Self-consistent GW in Practice

Chia-Nan Yeh

Department of Physics
University of Michigan, Ann Arbor
Michigan

May. 4 2022, Gull's Group meeting
University of Michigan, Ann Arbor

Outline

- **Electronic structure problem**
- **Overview of real material simulations in UGF2**
- **sc*GW* in UGF2**
- **Examples: Silicon**

Electronic structure simulation of real materials

- Molecules
  - Finite-size system
- **Solids**
  - Infinite periodic unit cells
  - Atoms in a unit cell + Translational vectors
- **Born-Oppenheimer approximation**
  - Stationary nuclei
  - Neglect the kinetic energy of nuclei
  - Coulomb repulsion energy between nuclei can be view as a constant
- **Electronic Hamiltonian of an N-electron system with M nuclei**

$$H = \sum_{i=1}^{N} -\frac{1}{2}\nabla_i^2 + \sum_{i=1}^{N}\sum_{A=1}^{M} -\frac{Z_A}{r_{iA}} + \sum_{i=1}^{N}\sum_{j>i}^{N}\frac{1}{r_{ij}}$$

One-body part **two-body part**

Electronic Hamiltonian in second quantization

- Certain type of basis could benefit certain electronic structure methods

a) Position basis: $H = \int d\mathbf{r}\psi^\dagger(\mathbf{r})[\underbrace{-\frac{1}{2}\nabla^2 + V(\mathbf{r})}_{H_0(\mathbf{r})}]\psi(\mathbf{r}) + \frac{1}{2}\int d\mathbf{r}\int d\mathbf{r}' U(\mathbf{r}-\mathbf{r}')\psi^\dagger(\mathbf{r})\psi^\dagger(\mathbf{r}')\psi(\mathbf{r}')\psi(\mathbf{r})$

electron-nuclei Coulomb interaction

b) Localized atomic basis $\chi_i^{\mathbf{R}}(\mathbf{r})$: $\mathbf{R}$: unit cell index, i : orbital index

$$H = \sum_{\mathbf{RR}'}\sum_{ij}(H_0)_{ij}^{\mathbf{RR}'}c_i^{\mathbf{R}\dagger}c_j^{\mathbf{R}} + \frac{1}{2}\sum_{\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{R}_4}\sum_{ijkl}U_{ijkl}^{\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{R}_4}c_i^{\mathbf{R}_1\dagger}c_k^{\mathbf{R}_3\dagger}c_l^{\mathbf{R}_4}c_j^{\mathbf{R}_2}$$

E.g. Multi-orbital Hubbard model

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

$$U_{ijkl}^{\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{R}_4} = \int d\mathbf{r}\int d\mathbf{r}'\,\chi_i^{\mathbf{R}_1*}(\mathbf{r})\chi_j^{\mathbf{R}_2}(\mathbf{r})U(\mathbf{r}-\mathbf{r}')\chi_k^{\mathbf{R}_3*}(\mathbf{r}')\chi_l^{\mathbf{R}_4}(\mathbf{r}')$$

c) Bloch localized atomic basis $\chi_i^{\mathbf{k}}(\mathbf{r})$: $\chi_i^{\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{R}}\chi_i^{\mathbf{R}}(\mathbf{r})e^{i\mathbf{kR}},\ \chi_i^{\mathbf{k}}(\mathbf{r}+\mathbf{R}) = \chi_i^{\mathbf{k}}(\mathbf{r})$

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

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

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

UGF2: MBPT for periodic systems

Github: https://github.com/CQMP/UGF2

c) Bloch localized atomic basis $\chi_i^{\mathbf{k}}(\mathbf{r})$: $\chi_i^{\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{R}} \chi_i^{\mathbf{R}}(\mathbf{r}) e^{i\mathbf{k}\mathbf{R}}, \ \chi_i^{\mathbf{k}}(\mathbf{r}+\mathbf{R}) = \chi_i^{\mathbf{k}}(\mathbf{r})$

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

- Once the Hamiltonians have been computed, the UGF2 program
  - Performs many-body perturbation theory (MBPT) using different MBPT solvers $\mathscr{F}^{\mathrm{MBPT}}$

MBPT self-consistent loop:

$$G_{ij}^{\mathbf{k}}(i\omega_n) = [(i\omega_n + \mu)S^{\mathbf{k}} - H_0^{\mathbf{k}} - \Sigma^{\mathbf{k}}(i\omega_n)]_{ij}^{-1}$$

$$\Sigma_{ij}^{\mathbf{k}} = \mathscr{F}^{\mathrm{MBPT}}[G]$$

  - $\mathscr{F}^{\mathrm{MBPT}}$: HF, GF2, *GW* etc
  - Assumes Bloch Gaussian-type orbitals as the one-particle basis
  - Assumes decomposed three-index Coulomb integrals $V_{i\ j}^{\mathbf{k}\mathbf{k}-\mathbf{q}}(Q)$

$$U_{i\ j\ k\ l}^{\mathbf{k}\ \mathbf{k}-\mathbf{q}\ \mathbf{k}'\ \mathbf{k}'+\mathbf{q}} = \sum_{Q} V_{i\ j}^{\mathbf{k}\mathbf{k}-\mathbf{q}}(Q) V_{k\ l}^{\mathbf{k}'\mathbf{k}'+\mathbf{q}}(Q)$$

  - Uses intermediate numerical representations (IR) and sparse-sampling technique to represent dynamic objects on the imaginary axes

Overview of material simulations in UGF2

- **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

Overview of material simulations in UGF2

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

- **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

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

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
  - Python and C++
  - Band interpolation along the high-symmetry *k*-path
  - Analytical continuation

Building UGF2

- **Github**: https://github.com/CQMP/UGF2
- **Dependencies**
  - CMake
  - ALPSCore (https://github.com/ALPSCore/ALPSCore)
  - Eigen3
  - HDF5
  - CUDA (optional)
- **Example of building UGF2 on pauli:**

1. Load required libraries:
   module load BuildEnv/intel-2021.3.0
   module load alpscore/2.3.1a-mpi
2. git clone https://github.com/CQMP/UGF2.git and go to the UGF2 folder
3. mkdir build && cd build
4. CC=mpicc CXX=mpicxx cmake ..
   -DCMAKE_CXX_FLAGS='-DEIGEN_USE_BLAS'
   -DCMAKE_EXE_LINKER_FLAGS='-lopenblas'
   ] Use openblas as backends for some linear algebra functions (optional)
   -DWITH_CUDA=ON] Enable cuda HF/GW solver (optional)
5. make -j4

Running UGF2

Basic parameters

- scf_type - MBPT solver, e.g. HF, GF2, GW etc
- nel_cell - number of electrons per unit cell
- nao - number of atomic orbitals per unit cell
- nk - number of k-points
- ink - number of reduced k-points in the presence of inversion symmetry
- ns - number of spins (1: restricted, 2: unrestricted)
- NQ - number of auxiliary basis per unit cell
- beta - inverse temperature (Ha^{-1})
- ni - number of imaginary-time points in the IR gird
- TNL - IR-grid files for fermonic functions
- TNL_B - IR-grid files for bosonic functions
- intermax - maximum number of MBPT iterations
- E_thr - Energy convergence threshold
- input_file - input files with one-electron Hamiltonians
- dfintegral_file - density-fitted Coulomb integrals for GW, GF2
- HF_dfintegral_file - density-fitted Coulomb integrals for HF
- Results - HDF5 files to store results
- rst - read “Results” and restart the calculation

Outputs of UGF2/script/init_data_df.py

Try **./UGF2 --help** to check the complete parameter list!

Running UGF2

Basic parameters

- scf_type - MBPT solver, e.g. HF, GF2, GW etc
- nel_cell - number of electrons per unit cell
- nao - number of atomic orbitals per unit cell
- nk - number of k-points
- ink - number of reduced k-points in the presence of inversion symmetry
- ns - number of spins (1: restricted, 2: unrestricted)
- NQ - number of auxiliary basis per unit cell
- beta - inverse temperature (Ha^{-1})
- ni - number of imaginary-time points in the IR gird
- TNL - IR-grid files for fermonic functions
- TNL_B - IR-grid files for bosonic functions
- intermax - maximum number of MBPT iterations
- E_thr - Energy convergence threshold
- input_file - input files with one-electron Hamiltonians
- dfintegral_file - density-fitted Coulomb integrals for GW, GF2
- HF_dfintegral_file - density-fitted Coulomb integrals for HF
- Results - HDF5 files to store results
- rst - read “Results” and restart the calculation

Outputs of UGF2/script/init_data_df.py

Try **./UGF2 --help** to check the complete parameter list!

Running UGF2

Basic parameters

- scf_type - MBPT solver, e.g. HF, GF2, GW etc
- nel_cell - number of electrons per unit cell
- nao - number of atomic orbitals per unit cell
- nk - number of k-points
- ink - number of reduced k-points in the presence of inversion symmetry
- ns - number of spins (1: restricted, 2: unrestricted)
- NQ - number of auxiliary basis per unit cell
- beta - inverse temperature (Ha^{-1})
- ni - number of imaginary-time points in the IR gird
- TNL - IR-grid files for fermonic functions
- TNL_B - IR-grid files for bosonic functions
- intermax - maximum number of MBPT iterations
- E_thr - Energy convergence threshold
- input_file - input files with one-electron Hamiltonians
- dfintegral_file - density-fitted Coulomb integrals for GW, GF2
- HF_dfintegral_file - density-fitted Coulomb integrals for HF
- Results - HDF5 files to store results
- rst - read “Results” and restart the calculation

Outputs of UGF2/script/init_data_df.py

Try **./UGF2 --help** to check the complete parameter list!

IR representation for dynamic quantities

- ni - number of imaginary-time points in the IR gird
- TNL - IR-grid files for fermonic functions
- TNL_B - IR-grid files for bosonic functions

For all dynamic quantities, a set of IR non-uniform grids and the corresponding Fourier transformation matrices are pre-computed and stored at

/UGF2/MB_analysis/data/ir_grid/lambda_ni.h5

Ex: 1e3_72.h5, 1e4_104.h5, 1e5_136.h5, 1e6_168.h5

Phys. Rev. B 96, 035147 (2017)
Phys. Rev. B 101, 035144 (2020)

- lambda: controlled parameter in the IR representation.
  In principle, **lambda $\geq$ (beta) * (energy window of the system)**
- ni: number of imaginary-time points
- lambda_ni.h5 stores both the fermonic and bosonic grids.
- Lower temperature or larger energy window -> larger imaginary grids

Theoretical details can be found in
https://green.physics.lsa.umich.edu/mw19/index.php?title=IR_Basis_Set
https://green.physics.lsa.umich.edu/mw19/index.php?title=Sparse_Sampling_in_time_and_frequency

Hartree-Fock (scf_type=HF, HF_X2C1e)

Computational bottleneck:

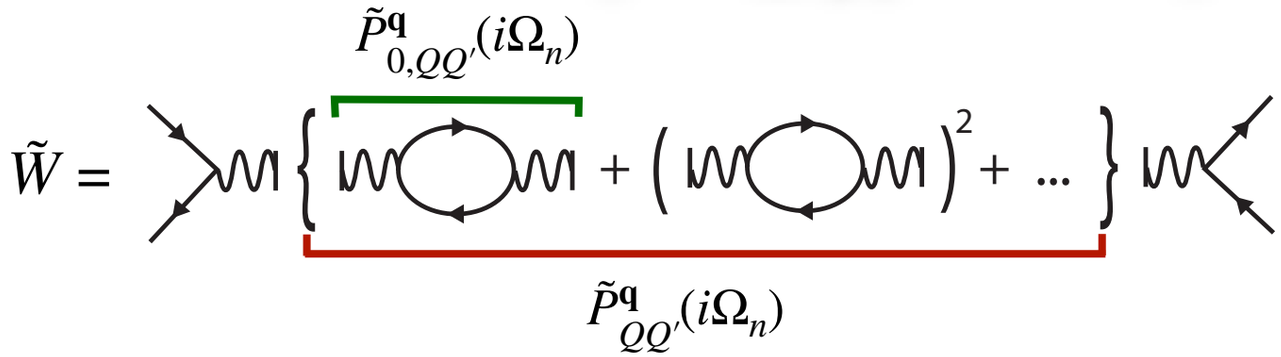
HF exchange diagram: $K_{ij}^{\mathbf{k}} = \frac{-1}{N_k} \sum_{\mathbf{k}'} \sum_{ab} G_{ab}^{\mathbf{k}'}(\tau = \beta^-) U_{i\,a\,b\,j}^{\mathbf{k}\mathbf{k}'\mathbf{k}'\mathbf{k}}$

- **Non-relativistic or spin-free relativistic HF**:
  scf_type=HF

- **Two-component relativistic HF**:
  scf_type=HF_X2C1e
  X2C=true

- **MPI parallelization over the *(ink)* independent *k*-points**

Self-consistent GW (scf_type=GW, GW_X2C1e)



$$\tilde{P}^{\mathbf{q}}_{0,QQ'}(\tau) = \frac{-1}{N_k}\sum_{\mathbf{k}}\sum_{\sigma}\sum_{abcd} V^{\mathbf{k},\mathbf{k}+\mathbf{q}}_{d\ a}(Q) G^{\mathbf{k}}_{c\sigma,d\sigma}(-\tau) G^{\mathbf{k}+\mathbf{q}}_{a\sigma\ b\sigma}(\tau) V^{\mathbf{k}+\mathbf{q},\mathbf{k}}_{b\ c}(Q')$$

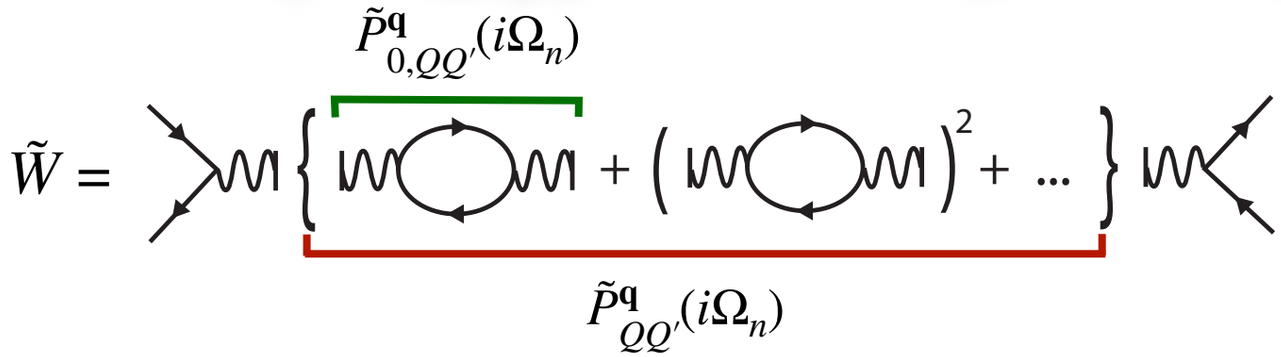
- **Non-relativistic or spin-free relativistic GW**: scf_type=GW
- **Two-component relativistic GW**: scf_type=GW_X2C1e X2C=true

- **Two-level MPI parallelization**
  - controlled by **ntauspinprocs** - number of processes for the second layer of MPI parallelization on τ and spin axes
  - *(ink)* independent ***q***-points
  - *(ni)* independent τ-points

Examples:
Given *N* MPI processes and ntauspinprocs = *x* —>

- *N/x* of processes on the ***q***-axis
- *x* of processes on the τ-axis

Cuda GW (scf_type=cuGW, cuGW_X2C1e)



$$\tilde{P}^{\mathbf{q}}_{0,QQ'}(\tau) = \frac{-1}{N_k} \sum_{\mathbf{k}} \sum_{\sigma} \sum_{abcd} V^{\mathbf{k},\mathbf{k}+\mathbf{q}}_{d\ a}(Q) G^{\mathbf{k}}_{c\sigma,d\sigma}(-\tau) G^{\mathbf{k}+\mathbf{q}}_{a\sigma\ b\sigma}(\tau) V^{\mathbf{k}+\mathbf{q},\mathbf{k}}_{b\ c}(Q')$$

- **Non-relativistic or spin-free relativistic GW**: scf_type=cuGW
- **Two-component relativistic GW**: scf_type=cuGW_X2C1e X2C=true
- **Parallelization scheme**
  - MPI parallelization over the ***q***-axis
  - Asynchronous streams for the summation over ***k***-points
  - Batched ZGEMM over the τ-axis controlled by **nt_batch** - size of τ batch in cuda GW solver

Iterative solvers for self-consistency loop

$$G_{ij}^{\mathbf{k}}(i\omega_n) = [(i\omega_n + \mu)S^{\mathbf{k}} - H_0^{\mathbf{k}} - \Sigma^{\mathbf{k}}(i\omega_n)]_{ij}^{-1}$$

$$\Sigma_{ij}^{\mathbf{k}} = \mathscr{F}^{\mathrm{MBPT}}[G]$$

Parameters for iterative solvers

- damp - damping parameter for self-energy and Fock matrix (1: no damp, 0: full damp)
- DIIS_size - DIIS space size
- DIIS_start - iteration where the DIIS solver begins
- DIIS_interval - number of iteration between two DIIS extrapolations

More elaborate iterative solvers can be found in
https://green.physics.lsa.umich.edu/mw19/index.php?title=DIIS_and_Convergence_Acceleration

Example: Silicon

- Basis set: *gth-dzvp-molopt-sr,* Pseudopotential: *gth-pbe*

After running UGF2/script/init_data_df.py…

- input.h5: input file with all the one-electron quantities
- df_int: density-fitted Coulomb integrals for GW/GF2 with Madelung constant correction
- df_hf_int: density-fitted Coulomb integrals for HF

run.sh:

```
#!/bin/bash
#SBATCH -p super,batch
#SBATCH -t 48:00:00
#SBATCH -N 2
#SBATCH -n 128
#SBATCH -c 2
#SBATCH -o Si6_GW.o%j
#SBATCH -J Si6
#SBATCH --exclusive

#OpenMP settings:
export HDF5_USE_FILE_LOCKING=FALSE

export INT_DIR=/pauli-storage/cnyeh/Si/nk6/integrals/
export BinDir=/home/cnyeh/Project/UGF2_CQMP/build/
export IR_DIR=/home/cnyeh/Project/UGF2_CQMP/MB_analysis/data/ir_grid/

date
srun -n 128 -c 2 --cpu_bind=cores $BinDir/UGF2 gw_param --dfintegral_file=$INT_DIR/df_int
 --HF_dfintegral_file=$INT_DIR/df_hf_int --input_file=input.h5 --TNL=$IR_DIR/1e6_168.h5 -
-TNL_B=$IR_DIR/1e6_168.h5 --Results=sim.h5 --ntauspinprocs=2
date
```

gw_param:

```
nel_cell=8
nao=26
nk=216
ink=112
ns=1
NQ=124
ni=168
beta=1000
itermax=15
rst=false
scf_type=GW
IR=true
CONST_DENSITY=true
E_thr=1e-5
damp=0.7
DIIS_size=4
DIIS_start=4
DIIS_interval=1
```

Example: Silicon

- Basis set: *gth-dzvp-molopt-sr,* Pseudopotential: *gth-pbe*

input.h5:
The output file after computing the matrix elements using UGF2/script/init_data_df.py. Most of the GW parameters can be found here.

```
(base) [cnyeh@pauli-master GW]$ h5dump -n input.h5
HDF5 "input.h5" {
FILE_CONTENTS {
 group      /
 dataset    /Cell
 group      /HF
 dataset    /HF/Energy
 dataset    /HF/Energy_nuc
 dataset    /HF/Fock-k
 dataset    /HF/H-k
 dataset    /HF/Nk
 dataset    /HF/S-k
 dataset    /HF/madelung
 dataset    /HF/mo_coeff
 dataset    /HF/mo_energy
 dataset    /HF/nk
 group      /grid
 dataset    /grid/conj_list
 dataset    /grid/index
 dataset    /grid/ink
 dataset    /grid/ir_list
 dataset    /grid/k_mesh
 dataset    /grid/k_mesh_scaled
 dataset    /grid/weight
 group      /mulliken
 dataset    /mulliken/Zs
 dataset    /mulliken/last_ao
 group      /params
 dataset    /params/NQ
 dataset    /params/nao
 dataset    /params/nel_cell
 dataset    /params/nk
 }
}
```

Fock matrix from HF or DFT, $F_{ij}^{\mathbf{k}}$: (ns, nk, nao, nao)

$(H_0^{\mathbf{k}})_{ij}$: (ns, nk, nao, nao)

Overlap matrix $S_{ij}^{\mathbf{k}}$: (ns, nk, nao, nao)

- nel_cell - number of electrons per unit cell
- nao - number of atomic orbitals per unit cell
- nk - number of k-points
- ink - number of reduced k-points in the presence of inversion symmetry
- NQ - number of auxiliary basis per unit cell

Example: Silicon

- Basis set: *gth-dzvp-molopt-sr,* Pseudopotential: *gth-pbe*

sim.h5: UGF2 output file

```
group      /iter9/Energy
dataset    /iter9/Energy/2nd
dataset    /iter9/Energy/hf
dataset    /iter9/Fock-k
group      /iter9/G_tau
dataset    /iter9/G_tau/data
dataset    /iter9/G_tau/mesh
group      /iter9/Selfenergy
dataset    /iter9/Selfenergy/data
dataset    /iter9/Selfenergy/mesh
dataset    /iter9/mu
```

Correlation energy

Hartree-Fock energy

Fock matrix $F_{ij}^{\mathbf{k}}$: (ns, ink, nao, nao)

Green's function $G_{ij}^{\mathbf{k}}(\tau)$: (ni+2, ns, ink, nao, nao)

The τ-mesh for Green's function:
IR τ-mesh + $\tau = 0,\ \beta$

Self-energy $\Sigma_{ij}^{\mathbf{k}}(\tau)$: (ni+2, ns, ink, nao, nao)

Chemical Potential

Comments

- Define the problem
  - Gaussian basis set → basis set convergence
  - Size of ***k***-mesh: → finite-size effect

    Typically, 6x6x6 is good enough for insulator. The larger the unit cell, the less ***k***-points needed.

- Compute matrix elements of the Hamiltonian
  - Python scripts using PySCF
  - Density fitting for the two-electron Coulomb interaction

    → This step could be very time-consuming!

    J. Chem. Phys. 147, 164119 (2017)
    J. Chem. Phys. 154, 131104 (2021)

- **Run many-body perturbation theory (MBPT) using UGF2**
  - GW $\sim O(N_\tau N_k^2 N_{ao}^4)$
  - 10~20 iterations to reach the self-consistency
  - beta is typically smaller than 1000 (i.e. ~ 315 K)