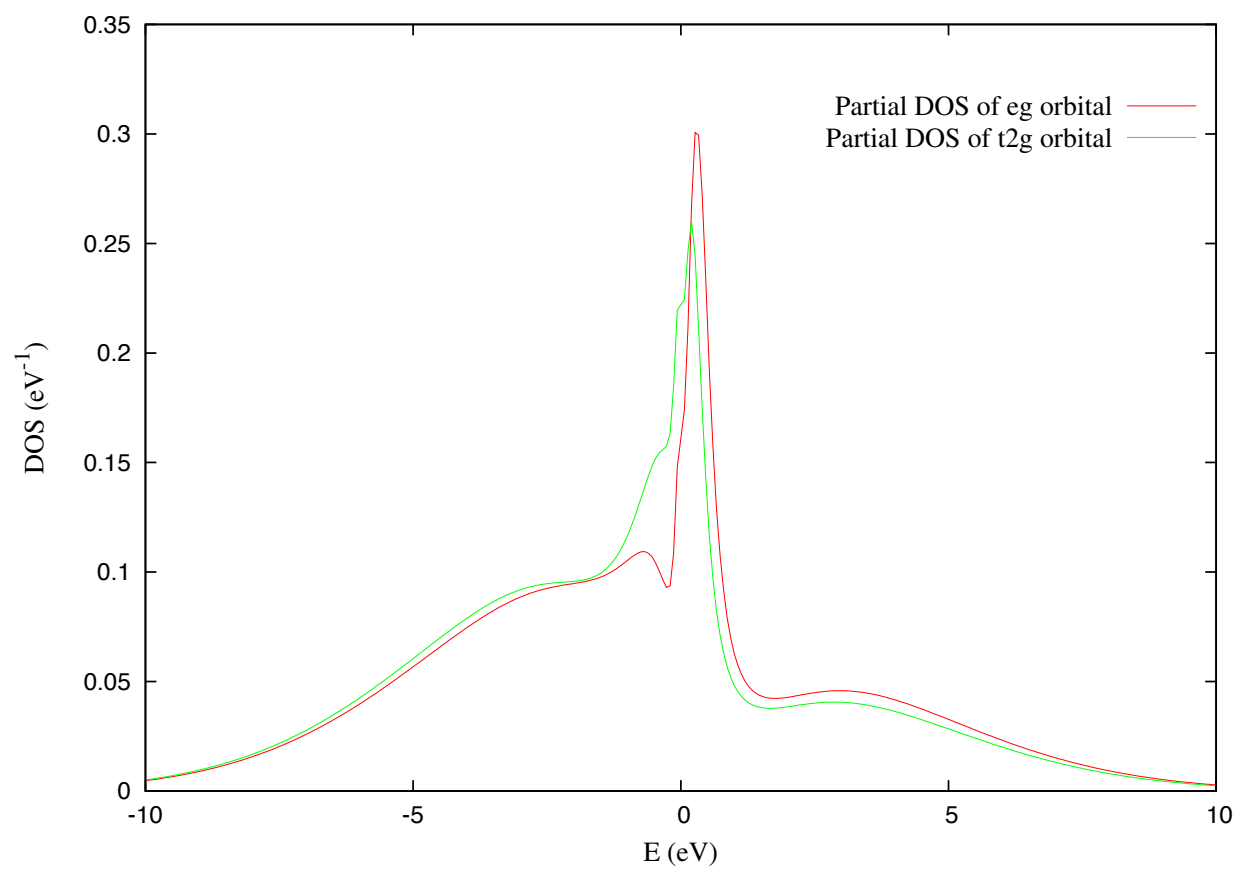
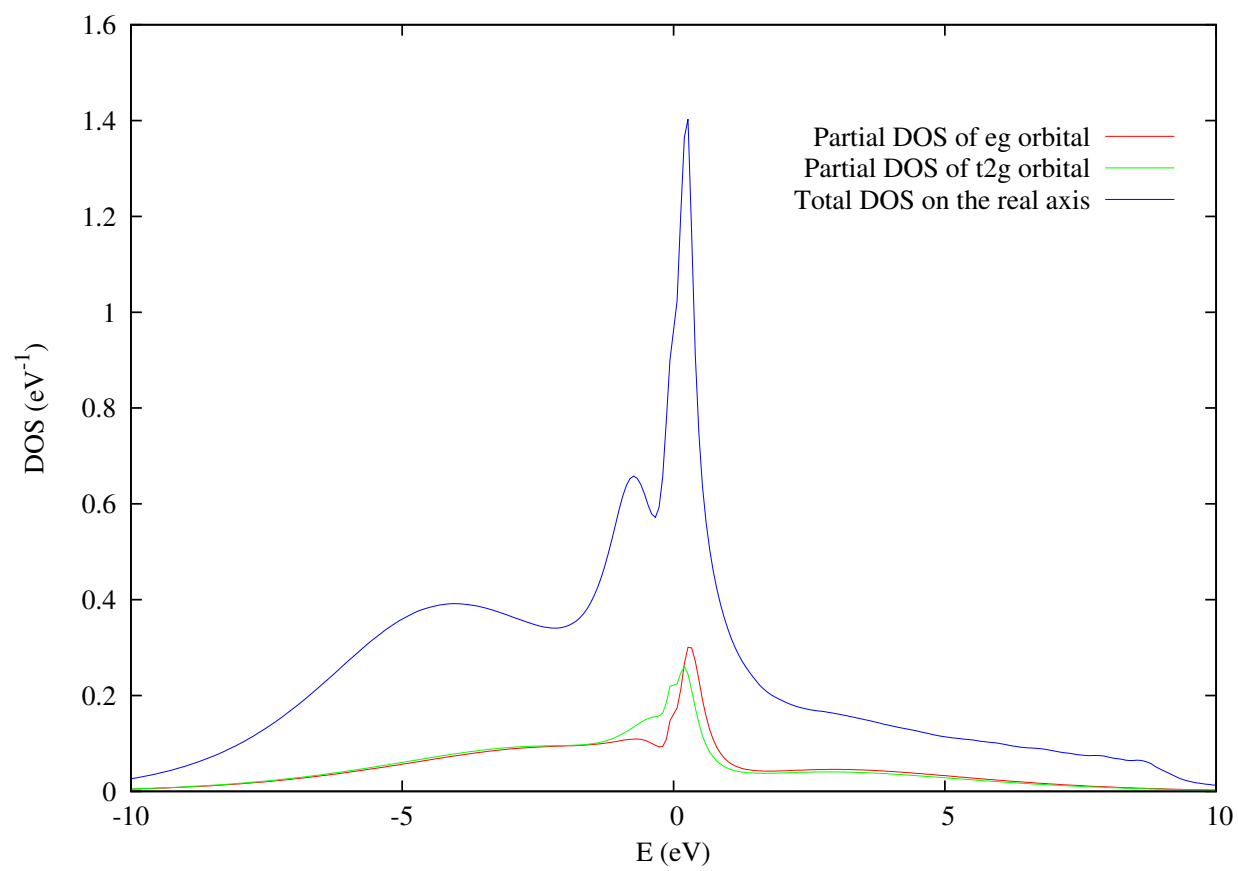


Figure 22: Partial Density of States and Total Density of States on the real axis
for FCC Fe



IV.1.2.3 Spectral functions

1. Preparation: Make a new folder containing all the results of calculations on Spectral functions (momentum resolved spectral function, equivalent to band structures in DFT).

```
mkdir -p /FCC_Fe/spectra/FCC_Fe
```

2. Copy all the required files from DMFT calculations to the current directory.

```
dmft_copy.py ../../ -a
```

Here make sure you have *EF.dat* file, if you don't, find the Fermi Energy and store the Fermi Energy to *EF.dat* file in eV.

3. Replace the imaginary self-energy by self-energy in real frequency.

```
cp /FCC_Fe/mem/Sig_real/Sig.out sig.inp
```

4. Edit *FCC_Fe.indmfl* file.

On the second line of *FCC_Fe.indmfl* file, you will see,

```
1 0.025 0.025 600 -3.0000000 1.0000000 # matsubara, broadening-corr,  
broadening-noncorr, nomega, omega_min, omega_max (in eV)
```

Change the “*matsubara*” value from **1** to **0** in order to switch from imaginary axis to real axis. Change the frequency range: “*omega_min*” from **-3** to **-5**; “*omega_max*” from **1** to **5**.

5. Create *FCC_Fe.klist_band* file to choose the path in momentum space (first brillouin zone).

Figure 23 illustrates the first Brillouin zone of a face centered cubic lattice, e.g. FCC Fe.

There are two ways to create this file. One is to write it by hand. The other method is to make a copy from WIEN2k prepared templates. Because FCC iron is the current system, we can simply copy *fcc.klist* from WIEN2k templates.

```
cp $WIENROOT/SRC_templates/fcc.klist FCC_Fe.klist band
```

6. Run the following commands to execute DMFT calculations on the mesh in the current */spectra/FCC_Fe/* directory.

```
x lapw0
```

```
x lapw1 -band
```

```
ssplit.py -i sig.inp
```

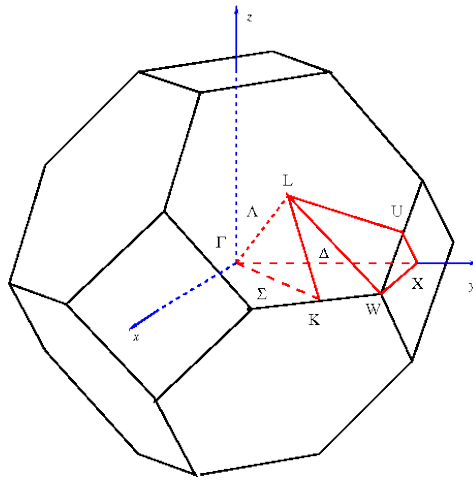
```
x dmft.py dmftp
```

7. Plot the band structures by,

```
wakplot.py
```

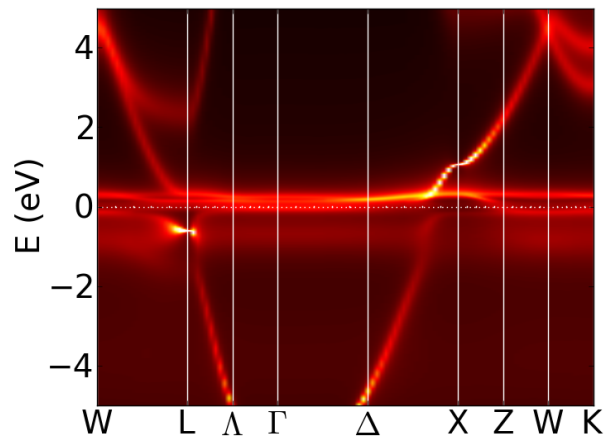
You can increase the intensity of brightness by reducing it from 0.2 (default) to any smaller number. Here shows how the plot looks like (see Figure 24),

Figure 23: First Brillouin Zone of FCC Fe lattice



Source: http://commons.wikimedia.org/wiki/File:Fcc_brillouin.png

Figure 24: Spectral functions of FCC Fe



IV.1.2.4 Optics including conductivity

1. Preparation: make a new folder in *FCC_Fe*,

```
mkdir -p FCC_Fe/optics/FCC_Fe
```

2. Copy all the required files from DMFT calculations to the current directory,

```
dmft_copy.py ../../ -a
```

3. Prepare *FCC_Fe.inop* file. Refer to WIEN2k userguide for example of *.inop* file. It should contain the following information.

```
99999 1 : NKMAX, NKFIRST
```

```
-5.0 5.0 : EMIN, EMAX, NBvalMAX
```

```
3 : number of choices (columns in *symmat)
```

```
1 : Re xx
```

```
2 : Re yy
```

```
3 : Re zz
```

```
ON : ON/OFF writes MME to unit 4
```

4. Prepare *dmftopt.in* file. Make sure “# dwindow” value is large enough to get better results. Below shows the contents of this file.

```
0 # updn: [0—up—dn] – 0 for non-magnetic calculation
```

```
Fe # case
```

```
0.01 # gamma – broadening of all bands / we recommend to keep nonzero  
value
```

0.0 # gammac – broadening of the correlated bands (in addition to the self-energy)

15 # ommax – maximum frequency for the optics calculation

1e-2 # delta – minimum separation of frequency for logarithmic mesh in frequency

8 # Nd – number of points in each linear mesh, a subset of logarithmic mesh

T # Qsym: [F—T]. Do we need to symmetrize? (over all irreducible only or over all.)

F # InterbandOnly [F—T] (F – all, T–interband)

20 # dwindow – Not all bands are used in computation of optics, but only those from [-omega-dwindow, omega+dwindow].

3 # Ndirection – How many optics type do you want to compute: xx, yy, zz,...

1.0 0.0 0.0 # alphaV(1,:)

0.0 0.0 0.0 # alphaV(2,:)

0.0 0.0 0.0 # alphaV(3,:)

0.0 0.0 0.0 # alphaV(1,:)

0.0 1.0 0.0 # alphaV(2,:)

0.0 0.0 0.0 # alphaV(3,:)

0.0 0.0 0.0 # alphaV(1,:)


```
0.0 0.0 0.0 # alphaV(2,:)
```

```
0.0 0.0 1.0 # alphaV(3,:)
```

5. Replace the imaginary self-energy by self-energy in real frequency.

```
cp /FCC_Fe/mem/Sig_real/Sig.out sig.inp
```

6. Edit *FCC_Fe.indmfl* file.

On the second line of *FCC_Fe.indmfl* file, Change the “*matsubara*” value from **1** to **0** in order to switch from imaginary axis to real axis.

7. Run **x kgen** and specify number of k-points in whole cell, **8000**
8. Run the following commands to execute DMFT calculations in the */optics/FCC_Fe/* directory.

```
x lapw0
```

```
x lapw1
```

```
x optic
```

```
cp FCC_Fe.mommat2 FCC_Fe.mommat
```

```
ssplit.py -i sig.inp
```

```
x_dmft.py dmftu
```

```
dmftopt
```

9. Plot the results stored in files, *optics.dat* and *optdos.dat*. See Figures 25, 26.

Figure 25: Optical conductivity of FCC Fe

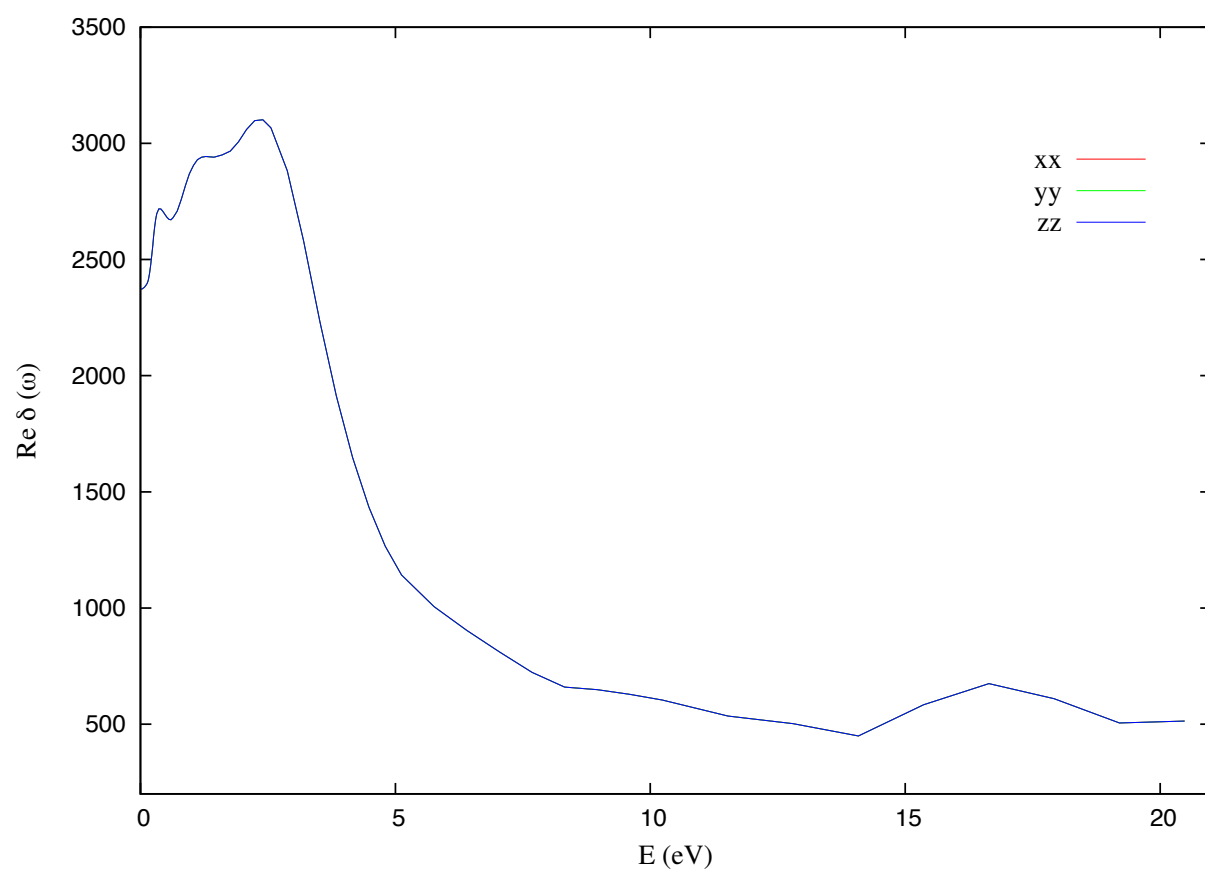
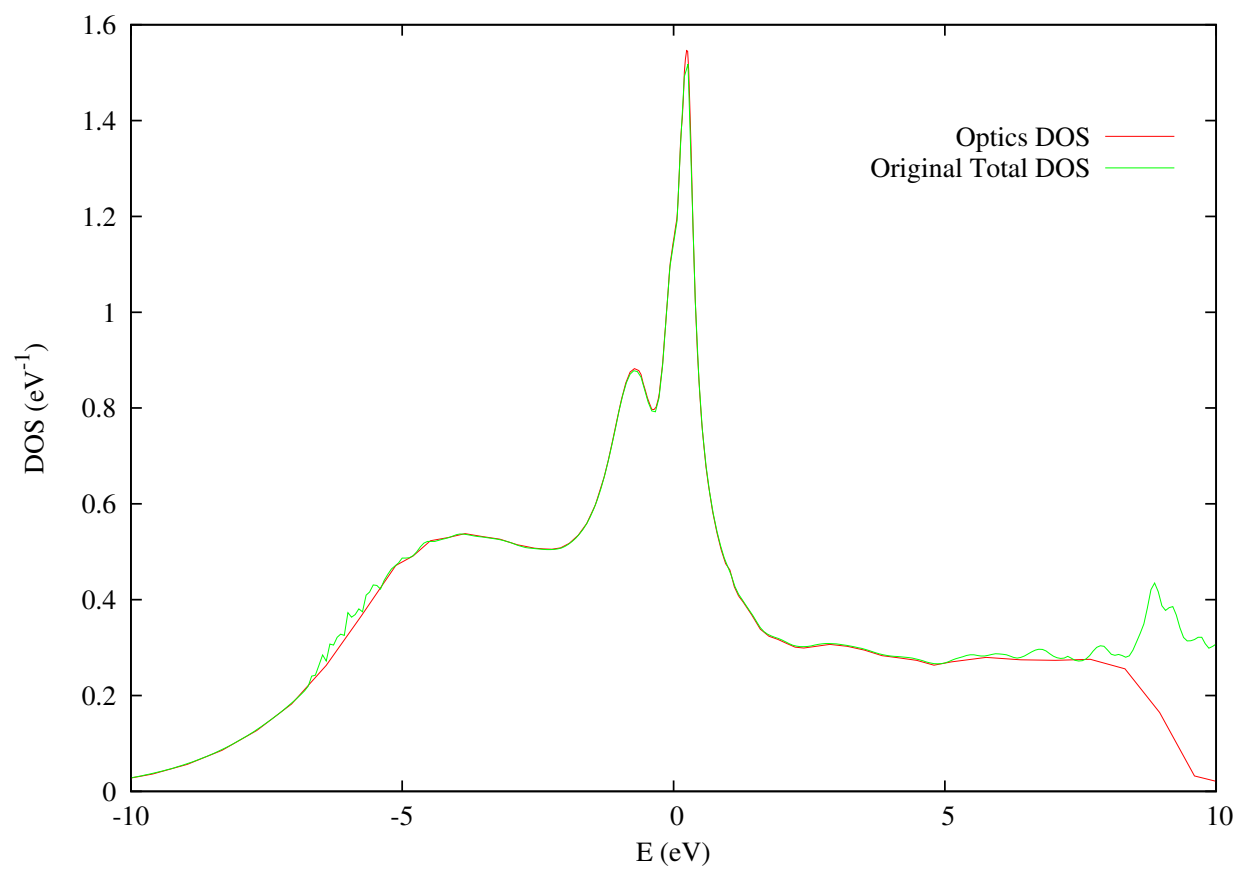


Figure 26: Density of States based on optics calculations and comparison
with Total DOS for FCC Fe



IV.1.2.5 Fermi Surface

References

P Blaha, K Schwarz, and GKH Madsen. WIEN2K, An Augmented Plane Wave+ Local Orbitals Program for Calculating Crystal Properties (TU Wien, Austria, 2001). ISBN 3-9501031-1-2, page 2001, 2001. ISSN 39501311. doi: citeulike-article-id:6205108. 2

Kristjan Haule, Chuck-Hou Yee, and Kyoo Kim. Dynamical mean-field theory within the full-potential methods: Electronic structure of CeIrIn_5 , CeCoIn_5 , and CeRhIn_5 , 2010. ISSN 1098-0121. 2, II

G Kotliar, S Y Savrasov, K Haule, V S Oudovenko, O Parcollet, and C A Marianetti. Electronic structure calculations with dynamical mean-field theory. *Rev. Mod. Phys.*, 78:865–951, 2006. ISSN 0034-6861. doi: 10.1103/RevModPhys.78.865. 2, I.1, I.2.2, I.3.1

Gabriel Kotliar and Dieter Vollhardt. Strongly Correlated Materials: Insights From Dynamical Mean-Field Theory. *Physics Today*, 57:53, 2004. ISSN 00319228. doi: 10.1063/1.1712502. URL <http://link.aip.org/link/PHTOAD/v57/i3/p53/s1\&Agg=doi>. 2, I.3.1