AMULET

Advanced Materials simULation Ekaterinburg's Toolbox

Manual

Version 12.01.19

September 5, 2019

Contents

1 Introduction

$\mathcal{AMULET}$ is a suite of computer codes for a first principles electronic structure calculations of strongly correlated materials. It is based on density functional theory (DFT) combined with dynamical mean-field theory (DMFT). The later is the best local approximation for a solution of strongly correlated problem and allows one to describe properly spin, orbital and charge degrees of freedom. Being a self-consistent time-dependent theory, DMFT treats on equal footing different energy scales that makes it applicable for an investigation of an entire phase diagram. The density functional theory is used to obtain a material specific part of the DFT+DMFT problem. The main code can perform the DFT+DMFT calculations with use of different solvers for an effective quantum impurity model. Different magnetic, electronic and structural properties can be evaluated in paramagnetic or magnetically ordered phases. The calculations of properties of chemically disordered compounds and alloys are possible within CPA+DMFT formalism. Post-processing tools include calculations of ARPES spectra, magnetic susceptibility, internal energy, etc. At present, the code integrated with Quantum Espresso, ELK-code and Stuttgart's TB-LMTO.

Short highlights of the $\mathcal{AMULET}$'s features:

- Segment version of the continuous time quantum Monte-Carlo method permits efficient calculations of properties down to low temperatures.
- Our implementation of classical Hirsh-Fye quantum Monte-Carlo is able to evaluate off-diagonal elements of the Green function. Therefore, this solver can be utilized for cluster DMFT calculations.
- Exact diagonalization (experimental) uses full rotationally invariant Coulomb interaction matrix.
- Different type of correlated impurities can be set up simultaneously for complicated compounds (for example, d and f orbitals at the same time).
- Calculations can be performed in paramagnetic or magnetically ordered regimes.
- Any kind of magnetic ordering can be specified.
- Calculation of magnetic susceptibilities.
- Evaluation of the $\mathbf{k}$-resolved spectral functions, $A(\mathbf{k}, \omega)$.
- CPA+DMFT calculations for materials with a substitutional disorder.
- Calculation of DFT+DMFT internal energy.
- DFT+DMFT calculations with spin-orbital interaction (experimental).
- MPI parallelized.
- Simple input format allows one to use almost any band structure code that is able to construct the Hamiltonian in a localized basis set (Wannier like).

2 Conventions

Physical units

$\mathcal{AMULET}$ uses next conventions for physical units. The atomic unit is used for the length, 1 a.u. = 0.52917720859(36) Å. Electron-volt is used as an energy unit, 1 Ry = 13.605692 eV.

Notational conventions

This manual uses the following conventions. Filenames are in ***brick-red bold italic font***. All files with an ***.ini*** extension are input files. Files with a ***.dat*** extension are output. Many files are named as ***Filename_N_x,y.dat***, where N is for impurity number (see section 5) and x, y denote orbital indexes (for more details see section 7). Tokens are in *dark-blue skewed font*. Executables are in ***dark-green bold italic font***.

3 License

The $\mathcal{AMULET}$ is a free software for scientific and/or educational purposes and it is distributed under FreeBSD License (see below). To request a source code, a brief description of a planned research has to be submitted to developers at *code.amulet@gmail.com*. The source code will be e-mailed to you if it is suitable for your study.

4 Installation

Installation of the $\mathcal{AMULET}$ package is very easy. Once you have a source code, one needs to create a directory where you will build the code and run there cmake.

```
==>  mkdir Build_AMULET
==>  cd Build_AMULET
==>  CC=mpicc CXX=mpicxx FC=mpif90 cmake ../AMULET
==>  make
```

In most cases this will detect all necessary compilers and dependencies automatically.

Compilers

The $\mathcal{AMULET}$ package requires Fortran90 and C/C++ compilers. At present, the code can be correctly compiled with the Intel Fortran/C/C++ Compiler or GFortran/GCC compilers. The other compilers can also be used.

Required dependencies

All required dependencies listed below are usually installed on a system or can be easily added via a package manager.

Linear algebra package

The classical library for the linear algebra manipulations is used in the code. One can use LAPACK with standard C language APIs or its MKL version. It is important to note that standard C language APIs for LAPACK has to be installed on your machine (LAPACKE). The other vendor's versions of LAPACK (like ESSL) might work as well but they are not tested.

Boost

The part of the package written on C/C++ uses the boost library headers. They can be downloaded either from the boost website or Boost library can be found directly at this link. You don't need to install these libraries, just place it somewhere in your home directory.

5 Input files

In order to start calculations with the $\mathcal{AMULET}$ package several input files are required. For a minimal setup you will need ***amulet.ini***, ***impurity_1.ini*** and ***hamilt.am*** files. Also, there are optional complementary input files to setup/simplify different features. Below one will find out descriptions of all input files.

amulet.ini file

This is a main input file where most of the options controlling an execution process are located. In the example one can see that it is in a free format form. The order of variables in the file is not important. Comments (! or #) can occur at any place.

```
beta = 50
 L = 500
ecut = 5000

mu = 0
ntotal = -1     !   Negative number of particles fixes chemical potential
nspin = 1

  !   Just a comment

istart = 0
niter = 10
alpha = 0.9
```

Energy grid options

TineV (real, optional, default=0.1)
temperature, $k_BT = 1/\beta$ (in eV). If this token is defined, then *TinK* and *beta* tokens will be ignored.

TinK (real, optional, default=1160)
temperature, k_BT (in K). If this token is defined, then *beta* token will be ignored.

beta (real, optional, default=10)
an inverse temperature, $\beta = 1/k_BT$ (in eV^{-1}). It defines an imaginary time interval $[0, \beta]$ for the quantum Monte Carlo simulations.

L (integer, optional, default=50)
number of time points (slices) for an imaginary time grid.

ecut (real,optional, default=628)
cutoff energy for the Matsubara grid, $\omega_n = (2n-1)\pi/\beta$. The largest Matsubara frequency is *Iwmax* = (*ecut*$\times\beta/\pi$+1)/2. If this token is defined, then the *Iwmax* token will be ignored.

Iwmax (integer, optional, default=1000)
number of energy points for the imaginary (Matsubara) or real energy grid. The later is defined on the interval [*emin*, *emax*].

emin (real, optional, default=-20)
a minimal energy for the real energy grid.

emax (real, optional, default=20)
a maximal energy for the real energy grid.

eta (real, optional, default=π/β)
small shift to an imaginary plain for the real energy grid.

mu (real, optional, default=0)
chemical potential, μ. The behavior of this token depends on the sign of the *ntotal*.

ntotal > 0 – chemical potential is adjusted to the value of *ntotal* over the self-consistent DFT+DMFT loop. The μ value, provided in the ***amulet.ini*** file, is used as a starting guess on the first iteration.

ntotal < 0 – chemical potential is fixed over the self-consistent DFT+DMFT loop.

Compound description options

ntotal (real, required)
total number of particles. If this quantity is positive then the chemical potential, μ, is adjusted, otherwise the chemical potential is fixed.

spinpol (logical, optional, default=no)
defines magnetic type of calculation.

***spinpol* = no** – paramagnetic calculation with spin averaged directions (*nspin*=1).

***spinpol* = yes** – spin directions are not averaged that allows one for magnetic type of solution (*nspin*=2). One should keep in mind that at high temperatures, the DMFT solution can be paramagnetic even if *spinpol* = yes.

n_imp_type (integer, optional, default=1)
number of different types of impurities. If *n_imp_type* > 1, then the corresponding number of ***impurity_N.ini*** files has to be present.

H_field (real, optional, default=0)
global magnetic field, B, which will be added to the Hamiltonian for different spin projections: $H_\sigma \to H_\sigma + \sigma B$ ($\sigma = \pm 1/2$). It has to be in the same energy units as the Hamiltonian (eV).

rhtm (character, optional, default=undefined)
order of the real harmonics in the Hamiltonian.

***rhtm* = vasp** – use VASP, ELK or Exciting-plus order for the real harmonics $(d_{xy}, d_{yz}, d_{z^2}, d_{xz}, d_{x^2-y^2})$.

***rhtm* = espresso** – use Quantum Espresso order for the real harmonics $(d_{z^2}, d_{xz}, d_{yz}, d_{x^2-y^2}, d_{xy})$.

See also the token *cubic_harmonic_order* in the ***impurity.ini*** file, which overwrites the behaviour of *rhtm*.

soc (logical, optional, default=no)
calculations with a spin-orbital Hamiltonian.

***soc* = no** – no spin-orbit coupling in the Hamiltonian.

***soc* = yes** – spin-orbit is included to the Hamiltonian. This regime is valid for CT-QMC-W and Hubbard I solvers only.

For a better understanding of the Hamiltonian format see ***hamilt.am*** file section.

cpa (logical, optional, default=no)
coherent potential approximation (CPA) calculations.

Controlling options

name4hamiltonian (character(128), optional)
the Hamiltonian filename. It can be up to 128 characters, and for example, can refer to an absolute path of file. It is supposed to be in a tag-based format described in the Hamiltonian section.

ignorelockfile (logical, optional, default=no)
At the beginning of the execution, the ***amulet.mpi*** program checks existence of the ***lock.amulet*** file in the working directory. Execution terminates if this file exists. This prevents data loss due to overwriting. Such behaviour can be changed by setting *ignorelockfile* = yes.

verbos (integer, optional, default=3) – verbosity level that determines which output files will be printed.

verbos $\geq$ 10 – ***Green_N_x,y.dat***, ***Delta_tau_N_x,y.dat***, ***G0tau_N_x,y.dat***;
verbos $\geq$ 15 – ***Delta_N_x,y.dat***;
verbos $\geq$ 20 – ***Green_imp_N_x,y.dat***, ***Green0_N_x,y.dat***, ***Delta_fit_N_x,y.dat***, ***G_cpa_N_x,y.dat***, ***Sigma_cpa_N_x,y.dat***;
verbos $\geq$ 25 – ***Delta_imp_N_x,y.dat***, ***Green0_imp_N_x,y.dat***, ***SigmaG_N_x,y.dat***, ***Gtau_actime_N_x,y.dat***;
verbos $\geq$ 30 – CT-QMC log files.

use_atm (logical, optional, default=.false.) – use analytical tetrahedron method to calculate Green function.

use_blockinverse (logical, optional, default=.false.) – use block inverse for Green's function calculation.

fentoler (real, optional, default=10^{-3}) – energy tolerance for a Fermi level search.

fpatoler (real, optional, default=10^{-6}) – total number of particles tolerance for a Fermi level search.

ntail (integer, optional) – number of point in function tail to use for different high-energy fits.

ecut4fit (real, optional, default=10) – energy cutoff for a Lorentzian fit.

n_lor (integer, optional, default=101) – number of Lorentzians to use for a function fit.

ecut_prn (real, optional, default=20) – energy cutoff parameter for printing *.dat files. All functions in the *.dat files will not be printed above this value, which depends on *verbos*.
$verbos < 10 \Longrightarrow ecut\_prn = 20$
$verbos < 20 \Longrightarrow ecut\_prn = 50$
$verbos < 30 \Longrightarrow ecut\_prn = 100$
$verbos \geq 30 \Longrightarrow ecut\_prn = ecut$

Self-consistency options

istart (integer, optional, default=1) – how to make LDA+DMFT iterations to self-consistency.
istart = 0 – start from scratch.
istart = 1 – continue from saved run. If saved run is absent then *istart*=0 .
istart = 2 – after self-consistency make one run on the real energy grid (valid for ED only).

niter (integer, optional, default=10) – number of LDA+DMFT iterations.

alpha (real, optional, default=0.5) – mixing coefficient for self-energy ($\Sigma = \alpha\Sigma_{new} + (1-\alpha)\Sigma_{old}$).

Additional parameters for tools

use_sigma (logical, optional, default=.true.) – explicitly tells to routine to use or not a self-energy.

impurity_N.ini file

In the dynamical mean field theory a complicated lattice problem is replaced by an effective quantum impurity embedded in a self-consistent bath. In an ***impurity_N.ini*** file an information about the effective quantum impurity is stored. Depending on a compound of interest different effective quantum impurities are possible. Number of symmetry inequivalent impurities is defined in the ***amulet.ini*** file with a token *n_imp_type*. ***N*** in the ***impurity_N.ini*** filename is an integer from 1 to *n_imp_type* and proper amount of ***impurity_N.ini*** files has to exist. For the simplest calculations (like Bethe lattice) one should have ***impurity_1.ini*** file only. The file format is free. For a line continuation a backslash or ampersand (\ or &) can be used.

Impurity description

name (character, optional, default=IMP1)
impurity label which is used for output purposes.

nlm (integer, optional, default=1)
number of orbitals on impurity or orbital degeneracy. To some extent it can be regarded as "$2l+1$", where l is an angular momentum quantum number. *nlm* can be even number (e.g. for e_g orbitals), and therefore, it is more convenient to use it in such form instead of the angular momentum quantum number, l.

cubic_harmonic_order (integer array of size *nlm*, optional, default=-1)
defines order of the cubic harmonics. If this token is present, it overwrites a behavior of the token *rhtm* from the ***amulet.ini*** file. The numbering of the cubic (real) harmonics is listed in the Table 1. Therefore, if one wants to use the Quantum Espresso order for d orbitals ($d_{z^2}, d_{xz}, d_{yz}, d_{x^2-y^2}, d_{xy}$) one needs to set *cubic_harmonic_order* = 7 8 6 9 5.

Table 1: Numbering of the cubic harmonics.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
s	p_y	p_z	p_x	d_{xy}	d_{yz}	d_{z^2}	d_{xz}	$d_{x^2-y^2}$	$f_{y(3x^2-y^2)}$	f_{xyz}	f_{yz^2}	f_{z^3}	f_{xz^2}	$f_{z(x^2-y^2)}$	$f_{x(x^2-3y^2)}$

n_imp (integer, optional, default=1)
number of symmetry equivalent impurities of this type.

himppos (integer array of size *n_imp*, required)
positions of the equivalent impurities in the Hamiltonian.

magtype (integer array of size *n_imp*, optional, default=1)
defines type of magnetic ordering (ferromagnetic by default). This token implies that *spin-pol*=yes. In case of few equivalent impurities of this type, *n_imp*>1, the antiferromagnetic order can be specified. In the example below, the first and fourth equivalent impurities have opposite spin projection to the second and third.

```
...
  n_imp = 4
  magtype = 1 -1 -1 1
```

```
...
```

h_field_local (real, optional, default=0)
local magnetic field oriented according to *magtype*.

smask (integer matrix of size *nlm*, optional)
sets orbital symmetry. By default there is no orbital symmetry and all elements of *smask* matrix are different. If *smask(i,j)=0*, then the corresponding (i, j) element of the function (Green function, self-energy, etc.) will be nullified. Below is an example from the ***impurity_N.ini*** file showing how to set $t_{2g} - e_g$ symmetry for the d-type correlated impurity.

```
...
  smask = 1 0 0 0 0    \            !   1  ->  t2g
          0 1 0 0 0    \            !   2  ->  eg
          0 0 2 0 0    \
          0 0 0 1 0    \
          0 0 0 0 2

...
```

cpoc (real matrix of size *nlm*×*nlm*×*nspin*, optional)
defines an orbital and spin dependent potential that will be added to a local on-site part of the quantum impurity Hamiltonian.

cpoc_only (logical, optional, default=.false.)
fixes a local on-site part of the quantum impurity Hamiltonian to the value of *cpoc*.

Vcpa (real matrix of size *nlm*×*nlm*×*nspin*, optional)
defines an orbital and spin dependent on-site potential for CPA.

Setting up interaction and double counting

The matrix of the Coulomb interaction, $\langle m, m'|V_{ee}|m'', m'''\rangle$, can be defined in different ways. The most general way to setup the Coulomb interaction is to use a ***umtrx_N.ini*** file (for details of format you should read section 5). Below the tokens, which can be specified in the ***impurity_N.ini***, are listed in a descending priority order. If two definitions are used at the same time, those with lower priority will be ignored. At least one of the definition must be present for a proper definition of the interaction matrix and code execution.

ummss (real matrix of size 2×*nlm*, optional)
matrix of the Coulomb interaction in the spin and orbital basis for the density-density part of the interaction.

F (real array of size $(nlm + 1)/2$, optional)
Slater integrals, F_0, F_2, ...

U (real, optional)
value of the screened Coulomb interaction.

J (real, optional, default=0)
value of the Hund's exchange interaction.

kanamori (logical, optional, default=no)
use the Kanamori parametrization for the Coulomb interaction matrix, which assumes that $U^{\sigma\bar{\sigma}}_{mm} = U$, $U^{\sigma\bar{\sigma}}_{m\neq m'} = U - 2J$ and $U^{\sigma\sigma}_{m\neq m'} = U - 3J$. This is a default choice (*kanamori*=yes) for even number of orbitals, *nlm*.

lcs (real matrix of size *nlm*, optional)
transformation of the Coulomb interaction matrix to the local coordinate system. It works when $\langle m, m'|V_{ee}|m'', m'''\rangle$ is defined via U and J or via Slater integrals, F.

dc_type (character, optional, default=SFLL)
type of the double counting correction to use.

***dc_type* = FLL** – fully localized limit [1], $\varepsilon^{\sigma}_{dc} = -U(N - 1/2) + J(N^{\sigma} - 1/2)$.

***dc_type* = FLL2** – non-magnetic version of fully localized limit,
$\varepsilon^{\sigma}_{dc} = -U(N - 1/2) + J(N - 1)/2$.

***dc_type* = AMF** – around mean-field [2], $\varepsilon^{\sigma}_{dc} = -U(N - \frac{1}{\lambda}N_{\sigma}) + J\frac{\lambda-1}{\lambda}N_{\sigma}$.

***dc_type* = AMF2** – non-magnetic version of around mean-field,
$\varepsilon^{\sigma}_{dc} = -(U(2\lambda - 1) - J(\lambda - 1))N/(2\lambda)$.

***dc_type* = SFLL** – simplified version of FLL2, $\varepsilon^{\sigma}_{dc} = -\bar{U}(N - x/2)$.
If J=0 and x=1, this double counting reduces to the FLL2 (or FLL).

***dc_type* = SFLL2** – double counting from [3], $\varepsilon^{\sigma}_{dc} = -\bar{U}N\frac{2\lambda-1}{2\lambda}$. If the interaction is defined via U and J or Slater's integrals, F, it reduces to AMF2.

dc_type* = *value – if a number is entered then this *value* will be used as a constant over the DMFT self-consistent loop.

In the above equations N (N^{σ}) is a total (per spin) occupation of the impurity, λ is the degeneracy of impurity, *nlm*, $\bar{U}$ is a mean value of density-density interaction terms, *ummss*. And for non-magnetic formulas $N^{\uparrow} = N^{\downarrow} = N/2$.

x_sfll (real, optional, default=1)
value of x for the simplified fully localized limit (SFLL).

Quantum impurity solvers

$\mathcal{AMULET}$ can use few different quantum impurity solvers. Below the information about options for each type of solvers is presented.

solver (character, optional, default=NONE)
name of the quantum impurity solver to be used.

***solver* = NONE** – skip a solution of the impurity problem but use corresponding self-energy if it exists. This option can be very useful in case of surface/slab calculations. The bulk's self-energy can be evaluated in advance and then used in a fixed form for slab.
***solver* = HF-QMC** – Hirsch-Fye quantum Monte-Carlo [6, 7]. Implementation with off-diagonal elements of the Green function is used [8].
***solver* = CT-QMC-W** – segment version of the continuous time quantum Monte-Carlo method.
solver* = CPA** – coherent potential approximation will be applied to uncorrelated orbitals. This is default solver if *cpa=yes* in the ***amulet.ini file.
***solver* = Hubbard_I** – Hubbard I solver. An atomic self-energy is used.
***solver* = ED** – exact diagonalization method (in unstable state now).

General options for QMC solvers

These options are applicable to *solver=HF-QMC* and *solver=CT-QMC-W*.

nqmc (integer, optional, default=1000000)
number of QMC measurements. This value will be divided by number of processors (cores).

nqmcwarm (integer, optional, default=500)
number of thermalization cycles in QMC per processor.

ninj (logical, optional, default=no)
turns on a calculation of the spin and orbital resolved time dependent density-density correlation function in QMC, $\langle \hat{n}_{m\sigma}(\tau)\hat{n}_{m'\sigma'}(0)\rangle$.

Options for *solver=CT-QMC-W*

nlegendre (integer, optional, default=-1) – maximum number of Legendre polynomials. If it is negative the Legendre polynomial representation for impurity Green function will not be used.

legendre_cutoff (integer array of size $2 * nlm$, optional, default=-1) – control number of Legendre polynomials for Fourier transform of each orbital.
legendre_cutoff=-1 – let algorithm to find the best number based on moment expansion.
legendre_cutoff= 0 – use maximum available (*nlegendre*).
legendre_cutoff=value – use this number (*value*).

improved_estimator (logical, optional)
calculate a self-energy directly on the Matsubara frequencies [4]. The default value depends on *nlegendre*. If *nlegendre*>0 then *improved_estimator*=yes.

measure_hstate_weight (logical, optional, default=.false.) - measure Hilbert state weights of the local impurity Hamiltonian and write it to ***hstate_weight_X.dat*** file.

L4Delta (integer, optional, default=L) – number of time points for a hybridization function, $\Delta(\tau)$.

meas_bin (integer, optional, default=2000) – number of CT-QMC updates that form a single measurement bin.

observable_bin_size (integer, optional, default=-1) – number of CT-QMC updates to make a single jack-knife bin. By default AMULET tries to slice whole calculation into 100 bins, unless this parameter is manually specified.

ctqmc_updates (string, optional, default='basic+shift:0.1+sz_swap:0.01') – specify type of CT-QMC updates and their weights.
sz_swap – global spin swap.
shift – shifts position of creation/annihilation operators. This type of update does not change a perturbation order and has high acceptance rate.
symmetry_swap – swaps of a general kind, requires symmetry specification in the ***impurity_name.symmetry*** file.

random_generator (integer, optional, default='boost_mt19937') – specifies a pseudo-random number generator.
random_generator=dsfmt – Mersenne-Twister rng. The best choice, SSE2 optimized.
random_generator=boost_mt19937 – Boost implementation of a Mersenne-Twister, compatible with most architectures but a bit slower.

delta_min (real, optional, default=0) limits and adjusts minimal value of produced hybridization functions to eliminate negative values due to statistical noise. Optimal value varies by the system, we recommend 1.e-7, but no larger than 1.e-5, otherwise insulating solution will be lost.

Options for *solver=ED*

nbath (integer, optional, default=nlm) – number of bathes for a discretization of a hybridization function in exact diagonalization.

Unn (logical, optional, default=.false.) – use only density-density part of the Coulomb interaction matrix.

ed_opt (integer, optional, default=1) – optimization of exact diagonalization procedure.
ed_opt = 0 – no optimization.
ed_opt = 1 – some basic optimization.
ed_opt = 2 – most optimized version (experimental, not yet completely tested).

fit (integer, optional, default=-1) – which function to fit for the exact diagonalization.
fit = -1 – fit bath Green function, $G^0_{m\sigma}(i\omega_n)$, for each orbital.
fit = 1 – fit hybridization function, $\Delta_{m\sigma}(i\omega_n)$, for each orbital.
fit = 2 – fit matrix of hybridization function, $\hat{\Delta}(i\omega_n)$.

fix_ed (logical, optional, default=no) – fix impurity level, e_d, in $G^0_{m\sigma}(i\omega_n) = i\omega_n - e_d - \Delta_{m\sigma}(i\omega_n)$. This option is valid for *fit=-1* only.

weight (integer, optional, default=1) – type of uncertainty function for fit.
weight = 1 – $\sigma(i\omega_n) = 1$ for all Matsubara frequencies.
weight = 2 – $\sigma(i\omega_n) = log(1+n)$, where $i\omega_n = (2n+1)\pi/\beta$.
weight = 3 – $\sigma(i\omega_n) = 5 + n$.
weight = 4 – $\sigma(i\omega_n) = A + (1-A)/(1+e^{C(i\omega_n - B)})$.
weight = 5 – $\sigma(i\omega_n) = n^3$.

ufp (real, optional, array of size 3, default=10, 30, 0.5) – defines A, B and C for *weight=4*.

ed_gf_nev (integer, optional, default=6×*nlm*)
number of eigenvalues to use for the Green function calculations.

ed_arno_nev (integer, optional, default=*ed_gf_nev*) – number of the lowest eigenvalues to find in a sector.

ed_gf_tol (real, optional, default=10^{-8}) – tolerance for Green function calculation.

ed_arno_tol (real, optional, default=-1) – tolerance for an eigenvalue search in Arnoldi. It will be setup automatically if negative.

ed_gf_decomp (real, optional, default=0.01) – probability cut for an eigenvector decomposition.

Options for *solver=CPA*

x_cpa (real, optional, default=1)
concentration of the impurity.

Options for *solver=Hubbard_I*

For the moment this solver doesn't require any options.

umtrx_N.ini file

This file is optional and can be used to set up the matrix of Coulomb interaction, $\langle m, m'|V_{ee}|m'', m'''\rangle$, in the most general way. The file format is very simple and contains five columns,

$$m \quad m' \quad m'' \quad m''' \qquad \langle m, m'|V_{ee}|m'', m'''\rangle,$$

where the first four integers are orbital indexes and the last column is the real number with the corresponding value of Coulomb interaction. Zero's values of interaction can be safely omitted.

system.am file

The ***system.am*** file has a tag-based format. This leads to a flexibility of adding new tags/data fields to a file.

```
...
& tagname
   data field
...
```

Ampersand (&) starts a section name followed by a tagname. Next line(s) contain(s) a data field in a formatted form, which depends on the data. The section has to be finished with an empty line. The order of sections is completely arbitrary and some of them can be optional. Comments can be placed between sections and should be separated by empty lines.

Below, a list of all tags with its description is presented.

Tagname	Attribute	Description
hash	optional	internal identifier, that can be used to check the consistency of different files. The UNIX timestamp is usually used.
codestamp	optional	name of the band structure package in which ***system.am*** file was generated.
cell	mandatory	scaling parameter (*alat*) and crystal lattice vectors in Cartesian coordinates (the first line defines the first vector and so on). The data are in atomic units.
crystcoord	optional (default=false)	use crystal (or fractional) coordinates (true) or Cartesian coordinates (false) for atomic positions (in units of *alat*).
atoms	mandatory	contains information about atoms. The first line provides a number of atoms. The rest of the lines set atomic labels and atomic positions in units that depend on *crystcoord*.
nelec	optional	number of electrons to adjust Fermi level.
efermi or *fermi*	optional	self-consistent value of the Fermi level obtained in DFT calculation (in eV).
basis	mandatory	defines a localized (Wannier-like) basis functions of the Hamiltonian. The first line sets the total basis dimension and number of the basis functions. Then, for each basis (Wannier) function, there is a line, which defines next parameters: atomic label, reference to atom, orbital symmetry of a wave function, position in the Hamiltonian and order of the cubic harmonics.

The example of ***system.am*** file for bcc-Fe with explanations is shown below.

```
# This file was written on: 03.12.2018  08:20:35

& hash
  1544430035

&Codestamp
 exciting-plus
```

```
&cell
 1.000000000
  2.76987  2.76987  2.76987
  2.76987  2.76987 -2.76987
  2.76987 -2.76987  2.76987

&fermi
   8.86799076094252

&crystcoord
 true

#   The tag below is absent in the real exciting-plus output but
#   we added it for educative purpose

& NELEC
  8

&atoms
           1
  Fe   0.00000  0.00000  0.00000

# Basis description:
# dim, nblocks
# atom_sym, atom_num, l_sym, block_dim, block_start, orbitals(1:block_dim)
&basis
 9   3
 Fe  1 s 1   1     1
 Fe  1 p 3   2     2 3 4
 Fe  1 d 5   5     5 6 7 8 9
```

Most of tags are well explained, and hence, we will concentrate how the data from ***system.am*** file can be transferred to the ***amulet.ini*** and ***impurity_X.ini*** files. In case of iron, the total valence charge is 16 (***grep*** 'Total valence charge' INFO.OUT), while the *nelec* provides 8 electrons only. The difference is due to a projection to energy bands close to the Fermi level, or in the other words, due to the elimination of the occupied semicore $3s$ and $3p$ bands (8 electrons). The value of *nelec* can be used as *ntotal* token in the ***amulet.ini*** file. The *fermi* tag provides a self-consistent value for the Fermi energy out of the DFT calculation and it can be used as an initial value for the token *mu* in the ***amulet.ini*** file.

The *basis* tag contains the information that can be directly utilized in the ***impurity_X.ini*** file. The first line sets a total number of the basis functions (dimension of the Hamiltonian) and a number of groups of Wannier functions with different symmetry. Then each group is described. In the above example, all orbital groups belong to Fe atom number one (unfortunately, in this example there is one atom only). Then the orbital symmetry is specified (s, p and d in our case) followed by degeneracy (1, 3, 5). Then the starting positions in the Hamiltonian (1, 2 and 5) are provided. The rest of the data sets the order of cubic harmonics (for numbering of cubic

harmonics see table 1).

Thus, if one wants to use Fe d states as correlated in the DFT+DMFT calculations, we should setup next tokens in the ***impurity_1.ini*** file.

```
  name = Fe                              !  label
  nlm = 5                                !  degeneracy
  himppos = 5                            !  starting position in the Hamiltonian
  cubic_harmonic_order = 5 6 7 8 9       !  order of cubic harmonic
   ...
```

hamilt.am file

The Hamiltonian file is in tag-based format. The tags can be randomly placed in the ***hamilt.am*** file. Each tag starts from an ampersand followed by a tag-name (& *tagname*). Comments start from a hash sign (#). Below is an example of the Hamiltonian file.

```
# This file was generated on:  10Aug2015 16:35:28

& Codestamp
TB-LMTO

&hash
  1439206528

& nspin
2

&nkp
  512

 &   dim
 16

&kpoints
 0.001953125000  0.03125  0.03125  0.03125
 0.001953125000  0.21875 -0.03125 -0.03125
 0.001953125000  0.40625 -0.09375 -0.09375
 ...
 ...
 0.001953125000 -0.03125 -0.03125 -0.03125

& Hamiltonian
 -0.700581156579864       1.779754552394368E-014  1.120032799947242E-014
  3.444590910039907E-014 -1.267478085593797E-016  3.734663777632630E-016
 -4.982928714116384E-017 -6.464169035639541E-017  6.169492808056197E-017
 ...
```

```
...
...
```

As usually, all tags are divided on required and optional.

nkp	(required)	–	number of **k**-points in the Hamiltonian.
dim	(required)	–	Hamiltonian dimension.
nspin	(optional)	–	number of spins (default *nspin=1*).
kpoints	(required)	–	weights and positions of **k**-points.
Hamiltonian	(required)	–	the Hamiltonian.
Codestamp	(optional)	–	name of code used for the Hamiltonian production.
hash	(optional)	–	UNIX timestamp.

A part of code that reads weights and positions of **k**-points (from ***src/read_hamiltonian.f90***):

```
...
...
  do ikp = 1, nkp
    read(iuham,*,iostat=ios) wtkp(ikp), ( bk(i,ikp), i=1,3 )
  end do
...
```

A part of code that reads the Hamiltonian looks like:

```
...
...
  do is = 1, nspin
    do ikp = 1, nkp
      read(iuham,*) ( ( hr(i,j,ikp,is), hi(i,j,ikp,is), j=i,dim ), i=1,dim )
    end do
  end do
...
```

It is important to note that an upper triangle of a matrix is written assuming that the Hamiltonian is hermitian.

bare_dos_N.ini file

Alternative way to construct a Green function is to use a density of states, $G(i\omega_n) = \int \rho(\epsilon) d\epsilon / (i\omega_n - \epsilon - \Sigma(i\omega_n))$. For this purpose the ***bare_dos_N.ini*** file with the density of states (DOS) must be present instead of the Hamiltonian. It has a format, where the first column is an energy and the rest of columns is the DOS itself for the nlm orbitals. In spin-polarized calculations a number of orbitals is doubled. Part of code that reads the DOS is presented (from ***src/read_dos.f90***).

```
  ...
  ...
!       Read bare DOS
  do ie = 1, n_dos
    read(iunit,*,iostat=ierr) edos(ie), (( dos(i,ie,is), i=1,n ), is=1,nspin )
  end do
  ...
```

sigma_N file

The ***sigma_N*** file contains a self-energy and related data for each type of impurity. The data are stored in a tag-based format on each iteration (overwritten). This is the input/output file and it is used for a self-consistency continuation.

6 Executables and control files

The $\mathcal{AMULET}$ toolbox consists of several executable routines and control files. The main program is ***amulet.mpi***. It can be executed with a mpi-launcher installed on your system, for example:

```
[gandalf@amulet_user]>  srun -n 64 -o out.msg -e err.msg ~/bin/amulet.mpi
```

This program requires next input files to be run successfully: ***amulet.ini***, ***impurity_N.ini***, ***hamilt.am*** (or ***bare_dos_N.ini***).

There are two files named ***lock.amulet*** and ***stop.amulet*** that controls an execution of the main program. The first file is created in a working directory at the beginning and will be deleted at the end of successful run. It prevents overwriting of output files by launching of the second job in the same working directory. The *ignorelockfile* token in ***amulet.ini*** file can be used to change default behavior. The ***stop.amulet*** file can be used to interrupt softly the code execution. If during the run ***amulet.mpi*** will find this file in the working directory it will finish current iteration and stop.

Preprocessing routines

wannier2amulet

wannier2amulet is an interface between Wannier90 code and $\mathcal{AMULET}$ package. By this any DFT code which has a possibility to construct maximally localized Wannier functions can be used with the $\mathcal{AMULET}$ (for example VASP, Wien2k and many others). On input it requires ***hrup.ini*** (and ***hrdn.ini*** if you have spin-polarized DFT calculation) and optionally ***klist.ini*** files. File ***hrup.ini*** (***hrdn.ini***) contains an information about real space Hamiltonian obtained with the Wannier90 code. ***klist.ini*** file is optional and contains **k**-points. If you have no this file the ***wannier2amulet*** routine will prompt you about divisions of a reciprocal space. On output the ***hamilt.am*** file will be generated.

rotate_hamiltonian

rotate_hamiltonian routine is designed for a rotation of certain blocks in the Hamiltonian. There are three reasons to make such rotation. First, there are equivalent atoms in a compound under consideration. They are symmetry connected but $\mathcal{AMULET}$ knows nothing about this symmetry and one needs to impose it manually. Second reason is to use a set of harmonics different from standard real (cubic), or to be more precise to use a mixer of it (for example in trigonal symmetry). And the last reason is to reduce off-diagonal elements of Green function. One should mention that all these cases are not mutually exclusive and can be realized in one compound. ***rotate_hamiltonian*** uses ***hamdiag.ini*** input file which has a next form:

```
1    !   Use (1) rotation matrix or generate it (0) from H(R)
2    !   number of atoms to rotate
1    !   position in hamilt.am
5    !   degeneracy
 0.0         0.81649658  0.0         0.57735027  0.0
 0.0         0.0        -0.57735027  0.0        -0.81649658
 0.0        -0.57735027  0.0         0.81649658  0.0
```

```
 0.0         0.0        -0.81649658  0.0         0.57735027
 1.0         0.0         0.0         0.0         0.0
6    !   position in hamilt.am
5    !   degeneracy
 0.0         0.81649658  0.0         0.57735027  0.0
 0.0         0.0        -0.57735027  0.0        -0.81649658
 0.0        -0.57735027  0.0         0.81649658  0.0
 0.0         0.0        -0.81649658  0.0         0.57735027
 1.0         0.0         0.0         0.0         0.0
```

The first string defines use of given rotation matrix (1) or it can be generated from corresponding $H(\mathbf{R})$ block. Second string is a number of atoms which blocks will be rotated. Than, for each atom one needs to specify a position in the Hamiltonian, degeneracy and if the first parameter is equal to one, one needs to add a rotation matrix for this block (like in above example). In case of the rotation based on the $H(\mathbf{R})$ the rotation matrices are absent and ***hamdiag.ini*** file looks like this:

```
0    !   Use (1) rotation matrix or generate it (0) from H(R)
2    !   number of atoms to rotate
1    !   position in hamilt.am
5    !   degeneracy
6    !   position in hamilt.am
5    !   degeneracy
```

After execution of the ***rotate_hamiltonian*** one will find a ***hamdiag.out*** file with the information about rotation and ***rotated_hamilt.am*** file with the rotated Hamiltonian.

green_function.mpi

green_function.mpi routine can be used to calculate the Green function on the real/imaginary energies. This interactive program will produce the ***Gf_x,y.dat*** files which hold the Green functions. ***green_function.mpi*** requires a ***hamilt.am*** file only, and it can be very useful to produce LDA density of states or to verify the rotation to the local coordinate system.

hk2hr

hk2hr routine is an interactive tool to compute hopping integrals. It requires two input files – ***system.am*** and ***hamilt.am***. On output two files will be created with on-site crystal field splitting (***crystal_field.out***) and the hopping integrals (***hoppings.out***).

Postprocessing routines

Density of states

dmft_dos routine can be used to calculate a density of states. It requires all main input files: ***amulet.ini***, ***impurity_N.ini***, ***hamilt.am*** and ***sigma_re_N*** file. The last contains the self-energy on the real energy axis. On output, ***total_dos.dat*** and ***dos_z.dat*** files will be created with z numbering all states of the Hamiltonian.

Spectral function

akw routine can be used to calculate a spectral function, $A(\mathbf{k},\varepsilon) = -\Im G(\mathbf{k},\varepsilon)/\pi$. It requires all main input files: ***amulet.ini***, ***impurity_N.ini***, ***hamilt.am*** and ***sigma_re_N*** file. One should note that the ***hamilt.am*** file has to contain the information about **k**-points and the Hamiltonian along high symmetry directions of the Brilluoin zone. On output, ***Akw_total.dat***, ***Akw_z.dat*** and ***arpes.gpi*** files will be created. ***Akw_total.dat*** and ***Akw_z.dat*** contain the total and orbitally resolved spectral functions. The ***arpes.gpi*** file is a corresponding gnuplot script.

Maximum entropy method

maxent.mpi routine is designed for analytic continuation of the Green function obtained with the quantum Monte Carlo methods. This program is based on the paper of Sandvik [5] and it implements a classic maximum entropy method with a stochastic minimization. The ***maxent.mpi*** program is MPI parallelized and requires its own input file ***mem.ini*** and the file(s) with the Green function and corresponding variance on imaginary time grid, ***Gtau_N_x,x.dat*** and ***Gtau_var_N_x,x.dat***. The later input files have to be supplied from a command line, for example:

```
[user]> maxent.mpi --input Gtau_1_1,1.dat Gtau_var_1_1,1.dat Gtau_2_4,4.dat Gtau_var_2_4,4.dat
```

This will produce two output files, ***Spectrum_1_1,1.dat*** and ***Spectrum_2_4,4.dat***.

The configuration file ***mem.ini*** is divided into two sections: a general section contains general parameters related to the maximum entropy method and an input section is about your data. The general parameters are

emin (real, optional, default=-20) – is a lower bound for the real energy grid.

emax (real, optional, default=20) – is a upper bound for the real energy grid.

estep (real, optional, default=0.1) – is a step for the real energy grid.

annealing_steps (integer, optional, default=6000) – is a number of Monte Carlo steps at each minimization cycle.

alpha (real, optional, default=1200) – is an entropy alpha parameter (see paper of Sandvik [5]).

seed (integer, optional, default=0) – is a random number generator seed.

random_restart (logical, optional, default=true) – if it is true, each consecutive run (see *num_runs*) will start from random spectra and initial parameters. Otherwise spectra and parameters from previous run will be used to start.

verbosity (integer, optional, default=3) – is a verbosity level.

num_runs (integer, optional, default=1) – is a number of simulations to run for each input file.

max_iter (integer, optional, default=30) – is a number of self-consistent iterations to determine optimal alpha.

theta (real, optional, default=10) – is a theta parameter (see paper of Sandvik [5]).

rfac (real, optional, default=1) is an amplitude multiplier for the Monte Carlo.

The parameters for the input section are

variance_multiplier (real, optional, default=1) is a multiplier for the variance.

starting_spectra (character) – filename for the spectra to start first simulation if *random_restart*=false.

Below is an example of the ***mem.ini*** file:

```
[general]
 emin = -10
 emax = 10
 estep = 0.005
 annealing_steps = 6000
 alpha = 1200
 seed = 987204
 random_restart = false
 num_runs = 1
 max_iter = 99
 theta = 10.0
 rfac = 1.0

[input]
 variance_multiplier = 1.0
 starting_spectra = previous_spectra.dat
```

Analyzing quantum impurity configuration (*sector_statistics*)

Weights of many-body states of the quantum impurity are stored in the ***hstate_weight_1.dat*** file if *solver=ct-qmc-w* and *measure_hstate_weight = yes* in the ***impurity_1.ini*** file. The example of the ***hstate_weight_1.dat*** file for the five-degenerate case, *nlm=5*, is presented below. In this case, the may-body state consists of five digits (columns) for one spin direction and the next five digits for the opposite (second quantization notations). The last column is a weight of the given configuration. The order of orbitals corresponds to the *cubic_harmonic_order* or *rhtm* token. Therefore, the probability to find three electrons siting in $d^{\uparrow}_{xy}$, $d^{\uparrow}_{xz}$ and $d^{\downarrow}_{x^2-y^2}$ is 0.0001989.

```
...
1 0 0 1 0 0 0 0 0 1     0.0001989
0 1 0 1 1 0 0 0 0 0     0.0002422
0 0 1 1 1 0 0 0 0 0     0.0132168
```

```
  1 1 0 0 0 1 0 0 0 0     0.0000001
  1 0 1 0 0 1 0 0 0 0     0.0000029
  ...
```

sector_statistics routine allows one to analyze the many-body configuration stored in this file. It requires the ***sector_statistics.ini*** file shown below.

```
10          !  spin-orbital degeneracy
2           !  number of sub-shells to analyze
t2g         !  sub-shell name
3           !  number of orbitals in sub-shell
1 2 4       !  orbitals positions
eg          !  sub-shell name
2           !  number of orbitals in sub-shell
3 5         !  orbitals positions
```

The first line contains spin-orbital degeneracy ($2\times nlm$). The second line sets a number of sub-shells to analyze. Then for each sub-shell one needs to provide name, the orbital degeneracy and positions. One should note here that the doubled sum of the sub-shell degeneracies (fourth and seventh lines) must be equal to the spin-orbital degeneracy (first line). In the above example, one has two sub-shells to analyze. The first sub-shell is named **t2g** with the degeneracy equal to **3** and positions **1 2 4**. This order corresponds to the *rhtm=VASP* token. The second sub-shell is **eg** with the degeneracy equal to **2** and positions **3 5**.

The ***sector_statistics*** routine reads from standard input and writes to standard output and it has to be executed as:

```
[user]> sector_statistics < hstate_weight_1.dat > sector_statistics.out
```

One should note here that ***hstate_weight_1.dat*** file must be used as standard input. Below is a part of ***sector_statistics.out*** file.

```
          Sector statistics
 Ntot     Weight     Sz(i)      0          1          2          3          4          5
  0     0.0000000    |        ----       ----       ----       ----       ----       ----
  1     0.0000000    |        ----       ----       ----       ----       ----       ----
  2     0.0000000    |        ----       ----       ----       ----       ----       ----
  3     0.0000000    |        ----       ----       ----       ----       ----       ----
  4     0.0000000    |        ----       ----       ----       ----       ----       ----
  5     0.0000180    |        ----   0.0000051        ----   0.0000109        ----   0.0000020
  6     0.0090957    |   0.0000167        ----   0.0046116        ----   0.0044674        ----
  7     0.8118625    |        ----   0.0043162        ----   0.8075463        ----        ----
  8     0.1724999    |   0.0005869        ----   0.1719130        ----        ----        ----
  9     0.0064779    |        ----   0.0064779        ----        ----        ----        ----
 10     0.0000455    |   0.0000455        ----        ----        ----        ----        ----

SUM(Sz(i))      0          1          2          3          4          5
            0.000649   0.010799   0.176525   0.807557   0.004467   0.000002

          Orbital configurations
...
 Ntot     Weight
  7      0.8118625
Sz |  t2g  eg    Weight
  3|     3     4       0.0001444
  3|     4     3       0.0106125
  3|     5     2       0.7967894
```

```
  1|     3     4       0.0000066
  1|     4     3       0.0008948
  1|     5     2       0.0031132
  1|     6     1       0.0003016

 Ntot    Weight
  8      0.1724999
Sz |  t2g  eg    Weight
  2|     4     4       0.0009147
  2|     5     3       0.1396354
  2|     6     2       0.0313629
  0|     4     4       0.0000598
  0|     5     3       0.0004411
  0|     6     2       0.0000860
...
```

The beginning of the output file repeats the information that can be found in the ***amulet.out*** file. It reports the weights of different ionic and spin state configurations. In the above example, the effective quantum impurity under consideration is in d^7 and d^8 high spin states with probabilities 0.812 and 0.172, respectively.

Next blocks of the output file contain a similar information collected for sub-shells requested in the input file. One can see that for d^7 ionic state (block) the most probable configuration is $t_{2g}^5 e_g^2$ with S_z=3, while for d^8 block it is $t_{2g}^5 e_g^3$ with S_z=2.

7 Output files

amulet.out file

amulet.out is a main output file and it is self-explanatory.

**_N_x,y.dat* files

In this subsection the files with a mask ****_N_x,y.dat*** are listed. One should remind that N is for impurity number and x, y denote orbital indexes. In a multi-orbital case the data with small values will not be printed. It is important to note that the data from different iterations are separated by two empty lines to simplify a use of plotting utilities. The output format is very simple. The first column is for mesh and the rest are for data. If a ***tau*** suffix present in the filename then the data are on imaginary time mesh, otherwise it is energy data. Suffix ***re*** is utilized for the real energy mesh.

Below is an example of ***Sigma_1_1,1.dat*** file for the first orbital of the first impurity. The second and third columns are for real and imaginary parts of self-energy for one spin direction, while the last two columns are for opposite spin. If the calculations are carried out with *spinpol=no*, then the file will have three columns only.

```
7.85398E-02  -3.16591687E-01 -1.86908545E-02  2.64112460E-01 -1.89984679E-02
2.35619E-01  -3.16995088E-01 -5.62258463E-02  2.65302422E-01 -5.69286817E-02
3.92699E-01  -3.17830836E-01 -9.26083325E-02  2.67386279E-01 -9.38947033E-02
5.49779E-01  -3.19019407E-01 -1.27799669E-01  2.70208723E-01 -1.29643817E-01
...
...
```

Below is a list of files created for different values of verbosity.

Sigma_N_x,y.dat – self-energy, $\Sigma(i\omega_n)$.
Green_total.dat – total Green function.
Gtau_N_x,y.dat – impurity Green function, $G(\tau)$.
Gtau_var_N_x,y.dat – variance for the impurity Green function, $var[G(\tau)]$.
verbos ≥ 10
Green_N_x,y.dat – local Green function, $G(i\omega_n)$.
Delta_tau_N_x,y.dat – hybridization function, $\Delta(\tau)$.
G0tau_N_x,y.dat – bare Green function, $\mathcal{G}_0(\tau)$.
verbos ≥ 15
Delta_N_x,y.dat – hybridization function, $\Delta(i\omega_n)$.
verbos ≥ 20
Green_imp_N_x,y.dat – impurity Green function, $G_{imp}(i\omega_n)$.
Green0_N_x,y.dat – Green function of the thermostat, $\mathcal{G}_0^{-1}(i\omega_n) = G^{-1}(i\omega_n) + \Sigma(i\omega_n)$.
Delta_fit_N_x,y.dat – fit of the hybridization function, $\Delta(i\omega_n)$.
G_cpa_N_x,y.dat – CPA Green function.
Sigma_cpa_N_x,y.dat – CPA self-energy.
verbos ≥ 25
Delta_imp_N_x,y.dat – impurity hybridization function.
Green0_imp_N_x,y.dat – bare impurity Green function.
SigmaG_N_x,y.dat – $\Sigma G[i\omega_n]$.
Gtau_actime_N_x,y.dat – autocorrelation time for $G(\tau)$.

ninj_N_x,y.dat file

The ***ninj_N_x,y.dat*** file will be created if the token *solver* will be set to *ct-qmc-w* or *hf-qmc* simultaneously with *ninj = yes*. Than the correlator $\langle n_{x\sigma}(\tau) n_{y\sigma'}(0)\rangle$ will be written

```
0.00E+00  5.0007677579E-01 1.5311045058E-01 1.5311045058E-01 4.9992322452E-01
2.00E-01  4.6895918799E-01 1.5378697502E-01 1.5379622445E-01 4.6882894305E-01
4.00E-01  4.4380534999E-01 1.5534846777E-01 1.5536234552E-01 4.4367014630E-01
6.00E-01  4.2327476876E-01 1.5747625393E-01 1.5748664577E-01 4.2313533367E-01
8.00E-01  4.0637042746E-01 1.5993482695E-01 1.5994142960E-01 4.0622816759E-01
...
...
```

where the first column is the imaginary time and the rest are $\langle n_{x\uparrow}(\tau) n_{y\uparrow}(0)\rangle$, $\langle n_{x\uparrow}(\tau) n_{y\downarrow}(0)\rangle$, $\langle n_{x\downarrow}(\tau) n_{y\uparrow}(0)\rangle$, $\langle n_{x\downarrow}(\tau) n_{y\downarrow}(0)\rangle$, respectively.

hstate_weight_N.dat file

The file ***hstate_weight_N.dat*** contains weights for different multi-electron configurations obtained in the CT-QMC-HYB calculation. It will appear if tokens *solver=ct-qmc-w* and *measure_hstate_weight=yes* will be set in the ***impurity_N.ini*** file. Below is an example of the file

```
...
...
1 0 0 1 1 0 0 0 0 0     0.0001989
0 1 0 1 1 0 0 0 0 0     0.0002422
0 0 1 1 1 0 0 0 0 0     0.0132168
1 1 0 0 0 1 0 0 0 0     0.0000001
1 0 1 0 0 1 0 0 0 0     0.0000029
...
...
```

where the multi-electron configuration is shown by zeros and ones and the corresponding weight is in the last column.

8 Acknowledgements

We would like to thank Alexander Lichtenstein, Antoine Georges, Gabriel Kotliar, Silke Biermann, Eva Pavarini, Ole Andersen, Olivier Parcollet and all friends who participated by ideas and recommendations at earlier reincarnations of $\mathcal{AMULET}$.

References

[1] V.I. Anisimov, F. Aryasetiawan, and A.I. Lichtenstein,
First-principles calculations of the electronic structure and spectra of strongly correlated systems: the LDA+U method.
Journal of Physics: Condensed Matter **9**, 767 (1997).

[2] M.T. Czyźyk, and G.A. Sawatzky,
Local-density functional and on-site correlations: The electronic structure of La_2CuO_4 and $LaCuO_3$.
Physical Review B **49**, 14211 (1994).

[3] J. Kuneš, V.I. Anisimov, A.V. Lukoyanov, and D. Vollhardt,
Local correlations and hole doping in NiO: A dynamical mean-field study.
Phys. Rev. B **75**, 165115 (2007).

[4] Hartmut Hafermann, Kelly R. Patton, and Philipp Werner,
Improved estimators for the self-energy and vertex function in hybridization-expansion continuous-time quantum Monte Carlo simulations.
Phys. Rev. B **85**, 205106 (2012).

[5] A. Sandvik,
Stochastic method for analytic continuation of quantum Monte Carlo data.
Phys. Rev. B **57**, 10287 (1998).

[6] J. E. Hirsch,
Discrete Hubbard-Stratonovich transformation for fermion lattice models.
Phys. Rev. B **28**, 4059 (1983).

[7] J. Hirsch, and R. Fye,
Monte Carlo Method for Magnetic Impurities in Metals.
Phys. Rev. Lett. **56**, 2521 (1986).

[8] A.I. Poteryaev, J.M. Tomczak, S. Biermann, A. Georges, A.I. Lichtenstein, A.N. Rubtsov, T. Saha-Dasgupta, and O.K. Andersen,
Enhanced crystal-field splitting and orbital-selective coherence induced by strong correlations in V_2O_3.
Phys. Rev. B **76**, 85127 (2007).