Hands-on ComCTQMC

Corey Melnick (cmelnick@bnl.gov)

Brookhaven National Laboratory

comscope
from codes to spectroscopies

Outline

1. A very brief tour of the code
2. Compilation
3. Usage
4. Results
5. Troubleshooting and Guidance

Tour

cd ~/codes/Compiled_ComsuiteCode/ctqmc.June.18.2021

- The software is comprised of two main codes, CTQMC and EVALSIM
  - CTQMC is the main algorithm which solves the impurity problem
    - Massively parallelizable with ideal scaling in the main algorithm
    - GPU accelerated for f-shell or CDMFT problems
  - EVALSIM is post-processing algorithms which translate the raw results into observables
    - Nearly serial (but runs quickly)
- These are C++ and CUDA codes which use MPI to handle parallelization.
  - LAPACK / BLAS are used for linear algebra on the CPU (or host)
  - CUTLASS is used for matrix multiplications on the GPU (or device)
- There is a python script, plot.py, which helps users to inspect their results
- There is a library libctqmc.so which helps users to embed CTQMC in their own project

Compilation

- Two options are available for compilation: GNU make, or Cmake

- GNU make
  - First, you need to configure makefile.in so that the makefile knows where to find the various libraries, to know which compilers to use, etc.
  - Makefile.help.in provides an explanation of many options
  - Makefile_Examples/ provides many examples of configuration files for different systems
    - For your virtual box, you can use Makefile_Examples/MakefileGNU.in

  cp Makefile_Examples/MakefileGNU.in Makefile.in
  make
- Cmake
  - Cmake is a way to generate a configuration file automatically

  mkdir build; cd build
  cmake .. (won't work on your virtual machine – need to install newer cmake)
  make

Usage

- First, let's go to a directory

cd examples/Hubbard

- To use ComCTQMC, you must first make an input file parameterizing the system. Let's take a look at the existing file

vim params.json

(use whichever editing tool you'd like, ":q" exits vim)

- We also need a file describing the hybridization functions

vim hyb.json

- Now, we can run

../../bin/CTQMC params

../../bin/EVALSIM params

(*params* is the name you give to the input file)

Output: stdout

There are a few lines from the stdout (output to your terminal or log file) which are worth noting.

Number of invariant subspaces: 4
Dimension of the biggest subspace: 1

- If you recall from my earlier talk – we partition the Hilbert space into subspaces. The more subspaces we get and the smaller they are, the faster CTQMC will run.
- This problem is as simple as it gets (1-band hubbard), so our operator matrices are only of rank 1!

partition eta = 1

- This gives a list of the relative size of the configuration spaces sampled
- If you are using the worm algorithm, you'll see a list of eta's. Each eta corresponds to the size of that space relative to the partition space.
- Try changing the input "green-": { … } field to turn on the green's function configuration space

Output: files

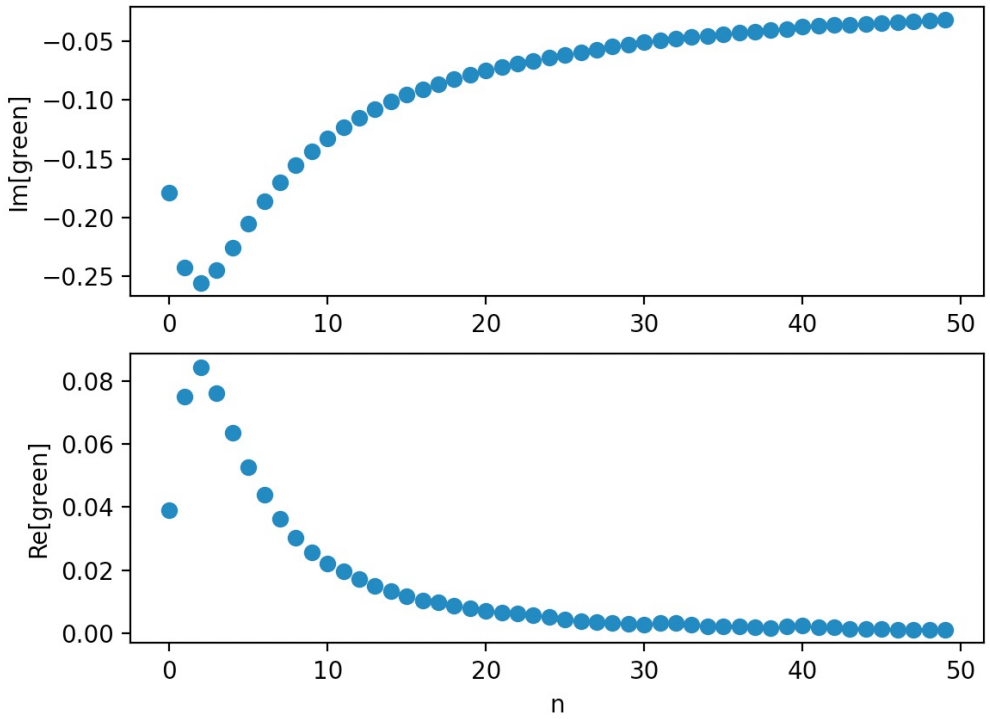
- A few files are produced by CTQMC
  1. *params.info.json*
     - Description of the run
  2. *defaults.json*
     - All of your input parameters, plus any options which were left to the default values
  3. *params.meas.json*
     - The raw results in base 64.
  4. *params.err.json* (when run in parallel with mpirun -n {N > 1})
     - Error estimates for observables
  5. *config_x.json*
     - A description of the final state of each Markov chain (reduces thermalization time)

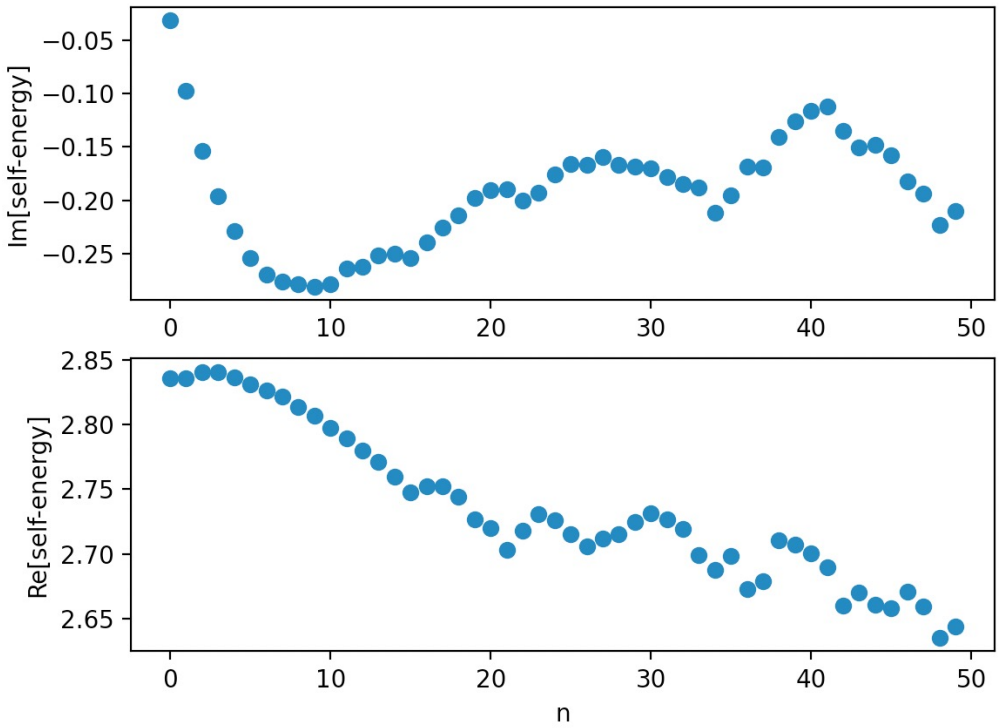
- And one file is produced by EVALSIM
  1. *params.obs.json*
     - The collection of observables

Output

1. Let's look at the self-energy and green's function

python ../../bin/plot.py --field=green ; python ../../bin/plot.py --field=self-energy





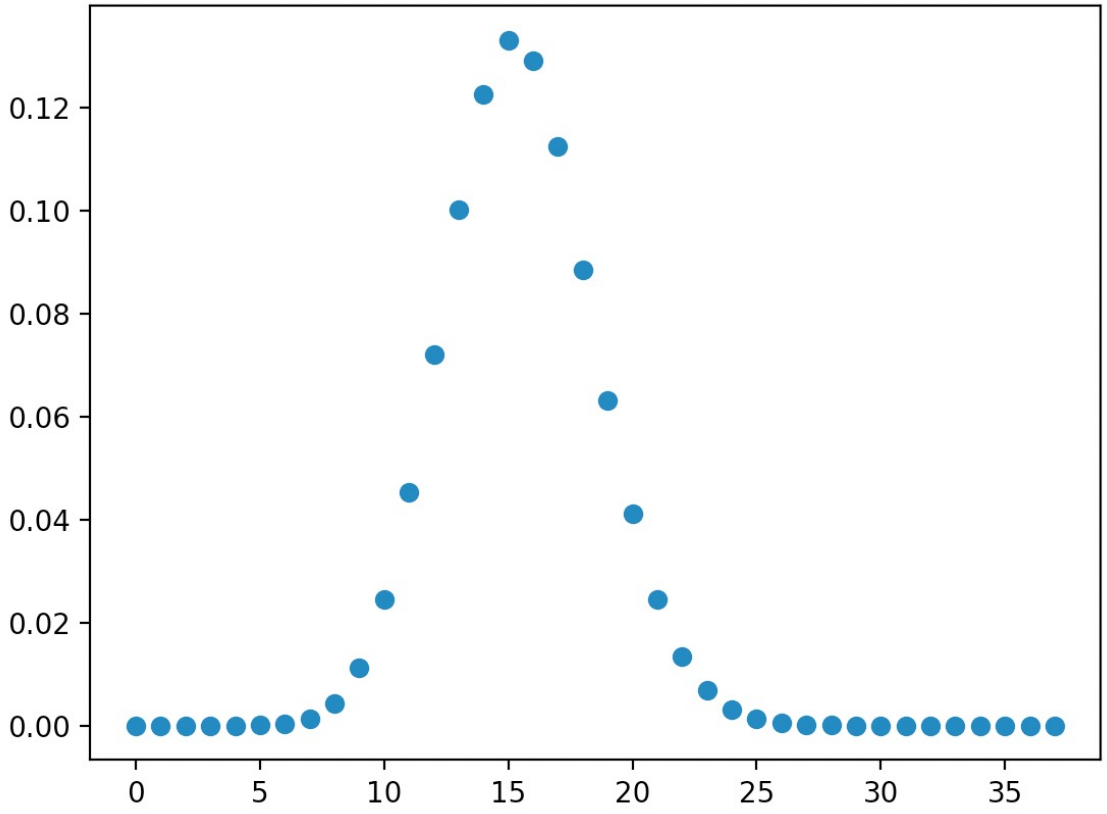
Brookhaven National Laboratory

COMSCOPE
from codes to spectroscopies

Output

1. Let's look at the expansion histogram
   python ../../bin/plot.py --field="expansion histogram"

- This gives you idea of how hard your problem is.
  - Here, we see that we are having to deal with roughly k=16, a fairly low order of expansion.
  - If we reduced the temperature, we'd end up with larger and larger expansions.



Troubleshooting and Guidance

- How do I create inputs?
  - Typically, CTQMC will be embedded in another code!
    - DMFT
    - Periodic-Anderson model
    - CDMFT
    - DCA
  - These codes need to handle the generation of your inputs, as they need to compute the hybridization functions and supply a description of the atomic or local Hamiltonian

  - We provide some options to simplify a description of the interaction tensor
    - Slater-Condon interaction U_{ijkl} in some basis
      - In relativistic (“coupled”)
      - Non-relativistic (“product real”)
      - With or without some transformation
      - With or without Ising approximation

Troubleshooting and Guidance

- How much time do I give to ComCTQMC?
  1. Thermalization time
     - No great way to tell if a Markov chain is thermalized before measurement starts.
     - Fortunately, *config_x.json* saves your old configurations and removes the need for substantial thermalization (if you are converging DMFT)
     - If you are doing a one-shot CTQMC, you need to test that results do not change as you increase thermalization.

  2. Measurement time
     - Running in parallel: check *params.err.json*
       - Relative error of 10-20% good enough for DMFT (params.err.json has raw error)

$$\frac{\boldsymbol{error}}{\boldsymbol{time}} \propto \boldsymbol{time}^{-\alpha}$$

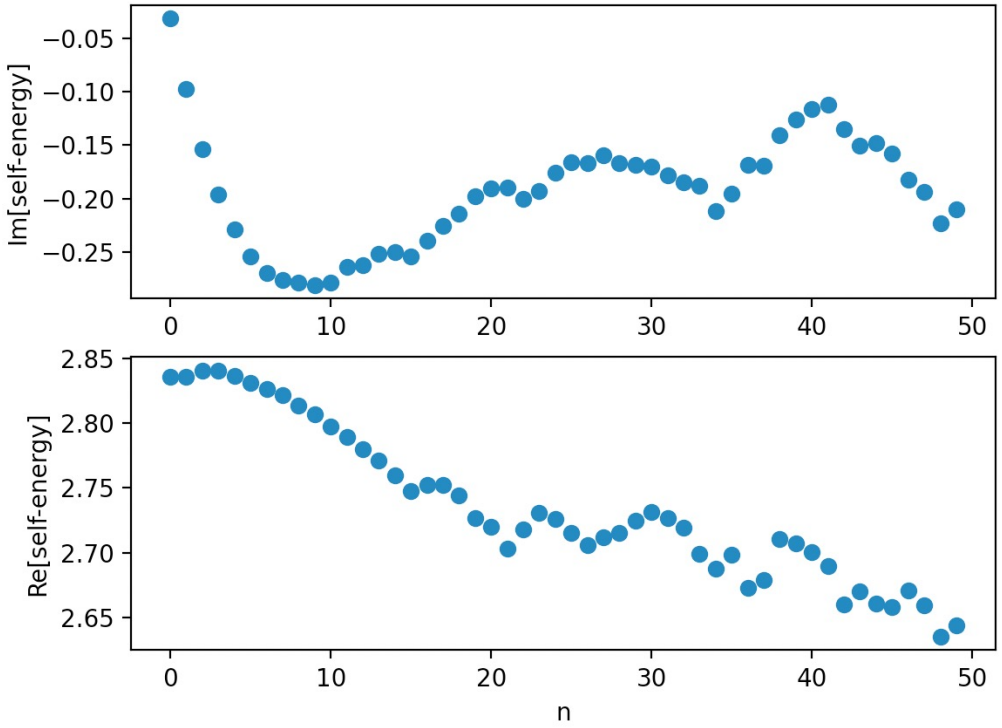
     - Eye test: Just look at your self-energy and see that it's smooth at high frequencies!

Brookhaven National Laboratory

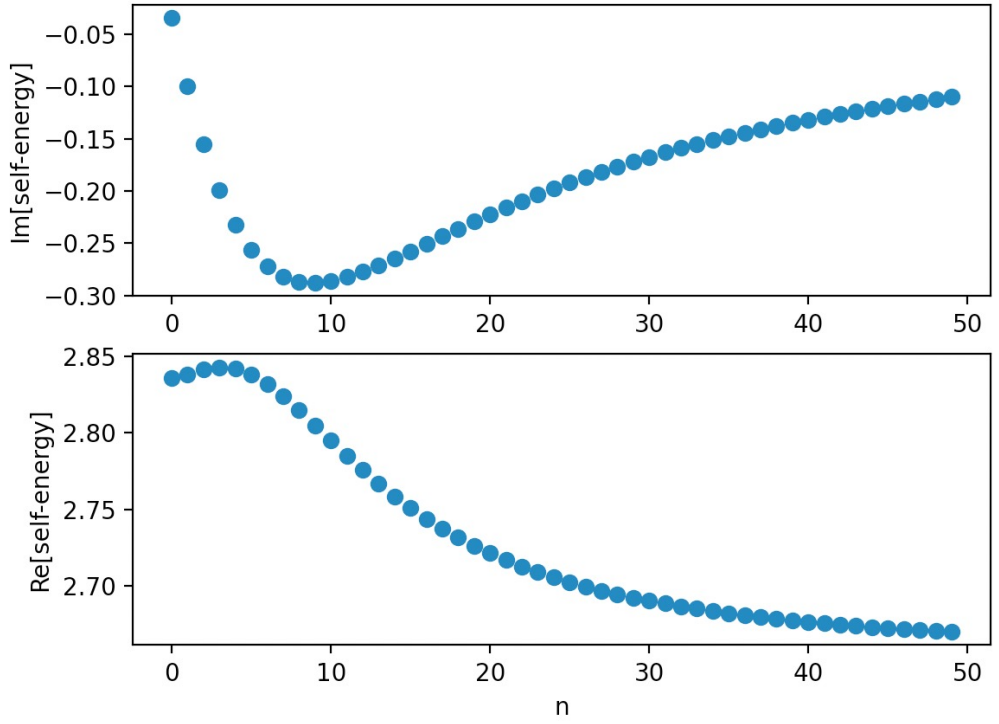
COMSCOPE
from codes to spectroscopies

Troubleshooting and Guidance

- Eye test: Just look at your self-energy and see that it's smooth at high frequencies!



- Not enough time! (Too wobbly)



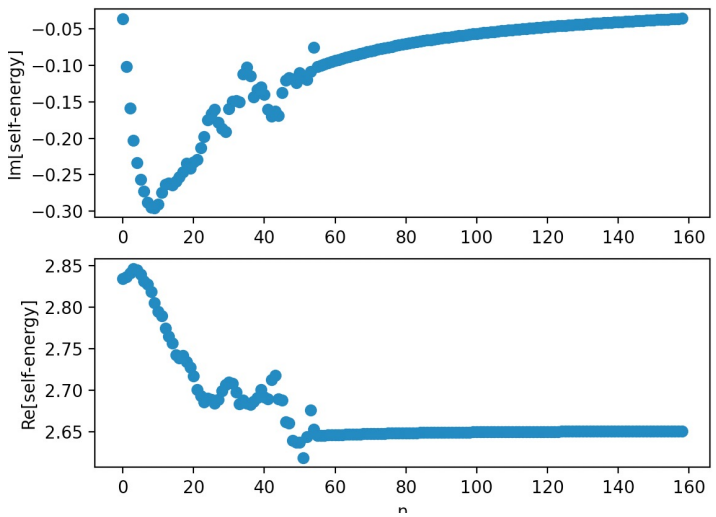
More than enough time

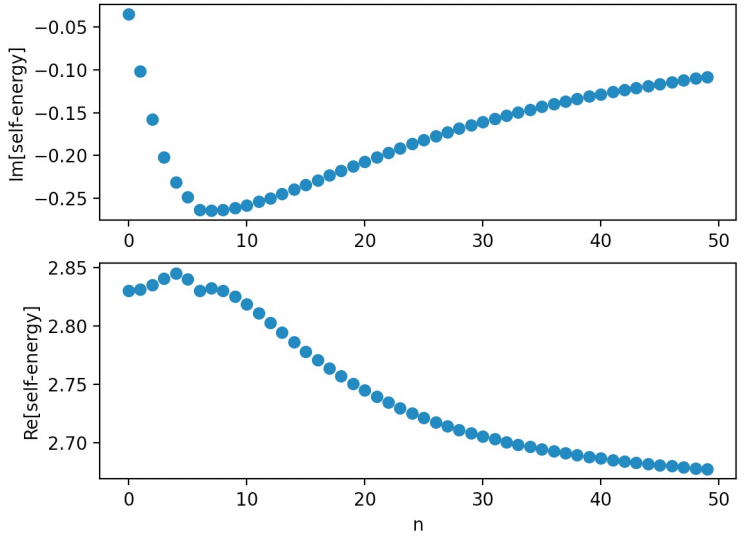
Brookhaven National Laboratory

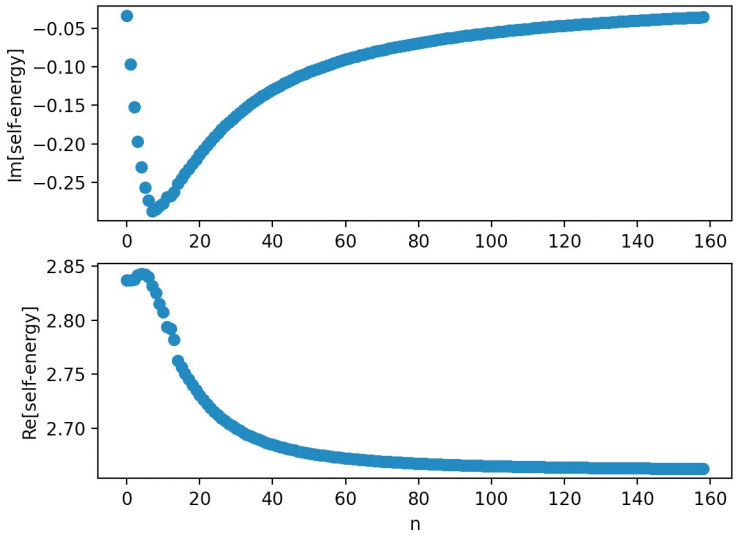
comscope
from codes to spectroscopies

Troubleshooting and Guidance

- How high of an energy cutoff?
  - 10-15 eV tends to be good
  - Check that the high frequency tail looks appropriate
  - Too high = high error | Too low = “kink” at the transition | Just right = smooth transition







Brookhaven National Laboratory

comscope
from codes to spectroscopies

Troubleshooting and Guidance

- Quantum numbers
  - Array of values corresponding to each orbital on the impurity, $\boldsymbol{q_{iI}}$
    - These define bilinears used by CTQMC to deal with dynamical interactions
      - $\boldsymbol{Q_J = q_{iJ} c_i^\dagger c_i}$
    - We require
      - $\boldsymbol{[H, Q_I] = 0}$ (we will throw an error message if this is not met)
  - Unless you are
    - Measuring a quantum number susceptibility
    - Applying a dynamical interaction on that quantum number
  - Save yourself some trouble by testing quantum numbers after CTQMC
    - Enter them into *params.json* after CTQMC but before EVALSIM
  - Examples:
    - N = [1,1,…,1] is *always* a good quantum number
    - S_z = [0.5,…,-0.5,…] is *often* a good quantum number for a non-relativistic impurity

Troubleshooting and Guidance

- Input cutoff energies
  - Hybridization (and dynamical functions) should be smoothly going to zero at high frequencies.
  - You need to provide sufficient frequencies to reach this asymptotic behavior.

- Non-physical inputs can lead to crashes in the algorithm (overflows and underflows)
  - Crash like: “Zahl:: constructor is not a number”
  - Input cutoff energies are too low
  - Hybridization functions not going to zero
  - Fourier transform of Hybridization function extremely numerically sensitive (likely unphysical)