ALF Tutorial 2.0

Introductory examples and exercises

This notebook is part of the Tutorial 2.0 (https://git.physik.uni-wuerzburg.de/ALF/ALF_Tutorial) for the quantum Monte Carlo simulation package *Algorithms for Lattice Fermions* - ALF (https://git.physik.uni-wuerzburg.de/ALF/ALF_code), and can be found, together with its required files, in the pyALF repository (https://git.physik.uni-wuerzburg.de/ALF/pyALF).

ALF is compiled from source, which is downloaded from the ALF repository (https://git.physik.uni-wuerzburg.de:ALF) when not found locally.

[REMEMBER TO UPDATE GIT ADDRESSES]

A minimal ALF run

In this bare-bones example we use the pyALF interface to run the canonical Hubbard model on a default configuration: a $6\times6$ square grid, with interaction strength $U=4$ and inverse temperature $\beta = 5 $.

Bellow we go through the steps for performing the simulation and outputting observables.

---

1. Import `Simulation` class from the `py_alf` python module, which provides the interface with ALF:

In [1]:

```
from py_alf import Simulation            # Interface with ALF
```

2. Create an instance of `Simulation`, setting parameters as desired:

In [4]:

```
sim = Simulation(
    "Hubbard",                           # Hamiltonian
    {                                    # Model and simulation parameters for each Simulation instance
    "Model": "Hubbard",                  # Base model
    "Lattice_type": "Square"},           # Lattice type
)
```

3. Compile ALF, downloading it first if not found locally. This may take a few minutes:

In [5]:

```
sim.compile()                            # Compilation needs to be performed only once
```

```
Compiling ALF... Done.
```

4. Perform the simulation as specified in `sim`:

In [6]:

```
sim.run()                                # Perform the actual simulation in ALF
```

```
Prepare directory "/home/jonas/Programs/pyALF/Hubbard_Hubbard_Square" for Monte Carlo run.
Create new directory.
Run /home/jonas/Programs/pyALF/ALF/Prog/Hubbard.out
```

5. Perform some simple analyses:

In [7]:

```
sim.analysis()                                 # Perform default analysis; list observables
```

```
Analysing Part_scal
Analysing Kin_scal
Analysing Ener_scal
Analysing Pot_scal
Analysing SpinT_eq
Analysing SpinZ_eq
Analysing SpinXY_eq
Analysing Green_eq
Analysing Den_eq
Analysing SpinZ_tau
Analysing Den_tau
Analysing SpinXY_tau
Analysing SpinT_tau
Analysing Green_tau
```

6. Store computed observables list:

In [8]:

```
obs = sim.get_obs()                            # Dictionary for the observables
```

which are available for further analyses. For instance, the internal energy of the system (and its error) is accessed by:

In [9]:

```
obs['Ener_scalJ']['obs']
```

Out[9]:

```
array([[-29.983503,   0.232685]])
```

---

7. Running again: The simulation can be resumed to increase the precision of the results.

In [10]:

```
sim.run()
sim.analysis()
obs2 = sim.get_obs()
print(obs2['Ener_scalJ']['obs'])
print("\nRunning again reduced the error from ", obs['Ener_scalJ']['obs'][0][1]," to ", obs2['Ener_scalJ']['obs'][0
][1], ".")
```

```
Prepare directory "/home/jonas/Programs/pyALF/Hubbard_Hubbard_Square" for Monte Carlo run.
Resuming previous run.
Run /home/jonas/Programs/pyALF/ALF/Prog/Hubbard.out
Analysing Part_scal
Analysing Kin_scal
Analysing Ener_scal
Analysing Pot_scal
Analysing SpinT_eq
Analysing SpinZ_eq
Analysing SpinXY_eq
Analysing Green_eq
Analysing Den_eq
Analysing SpinZ_tau
Analysing Den_tau
Analysing SpinXY_tau
Analysing SpinT_tau
Analysing Green_tau
[[-29.819654   0.135667]]

Running again reduced the error from  0.232685  to  0.135667 .
```

Note: To run a fresh simulation - instead of performing a refinement over previous run(s) - the Monte Carlo run directory should deleted before rerunning.

Exercises

1. Rerun once again and check the new improvement in precision.
2. Look at a few other observables (`sim.analysis()` outputs the names of those available).
3. Change the lattice size by adding, e.g., `"L1": 4,` and `"L2": 1,` to the simulation parameters definitions of `sim` (step 2).