Running self-energy embedding theory (SEET)

Chia-Nan Yeh

Department of Physics
University of Michigan, Ann Arbor
Michigan

June 8 2022, Gull's Group meeting
University of Michigan, Ann Arbor

Outline

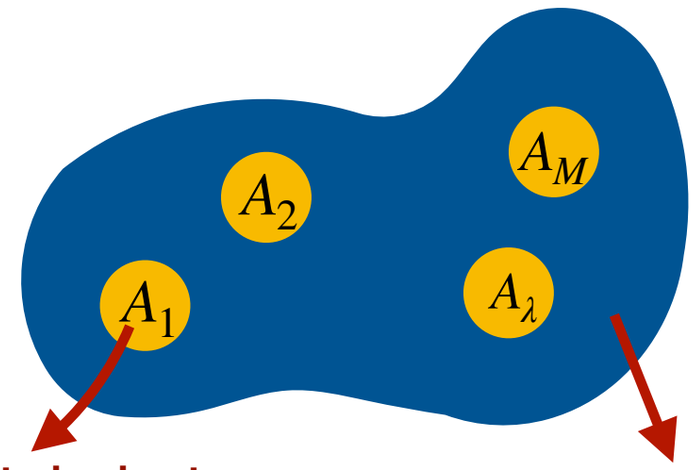
- **Quantum embedding method for Electronic structure problem**
- **Overview of SEET in (UGF2 + PBC_SEET)**
- **Example: AFM MnO**

Quantum Embedding method for electronic structure method

- **SEET:**

Zgid and Gull, New J. Phys. 19 023047 (2017)

For M disjoint strongly correlated subsets of local orbitals $A_1, \ldots, A_M$.
(The only parameter)



Strongly correlated subsets:
small; solved exactly;
ED in the present work

Weakly correlated environment:
Large system size; solved approximately;
sc*GW* in the present work

In k space:

$$(\Sigma^{\mathrm{SEET}})^{\mathbf{k}}_{ij} = (\Sigma^{GW})^{\mathbf{k}}_{ij} + \sum_{\lambda=1}^{M} (\Sigma^{\mathrm{imp}}_{A_\lambda,ij} - \Sigma^{\mathrm{DC},GW}_{A_\lambda,ij})\delta_{(ij)\in A_\lambda}$$

$$(G^{\mathrm{SEET}})^{\mathbf{k}} = [(i\omega_n + \mu) - H_0^{\mathbf{k}} - (\Sigma^{\mathrm{SEET}})^{\mathbf{k}}]^{-1}$$

The embedding self-consistent condition in real space:

$$(G^{\mathrm{SEET}})^{\mathrm{loc}} = [(i\omega_n + \mu) - H_0^{\mathrm{loc}} - (\Sigma^{\mathrm{SEET}})^{\mathrm{loc}} - \Delta^{\mathrm{SEET}}]^{-1}$$

$$= [(i\omega_n + \mu) - \tilde{F}^{\mathrm{loc}} - \Sigma^{\mathrm{imp}} - \underline{\tilde{\Sigma}^{\mathrm{non-loc}}_{GW} - \Delta^{\mathrm{SEET}}}]^{-1} = \left[\underline{[G^{\mathrm{imp}}]^{-1} - \tilde{\Sigma}^{\mathrm{non-loc}}_{GW}}\right]^{-1}$$

$\tilde{\Sigma}^{\mathrm{non-loc}}_{GW} = (\Sigma^{GW})^{\mathrm{loc}} - \Sigma^{\mathrm{DC},GW}$ contains the non-local contributions at the GW level

Quantum Embedding method for electronic structure method

Electronic Hamiltonian in Bloch basis: $\chi_i^{\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{R}} \chi_i^{\mathbf{R}}(\mathbf{r}) e^{i\mathbf{k}\mathbf{R}},\ \chi_i^{\mathbf{k}}(\mathbf{r}+\mathbf{R}) = \chi_i^{\mathbf{k}}(\mathbf{r})$

$$H = \sum_{\mathbf{k}} \sum_{ij} (H_0)_{ij}^{\mathbf{k}} c_i^{\mathbf{k}\dagger} c_j^{\mathbf{k}} + \frac{1}{2N_k} \sum_{\mathbf{k}\mathbf{k}'\mathbf{q}} \sum_{ijkl} U_{\substack{i\ j\ k\ l}}^{\mathbf{k}\ \mathbf{k}-\mathbf{q}\ \mathbf{k}'\ \mathbf{k}'+\mathbf{q}} c_i^{\mathbf{k}\dagger} c_k^{\mathbf{k}'\dagger} c_l^{\mathbf{k}'+\mathbf{q}} c_j^{\mathbf{k}-\mathbf{q}}$$

$$(H_0)_{ij}^{\mathbf{k}} = \int_{\Omega} d\mathbf{r} \chi_i^{\mathbf{k}*}(\mathbf{r}) [-\frac{1}{2}\nabla^2 + V(\mathbf{r})] \chi_j^{\mathbf{R}'}(\mathbf{r})$$

$$U_{\substack{i\ j\ k\ l}}^{\mathbf{k}\ \mathbf{k}-\mathbf{q}\ \mathbf{k}'\ \mathbf{k}'+\mathbf{q}} = \int_{\Omega} d\mathbf{r} \int d\mathbf{r}' \chi_i^{\mathbf{k}*}(\mathbf{r}) \chi_j^{\mathbf{k}-\mathbf{q}}(\mathbf{r}) U(\mathbf{r}-\mathbf{r}') \chi_k^{\mathbf{k}'*}(\mathbf{r}') \chi_l^{\mathbf{k}'+\mathbf{q}}(\mathbf{r}')$$

Impurity Hamiltonian in localized atomic basis: $\chi_i^{\mathbf{R}=\mathbf{0}}(\mathbf{r})$

$$H_{imp} = \sum_{ij} (\tilde{H}_0)_{ij} c_i^\dagger c_j + \frac{1}{2} \sum_{ijkl} U_{ijkl} c_i^\dagger c_k^\dagger c_l c_j + \sum_b \epsilon_b a_b^\dagger a_b + \sum_{ib} V_{ib} c_i^\dagger a_b + h.c.$$

$\tilde{H}_0$, ϵ_b, and V_{ib} depend on the scGW solution

$$U_{ijkl} = \int d\mathbf{r} \int d\mathbf{r}'\ \chi_i^{\mathbf{0}*}(\mathbf{r}) \chi_j^{\mathbf{0}}(\mathbf{r}) U(\mathbf{r}-\mathbf{r}') \chi_k^{\mathbf{0}*}(\mathbf{r}') \chi_l^{\mathbf{0}}(\mathbf{r}')$$

Overview of material simulations in UGF2

- **Define the problem**
  - Atoms in the primitive unit cell
  - Translational vectors
  - Gaussian basis set
  - ***k***-mesh

- **Compute matrix elements of the Hamiltonian**
  - Python scripts using PySCF
  - init_data_df.py script in UGF2/scripts/
  - Density fitting for the two-electron Coulomb interaction
  - Store the Hamiltonian matrix elements in HDF5 files

- **Run many-body perturbation theory (MBPT) using UGF2**
  - C++ and CUDA
  - Input: Matrix elements stored in the HDF5 output files from UGF2/script/init_data_df.py
  - Output: Green's function and Self-energy

- **Post-processing:** Band structure, Fermi surface etc

Overview of material simulations in UGF2 + PBC_SEET

- **Define the problem**
  - Atoms in the primitive unit cell
  - Translational vectors
  - Gaussian basis set
  - ***k***-mesh
- **Compute matrix elements of the Hamiltonian**
  - Python scripts using PySCF
  - init_data_df.py script in UGF2/scripts/
  - Density fitting for the two-electron Coulomb interaction
  - Store the Hamiltonian matrix elements in HDF5 files

- **Run many-body perturbation theory (MBPT) using UGF2**
  - C++ and CUDA
  - Input: Matrix elements stored in the HDF5 output files from UGF2/script/init_data_df.py
  - Output: Green's function and Self-energy

- **Run SEET using PBC_SEET**
  - Python and C++ (for impurity solvers)
  - Input: Impurity matrix elements, Green's function and Self-energy from UGF2

- **Post-processing:** Band structure, Fermi surface etc

Building PBC_SEET

- **Github**: https://github.com/CQMP/PBC_SEET
- **Dependencies**
  - CMake
  - HDF5
  - EDLib (for ED impurity solver)
  - gfmol (for double counting self-energy)

- **Building PBC_SEET**

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

Running PBC_SEET

- **Run SEET using PBC_SEET**
  - Python and C++ (for impurity solvers)
  - Input: Impurity matrix elements, Green's function and Self-energy from UGF2
  - **Workflow:**
    (1) **Setup SEET problem (seet_transform.py)**
    (2) **Transform Coulomb integrals (int-transform)**
    (3) **Run SEET (seet_main.py)**

Impurity Hamiltonian in localized atomic basis: $\chi_i^{\mathbf{R=0}}(\mathbf{r}), i \in A$

$$H_{imp} = \sum_{ij} (\tilde{H}_0)_{ij} c_i^\dagger c_j + \frac{1}{2} \sum_{ijkl} U_{ijkl} c_i^\dagger c_k^\dagger c_l c_j + \sum_b \epsilon_b a_b^\dagger a_b + \sum_{ib} V_{ib} c_i^\dagger a_b + h.c.$$

$\tilde{H}_0$, ϵ_b, and V_{ib} depend on the scGW solution

$$U_{ijkl} = \int d\mathbf{r} \int d\mathbf{r}' \, \chi_i^{\mathbf{0}*}(\mathbf{r}) \chi_j^{\mathbf{0}}(\mathbf{r}) U(\mathbf{r} - \mathbf{r}') \chi_k^{\mathbf{0}*}(\mathbf{r}') \chi_l^{\mathbf{0}}(\mathbf{r}')$$

Running PBC_SEET

Impurity Hamiltonian in localized atomic basis: $\chi_i^{\mathbf{R=0}}(\mathbf{r}), i \in A$

$$H_{imp} = \sum_{ij} (\tilde{H}_0)_{ij} c_i^\dagger c_j + \frac{1}{2} \sum_{ijkl} U_{ijkl} c_i^\dagger c_k^\dagger c_l c_j + \sum_b \epsilon_b a_b^\dagger a_b + \sum_{ib} V_{ib} c_i^\dagger a_b + h.c.$$

$\tilde{H}_0$, ϵ_b, and V_{ib} depend on the scGW solution

$$U_{ijkl} = \int d\mathbf{r} \int d\mathbf{r}' \, \chi_i^{\mathbf{0}*}(\mathbf{r}) \chi_j^{\mathbf{0}}(\mathbf{r}) U(\mathbf{r} - \mathbf{r}') \chi_k^{\mathbf{0}*}(\mathbf{r}') \chi_l^{\mathbf{0}}(\mathbf{r}')$$

(1) **Setup SEET problem (seet_transform.py)**

- Choose impurity subspaces A
- compute transformation matrices

```
• 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
```

Running PBC_SEET

Impurity Hamiltonian in localized atomic basis: $\chi_i^{\mathbf{R=0}}(\mathbf{r}), i \in A$

$$H_{imp} = \sum_{ij} (\tilde{H}_0)_{ij} c_i^\dagger c_j + \frac{1}{2} \sum_{ijkl} U_{ijkl} c_i^\dagger c_k^\dagger c_l c_j + \sum_b \epsilon_b a_b^\dagger a_b + \sum_{ib} V_{ib} c_i^\dagger a_b + h.c.$$

$\tilde{H}_0$, ϵ_b, and V_{ib} depend on the scGW solution

$$U_{ijkl} = \int d\mathbf{r} \int d\mathbf{r}' \, \chi_i^{\mathbf{0}*}(\mathbf{r}) \chi_j^{\mathbf{0}}(\mathbf{r}) U(\mathbf{r} - \mathbf{r}') \chi_k^{\mathbf{0}*}(\mathbf{r}') \chi_l^{\mathbf{0}}(\mathbf{r}')$$

(2) **Transform Coulomb integrals (int-transform)**

- Project the Coulomb integrals for the whole system onto local impurity subspaces

$$U_{i\ j\ k\ l}^{\mathbf{k}\ \mathbf{k-q}\ \mathbf{k}'\ \mathbf{k}'+\mathbf{q}} \rightarrow U_{ijkl}$$

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

Running PBC_SEET

(3) **Run SEET (seet_main.py)**

Basic parameters

- orth - Apply orthogonalization to input data
- compute_energy - Compute and print energy
- from_ibz - Whether input data is in the reduced Brillouin zone
- input_file - Input of one-body matrix elements (input.h5 from init_data_df.py)
- gf2_input_file - Input of weak-coupling solution from UGF2 (sim.h5)
- output_file - SEET output file name
- integral_file - Coulomb integral transformed into active space
- transform_file - Transformation matrices
- grid_transforms_file - Freq/time transform file name
- grid_transforms_path - Freq/time transform base path
- lmbda - IR Lambda
- beta - Inverse temperatrue
- nel - Number of electrons per unit cell
- const_mu - Fix chemical potential
- max_iter - Number of iterations
- impurity_solver - Name of the impurity solver class
- dc_command - Command to run weakly correlated solver to compute double counting
- number_of_impurities - Number of impurities
- damp - Damping for SEET self-energy
- fixed_DC - Fix double counting self-energy from the outer-loop

Try **python seet_main.py --help** to check the complete parameter list!

Running PBC_SEET

(3) **Run SEET (seet_main.py)**

Basic parameters

- orth - Apply orthogonalization to input data
- compute_energy - Compute and print energy
- from_ibz - Whether input data is in the reduced Brillouin zone
- input_file - Input of one-body matrix elements (input.h5 from init_data_df.py)
- gf2_input_file - Input of weak-coupling solution from UGF2 (sim.h5)
- output_file - SEET output file name
- integral_file - Coulomb integral transformed into active space
- transform_file - Transformation matrices
- grid_transforms_file - Freq/time transform file name
- grid_transforms_path - Freq/time transform base path
- lmbda - IR Lambda
- beta - Inverse temperatrue
- nel - Number of electrons per unit cell
- const_mu - Fix chemical potential
- max_iter - Number of iterations
- impurity_solver - Name of the impurity solver class
- dc_command - Command to run weakly correlated solver to compute double counting
- number_of_impurities - Number of impurities
- damp - Damping for SEET self-energy
- fixed_DC - Fix double counting self-energy from the outer-loop

Outputs of the previous two steps (seet_transform.py and int-transfrom)

Try **python seet_main.py --help** to check the complete parameter list!

Running PBC_SEET

(3) **Run SEET (seet_main.py)**

Useful parameters when bath fitting is required

- spin_symmetrization - Apply spin symmetrization for impurity Green's function
- orb_sym_block - Copy the SEET self-energies to orbitals that have the same symmetry of impurities
- orb_sym_groups - Group orbitals with the same symmetry within an impurity. (Only works when quantities are diagonal in the active space)

Important to reduce accumulated error due to the bath fitting!

ED parameters

- ed_input_file - Name of the input file for ED solver to be generated
- ed_params - ED parameter file (Check https://github.com/Q-solvers/EDLib)
- ed_command - Command to run ED solver along with necessary parameters (e.g. PBC_SEET/build/ed_solver/anderson-example —FREQ_FILE=/data/common/ir/1e7_202.hdf5)
- imp_size - Size of each impurity problem
- min_type_file - File with minimization types for different impurities
- bath_file - File with initial bath parameters

Running PBC_SEET

(3) **Run SEET (seet_main.py)**

Useful parameters when bath fitting is required

- spin_symmetrization - Apply spin symmetrization for impurity Green's function
- orb_sym_block - Copy the SEET self-energies to orbitals that have the same symmetry of impurities
- orb_sym_groups - Group orbitals with the same symmetry within an impurity. (Only works when quantities are diagonal in the active space)

Important to reduce accumulated error due to the bath fitting!

ED parameters

- ed_input_file - Name of the input file for ED solver to be generated
- ed_params - ED parameter file (Check https://github.com/Q-solvers/EDLib)
- ed_command - Command to run ED solver along with necessary parameters (e.g. PBC_SEET/build/ed_solver/anderson-example —FREQ_FILE=/data/common/ir/1e7_202.hdf5)
- imp_size - Size of each impurity problem
- min_type_file - File with minimization types for different impurities
- bath_file - File with initial bath parameters

0: $R(i\omega_n) = [\Delta(i\omega_n) - \Delta^{\mathrm{fit}}(i\omega_n)]\sqrt{\omega_n}$

1: $R(i\omega_n) = \Delta(i\omega_n) - \Delta^{\mathrm{fit}}(i\omega_n)$

2: $R(i\omega_n) = [\Delta(i\omega_n) - \Delta^{\mathrm{fit}}(i\omega_n)]\omega_n$

Example: AFM MnO

- Basis set: *gth-dzvp-molopt-sr,* Pseudopotential: *gth-pbe*
- Impurities: Mn t_{2g}; Mn e_g; O p

min_type.txt:

```
{"0": [1, 1, 1], "1": [1, 1], "2": [1, 1, 1]}
```

orb_sym_block.txt:

```
{"0": [[-48, -49, -51]], "1": [[-50, -52]], "2": [[-67, -68, -69]]}
```

```
python $SEET_dir/scripts/seet_main.py \
     --orth 1 \
     --from_ibz 1 \
     --compute_energy 1 \
     --input_file /home/cnyeh/calc/MnO/nk6/LDA/input.h5 \
     --gf2_input_file /home/cnyeh/calc/MnO/nk6/GW/sim_last.h5 \
     --integral_file /home/cnyeh/calc/MnO/nk6/SEET/t2g_eg_p/integrals/df_int_transform.h5 \
     --transform_file  /home/cnyeh/calc/MnO/nk6/SEET/t2g_eg_p/integrals/transform.h5 \
     --grid_transforms_file /home/cnyeh/Project/gf2plus/data/ir/1e6_168.hdf5 \
     --grid_transforms_path=/home/cnyeh/Project/gf2plus/data/ \
     --lmbda=1e6 \
     --beta 700.0 \
     --nel 42 \
     --const_mu 0 \
     --max_iter 10 \
     --damp 0.5 \
     --fixed_DC 1 \
     --number_of_impurities 3 \
     --dc_command="gfmol_seet" \
     --impurity_solver=impuritysolver.edsolver.solver \
     --ed_input_file ed.input.h5 \
     --ed_params anderson.param \
     --ed_output_file=ed.sim.h5 \
     --ed_command="srun --export=ALL,OMP_NUM_THREADS=1,MKL_NUM_THREADS=1 --cpu-bind=cores -n 64 -u
/home/cnyeh/Project/PBC_SEET/build/ed_solver/anderson-example --FREQ_FILE=/home/cnyeh/Project/gf2pl
us/data/ir/1e6_168.hdf5" \
     --imp_size 12 10 12 \
     --min_type_file min_type.txt \
     --bath_file bath.txt \
     --orb_sym_groups 0 0 0 \
     --orb_sym_groups 0 0 \
     --orb_sym_groups 0 0 0 \
     --orb_sym_block orb_sym_block.txt
```