2. CMS exercise: Impurity Green's function of the Anderson impurity model with Exact Diagonalization

Introduction

Solving the Anderson Impurity Model (AIM) represents a very important first step in several different contexts of computational material science, such as the study of nanoscopic systems, impurity scattering, and metal-insulator transitions in bulk systems, e.g., via the Dynamical Mean Field Theory (more about the latter in the last CMS exercise). With the exception of some limiting cases, such as $U = 0$ (non-interacting case) or all $V = 0$ ("atomic" limit), where even an analytical calculation is possible, the solution of an AIM can be obtained numerically with high precision through several techninques (Exact Diagonalization and/or Lanczos, Numerical Renormalization Group, Density Matrix Renormalization Group). Here, we will consider the most "transparent" solver, e.g., Exact Diagonalization (at finite temperature). More details about exact diagonalization (ED) codes for the numerical solution of the AIM can be also found in the Dynamical Mean Field Theory review article by A. Georges, *et al.*, *Dynamical mean-field theory of strongly correlated fermion systems and the limit of infinite dimensions* in Review of Modern Physics **68**, 13 (1996), and in the references therein.

Files for the exercise

In the folder `UE_AIM_2013` on your computer you find a specific version of an ED impurity solver with some comments related to this exercise. In particular the folder `UE_AIM_2013` contains the following files:

1. `ed_aim_CMS.f`: Fortran code
2. `run.scr`: shell script for compiling and running the code
3. `init.h`: supplementary file with the definition of the important parameter `nmaxx`, which sets the size of your arrays depending how many levels (N_s) are you including for the (discretized) parametrization of your ED problem. $N_s = N_{host} + 1$, since one has to include, beyond the the impurity site, a finite number (N_{host}) of host electronic levels.
4. `hubb.dat`: input file for values of inverse temperature $\beta = 1/T$, Hubbard interaction strength U, the extremes of your real frequency range (variable *wup*, *wlow*) and the lorzentian broadening δ (variable "*deltino*"). The other parameters contained in `hubb.dat` need not be changed.
5. `hubb.andpar`: input values of on-site energies of the host levels (array $Epsk(2, \cdots, N_s)$) and hybridizations V (array $tpar(2, \cdots, N_s)$). Note, instead, that the energy value of the impurity site $\epsilon_d = Epsk(1)$ is read as chemical potential, i.e., as a last entry in `hubb.andpar` (but, watch out! $\mu = -\epsilon_d$). The "half-filling" condition ($\langle n_{imp} \rangle = 1$) is obtained for a "symmetric" choice of the host lattice parameters (ϵ and V) and $\mu = U/2$.

To run the program, simply execute the script `run.scr` from command line. If the program runs properly it will produce the following files

1. `Gw.dat_XXX` where `XXX` is a text string defined when running `run.scr` which will contain the spectrum (goal of the present exercise)

2. `out_XXX` containing the standard output

Changes to the code

In the present form, the code produces only a Green's array, filled with zeros. The lines where the **retarded** Green's function is computed on the real-axis (more precisely on $\omega + i\delta$, with $\delta \to 0^+$) have to be added. This should done directly in the subroutine `Gw_CMS`, which is called right after the subroutine `diag`. Specifically the array *Gwreal* (already defined) will be passed as an argument of subroutine `Gw_CMS` and printed out on file `Gw.dat_XXX` in the main program.

All missing lines are marked by the following comment:

```
CMSUebung************
```

find all of them and insert the missing pieces or remove the commented lines if these are necessary, up to line

```
**********************
```

The definition ("Lehmann representation") of the retarded impurity Green's function to be used has been given in the lecture:

$$G_d^{ret}(\omega) = \frac{1}{Z} \sum_{n,m} \frac{\left(\langle n|d_\uparrow^\dagger|m\rangle\right)^2}{\omega + i\delta - (E_n - E_m)} \left(\mathrm{e}^{-\beta E_n} + \mathrm{e}^{-\beta E_m}\right) \tag{1}$$

- complex array *Gwreal* is declared at the beginning of the program (and in the subroutine `Gw_CMS`)

- β is written in variable *beta*, otherwises all other ingredients for your Lehmann representation of $G_d^{ret}(\omega)$ will be calculated (and provided) by the diagonalization subroutine `diag`

- the partition function Z, is calculated in `diag`, and written in the variable *zpart*.

- The ED obviuosly exploits the block-structure of the AIM Hamiltonian, which is defined by the couple of quantum numbers ($N_\uparrow$,$N_\downarrow$). Hence, the indexes of the eigenvectors/eigenenergies arrays are (or have to be arranged) in such a block structure. More details about the block-structure are given right at the end of this exercise sheet. Note that all index variables mentioned in that subsection are already defined in the program, and you can use them for your purposes.

- all Eigenenergies E_m, E_n are written in the two-index array $eigCMS(m,i)$, whose last index i defines the block to which the eigenvalue E_m belongs, and m identifies the specific eigenvalue in this block.

- the values of the (squared) matrix elements $\left(\langle m|d^\dagger_\uparrow|n\rangle\right)^2$ are written in the array $rlehm(m,n,i)$, which is defined by three indexes (as before, the last one i defines to which block of the Hamiltonian the eigenvector labelled by the first index m belongs).

- to define the frequency *omr* it is useful to use the parameter $Iwmaxreal$, Xi (complex i), *deltino* (lorentzian broadening), and $wup, wlow$ (extremes of the frequency range), which are already defined.

- for testing, do not increase too much the size of the system (impurity + host bath) (set by the variable `Ns`, declared inside file `init.h`). This can be made larger, later, to produce a nicer spectrum after the code works.

- test of your program: compare your Green's function to file `Gw_check`, which refers to the following set of parameters: $N_s = 4$, $\beta = 50$, $U = 40$, $\mu = 20$, bath energies $\epsilon = (-1.9, -0.01, 1.9)$, hybridizations $V = (0.4, 0.17, 0.4)$, lorentzian broadening $\delta = 0.1$.

Additional info: Block structure of the AIM Hamiltonian

The AIM has a block diagonal form. For a given system size (N_s) each block sector can be classified by the couple of quantum numbers ($N_\uparrow = 0, 1, ..., N_s$, $N_\downarrow = 0, 1, .., N_s$). The different blocks are labelled by an index *indblock* running from 1 to *nblock*. The block order is related to the quantum numbers ($N_\uparrow = 0, 1, ..., N_s$, $N_\downarrow = 0, 1, .., N_s$), according to the following labelling prescription: $indblock = N_\uparrow + (N_s + 1) \times N_\downarrow + 1$. With this choice the first block ($indblock = 1$) is the empty one (i.e., $N_\uparrow = 0, N_\downarrow = 0$), the second has a single spin $\uparrow$, etc..., and the last one ($indblock = nblock$) is the fully occupied one ($N_\uparrow = N_s, N_\downarrow = N_s$). Each block has a size of $nleng \times nleng$, and the value of $nleng$ is written in the variable $nleng(indblock)$ with $indblock = 1, ..., nblock$.

Purpose of the exercise

First of all try to understand, **(i)** why it is convenient to work in the given $(N_\uparrow, N_\downarrow)$ block structure, and **(ii)** which blocks are actually connected by the matrix elements of the Lehmann representation. Then, after having modified and tested the code with the benchmark data (**iii)**, use it to calculate **(iv)** the non interacting case and **(v)** the atomic limit. Finally **(vi)**, considering the particle-hole symmetric case $\epsilon_d = -\mu = -\frac{U}{2}$ produce some spectra for the generic case of intermediate-small values of U and follow the central (Kondo-like) peak as a function of temperature and of hybridization strength. Please report per e-mail your considerations, comments and numerical results for all points (i), ..., (vi), in a **unique** .pdf file, and attach additionally your source file (*.f).

Viel Spaß ... und Erfolg!