SEET Tutorial

Sergei Iskakov

March 30, 2022

Prerequisite

- Python 3.x (pyscf, irbasis)
- ALPSCore (https://github.com/ALPSCore/ALPSCore)
  - UGF2 (https://github.com/CQMP/UGF2)
  - gfmol (https://github.com/CQMP/gfmol)
  - EDLib (https://github.com/Q-Solvers/EDLib)
- ARPACK-ng (https://github.com/opencollab/arpack-ng)
- IR-grid files

Preparation

1. Create directories

```
> mkdir /data/distr
```

2. Load required modules

```
> module load BuildEnv/gcc-10.3.0
```

Building ALPSCore

```
> git clone https://github.com/ALPSCore/ALPSCore
> mkdir build && cd build
> cmake -DCMAKE_INSTALL_PREFIX=`pwd`/../install ../ALPSCore
> make -j 8 && make test && make install
> rm -rf *
> cd ..
```

Building ARPACK

```
> git clone https://github.com/opencollab/arpack-ng
> cd build
> cmake -DCMAKE_INSTALL_PREFIX=`pwd`/../install
-DMPI=ON ../arpack-ng
> make -j 8 && make test && make install
> rm -rf *
> cd ..
```

Building EDLib

```
> git clone https://github.com/Q-Solvers/EDLib
> cd build
> cmake -DCMAKE_INSTALL_PREFIX=`pwd`/../install
-DUSE_MPI=ON -DARPACK_DIR=../install -DALPSCore_DIR=../
install/share/ALPSCore ../EDLib
> make -j 8 && make install
> rm -rf *
> cd ..
```

Building gfmol

```
> git clone https://github.com/CQMP/gfmol
> cd build
> cmake -DCMAKE_INSTALL_PREFIX=`pwd`/../install
-DALPSCore_DIR=../install/share/ALPSCore ../
> make -j 8 && make test && make install
> rm -rf *
> cd ..
```

Building UGF2

```
> git clone https://github.com/CQMP/UGF2
> cd build
> cmake -DCMAKE_INSTALL_PREFIX=`pwd`/../install
-DALPSCore_DIR=../install/share/ALPSCore ../UGF2
> make -j 8 && make install
> rm -rf *
> cd ..
```

Building SEET

```
> git clone https://github.com/CQMP/PBC_SEET
> cd PBC_SEET && mkdir build && cd build
> cmake -DALPSCore_DIR=`pwd`/../../install/share/ALPSCore
-DEDLib_DIR=`pwd`/../../install/share/EDLib/cmake
-Dgfmol_DIR=`pwd`/../../install/share/gfmol/cmake
-DUSE_MPI=ON ..
> make -j 8 && make test
> cd ..
```

SEET Workflow

- Obtain integrals and initial solution (pyscf)
- Obtain weak coupling solution (UGF2)
- Setup transformation matrices (seet_transform.py)
- Transform Coulomb integrals (int-transform)
- Run SEET (seet_main.py)

Setup system

System geometry

1. Lattice vectors

2. Atoms position in the unit cell

e.g. LiH:

a.dat:
```
2.03275,    2.03275,    0.0
0.0,      2.03275,  2.03275
2.03275,    0.0,    2.03275
```

atoms.dat:
```
H   0.0   0.0    0.0
Li 2.03275 2.03275 2.03275
```

Setup system

All electron setup:

```
> python ../../../UGF2/scripts/init_data_df.py
--a a.dat --atom atoms.dat --basis Li Li.dat H
H.dat
```

Pseudo potentials

```
> python ../../../UGF2/scripts/init_data_df.py
--a a.dat --atom atoms.dat --basis gth-dzvp-
molopt-sr --pseudo gth-pbe
```

Output:

- Hartree-Fock solution:
  - input.h5
- Coulomb integrals:
  - df_int
  - df_hf_int

Weak-coupling solution

GF2/GW solver (UGF2) main parameters:

- nel_cell - number of electrons in unit cell
- nao - number of orbitals in unit cell
- nk - total number of k-points
- ink - number of irreducible k-points
- ns - number of spins
- NQ - number of aux basis functions in Vq
- ni - dimension of sparse grid basis
- beta - inverse temperature
- itermax - number of iterations
- scf_type - weak-coupling solver (GW, GF2, cuGW, …)
- IR - use IR for sparse grid basis
- TNL - path to the sparse grid file
- TNL_B - same as TNL

Default output: sim.h5

Choose SEET problem (seet_transform.py)

- Choose orbitals for active space
- Orthogonalize integrals or not
- Check if your input is obtained on irreducible k-grid

- python seet_transform.py
  --orth true
  --active_space 0 1
  --active_space 2 3
  --orth_method symmetrical_orbitals
  --from_ibz true
  --transform_file transform.h5

Transform integrals (int-transform)

Using transform.h5 from previous step we map the Coulomb integrals for the whole system onto local projected subspace:

```
• int-transform
    --input_file transform.h5
    --in_int_file df_int
    --transform=1
    --out_int_file Uijkl.h5
```

SEET

Input files for SEET:

- input.h5
- df_int
- df_hf_int
- sim.h5
- transform.h5
- Uijkl.h5
- type of impurity solver
  - ED parameters:
    - anderson.param
    - bath.txt - initial guess for impurity bath
    - min_types.txt - type of minimization for bath
-

SEET with ED solver

- ED parameters:
  - anderson.param
  - bath.txt - initial guess for impurity bath
  - min_types.txt - type of minimization for bath

```
python seet_main.py
    --orth true
    --from_ibz 1
    --input_file input.h5
    --integral_file Uijkl.h5
    --transform_file transform.h5
    --grid_transforms_path /data/common/
    --grid_transforms_file /data/common/ir/1e7_202.h5
    --beta 300
--impurity_solver impuritysolver.edsolver.solver
--number_of_impurities 2
--dc_command gfmol_seet
--ed_command PBC_SEET/build/ed_solver/anderson-
example
--ed_input_file ed.input.h5
--min_type_file min_types.txt
--bath_file bath.txt
```