Documentation – Hybridization Expansion CT-QMC solver version 3.0b1

Emanuel Gull,[1,2] Hartmut Hafermann,[3] and Philipp Werner[4]

[1] *Max Planck Institute for the Physics of Complex Systems, Dresden, Germany*
[2] *Physics Department, University of Michigan, Ann Arbor, MI*
[3] *École Polytechnique, CNRS, 91128 Palaiseau Cedex, France*
[4] *Department of Physics, University of Fribourg, 1700 Fribourg, Switzerland*

(Dated: January 23, 2013)

This is the user documentation for the hybridization expansion solver of the ALPS DMFT code. It documents installation, running, and integration of the hybridization code into the ALPS self-consistencies of the ALPS DMFT code and the solver's Python interface. It is designed to be a useful documentation for users, primarily in the LDA+DMFT community.

I. INTRODUCTION

Welcome! This is the user documentation for the third version of the ALPS[1] hybridization expansion[2] CT-QMC[3] code, written by Emanuel Gull, in collaboration with Hartmut Hafermann and Philipp Werner, based on an earlier version[4] by Philipp Werner, Emanuel Gull, Brigitte Surer, and Matthias Troyer. This user documentation is designed to make the program useful for users, in particular users from the LDA+DMFT field, who want to replace their solvers (IPT, Hirsch Fye, other versions of CT-QMC) with the ALPS hybridization code. A basic understanding of impurity models, Monte Carlo simulations and, where needed, LDA+DMFT is assumed. Knowledge of the inner workings of CT-QMC is not required, and we strive to make the code accessible to users who are not Monte Carlo experts.

This documentation, as well as version 3 of the hybridization expansion code, is open source. Bug reports and bug fixes, problem reports, and suggestion for improvements are most welcome.

II. PREREQUISITES

A. Downloading

The hybridization solver is included in the compiled ALPS packages as a binary. Release versions and nightly builds are available from `alps.comp-phys.org`. You can find the binary version at `alps-prefix/bin/hybridization`.

B. Building

The hybridization expansion code uses several ALPS libraries[1] and requires ALPS to build. You will therefore need to obtain the source code for ALPS from alps.comp-phys.org, either by downloading a current nightly build or by requesting a subversion account and downloading the svn version of ALPS. The ALPS home page has detailed instructions on how this works. In case of problems building ALPS, the ALPS user mailing list will help you out. If this does not work, please don't hesitate to contact Emanuel .

The ALPS DMFT code is dependent on two additional modules: HDF5 and MPI are required. Once ALPS and the hybridization code are installed in `$ALPS_PREFIX/bin/hybridization3`, try running it. You will see

```
terminate called after throwing\
 an instance of 'std::invalid_argument'
  what():  No job file specified
In $ALPS_ROOT/src/alps/ngs/lib/mcoptions.cpp\
 on 62 in mcoptions
```

If you have an MPI environment you can also run the code using, *e.g.*,

```
  $MPI_RUN_COMMAND -np 3 $ALPS_PREFIX/ \
bin/hybridization3
```

and you'll see the same output three times. `$MPI_RUN_COMMAND` is platform dependent, common options are `mpirun`, `mpiexec`, `openmpirun`, and so on. Consult your MPI manual on what you need to use.

III. RUNNING THE HYBRIDIZATION CODE

The input of the solver consists of a parameter file, the hybridization function $\Delta(\tau)$ and optionally, the retarded interaction function $K(\tau)$ and its first derivative $K'(\tau)$.

A. Using the standalone executable

1. *Specifying a parameter file*

We start with a simple parameter file to solve a single site impurity problem. A minimal parameter file, call it `hyb1.param`, could look like this:

```
SWEEPS = 100000000
MAX_TIME = 60
THERMALIZATION = 1000
SEED = 0
```

```
N_MEAS = 50
N_HISTOGRAM_ORDERS = 50
N_ORBITALS = 2
U =  4.0
MU = 2.0
DELTA = "delta.dat"
N_TAU = 1000
BETA = 45
```

This is a parameter file for a single impurity Anderson model (two "orbitals", i.e. $N_{\mathrm{orb}} = 2$, one for spin "up" and one for spin "down") with $U = 5$, $\mu = 2.5$, and $\beta = 30$ for which the hybridization function will be read from the file `delta.dat`. The energy units are the ones of `delta.dat`. This simulation runs for sixty seconds.

Using the command `p2h5` this parameter file is converted into an hdf5 file:

```
$ALPS_PREFIX/bin/p2h5 hyb1.h5 < hyb1.param
```

2. *Running the solver*

We can then run the program using this parameter file (make sure this is in a directory with read/write access and the file `delta.dat` is present):

```
$ALPS_PREFIX/bin/hybridization3 hyb1.h5
```

A typical output is:

```
U matrix with 2 orbitals:
0 5
5 0
chemical potential with 2 orbitals:
2.5 2.5

local configuration:
0 empty
1 empty

the hybridization function is:
0 -0.5 -0.5
1 -0.492823 -0.492823
2 -0.485805 -0.485805
3 -0.478945 -0.478945
4 -0.47224 -0.47224
5 -0.465686 -0.465686
6 -0.459282 -0.459282
7 -0.453024 -0.453024
8 -0.446909 -0.446909
9 -0.440935 -0.440935
... *** etc *** ...
1000 -0.5 -0.5

process 0 starting simulation
```

Running the same simulation under MPI will produce additional lines indicating the process number: `process 1 starting simulation` and so on. After `MAX_TIME` is up or the number of `SWEEPS` have been done, the code exits. The results will be written to the file `hyb1.out.h5` in this case. The results can also be written in human-readable format by adding the line

```
TEXT_OUTPUT = 1
```

to the parameter file. Do not forget to convert the parameter file to hdf5 format before running the solver.

B. Python interface

If ALPS is built with Python support (parameter `ALPS_BUILD_PYTHON=ON`), the solver is also built as a Python module. It can directly be called from within a Python script. This provides a flexible framework which allows one to easily set up tasks ranging from calculations for multiple parameters to complex selfconsistency schemes.

Basic usage of the Python interface is illustrated by the following script, which repeats the previous example for the standalone executable:

```
import pyalps.cthyb as cthyb # solver module
import pyalps.mpi as mpi     # MPI library

parms={
'SWEEPS'               : 100000000,
'MAX_TIME'             : 60,
'THERMALIZATION'       : 1000,
'SEED'                 : 0,
'N_MEAS'               : 50,
'N_HISTOGRAM_ORDERS'   : 50,
'N_ORBITALS'           : 2,
'U'                    : 4.0,
'MU'                   : 2.0,
'DELTA'                : "delta.dat",
'N_TAU'                : 1000,
'BETA'                 : 45,
'TEXT_OUTPUT'          : 1
}

if mpi.rank==0:
  f=open("delta.dat","w")
  for i in range(parms["N_TAU"]+1):
    f.write("%i %f %f\n"%(i,-0.5,-0.5))
  f.close()

cthyb.solve(parms)
```

The first line imports the solver module. Note that importing the MPI library in the second line is mandatory, even if the code is run on a single process. In the next four lines a simple (constant) hybridization function is written to file. The parameters are specified in the form of a Python `dict`. Finally the solver is invoked by executing the `solve` method of the Python module. The latter takes the parameter dict as an argument. The hybridization function is read from file. Results are written

to the file `results.out.h5`. The name (without suffix) can be altered by specifying the parameter `BASENAME`.

The script is executed as follows:

```
alpspython scriptname.py
```

or using MPI:

```
mpirun -np 2 alpspython scriptname.py
```

On some machines it may be necessary to use

```
mpirun -np 2 bash alpspython scriptname.py
```

instead. Note that on some platforms mpirun has to be replaced by a different command to invoke MPI. See your MPI manual for details.

The above example and more advanced examples on how to use the Python framework can be found in the hybridization tutorials within the `/tutorials` directory inside the ALPS installation directory:

```
/hybridization-01-python
/hybridization-02-kondo
/hybridization-03-retarded-interaction
/hybridization-04-spinfreezing
```

These tutorials illustrate how to run calculations for multiple parameters within a single script, plotting results, implementing a selfconsistency scheme, using the hdf5 interface, performing calculations with retarded interactions and for multiorbital models as well as performing simple computations with the measured observables.

IV. DETAILED INPUT DESCRIPTION

A. Hybridization function

We use the following definition of the hybridization function:

$$\Delta_{ab}(i\nu_n) = \sum_{kj} \frac{V_k^{aj} V_k^{*jb}}{i\nu_n - \epsilon_k^j} \tag{1}$$

$$\Delta(\tau) = \frac{1}{\beta} \sum_{n=-\infty}^{\infty} \Delta(i\nu_n) e^{-i\nu_n \tau} \tag{2}$$

where in the current implementation, the hybridization function is restricted to be diagonal, $\Delta_{ab}(\tau) = \Delta_a(\tau)\delta_{ab}$. As a consequence of this definition, the hybridization function is always negative. The code will throw a corresponding exception if this is not the case.

The hybridization function is read from a text file the name of which is specified by the parameter `DELTA`. The file has to contain a column with imaginary time values (or the time index, the actual values are ignored) and a column of hybridization values for each orbital. The number of lines has to match the number of time points $N_\tau + 1$ used in the imaginary-time Green's function measurement, where N_τ is specified by the parameter `N_TAU`.

The code will attempt to read the hybridization function data from an hdf5 archive if the optional parameter `DELTA_IN_HDF5` is set to 1. In this case the data has to be provided as a collection of one-dimensional arrays in the paths `/Delta_{i}`, one for each orbital i ($i = 0, \ldots, N_{\text{orb}} - 1$). Each array has length $N_\tau + 1$ and the n-th entry corresponds to the value of the hybridization function at time $\tau_n = n\beta/N_\tau$ ($n = 0, \ldots, N_\tau$). That the hybridization function has been read in correctly can be verified from the screen output of the solver immediately after starting the simulation.

Note that the level energies, or double counting terms should not be absorbed into the hybridization function. They have to be specified separately, see Sec. IV C.

In cases where the hybridization function becomes extremely small, of the order of the machine accuracy, numerical instabilities will occur, which render the results useless, since the algorithm operates on the *inverse* of the matrix of hybridization functions $[\Delta(\tau_i - \tau'_j)]^{-1}$. This may happen within a DMFT calculation deep in the insulating phase and at very low temperature, or when an orbital is completely filled (or empty). These cases however are easily identified at the level of the Green's function, which will exhibit a lot of noise or even may change sign.

B. Retarded interaction function

A retarded interaction occurs in models with phonons, when dynamical screening is considered, or in the context of extended dynamical mean-field theory.

The general algorithm for phonons is given in Ref. 5. The formalism for retarded interactions is described in Ref. 6. The function $K(\tau)$, corresponding to the twice-integrated retarded interaction (for details see Ref. 6) connects all pairs of creation and annihilation operators. It is symmetric, $K(-\tau) = K(\tau) = K(\beta - \tau)$, and hence is tabulated as a function $K(\tau)$ on $N_\tau + 1$ points between 0 and β. K is positive and equal for all orbitals. For the measurement of the improved estimator, the integrated retarded interaction, or equivalently, the first derivative of $K(\tau)$, i.e. $K'(\tau)$, is required. While in principle $K'(\tau)$ can be obtained from the knowledge of $K(\tau)$, the computation from finite differences on the imaginary time grid is not reliable. Hence the derivative should also be precomputed and be provided in tabulated form.

Providing the retarded interaction function and its derivative is optional. This feature is enabled by providing a filename through the parameter `RET_INT_K`. $K(\tau)$ and $K'(\tau)$ are provided either through a text file, or, if the (optional) parameter `K_IN_HDF5` is set to 1, within an hdf5 archive. The file format in both cases is similar as for the hybridization function. The text file should provide *three* columns. The first column is the time or index (the actual value is ignored), the second column is the tabulated value of $K(\tau)$ and the third column gives the corresponding value of $K'(\tau)$. Likewise, the array in

the hdf5 file should provide the $N_\tau+1$ values of K and K' within the paths `Ret_int_K` for $K(\tau)$ and `Ret_int_Kp` for $K'(\tau)$. Note that there is no orbital dependence because the retarded interaction couples to all orbitals equally.

Note that a retarded interaction causes a static shift of the interaction and chemical potential. This shift is given by $-2K'(0^+)$ and $-K'(0^+)$, respectively, and is automatically applied using the provided derivative $K'(\tau)$. Hence the static part of the interaction and the chemical potential as provided in the input file through `U` and `MU` or through the corresponding files correspond to the unshifted values.

C. Orbital chemical potential / double counting terms

In the simplest case, the level energies of all levels are the same, and a parameter `MU` sets them to that value. Note that in this code, there is no shift of the chemical potential μ to half filling: For a single impurity Anderson model with interaction U, half filling is at $\mu = U/2$.

Double counting terms or crystal field splittings enter the code as orbital-dependent chemical potentials. They can be read in from a file, which is specified as `MU_VECTOR`. In the file the level energies should be listed consecutively, in the order of the orbitals. For example, a two-orbital case with a small magnetic field $H = 0.02$, half filled, for an interaction $U = 4$, could be specified like this:

```
1.99 2.01
```

After reading the on-site level energy it is printed to the standard output, as:

```
chemical potential with 2 orbitals:
1.99 2.01
```

The chemical potential can also be provided in the form of an hdf5 archive. To this end, set `MU_IN_HDF5=1`. The level energies should be stored in a single array in the file path `/MUvector`, in the order of the orbitals.

D. General density-density interaction matrices

In the simplest case, repulsion terms are U. This is the default case that is chosen if only the parameter `U` is specified.

In multiorbital problems, more complicated interactions are needed. In this case, the "orbital index" i, which runs from 0 to $N_{\mathrm{orb}}-1$ used in the code represents a combined spin-orbital index $i = \{\sigma, m\}$. For example, for a model with two physical orbitals and two spins, we have $N_{\mathrm{orb}} = 4$. The code in general makes no assumption on the order of spins and physical orbitals. We recommend ordering these as follows: $1\uparrow, 1\downarrow, 2\uparrow, 2\downarrow$, which correspond to orbital indices $i = 0, 1, 2, 3$, respectively. The values of the interaction matrix can be specified in a file `U_MATRIX`, which has $N_{\mathrm{orb}} \times N_{\mathrm{orb}}$ entries specifying the matrix elements U_{ij} of the interaction term $\frac{1}{2}\sum_{ij} U_{ij} n_i n_j$.

To see how the interaction matrix for the above ordering is built consider the following Hamiltonian:

$$\begin{aligned} H_U = &\frac{1}{2}U \sum_{m,\sigma} n_{m\sigma} n_{m,-\sigma} \\ &+ \frac{1}{2}U' \sum_{m\neq m',\sigma} n_{m,\sigma} n_{m',-\sigma} \\ &+ \frac{1}{2}(U'-J) \sum_{m\neq m',\sigma} n_{m,\sigma} n_{m',\sigma}, \end{aligned} \tag{3}$$

$$\hat{U} = \begin{pmatrix} 0 & \mathrm{U} & \mathrm{U'\text{-}J} & \mathrm{U'} \\ \mathrm{U} & 0 & \mathrm{U'} & \mathrm{U'\text{-}J} \\ \mathrm{U'\text{-}J} & \mathrm{U'} & 0 & \mathrm{U} \\ \mathrm{U'} & \mathrm{U'\text{-}J} & \mathrm{U} & 0 \end{pmatrix} \tag{4}$$

Choosing $U = 4.0$, $U' = 3.6$ and $J = 0.2$, the file generating the interaction matrix of the example above is:

```
0.0 4.0 3.4 3.6
4.0 0.0 3.6 3.4
3.4 3.6 0.0 4.0
3.6 3.4 4.0 0.0
```

After reading the interactions, they are printed to the standard output like this:

```
U matrix with 4 orbitals:
0 4 3.4 3.6
4 0 3.6 3.4
3.4 3.6 0 4
3.6 3.4 4 0
chemical potential with 4 orbitals:
5.5 5.5 5.5 5.5
```

Since the above Hamiltonian is widely used, it is directly into the code. To use it, simply specify the parameters U,U' and J in the input file: adding the block

```
U =  4
U'=3.6
J=0.2
```

produces the same output as above. For a single physical orbital, it reduces to the Hubbard interaction $Un_\uparrow n_\downarrow$. Note that when using the built-in Hamiltonian, the order of spins and orbitals *is* relevant and has to be chosen as recommended above. For these interaction matrices, the condition for half-filling reads $\mu_{1/2} = \frac{1}{2}\sum_i \hat{U}_{ij}$ (which is independent of j). For the above example, $\mu_{1/2} = 5.5$. The interaction matrix will be read from an hdf5 archive if `UMATRIX_IN_HDF5=1`. The N_{orb}^2 values should be stored in the file path `/Umatrix` in a one-dimensional array, where U_{ij} is stored at position $iN_{\mathrm{orb}} + j$, i.e. elements within a row are contiguous.

V. DETAILED PARAMETER DESCRIPTION

A. Mandatory parameters

The following parameters are required for any simulation. The code will not work unless all of them are provided.

- `N_MEAS`={*natural number*} specifies how many updates need to be done between measurements. For e.g. `N_MEAS=50`, 50 modifications of the Monte Carlo configuration (such as segment insertions and removals) are attempted between measurements.
- `SWEEPS`={*natural number*}is the total number of Monte Carlo sweeps. A sweep consists of `N_MEAS` attempted modifications of the Monte Carlo configuration and a single the measurement of the observables. The code exits either after it has done at least `SWEEPS` Monte Carlo sweeps or after the maximum time has been reached.
- `MAX_TIME`={*natural number*} is the maximum runtime of the code, in seconds.
- `THERMALIZATION`={*natural number*} is the number of thermalization steps. These are steps that are done at the beginning that are needed to reach the equilibrium distribution.
- `N_ORBITALS`={*natural number*} is the number of (spin-)orbitals. For the single impurity Anderson model this is two (spin up and spin down), Cerium would have 14.
- `SEED`={*natural number*} is the random number seed.
- `N_TAU`={*natural number*} specifies the number N_τ of τ-points on which the Green function is measured, and also indicates the number of τ-points on which the hybridization function and the retarded interaction are specified. The imaginary time at index n is given by $\tau_n = n\beta/N_\tau$, where $n = 0, \ldots, N_\tau$, so that the first point (index $n = 0$) specifies $G(\tau = 0)$, the last point (index $n = N_\tau$ specifies $G(\tau = \beta)$. Note that the number of imaginary time points hence is $N_\tau + 1$.
- `N_HISTOGRAM_ORDERS`={*natural number*} is the maximum expansion order to which the order histogram is measured.
- `DELTA`={*filename*} specifies the name of the file which contains the hybridization function input data (in text format).
- `MU`={*real number*} is the chemical potential or level energy for model Hamiltonians. On-site level energies can be specified differently (in particular in an orbital dependent way, see section on double counting). This parameter is ignored if a file with the chemical potential values is specified using the parameter `MU_VECTOR`.
- `U`={*real number*} is the on-site interaction. In this case it is a term $\sum_i n_{i\uparrow} n_{i\downarrow}$. As in the case of the chemical potential, this can be chosen differently, see section on double counting.
- `BETA`={*real number*} specifies the inverse temperature $\beta = T^{-1}$.

B. Optional parameters

1. Physical parameters

These parameters are explained in Sec. IV D.

- `J`={*real number*} specifies the interaction parameters J.
- `U'`={*real number*} specifies the interaction parameters U'.

2. Measurement control parameters

The following optional parameters control the different measurements implemented in the solver. For details on those measurements, see Sec. VI.

- `MEASURE_time`={0,1} activates the Green's function measurement in imaginary time. Note: This measurement is turned on by default.
- `MEASURE_freq`={0,1} activates the Green's function measurement on Matsubara frequencies. Requires `N_MATSUBARA`.
- `N_MATSUBARA`={*natural number*} The number N_ν of (fermionic) Matsubara frequencies $\nu_n = (2n + 1)\pi/\beta$. The Green's function will be measured for all frequencies with $n = 0, \ldots, N_\nu - 1$.
- `MEASURE_legendre`={0,1} activates the measurement of coefficients of Green's function in the Legendre polynomial basis. Requires `N_LEGENDRE` and `N_MATSUBARA`.
- `N_LEGENDRE`={*natural number*} specifies the number of Legendre coefficients to be measured. Coefficiencts with indices $l = 0, \ldots, N_l - 1$ will be measured.
- `MEASURE_nn`={0,1} controls the measurement of the equal-time density-density correlation function.
- `MEASURE_nnt`={0,1} turns on or off the measurement of the time dependent density-density correlation function. Requires `N_nn`.

- `N_nn`={*natural number*} specifies the number of imaginary time points for which the density-density correlation function is measured.
- `MEASURE_g2w`={0,1} turns on the measurement of the two-particle Green's function. Requires `N_w2` and `N_W`.
- `MEASURE_h2w`={0,1} turns on the measurement of the three-particle correlator H (see Sec.??). Requires `N_w2` and `N_W`.
- `N_w2`={*natural number*} specifies the number $\tilde{N}_\nu$ of *fermionic* frequencies for the two-particle Green's function or the correlator H. These quantities are measured for frequencies with indices $n = -\tilde{N}_\nu/2, \ldots, \tilde{N}_\nu/2 - 1$.
- `N_W`={*natural number*} specifies the number N_ω of *bosonic* frequencies $\omega_m = 2\pi m/\beta$ for the two-particle Green's function or the correlator H. These quantities are measured for frequencies with indices $m = 0, \ldots N_\omega - 1$.
- `MEASURE_nnw`={0,1} turns on the measurement of the susceptibility in frequency. Requires `N_W`.
- `MEASURE_sector_statistics`={0,1} controls the measurement of the sector statistics.

3. *Optional control parameters*

- `SPINFLIP`={0,1} specifies whether to perform spin flip updates. In such an update a segment is moved from one orbital to another, if the latter is empty.
- `COMPUTE_VERTEX`={0,1} specifies whether the reducible impurity vertex function should be calculated. This parameter requires `MEASURE_g2w` or `MEASURE_h2w` or both be set to 1.
- `TEXT_OUTPUT`={0,1} specifies if the simulation results should be written to text-files in human-readable format (in addition to the hdf5 output).
- `VERBOSE`={0,1} When turned on, some additional information (e.g. which measurements are turned on) is displayed when starting the simulation.
- `OUTPUT_PERIOD`={natural number} If `VERBOSE=1`, it specifies the number of sweeps after which simulation details of the master thread (acceptance ratios etc.) are printed to `stdout`. The default value is 100 000.
- `DELTA_IN_HDF5`={0,1} specifies whether the hybridization function to be read from an hdf5 archive. The data must be stored in the path `/Delta`. For details on the file format, see Sec. IV A.
- `MU_VECTOR`={filename} specifies the file name for a text file containing the orbital dependent chemical potentials. The file format is such that the chemical potentials for all orbitals are listed consecutively, separated either by a space or a new line. If this parameter is specified, the parameter `MU` is ignored.
- `MU_IN_HDF5`={0,1} specifies whether the chemical potential should be read from an hdf5 archive. The data must be stored in the path `/MUvector` as an array of length N_{orb}. See Sec. IV C for more information.
- `U_MATRIX`={filename} specifies the file name for a text file containing the interaction matrix. For the file format, refer to Sec. IV D. If this parameter is specified, the parameter `U` is ignored.
- `UMATRIX_IN_HDF5`={0,1} specifies whether the interaction matrix should be read from an hdf5 archive. The data must be stored in the path `/Umatrix`.For the file format, see Sec. IV D.
- `RET_INT_K`={*filename*} specifies the filename for the retarded interaction function K. When specified, this feature is automatically turned on.
- `K_IN_HDF5`={0,1} forces the retarded interaction to be read from an hdf5 archive. Overrides the filename given in `RET_INT_K`. The data must be stored in the path `/Ret_int_K`. For the format, consult Sec. IV B.
- `BASENAME`={*name*} changes the name of the output file when the Python module is used. The file will be called *name*.out.h5 .

VI. DETAILED DESCRIPTION OF MEASUREMENTS

In this section, we describe the measurements supported by the code in detail. That is, we state what is measured exactly, and give the precise definition of the observable that we use, so that the code can be easily integrated into your projects. We further give some recommendations on how to use these measurements.

All measurements except for the one for the imaginary-time Green's function are performed once after `N_MEAS` attempted configuration changes.

A. Imaginary-time Green's function

We define the imaginary time Green's function as follows:

$$G_i(\tau) \coloneqq -\langle T_\tau c_i(\tau) c_i^\dagger(0^+)\rangle \tag{5}$$

The code measures this function by binning it on an equidistant grid on the interval $[0, \beta]$ with the number

of bins N_τ specified by the parameter `N_TAU`. The corresponding improved estimator

$$F_i(\tau) := -\frac{1}{2}\sum_j (U_{ij}+U_{ji})\langle T_\tau n_j(\tau)c_i(\tau)c_i^\dagger(0^+)\rangle \quad (6)$$

is measured automatically. The bins of width β/N_τ are centered around the equidistant grid points at $\tau_n = n\beta/N_\tau$. Hence $\tau_0 = 0$ and $\tau_{N_\tau} = \beta$. Note that because of this choice, the first and the last bin only have half the width of a regular bin. As a consequence of the above definition, the density is given by $-G(\beta - 0^+) = G_i(\tau = 0^-) = -\langle T_\tau c_i(0^-)c_i^\dagger(0^+)\rangle = +\langle c_i^\dagger(0^+)c_i(0^-)\rangle = \langle c_i^\dagger c_i\rangle = \langle n_i\rangle$.

This measurement is very efficient and the performance of the algorithm is hardly influenced by number of bins, which hence can be chosen large. We recommend to use a number of bins of the order of at least $\sim 5\beta U$. The measurement is turned on by default, but can nevertheless be turned off by setting `MEASURE_time=0`.

The raw data of the measurement is stored in the hdf5 output file in the path `/simulation/results/g_i/mean/value`, where i is the orbital index. Note that the data at the interval endpoints has less statistics because of the smaller bin size. When using the raw data, the values at the interval endpoints have to be multiplied by two (since the effective bin size at the boundary is half of the regular size) to get the correct value. The data for all orbitals is stored in a single one-dimensional array, where successive τ-points for a given orbital are consecutive in memory. The Green's function at time τ_n, $n = 0, \ldots, N_\tau$ and orbital $i = 0, \ldots, N_{\mathrm{orb}} - 1$ is hence stored at position $i(N_\tau + 1) + n$. During post processing, the data for $G(\tau)$ is written to the path `/G_tau/0/mean/value` for orbital 0 and likewise for the other orbitals in the output file. The values at the interval endpoints are thereby replaced by the corresponding values inferred from the densities. When `TEXT_OUTPUT=1`, the processed Green's function data is written to a human-readable text file `Gt.dat`, which allows for easy plotting, e.g. with gnuplot. The first column is the imaginary time and successive columns are listed in the order of the orbitals.

B. Frequency Space Measurements

The code measures the Fourier transform of Green's function

$$G(i\nu_n) = \int_0^\beta d\tau G(\tau)e^{i\nu_n\tau}, \quad (7)$$

where $G(\tau)$ is defined as in (5).

Measurements in frequency space are relatively expensive, as $\exp(i\nu_n(\tau_i - \tau_j))$ has to be evaluated at each measurement for each pair of operators (values for successive frequencies are generated by multiplication of these exponentials). Frequency space measurements are enabled by setting `MEASURE_freq=1` and specifying `N_MATSUBARA=512` for $N_\nu = 512$ Matsubara frequencies. The Green's function is measured for frequencies ν_n with indices $n = 0, \ldots, N_\nu - 1$. The number of frequencies compatible with our above recommendation for the number of time slices is $N_\nu \sim 5\beta U/2\pi$.

The improved estimators $F_i(i\nu_n)$ described in Ref. 7 only cause a small overhead and are automatically measured (using the same definitions as given in that reference). The resulting $F(i\nu_n)$ and $G(i\nu_n)$, as well as the self-energy $\Sigma(i\nu_n)$, are stored in the hdf5 file under `/G_omega`, `/F_omega`, and `/S_omega`, respectively, where the actual data is in the subpath `/i/mean/value` for orbital $i \in 0, \ldots, N_{\mathrm{orb}} - 1$. The data for each of these quantities is stored in *two*-dimensional array with dimensions $N_\nu N_{\mathrm{orb}} \times 2$. The index of the first dimension specifies the location $iN_\nu + n$ of the complex value for frequency ν_n and orbital i, and the second index ($\{0, 1\}$) selects real or imaginary part. The raw data for $G(i\nu_n)$ and $F(i\nu_n)$ is stored in path `simulation/results/<g/f>w_<re/im>_i`, where, e.g., the subpath is `gw_re_0` for the real part of G in orbital 0.

If text output is enabled (parameter `TEXT_OUTPUT=1`), G, F, and Σ are also written to the text files `Gw.dat`, `Fw.dat`, and `Sw.dat`, respectively, where the first column is the Matsubara frequency and the i-th *pair* of columns lists real and imaginary part of the i-th orbital.

C. Legendre Polynomial Measurements

The measurements can also be done using Legendre polynomials – a method recently introduced by Boehnke *et al.*[8]. The code measures the coefficients of the expansion of Green's function in terms of orthogonal Legendre polynomials. These coefficients are defined as

$$G_l = \int_0^\beta d\tau G(\tau)P_l(x(\tau)) \quad (8)$$

where $x(\tau) = 2\tau/\beta - 1$. Note that this definition for the Legendre coefficients differs from the one in Ref. 8 by a factor $\sqrt{2l+1}$, which we omit to save time during the simulation and reintroduce during post processing. Legendre measurements are enabled by specifying `MEASURE_legendre=1` and the number of Legendre coefficients N_l through `N_LEGENDRE`. The improved estimators F[7] are measured automatically.

The raw data (i.e. the coefficients) for orbital i are stored in the hdf5 output file in the path `/simulation/results/gl_i` and `/simulation/results/fl_i`. The actual values are stored in a one-dimensional array in the subpath `/mean/value`. After the simulation, the quantities $G(\tau)$, $F(\tau)$ as well as $G(i\nu_n)$, $F(i\nu_n)$ and $\Sigma(i\nu_n)$ are evaluated from the Legendre coefficients and stored in the

paths `G_l_tau`, `F_l_tau`, `G_l_omega`, `F_l_omega` and `S_l_omega`, respectively. The values actual in orbital i are found within the one-dimensional array stored in the subpath `/i/mean/value`. For the evaluation, the number of imaginary time points and Matsubara frequencies is determined from the parameters `N_TAU` and `N_MATSUBARA`. If text output is enabled (parameter `TEXT_OUTPUT=1`), G, F, and Σ are also written to the text files `Gtl.dat`, `Ftl.dat`, `Gwl.dat`, `Fwl.dat`, and `Swl.dat`.

This measurement yields comparatively smooth curves, but as the results for different frequencies are correlated a precise error estimate and accurate control of the number of legendre coefficients considered is important.

The number of Legendre coefficients required for the accurate representation of a given observable may depend on the quantity under consideration and is difficult to infer from looking at the coefficients themselves. Rather one should analyze the dependence of a given observable on the basis cutoff N_l. To this end, one can look at the files `Gl_conv.dat` and `Fl_conv.dat` which are written when text output is enabled. In the former, the Green's function at $\tau = 0$ and $\tau = \beta/2$ is written as a function of the basis cutoff up to the maximum cutoff N_l. The latter contains the same for the correlator F. To get a feeling for the required cutoff, one should run a simulation at the desired parameters with a large cutoff N_l and plot the data in these files. For a converged calculation and sufficiently large N_l, one should see an extended plateau, where the observable does not change as a function of the cutoff. For less coefficients, the expansion is not converged and for a larger number, Monte Carlo noise enters. The basis cutoff is well defined for an N_l within the plateau.

Recently, it has been proposed to improve this method by employing the Kernel polynomial method[9]. We do not provide an implementation here. While the method does damp spurious oscillations in the Green's function, it leads to rather larger systematic errors in the moments.

D. Density

The densities are always measured. They are stored in the hdf5 file in path `/simulation/results/density_i`. If text output is enabled, they are written to the file `observables.dat`.

E. Susceptibility – imaginary time

The time-dependent density-density correlation function, $\chi_{ij}(\tau) = \langle n_i(\tau) n_j(0) \rangle$, or susceptibility, is measured by setting `MEASURE_nnt=1`. Only components with $j \leq i$ are measured. The spin and charge susceptibilities can be obtained as linear combinations of these correlation functions. The number of imaginary time points can be different from the one for Green's function and has to be specified through the mandatory parameter `N_nn`. The imaginary-time grid is otherwise the same as for Green's function. Note however, that because of the way the algorithm works, these functions are not binned, but measured exactly at the grid points. Hence the raw data does not have to be rescaled at the interval endpoints (cf. Sec. VI A). The raw data is stored in the hdf5 output file in the path `simulation/results/nnt_i_j`, within the subpath `/mean/value`. It is additionally written to the path `/nnt_i_j` without modification. When text output is enabled, the data is written to the text file `nnt.dat` for all $j \leq i$. The imaginary time is listed in column 1 and the susceptibility $\chi_{ij}(\tau)$ resides in column $1 + j + \frac{1}{2}i(i+1)$, i.e. χ_{00} column 1, χ_{10} in column 2 and χ_{11} in column 3, etc. Alternatively to the imaginary time measurement, the susceptibility can be measured directly in frequency (see the following section).

F. Susceptibility – frequency

In the frequency measurement for the susceptibility, the algorithm computes

$$\chi_{ij}(i\omega_m) = \int_0^\beta d\tau \chi_{ij}(\tau) e^{i\omega_m \tau} \tag{9}$$

where $\omega_m = 2m\pi/\beta$. Note that this measurement does not involve any time discretization. It is particularly useful and fastest if only the static susceptibility $\chi_{ij}(\omega = 0)$ is desired. In most cases, i.e. if the perturbation order is not exceptionally large, it is still faster for a finite number of frequencies compared to the imaginary-time measurement (using N_τ time points and an "equivalent" number of frequency points $N_\omega \sim N_\tau/2\pi$, respectively). Only the real part is measured, since the function is symmetric around $\beta/2$. The measurement is turned on with `MEASURE_nnw=1`. The number of bosonic frequencies is determined by the (mandatory) parameter `N_W`. Note that the same parameter determines the number of bosonic frequencies for measurements of two-particle correlation functions (cf Sec. VI I). The raw data is stored in the hdf5 output file in the path `simulation/results/nnw_re_i_j`, within the subpath `/mean/value`. It is additionally written to the path `/nnw_re_i_j` without modification. When text output is enabled, the data is written to the text file `nnw.dat` for all $j \leq i$. The frequency is listed in column 1 and the susceptibility $\chi_{ij}(\omega)$ resides in column $1 + j + \frac{i}{2}(i+1)$.

G. Equal-time density-density correlation functions

The measurement for the density-density (equal-time) correlation functions $\langle n_i n_j \rangle$ is turned on by letting `MEASURE_nn=1`. Although these are a special case of the time-dependent correlation functions above, we provide a separate measurement. They are measured

for $j < i$ only, because the operators commute, $\langle n_i n_j \rangle = \langle n_j n_i \rangle$, and $\langle n_i n_i \rangle = \langle n_i^2 \rangle = \langle n_i \rangle$ would yield the density. The raw data is stored in the path `/simulation/results/nn_i_j`. No further post processing occurs. When text output is enabled, the values are appended to the file `observables.dat`.

H. Sector statistics measurement

The sector statistics give information on the fraction of (imaginary) time the impurity spends in a given state. The total number of possible states for this algorithm is $2^{N_{\mathrm{orb}}}$. We can represent any of these states in the form

$$|n_0 = \{0,1\} n_1 = \{0,1\} \dots n_{N_{\mathrm{orb}}-1} = \{0,1\}\rangle \qquad (10)$$

To be specific, for a two-orbital impurity, there are four different states: The impurity can be unoccupied, which corresponds to the state $|00\rangle$. Orbital 0 (spin up, say) can be occupied while orbital 1 (spin down) is empty, $|10\rangle$, or orbital 1 can be occupied with 0 being empty $|01\rangle$. Finally, the impurity can be doubly occupied, i.e. in state $|11\rangle$. The result of a calculation for such a system at half-filling could look as follows:

```
#state |n_1={0,1} n_2={0,1} ...> n_i={0,1}: \
orbital i {empty,occupied}
#rel weight (in %)
0 18.9302 |00>
1 31.0485 |10>
2 31.0745 |01>
3 18.9467 |11>
```

To understand this, recall that in the atomic limit (no hybridization), the energy is given by $Un_\uparrow n_\downarrow - \mu(n_\uparrow + n_\downarrow)$. Hence, at half filling, where $\mu = U/2$, the energy of the singly occupied states is $-\mu$ and hence they are degenerate. The unoccupied and doubly occupied states are degenerate as well, with energy 0 and $U - 2\mu = 0$, respectively. Hence their contributions should be equal (within the Monte Carlo error). The singly occupied states are lower in energy and so their contribution is higher. Since the model is particle-hole symmetric, we further know that the time spent in states $|00\rangle$ and $|01\rangle$ must be the same as for $|10\rangle$ and $|11\rangle$, which can readily be seen from the data. The fraction the impurity spends in state $|11\rangle$ corresponds to the integrated overlap and hence should be equal to the equal-time correlation function $\langle n_\downarrow n_\uparrow \rangle$, as can be verified from the file `observables.dat` (if `MEASURE_nn=1`):

```
n0=0.499892;
n1=0.500191;
n1n0=0.189423;
```

Note that the values are slightly different since they are measured in different ways. They will converge to one another as statistics is increased.

The data is stored in the hdf5 output file in the path `simulation/results/sector_statistics/mean/value` in a vector of length $2^{N_{\mathrm{orb}}}$. The position of a given state in this vector is simply the decimal representation of the binary number encoding the state, i.e. the state $|01\rangle$ has the index $0 \cdot 2^0 + 1 \cdot 2^1 = 2$. With text output enabled, a file `sector_statistics.dat` as shown above is produced.

I. Two-particle correlation functions

The use of two-particle correlation functions is gaining importance. They are used for calculating lattice susceptibilities within dynamical mean-field theory or novel diagrammatic extensions of DMFT. To meet these requirements, we provide a measurement for the two-particle Green's function:

$$G^{(2)}_{ij}(\tau_a, \tau_b, \tau_c, \tau_d) \coloneqq \langle c_i(\tau_a) c_i^\dagger(\tau_b) c_j(\tau_c) c_j^\dagger(\tau_d) \rangle \qquad (11)$$

where $G^{(2)}_{ij}$ is a shorthand notation for $G^{(2)}_{iijj}$. As a result of the fact that the hybridization is diagonal, the Green's function depends only on two independent orbital indices. An annihilator on orbital i has to come in pair with a creator on the same orbital, otherwise the average is zero. To generate the orbital combinations $G^{(2)}_{ijji}$, relations of the form $G^{(2)}_{ijji}(\tau_a, \tau_b, \tau_c, \tau_d) = -G^{(2)}_{iijj}(\tau_a, \tau_d, \tau_c, \tau_b)$ can be used (and the corresponding ones in frequency space). These functions are measured only for $j \leq i$, since measuring this quantity with indices i and j interchanged would yields exactly the same result, i.e. there is no gain through averaging. The diagonal quantities ($i = j$) are different for different orbitals even if they are degenerate due to the sampling error, so that they can (and should) be averaged if the orbitals are symmetry equivalent.

Due to time translational invariance, the function depends on three independent time differences, or equivalently three independent frequencies. Because of memory restrictions, the actual measurement is performed in frequency space, for which we use the following definition of the Fourier transform:

$$\begin{aligned} G^{(2)}_{ij}(\nu, \nu', \omega) \coloneqq \frac{1}{\beta} \int_0^\beta d\tau_a \int_0^\beta d\tau_b \int_0^\beta d\tau_c \int_0^\beta d\tau_d \quad (12) \\ \times G^{(2)}_{ij}(\tau_a, \tau_b, \tau_c, \tau_d) \\ \times e^{i(\nu+\omega)\tau_a} e^{-i\nu\tau_b} e^{-i\nu'\tau_c} e^{-i(\nu'+\omega)\tau_a} \end{aligned}$$

Here $\nu \equiv \nu_n = (2n+1)\pi/\beta$ and ν' are fermionic frequencies, whereas $\omega \equiv \omega_m = 2m\pi/\beta$ is bosonic. The number of fermionic frequencies $N^{(2)}_\nu$ is specified through the parameter `N_w2` and the number of bosonic ones (N_ω) through `N_W` (mind the capital "W"). The vertex is measured for fermionic frequencies with indices $n = -N_\nu/2, \dots, N_\nu/2 - 1$ and indices $m = 0, \dots N_\omega - 1$ for the bosonic frequency. For negative bosonic frequencies, the relation $G^{(2)}_{ij}(\nu, \nu', -\omega) = G^{(2)*}_{ij}(-\nu, -\nu', \omega)$ holds.

The real and imaginary part of the two-particle Green's function are stored in the hdf5 output file in the path `simulation/results` as `g2w_re_i_j` and `g2w_im_i_j`, respectively ($j \leq i$). The actual data is stored in a one-dimensional array of dimension $N_\nu^{(2)} N_\nu^{(2)} N_\omega$ in the subpath `/mean/value`. The value corresponding to frequencies ν_n, $\nu'_{n'}$ and ω_m is stored at position $m(N_\nu^{(2)})^2 + (n + N_\nu^{(2)}/2)N_\nu^{(2)} + (n' + N_\nu^{(2)}/2)$ (note that $n = -N_\nu^{(2)}/2, \ldots, N_\nu^{(2)}/2 - 1$). When text output is enabled, the two-particle Green's function is written to the file `g2w.dat`. The format allows to easily plot the data as a function of a single frequency using gnuplot, for example:

```
gnuplot> p "<cat g2w.dat |grep 'wp: 1 '|\
grep 'W: 1 '| grep 'i: 0 j: 0' " u 2:11 w lp
```

(note the whitespace in `'W: 1 '`). Column 11 is the real part, column 12 the imaginary part.

Due to the possibly large number of observables (frequencies) that have to be measured, this measurement may significantly slow down the code, or even exceed memory available to each process. In such cases, it may be desirable to use additional symmetries, if present, or altered frequency meshes. In addition, different methods may have different requirements and for these reasons, it is impossible to provide a one-fits-all implementation. However, the advanced user may regard the function `void hybridization::measure_G2w` in the source file `hybmeasurements.cpp` as an API and modify it according to her needs.

The code may also be used to calculate the vertex function of the impurity model, which is defined as

$$\gamma_{ij}(\nu, \nu', \omega) = \frac{G_{ij}^{(2)}(\nu, \nu', \omega) - \beta[G_i(\nu + \omega)G_j(\nu')\delta_{\omega,0} - \delta_{ij}G_i(\nu + \omega)G(\nu')\delta_{\nu\nu'}]}{G_i(\nu + \omega)G_i(\nu)G_j(\nu')G_j(\nu' + \omega)}. \tag{13}$$

We provide the measurement of the correlation function

$$H_{ij}^a(\tau_1\tau_2\tau_3\tau_4) \coloneqq \langle n_a(\tau_1)c_i(\tau_1)c_i^\dagger(\tau_2)c_j(\tau_3)c_j^\dagger(\tau_4)\rangle, \tag{14}$$

which represents an improved estimator for the vertex function[7] and will be measured in frequency if `MEASURE_h2w` is set to one. The vertex function is computed during post processing, if `COMPUTE_VERTEX=1`. In addition, the frequency measurement (`MEASURE_freq`) and at least one of the measurements `MEASURE_g2w` or `MEASURE_h2w` (or both) have to be turned on. The vertex will be evaluated according to the data available. Determining the vertex from $G^{(2)}$ only is least accurate. Using $G^{(2)}$ and H is the most accurate. The vertex may also be calculated from H only. This is somewhat less accurate, but saves a factor of 2 in memory. The number of frequencies for the single-particle Green's function that enters Eq. (13), has to fulfill the relation $N_\nu \leq N_\nu^{(2)} + N_\omega - 1$, otherwise the code will not work. There are different ways to evaluate the vertex. Depending on which of the latter two measurements are turned on, the method that yields the most accurate results will be chosen. The function H is stored analogously to the two-particle Green's function, as `h2w_re_i_j` and `h2w_im_i_j`, respectively and written to file `h2w.dat` when text output is enabled. When evaluated, the vertex function is written to the hdf5 output file in the path `/vertex_i_j` in a *two*-dimensional array with dimensions $N_\nu^{(2)} N_\nu^{(2)} N_\omega \times 2$. The index of the first dimension specifies the postion $m(N_\nu^{(2)})^2 + (n + N_\nu^{(2)}/2)N_\nu^{(2)} + (n' + N_\nu^{(2)}/2)$ of the complex value for frequencies ν_n, $\nu'_{n'}$ and ω_m and the second index ($\{0, 1\}$) selects real or imaginary part. With text output enabled. the vertex is written to the file `gammaw.dat`, in the same format as the two-particle Green's function.

Note that this vertex function is the full (i.e. reducible) vertex of the impurity model. If the irreducible or even fully irreducible vertex is desired, this has to be obtained by inverting a Bethe-Salpeter equation or from inverse Parquet, respectively.

VII. DMFT – SPECIFYING A SELF-CONSISTENCY

To run the impurity solver in a DMFT self-consistency, the output $G(\tau)$ has to be input into a self-consistency condition and a hybridization function $\Delta(\tau)$ has to be produced.[10,11] In the simplest case, for a DMFT self-consistency for the Bethe lattice with infinite coordination number (resulting in a semicircular density of states), we obtain $\Delta(\tau) = t^2 G(\tau)$. Note that in our code, both Δ and $G(\tau)$ are negative for $\tau > 0$. In practice, more elaborate self-consistency conditions are almost always needed. Some of them are included in the ALPS DMFT framework, but in general they are user provided.

A. ALPS DMFT framework

The hybridization solver is used in several ALPS tutorials, which illustrate the use of the solver with the ALPS DMFT framework. This comprises the tutorials in the following directories:

```
dmft-01-intro
dmft-02-hybridization
dmft-04-mott
```

```
dmft-05-osmt
dmft-06-paramagnet/hyb
dmft-08-lattices
```

within the `tutorials` subdirectory in the ALPS installation directory.

B. Python interface

Using alpspython together with the Python interface of the solver (the cthyb Python module) provides a very flexible framework for setting up any desired selfconsistency scheme. The selfconsistency is thereby implemented within a Python script which can be executed on a parallel machine using MPI. Two examples are given in the ALPS tutorials, i.e in the directories

```
/hybridization-02-retarded-interaction
```

and

```
/hybridization-03-spinfreezing
```

For how to run these tutorials, refer to Sec. III B.

VIII. SPECIFYING MEASUREMENTS

The following variables are measured by default:

- `Sign`: The Monte Carlo fermionic Sign.
- `order_histogram_total`: The histogram of expansion orders, containing all interaction vertices.
- `order_histogram_i`: with i from 0 to $N_{\rm orb}-1$: the expansion order in each orbital
- `density_i`: density in orbital i, wiht i from 0 to $N_{\rm orb}-1$.
- `g_i`: with i from 0 to $N_{\rm orb}-1$: Green's function in imaginary time, binned on `N_TAU` time slices between zero and β.

If the frequency measurement is enabled, the following variables are measured in addition:

- `gw_re_i`: with i from 0 to $N_{\rm orb}-1$: Green's function in frequency, real part.
- `gw_im_i`: with i from 0 to $N_{\rm orb}-1$: Green's function in frequency, imaginary part.
- `fw_re_i`: with i from 0 to $N_{\rm orb}-1$: F function for improved estimators in frequency, real part.
- `fw_im_i`: with i from 0 to $N_{\rm orb}-1$: F function for improved estimators in frequency, imaginary part.

If the legendre measurement is enabled, the following variables are measured in addition:

- `gl_i`: with i from 0 to $N_{\rm orb}-1$: Legendre coefficients of Green's function G.
- `fl_i`: with i from 0 to $N_{\rm orb}-1$: Legendre coefficients of F function for improved estimators.

These observables are written into a hdf5 file after the simulation exits. The standard hdf5 tools, `h5ls -r` and `h5dump` or `h5dump -d /simulation/results/Sign/mean/value` will give their values.

For all available measurements, we refer the reader to Sec. VI.

IX. ANALYSIS AND POST-PROCESSING

After the simulation exits, the simulation results have to be read in and evaluated. This is best done using one of the many hdf5 tools. The ALPS DMFT framework provides access from `C++`, but Python tools (`h5py` and similar) also work. The hybridization tutorials show how to use the ALPS Python hdf5 interface. Typically, the Green function, its error, and the densities need to be read in and post-processed.

A. Elementary checks: Expansion order and Sign problem

Each new run of the impurity solver should start with a check of the expansion order and the error on it. Plot the results of `order_histogram_i` including error bars. Does the statistics look good? is the expansion order reasonable? If there are degenerate orbitals, is their expansion order the same within error bars? For diagonal hybridizations, the Monte Carlo sign problem should be always one. As a rule of thumb: If your sign drops below about 0.75, you have to be careful. If your sign is below 0.1, your results are most likely too noisy to be useful.

B. Further checks: Errors of Green functions

In a next step, extract the Green function from the results and analyze them. Is the statistics good enough for your purposes? Do the error bars make sense? If you use `h5py`, you can extract the Green functions in τ from an output file `sim.h5` using

```
import h5py
h5_file = h5py.File("sim.h5", 'r')
g_mean =h5_file["/simulation/results/\
g_0/mean/value"].value
g_error=h5_file["/simulation/results/\
g_0/mean/error"].value
```

Once you trust your Green functions, Fourier transform to frequency and analyze the self energy (which will be

used in the self-consistency equations). The self energies are very sensitive.

C. Fourier transforms and high frequency tails

To get smooth and correct behavior you may need to use high frequency Fourier transform tricks (see e.g. the review, Ref. 3). Further information can be found in the references cited in the review chapter X.I.

D. Running MaxEnt

The Maximum entropy method, or MaxEnt, is used to analytically continue imaginary frequency or imaginary time data to the real axis to produce spectral functions that are consistent with the imaginary time data within statistical errors.[12] ALPS provides an implementation of a maximum entropy analytic continuation code. Note that MaxEnt is completely uncontrolled: no reliable error bars for the output exist. In addition, small differences in the input data will cause large differences in the output. Because of this, any conclusion drawn from continued data needs to be backed up by data on the imaginary axis, where error bars are available.

A correct error propagation that includes both the errors and the covariance matrices of the Green's function is absolutely essential for reliable continuations. Additionally, it helps to continue the self-energies instead of the Green's functions[13].

A tutorial for the ALPS maxent code will be provided elsewhere. Fig. 1 shows a simple parameter file

X. SCALING – ACCESSIBLE PROBLEMS

The algorithm scales cubically in matrix size (expansion order), and is therefore $O(n_o\beta^3)$ with a relatively small (but complicated) dependence of U. Typically problems with an expansion order of less than 400 per orbital are easy to do.

XI. WHAT IS NOT POSSIBLE WITH THE HYBRIDIZATION CODE

You cannot apply the code for calculations in the atomic limit, i.e. for vanishing hybridization.

The following problems are not feasible at the moment:

1. Problems with non-diagonal hybridization functions
2. Problems with general, non density-density interactions

It is planned to extend the code to the first two types of impurity problems. The second type of impurity problems may be considered at a later stage.

XII. LITERATURE AND FURTHER INFORMATION

- Dynamical mean field theory: please have a look at the original review[10] and the review by Kotliar *et al.*[11]. Introductory information can also be found in Antoine George's summer school lecture notes[14].
- LDA+DMFT: Apart from Ref. 11, Refs. 15 and 16 are good places to start.
- Continuous-time methods: The review[3], along with summer school lecture notes[17] and PhD theses[18] and the original literature.[2,19,20]

XIII. WHEN PROBLEMS APPEAR – TROUBLESHOOTING

This code is still in beta stage and problems will appear. To report a problem please contact the ALPS user mailing list or Emanuel via e-mail.

XIV. PUBLISHING – CITING THE ALPS HYBRIDIZATION EXPANSION CODE

Citations are important for us – they mean that we can justify investing time into the development of the hybridization expansion code and other open source projects. A paper should acknowledge the use of the ALPS libraries, the use of the hybridization code, and the original algorithm paper. A citation line could be:

Our simulations used an open source implementation[4] of the hybridization expansion continuous-time quantum Monte Carlo algorithm[2] and the ALPS[1] libraries.

Or, alternatively:

Our simulations used the ALPS open source hybridization expansion code[1,2,4].

---

[1] B. Bauer *et al.*, Journal of Statistical Mechanics: Theory and Experiment, **2011**, P05001 (2011).

[2] P. Werner, A. Comanac, L. de' Medici, *et al.*, Phys. Rev.

```
N_ALPHA = 60 //the number of alphas between alpha min and max
ALPHA_MIN = 0.05 //smallest alpha value to be evaluated
ALPHA_MAX = 5 //largest alpha value to be evaluated
NORM = 1.0 //normalization of the function to be continued
OMEGA_MAX = 25 //frequency range, symmetric by default. specify omega_min otherwise
KERNEL = fermionic //Maxent kernel
BETA = 60.0 //inverse temperature
NFREQ = 2001 //number of output frequencies
NDAT = 512 //number of input data points
FREQUENCY_GRID = Lorentzian //frequency grid, e.g. Lorentzian, Linear, ...
DATASPACE =frequency //dataspace, e.g. imag time or frequency
MAX_IT = 2000 //number of iterations for iterative minimization procedure
DEFAULT_MODEL = "flat"
{
PARTICLE_HOLE_SYMMETRY = 1  //either the model is particle hole symmetric or not
X_0 = -1.58151510066 //data point X_0, here the lowest frequency point
X_1 = -1.19058822793
...etc...
X_511 = -0.0186270492461
SIGMA_0 = 0.00526104728966 //errors of X_0
SIGMA_1 = 0.00199036770537
...etc...
SIGMA_511 = 3.42643209543e-07
}
```

FIG. 1. A simple parameter file for the maxent code.

Lett., **97**, 076405 (2006).

[3] E. Gull, A. J. Millis, A. I. Lichtenstein, A. N. Rubtsov, M. Troyer, and P. Werner, Rev. Mod. Phys., **83**, 349 (2011).

[4] E. Gull, P. Werner, S. Fuchs, B. Surer, T. Pruschke, and M. Troyer, Computer Physics Communications, **182**, 1078 (2011), ISSN 0010-4655.

[5] P. Werner and A. J. Millis, Physical Review Letters, **99**, 146404 (2007).

[6] P. Werner and A. J. Millis, Phys. Rev. Lett., **104**, 146401 (2010).

[7] H. Hafermann, K. R. Patton, and P. Werner, Phys. Rev. B, **85**, 205106 (2012).

[8] L. Boehnke, H. Hafermann, M. Ferrero, F. Lechermann, and O. Parcollet, Phys. Rev. B, **84**, 075145 (2011).

[9] L. Huang and L. Du, "Kernel polynomial representation of imaginary-time green's functions," (2012), arXiv:1205.2791.

[10] A. Georges, G. Kotliar, W. Krauth, and M. J. Rozenberg, Rev. Mod. Phys., **68**, 13 (1996).

[11] G. Kotliar, S. Y. Savrasov, K. Haule, *et al.*, Rev. Mod. Phys., **78**, 865 (2006).

[12] M. Jarrell and J. E. Gubernatis, Physics Reports, **269**, 133 (1996), ISSN 0370-1573.

[13] X. Wang, E. Gull, L. de' Medici, M. Capone, and A. J. Millis, Phys. Rev. B, **80**, 045101 (2009).

[14] A. Georges, LECTURES ON THE PHYSICS OF HIGHLY CORRELATED ELECTRON SYSTEMS VIII: Eighth Training Course in the Physics of Correlated Electron Systems and High-Tc Superconductors, **715**, 3 (2004).

[15] K. Held, I. A. Nekrasov, G. Keller, V. Eyert, N. Bluemer, A. K. McMahan, R. T. Scalettar, T. Pruschke, V. I. Anisimov, and D. Vollhardt, Phys. Status Solidi, **243**, 2599 (2006).

[16] K. Held, Advances in Physics, **56**, 829 (2007).

[17] P. Werner, Lecture notes for the International Summer School on Numerical Methods for Correlated Systems in Condensed Matter, Sherbrooke, Canada (2008).

[18] E. Gull, *Continuous-time quantum Monte Carlo algorithms for fermions*, Ph.D. thesis, ETH Zurich (2008).

[19] A. N. Rubtsov, V. V. Savkin, and A. I. Lichtenstein, Phys. Rev. B, **72**, 035122 (2005).

[20] A. N. Rubtsov and A. I. Lichtenstein, JETP Letters, **80**, 61 (2004).