Monte Carlo by Chuang Chen

Jinyuan Wu

November 6, 2021

1 Strongly correlated systems

Lattice models are often used to capture the essential physics in strongly correlated systems. A model of correlated electrons is the **Hubbard model**, i.e.

$$H = -t \sum_{\langle \boldsymbol{i}, \boldsymbol{j} \rangle, \sigma} c^\dagger_{\boldsymbol{i}\sigma} c_{\boldsymbol{j}\sigma} + U \sum_{\boldsymbol{i}} n_{\boldsymbol{i}\uparrow} n_{\boldsymbol{i}\downarrow} + \mu \sum_{\boldsymbol{i}, \sigma} n_{\boldsymbol{i}\sigma}. \tag{1}$$

It should be notice that a relative strong interaction does not necessarily cause strong correlation. Consider the following **Holstein model** about electron-phonon interaction:

$$H = H_{\text{el}} + H_{\text{lat}} + H_{\text{int}}, \tag{2}$$

where

$$H_{\text{el}} = -t \sum_{\langle \boldsymbol{i}, \boldsymbol{j} \rangle, \sigma} c^\dagger_{\boldsymbol{i}\sigma} c_{\boldsymbol{j}\sigma} + \mu \sum_{\boldsymbol{i}, \sigma} n_{\boldsymbol{i}\sigma}, \tag{3}$$

$$H_{\text{lat}} = \sum_{\boldsymbol{i}} \left(\frac{1}{2} M \Omega^2 X_{\boldsymbol{i}}^2 + \frac{1}{2M} P_{\boldsymbol{i}}^2 \right), \tag{4}$$

and

$$H_{\text{int}} = g \sum_{\boldsymbol{i}, \sigma} n_{\boldsymbol{i}\sigma} X_{\boldsymbol{i}}. \tag{5}$$

See I. Esterlis et al. PRB 97.140501 2018

It can be seen that before $\lambda_0 = 0.4$ the susceptibility perturbatively calculated and the susceptibility calculated with DQMC agree well, but when λ_0 is between 0.4 and 0.5, the DQMC results show a clear tendency of divergence, while the perturbation theory[1] does not show the trend. When the temperature is high, the perturbation theory works again.

2 The principles of DQMC

We can always introduce a (discrete) Hubbard-Stratonovich transformation to decouple the fermion interaction terms in the Hamiltonian into fermion-bosonic field interaction terms, and then use the formula

$$\operatorname{Tr} \prod_\tau \mathrm{e}^{-\Delta\tau \mathbf{c}^\dagger \mathbf{T}(\tau) \mathbf{c}} = \det \left(1 + \prod_\tau \mathrm{e}^{-\Delta\tau \mathbf{T}(\tau)} \right), \tag{6}$$

where $\mathbf{T}(\tau)$ includes both the hopping matrix in the fermion kinetic energy term and the interaction between the fermions and the bosonic fields (which are also included into the discrete path integral and therefore $\mathbf{T}$ depends on τ). So the partition function is finally

$$Z = \det \left(1 + \prod_\tau \mathbf{B}_\tau \right) \prod_\tau \mathrm{e}^{-\Delta\tau H_{\text{bosonic part}}}. \tag{7}$$

Monte Carlo simulation of this partition function is called **determinantal quantum Monte Carlo (DQMC)**.

[1] The perturbation theory used in the paper is something like RPA, which works best when there are no competing interaction channels, for example the electronic structure of strongly correlated systems whose basic degrees of freedom are still electron-like and the only interaction channel is the screened Coulomb interaction, or a system with BCS pairing channel only, but not a system with competing BCS and CDW orders.

World-line and determinantal quantum Monte Carlo Methods for spins, phonons and electrons F.F Assaad

The error in DQMC is composed of the error introduced by the Trotter error and the statistical error when doing the simulation.

The main obstacle of DQMC, however, is not the error but the **sign problem**, i.e. a weight in (7) may contains negative sign. The problem to identify and solve the sign problem has been argued to be *NP-hard*, and it is not feasible to obtain a general solution of the problem. Certain symmetry (for example an anti-unitary symmetry) in the model may protect us from the sign problem, and when the

Identification: PRB 71.155115, 115.250601, 116.250601

It is possible to simulate with the sign. A rule of thumb is that if the average sign is above 0.1, DQMC works well.

3 The update scheme

numerical stabilization

4 Tricks in implementation

Fortran is often used to implement DQMC. The most frequently used compiler is Intel Fortran because the binary code generated is so quick. GCC and PCFortran are also used. Intel has its own math library MKL, and OpenBLAS is also a good choice. Usually we have two platforms:

- Intel Xeon CPU, Intel Fortran Compiler, MKL.
- ARM, gFortran, BLAS + LAPACK.

Parallelization can be done with:

- MPI, for multiple Markov chains.
- OpenMP, for very large system size, where the memory is not sufficient and as a result it is impossible to start many Markov chains, and many cores are just idle.
- GPU.

GPU acceleration needs some further remark. Usually people write CUDA C, but there is a Fortran counterpart dubbed as CUDA Fortran. The compiler is an modified version of PGFortran with a highly incomplete cuBLAS library. A library called Magma should be used in place of LAPACK.

A CPU+GPU program does not necessarily perform better than a purely CPU one. GPU is not good at calculating the accept ratio, so the accept ratio is to be calculated by the CPU, but that means the Green functions must be passed from the video memory to the memory frequently, which is quite time-consuming. This additional amount of time often overweighs the time saved by the fast matrix multiplication done by the GPU when updating the Green function. The numerical stabilization and the Green function wrapping process involve both loops and matrix multiplication. The former should better be done on CPU while the latter should better be done on GPU. Therefore, if the number of loops in these two processes is not large enough, the CPU+GPU version performs *worse* than the pure CPU version.

5 Some applications