minimal_ALF_run

August 21, 2020

1 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 6×6 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:

[1]:
```
from py_alf import Simulation                    # Interface with ALF
```

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

[2]:
```
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 from the ALF repository if not found locally. This may take a few minutes:

[3]:
```
sim.compile()                                    # Compilation needs to be performed␣
 ↪only once
```

```
Repository /home/stafusa/ALF/pyALF/Notebooks/ALF does not exist, cloning from
git@git.physik.uni-wuerzburg.de:ALF/ALF.git
Compiling ALF... Done.
```

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

[4]:
```
sim.run()                                        # Perform the actual simulation in ALF
```

```
Prepare directory "/home/stafusa/ALF/pyALF/Notebooks/Hubbard_Square" for Monte
Carlo run.
```

```
Create new directory.
Run /home/stafusa/ALF/pyALF/Notebooks/ALF/Prog/Hubbard.out
```

5. Perform some simple analyses:

[5]:
```
sim.analysis()                          # Perform default analysis; list␣
 ↪observables
```

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

6. Store computed observables list:

[6]:
```
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:

[7]:
```
obs['Ener_scalJ']['obs']
```

[7]:
```
array([[-29.983503,   0.232685]])
```

---

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

[8]:
```
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/stafusa/ALF/pyALF/Notebooks/Hubbard_Square" for Monte
Carlo run.
Resuming previous run.
Run /home/stafusa/ALF/pyALF/Notebooks/ALF/Prog/Hubbard.out
Analysing Ener_scal
```

```
Analysing Part_scal
Analysing Pot_scal
Analysing Kin_scal
Analysing Den_eq
Analysing SpinZ_eq
Analysing Green_eq
Analysing SpinXY_eq
Analysing SpinT_eq
Analysing SpinXY_tau
Analysing SpinZ_tau
Analysing Den_tau
Analysing Green_tau
Analysing SpinT_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.

1.1 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).