0.1 Installation

I would recommend to use my version of Cedric's solver.

In the main directory, change the file Makefile.in with your configuration. In principle, the only thing which needs to be changed is the lapack/blas library. In my case, I use openblas which can be easily installed on any GNU/Linux system.

Then run ./run_install.sh and add ed_cweber/bin in your path.

It prints many warnings... but should works and create the folder bin which contains the main executables.

If problems, please let me know

0.2 ED solver

The auxiliary system is a local version of the Hubbard model, called an Anderson impurity model(AIM) which is described by the action

$$S = \int d\tau d\tau' \sum_{mn} c_m^\dagger(\tau)\mathcal{G}_{mn}^{0,-1}(\tau-\tau')c(\tau) + \sum_{mn} U_{mn} n_{m\uparrow} n_{n\downarrow}. \tag{1}$$

or in Hamiltonian formalism,

$$H = \sum_{i\sigma} \epsilon_{i\sigma} f_{i\sigma}^\dagger f_{i\sigma} + \sum_{im\sigma} V_i f_{im\sigma}^\dagger c_{m\sigma} + V_i^* c_\sigma^\dagger f_{i\sigma} + \sum_{mn} U_{mn} n_{m\uparrow} n_{n\downarrow} - \sum_{mn\sigma} (E_{mn} - \delta_{mn}\mu) c_{m\sigma}^\dagger c_{n\sigma} \tag{2}$$

Eq. (1) and (2) are equivalent if

$$\mathcal{G}_0^{-1}(\omega) = \omega - \mu - E - \Delta(\omega) \tag{3}$$

$$\Delta_{mn}(\omega) = \sum_i \frac{V_{im} V_{in}^\dagger}{\omega - \epsilon_i}. \tag{4}$$

The AIM represents a bath of particles f_i of energy ϵ_i which can hop on an impurity with the amplitude V_i. If there are two particles on the impurity, there is a cost representing by U. In an usual AIM, the bath is a non interacting particle host; in DMFT context, this bath is designed to reproduced the rest of the system.

This problem is hard to solve since the Hibert space is infinite. There are a infinite number of electron f_i.

In ED solver, the bath f_i is discretized

$\Delta \sim \Delta^{fit} = \sum_i^{n_{bath}} \frac{V_{im} V_{in}^\dagger}{\omega - \epsilon_i}$

Once it is done, the size of the Hilbert space is 2^{2N} with $N = n_{bath} + n_{imp}$ where n_{imp} is the number of orbital of the AIM.

To diagonalize this H, two options:

- full_ed : the Hamiltonian is diagonalized completly. works for $n \leq 8$
- normal : the Hamiltonian is diagonalized with Lanczos method. (This mode could go up to $n = 12$)

So we need to feed the code with :

- The hybridization : need to be write in delta1.inp
- the local Hamiltonian with U and E : see PARAMS file
- Some information for the solver in *ED/ED.in*

0.2.1 $\Delta(\omega)$

the hybridization is contained in the files delta1.inp (spin up) and delta2.inp.

the format is :

$$\omega_1\, \Re(\Delta_{11})\, \Im(\Delta_{11})\, \Re(\Delta_{12})\, \Im(\Delta_{12})\, \Re(\Delta_{21})\, \Im(\Delta_{21})\, \Re(\Delta_{22})\, \Im(\Delta_{22})$$
$$\omega_2\, \Re(\Delta_{11})\, \Im(\Delta_{11})\, \Re(\Delta_{12})\, \Im(\Delta_{12})\, \Re(\Delta_{21})\, \Im(\Delta_{21})\, \Re(\Delta_{22})\, \Im(\Delta_{22})$$
$$....$$

0.2.2 PARAMS

The PARAMS file contains all the information about the local hamiltonian.

The format of the file is :

```
n_imp ! number of site in the impurity ( For the rest of this example , I assume that
E_11_up E_12_up
E_21_up E_22_up
E_11_dn E_12_dn
E_21_dn E_22_dn
U_11 U_12
U_21 U_22
nomg! number of matsubara frequency
nomg ! number of matsubara frequency
nomg_real ! number of point on the real axis
F ! keep it unless you know what you are doing
1 ! not used
F ! T to compute the Green function on the real axis
0 ! wmin for real axis
0 ! wmax for real axis
0 ! Chemical potential (equivalent to shift E to E-mu
100.0 ! Inverse temperature
0 ! Do not touch any numbers after this one unless you know what you are doing.
 5.0000000000000001E-004
   1.4000000000000000E-002
   1.2999999999999999E-002
   3.5999999999999997E-002
   10
0.0
 F
F
F
F
0.0000
```

Example for 1 site

```
  1
-3.0
-3.0
6.0
1000
1000
1000
```

```
F
1
F
0.0000000000000000
0.0000000000000000
0.
100
0.0    !8 UU
1.0000000000000001E-002
3.4000000000000000E-001
3.2999999999999999E-001
   7.0999999999999997E-001
          10
  0.000000000000000  !0.7 ! JJ
 F
 F
 F
 F
0.0
```

example for 2 sites $U = 10$

```
  2
-5.0 0
0 -5.0
-5.0 0
0 -5.0
10  2
2 10
1000
1000
1000
F
1
F
0.0000000000000000
0.0000000000000000
0.
50
0.0    !8 UU
1.0000000000000001E-002
3.4000000000000000E-001
3.2999999999999999E-001
   7.0999999999999997E-001
          10
  0.000000000000000  !0.7 ! JJ
 F
 F
 F
 F
0.0
```

0.3 ED.in

This file is contained in ED/ED.in

```
  FIT_METH=CIVELLI <- fit method of a very nice guy
fit_nw=100 <- number of freq to fit
min_all_bath_param=6  <- number of site in the bath
nsec0=0
nsec=-1
which_lanczos=FULL_ED <- Full diag
Neigen=3
Nitermax=600 <- ?
Nitergreenmax=400 <- ?
FLAG_DUMP_INFO_FOR_GAMMA_VERTEX=.false. <- compute transition matrices <i|c|j>
FLAG_FULL_ED_GREEN=.false.
dEmax0=10 <-  =~ Thougw away state if -b*E <dEmax0
```

0.4 basic example

```
  #create ed_correl1 and put it in ED/
  # generate_edcorrel norb nspin
generate_edcorrel 1 2 \&\& mv ed_correl1 ED/.
#run ED solver
dmft_solver
```

g1.inp/g2.inp contains the green's function (same format than delta1.inp but without sig1.inp/sig2.inp contains the self energy (same format than delta1.inp but without

if real freq is use :

sigr contains the sigma on the real axis

green_output_real contains the diagonal part of the gf

0.5 Susceptibility

The difficulty of susceptibility calculation in ED is that transition between two nearby high energy states will contribute. That is why Lanczos is not a good option and only Full_ed works.

So the susceptibility calculation is limited to number site+bath <= 8.

To compute the suscpetibility :

1. modify ED/ED.im
   - FLAG_DUMP_INFO_FOR_GAMMA_VERTEX=.true.
   - Neigen=1000 !(this is for $n_{bath} + n_{imp} = 8$, it could be decrease if this number is smaller.
   - dEmax0=100000 ! This number controles the energy of higher energy state included. for a normal calculation, it has to be small in order to speed up the calculation, but in our case, we want as much as possible states.
   - Nitergreenmax=1 ! we do not want to compute the green fct, it would be too expensive with dEmax0 large
2. Run dmft_solver as usual
3. use omega_path N where N is the number of matsubara fermionic frequencies. It creates the file omega_list_path.

4. create the file cutoff with contains 4 numbers :

   ```
   1d-5
   1d-9
   ```

   The last two numbers are cutoffs used in the code. In practice, dmft_chiloc will compute :

   $$\chi = \sum_{ijkl} e^{-\beta E_i} \Phi(E_i, E_j, E_k, E_l, \nu_1, \nu_2,) c_{ij} c_{jk} c_{kl} c_{li} \tag{5}$$

   where E_i is the energy level of state i, c_{ij} a matrix transition between state ij, and $\Phi(E_i, E_j, E_k, E_l, \nu_1, \nu_2,)$ a function which depends of all the fermionic and bosonic frequencies.

   The number of element of this sum increases as N_{eigen}^3 ! It is crucial to reduce it as much as we can and that is the role of the cutoffs

   The element is computed if $e^{-\beta E_i} >$ cutoff$_1$ and $c_{ij} c_{jk} c_{kl} c_{li} >$ cutoff$_2$.

   dmft_chiloc works with mpi and openmp. Use as much threads as cpu available and mpi to share between nodes.

   The output of the dmft_chiloc is a series of files which contains the susceptibility. To convert them in a nice hdf5 file format, run the program readvertex

   readvertex takes 2 arguments : nomb beta

   The first one is the number of bosonic frequencies and beta is the inverse temperature.

   It creates the file chiS.h5 which contains 'chis' and 'chic' (charge and spin susceptibilities).

   there are both arrays $A[iv1, iv2, i1, i2, i3, i4, iom]$,

   iv1 and iv2 fermionic frequency index

   i1,i2,i3,i4 orbital index

   iom bosonic freqnency index

A simple python program to plot the susceptibility could be :

```
import h5py
import matplotlib.pyplot as plt
import numpy as np

f = h5py.File('chiS.h5','r')
cs = np.array(f['chis'])
cc = np.array(f['chic'])
i1,i2,i3,i4 = 0,0,0,0
iom = 0
plt.plot(np.diag(cs[:,:,i1,i2,i3,i4,iom].real))
plt.show()
```