CHAPTER I

Numerical Methods for Quantum Monte Carlo Simulations of the Hubbard Model*

Zhaojun Bai
Department of Computer Science and
Department of Mathematics
University of California
Davis, CA 95616, USA
E-mail: bai@cs.ucdavis.edu

Wenbin Chen
School of Mathematical Sciences
Fudan University
Shanghai 200433, China
E-mail: wbchen@fudan.edu.cn

Richard Scalettar
Department of Physics
University of California
Davis, CA 95616, USA
E-mail: scalettar@physics.ucdavis.edu

Ichitaro Yamazaki
Department of Computer Science
University of California
Davis, CA 95616, USA
E-mail: yamazaki@cs.ucdavis.edu

Abstract

One of the core problems in materials science is how the interactions between electrons in a solid give rise to properties like

*This work was partially supported by the National Science Foundation under Grant 0313390, and Department of Energy, Office of Science, SciDAC grant DE-FC02 06ER25793. Wenbin Chen was also supported in part by the China Basic Research Program under the grant 2005CB321701.

magnetism, superconductivity, and metal-insulator transitions? Our ability to solve this central question in quantum statistical mechanics numerically is presently limited to systems of a few hundred electrons. While simulations at this scale have taught us a considerable amount about certain classes of materials, they have very significant limitations, especially for recently discovered materials which have mesoscopic magnetic and charge order.

In this paper, we begin with an introduction to the Hubbard model and quantum Monte Carlo simulations. The Hubbard model is a simple and effective model that has successfully captured many of the qualitative features of materials, such as transition metal monoxides, and high temperature superconductors. Because of its voluminous contents, we are not be able to cover all topics in detail; instead we focus on explaining basic ideas, concepts and methodology of quantum Monte Carlo simulation and leave various part for further study. Parts of this paper are our recent work on numerical linear algebra methods for quantum Monte Carlo simulations.

1 Hubbard model and QMC simulations

The Hubbard model is a fundamental model to study one of the core problems in materials science: How do the interactions between electrons in a solid give rise to properties like magnetism, superconductivity, and metal-insulator transitions? In this lecture, we introduce the Hubbard model and outline quantum Monte Carlo (QMC) simulations to study many-electron systems. Subsequent lectures will describe computational kernels of the QMC simulations.

1.1 Hubbard model

The two-dimensional Hubbard model [8, 9] we shall study is defined by the Hamiltonian:

$$\mathcal{H} = \mathcal{H}_K + \mathcal{H}_\mu + \mathcal{H}_V, \tag{1.1}$$

where $\mathcal{H}_K$, $\mathcal{H}_\mu$ and $\mathcal{H}_V$ stand for kinetic energy, chemical energy and potential energy, respectively, and are defined as

$$\begin{aligned}
\mathcal{H}_K &= -t \sum_{\langle i,j\rangle,\sigma} (c^\dagger_{i\sigma} c_{j\sigma} + c^\dagger_{j\sigma} c_{i\sigma}), \\
\mathcal{H}_\mu &= -\mu \sum_i (n_{i\uparrow} + n_{i\downarrow}) \\
\mathcal{H}_V &= U \sum_i (n_{i\uparrow} - \frac{1}{2})(n_{i\downarrow} - \frac{1}{2})
\end{aligned}$$

and

- i and j label the spatial sites of the lattice. $\langle i, j\rangle$ represents a pair of nearest-neighbor sites in the lattice and indicates that the electrons only hopping to nearest neighboring sites,
- the operators $c^\dagger_{i\sigma}$ and $c_{i\sigma}$ are the fermion creation and annihilation operators for electrons located on the ith lattice site with z component of spin-up ($\sigma = \uparrow$) or spin-down ($\sigma = \downarrow$), respectively,
- the operators $n_{i\sigma} = c^\dagger_{i\sigma} c_{i\sigma}$ are the number operators which count the number of electrons of spin σ on site i,
- t is the hopping parameter for the kinetic energy of the electrons, and is determined by the overlap of atomic wave functions on neighboring sites,
- U is the repulsive Coulomb interaction between electrons on the same lattice site. The term $Un_{i\uparrow}n_{i\downarrow}$ represents an energy cost U for the site i has two electrons and describes a local repulsion between electrons,
- μ is the chemical potential parameter which controls the electron numbers (or density).

Note that we consider the case of a half-filled band. Hence the Hamiltonian is explicitly written in particle-hole symmetric form.

The expected value of a physical observable $\mathcal{O}$ of interest, such as density-density correlation, spin-spin correlation or magnetic susceptibility, is given by

$$\langle \mathcal{O} \rangle = \mathrm{Tr}(\mathcal{O}\mathcal{P}), \tag{1.2}$$

where $\mathcal{P}$ is a distribution operator defined as

$$\mathcal{P} = \frac{1}{\mathcal{Z}} e^{-\beta\mathcal{H}}, \tag{1.3}$$

and $\mathcal{Z}$ is the partition function defined as

$$\mathcal{Z} = \mathrm{Tr}(e^{-\beta\mathcal{H}}), \tag{1.4}$$

and β is proportional to the inverse of the product of the Boltzmann's constant k_B and the temperature T:

$$\beta = \frac{1}{k_B T}.$$

β is referred to as an *inverse temperature.*

"Tr" is a trace over the Hilbert space describing all the possible occupation states of the lattice:

$$\mathrm{Tr}(e^{-\beta\mathcal{H}}) = \sum_i \langle \psi_i | e^{-\beta\mathcal{H}} | \psi_i \rangle,$$

where $\{|\psi_i\rangle\}$ is an orthonormal basis of the Hilbert space. Note that the trace does not depend on the choice of the basis. A convenient choice of the basis is the so-called "occupation number basis (local basis)" as described below.

In a classical problem where $\mathcal{H} = E$ is the energy, a real variable, then $\exp(-\beta E)/Z$ is the probability, where $Z = \int e^{-\beta E}$. In quantum mechanics, as we shall see, we will need to recast the operator $\exp(-\beta\mathcal{H})$ into a real number. The "path integral representation" of the problem to do this was introduced by Feynman and Hibbs [3].

Remark 1.1. According to Pauli exclusion principle of electrons, there are four possible states at every site:

$\lvert\cdot\rangle$	no particle,
$\lvert\uparrow\rangle$	one spin up particle,
$\lvert\downarrow\rangle$	one spin down particle,
$\lvert\uparrow\downarrow\rangle$	two particles with different spin directions.

Therefore the dimension of the Hilbert space is 4^N, where N is the number of sites.

The actions of the spin creation operators $c_\sigma^\dagger$ on the four states are

	$\lvert\cdot\rangle$	$\lvert\uparrow\rangle$	$\lvert\downarrow\rangle$	$\lvert\uparrow\downarrow\rangle$
$c_\uparrow^\dagger$	$\lvert\uparrow\rangle$	0	$\lvert\uparrow\downarrow\rangle$	0
$c_\downarrow^\dagger$	$\lvert\downarrow\rangle$	$\lvert\uparrow\downarrow\rangle$	0	0

The actions of the spin annihilation operators c_σ are

	$\lvert\cdot\rangle$	$\lvert\uparrow\rangle$	$\lvert\downarrow\rangle$	$\lvert\uparrow\downarrow\rangle$
$c_\uparrow$	0	$\lvert\cdot\rangle$	0	$\lvert\downarrow\rangle$
$c_\downarrow$	0	0	$\lvert\cdot\rangle$	$\lvert\uparrow\rangle$

Remark 1.2. The states $|\cdot\rangle$ and $|\uparrow\rangle$ are the eigen-states of the number operator $n_\uparrow = c_\uparrow^\dagger c_\uparrow$:

$$n_\uparrow|\cdot\rangle = 0|\cdot\rangle = 0, \quad n_\uparrow|\uparrow\rangle = |\uparrow\rangle.$$

When the operator $n_\uparrow$ takes the actions on the states $|\downarrow\rangle$ and $|\uparrow\downarrow\rangle$, we have

$$n_\uparrow|\downarrow\rangle = 0, \quad n_\uparrow|\uparrow\downarrow\rangle = |\uparrow\downarrow\rangle.$$

The states $|\cdot\rangle$ and $|\downarrow\rangle$ are the eigen-states of the number operator $n_\downarrow = c_\downarrow^\dagger c_\downarrow$:

$$n_\downarrow|\cdot\rangle = 0|\cdot\rangle = 0, \quad n_\downarrow|\downarrow\rangle = |\downarrow\rangle.$$

When the operator $n_\downarrow$ on the state $|\uparrow\rangle$ and $|\uparrow\downarrow\rangle$, we have

$$n_\downarrow|\uparrow\rangle = 0, \quad n_\downarrow|\uparrow\downarrow\rangle = |\uparrow\downarrow\rangle.$$

The operator $U(n_\uparrow - \frac{1}{2})(n_\downarrow - \frac{1}{2})$ describes the potential energy of two electrons with different spin directions at the same site:

$$\begin{aligned} U(n_\uparrow - \tfrac{1}{2})(n_\downarrow - \tfrac{1}{2}) \ : |\cdot\rangle &= +\tfrac{U}{4}|\cdot\rangle, \quad |\uparrow\rangle = -\tfrac{U}{4}|\uparrow\rangle, \\ |\downarrow\rangle &= -\tfrac{U}{4}|\downarrow\rangle, |\uparrow\downarrow\rangle = +\tfrac{U}{4}|\uparrow\downarrow\rangle. \end{aligned}$$

These eigenenergies immediately illustrate a key aspect of the physics of the Hubbard model: The single occupied states $|\uparrow\rangle$ and $|\downarrow\rangle$ are lower in energy by U (and hence more likely to occur). These states are the ones which have nonzero magnetic moment $m^2 = (n_\uparrow - n_\downarrow)^2$. One therefore says that the Hubbard interaction U favors the presence of magnetic moments. As we shall see, a further question (when t is nonzero) is whether these moments will order in special patterns from site to site.

Remark 1.3. The creation operators $c_{i\sigma}^\dagger$ and the annihilation operators $c_{i\sigma}$ anticommute:

$$\begin{aligned} \{c_{j\sigma}, c_{\ell\sigma'}^\dagger\} &= \delta_{j\ell}\delta_{\sigma\sigma'}, \\ \{c_{j\sigma}^\dagger, c_{\ell\sigma'}^\dagger\} &= 0, \\ \{c_{j\sigma}, c_{\ell\sigma'}\} &= 0, \end{aligned}$$

where the anticommutator of two operators a and b is defined by $ab+ba$, i.e. $\{a,b\} = ab + ba$, and $\delta_{j\ell} = 1$ if $j = \ell$, and otherwise, $\delta_{j\ell} = 0$.

If we choose $\ell = j$ and $\sigma = \sigma'$ in the second anticommutation relation, we conclude that $(c_{j\sigma}^\dagger)^2 = 0$. That is, one cannot create two electrons on the same site with the same spin (Pauli exclusion principle). Thus the anticommutation relations imply the Pauli principle. If the site or spin indices are different, the anticommutation relations tell us that exchanging the order of the creation (or destruction) of two electrons introduces a minus sign. In this way the anticommutation relations also guarantee that the wave function of the particles being described is *antisymmetric*, another attribute of electrons (fermions). Bosonic particles (which have *symmetric* wave functions) are described by creation and destruction operators which commute.

Remark 1.4. When the spin direction σ and the site i are omitted, a quantization to describe the states is

$$\begin{aligned} &|0\rangle : \text{no particle}, \\ &|1\rangle : \text{one particle}. \end{aligned}$$

The actions of the creation and destruction operators on the states are

$$\begin{aligned} c\ \ &: |0\rangle \to 0, \quad |1\rangle \to |0\rangle, \\ c^\dagger &: |0\rangle \to |1\rangle, |1\rangle \to 0, \end{aligned} \tag{1.5}$$

Subsequently, the eigen-states of the number operator $n = c^\dagger c$ are

$$n \ : \ |0\rangle = 0, \quad |1\rangle = |1\rangle.$$

In addition, the operator $c_i^\dagger c_{i+1}$ describes the kinetic energy of the electrons on nearest neighbor sites:

$$\begin{aligned} c_i^\dagger c_{i+1} : |00\rangle &\to 0, \, |01\rangle \to |10\rangle, \\ |10\rangle &\to 0, \, |11\rangle \to c_i^\dagger |10\rangle \to 0. \end{aligned}$$

Therefore, if there is one particle on the $i+1$th site, and no particle on the ith site, the operator $c_i^\dagger c_{i+1}$ annihilates the particle on the $i+1$th site and creates one particle on the ith site. We say that the electron hops from site $i+1$ to site i after the action of the operator $c_i^\dagger c_{i+1}$.

1.1.1 Hubbard model with no hopping

Let us consider a special case of the Hubbard model, namely, there is only one site and no hopping, $t = 0$. Then the Hamiltonian $\mathcal{H}$ is

$$\mathcal{H} = U(n_\uparrow - \frac{1}{2})(n_\downarrow - \frac{1}{2}) - \mu(n_\uparrow + n_\downarrow).$$

It can be verified that the orthonormal eigen-states ψ_i of the operator n_σ are the eigen-states of the Hamiltonian $\mathcal{H}$:

$$\begin{aligned} \mathcal{H} : |\cdot\rangle &= \tfrac{U}{4}|\cdot\rangle, & |\uparrow\rangle &= \left(\tfrac{U}{4} - (\mu + \tfrac{U}{2})\right)|\uparrow\rangle, \\ |\downarrow\rangle &= \left(\tfrac{U}{4} - (\mu + \tfrac{U}{2})\right)|\downarrow\rangle, & |\uparrow\downarrow\rangle &= \left(\tfrac{U}{4} - 2\mu\right)|\uparrow\downarrow\rangle. \end{aligned}$$

The Hamiltonian $\mathcal{H}$ is diagonalized under the basis $\{\psi_i\}$:

$$\mathcal{H} \longrightarrow \left(\langle\psi_i|\mathcal{H}|\psi_j\rangle\right) = \begin{bmatrix} \frac{U}{4} & & & \\ & \frac{U}{4} - \left(\mu + \frac{U}{2}\right) & & \\ & & \frac{U}{4} - \left(\mu + \frac{U}{2}\right) & \\ & & & \frac{U}{4} - 2\mu \end{bmatrix}.$$

Consequently, the operator $e^{-\beta\mathcal{H}}$ is diagonalized:

$$e^{-\beta\mathcal{H}} \longrightarrow e^{-\frac{U\beta}{4}} \mathrm{diag}\left(1, e^{\beta(U/2+\mu)}, e^{\beta(U/2+\mu)}, e^{2\mu\beta}\right).$$

The partition function $\mathcal{Z}$ becomes

$$\mathcal{Z} = \mathrm{Tr}(e^{-\beta\mathcal{H}}) = \sum_i \langle\psi_i|e^{-\beta\mathcal{H}}|\psi_i\rangle \longrightarrow Z = e^{-\frac{U\beta}{4}}\left(1 + 2e^{\left(\frac{U}{2}+\mu\right)\beta} + e^{2\mu\beta}\right).$$

The operators $\mathcal{H}e^{-\beta\mathcal{H}}$, $n_\uparrow e^{-\beta\mathcal{H}}$, $n_\downarrow e^{-\beta\mathcal{H}}$ and $n_\uparrow n_\downarrow e^{-\beta\mathcal{H}}$ required for calculating physical observables $\mathcal{O}$ of interest become

$$\begin{aligned}
\mathcal{H}e^{-\beta\mathcal{H}} &\longrightarrow e^{-\frac{U\beta}{4}}\mathrm{diag}\Big(\frac{U}{4}, (-\mu-\frac{U}{4})e^{\beta(U/2+\mu)}, (-\mu-\frac{U}{4})e^{\beta(U/2+\mu)}, \\
&\qquad\qquad (\frac{U}{4}-2\mu)e^{2\mu\beta}\Big), \\
n_\uparrow e^{-\beta\mathcal{H}} &\longrightarrow e^{-\frac{U\beta}{4}}\mathrm{diag}\Big(0, e^{\beta(U/2+\mu)}, 0, e^{2\mu\beta}\Big), \\
n_\downarrow e^{-\beta\mathcal{H}} &\longrightarrow e^{-\frac{U\beta}{4}}\mathrm{diag}\Big(0, 0, e^{\beta(U/2+\mu)}, e^{2\mu\beta}\Big), \\
n_\uparrow n_\downarrow e^{-\beta\mathcal{H}} &\longrightarrow e^{-\frac{U\beta}{4}}\mathrm{diag}\big(0, 0, 0, e^{2\mu\beta}\big).
\end{aligned}$$

The traces of these operators are

$$\begin{aligned}
\mathrm{Tr}(\mathcal{H}e^{-\beta\mathcal{H}}) &= e^{-\frac{U\beta}{4}}\Big(\frac{U}{4} + 2(-\mu-\frac{U}{4})e^{\beta(U/2+\mu)} + (\frac{U}{4}-2\mu)e^{2\mu\beta}\Big), \\
\mathrm{Tr}((n_\uparrow + n_\downarrow)e^{-\beta\mathcal{H}}) &= e^{-\frac{U\beta}{4}}\Big(2e^{\beta(U/2+\mu)} + 2e^{2\mu\beta}\Big), \\
\mathrm{Tr}(n_\uparrow n_\downarrow e^{-\beta\mathcal{H}}) &= e^{-\frac{U\beta}{4}}e^{2\mu\beta}.
\end{aligned}$$

By the definition (1.2), the following physical observables $\mathcal{O}$ can be computed exactly:

1. The one-site density $\rho = \langle n_\uparrow\rangle + \langle n_\downarrow\rangle$ to measure the average occupation of each site:

$$\begin{aligned}
\rho = \langle n_\uparrow\rangle + \langle n_\downarrow\rangle &= \frac{\mathrm{Tr}\left((n_\uparrow + n_\downarrow)e^{-\beta\mathcal{H}}\right)}{\mathrm{Tr}(\mathcal{Z})} \\
&= \frac{2e^{\left(\frac{U}{2}+\mu\right)\beta} + 2e^{2\mu\beta}}{1 + 2e^{\left(\frac{U}{2}+\mu\right)\beta} + e^{2\mu\beta}}.
\end{aligned}$$

   When there is no chemical potential, i.e., $\mu = 0$, $\rho = 1$ for any U and β. It is referred to as "half-filling" because the density is one-half the maximal possible value.

2. The one-site energy $E = \langle\mathcal{H}\rangle$:

$$\begin{aligned}
E = \langle\mathcal{H}\rangle &= \frac{\mathrm{Tr}(\mathcal{H}e^{-\beta\mathcal{H}})}{\mathrm{Tr}(\mathcal{Z})} \\
&= \frac{U}{4} - \frac{(2\mu+U)e^{\left(\frac{U}{2}+\mu\right)\beta} + 2\mu e^{2\mu\beta}}{1 + 2e^{\left(\frac{U}{2}+\mu\right)\beta} + e^{2\mu\beta}}.
\end{aligned}$$

   When there is no chemical potential, i.e., $\mu = 0$,

$$E = \frac{U}{4} - \frac{U}{2(1 + e^{-\frac{U\beta}{2}})}.$$

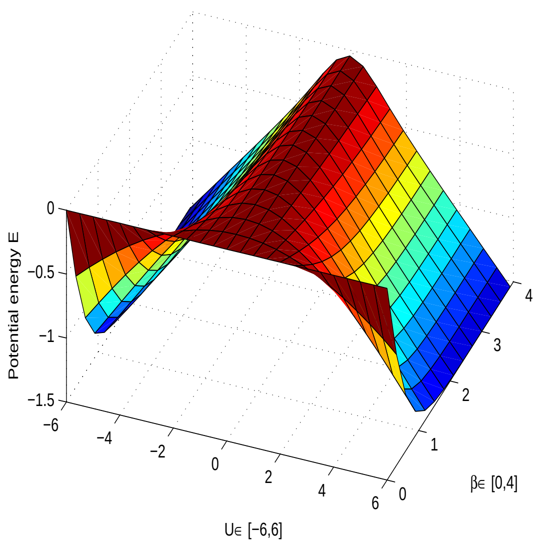

Figure 1.1: Potential energy E for $t = 0, \mu = 0$

Figure 1.1 shows the plot of E versus U and β.

3. The double occupancy $\langle n_\uparrow n_\downarrow \rangle$ is

$$\langle n_\uparrow n_\downarrow \rangle = \frac{\mathrm{Tr}(n_\uparrow n_\downarrow e^{-\beta\mathcal{H}})}{\mathrm{Tr}(\mathcal{Z})} = \frac{e^{2\mu\beta}}{1 + 2e^{(\frac{U}{2}+\mu)\beta} + e^{2\mu\beta}}.$$

When there is no chemical potential, i.e., $\mu = 0$,

$$\langle n_\uparrow n_\downarrow \rangle = \frac{1}{2(1 + e^{\frac{U}{2}\beta})}.$$

Note that as U or β increases, the double occupancy goes to zero.

1.1.2 Hubbard model without interaction

When there is no interaction, $U = 0$, the spin-up and spin-down spaces are independent. $\mathcal{H}$ breaks into the spin-up ($\uparrow$) and spin-down ($\downarrow$) terms. We can consider each spin space separately. Therefore, by omitting the spin, the Hamiltonian $\mathcal{H}$ becomes

$$\mathcal{H} = -t \sum_{\langle i,j \rangle} (c_i^\dagger c_j + c_j^\dagger c_i) - \mu \sum_i n_i.$$

It can be recast as a bilinear form:

$$\mathcal{H} = \vec{c}^{\dagger}(-tK - \mu I)\,\vec{c},$$

where

$$\vec{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix} \quad \text{and} \quad \vec{c}^{\dagger} = [c_1^{\dagger},\ c_2^{\dagger},\ \cdots,\ c_N^{\dagger}],$$

and I is the identity matrix, $K = (k_{ij})$ is a matrix to describe the hopping lattice geometry $\langle i, j\rangle$:

$$k_{ij} = \begin{cases} 1, \text{ if } i \text{ and } j \text{ are nearest neighbors} \\ 0, \text{ otherwise} \end{cases}.$$

For instance, for an one dimensional (1D) lattice of N_x sites, K is an $N_x \times N_x$ matrix given by

$$K = K_x = \begin{bmatrix} 0 & 1 & & & 1 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ 1 & & & 1 & 0 \end{bmatrix}.$$

The $(1, N_x)$ and $(N_x, 1)$ elements of K incorporate the so-called "periodic boundary conditions (PBCs)" in which sites 1 and N_x are connected by t. The use of PBC reduces finite size effects. For example, the energy on a finite lattice of length N with open boundary conditions (OBCs) differs from the value in the thermodynamic limit ($N \to \infty$) by a correction of order $1/N$ while with PBCs, the correction is order $1/N^2$. [1] The use of PBCs also makes the system translationally invariant. The density of electrons per site, and other similar quantities, will not depend on the site in question. With OBCs quantities will vary with the distance from the edges of the lattice.

For a two dimensional (2D) rectangle lattice of $N_x \times N_y$ sites, K is an $N_xN_y \times N_xN_y$ matrix given by

$$K = K_{xy} = I_y \otimes K_x + K_y \otimes I_x,$$

where I_x and I_y are identity matrices with dimensions N_x and N_y, respectively. $\otimes$ is the matrix Kronecker product.

[1] A simple analogy is this: Consider numerical integration of $f(x)$ on an interval $a \leq x \leq b$. The only difference between the rectangle and trapezoidal rules is in their treatment of the boundary point contributions $f(a)$ and $f(b)$, yet the integration error changes from linear in the mesh size to quadratic.

The matrix K of 1D or 2D lattice has the exact eigen-decomposition

$$K = F^T \Lambda F, \quad F^T F = I,$$

where $\Lambda = \mathrm{diag}(\lambda_k)$ is a diagonal eigenvalue matrix, see Lemma 2.1. Let

$$\vec{\tilde{c}} = F\vec{c} \quad \text{and} \quad \vec{\tilde{c}}^{\dagger} = (F\vec{c})^{\dagger},$$

then the Hamiltonian $\mathcal{H}$ is diagonalized:

$$\mathcal{H} = \vec{\tilde{c}}^{\dagger}(-t\Lambda - \mu I)\vec{\tilde{c}} = \sum_k \epsilon_k \tilde{n}_k,$$

where $\epsilon_k \equiv -t\lambda_k - \mu$, and $\tilde{n}_k = \tilde{c}_k^{\dagger}\tilde{c}_k$.

It can be shown that the operators $\tilde{c}_k$ obey the same anticommutation relations as the original operators c_i. Hence they too appropriately describe electrons. Indeed, the original operators create and destroy particles on particular spatial sites i while the new ones create and destroy with particular momenta k. Either set is appropriate to use, however, the interaction term in the Hubbard model is fairly complex when written in momentum space.

Lemma 1.1. If the operator $\mathcal{H}$ is in a quadratic form of fermion operators

$$\mathcal{H} = c^{\dagger} H c,$$

where H is an $N \times N$ Hermitian matrix. Then

$$\mathrm{Tr}(e^{-\beta\mathcal{H}}) = \prod_{i=1}^{N}(1 + e^{-\beta\lambda_{k_i}}),$$

where λ_{k_i} are the eigenvalues of H.

Proof. First let us assume that $H = \mathrm{diag}(\lambda_{k_1}, \lambda_{k_2}, \cdots, \lambda_{k_N})$, then the Hamiltonian $\mathcal{H}$ is

$$\mathcal{H} = c^{\dagger} H c = c^{\dagger}\mathrm{diag}(\lambda_{k_1}, \lambda_{k_2}, \cdots, \lambda_{k_N})c = \sum_{i=1}^{N} \lambda_{k_i} n_{k_i}.$$

The lemma can be proved by induction. When $N = 1$, for the two eigenstates $|0\rangle$ and $|1\rangle$ of the number operator n_{k_1}, we have

$$\mathrm{Tr}(e^{-\beta\mathcal{H}}) = \langle 0|e^{-\beta\lambda_{k_1} n_{k_1}}|0\rangle + \langle 1|e^{-\beta\lambda_{k_1} n_{k_1}}|1\rangle = e^0 + e^{-\beta\lambda_{k_1}}.$$

Assume that for $N - 1$, we have

$$\mathrm{Tr}(e^{-\beta\sum_{i=1}^{N-1}\lambda_{k_i} n_{k_i}}) = \prod_{i=1}^{N-1}(1 + e^{-\beta\lambda_{k_i}}). \tag{1.6}$$

Then, by the definition of the trace, we have

$$\begin{aligned}
&\mathrm{Tr}(e^{-\beta\sum_{i=1}^{N}\lambda_{k_i}n_{k_i}}) \\
&= \sum_{k_1,\cdots,k_N} \langle \psi_1^{k_1}\cdots\psi_N^{k_N}|e^{-\beta\sum_{i=1}^{N}\lambda_{k_i}n_{k_i}}|\psi_1^{k_1}\cdots\psi_N^{k_N}\rangle. \qquad (1.7)\\
&= \sum_{k_1,\cdots,k_{N-1}} \Big\{\langle \psi_1^{k_1}\cdots\psi_{N-1}^{k_{N-1}}0|e^{-\beta\sum_{i=1}^{N-1}\lambda_{k_i}n_{k_i}}e^{-\beta\lambda_{k_N}n_{k_N}}|\psi_1^{k_1}\cdots\psi_{N-1}^{k_{N-1}}0\rangle + \\
&\qquad \langle \psi_1^{k_1}\cdots\psi_{N-1}^{k_{N-1}}1|e^{-\beta\sum_{i=1}^{N-1}\lambda_{k_i}n_{k_i}}e^{-\beta\lambda_{k_N}n_{k_N}}|\psi_1^{k_1}\cdots\psi_{N-1}^{k_{N-1}}1\rangle\Big\}\\
&= (1+e^{-\beta\lambda_{k_N}})\sum_{k_1,\cdots,k_{N-1}}\langle \psi_1^{k_1}\cdots\psi_{N-1}^{k_{N-1}}|e^{-\beta\sum_{i=1}^{N-1}\lambda_{k_i}n_{k_i}}|\psi_1^{k_1}\cdots\psi_{N-1}^{k_{N-1}}\rangle\\
&= (1+e^{-\beta\lambda_{k_N}})\prod_{i=1}^{N-1}(1+e^{-\beta\lambda_{k_i}}) = \prod_{i=1}^{N}(1+e^{-\beta\lambda_{k_i}}). \qquad (1.8)
\end{aligned}$$

For a general Hermitian matrix H, there exists a unitary matrix Q such that

$$Q^*HQ = \Lambda = \mathrm{diag}(\lambda_{k_1}, \lambda_{k_2}, \cdots, \lambda_{k_N}).$$

Let $\tilde{c} = Qc$ and $\tilde{n}_i = \tilde{c}_i^\dagger \tilde{c}_i$, then we have

$$\mathcal{H} = c^\dagger H c = \tilde{c}^\dagger \Lambda \tilde{c} = \sum_{i=1}^{N} \lambda_{k_i}\tilde{n}_{k_i}.$$

Since the trace is independent of the basis functions and by the equation (1.8), we have

$$\mathrm{Tr}(e^{-\beta\mathcal{H}}) = \mathrm{Tr}\left(\prod_{i=1}^{N} e^{-\beta\lambda_{k_i}\tilde{n}_{k_i}}\right) = \prod_{i=1}^{N}(1+e^{-\beta\lambda_{k_i}}).$$

The lemma is proved. □

By Lemma 1.1, the partition function $\mathcal{Z}$ is given by

$$Z = \prod_k (1+e^{-\beta\epsilon_k}).$$

Subsequently, we have the exact expressions for the following physical observables $\mathcal{O}$:

1. the density ρ, the average occupation of each site:

$$\rho = \langle n\rangle = \langle \tilde{n}\rangle = \frac{1}{N}\sum_{k=1}^{N}\langle \tilde{n}_k\rangle = \frac{1}{N}\sum_k \frac{1}{1+e^{\beta\epsilon_k}},$$

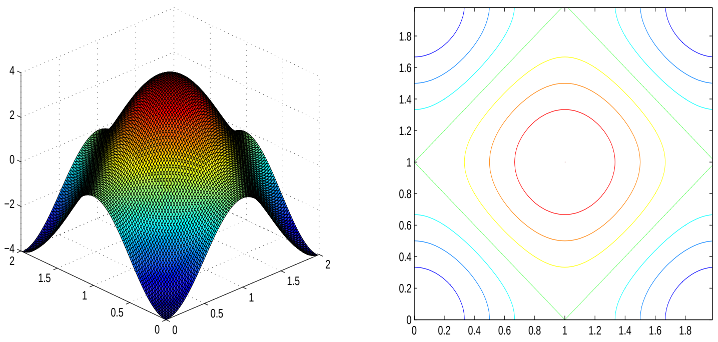

Figure 1.2: ϵ_k for $U = 0$ and $\mu = 0$ (left). Contour plot of ϵ_k (right)

2. the energy E:

$$E = \langle \mathcal{H} \rangle = \frac{1}{N} \sum_k \frac{\epsilon_k}{e^{\beta\epsilon_k} + 1}. \tag{1.9}$$

For the sake of completeness, let us write down the expression for the Green's function, which plays a key role for computing the physical observables:

$$G_{\ell,j} = \langle c_\ell c_j^\dagger \rangle = \frac{1}{N} \sum_k e^{ik\cdot(j-\ell)}(1 - f_k), \tag{1.10}$$

where $f_k = 1/[1+e^{\beta(\epsilon_k-\mu)}]$. Notice that G is a function of the difference $j-\ell$. This is a consequence of the fact that the Hamiltonian is translationally invariant, that is, with PBCs, there is no special site which is singled out as the origin of the lattice. All sites are equivalent.

At $T = 0$ ($\beta = \infty$), the contours in the right side of Figure 1.2 separate the k values where the states are occupied $f_k = 1$ (inside the contour) from those where the states are empty $f_k = 0$ (outside the contour). The contour is often referred to as the "Fermi surface". There is actually a lot of interesting physics which follows from the geometry of the contour plot of ϵ_k. For example, one notes that the vector (π, π) connects large regions of the contour in the case when $\rho = 1$ and the contour is the rotated square connecting the points $(\pi, 0)$, $(0, \pi)$, $(-\pi, 0)$, and $(0, -\pi)$. One refers to this phenomenon as "nesting" and to (π, π) as the "nesting wave vector". Because the Fermi surface describes the location where the occupation changes from 0 to 1, the electrons are most active there. If there is a wave vector k which connects big expanses of these active regions, special order is likely to occur with that wave vector. Thus the contour plot of ϵ_k is one way of understanding the tendency of the Hubbard model to have *antiferromagnetic* order (magnetic order at $k = (\pi, \pi)$ for half-filling.)

1.2 Determinant QMC

In this section, we first introduce a computable approximation of the distribution operator $\mathcal{P}$ defined in (1.3) by using a discrete Hubbard-Stratonovich transformation, and then a so-called determinant QMC (DQMC) algorithm to generate samples that follow the distribution. For simplicity, we assume that the chemical potential $\mu = 0$ which corresponds to the important half-filled-band case (and there is no the "sign problem"). It turns out that many of the most interesting phenomena of the Hubbard model, like magnetic ordering and insulating-metal transition, occur at half-filling.

1.2.1 Computable approximation of distribution operator $\mathcal{P}$

The gist of a computable approximation of the distribution operator $\mathcal{P}$ defined in (1.3) is on the approximation of the partition function $\mathcal{Z}$. Since the operators $\mathcal{H}_K$ and $\mathcal{H}_V$ do not commute, we apply the Trotter-Suzuki decomposition to approximate $\mathcal{Z}$. Specifically, by dividing the imaginary-time interval $[0, \beta]$ into L equal subintervals of the width $\Delta\tau = \frac{\beta}{L}$, then $\mathcal{Z}$ can be written as

$$\begin{aligned}\mathcal{Z} &= \operatorname{Tr}\left(e^{-\beta\mathcal{H}}\right)\\ &= \operatorname{Tr}\left(\prod_{\ell=1}^{L} e^{-\Delta\tau\mathcal{H}}\right)\\ &= \operatorname{Tr}\left(\prod_{\ell=1}^{L} e^{-\Delta\tau\mathcal{H}_K} e^{-\Delta\tau\mathcal{H}_V}\right) + O(\Delta\tau^2). \qquad (1.11)\end{aligned}$$

The kinetic energy term $e^{-\Delta\tau\mathcal{H}_K}$ is quadratic in the fermion operators and the spin-up and spin-down operators are independent. Therefore, it can be written as

$$e^{-\Delta\tau\mathcal{H}_K} = e^{-\Delta\tau\mathcal{H}_{K_+}} e^{-\Delta\tau\mathcal{H}_{K_-}},$$

where the operators $\mathcal{H}_{K_+}$ and $\mathcal{H}_{K_-}$ correspond to kinetic energy with spin-up and spin-down respectively, and are of the forms

$$\mathcal{H}_{K_\sigma} = -t\, \vec{c}_\sigma^{\,\dagger} K \vec{c}_\sigma.$$

On the other hand, the potential energy term $e^{-\Delta\tau\mathcal{H}_V}$ is quartic in the fermion operators. It is necessary to recast it in a quadratic form to use something like Lemma 1.1. To do so, first, note that the number operators $n_{i\sigma}$ are independent on different sites, we have

$$\begin{aligned} e^{-\Delta\tau\mathcal{H}_V} &= e^{-U\Delta\tau\sum_{i=1}^{N}(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})} \\ &= \prod_{i=1}^{N} e^{-U\Delta\tau(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})}. \end{aligned}$$

To treat the term $e^{-U\Delta\tau(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})}$, we use the following discrete Hubbard-Stratonovich transformation. It replaces the interaction quartic term $(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})$ by the quadratic one $(n_{i+}-n_{i-})$.

Lemma 1.2. (Discrete Hubbard-Stratonovich transformation [2, 5]) If $U > 0$, then

$$e^{-U\Delta\tau(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})} = C_1 \sum_{h_i=\pm 1} e^{\nu h_i(n_{i+}-n_{i-})}, \tag{1.12}$$

where the constant $C_1 = \frac{1}{2}e^{-\frac{U\Delta\tau}{4}}$ and the scalar ν is defined by $\cosh\nu = e^{\frac{U\Delta\tau}{2}}$.

Proof. First, the following table lists the results of actions of the operators $(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})$ and $(n_{i+}-n_{i-})$ on the four possible eigenstates $|\cdot\rangle$, $|\uparrow\rangle$, $|\downarrow\rangle$ and $|\uparrow\downarrow\rangle$:

ψ	$(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})$	$n_{i+}-n_{i-}$
$\vert\cdot\rangle$	$\frac{1}{4}\vert\cdot\rangle$	$0\,\vert\cdot\rangle$
$\vert\uparrow\rangle$	$-\frac{1}{4}\vert\uparrow\rangle$	$\vert\uparrow\rangle$
$\vert\downarrow\rangle$	$-\frac{1}{4}\vert\downarrow\rangle$	$\vert\downarrow\rangle$
$\vert\uparrow\downarrow\rangle$	$\frac{1}{4}\vert\uparrow\downarrow\rangle$	$0\,\vert\uparrow\downarrow\rangle$

For the operator of the left-hand side of (1.12):

$$e^{-U\Delta\tau(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})}\psi = e^{-\frac{U\Delta\tau}{4}}\psi \quad \text{if} \quad \psi = |\cdot\rangle \text{ or } |\uparrow\downarrow\rangle,$$

and

$$e^{-U\Delta\tau(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})}\psi = e^{\frac{U\Delta\tau}{4}}\psi \quad \text{if} \quad \psi = |\uparrow\rangle \text{ or } |\downarrow\rangle.$$

On the other hand, for the operator of the left-hand side of (1.12):

$$C_1 \sum_{h_i=\pm 1} e^{\nu h_i(n_{i+}-n_{i-})}\psi = e^{-\frac{U\Delta\tau}{4}}\psi \quad \text{if} \quad \psi = |\cdot\rangle \text{ or } |\uparrow\downarrow\rangle,$$

and

$$C_1 \sum_{h_i=\pm 1} e^{\nu h_i(n_{i+}-n_{i-})}\psi = \frac{1}{2}e^{-\frac{U\Delta\tau}{4}}(e^{\nu}+e^{-\nu})\psi \quad \text{if} \quad \psi = |\uparrow\rangle \text{ or } |\downarrow\rangle.$$

Since

$$\cosh \nu = \frac{e^{\nu} + e^{-\nu}}{2} = e^{\frac{U\Delta\tau}{2}},$$

the discrete Hubbard-Stratonovich transformation (1.12) holds. □

Remark 1.5. Note that in the previous proof, U is required to be positive, otherwise there is no real number ν such that $\cosh \nu = e^{\frac{U\Delta\tau}{2}}$. When $U < 0$, the Hubbard model is called the attractive Hubbard model. A similar discrete Hubbard-Stratonovich transformation exists [6, 7]. Other transformations for treating the quartic term can also be founded in [13, 17].

Let us continue to reformulate the term $e^{-\Delta\tau\mathcal{H}_V}$. By the discrete Hubbard-Stratonovich transformation (1.12), we have

$$e^{-\Delta\tau\mathcal{H}_V} = \prod_{i=1}^{N} \left(C_1 \sum_{h_i=\pm 1} e^{\nu h_i (n_{i+} - n_{i-})} \right). \tag{1.13}$$

$\{h_i\}$ are referred to as auxiliary variables. The collection of all these variables is called the Hubbard-Stratonovich field or configurations.

For the sake of simplicity, let us consider the case $N = 2$ of the expression (1.13):

$$\begin{aligned} e^{-\Delta\tau\mathcal{H}_V} &= (C_1)^2 \left(\sum_{h_i=\pm 1} e^{\nu h_i (n_{1+} - n_{1-})} \right) \left(\sum_{h_i=\pm 1} e^{\nu h_i (n_{2+} - n_{2-})} \right) \\ &= (C_1)^2 \sum_{h_i=\pm 1} e^{\sum_{i=1}^{2} \nu h_i (n_{i+} - n_{i-})} \\ &\equiv (C_1)^2 \mathrm{Tr}_h e^{\sum_{i=1}^{2} \nu h_i (n_{i+} - n_{i-})}, \end{aligned}$$

where the new notation Tr_h represents the trace for $h_i = \pm 1$.

In general, we have

$$\begin{aligned} e^{-\Delta\tau\mathcal{H}_V} &= (C_1)^N \mathrm{Tr}_h e^{\sum_{i=1}^{N} \nu h_i (n_{i+} - n_{i-})} \\ &= (C_1)^N \mathrm{Tr}_h \left(e^{\sum_{i=1}^{N} \nu h_i n_{i+}} e^{-\sum_{i=1}^{N} \nu h_i n_{i-}} \right) \\ &\equiv (C_1)^N \mathrm{Tr}_h (e^{\mathcal{H}_{V_+}} e^{\mathcal{H}_{V_-}}), \end{aligned} \tag{1.14}$$

where $\mathcal{H}_{V_+}$ and $\mathcal{H}_{V_-}$ correspond to spin-up and spin-down, respectively, and are of the forms

$$\mathcal{H}_{V_\sigma} = \sum_{i=1}^{N} \nu h_i n_{i\sigma} = \sigma\nu\, \vec{c}_\sigma^{\dagger} V(h) \vec{c}_\sigma,$$

and $V(h)$ is a diagonal matrix $V(h) = \mathrm{diag}(\, h_1, h_2, \ldots, h_N \,)$.

Taking into account the partition of the inverse temperature β into L imaginary time slices, $L = \beta/\Delta\tau$, the Hubbard-Stratonovich variables h_i are changed to have two subindices $h_{\ell,i}$, where i is for the spatial site and ℓ is for the imaginary time slice. Correspondingly, the index ℓ is also introduced for the diagonal matrix V and operators $\mathcal{H}_{V_\sigma}$:

$$h_i \longrightarrow h_{\ell,i}, \qquad V \longrightarrow V_\ell, \qquad \mathcal{H}_{V_\sigma} \longrightarrow \mathcal{H}^{\ell}_{V_\sigma}.$$

Subsequently, by applying the Trotter–Suzuki approximation (1.11) and the expression (1.14) and interchanging the traces, the partition function $\mathcal{Z}$ can be approximated by

$$\mathcal{Z} = (C_1)^{NL}\mathrm{Tr}_h \mathrm{Tr}\left(\prod_{\ell=1}^{L} e^{-\Delta\tau\mathcal{H}_{K_+}} e^{\mathcal{H}^{\ell}_{V_+}}\right)\left(\prod_{\ell=1}^{L} e^{-\Delta\tau\mathcal{H}_{K_-}} e^{\mathcal{H}^{\ell}_{V_-}}\right), \quad (1.15)$$

where for $\sigma = \pm$,

$$\begin{aligned}\mathcal{H}_{K_\sigma} &= -t\,\vec{c}_\sigma^{\,\dagger} K \vec{c}_\sigma,\\ \mathcal{H}^{\ell}_{V_\sigma} &= \sigma \sum_{i=1}^{N} \nu h_{\ell,i} n_{i+} = \sigma\nu \vec{c}_\sigma^{\,\dagger} V_\ell(h_\ell)\vec{c}_\sigma\end{aligned}$$

and $V_\ell(h_\ell)$ is a diagonal matrix

$$V_\ell(h_\ell) = \mathrm{diag}(\, h_{\ell,1}, h_{\ell,2}, \ldots, h_{\ell,N}\,).$$

At this point, all operators $\mathcal{H}_{K_+}$, $\mathcal{H}_{K_-}$, $\mathcal{H}^{\ell}_{V+}$ and $\mathcal{H}^{\ell}_{V-}$ are quadratic in the fermion operators. We can apply the following lemma presented in [2, 5][2]:

Lemma 1.3. If operators $\mathcal{H}_\ell$ are in the quadratic forms of the fermion operators

$$\mathcal{H}_\ell = \sum_{i,j} c_i^\dagger (H_\ell)_{ij} c_j,$$

where H_ℓ are matrices of real numbers. Then

$$\mathrm{Tr}(e^{-\mathcal{H}_1} e^{-\mathcal{H}_2} \cdots e^{-\mathcal{H}_L}) = \det(I + e^{-H_L} e^{-H_{L-1}} \cdots e^{-H_1}). \quad (1.16)$$

Note that while "Tr" is over the quantum mechanical Hilbert space whose dimension is 4^N, since by Pauli exclusion principle, there are 4 possible states in every lattice: no electron, one electron with spin-up, one electron with spin-down and two electron with different spin. The "det" is the determinant of a matrix.

[2]We are unable to provide a rigorous proof for this important identity. The special case $L = 1$ is Lemma 1.1.

By using the identity (1.16), the partition function $\mathcal{Z}$ described in (1.15) is turned into the following computable form

$$Z_h = (C_1)^{NL} \mathrm{Tr}_h \det[M_+(h)] \det[M_-(h)], \tag{1.17}$$

where for $\sigma = \pm$ and $h = (h_1, h_2, \ldots, h_L)$, the fermion matrices

$$M_\sigma(h) = I + B_{L,\sigma}(h_L) B_{L-1,\sigma}(h_{L-1}) \cdots B_{1,\sigma}(h_1), \tag{1.18}$$

and matrices $B_{\ell,\sigma}(h_\ell)$ are associated with the operators $e^{-\Delta\tau\mathcal{H}_K} e^{-\Delta\tau\mathcal{H}^{\ell}_{V_\sigma}}$, and are defined as

$$B_{\ell,\sigma}(h_\ell) = e^{t\Delta\tau K} e^{\sigma\nu V_\ell(h_\ell)}.$$

By the expression (1.17), we have a computable approximation of the distribution operator $\mathcal{P}$ defined in (1.3):

$$P(h) = \frac{\eta_d}{Z_h} \det[M_+(h)] \det[M_-(h)], \tag{1.19}$$

where $\eta_d = (C_1)^{NL}$ is a normalizing constant.

Remark 1.6. When $U = 0$, $\nu = 0$ and $M_\sigma(h)$ is a constant matrix and does not depend on the configuration h. The Trotter-Suzuki approximation is exact. The Hubbard Hamiltonian is computed exactly after a single evaluation of the matrix $M_\sigma(h)$.

Remark 1.7. It is a rather amazing thing that a quantum problem can be re-written as a classical one. The price for this is that the classical problem is in one higher dimension than the original quantum one: the degrees of freedom in the quantum problem c_i had a single spatial index i while the Hubbard Stratonovich variables which replace them have an additional imaginary time index ℓ. This mapping is by no means restricted to the Hubbard Hamiltonian, but is generally true for all quantum mechanics problems.

1.2.2 Algorithm

Our computational task now becomes a classical Monte Carlo problem: sample Hubbard-Stratonovich variables (configurations) h that follow the probability distribution function $P(h)$ defined in (1.19). Recall that the dimension of the configuration sample space $\{h\}$ is 2^{NL}. For an efficient Monte Carlo procedure there are two essential questions:

1. How to move to a new configuration h' from the current h?
   A simple strategy is to flip only at one selected site (ℓ, i)

   $$h'_{\ell,i} = -h_{\ell,i},$$

   and leave the rest components of h unchanged.

2. How to ensure that the accepted sample configuration h follows the desired distribution $P(h)$?
   This is answered by the Metropolis-Hasting algorithm, for example, see [10, p.111].

Combining the answers of these two questions, we have the following so-called determinant QMC (DQMC) algorithm, first presented in [2].

```
DQMC
• Initialize $h = (h_{\ell,i}) = (\pm 1)$
• MC loop (total steps = warm up + measurement)
  1. set $(\ell, i) = (1, 1)$
  2. $(\ell, i)$–loop:
     (a) propose a new configuration $h'$ by flipping at
         the site $(\ell, i)$: $h'_{\ell,i} = -h_{\ell,i}$
     (b) compute the Metropolis ratio
         $$r_{\ell,i} = \frac{\det[M_+(h')]\det[M_-(h')]}{\det[M_+(h)]\det[M_-(h)]}$$
     (c) Metropolis acceptance-rejection:
         $$h = \begin{cases} h', \text{ if } r \le \min\{1, r_{\ell,i}\} \\ h, \text{ otherwise.} \end{cases}$$
         where $r$ is a random number and $r \sim \text{Uniform}[0, 1]$.
     (d) go to the next site $(\ell, i)$, where
         – if $i < N$, then $\ell := \ell$, $i := i + 1$,
         – if $i = N$ and $\ell < L$, then $\ell := \ell + 1, i = 1$,
         – if $i = N$ and $\ell = L$, then $\ell := 1, i = 1$
  3. after the warm up steps, perform physical measure-
     ments, see section 1.2.3
```

Note that the one-site update at the inner loop leads to a simple rank-one updating of the matrix $M_\sigma(h)$. Based on this observation, one can efficiently compute the Matropolis ratio $r_{\ell,i}$, see Appendix A for detail.

1.2.3 Physical measurements

How is the physics extracted from QMC simulations? Two of the most elementary physical observables of interest are the *density* and *kinetic* *energy*, which can be obtained from the single-particle Green's function,

$$\begin{aligned}G^{\sigma}_{ij} &= \langle c_{i\sigma} c^{\dagger}_{j\sigma} \rangle \\ &= \left(M^{-1}_{\sigma}(h)\right)_{ij} \\ &= \left([I + B_{L,\sigma}(h_L) B_{L-1,\sigma}(h_{L-1}) \cdots B_{1,\sigma}(h_1)]^{-1}\right)_{ij}.\end{aligned}$$

The density of electrons of spin σ on site i is

$$\rho_{i,\sigma} = \langle n_{i,\sigma} \rangle = \langle c^{\dagger}_{i,\sigma} c_{i,\sigma} \rangle = 1 - \langle c_{i,\sigma} c^{\dagger}_{i,\sigma} \rangle = 1 - G^{\sigma}_{ii},$$

where the third identity arises from the use of the anticommutation relations in interchanging the order of creation and destruction operators.

The Hubbard Hamiltonian is translationally invariant, so one expects $\rho_{i,\sigma}$ to be independent of spatial site i. Likewise, up and down spin species are equivalent. Thus to reduce the statistical errors, one usually averages all the values and have the code report back

$$\rho = \frac{1}{2N} \sum_{\sigma} \sum_{i=1}^{N} \rho_{i,\sigma}.$$

We will not emphasize this point further, but such averaging is useful for most observables. [3] As is true in any classical or quantum Monte Carlo, these expectation values are also averaged over the sequence of Hubbard-Stratonovich configurations generated in the simulation.

The kinetic energy is obtained from the Green's function for pairs of sites i, j which are near neighbors,

$$\begin{aligned}\langle \mathcal{H}_K \rangle &= -t \, \langle \sum_{\langle i,j \rangle, \sigma} (c^{\dagger}_{i\sigma} c_{j\sigma} + c^{\dagger}_{j\sigma} c_{i\sigma}) \rangle \\ &= +t \sum_{\langle i,j \rangle, \sigma} (G^{\sigma}_{ij} + G^{\sigma}_{ji}).\end{aligned}$$

An extra minus sign arose from interchanging the fermion operator order so that the creation operator was at the right.

Extended physical measurements. Interesting types of magnetic, charge, and superconducting order, and associated phase transitions, are determined by looking at correlation functions of the form:

$$c(j) = \langle \mathcal{O}_{i+j} \mathcal{O}^{\dagger}_{i} \rangle - \langle \mathcal{O}_{i+j} \rangle \, \langle \mathcal{O}^{\dagger}_{i} \rangle \qquad (1.20)$$

where, for example,

[3] There are some situations where translation invariance is broken, for example if randomness is included in the Hamiltonian.

- for spin order in z direction (magnetism):

$$\mathcal{O}_i = n_{i,\uparrow} - n_{i,\downarrow}, \quad \mathcal{O}_i^\dagger = n_{i,\uparrow} - n_{i,\downarrow};$$

- for spin order in x/y direction (magnetism):

$$\mathcal{O}_i = c_{i,\downarrow}^\dagger c_{i,\uparrow}, \quad \mathcal{O}_i^\dagger = c_{i,\uparrow}^\dagger c_{i,\downarrow};$$

- for charge order:

$$\mathcal{O}_i = n_{i,\uparrow} + n_{i,\downarrow}, \quad \mathcal{O}_i^\dagger = n_{i,\uparrow} + n_{i,\downarrow};$$

- for pair order (superconductivity):

$$\mathcal{O}_i = c_{i,\downarrow} c_{i,\uparrow}, \quad \mathcal{O}_i^\dagger = c_{i,\uparrow}^\dagger c_{i,\downarrow}^\dagger.$$

In words, what such correlation functions probe is the relationship between spin, charge, pairing on an initial site i with that on a site $i+j$ separated by a distance j. It is plausible that at high temperatures where there is a lot of random thermal vibration, the values of $\mathcal{O}_i^\dagger$ and $\mathcal{O}_{i+j}$ will not 'know about' each other for large j. In such a case, the expectation value $\langle \mathcal{O}_{i+j}\mathcal{O}_i^\dagger \rangle$ factorizes to $\langle \mathcal{O}_{i+j} \rangle \, \langle \mathcal{O}_i^\dagger \rangle$ and $c(j)$ vanishes. The more precise statement is that at high temperatures $c(j)$ decays exponentially, $c(j) \propto e^{-l/\xi}$. The quantity ξ is called the "correlation length". On the other hand, at low temperatures $c(j) \propto \alpha^2$, a nonzero value, as $j \to \infty$. The quantity α is called the 'order parameter'. [4]

As one can well imagine from this description, one needs to be very careful in analyzing data on finite lattices if the $j \to \infty$ behavior is what is crucial to determining the physics. The techniques of 'finite size scaling' provide methods for accomplishing this.

How are these correlation functions actually evaluated? As commented above in describing measurements of the density and kinetic energy, expectation values of two fermion operators are simply expressed in terms of the Green's function. The general rule for expectation values of more than two fermion creation and destruction operators is that they reduce to products of expectation values of pairs of creation and destruction operators, the famous "Wick's Theorem" of many body physics. For example, for spin order in the x/y direction,

$$\langle c(j) \rangle = \langle c_{i+j,\downarrow}^\dagger c_{i+j,\uparrow} c_{i,\uparrow}^\dagger c_{i,\downarrow} \rangle = G_{i+j,i}^{\uparrow} \, G_{i,i+j}^{\downarrow}.$$

[4] It is interesting to note what happens right at the critical point T_c separating the high temperature disordered phase from the low temperature ordered one. In what are termed 'second order' phase transitions, the correlation length diverges $\xi \propto 1/(T-T_c)^\nu$. Right at T_c the correlation function decays as a power law, $c(j) \propto 1/j^\eta$, a behavior intermediate between its high temperature exponential decay and its low temperature nonzero value.

Similarly, for superconducting order,

$$\langle c(j) \rangle = \langle c_{i+j,\downarrow} c_{i+j,\uparrow} c^{\dagger}_{i,\uparrow} c^{\dagger}_{i,\downarrow} \rangle = G^{\uparrow}_{i+j,i} \, G^{\downarrow}_{i+j,i}.$$

We conclude with two comments. First, it is useful to look at correlation functions where the operators $\mathcal{O}_{i+j}$ and $\mathcal{O}^{\dagger}_{i}$ are separated in imaginary time as well as in space. We will come to this point when we discuss measurements in the hybrid QMC algorithm. Second, one often considers the Fourier transform of the real space correlation function $S(q) = \sum_j e^{iqj} \, c(j)$. This quantity is often referred to as the 'structure factor', and is important because it is in fact the quantity measured by experimentalists. For example, the scattering rate of a neutron off the electron spins in a crystal is proportional to $S(q)$ where q is the change in momentum of the neutron and the $c(l)$ under consideration is the spin correlation function.

1.3 Hybrid QMC

The procedure summarized in section 1.2.2 is the one used in most DQMC codes today. Many interesting physical results have been obtained with it. However, it has a crucial limitation. At the heart of the procedure is the need to compute the ratio of determinants of matrices which have a dimension N, the spatial size of the system. Thus the algorithm scales as N^3. In practice, this means simulations are limited to a few hundred sites. In order to circumvent this bottleneck and develop an algorithm which potentially scales linearly with N, we reformulate our problem as the following:

1. Replace the discrete Hubbard-Stratonovich field by a continuous one;
2. Express the determinant of the dense $N \times N$ matrices $M_\sigma(h)$ as Gaussian integrals over NL-dimensional space to lead a $NL \times NL$ sparse matrix calculations.

We shall now describe these steps in detail.

1.3.1 Computable approximation of distribution operator $\mathcal{P}$

Instead of using discrete Hubbard-Stratonovich transformation as described in section 1.2.1, one can use a continuous Hubbard-Stratonovich transformation to derive a computable approximation of the distribution operator $\mathcal{P}$. First, recall the identity[5]:

$$e^{\frac{1}{2}a^2} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}z^2 - za} dz \qquad (1.21)$$

for any scalar $a > 0$.

Lemma 1.4. (Continuous Hubbard-Stratonovich transformation) For $U > 0$, we have

$$e^{-U\Delta\tau(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})} = C_2 \int_{-\infty}^{\infty} e^{-\Delta\tau[x^2+(2U)^{\frac{1}{2}}x(n_{i+}-n_{i-})]} dx, \qquad (1.22)$$

where $C_2 = (\Delta\tau e^{-\frac{U\Delta\tau}{2}}/\pi)^{\frac{1}{2}}$.

Proof. It is easy to verify that

$$(n_{i+} - \frac{1}{2})(n_{i-} - \frac{1}{2}) = -\frac{1}{2}(n_{i+} - n_{i-})^2 + \frac{1}{4}.$$

Note that $(n_{i+} - n_{i-})^2$ and $(n_{i+} - n_{i-})$ can be diagonalized based on the eigen-states of the operators $n_{i\sigma}$, then the identity (1.21) holds if we replace the scalar α by the operator $(n_{i+} - n_{i-})$:

$$e^{\frac{U\Delta\tau}{2}(n_{i+}-n_{i-})^2} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}x^2 - (U\Delta\tau)^{\frac{1}{2}}(n_{i+}-n_{i-})x} dx.$$

Let $x' = \frac{x}{\sqrt{2\Delta\tau}}$, we have

$$e^{\frac{U\Delta\tau}{2}(n_{i+}-n_{i-})^2} = \frac{\sqrt{\Delta\tau}}{\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-\Delta\tau(x^2+(2U)^{\frac{1}{2}}(n_{i+}-n_{i-})x)} dx.$$

Combining the above equations, we obtain the identity (1.22). $\Box$

Returning to the approximation of the partition function $\mathcal{Z}$ by the Trotter-Suzuki decomposition (1.11), by the continuous Hubbard-Stratonovich

[5]Note that the identity can be easily verified by using the following well-known identity

$$\int_{-\infty}^{\infty} e^{-z^2} dz = \sqrt{\pi}.$$

In fact, we have

$$\begin{aligned}\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}z^2 - za} dz &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}(z^2+2za+a^2-a^2)} dz \\ &= e^{\frac{1}{2}a^2} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}(z+a)^2} dz \\ &= e^{\frac{1}{2}a^2}.\end{aligned}$$

identity (1.22), we have

$$\begin{aligned} e^{-\Delta\tau\mathcal{H}_V^\ell} &= (C_2)^N \int_{-\infty}^{+\infty} e^{-\Delta\tau\sum_i x_{\ell,i}^2} e^{\Delta\tau\sum_i (2U)^{\frac{1}{2}} x_{\ell,i} n_{i+}} e^{-\Delta\tau\sum_i (2U)^{\frac{1}{2}} x_{\ell,i} n_{i-}} dx_{\ell,i} \\ &\equiv (C_2)^N \int [\delta x] e^{-S_B(x)} e^{\Delta\tau\mathcal{H}_{V+}^\ell} e^{\Delta\tau\mathcal{H}_{V-}^\ell}, \end{aligned}$$

where

$$\begin{aligned} S_B(x) &= \Delta\tau \sum_{\ell,i} x_{\ell,i}^2, \\ \mathcal{H}_{V_\sigma}^\ell &= \sum_i (2U)^{\frac{1}{2}} x_{\ell,i} n_{i\sigma} = \sigma(2U)^{\frac{1}{2}} \vec{c}_\sigma^{\,\dagger} V_\ell(x_\ell)\vec{c}_\sigma, \end{aligned}$$

and $V_\ell(x_\ell)$ is a diagonal matrix,

$$V_\ell(x_\ell) = \operatorname{diag}(x_{\ell,1}, x_{\ell,2}, \ldots, x_{\ell,N}).$$

By an analogous argument as in section 1.2.1, we have the following approximation of the partition function $\mathcal{Z}$:

$$\begin{aligned} \mathcal{Z} &= \operatorname{Tr}\left(\prod_{\ell=1}^L e^{-\Delta\tau\mathcal{H}_K} e^{-\Delta\tau\mathcal{H}_V}\right) \\ &= (C_2)^{NL} \int [\delta x] e^{-S_B(x)} \operatorname{Tr}\left(\prod_{\ell=1}^L e^{-\Delta\tau\mathcal{H}_{K+}} e^{\Delta\tau\mathcal{H}_{V+}^\ell}\right) \times \\ &\qquad\qquad \operatorname{Tr}\left(\prod_{\ell=1}^L e^{-\Delta\tau\mathcal{H}_{K-}} e^{\Delta\tau\mathcal{H}_{V-}^\ell}\right). \end{aligned}$$

Note that all the operators $e^{-\Delta\tau\mathcal{H}_K}$, $e^{-\Delta\tau\mathcal{H}_{V+}^\ell}$ and $e^{-\Delta\tau\mathcal{H}_{V-}^\ell}$ are quadratic in the fermion operators. By the argument (1.16), we derive the following path integral expression for the partition function $\mathcal{Z}$:

$$\begin{aligned} Z_x &= (C_2)^{NL} \int [\delta x] e^{-S_B(x)} \det\left(I + \prod_{\ell=1}^L e^{t\Delta\tau K} e^{(2U)^{\frac{1}{2}}\Delta\tau V_l(x_\ell)}\right) \times \\ &\qquad\qquad \det\left(I + \prod_{\ell=1}^L e^{t\Delta\tau K} e^{-(2U)^{\frac{1}{2}}\Delta\tau V_\ell(x_\ell)}\right) \\ &= (C_2)^{NL} \int [\delta x] e^{-S_B(x)} \det[M_+(x)] \det[M_-(x)], \end{aligned} \tag{1.23}$$

where for $\sigma = \pm$, the fermion matrices

$$M_\sigma(x) = I + B_{L,\sigma}(x_L) B_{L-1,\sigma}(x_{L-1}) \cdots B_{1,\sigma}(x_1), \tag{1.24}$$

and for $\ell = 1, 2, \ldots, L$,

$$B_{\ell,\sigma}(x_\ell) = e^{t\Delta\tau K} e^{\sigma(2U)^{\frac{1}{2}}\Delta\tau V_\ell(x_\ell)}. \tag{1.25}$$

By a so-called particle-hole transformation (see Appendix B), we have

$$\det[M_-(x)] = e^{-\Delta\tau(2U)^{\frac{1}{2}}\sum_{\ell,i} x_{\ell,i}} \det[M_+(x)]. \tag{1.26}$$

Therefore, the integrand of Z_x in (1.23) is positive definite.[6]

Consequently, a computable approximation of the distribution operator $\mathcal{P}$ is given by

$$P(x) = \frac{\eta_h'}{Z_x} e^{-S_B(x)} \det[M_+(x)] \det[M_-(x)]. \tag{1.27}$$

where $\eta_h' = (C_2)^{NL}$ is a normalizing constant.

1.3.2 Algorithm

At this point, our computational task becomes how to generate Hubbard-Stratonovich variables (configurations) x that follow the probability distribution function $P(x)$ defined as (1.27). To develop an efficient Monte Carlo method, we reformulate the the function $P(x)$. First, let us recall the following two facts:

1. Let $M_\sigma(x)$ denote a $L \times L$ block matrix[7]

$$M_\sigma(x) = \begin{bmatrix} I & & & & B_{1,\sigma}(x_1) \\ -B_{2,\sigma}(x_2) & I & & & \\ & -B_{3,\sigma}(x_3) & I & & \\ & & \ddots & \ddots & \\ & & & -B_{L,\sigma}(x_L) & I \end{bmatrix}, \tag{1.28}$$

where $B_{\ell,\sigma}(x_\ell)$ are $N \times N$ matrices as defined in (1.25). Then we have[8]

$$\det[M_\sigma(x)] = \det[I + B_{L,\sigma}(x_L) B_{L-1,\sigma}(x_{L-1}) \cdots B_{1,\sigma}(x_1)]. \tag{1.29}$$

[6] Note that we assume that $\mu = 0$ (half-filling case). Otherwise, there exists "sign problem": $P(x)$ may be negative and can not be used as a probability distribution function.

[7] We use the same notation $M_\sigma(x)$ to denote the $N \times N$ matrix as defined in (1.24), and $NL \times NL$ matrix as defined in (1.28). It depends on the context which one we refer to.

[8] The identity can be easily derived based on the following observation. If A is a 2×2 block matrix,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

then $\det(A) = \det(A_{22})\det(F_{11}) = \det(A_{11})\det(F_{22})$, where $F_{11} = A_{11} - A_{12}A_{22}^{-1}A_{21}$ and $F_{22} = A_{22} - A_{21}A_{11}^{-1}A_{12}$.

2. If F is an $N \times N$ symmetric and positive definite matrix, then[9].

$$\int e^{-v^T F^{-1} v} dv = \pi^{\frac{N}{2}} \det[F^{\frac{1}{2}}], \tag{1.30}$$

Now, by introducing two auxiliary fields Φ_σ, $\sigma = \pm$, we have

$$|\det[M_\sigma(x)]| = \det\left[M_\sigma^T(x) M_\sigma(x)\right]^{\frac{1}{2}} = \pi^{-\frac{NL}{2}} \int e^{-\Phi_\sigma^T A_\sigma^{-1}(x) \Phi_\sigma}, \tag{1.31}$$

where

$$A_\sigma(x) = M_\sigma(x)^T M_\sigma(x).$$

By combining (1.23) and (1.31), the expression (1.23) of the partition function Z_x can be recast as the following[10]

$$\begin{aligned} Z_x &= (C_2)^{NL} \int [\delta x] e^{-S_B(x)} \det[M_+(x)] \det[M_-(x)] \\ &= \left(\frac{C_2}{\pi}\right)^{NL} \int [\delta x \delta\Phi_+ \delta\Phi_-] e^{-\left(S_B(x) + \Phi_+^T A_+^{-1}(x)\Phi_+ + \Phi_-^T A_-^{-1}(x)\Phi_-\right)} \\ &\equiv \left(\frac{C_2}{\pi}\right)^{NL} \int [\delta x \delta\Phi_\sigma] e^{-V(x,\Phi_\sigma)}, \end{aligned}$$

where

$$V(x, \Phi_\sigma) = S_B(x) + \Phi_+^T A_+^{-1}(x)\Phi_+ + \Phi_-^T A_-^{-1}(x)\Phi_-. \tag{1.32}$$

Now let us consider how to move the configuration x satisfying the distribution:

$$P(x, \Phi_\sigma) \propto \frac{1}{Z_x} e^{-V(x,\Phi_\sigma)}. \tag{1.33}$$

Similar to the DQMC method, at each Monte Carlo step, we can try to move $x = \{x_{\ell,i}\}$ with respect to each imaginary time ℓ and spatial site i. Alternatively, we can also move the entire configuration x by adding a Gaussian noise

$$x \longrightarrow x + \Delta x,$$

where

$$\Delta x = -\frac{\partial V(x, \Phi_\sigma)}{\partial x} \Delta t + \sqrt{\Delta t}\, w_t,$$

Δt is a parameter of step size, and w_t follows a Gaussian distribution $N(0,1)$ with mean 0 and variance 1. It is known as Langevin-Euler move, for example, see [10, p.192].

[9]The identity can be proven by using the eigen-decomposition of F. For example, see page 97 of [R. Bellman, Introduction to Matrix Analysis, SIAM Edition, 1997].

[10]Here we assume that $\det[M_\sigma(x)]$ is positive. Otherwise, we will have the so-called "sign problem".

Spurred by the popularity of the molecular dynamics (MD) method, Scalettar *et al* [15] proposed a hybrid method to move x by combining Monte Carlo and molecular dynamics and derived a so-called Hybrid Quantum Monte Carlo (HMQC) method. In the HQMC, an additional auxiliary momentum field $p = \{p_{\ell,i}\}$ is introduced. By the identity

$$\int_{-\infty}^{\infty} e^{-z^2} dz = \sqrt{\pi},$$

we see that the partition function Z_x can be rewritten as

$$\begin{aligned} Z_x &= (C_2)^{NL}\,\pi^{-NL} \int [\delta x \delta \Phi_\sigma] e^{-V(x,\Phi_\sigma)} \\ &= (C_2)^{NL}\pi^{-\frac{3NL}{2}} \int [\delta x \delta p \delta \Phi_\sigma] e^{-\left[\sum_{\ell,i} p_{\ell,i}^2 + V(x,\Phi_\sigma)\right]} \\ &= (C_2)^{NL}\pi^{-\frac{3NL}{2}} \int [\delta x \delta p \delta \Phi_\sigma] e^{-H(x,p,\Phi_\sigma)} \\ &\equiv Z_H, \end{aligned}$$

where

$$\begin{aligned} H(x,p,\Phi_\sigma) &= p^T p + V(x,\Phi_\sigma) \\ &= p^T p + S_B(x) + \Phi_+^T A_+^{-1}(x)\Phi_+ + \Phi_-^T A_-^{-1}(x)\Phi_-. \end{aligned} \tag{1.34}$$

At this point, the computational task becomes a classical Monte Carlo problem: seek the configurations $\{x, p, \Phi_\sigma\}$ that obey the following probability distribution function

$$P(x,p,\Phi_\sigma) = \frac{\eta_h}{Z_H} e^{-H(x,p,\Phi_\sigma)}, \tag{1.35}$$

where $\eta_h = (C_2)^{NL}\pi^{-\frac{3NL}{2}}$ is a normalizing constant.

Recall that initially we are interested in drawing samples x from the distribution $P(x,\Phi_\sigma)$ defined in (1.33). The motivation of introducing the auxiliary momentum variable p is as the following. If we draw samples (x,p) from the distribution $P(x,p,\Phi_\sigma)$ defined in (1.35), then marginally, it can be shown that x follow the desired distribution $P(x,\Phi_\sigma)$ and entries of p follow the Gaussian distribution $N(0,\frac{1}{2})$ with mean 0 and variance $\frac{1}{2}$ (the standard deviation $\frac{1}{\sqrt{2}}$).

If the configurations (x,p) are moved satisfying the Hamiltonian equations

$$\dot{p}_{\ell,i} = -\frac{\partial H}{\partial x_{\ell,i}} = -\frac{\partial V}{\partial x_{\ell,i}}, \tag{1.36}$$

$$\dot{x}_{\ell,i} = \frac{\partial H}{\partial p_{\ell,i}} = 2p_{\ell,i}, \tag{1.37}$$

then by Liouville's theorem[11], (x, p) is moved along a trajectory in which both H and the differential volume element in phase space are constant and the system is preserved in equilibrium.

In our current implementation of the HQMC method, Hamiltonian equations (1.36) and (1.37) are solved by Verlet (leap-frog) method, see [4]. It is a simple and efficient numerical method to move the configuration from $(x(0), p(0))$ to $(x(T), p(T))$. Since the MD algorithm is "time reversible", HQMC transition guarantees the invariance of the target distribution. This is to say that if the starting x follows the target distribution $P(x, \Phi_\sigma)$, then the new configuration $x(T)$ still follows the same distribution [10, Chap. 9].

Finally, the update of the auxiliary field Φ_σ is simply done by the MC strategy:

$$\Phi_\sigma = M_\sigma^T R_\sigma,$$

where the entries of the vector R_σ are chosen from the distribution $N(0, \frac{1}{2})$. Note that the fields Φ_σ and p are both auxiliary. We first fix the fields Φ_σ, vary the fields p, and move (x, p) together. It is possible to use a different moving order.

The following is an outline of the Hybrid Quantum Monte Carlo (HQMC) method.

```
HQMC
• Initialize
    – $x(0) = 0$,
    – $\Phi_\sigma = M_\sigma^T(x(0))R_\sigma, = M_\sigma^T(x(0))(R^\sigma_{\ell,i})$, $R^\sigma_{\ell,i} \sim N(0, \frac{1}{2})$.
• MC loop (total steps = warm up + measurement )
    1. MD steps $(x(0), p(0)) \longrightarrow (x(T), p(T))$ with step
       size $\Delta t$:
       (a) Initialize $p(0) = (p_{\ell,i})$, $p_{\ell,i} \sim N(0, \frac{1}{2})$.
       (b) $p$ is evolved in a half time step $\frac{1}{2}\Delta t$:
           $$p_{\ell,i}(\frac{1}{2}\Delta t) = p_{\ell,i}(0) - \left[\frac{\partial V(x(0), \Phi_\sigma)}{\partial x_{\ell,i}}\right](\frac{1}{2}\Delta t).$$
       (c) For $n = 0, 1, \ldots, N_T - 2$,
           $$x_{\ell,i}(t_n + \Delta t) - x_{\ell,i}(t_n) = 2p_{\ell,i}(t_n + \frac{1}{2}\Delta t)\Delta t,$$
           $$p_{\ell,i}(t_n + \frac{3}{2}\Delta t) - p_{\ell,i}(t_n + \frac{1}{2}\Delta t) = - \left[\frac{\partial V(x(t_n + \Delta t), \Phi_\sigma)}{\partial x_{\ell,i}}\right]\Delta t.$$
```

[11] For example, see [1, Chap.3] or [12, Chap.2]

where $N_T = T/\Delta t$, $t_n = n\Delta t$ and $t_0 = 0$.

(d) For $n = N_T - 1$:

$$x_{\ell,i}(T) - x_{\ell,i}(t_n) = 2p_{\ell,i}(t_n + \frac{1}{2}\Delta t)\Delta t,$$
$$p_{\ell,i}(T) - p_{\ell,i}(t_n + \frac{1}{2}\Delta t) = -\left[\frac{\partial V(x(T),\Phi_\sigma)}{\partial x_{\ell,i}}\right]\Delta t.$$

2. Metropolis acceptance-rejection:

$$x(T) = \begin{cases} x(T), \text{ if } r \le \min\left\{1, \frac{e^{-H(x(T),p(T),\Phi_\sigma)}}{e^{-H(x(0),p(0),\Phi_\sigma)}}\right\} \\ x(0), \text{ otherwise,} \end{cases}$$

where r is a random number chosen from Uniform$[0,1]$.

3. Perform the "heat-bath" step

$$\Phi_\sigma = M_\sigma^T(x(T))R_\sigma = M_\sigma^T(x(T))(R_{\ell,i}^\sigma),$$

where $R_{\ell,i}^\sigma \sim N(0, \frac{1}{2})$.

4. Perform physical measurements after warm up steps.
5. Set $x(0) := x(T)$, go to MD step 1.

Remark 1.8. The Langevin-Euler update is equivalent to a single-step HQMC move [10]. The MD and Langevin-Euler are two alternate methods which both have the virtue of moving all the variables together. Which is better depends basically on which allows the larger step size, the fastest evolution of the Hubbard-Stratonovich fields to new values.

Remark 1.9. If Δt is sufficient small,

$$H(x(T),p(T),\Phi_\sigma) = H(x(0),p(0),\Phi_\sigma)$$

since the MD conserves H. Then at step 2 of the MC loop, all moves will be accepted. However, in practice, for finite Δt the integrator does not conserve H, so step 2 of the MC loop is needed to keep the algorithm exact.

To this end, let us consider how to compute the force term $\frac{\partial V(x,\Phi_\sigma)}{\partial x_{\ell,i}}$. First, we note the matrix $M_\sigma(x)$ defined in (1.28) can be compactly written as

$$M_\sigma(x) = I - K_{[L]}D_{[L]}^\sigma(x)\Pi, \tag{1.38}$$

where

$$K_{[L]} = \operatorname{diag}\left(e^{t\Delta\tau K}, \dots, e^{t\Delta\tau K}\right),$$
$$D_{[L]}^\sigma(x) = \operatorname{diag}\left(e^{\sigma(2U)^{\frac{1}{2}}\Delta\tau V_1(x_1)}, \dots, e^{\sigma(2U)^{\frac{1}{2}}\Delta\tau V_L(x_L)}\right)$$

and

$$\Pi = \begin{bmatrix} 0 & & & -I \\ I & 0 & & \\ & \ddots & \ddots & \\ & & I & 0 \end{bmatrix}.$$

By the definition of $V(x, \Phi_\sigma)$ in (1.32), and recall that

$$A_\sigma(x) = M_\sigma^T(x) M_\sigma(x)$$

and

$$X_\sigma = A_\sigma^{-1}(x)\Phi_\sigma,$$

we have

$$\begin{aligned} \frac{\partial[\Phi_\sigma^T A_\sigma^{-1}(x)\Phi_\sigma]}{\partial x_{\ell,i}} &= -\Phi_\sigma^T A_\sigma^{-1}(x) \frac{\partial A_\sigma(x)}{\partial x_{\ell,i}} A_\sigma^{-1}(x)\Phi_\sigma \\ &= -X_\sigma^T \frac{\partial A_\sigma(x)}{\partial x_{\ell,i}} X_\sigma \\ &= 2\left[M_\sigma(x) X_\sigma\right]^T \frac{\partial M_\sigma(x)}{\partial x_{\ell,i}} X_\sigma. \end{aligned}$$

By the expression (1.38) of $M_\sigma(x)$, we have

$$\frac{\partial M_\sigma(x)}{\partial x_{\ell,i}} = -K_{[L]} \frac{\partial D_{[L]}^\sigma(x)}{\partial x_{\ell,i}} \Pi.$$

Note that $D_{[L]}^\sigma(x)$ is a diagonal matrix, and only the (ℓ, i)-diagonal element

$$d_{\ell,i}^\sigma = \exp(\sigma (2U)^{\frac{1}{2}} \Delta\tau x_{\ell,i})$$

depends on $x_{\ell,i}$. Therefore,

$$\frac{\partial M_\sigma(x)}{\partial x_{\ell,i}} = -K_{[L]} \frac{\partial D_{[L]}^\sigma(x)}{\partial x_{\ell,i}} \Pi.$$

we have

$$\frac{\partial[\Phi_\sigma^T A_\sigma^{-1}(x)\Phi_\sigma]}{\partial x_{\ell,i}} = -2 \frac{\partial d_{\ell,i}^\sigma}{\partial x_{\ell,i}} \left[K_{[L]}^T M_\sigma(x) X_\sigma\right]_{\ell,i} (\Pi X_\sigma)_{\ell,i}.$$

Subsequently, the force term is computed by

$$\begin{aligned} \frac{\partial V(x, \Phi_\sigma)}{\partial x_{\ell,i}} &= 2\Delta\tau x_{\ell,i} - 2(2U)^{\frac{1}{2}} \Delta\tau d_{\ell,i}^{+} \left[K_{[L]}^T M_{+}(x) X_{+}\right]_{\ell,i} [\Pi X_{+}]_{\ell,i} \\ &\quad + 2(2U)^{\frac{1}{2}} \Delta\tau d_{\ell,i}^{-} \left[K_{[L]}^T M_{-}(x) X_{-}\right]_{\ell,i} [\Pi X_{-}]_{\ell,i}. \end{aligned}$$

Therefore, the main cost of the HQMC is on computing the force term, which in turn is on solving the symmetric positive definite linear system

$$A_\sigma(x)X_\sigma = \Phi_\sigma$$

where $\sigma = \pm$. In sections 3 and 4, we will discuss direct and iterative methods to such linear system of equations.

1.3.3 Physical measurements

In section 1.2.3 we described how measurements are made in a determinant QMC code. The procedure in the hybrid QMC is identical in that one expresses the desired quantities in terms of precisely the same products of Green's functions. The only difference is that these Green's functions are obtained from matrix-vector products instead of the entries of the Green's function. The basic identity is this:

$$2\,\langle X_{\sigma,i}R_{\sigma,j}\rangle \leftrightarrow (M_\sigma)^{-1}_{i,j} = G^\sigma_{ij}.$$

This follows from the fact that

$$X_\sigma = A^{-1}_\sigma(x)\Phi_\sigma = M^{-1}_\sigma(x)R_\sigma$$

and that the components R_i of R_σ are independently distributed Gaussian random numbers satisfying $\langle R_iR_j\rangle = \frac{1}{2}\delta_{i,j}$.

Hence, the expression for the spin-spin correlation function would become

$$\begin{aligned}\langle c(l)\rangle &= \langle c^\dagger_{i+l,\downarrow}c_{i+l,\uparrow}c^\dagger_{i,\uparrow}c_{i,\downarrow}\rangle \\ &= G^\uparrow_{i+l,i}\,G^\downarrow_{i,i+l} \leftrightarrow 4\,\langle R_{\uparrow,i+l}X_{\uparrow,i}R_{\downarrow,i}X_{\downarrow,i+l}\rangle.\end{aligned}$$

The only point to be cautious of concerns the evaluation of expectation values of four fermion operators if the operators have the same spin index. There it is important that two different vectors of random numbers are used: R_σ, X_σ and R'_σ, X'_σ. Otherwise the averaging over the Gaussian random numbers generates additional, unwanted, values:

$$\langle R_iR_jR_kR_l\rangle = \frac{1}{4}(\delta_{i,j}\delta_{k,l} + \delta_{i,k}\delta_{j,l} + \delta_{i,l}\delta_{j,k}),$$

whereas

$$\langle R'_iR'_jR_kR_l\rangle = \frac{1}{4}\delta_{i,j}\delta_{k,l}.$$

It should be apparent that if the indices i and j are in the same N dimensional block, we get the equal-time Green's function

$$\begin{aligned}G^\sigma &= M^{-1}_\sigma \\ &= [I + B_{L,\sigma}B_{L-1,\sigma}\cdots B_{1,\sigma}]^{-1},\end{aligned}$$

which is the quantity used in traditional determinant QMC.

However, choosing i, j in different N dimensional blocks, we can also access the nonequal-time Green's function,

$$\begin{aligned} G_{\ell_1,i;\ell_2,j} &= \langle c_i(\ell_1) c_j^\dagger(\ell_2) \rangle \\ &= \left(B_{\ell_1} B_{\ell_1 - 1} \cdots B_{\ell_2+1} (I + B_{\ell_2} \cdots B_1 B_L \cdots B_{\ell_2+1})^{-1} \right)_{ij} . \end{aligned}$$

At every measurement step in the HQMC simulation, the equal-time Green's function G_{ij} can be obtained from the diagonal block of M_σ^{-1} and the unequal-time Green's function $G_{\ell_1,i;\ell_2,j}$ can be computed from the (ℓ_1, ℓ_2) block submatrix of M_σ^{-1}.

As we already remarked in describing the measurements in determinant QMC, it is often useful to generalize the definition of correlation functions so that the pair of operators are separated in imaginary time as well as spatially. The values of the non-equal time Green's function allow us to evaluate these more general correlation functions $c(l, \Delta\tau)$. To motivate their importance we comment that just as the structure factor $S(q)$, the Fourier transform of the real space correlation function $c(l)$, describes the scattering of particles with change in momentum q, the Fourier transform of $c(l, \Delta\tau)$ into what is often called the susceptibility $\chi(q, \omega)$, tells us about scattering events where the momentum changes by q and the energy changes by ω.

2 Hubbard matrix analysis

For developing robust and efficient algorithmic techniques and high performance software for the QMC simulations described in section 1, it is important to understand mathematical and numerical properties of the underlying matrix computation problems. In this section, we study the dynamics and transitional behaviors of these properties as functions of multi-length scale parameters.

2.1 Hubbard matrix

To simplify the notation, we write the matrix M introduced in (1.28) as

$$M = \begin{bmatrix} I & & & & B_1 \\ -B_2 & I & & & \\ & -B_3 & I & & \\ & & \ddots & \ddots & \\ & & & -B_L & I \end{bmatrix} , \qquad (2.1)$$

and refer to it as the *Hubbard matrix*, where

- I is an $N \times N$ identity matrix,
- B_ℓ is an $N \times N$ matrix of the form

$$B_\ell = BD_\ell, \tag{2.2}$$

  and

$$B = e^{t\Delta\tau K} \quad \text{and} \quad D_\ell = e^{\sigma\nu V_\ell(h_\ell)},$$

- t is a hopping parameter,
- $\Delta\tau = \beta/L$, β is the inverse temperature, and L is the number of imaginary time slices,
- $\sigma = +$ or $-$,
- ν is a parameter related to the interacting energy parameter U,

$$\nu = \cosh^{-1} e^{\frac{U\Delta\tau}{2}} = (\Delta\tau U)^{\frac{1}{2}} + \frac{1}{12}(\Delta\tau U)^{\frac{3}{2}} + O((\Delta\tau U)^2)$$

  in the DQMC and

$$\nu = (2U)^{\frac{1}{2}}\Delta\tau$$

  in the HQMC,
- $K = (k_{ij})$ is a matrix describing lattice structure:

$$k_{ij} = \begin{cases} 1, \text{ if } i \text{ and } j \text{ are nearest neighbor,} \\ 0, \text{ otherwise,} \end{cases}$$

- $V_\ell(h_\ell)$ is an $N \times N$ diagonal matrix,

$$V_\ell(h_\ell) = \operatorname{diag}(h_{\ell,1}, h_{\ell,2}, \ldots, h_{\ell,N}),$$

- $h_{\ell,i}$ are random variables, referred to as Hubbard-Stratonovich field or configurations, $h_{\ell,i} = 1$ or -1 in the DQMC. In the HQMC, $h_{\ell,i}$ are obtained from the MD move.

The Hubbard matrix M displays the property of *multi-length scaling*, since the dimensions and numerical properties of M are characterized by multiple length and energy parameters, and random variables. Specifically, we have

1. Length parameters: N and L
   - N is the spatial size. If the density ρ is given, N also measures the number of electrons being simulated.
   - L is the number of blocks related to the inverse temperature $\beta = L\Delta\tau$.
2. Energy-scale parameters: t, U and β

   - t determines the hopping of electrons between different atoms in the solid and thus measures the material's kinetic energy.
   - U measures the strength of the interactions between the electrons, that is the potential energy.
   - β is the inverse temperature, $\beta = \frac{1}{k_B T}$, where k_B is the Boltzmann's constant and T is the temperature.

3. The parameter connecting length and energy scales: $\Delta\tau = \beta/L$.

   - $\Delta\tau$ is a discretization parameter, a measure of the accuracy of the Trotter-Suzuki decomposition, see (1.11).

In more complex situations other energy scales also enter, such as the frequency of ionic vibrations (phonons) and the strength of the coupling of electrons to those vibrations.

Under these parameters of multi-length scaling, the Hubbard matrix M has the following features:

- M incorporates multiple structural scales: The inverse temperature β determines the number of blocks $L = \beta/\Delta\tau$, where $\Delta\tau$ is a discretization step size. Typically $L = O(10^2)$. The dimension of the individual blocks is set by N the number of spatial sites. In a typical 2D simulations $N = N_x \times N_y = O(10^3)$. Thus the order of the Hubbard matrix M is about $O(10^5)$.
- M incorporates multiple energy scales: The parameter t determines the kinetic energy of the electrons, and the interaction energy scale U determines the potential energy. We will see that the condition number and eigenvalue distribution of the Hubbard matrix M are strongly influenced by these energy parameters.
- M is a function of NL random variables $h_{\ell i}$, the so-called Hubbard-Stratonovich field or configurations. The goal of the simulation is to sample these configurations which make major contributions to physical measurements. Therefore, the underlying matrix computation problems need to be solved several thousand times in a full simulation. Figure 2.1 shows a typical MD trajectory in a HQMC simulation, where at every step, we need to a linear system of equations associated with the Hubbard matrix M.

The matrix computation problems arising from the quantum Monte Carlo simulations include

1. Computation of the ratio of the determinants $\frac{\det[\widehat{M}]}{\det[M]}$, where $\widehat{M}$ is a low-rank update of M, see the Metropolis acceptance-rejection step in the DQMC algorithm, see section 1.2.2 and Appendix A.

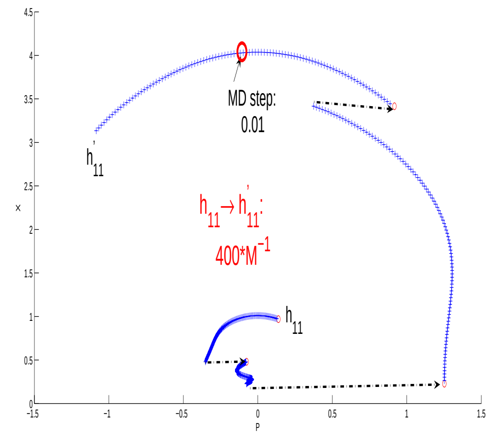

Figure 2.1: A typical MD trajectory in a HQMC simulation.

2. Solution of linear systems of the form $M^T M x = b$, see the HQMC algorithm for computing the force term in the molecular dynamics step, section 1.3.2.

3. Computation of certain entries and the traces of the inverse of the matrix M for physical observables, such as energy, density, moments, magnetism and superconductivity, see sections 1.2.3 and 1.3.3.

One of computational challenges associated with the QMC simulation of the Hubbard model is the wide range of values of parameters. For example, the spatial dimension is $N = N_x \times N_y$. When N is increased from $O(10^2)$ to $O(10^4)$, that is, to do a 10000 electron QMC simulation, it would have a tremendous impact on our understanding of strongly interacting materials. It would allow for the first time the simulation of systems incorporating a reasonable number of mesoscopic structures, such as a "checkerboard" electronic crystal [18], and stripe structure arising from removing electrons from the filling of one electron per site [21].

Another computational challenge is on the ill-conditioning of the underlying matrix computation problems when the energy scale parameters U and β are in certain ranges. In the rest of this section, we will illustrate these challenges in detail.

2.2 Basic properties

Let us first study some basic properties of the Hubbard matrix M as defined in (2.1).

1. The Hubbard matrix M can be compactly written as

$$M = I_{NL} - \operatorname{diag}(B_1, B_2, \dots, B_L)\Pi \tag{2.3}$$

or

$$M = I_{NL} - (I_N \otimes B)D_{[L]}(P \otimes I_N), \tag{2.4}$$

where I_N and I_{NL} are $N \times N$ and $NL \times NL$ unit matrices, respectively,

$$P = \begin{bmatrix} 0 & & & -1 \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix}, \quad \Pi = P \otimes I_N = \begin{bmatrix} 0 & & & -I_N \\ I_N & 0 & & \\ & \ddots & \ddots & \\ & & I_N & 0 \end{bmatrix}$$

and

$$\begin{aligned} B &= e^{t\Delta\tau K}, \\ D_{[L]} &= \operatorname{diag}(D_1, D_2, \dots, D_L). \end{aligned}$$

2. A block LU factorization of M is given by

$$M = LU, \tag{2.5}$$

where

$$L = \begin{bmatrix} I & & & & \\ -B_2 & I & & & \\ & -B_3 & I & & \\ & & \ddots & \ddots & \\ & & & -B_L & I \end{bmatrix}$$

and

$$U = \begin{bmatrix} I & & & & B_1 \\ & I & & & B_2B_1 \\ & & \ddots & & \vdots \\ & & & I & B_{L-1}B_{L-2}\cdots B_1 \\ & & & & I + B_LB_{L-1}\cdots B_1 \end{bmatrix}.$$

3. The inverses of the factors L and U are given by

$$L^{-1} = \begin{bmatrix} I & & & & \\ B_2 & I & & & \\ B_3B_2 & B_3 & I & & \\ \vdots & \ddots & \ddots & \ddots & \\ B_L\cdots B_2 & B_L\cdots B_3 & \cdots & B_L & I \end{bmatrix}$$

and

$$U^{-1} = \begin{bmatrix} I & & & & -B_1F \\ & I & & & -B_2B_1F \\ & & \ddots & & \vdots \\ & & & I & -B_{L-1}B_{L-2}\cdots B_1F \\ & & & & F \end{bmatrix}$$

where $F = (I + B_LB_{L-1}\cdots B_2B_1)^{-1}$.

4. The inverse of M is explicitly given by

$$M^{-1} = U^{-1}L^{-1} = W^{-1}Z, \tag{2.6}$$

where

$$W = \begin{bmatrix} I + B_1B_L\cdots B_2 & & & \\ & I + B_2B_1B_L\cdots B_3 & & \\ & & \ddots & \\ & & & I + B_LB_{L-1}\cdots B_1 \end{bmatrix}$$

and

$$Z = \begin{bmatrix} I & -B_1B_L\cdots B_3 & \cdots & -B_1 \\ B_2 & I & \cdots & -B_2B_1 \\ B_3B_2 & B_3 & \cdots & -B_3B_2B_1 \\ \vdots & \vdots & \vdots & \vdots \\ B_{L-1}\cdots B_2 & B_{L-1}\cdots B_3 & \cdots & -B_{L-1}\cdots B_2B_1 \\ B_L\cdots B_2 & B_L\cdots B_3 & \cdots & I \end{bmatrix}.$$

In other words, the (i,j) block submatrix of M^{-1} is given by

$$\{M^{-1}\}_{i,j} = (I + B_i\cdots B_1B_L\cdots B_{i+1})^{-1}Z_{ij}$$

where

$$Z_{ij} = \begin{cases} -B_iB_{i-1}\cdots B_1B_LB_{L-1}\cdots B_{j+1}, & i<j \\ I, & i=j \\ B_iB_{i-1}\cdots B_{j+1}, & i>j \end{cases}.$$

5. By the LU factorization (2.5), we immediately have the following determinant identity:

$$\det[M] = \det[I + B_LB_{L-1}\cdots B_1]. \tag{2.7}$$

2.3 Matrix exponential $B = e^{t\Delta\tau K}$

In this section, we discuss how to compute the matrix exponential

$$B = e^{t\Delta\tau K},$$

where K defines a 2-D $N_x \times N_y$ rectangle spatial lattice:

$$K = I_y \otimes K_x + K_y \otimes I_x,$$

I_x and I_y are unit matrices of dimensions N_x and N_y, respectively, and K_x and K_y are $N_x \times N_x$ and $N_y \times N_y$ matrices of the form

$$K_x, K_y = \begin{bmatrix} 0 & 1 & & & 1 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 0 & 1 \\ 1 & & & 1 & 0 \end{bmatrix},$$

and $\otimes$ is the Kronecker product.

A survey of numerical methods for computing the matrix exponential can be found in [19]. A simple approach is to use the eigendecomposition of the matrix K. First, by the definition of K and the property of the Kronecker product (see Appendix B.4), the matrix exponential $e^{t\Delta\tau K}$ can be written as the product of two matrix exponentials:

$$B = e^{t\Delta\tau K} = (I_y \otimes e^{t\Delta\tau K_x})(e^{t\Delta\tau K_y} \otimes I_x) = e^{t\Delta\tau K_y} \otimes e^{t\Delta\tau K_x}.$$

By straightforward calculation, we can verify the following lemma.

Lemma 2.1. The eigenvalues of K are

$$\kappa_{ij} = 2(\cos\theta_i + \cos\theta_j), \tag{2.8}$$

where

$$\theta_i = \frac{2i\pi}{N_x}, \quad \text{for} \quad i = 0, 1, 2, \ldots, N_x - 1$$
$$\theta_j = \frac{2j\pi}{N_y}, \quad \text{for} \quad j = 0, 1, 2, \ldots, N_y - 1.$$

The corresponding eigenvectors are

$$v_{ij} = u_j \otimes u_i,$$

where

$$u_i = \frac{1}{\sqrt{N_x}}[1, e^{\mathrm{i}\theta_i}, e^{\mathrm{i}2\theta_i}, \ldots, e^{\mathrm{i}(N_x-1)\theta_i}]^T,$$
$$u_j = \frac{1}{\sqrt{N_y}}[1, e^{\mathrm{i}\theta_j}, e^{\mathrm{i}2\theta_j}, \ldots, e^{\mathrm{i}(N_y-1)\theta_j}]^T.$$

By Lemma 2.1, we can use the FFT to compute B. The computational complexity of formulating the matrix B explicitly is $O(N^2)$. The cost of the matrix-vector multiplication is $O(N \log N)$.

We now consider a computational technique referred to as the "*checkerboard method*" in computational physics. The method is particularly useful when the hopping parameter t depends on the location (i,j) on the lattice, i.e. t is not a constant. The checkerboard method only costs $O(N)$. Let us describe this method in detail. For simplicity, assume that N_x and N_y are even. Write K_x as

$$K_x = K_x^{(1)} + K_x^{(2)},$$

where

$$K_x^{(1)} = \begin{bmatrix} D & & & \\ & D & & \\ & & \ddots & \\ & & & D \end{bmatrix}, \quad K_x^{(2)} = \begin{bmatrix} 0 & & & & 1 \\ & D & & & \\ & & \ddots & & \\ & & & D & \\ 1 & & & & 0 \end{bmatrix}, \quad D = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Note that for any scalar $\alpha \neq 0$, the matrix exponential $e^{\alpha D}$ is given by

$$e^{\alpha D} = \begin{bmatrix} \cosh\alpha & \sinh\alpha \\ \sinh\alpha & \cosh\alpha \end{bmatrix}.$$

Therefore, we have

$$e^{\alpha K_x^{(1)}} = \begin{bmatrix} e^{\alpha D} & & & \\ & e^{\alpha D} & & \\ & & \ddots & \\ & & & e^{\alpha D} \end{bmatrix}, \quad e^{\alpha K_x^{(2)}} = \begin{bmatrix} \cosh\alpha & & & & \sinh\alpha \\ & e^{\alpha D} & & & \\ & & \ddots & & \\ & & & e^{\alpha D} & \\ \sinh\alpha & & & & \cosh\alpha \end{bmatrix}.$$

Since $K_x^{(1)}$ does not commute with $K_x^{(2)}$, we use the Trotter-Suzuki approximation

$$e^{\alpha K_x} = e^{\alpha K_x^{(1)}} e^{\alpha K_x^{(2)}} + O(\alpha^2).$$

By an exactly analogous calculation, we have the approximation

$$e^{\alpha K_y} = e^{\alpha K_y^{(1)}} e^{\alpha K_y^{(2)}} + O(\alpha^2).$$

Subsequently, we have

$$\begin{aligned} B &= e^{t\Delta\tau K_y} \otimes e^{t\Delta\tau K_x} \\ &= (e^{t\Delta\tau K_y^{(1)}} e^{t\Delta\tau K_y^{(2)}}) \otimes (e^{t\Delta\tau K_x^{(1)}} e^{t\Delta\tau K_x^{(2)}}) + O((t\Delta\tau)^2). \end{aligned}$$

Therefore, when $t\Delta\tau$ is small, the matrix B can be approximated by

$$\widehat{B} = (e^{t\Delta\tau K_y^{(1)}} e^{t\Delta\tau K_y^{(2)}}) \otimes (e^{t\Delta\tau K_x^{(1)}} e^{t\Delta\tau K_x^{(2)}}). \tag{2.9}$$

There are no more than 16 nonzero elements in each row and column of the matrix $\widehat{B}$. If $\cosh\alpha$ and $\sinh\alpha$ are computed in advance, the cost of constructing the matrix $\widehat{B}$ is $16N$.

Note the the approximation $\widehat{B}$ is not symmetric. A symmetric approximation of B is given by

$$\widehat{B} = (e^{\frac{t\Delta\tau}{2} K_y^{(2)}} e^{t\Delta\tau K_y^{(1)}} e^{\frac{t\Delta\tau}{2} K_y^{(2)}}) \otimes (e^{\frac{t\Delta\tau}{2} K_x^{(2)}} e^{t\Delta\tau K_x^{(1)}} e^{\frac{t\Delta\tau}{2} K_x^{(2)}}).$$

In this case, there are no more than 36 nonzero elements in each row and column.

Sometimes, it is necessary to compute the matrix-vector multiplication $\widehat{B}w$. Let w be a vector of the dimension $N = N_x \times N_y$ obtained by stacking the columns of an $N_x \times N_y$ matrix W into one long vector:

$$w = \text{vec}(W).$$

Then it can be verified that

$$\widehat{B}w = \text{vec}(e^{t\Delta\tau K_x^{(2)}} e^{t\Delta\tau K_x^{(1)}} W e^{t\Delta\tau K_y^{(1)}} e^{t\Delta\tau K_y^{(2)}}). \tag{2.10}$$

As a result, the cost of the matrix-vector multiplication $\widehat{B}w$ is $12N$ flops. It can be further reduced by rewriting the block $e^{\alpha D}$ as

$$e^{\alpha D} = \cosh\alpha \begin{bmatrix} 1 & \tanh\alpha \\ \tanh\alpha & 1 \end{bmatrix}.$$

Using this trick, the cost of the matrix-vector multiplication $\widehat{B}w$ is $9N$ flops.

2.4 Eigenvalue distribution of M

The study of eigenvalues of a cyclic matrix of the form (2.1) can be traced back to the work of Frobenius, Romanovsky and Varga, see [20]. The following theorem characterizes the eigenvalues of the Hubbard matrix M defined in (2.1).

Theorem 2.1. For each eigenpair (θ, z) of the matrix $B_L \cdots B_2 B_1$:

$$(B_L \cdots B_2 B_1)z = \theta z,$$

there are L corresponding eigenpairs (λ_ℓ, v_ℓ) of the matrix M:

$$M v_\ell = \lambda_\ell v_\ell \quad \text{for } \ell = 0, 1, \ldots, L-1,$$

where

$$\lambda_\ell = 1 - \mu_\ell \quad \text{and} \quad v_\ell = \begin{bmatrix} (B_L \cdots B_3 B_2)^{-1} \mu_\ell^{L-1} z \\ (B_L \cdots B_3)^{-1} \mu_\ell^{L-2} z \\ \vdots \\ B_L^{-1} \mu_\ell z \\ z \end{bmatrix}.$$

and $\mu_\ell = \theta^{\frac{1}{L}} e^{\mathrm{i}\frac{(2\ell+1)\pi}{L}}$

PROOF. By verifying $(I - M)v_\ell = \mu_\ell v_\ell$. □

2.4.1 The case $U = 0$

When the Hubbard system without Coulomb interaction, $U = 0$, we have

$$B_1 = B_2 = \cdots = B_L = B = e^{t\Delta\tau K}.$$

In this case, the eigenvalues of the matrix M are known explicitly.

Theorem 2.2. When $U = 0$, the eigenvalues of the matrix M are

$$\lambda(M) = 1 - e^{t\Delta\tau\kappa_{ij}} e^{\mathrm{i}\frac{(2\ell+1)\pi}{L}}, \tag{2.11}$$

for $i = 0, 1, \ldots, N_x - 1$, $j = 0, 1, \ldots, N_y - 1$ and $\ell = 0, 1, \ldots L - 1$, where κ_{ij} is defined in (2.8). Furthermore,

$$\max |1 - \lambda(M)| = e^{4t\Delta\tau} \quad \text{and} \quad \min |1 - \lambda(M)| = e^{-4t\Delta\tau}.$$

Figure 2.2 shows the eigenvalue distribution of the matrix M with the setting $(N, L, U, \beta, t) = (4 \times 4, 8, 0, 1, 1)$. In this case, the order of the matrix M is $NL = 4 \times 4 \times 8 = 128$. Theorem 2.2 can be used to interpret the distribution. It has a ring structure, centered at $(1, 0)$. On every ring there are $L = 8$ circles. Alternatively, we can also view that the eigenvalues are distributed on $L = 8$ rays, originated from the point $(1, 0)$. The eigenvalues κ_{ij} of the matrix K only have 5 different values. There are total 40 circles, which indicate the multiplicity of some eigenvalues.

Let us examine the dynamics of eigenvalue distributions of M under the variation of the parameters N, L, U and t.

1. Lattice size N: Figure 2.3 shows the eigenvalue distributions for $N = 4 \times 4$ and $N = 16 \times 16$. Other parameters are set as $(L, U, \beta, t) = (8, 0, 1, 1)$. Since L is fixed, the number of rays does not change. When N is increased from 4×4 to 16×16, there are more points (eigenvalues) on each ray. Note that the range of eigenvalue distribution on each ray stays the same.

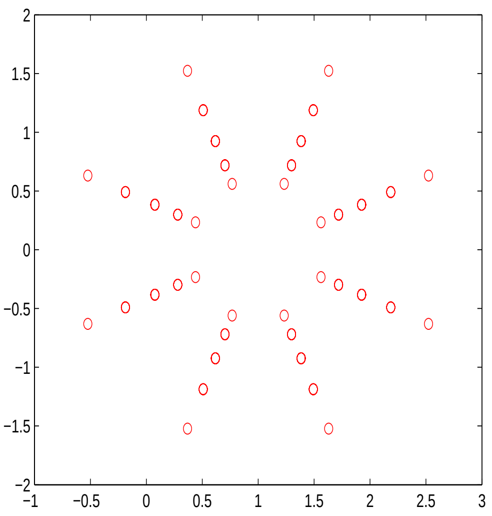

Figure 2.2: Eigenvalue distribution of M.

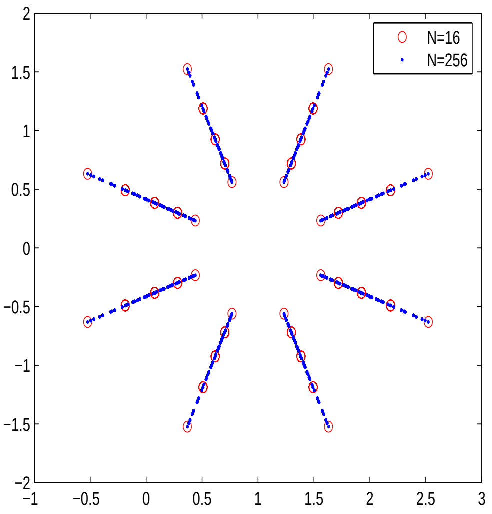

Figure 2.3: Eigenvalue distributions of M for different N.

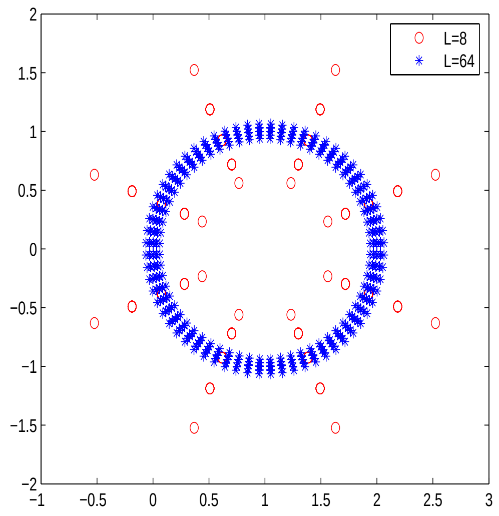

Figure 2.4: Eigenvalue distributions of M for different L.

2. Block number L: Figure 2.4 shows the eigenvalue distribution for block numbers $L = 8$ and $L = 64$. Other parameters are set as $(N, U, \beta, t) = (4 \times 4, 0, 1, 1)$. As we observe that as L increases, the number of rays increases, and the range of the eigenvalue distribution on each ray shrinks and becomes more clustered.

3. Block number L and hopping parameter t: Figure 2.5 shows the eigenvalue distributions for pairs $(L, t) = (8, 1)$ and $(64, 8)$. Other parameters are set as $(N, U, \beta) = (4 \times 4, 0, 1)$. By Theorem 2.2, we know that the points (eigenvalues) on each ring are L. When L increases, the points on each ring increase. At the same time, since $\Delta\tau = \frac{1}{L}$, the range of $|1 - \lambda(M)|$ is $[e^{-\frac{4t}{L}}\ e^{\frac{4t}{L}}]$, the bandwidth of the range will shrink when L increases. Since the ratio $\frac{t}{L}$ is fixed, the bandwidth of the ring keeps the same.

2.4.2 The case $U \neq 0$

Unfortunately, there is no explicit expression for the eigenvalues of the matrix M when $U \neq 0$. Figure 2.6 shows that as U increases, the eigenvalues on each ray tend to spread out. Other parameters are set as $(N, L, \beta, t) = (4 \times 4, 8, 1, 1)$.

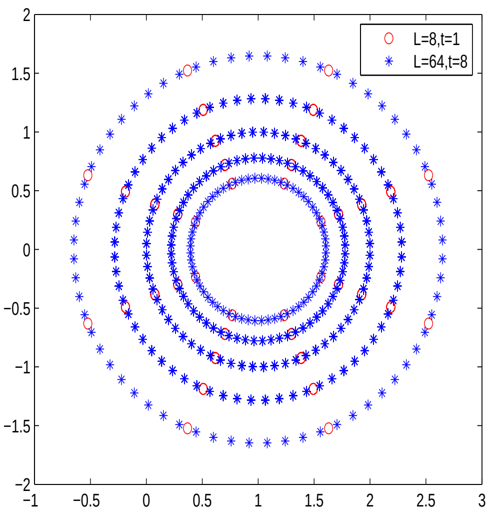

Figure 2.5: Eigenvalue distributions of M for different (L, t).

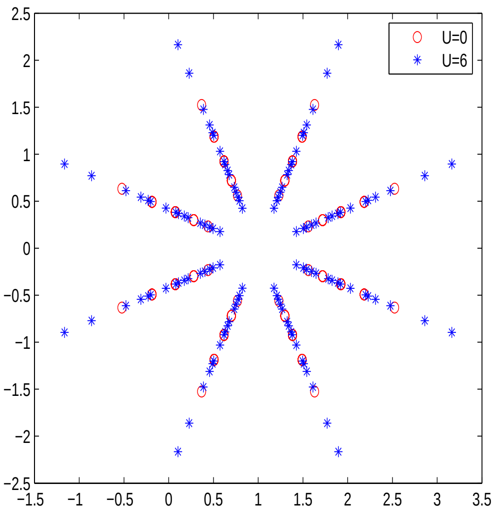

Figure 2.6: Eigenvalue distributions of M for different U.

2.5 Condition number of M

In this section, we study the condition number of the Hubbard matrix M defined in (2.1).

2.5.1 The case $U = 0$

When $U = 0$, M is a deterministic matrix and $B_1 = \cdots = B_L = B = e^{t\Delta\tau K}$. First, we have the following lemma about the eigenvalues of the symmetric matrix $M^T M$.

Lemma 2.2. When $U = 0$, the eigenvalues of $M^T M$ are

$$\lambda_\ell(M^T M) = 1 + 2\lambda(B)\cos\theta_\ell + (\lambda(B))^2, \tag{2.12}$$

where

$$\theta_\ell = \frac{(2\ell+1)\pi}{L}$$

for $\ell = 0, 1, \cdots, L-1$.

Proof. The lemma is based on the following fact. For any real number a, the eigenvalues of the matrix

$$A(a) = \begin{bmatrix} 1+a^2 & -a & & a \\ -a & 1+a^2 & -a & \\ & \ddots & \ddots & \ddots \\ a & & -a & 1+a^2 \end{bmatrix}$$

are $\lambda(A(a)) = 1 - 2a\cos\theta_\ell + a^2$. $\square$

We note that when a is a real number,

$$\sin^2\theta \leq 1 - 2a\cos\theta + a^2 \leq (1+|a|)^2.$$

Therefore, by the expression (2.12), we have the following inequalities to bound the largest and smallest eigenvalues of the matrix $M^T M$:

$$\max\lambda_\ell(M^T M) \leq (1 + \max|\lambda(B)|)^2$$

and

$$\min\lambda_\ell(M^T M) \geq \sin^2\frac{\pi}{L}.$$

By these inequalities, the norms of M and M^{-1} are bounded by

$$\|M\| = \max\lambda_\ell(M^T M)^{\frac{1}{2}} \leq 1 + \max|\lambda(B)|,$$

and

$$\|M^{-1}\| = \frac{1}{\min\lambda_\ell(M^T M)^{\frac{1}{2}}} \leq \frac{1}{\sin\frac{\pi}{L}}. \tag{2.13}$$

Note that $B = e^{t\Delta\tau K}$ and $\lambda_{\max}(K) = 4$, we have the following upper bound of the condition number $\kappa(M)$ of M.

Theorem 2.3. When $U = 0$,

$$\kappa(M) = \|M\| \, \|M^{-1}\| \leq \frac{1 + e^{4t\Delta\tau}}{\sin \frac{\pi}{L}} = O(L). \tag{2.14}$$

By the theorem, we conclude that when $U = 0$, the matrix M is well-conditioned.

2.5.2 The case $U \neq 0$

We now consider the situation when $U \neq 0$. By the representation of M in (2.3), we have

$$\|M\| \leq 1 + \max_{\ell} \|B_\ell\| \, \|\Pi\| = 1 + \max_{\ell} \|B_\ell\| \leq 1 + e^{4t\Delta\tau + \nu}. \tag{2.15}$$

To bound $\|M^{-1}\|$, we first consider the case where U is small. In this case, we can treat it as a small perturbation of the case $U = 0$. We have the following result.

Theorem 2.4. If U is sufficient small such that

$$e^{\nu} < 1 + \sin \frac{\pi}{L}, \tag{2.16}$$

then

$$\kappa(M) = \|M\| \, \|M^{-1}\| \leq \frac{1 + e^{4t\Delta\tau + \nu}}{\sin \frac{\pi}{L} + 1 - e^{\nu}}.$$

PROOF: First we note that M can be written as a perturbation of M at $U = 0$:

$$M = M_0 + \mathrm{diag}(B - B_\ell)\Pi,$$

where M_0 denotes the matrix M when $U = 0$. Therefore, if

$$\|M_0^{-1} \mathrm{diag}(B - B_\ell)\| < 1,$$

then we have

$$\|M^{-1}\| \leq \frac{\|M_0^{-1}\|}{1 - \|M_0^{-1} \mathrm{diag}(B - B_\ell)\|}. \tag{2.17}$$

Note that $\|\Pi\| = 1$.

Since the block elements of the matrix M_0 are B or I, and the eigenvalues of the matrix B and B^{-1} are the same, by following the proof of Lemma 2.2, we have

$$\|M_0^{-1} \mathrm{diag}(B)\| \leq \frac{1}{\sin \frac{\pi}{L}}.$$

Hence

$$\begin{aligned}\|M_0^{-1}\mathrm{diag}(B-B_\ell)\| &\le \|M_0^{-1}\mathrm{diag}(B)\|\cdot\|\mathrm{diag}(I-D_\ell)\| \\ &\le \frac{e^\nu-1}{\sin\frac{\pi}{L}}.\end{aligned}$$

If

$$\frac{e^\nu-1}{\sin\frac{\pi}{L}}<1,$$

then by (2.17) and (2.13), we have

$$\|M^{-1}\| \le \frac{\frac{1}{\sin\frac{\pi}{L}}}{1-\frac{e^\nu-1}{\sin\frac{\pi}{L}}} = \frac{1}{\sin\frac{\pi}{L}+1-e^\nu}. \tag{2.18}$$

The theorem is proven by combining inequalities (2.15) and (2.18). □

Note that the Taylor expansion of ν gives the expression

$$\nu=\sqrt{U\Delta\tau}+\frac{(U\Delta\tau)^{\frac{3}{2}}}{12}+O(U^2\Delta\tau^2). \tag{2.19}$$

Therefore, to the first-order approximation, the condition (2.16) is equivalent to

$$\sqrt{U}\le\frac{\pi}{\beta}\sqrt{\Delta\tau}+O(\Delta\tau). \tag{2.20}$$

Consequently, to the first order approximation, we have

$$\kappa(M)\le\frac{L(1+e^{4t\Delta\tau+\nu})}{\pi-\beta\sqrt{U\Delta\tau}-U\beta/2}+O(U^{\frac{3}{2}}\beta\Delta\tau^{\frac{1}{2}}).$$

By the inequality, we conclude that when U is sufficient small enough, M is well-conditioned and $\kappa(M)=O(L)$.

It is an open problem to find a rigorous sharp bound of $\kappa(M)$ when $U\neq 0$. Figure 2.7 shows the averages of the condition numbers of M for 100 Hubbard-Stratonovich configurations $h_{\ell,i}=\pm1$, where $N=16$, $L=8\beta$ with $\beta=[1:10]$, and $t=1$.

Figure 2.7 reveals several key issues concerning the transitional behaviors of the condition number of $\kappa(M)$:

1. The condition number increases much more rapidly than the linear rise which we know analytically at $U=0$.
2. Not only does the condition number increase with U, but also so do its fluctuations over the 100 chosen field configurations.
3. When the inverse temperature β increases, the conditioning of M becomes worse.

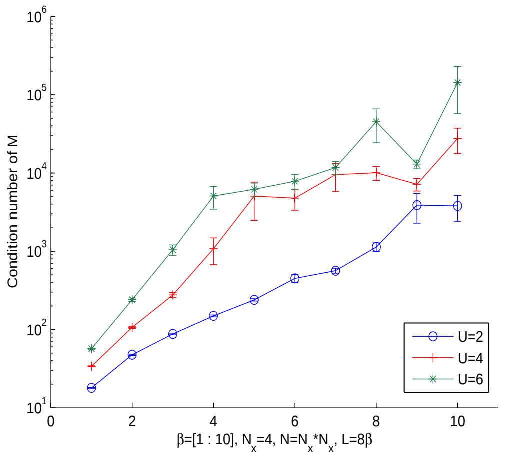

Figure 2.7: Condition numbers $\kappa(M)$ for different U

These observations tell us that the energy parameters U and β, which in turn determines L, are two critical parameters for the conditioning of the underlying matrix problems. The matrix M becomes ill-conditioned at low temperature (large β) or strong coupling (large U). It suggests that widely varying conditioning of the matrix problems is encountered in the course of a simulation, robust and efficient matrix solvers need to adopt different solution strategies depending on the conditioning of the underlying matrix problems.

2.6 Condition number of $M^{(k)}$

For an integer $k \leq L$, a structure-preserving factor-of-k reduction of the matrix M leads a matrix $M^{(k)}$ of the same block form

$$M^{(k)} = \begin{bmatrix} I & & & & B_1^{(k)} \\ -B_2^{(k)} & I & & & \\ & -B_3^{(k)} & I & & \\ & & \ddots & \ddots & \\ & & & -B_{L_k}^{(k)} & I \end{bmatrix}.$$

where $L_k = \lceil \frac{L}{k} \rceil$ is the number of blocks and

$$\begin{aligned} B_1^{(k)} &= B_k B_{k-1} \cdots B_2 B_1 \\ B_2^{(k)} &= B_{2k} B_{2k-1} \cdots B_{k+2} B_{k+1} \\ &\vdots \\ B_{L_k}^{(k)} &= B_L B_{L-1} \cdots B_{(L_k-1)k+1}. \end{aligned}$$

First we have the following observation. The inverse of $M^{(k)}$ is a "submatrix" of the inverse of M. Specifically, since M and $M^{(k)}$ have the same block cyclic structure, by the expression (2.6), we immediately have the following expression for the (i,j) block of $\{M^{(k)}\}^{-1}_{i,j}$:

$$\{M^{(k)}\}^{-1}_{i,j} = (I + B_i^{(k)} \cdots B_1^{(k)} B_L^{(k)} \cdots B_{i+1}^{(k)})^{-1} Z_{i,j}^{(k)}$$

where

$$Z_{i,j}^{(k)} = \begin{cases} -B_i^{(k)} \cdots B_1^{(k)} B_L^{(k)} \cdots B_{j+1}^{(k)}, & i < j \\ I, & i = j \\ B_i^{(k)} B_{i-1}^{(k)} \cdots B_{j+1}^{(k)}, & i > j \end{cases},$$

By the definition of $B_i^{(k)}$, and if $i \neq L^{(k)}$

$$\begin{aligned} \{M^{(k)}\}^{-1}_{i,j} = (I + B_{ik} \cdots B_1 B_L \cdots B_{i*k+1})^{-1} \times \\ \begin{cases} -B_{i*k} \cdots B_1 B_L \cdots B_{j*k+1}, & i < j \\ I, & i = j \\ B_{i*k} \cdots B_{j*k+1}, & i > j \end{cases}. \end{aligned}$$

Hence if $i \neq L^{(k)}$,

$$\{M^{(k)}\}^{-1}_{i,j} = \{M^{-1}\}_{i*k, j*k}.$$

If $i = L^{(k)}$, we have

$$\{M^{(k)}\}^{-1}_{i,j} = (I + B_L \cdots B_1)^{-1} \begin{cases} B_L \cdots B_{j*k+1}, & j < L \\ I, & j = L \end{cases}.$$

Hence if $i = L^{(k)}$,

$$\{M^{(k)}\}^{-1}_{L^{(k)},j} = \{M^{-1}\}_{L, j*k}.$$

We now turn to the discussion of the condition number of the matrix $M^{(k)}$. We have the following two immediate results:

1. The upper bound of the norm of the matrix $M^{(k)}$ is given by

$$\|M^{(k)}\| \leq 1 + e^{(4t\Delta\tau+\nu)k}. \tag{2.21}$$

This is due to the fact that $\|B_\ell^{(k)}\| \leq e^{(4t\Delta\tau+\nu)k}$.

2. The norm of the inverse of the matrix $M^{(k)}$ is bounded by

$$\|(M^{(k)})^{-1}\| \leq \|M^{-1}\|. \tag{2.22}$$

This is due to the fact that $(M^{(k)})^{-1}$ is a "submatrix" of M^{-1}.

By combining (2.21) and (2.22), we have an upper bound of the condition number of the reduced matrix $M^{(k)}$ in terms of the condition number of the original matrix M:

$$\begin{aligned}\kappa(M^{(k)}) &= \|M^{(k)}\|\,\|(M^{(k)})^{-1}\| \\ &\leq \frac{\|M^{(k)}\|}{\|M\|}\kappa(M) \\ &\leq \frac{1+e^{(4t\Delta\tau+\nu)k}}{\|M\|}\kappa(M) \\ &\leq c\,e^{(4t\Delta\tau+\nu)k}\kappa(M), \end{aligned}\tag{2.23}$$

where c is some constant. The inequality (2.23) shows that comparing to the condition number of M, the condition number of $M^{(k)}$ is amplified by a factor proportional to the reduction factor k.

For further details, we consider the following three cases:

1. $U = 0$ and $k = L$. In this case, the matrix M is reduced to a single block

$$\begin{aligned}M^{(L)} &= I + B_L \cdots B_2 B_1 \\ &= I + B \cdots BB \\ &= I + B^L \\ &= I + (e^{t\Delta\tau K})^L \\ &= I + e^{t\beta K}.\end{aligned}$$

By the eigendecomposition of the matrix K, see Lemma 2.1, the condition number of $M^{(L)}$ is given by

$$\kappa(M^{(L)}) = \frac{1+e^{4t\beta}}{1+e^{-4t\beta}}.$$

Therefore, for large β, $M^{(L)}$ is extremely ill-conditioned.

2. $U = 0$ and $k < L$ and $L_k = \frac{L}{k}$ is an integer. In this case, we have

$$\kappa(M^{(k)}) \leq \frac{1+e^{4t\Delta\tau k}}{\sin\frac{\pi}{L_k}}.$$

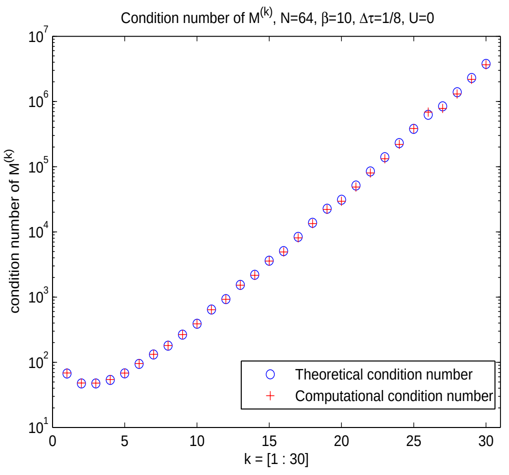

Figure 2.8: Condition numbers of $M^{(k)}$, $U = 0$.

The inequality can be proven similarly as the proof of Lemma 2.2. It is anticipated that the bound is still true when L/k is not an integer. Figure 2.8 shows condition numbers of $M^{(k)}$ with respect to the reduction factor k when $U = 0$. The computational and estimated condition numbers fit well.

3. For the general case of $U \neq 0$, we have the bound (2.23). Figure 2.9 shows the condition numbers of sample matrices $M^{(k)}$ (solid lines) for $U = 4$ and $U = 6$. The upper bound (2.23) are dashed line with circles. The mean of condition numbers $\kappa(M)$ of sample matrices M are used in the bound (2.23). It is clear that the upper bound (2.23) of the condition numbers of $M^{(k)}$ is an over estimation of the actual conditioning of $M^{(k)}$. This may partially due to the over-estimation of the norm of $M^{(k)}$. We observe that the condition number of $M^{(k)}$ is closer to $e^{\frac{1}{2}k(4t\Delta\tau+\nu)}\kappa(M)$. This is shown as the dashed lines with diamonds in the following plots. It is a subject of further study.

3 Self-adaptive direct linear solvers

In this section, we consider one of the computational kernels of the QMC simulations discussed in sections 1 and 2, namely solving the linear system of equations

$$Mx = b, \tag{3.1}$$

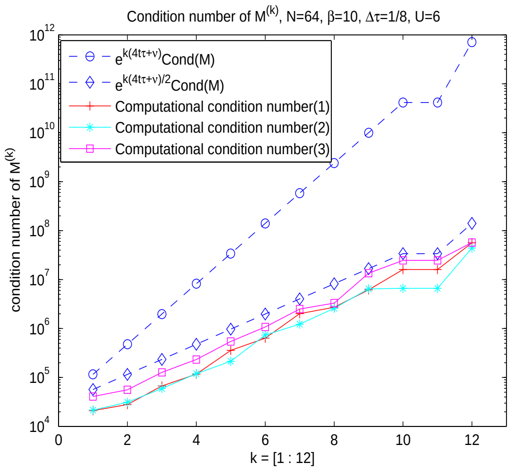

Figure 2.9: Condition numbers of $M^{(k)}$, $U = 6$.

where the coefficient matrix M is the Hubbard matrix as defined in (2.1) of section 2. One of challenges in QMC simulations is to develop algorithmic techniques that can robustly and efficiently solve numerical linear algebra problems with underlying multi-length scale coefficient matrices in a self-adapting fashion. In this section, we present such an approach for solving the linear system (3.1).

The structure of Hubbard matrix M exhibits the form of so-called block p-cyclic consistently ordered matrix [20]. p-cyclic matrices arise in a number of contexts in applied mathematics, such as numerical solution of boundary value problems for ordinary differential equations [28], finite-difference equations for the steady-state solution of parabolic equations with periodic boundary conditions [27], and the stationary solution of Markov chains with periodic graph structure [26]. It is known that the block Gaussian elimination with and without pivoting for solving p-cyclic linear systems could be numerically unstable, as shown in the cases arising from the multiple shooting method for solving two-point boundary value problems [30, 23] and Markov chain modeling [25]. Block cyclic reduction [22] is a powerful idea to solve such p-cyclic system since the number of unknowns is reduced exponentially. However, a full block cyclic reduction is inherently unstable and is only applicable for some energy scales, namely $U \leq 1$, due to ill-conditioning of the reduced system. A stable p-cyclic linear system solver is based on the structural orthogonal factorization [29, 23]. However, the costs of memory requirements and flops are prohibitively expensive when the length scale parameters N and L increase.

To take advantages of the block cyclic reduction and the structural orthogonal factorization method, in this section, we present a hybrid method to solve the linear system (3.1). The hybrid method performs a partial cyclic reduction in a self-adaptive way depending on system parameters such that the reduced system is still sufficiently well-conditioned to give rise a solution of the desired accuracy computed by the structural orthogonal factorization method. The hybrid method is called *self-adaptive block cyclic reduction*, or SABCR in short.

3.1 Block cyclic reduction

Consider the following 16×16 block cyclic linear system (3.1):

$$Mx = b,$$

where

$$M = \begin{bmatrix} I & & & & & & B_1 \\ -B_2 & I & & & & & \\ & -B_3 & I & & & & \\ & & -B_4 & I & & & \\ & & & \ddots & \ddots & & \\ & & & & -B_{15} & I & \\ & & & & & -B_{16} & I \end{bmatrix}.$$

Correspondingly, partitions of the vectors x and b are conformed to the blocks of M:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{15} \\ x_{16} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_{15} \\ b_{16} \end{bmatrix}.$$

A factor-of-four block cyclic reduction (BCR) leads to the following 4×4 block cycle system:

$$M^{(4)} x^{(4)} = b^{(4)}, \tag{3.2}$$

where

$$M^{(4)} = \begin{bmatrix} I & & & B_1^{(4)} \\ -B_2^{(4)} & I & & \\ & -B_3^{(4)} & I & \\ & & -B_4^{(4)} & I \end{bmatrix}, \quad x^{(4)} = \begin{bmatrix} x_4 \\ x_8 \\ x_{12} \\ x_{16} \end{bmatrix}, \quad b^{(4)} = \begin{bmatrix} b_1^{(4)} \\ b_2^{(4)} \\ b_3^{(4)} \\ b_4^{(4)} \end{bmatrix}$$

and

$$\begin{aligned} B_1^{(4)} &= B_4B_3B_2B_1, \\ B_2^{(4)} &= B_8B_7B_6B_5, \\ B_3^{(4)} &= B_{12}B_{11}B_{10}B_9, \\ B_4^{(4)} &= B_{16}B_{15}B_{14}B_{13}, \end{aligned}$$

and

$$\begin{aligned} b_1^{(4)} &= b_4 + B_4b_3 + B_4B_3b_2 + B_4B_3B_2b_1, \\ b_2^{(4)} &= b_8 + B_8b_7 + B_8B_7b_6 + B_8B_7B_6b_5, \\ b_3^{(4)} &= b_{12} + B_{12}b_{11} + B_{12}B_{11}b_{10} + B_{12}B_{11}B_{10}b_9, \\ b_4^{(4)} &= b_{16} + B_{16}b_{15} + B_{16}B_{15}b_{14} + B_{16}B_{15}B_{14}b_{13}. \end{aligned}$$

Therefore, to solve the original linear system (3.1), one can first solve the reduced system (3.2). Once the vector $x^{(4)}$ is computed, i.e, the block components x_4, x_8, x_{12} and x_{16} of the solution x are computed, the rest of block components x_i of the solution x can be computed by the following forward and back substitutions:

- Forward substitution:

$$\begin{aligned} x_1 &= b_1 - B_1x_{16}, & x_2 &= b_2 + B_2x_1, \\ x_5 &= b_5 + B_5x_4, & x_6 &= b_6 + B_6x_5, \\ x_9 &= b_9 + B_9x_8, & x_{10} &= b_{10} + B_{10}x_9, \\ x_{13} &= b_{13} + B_{13}x_{12}, & x_{14} &= b_{14} + B_{14}x_{13}, \end{aligned}$$

- Back substitution:

$$\begin{aligned} x_3 &= B_4^{-1}(x_4 - b_4), & x_7 &= B_8^{-1}(x_8 - b_8), \\ x_{11} &= B_{12}^{-1}(x_{12} - b_{12}), & x_{15} &= B_{16}^{-1}(x_{16} - b_{16}), \end{aligned}$$

The use of both forward and back substitutions is intended to minimize the propagation of rounding errors in the computed components x_4, x_8, x_{12} and x_{16}.

The pattern for a general factor-of-k reduction is clear. Given an integer $k \leq L$, a-factor-of-k BCR leads to a $L_k \times L_k$ block cycle linear system:

$$M^{(k)}x^{(k)} = b^{(k)}, \tag{3.3}$$

where $L_k = \lceil \frac{L}{k} \rceil$,

$$M^{(k)} = \begin{bmatrix} I & & & & B_1^{(k)} \\ -B_2^{(k)} & I & & & \\ & -B_3^{(k)} & I & & \\ & & \ddots & \ddots & \\ & & & -B_{L_k}^{(k)} & I \end{bmatrix},$$

with

$$\begin{aligned} B_1^{(k)} &= B_k B_{k-1} \cdots B_2 B_1 \\ B_2^{(k)} &= B_{2k} B_{2k-1} \cdots B_{k+2} B_{k+1} \\ &\vdots \\ B_{L_k}^{(k)} &= B_L B_{L-1} \cdots B_{(L_k-1)k+1}, \end{aligned}$$

and the vectors $x^{(k)}$ and $b^{(k)}$ are

$$x^{(k)} = \begin{bmatrix} x_k \\ x_{2k} \\ \vdots \\ x_{(L_k-1)k} \\ x_L \end{bmatrix}$$

and

$$b^{(k)} = \begin{bmatrix} b_k + \sum_{t=1}^{k-1} B_k \cdots B_{t+1} b_t \\ b_{2k} + \sum_{t=k+1}^{2k-1} B_{2k} \cdots B_{t+1} b_t \\ \vdots \\ b_{(L_k-1)k} + \sum_{t=(L_k-2)k+1}^{(L_k-1)k-1} B_{(L_k-1)k} \cdots B_{t+1} b_t \\ b_L + \sum_{t=(L_k-1)k+1}^{L-1} B_L \cdots B_{t+1} b_t \end{bmatrix}.$$

After the solution vector $x^{(k)}$ of the reduced system (3.3) is computed, the rest of block components x_i of the solution vector x are obtained by forward and back substitutions as shown in the following:

```
Forward and back substitutions
1. Let $\ell = [k, 2k, \cdots, (L_k - 1)k, L]$
2. For $j = 1, 2, \cdots, L_k$
   (a) $x_{\ell(j)} = x_j^{(k)}$
   (b) forward substitution
       For $i = \ell(j-1)+1, \ldots, \ell(j-1) + \lceil \frac{1}{2}(\ell(j) - \ell(j-1) - 1) \rceil$ with $\ell(0) = 0$:
           If $i = 1$, $x_1 = b_1 - B_1 x_L$
           else $x_i = b_i + B_i x_{i-1}$
   (c) back substitution
       For $i = \ell(j)-1, \ell(j)-2, \ldots, \ell(j) - \lfloor \frac{1}{2}(\ell(j) - \ell(j-1) - 1) \rfloor$,
           $x_i = B_{i+1}^{-1}(x_{i+1} - b_{i+1})$.
```

Remark 3.1. If the reduction factor $k = L$, then $L_k = 1$. The reduced system is

$$M^{(L)} x_L = b^{(L)},$$

i.e,

$$(I + B_L B_{L-1} \cdots B_1) x_L = b_L + \sum_{\ell=1}^{L-1} (B_L B_{L-1} \cdots B_{\ell+1}) b_\ell.$$

Remark 3.2. There are a number of ways to derive the reduced system (3.3). For example, we can use the block Gaussian elimination. Writing the matrix M of the original system (3.1) as a L_k by L_k block matrix:

$$M = \begin{bmatrix} D_1 & & & & \widehat{B}_1 \\ -\widehat{B}_2 & D_2 & & & \\ & -\widehat{B}_3 & D_3 & & \\ & & \ddots & \ddots & \\ & & & -\widehat{B}_{L_k} & D_{L_k} \end{bmatrix},$$

where D_i are $k \times k$ black matrices defined as

$$D_i = \begin{bmatrix} I & & & & \\ -B_{(i-1)k+2} & I & & & \\ & -B_{(i-1)k+3} & I & & \\ & & \ddots & \ddots & \\ & & & -B_{ik} & I \end{bmatrix},$$

and $\widehat{B}_i$ are $k \times k$ block matrices defined as

$$\widehat{B}_i = \begin{bmatrix} 0 & 0 & \cdots & B_{(i-1)k+1} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Define $\widehat{D} = \mathrm{diag}(D_1, D_2, \cdots, D_{L_k})$, then

$$\widehat{D}^{-1} M = \begin{bmatrix} I & & & & D_1^{-1}\widehat{B}_1 \\ -D_2^{-1}\widehat{B}_2 & I & & & \\ & -D_3^{-1}\widehat{B}_3 & I & & \\ & & \ddots & \ddots & \\ & & & -D_{L_k}^{-1}\widehat{B}_{L_k} & I \end{bmatrix},$$

where the matrix $D_i^{-1}\widehat{B}_i$ is given by

$$D_i^{-1}\widehat{B}_i = \begin{bmatrix} 0 & 0 & \cdots & B_{(i-1)k+2}B_{(i-1)k+1} \\ 0 & 0 & \cdots & B_{(i-1)k+3}B_{(i-1)k+2}B_{(i-1)k+1} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & B_{(i-1)k+k}\cdots B_{(i-1)k+2}B_{(i-1)k+1} \end{bmatrix}.$$

Therefore, we immediately see that $M^{(k)}$ is a submatrix of $\widehat{D}^{-1}M$, namely, there exists a matrix P, such that

$$M^{(k)} = P^T\widehat{D}^{-1}MP,$$

where the matrix P is $NL \times (NL/k)$ matrix, whose $(i-1)N+1$ to iN columns are the $(ik-1)N+1$ to ikN columns of the identity matrix I_{NL}.

3.2 Block structural orthogonal factorization

Comparing with the Gaussian elimination, the block structural orthogonal factorization (BSOF) method presented in this section is computationally more expensive, but numerically backward stable.

By multiplying a sequence of orthogonal transformation matrices Q_i, the block cyclic matrix M of the system (3.1) is transformed to a block upper triangular matrix R:

$$Q_{L-1}^T \cdots Q_2^T Q_1^T M = R, \tag{3.4}$$

where

$$R = \begin{bmatrix} R_{11} & R_{12} & & & R_{1L} \\ & R_{22} & R_{23} & & R_{2L} \\ & & \ddots & \ddots & \vdots \\ & & & R_{L-1,L-1} & R_{L-1,L} \\ & & & & R_{LL} \end{bmatrix},$$

and diagonal blocks $R_{\ell\ell}$ are upper triangular, Q_ℓ are orthogonal matrices of the form

$$Q_\ell = \begin{bmatrix} I & & & & \\ & \ddots & & & \\ & & Q_{11}^{(\ell)} & Q_{12}^{(\ell)} & & \\ & & Q_{21}^{(\ell)} & Q_{22}^{(\ell)} & & \\ & & & & \ddots & \\ & & & & & I \end{bmatrix}.$$

The subblocks $Q_{ij}^{(\ell)}$ are defined by the orthogonal factor of the QR decomposition:

$$\begin{bmatrix} \widetilde{M}_{\ell\ell} \\ -B_{\ell+1} \end{bmatrix} = \begin{bmatrix} Q_{11}^{(\ell)} & Q_{12}^{(\ell)} \\ Q_{21}^{(\ell)} & Q_{22}^{(\ell)} \end{bmatrix} \begin{bmatrix} R_{\ell\ell} \\ 0 \end{bmatrix},$$

where

$$\widetilde{M}_{\ell\ell} = \begin{cases} I, & \text{for } \ell = 1 \\ (Q_{22}^{(\ell-1)})^T, & \text{for } \ell = 2, 3, \ldots, L-2, \end{cases}$$

except for $\ell = L-1$, it is defined by the QR decomposition:

$$\begin{bmatrix} \widetilde{M}_{L-1,L-1} & R_{L-1,L} \\ -B_L & I \end{bmatrix} = \begin{bmatrix} Q_{11}^{(L-1)} & Q_{12}^{(L-1)} \\ Q_{21}^{(L-1)} & Q_{22}^{(L-1)} \end{bmatrix} \begin{bmatrix} R_{L-1,L-1} & \check{R}_{L-1,L} \\ 0 & R_{LL} \end{bmatrix}.$$

The following is a pseudo-code for the BSOF method to solve the block cyclic system (3.1).

```
BSOF METHOD
1. Set $M_{11} = I, R_{1L} = B_1$ and $c_1 = b_1$
2. For $\ell = 1, 2, \cdots, L-2$
   (a) Compute the QR decomposition
       $$\begin{bmatrix} M_{\ell\ell} \\ -B_{\ell+1} \end{bmatrix} = \begin{bmatrix} Q_{11}^{(\ell)} & Q_{12}^{(\ell)} \\ Q_{21}^{(\ell)} & Q_{22}^{(\ell)} \end{bmatrix} \begin{bmatrix} R_{\ell\ell} \\ 0 \end{bmatrix}$$
   (b) Set $\begin{bmatrix} R_{\ell,\ell+1} \\ M_{\ell+1,\ell+1} \end{bmatrix} = \begin{bmatrix} Q_{11}^{(\ell)} & Q_{12}^{(\ell)} \\ Q_{21}^{(\ell)} & Q_{22}^{(\ell)} \end{bmatrix}^T \begin{bmatrix} 0 \\ I \end{bmatrix}$
   (c) Update $\begin{bmatrix} R_{\ell L} \\ R_{\ell+1,L} \end{bmatrix} := \begin{bmatrix} Q_{11}^{(\ell)} & Q_{12}^{(\ell)} \\ Q_{21}^{(\ell)} & Q_{22}^{(\ell)} \end{bmatrix}^T \begin{bmatrix} R_{\ell L} \\ 0 \end{bmatrix}$
   (d) Set $\begin{bmatrix} c_\ell \\ c_{\ell+1} \end{bmatrix} := \begin{bmatrix} Q_{11}^{(\ell)} & Q_{12}^{(\ell)} \\ Q_{21}^{(\ell)} & Q_{22}^{(\ell)} \end{bmatrix}^T \begin{bmatrix} c_\ell \\ b_{\ell+1} \end{bmatrix}$
3. Compute the QR decomposition
   $$\begin{bmatrix} M_{L-1,L-1} & R_{L-1,L} \\ -B_L & I \end{bmatrix} = \begin{bmatrix} Q_{11}^{(L-1)} & Q_{12}^{(L-1)} \\ Q_{21}^{(L-1)} & Q_{22}^{(L-1)} \end{bmatrix} \begin{bmatrix} R_{L-1,L-1} & R_{L-1,L} \\ 0 & R_{LL} \end{bmatrix}.$$
4. Set $\begin{bmatrix} c_{L-1} \\ c_L \end{bmatrix} := \begin{bmatrix} Q_{11}^{(L-1)} & Q_{12}^{(L-1)} \\ Q_{21}^{(L-1)} & Q_{22}^{(L-1)} \end{bmatrix}^T \begin{bmatrix} c_{L-1} \\ b_L \end{bmatrix}$
5. Back substitution to solve the block triangular system
   $Rx = c$
```

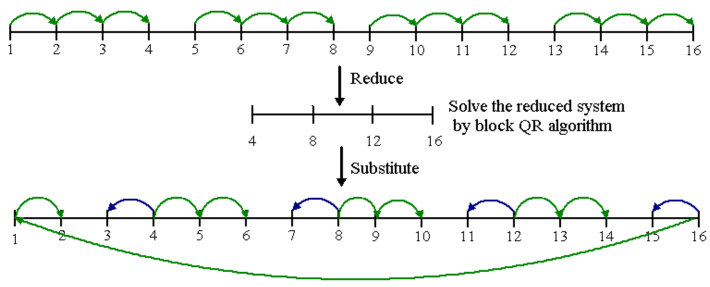

Figure 3.1: A schematic map of the hybrid method.

(a) Solve $R_{LL}x_L = c_L$ for x_L.
(b) Solve $R_{L-1,L-1}x_{L-1} = c_{L-1} - R_{L-1,L}x_L$ for x_{L-1}.
(c) For $\ell = L-2, L-3, \ldots, 1$,
solve $R_{\ell\ell}x_\ell = c_\ell - R_{\ell,\ell+1}x_{\ell+1} - R_{\ell L}x_L$ for x_ℓ.

Floating point operations of the BSOF method is about $15N^3L$. The memory requirement is $3N^2L$.

3.3 A hybrid method

To take advantages of block order reduction of the BCR method and numerical stability of the BSOF method, we propose a hybrid method:

Step 1. Perform a factor-of-k BCR of the original system (3.1) to derive a reduced block cyclic system (3.3) of the block order L/k.

Step 2. Solve the reduced block cyclic system (3.3) by using the BSOF method.

Step 3. Forward and back substitute to find the rest of block components x_i of the solution x:

$$\{x_i\} \quad \longleftarrow \quad x^{(k)} \quad \longrightarrow \quad \{x_j\}.$$

Figure 3.1 is a schematic map of the hybrid method for a 16-block cyclic system with a reduction factor $k = 4$. We use both forward and back substitutions to minimize the propagation of rounding errors induced at Steps 1 and 2.

By Step 1, the order of the original M is reduced by a factor of k. Consequently, the computational cost of the BSOF method at Step 2 is reduced to $O(N^3\frac{L}{k})$, a factor of k speedup. Therefore, the larger k is, the better CPU performance is. However, the condition number of $M^{(k)}$

increases as k increases, see the analysis presented in section 2.6. As a result, the accuracy of the computed solution decreases. An interesting question is how to find a reduction factor k, such that the computed solution meets the required accuracy in QMC simulations. In practice, such a reduction factor k should be determined in a *self-adapting* fashion with respect to the changes of underlying multi-length and energy scales in QMC simulations. This is discussed in the next section.

3.4 Self-adaptive reduction factor k

Let us turn to the question of determining the reduction factor k for the BCR step of the proposed hybrid method. Since the BSOF method is backward stable, by well-established error analysis of the linear system, for example, see [24, p.120], we know that the relative error of the computed solution $\widehat{x}^{(k)}$ of the reduced system (3.3) is bounded by $\kappa(M^{(k)})\epsilon$, i.e.,

$$\frac{\|\delta x^{(k)}\|}{\|x^{(k)}\|} \equiv \frac{\|x^{(k)} - \widehat{x}^{(k)}\|}{\|x^{(k)}\|} \leq \kappa(M^{(k)})\epsilon, \tag{3.5}$$

where ϵ is the machine precision. For the clarity of notation, we only use the first-order upper bound and ignore the small constant coefficient.

To consider the propagation of the errors in the computed solution $\widehat{x}^{(k)}$ in the back and forward substitutions, let us start with the computed L-th block component $\widehat{x}_L$ of the solution vector x,

$$\widehat{x}_L = x_L + \delta x_L,$$

where δx_L is the computed error, with a related upper bound defined in (3.5). By the forward substitution, the computed first block component $\widehat{x}_1$ of the solution x satisfies

$$\begin{aligned} \widehat{x}_1 &= b_1 - B_1 \widehat{x}_L \\ &= b_1 - B_1(x_L + \delta x_L) \\ &= b_1 - B_1 x_L + B_1 \delta x_L \\ &= x_1 + \delta x_1, \end{aligned}$$

where $\delta x_1 = B_1 \delta x_L$ is the error propagated by the error in $\widehat{x}_L$. Note that the computational error induced in the substitution is ignored, since it is generally much smaller than the the error in $\widehat{x}_L$.

By the relative error bound (3.5) of δx_L, it yields that the error in the computed $\widehat{x}_1$ could be amplified by the factor $\|B_1\|$, namely,

$$\frac{\|\delta x_1\|}{\|x_1\|} \leq \|B_1\| \kappa(M^{(k)})\, \epsilon.$$

Subsequently, the relative error of the computed $\widehat{x}_2$ obtained by the forward substitution from x_1 is bounded by

$$\frac{\|\delta x_2\|}{\|x_2\|} \leq \|B_2\|\|B_1\|\kappa(M^{(k)})\,\epsilon.$$

This process can be continued to bound the errors of δx_3 and so on until all that is left is $x_{\frac{k}{2}}$. The related error bound of computed $x_{\frac{k}{2}}$ is given by

$$\frac{\|\delta x_{\frac{k}{2}}\|}{\|x_{\frac{k}{2}}\|} \leq \|B_{\frac{k}{2}}\|\cdots\|B_2\|\,\|B_1\|\kappa(M^{(k)})\,\epsilon.$$

By analogous calculations for the rest of substitutions, we conclude that for any computed block component $\widehat{x}_\ell$ of the solution x, where $\ell = 1, 2, \ldots, L$, the related error is bounded by

$$\begin{aligned}\frac{\|\delta x_\ell\|}{\|x_\ell\|} &\leq \|B_{\frac{k}{2}}\|\cdots\|B_2\|\,\|B_1\|\kappa(M^{(k)})\,\epsilon \\ &\leq c\,e^{\frac{1}{2}k(4t\Delta\tau+\nu)}\cdot e^{k(4t\Delta\tau+\nu)}\kappa(M)\,\epsilon \\ &= c\,e^{\frac{3}{2}k(4t\Delta\tau+\nu)}\kappa(M)\,\epsilon, \end{aligned} \tag{3.6}$$

where for the second inequality we have used the upper bounds (2.15) and (2.23) for the norm of B_ℓ and the condition number of the matrix $M^{(k)}$.

Assume that the desired relative accuracy of the computed solution x is "tol", i.e.,

$$\frac{\|\delta x\|}{\|x\|} \leq \text{tol}. \tag{3.7}$$

Then by inequalities (3.6) and (3.7), a plausible choice of the reduction factor k is

$$k = \left\lfloor \frac{\frac{2}{3}\ln(\text{tol}/\epsilon)}{4t\Delta\tau+\nu} \right\rfloor. \tag{3.8}$$

To balance the number of the matrices B_ℓ in the product $B_\ell^{(k)}$, after k is computed, the final k is then slightly adjusted by $k = \left\lceil \frac{L}{L_k} \right\rceil$, where $L_k = \left\lceil \frac{L}{k} \right\rceil$.

We note that the factor of $\ln\kappa(M)$ is dropped in deciding reduction factor k. The reason is that as we discussed in section 2.5, $\kappa(M)$ grows slowly in the range of parameters of interest, $\ln\kappa(M)$ is small.

By expression (3.8), the reduction factor k is determined in a *self-adaptive* fashion. When energy parameters U and $\beta = L\cdot\Delta\tau$ change, k is determined adaptively to achieve the desired accuracy "tol". For example, let $t = 1$ and $\Delta\tau = \frac{1}{8}$, if the desired accuracy threshold is tol $= 10^{-8}$, then with double precision arithmetic and machine precision $\epsilon =$

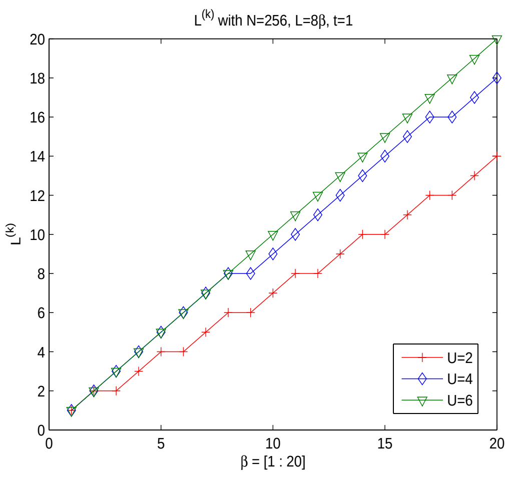

Figure 3.2: Reduced block size L_k

10^{-16}, the reduction factor k with respect to different energy parameter U is shown in the following table:

U	0	1	2	3	4	5	6
k	24	14	12	10	9	9	8

Figure 3.2 shows the reduced block sizes L_k with respect to different values of U, where $L = 8\beta$, $\beta = 1, 2, \dots, 20$ and $t = 1$ The accuracy is set to be half of the machine precision, i.e., tol $= \sqrt{\epsilon} = 10^{-8}$.

3.5 Self-adaptive block cyclic reduction method

In summary, the self-adaptive block cyclic reduction method, SABCR in short, to solve the linear system (3.1) may be condensed as the following:

```
SABCR METHOD
1. Determine the reduction factor $k$ by (3.8)
2. Reduce $(M, b)$ to $(M^{(k)}, b^{(k)})$ by the BCR
3. Solve the reduced system $M^{(k)}x^{(k)} = b^{(k)}$ for $x^{(k)}$ by
   the BSOF method
4. Use forward and back substitutions to compute the re-
   maining block components $x_\ell$ of $x$
```

3.6 Numerical experiments

In this section, we present numerical results of the SABCR method. SABCR is implemented in Fortran 90 using LAPACK and BLAS. All numerical results are performed on an HP Itanium2 workstation with 1.5GHZ CPU and 2GB core memory. The threshold of desired relative accuracy of computed solution $\widehat{x}$ is set at the order of $\sqrt{\epsilon}$, namely,

$$\frac{\|\widehat{x} - x\|}{\|x\|} \leq \text{tol} = 10^{-8}.$$

Example 1. In this example, we examine numerical accuracy and performance of the SABCR method when $U = 0$. The other parameters of the coefficient matrix M is set as $N = 16\times16$, $L = 8\beta$ for $\beta = 1, 2, \ldots, 20$, $t = 1$, $\Delta\tau = \frac{1}{8}$. Figure 3.3 shows the relative errors of computed solutions $\widehat{x}$ by the BSOF and SABCR methods. As we see, the BSOF method is of full machine precision $O(10^{-15})$. It indicates that the underlying linear system is well-conditioned and the BSOF method is backward stable. On the other hand, as shown in the figure, the relative errors of the SABCR method are at $O(10^{-8})$ as desired.

Table 1 shows the reduction factor k and the reduced block size L_k with respect to the inverse temperature β, CPU elapsed time and speedups of the SABCR method comparing to the BSOF method. Note that for when $\beta \leq 3$, the reduction factor $k = L$ and the number of reduced block $L_k = 1$. As β increases, the reduction factor k decreases. For example, when $\beta = 20$, the reduction factor $k = 23$ and the number of reduced blocks $L_k = \lfloor \frac{160}{23} \rfloor + 1 = 7$.

Example 2. In this example, we examine numerical accuracy and performance of the SABCR method for $U = 2, 4, 6$. The other parameters of the coefficient matrix M are $N = 16 \times 16 = 256$, $L = 8\beta$ with $\beta = 1, 2, \ldots, 20$. $t = 1$ and $\Delta\tau = \frac{1}{8}$. Figure 3.4 shows that the relative errors of the computed solution $\widehat{x}$ are under $\text{tol} = 10^{-8}$ as required. Table 2 shows the the reduced block size L_k, CPU elapsed time of the BSOF and SABCR methods with respect to U and β. The speedups of SABCR are shown in Figure 3.5.

Example 3. In this example, we examine computational efficiency of the SABCR solver with respect to the number of time slices $L = \beta/\Delta\tau$ with $\beta = 1, 2, \ldots, 20$ and $\Delta\tau = \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$. The dimensions of the coefficient matrices M vary from $NL = 2,048$ $(\beta = 1, \Delta\tau = \frac{1}{8})$ to $NL = 163,840$ $(\beta = 20, \Delta\tau = \frac{1}{32})$. The other parameters are $N = 16 \times 16$, $U = 6$ and $t = 1$.

Table 3 shows the the reduced block size L_k with respect to β and $\Delta\tau$ $(L = \beta\Delta\tau)$, CPU elapsed time in seconds of the BSOF and SABCR methods. Speedups are plotted in Figure 3.6.

For large energy scale parameters t, β and U, small $\Delta\tau$ is necessary for the accuracy of the Trotter-Suzuki decomposition. Small $\Delta\tau$ implies

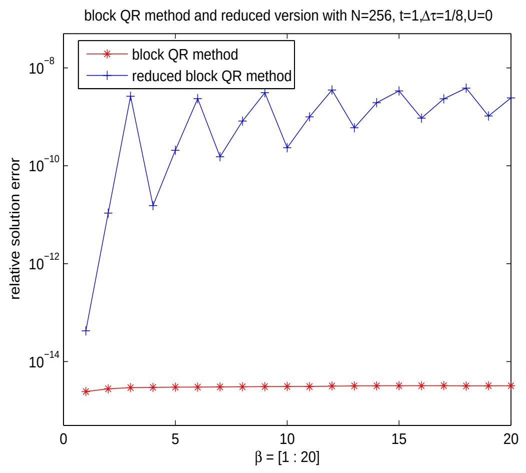

Figure 3.3: Relative errors, $U = 0$

Table 1: Performance data of BSOF and SABCR methods, $U = 0$

β	$L = 8\beta$	k	L_k	BSOF	SABCR	speedup
1	8	8	1	3.19	0.0293	108
2	16	16	1	7.06	0.042	168
3	24	24	1	10.8	0.0547	197
4	32	16	2	14.6	0.303	48
5	40	20	2	18.6	0.326	57
6	48	24	2	23.1	0.342	67
7	56	19	3	27.2	0.666	40
8	64	22	3	31.3	0.683	45
9	72	24	3	35.1	0.675	52
10	80	20	4	38.0	1.18	32
11	88	22	4	42.0	1.18	35
12	96	24	4	46.0	1.20	38
13	104	21	5	49.9	1.28	38
14	112	23	5	54.0	1.28	42
15	120	24	5	58.2	1.32	44
16	128	22	6	62.9	1.67	37
17	136	23	6	68.3	1.72	39
18	144	24	6	73.2	1.73	42
19	152	22	7	75.3	1.98	38
20	160	23	7	80.2	2.02	39

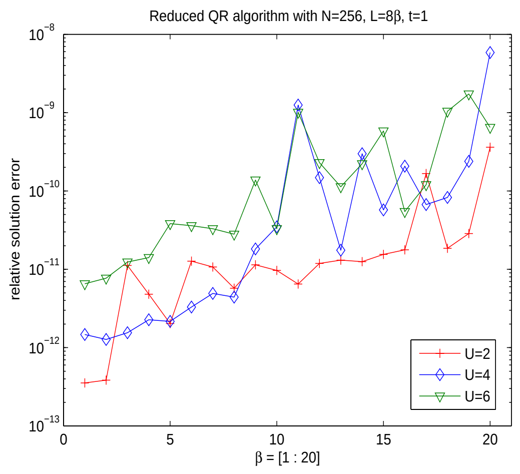

Figure 3.4: Relative errors, $U = 2, 4, 6$

large $L = \frac{\beta}{\Delta\tau}$. For the SABCR solver, small $\Delta\tau$ implies a large reduction factor k. The SABCR is more efficient for small $\Delta\tau$.

Example 4. Let us examine the memory limit with respect to the increase of the lattice size parameter N. The memory requirement of the BSOF method is $3N^2L = 3N_x^4L$. If $N_x = N_y = 32$, the memory storage of one $N \times N$ matrix is 8MB. Therefore for a 1.5GB memory machine, the number L of time slices is limited to $L < 63$. It implies that when $\Delta\tau = \frac{1}{8}$, the inverse temperature β must be smaller than 8. The BSOF method will run out of memory when $\beta \geq 8$. On the other hand, the SABCR solver should continue work for $L = 8\beta$ and $\beta = 1, 2, \ldots, 10$. Table 4 shows the memory limitation of the BSOF method, where $t = 1$ and $U = 6$.

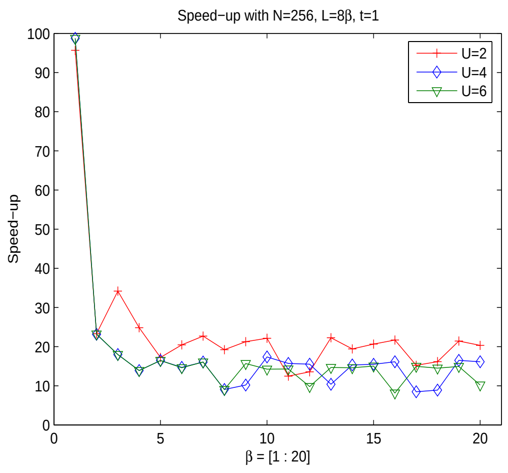

Figure 3.5: Speedups of SABCR for $U = 2, 4, 6$

Table 2: Performance data of BSOF and SABCR methods, $U = 2, 4, 6$

	$U = 2$			$U = 4$			$U = 6$		
β	L_k	BSOF	SABCR	L_k	BSOF	SABCR	L_k	BSOF	SABCR
1	1	3.08	0.0322	1	3.08	0.0312	1	3.08	0.0312
2	2	6.83	0.293	2	6.83	0.295	2	6.83	0.294
3	2	10.7	0.313	3	10.7	0.595	3	10.7	0.595
4	3	14.6	0.588	4	14.6	1.05	4	14.6	1.05
5	4	18.3	1.06	5	18.3	1.11	5	18.3	1.11
6	4	22.1	1.08	6	22.1	1.50	6	22.1	1.51
7	5	25.9	1.14	7	25.9	1.61	7	25.9	1.61
8	6	29.6	1.54	8	29.6	3.26	8	29.6	3.26
9	6	33.4	1.57	8	33.4	3.27	9	33.4	2.13
10	7	37.2	1.68	9	37.2	2.14	10	37.2	2.61
11	8	41.2	3.30	10	41.2	2.62	11	41.2	2.88
12	8	45.0	3.31	11	45.0	2.90	12	45.0	4.59
13	9	48.8	2.19	12	48.8	4.69	13	48.9	3.33
14	10	52.6	2.71	13	52.6	3.43	14	52.6	3.60
15	10	56.4	2.73	14	56.4	3.64	15	56.4	3.75
16	11	60.5	2.79	15	60.5	3.75	16	60.5	7.40
17	12	64.2	4.21	16	64.2	7.55	17	64.2	4.29
18	12	67.9	4.20	16	67.9	7.61	18	67.9	4.68
19	13	71.8	3.35	17	71.8	4.35	19	71.9	4.81
20	14	75.7	3.72	18	75.7	4.69	20	75.7	7.42

Table 3: Performance data of BSOF and SABCR methods, $L = \beta/\Delta\tau$

$\Delta\tau$	1/8			1/16			1/32		
β	L_k	BSOF	SABCR	L_k	BSOF	SABCR	L_k	BSOF	SABCR
1	1	3.25	0.0293	2	7.25	0.306	2	15.5	0.34
2	2	7.28	0.305	3	15.1	0.596	4	32.9	1.15
3	3	11.2	0.605	4	23.0	1.11	5	47.3	1.36
4	4	15.1	1.10	5	32.0	1.27	7	63.6	1.97
5	5	19.2	1.23	7	39.1	1.85	8	80.3	3.58
6	6	23.0	1.62	8	47.2	3.43	10	97.9	3.03
7	7	27.2	1.87	9	55.4	2.47	11	112	3.54
8	8	32.1	3.38	10	63.4	2.93	13	140	3.95
9	9	35.3	2.38	12	71.1	4.26	14	150	4.57
10	10	39.1	2.86	13	79.3	3.91	16	167	8.06
11	11	43.2	3.08	14	87.6	4.39	17	180	5.40
12	12	47.2	4.39	15	95.7	4.50	19	196	6.00
13	13	51.7	3.71	16	103	8.00	20	209	7.99
14	14	55.3	4.06	18	112	5.61	22	224	7.03
15	15	59.2	4.26	19	120	5.64	23	240	7.05
16	16	63.5	7.54	20	128	7.58	25	258	7.83
17	17	67.3	4.92	21	136	6.23	26	273	8.42
18	18	71.2	5.78	23	144	6.88	28	290	11.2
19	19	75.3	5.58	24	152	12.0	29	305	9.03
20	20	79.3	7.36	15	160	7.49	31	321	9.60

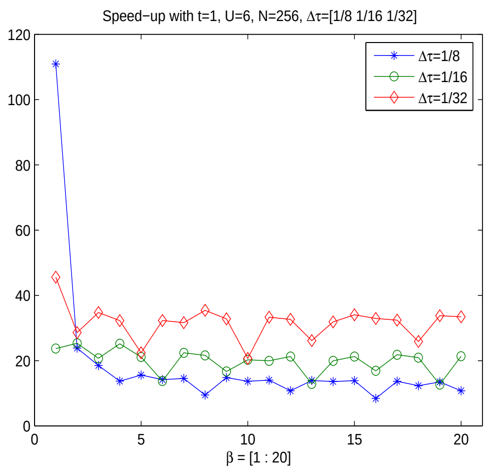

Figure 3.6: Speedups of SABCR for $L = \beta/\Delta\tau$

Table 4: SABCR for large systems, $N = 32 \times 32$

β	L	k	L_k	BSOF(sec.)	SABCR (sec.)	Speedup ($\times$)
1	8	8	1	148.00	2.10	70
2	16	8	2	322.00	17.8	18
3	24	8	3	509.00	40.1	12.7
4	32	8	4	689.00	64.5	10.6
5	40	8	5	875.00	88.6	9.8
6	48	8	6	1060.00	110.00	9.6
7	56	8	7	1250.00	131.00	9.5
8	64	8	8	out of memory	150.00	
9	72	8	9	out of memory	172.00	
10	80	8	10	out of memory	200.00	

4 Preconditioned iterative linear solvers

As discussed in sections 1 and 2, one of the computational kernels of the hybrid quantum Monte Carlo (HQMC) simulation is to repeatedly solve the linear system of equations

$$Ax = b, \tag{4.1}$$

where A is a symmetric positive definite matrix of the form

$$A = M^T M$$

and M is the Hubbard matrix as defined in (2.1).

One can solve the linear system (4.1) by solving the coupled systems $M^T y = b$ for y and $Mx = y$ for x using the SABCR method described in section 3. However, the computational complexity will be $O(N^3 L/k)$, which is prohibitive for large lattice size N. In this section, we consider preconditioned iterative solvers. It is our goal to develop an efficient a preconditioned iterative solver that exhibits an optimal linear-scaling, namely, the computational complexity scales linearly with respect to the lattice size N. At the end of this section, we will see that so far, we are only able to achieve the linear-scaling for moderately interacting systems, namely U is small.

4.1 Iterative solvers and preconditioning

We have conducted some numerical study of applying GMRES, QMR and Bi-CGSTAB methods to solve the p-cyclic system

$$Mx = b.$$

We observed that these methods suffer from slow convergence rates or erratic convergence behaviors. Figures 4.1 and 4.2 show the typical convergence behaviors of these methods. The parameters of the matrices M are set as $(N, L, U, t, \beta, \mu) = (8 \times 8, 24, 4, 1, 3, 0)$ and $h = \pm 1$ with equal probability, and the entries of the right-hand-side vector b are random numbers chosen from a uniform distribution on the interval $(0, 1)$.

Although the convergence of conjugate gradient (CG) method to solve the symmetric positive definite system (4.1) is slow but robust in the sense that residual error decreases steadily as shown in Figure 4.3.

As it is well known, the convergence rate of CG could be improved dramatically by using a proper preconditioner R, which symmetrically preconditions the system (4.1):

$$R^{-1} A R^{-T} \cdot R^T x = R^{-1} b. \tag{4.2}$$

An ideal preconditioner R satisfies the following three conditions:

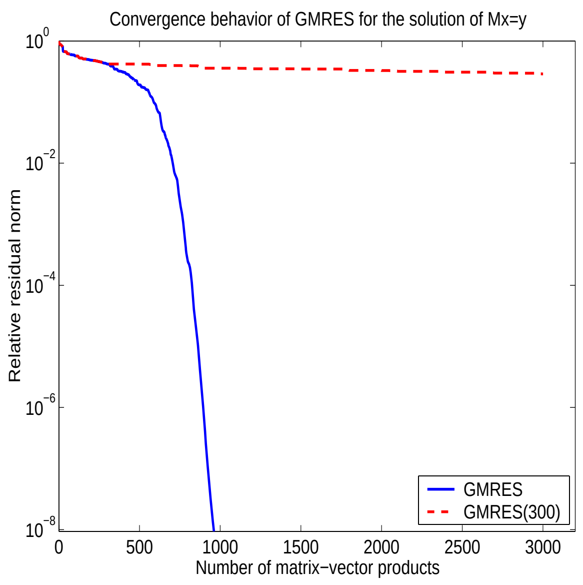

Figure 4.1: Relative residual norms of GMRES and GMRES(300).

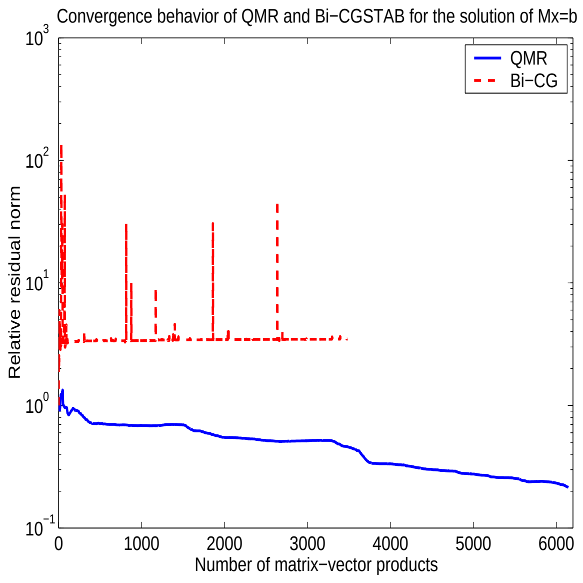

Figure 4.2: Relative residual norms of Bi-CGSTAB and QMR.

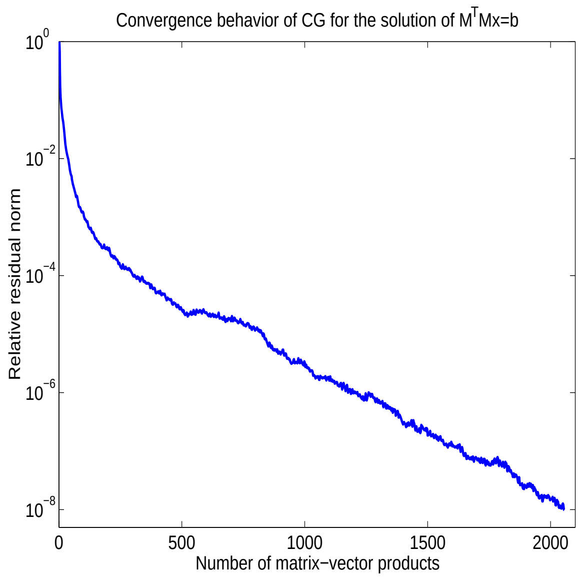

Figure 4.3: Relative residual norms of CG.

1) The cost of constructing R is cheap.
2) The application of R, i.e., solving $Rz = r$ for z, is not expensive.
3) RR^T is a good approximation of A.

However, in practice, there is a trade-off between the costs 1) and 2) and the quality 3). In this section, we focus on the development of robust and efficient preconditioning techniques for an optimal balance between costs and quality.

For all numerical results presented in this section, Hubbard matrices M are generated with the Hubbard-Stratonovich configurations $h_{\ell,i}$ such that the average condition numbers of the resulting test matrices are close to the ones arising in a full HQMC simulation. The right-hand-side vector b is set so that entries of the (exact) solution vector x are uniformly distributed on the interval $(0, 1)$. The required accuracy of the computed solution $\widehat{x}$ is set as

$$\frac{\|x - \widehat{x}\|_2}{\|x\|_2} \leq 10^{-3}.$$

This is sufficient for the HQMC simulation.

All preconditioning algorithms presented in this section are implemented in Fortran 90. The numerical performance data are collected

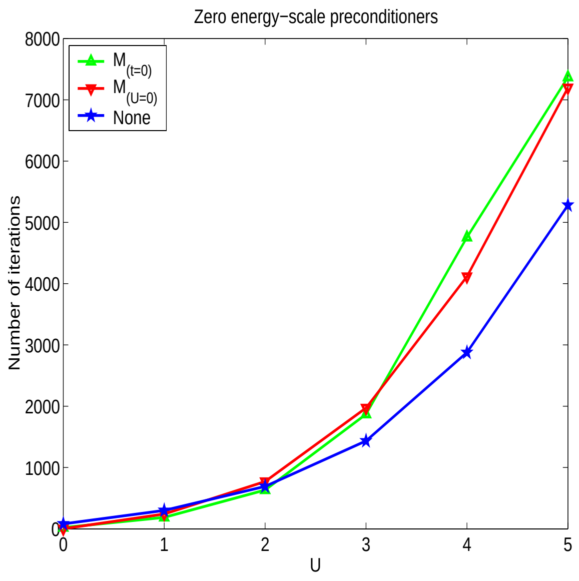

Figure 4.4: Number of PCG iterations using preconditioners $R = M_{(U=0)}$ and $R = M_{(t=0)}$.

on an HP Itanium2 workstation with 1.5GHz CPU and 2GB of main memory. Intel Math Kernel Library (MKL) 7.2.1 and `-O3` optimization option in `ifort` are used to compile the codes.

4.2 Previous work

There are a number of studies on preconditioning techniques to improve the convergence rate of PCG solver for the QMC simulations. One attempt is to precondition the system with $R = M_{(U=0)}$ [34, 49]. By using the Fast Fourier Transform, the computational cost of applying this preconditioner is $O(NL\log(NL))$. However, the quality of the preconditioner is poor for moderately and strongly interacting systems ($U \geq 3$), as shown in Figure 4.4. The results are the averages of 50 solutions of the systems $(N, L, t, \beta, \mu) = (8 \times 8, 40, 1, 5, 0)$.

It is suggested to use the preconditioner $R = M_{(t=0)}$ [34]. In this case, the submatrices B_ℓ are diagonal. The cost of applying the preconditioner R is $O(NL)$. However, the quality is poor, particularly for strongly interacting systems, as shown in Figure 4.4.

Jacobi preconditioner $R = \mathrm{diag}(a_{ii}^{1/2})$ is used in [42], where a_{ii} are the diagonal elements of A. The PCG convergence rate is improved consistently as shown in Figure 4.5. However, this is still insufficient for

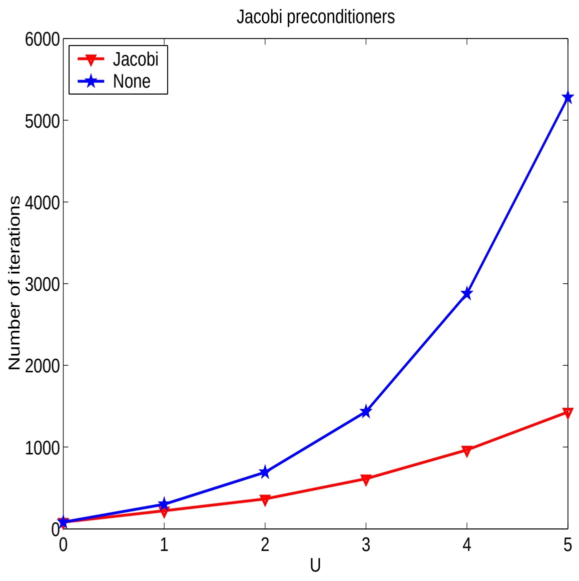

Figure 4.5: Number of PCG iterations using Jacobi preconditioner R

the full HQMC simulation. For example, for a small and moderately-interacting system $(N, L, U, t, \beta) = (8 \times 8, 40, 4, 1, 5)$, Jacobi-based PCG solver requires $3,569$ iterations and 1.78 seconds. A full HQMC simulation typically requires to solve $10,000$ linear systems. By this rate, a full HQMC simulation of 8×8, would take 4.9 hours. When N is increased to 32×32, the PCG takes $10,819$ iterations and 87.80 seconds for solving one system. By this rate, a full HQMC simulation of a 32×32 lattice would need more than 20 days.

It is proposed to use an incomplete Cholesky (IC) preconditioner R, where R is lower triangular and has the same block structure of A [49]. Although the PCG convergence rate is improved considerably, it becomes impractical due to the cost of $O(N^2 L)$ in storing and applying the IC preconditioner. Furthermore, it is not robust and suffers breakdown for strongly interacting systems as we will see in section 4.4.

4.3 Cholesky factorization

We begin with a review of the Cholesky factorization of an $n \times n$ symmetric positive definite (SPD) matrix A:

$$A = RR^T, \tag{4.3}$$

where R is lower-triangular with positive diagonal entries. R is referred to as the Cholesky factor.

We follow the presentation in [38]. The Cholesky factorization (4.3) can be computed by using the following partition and factorization:

$$A = \begin{bmatrix} a_{11} & \widehat{a}^T \\ \widehat{a} & A_1 \end{bmatrix} = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A_1 - \widehat{r}\widehat{r}^T \end{bmatrix} \begin{bmatrix} r_{11} & \widehat{r}^T \\ 0 & I \end{bmatrix}. \tag{4.4}$$

By the first columns of the both sides of the factorization, we have

$$\begin{aligned} a_{11} &= r_{11}^2, \\ \widehat{a} &= \widehat{r}\, r_{11}. \end{aligned}$$

Therefore,

$$\begin{aligned} r_{11} &= \sqrt{a_{11}}, \\ \widehat{r} &= \frac{1}{r_{11}}\widehat{a}. \end{aligned}$$

If we have the Cholesky factorization of the $(n-1) \times (n-1)$ matrix $A_1 - \widehat{r}\widehat{r}^T$:

$$A_1 - \widehat{r}\widehat{r}^T = R_1 R_1^T, \tag{4.5}$$

then the Cholesky factor R is given by

$$R = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & R_1 \end{bmatrix}.$$

Hence the Cholesky factorization can be obtained through the repeated applications of (4.4) on (4.5). The resulting algorithm is referred to as a *right-looking* Cholesky algorithm, since after the first column r_1 of R is computed, it is used to update the matrix A_1 to compute the remaining columns of R, which are on the right side of r_1.

There is a left-looking version of the Cholesky algorithm. By comparing the jth column of the factorization (4.3), we have

$$a_j = \sum_{k=1}^{j} r_{jk} r_k.$$

This says that

$$r_{jj} r_j = a_j - \sum_{k=1}^{j-1} r_{jk} r_k.$$

Hence, to compute the jth column r_j of R, one first computes

$$v = a_j - \sum_{k=1}^{j-1} r_{jk} r_k,$$

Table 5: Performance of CHOLMOD.

N	8×8	16×16	24×24	32×32	40×40	48×48
P-time	0.08	1.99	17.56	90.68	318.04	offmem
S-time	0.00	0.04	0.29	0.52	1.22	offmem
T-time	0.08	2.03	17.85	91.20	319.26	offmem

and then

$$r_{ij} = \frac{v_i}{\sqrt{v_j}}, \quad \text{for } i = j, j+1, \ldots, n.$$

It is a *left-looking* algorithm since the jth column r_j of R is computed through referencing the computed columns $r_1, r_2, \ldots, r_{j-1}$ of R, which are on the left of r_j.

The Cholesky factorization could fail due to a *pivot breakdown*, namely, at the jth step, the diagonal element $a_{jj} \leq 0$. When A is SPD, the diagonal element a_{11} must be positive. Furthermore, $A_1 - \widehat{r}\,\widehat{r}^T$ is SPD since it is a principal submatrix of the SPD matrix X^TAX, where

$$X = \begin{bmatrix} 1 & -\widehat{r}^T \\ 0 & I \end{bmatrix}.$$

Therefore, there is no pivot breakdown for an SPD matrix.

HQMC application. When the Cholesky factor R of A is used as a preconditioner, the HQMC linear system (4.1) is solved exactly. Table 5 records the CPU time in seconds with respect to different lattice sizes N by using CHOLMOD developed by Timothy A. Davis.[12] CHOLMOD is one of the state-of-art implementations of the sparse Cholesky factorization and is used in MATLAB version 7.2.

The other parameters are set as $(L, U, t, \beta, \mu) = (80, 4, 1, 10, 0)$. We note that, when $N = 48 \times 48$, it runs out of memory ("offmem") before completing the Cholesky factorization of $A = M^TM$. In the table, "P-time" stands for the CPU time for computing the preconditioner, "S-time" for the CPU time for PCG iterations, and "T-time" for the total CPU time. With an approximate minimum degree (AMD) recording of the matrix A, the CPU time was reduced slightly as shown in Table 6.

The Cholesky factorization is not affected by the potential energy parameter U. Therefore, the performance of CHOLMOD is expected to be the same for different U. By an interpolation of the performance data of CHOLMOD with AMD reordering, the computational complexity of the Cholesky factorization is $O(N^2)$. By this rate, if we were able to

[12]http://www.cise.ufl.edu/research/sparse/cholmod/

Table 6: Performance of CHOLMOD with AMD recording.

N	8×8	16×16	24×24	32×32	40×40	48×48
P-time	0.11	1.63	12.40	57.85	297.27	offmem
S-time	0.00	0.04	0.13	0.34	0.95	offmem
T-time	0.11	1.67	12.53	58.19	298.22	offmem

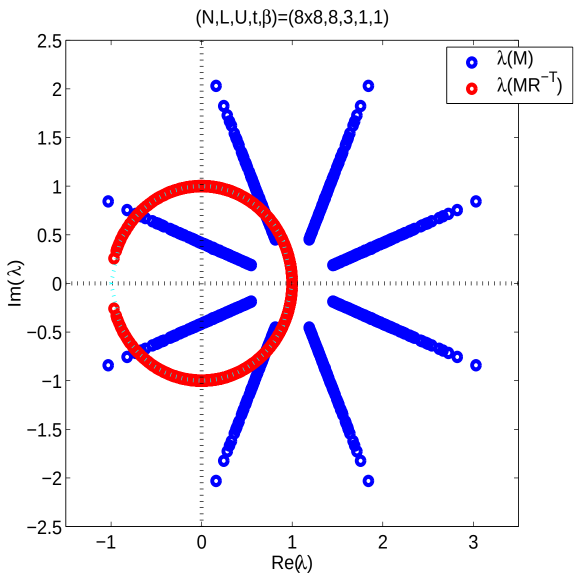

Figure 4.6: Eigenvalues of M and MR^{-1}, where R is a Cholesky factor.

solve the Hubbard system with $NL = 48 \times 48 \times 80 = 184,320$, the CPU elapsed time would take about 700 seconds.

To end this section, we note that with the Cholesky factor R, the preconditioned matrix MR^{-T} becomes orthogonal. The eigenvalues of MR^{-T} are on the unit circle, as shown in Figure 4.6. Therefore, to assess the quality of a preconditioner R, one can examine how close the eigenvalues of the preconditioned matrix MR^{-T} are to the unit circle. Of course, this can be checked only for small matrices.

4.4 Incomplete Cholesky factorizations

4.4.1 IC

To reduce the computational and storage costs of the Cholesky factorization (4.3), a preconditioner R can be constructed based on an incomplete

Cholesky (IC) factorization:

$$A = RR^T + S + S^T, \tag{4.6}$$

where R is lower-triangular and is referred to as an IC factor, and S is a strictly lower-triangular matrix (therefore, the diagonal elements of A and RR^T are the same). $E = S + S^T$ is the error matrix of the incomplete factorization. The sparsity of R is controlled by a set $\mathcal{Z}$, which is a set of ordered pairs (i, j) of integers from $\{1, 2, \ldots, n\}$ such that if $(i, j) \notin \mathcal{Z}$, then $r_{ij} = 0$.

The IC factor R can be computed based on the following partition and factorization:

$$A = \begin{bmatrix} a_{11} & \widehat{a}^T \\ \widehat{a} & A_1 \end{bmatrix} = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A_1 - \widehat{r}\widehat{r}^T \end{bmatrix} \begin{bmatrix} r_{11} & \widehat{r}^T \\ 0 & I \end{bmatrix} + \begin{bmatrix} 0 & \widehat{s}^T \\ \widehat{s} & 0 \end{bmatrix}. \tag{4.7}$$

By multiplying out the first column of the both sides of the factorization, we have

$$\begin{aligned} a_{11} &= r_{11}^2, \\ \widehat{a} &= \widehat{r}\, r_{11} + \widehat{s}. \end{aligned}$$

Therefore, if the pivot $a_{11} > 0$, we have

$$r_{11} = \sqrt{a_{11}}.$$

The vector $\widehat{r}$ and $\widehat{s}$ are computed based on the sparsity set $\mathcal{Z}$:

$$\begin{aligned} &\text{for } i \le n-1, \\ &\begin{cases} \widehat{r}_i = \widehat{a}_i / r_{11},\ \widehat{s}_i = 0, & \text{if } (i+1, 1) \in \mathcal{Z} \\ \widehat{r}_i = 0, \qquad\ \ \widehat{s}_i = \widehat{a}_i, & \text{otherwise.} \end{cases} \end{aligned} \tag{4.8}$$

Therefore, if we have an IC factorization of the $(n-1) \times (n-1)$ matrix $A_1 - \widehat{r}\widehat{r}^T$:

$$A_1 - \widehat{r}\widehat{r}^T = R_1 R_1^T + S_1 + S_1^T, \tag{4.9}$$

then the IC factorization (4.6) of A is given by

$$R = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & R_1 \end{bmatrix} \quad \text{and} \quad S = \begin{bmatrix} 0 & 0 \\ \widehat{s} & S_1 \end{bmatrix}.$$

Thus, the IC factorization can be computed by the repeated application of (4.7) on (4.9) as long as $A - \widehat{r}\widehat{r}^T$ is SPD. Note that when non-zero element $\widehat{a}_i$ is discarded, i.e., $\widehat{r}_i = 0$, in (4.8), the operations to update A_1 with respect to the element $\widehat{r}_i$ in (4.9) are skipped.

The following algorithm computes an IC factor R in the right-looking fashion, where in the first line, R is initialized by the lower-triangular part of A, and the update of A_1 is performed directly on R.

```
    IC (RIGHT-LOOKING VERSION)
 1. $R = \text{lower}(A)$
 2. For $j = 1, 2, \ldots, n$
 3.    $r(j,j) := \sqrt{r(j,j)}$ if $r(j,j) > 0$
 4.    for $i = j+1, j+2, \ldots, n$
 5.       if $(i,j) \in \mathcal{Z}$
 6.          $r(i,j) := r(i,j)/r(j,j)$
 7.       else
 8.          $r(i,j) = 0$
 9.       end if
10.    end for
11.    for $k = j+1, j+2, \ldots, n$
12.       $r(k:n,k) := r(k:n,k) - r(k,j)r(k:n,j)$
13.    end for
14. end for
```

Note that all computational steps of the previous algorithm and the rest of algorithms presented in this section are performed with regard to the sparsity of matrices and vectors involved.

There is an left-looking algorithm. By comparing the jth column of the factorization (4.6), we have

$$a_{jj} = \sum_{k=1}^{j} r_{jk}^2, \tag{4.10}$$

$$a_{ij} = \sum_{k=1}^{j} r_{jk} r_{ik} + s_{ij}, \quad \text{for } i = j+1, \ldots, n. \tag{4.11}$$

This says that

$$\begin{aligned} r_{jj}^2 &= a_{jj} - \sum_{k=1}^{j-1} r_{jk}^2, \\ r_{jj} r_{ij} + s_{ij} &= a_{ij} - \sum_{k=1}^{j-1} r_{jk} r_{ik}, \quad \text{for } i = j+1, \ldots, n. \end{aligned}$$

Thus, to compute the jth column in the IC factorization, one first computes

$$v = a_j - \sum_{k=1}^{j-1} r_{jk} r_k. \tag{4.12}$$

Then, the pivot $v_j > 0$, the jth diagonal entry of R is then given by

$$r_{jj} = \sqrt{v_j},$$

and the rest of non-zero elements of r_j and the discarded elements of s_j are computed based on the sparsity constraint $\mathcal{Z}$:

$$\text{for } i \geq j+1, \quad \begin{cases} r_{ij} = v_i/r_{jj},\ s_{ij} = 0, & (i,j) \in \mathcal{Z}, \\ r_{ij} = 0, \qquad\quad s_{ij} = v_i, & \text{otherwise.} \end{cases}$$

A pseudo-code of the left-looking algorithm is as the following:

```
    IC (LEFT-LOOKING VERSION)
 1. for $j = 1, 2, \ldots, n$
 2.   $v(j:n) = a(j:n, j)$
 3.   for $k = 1, 2, .., j-1$
 4.     $v(j:n) := v(j:n) - r(j,k)r(j:n,k)$
 5.   end for
 6.   $r(j,j) = \sqrt{a(j,j)}$ if $a(j,j) > 0$
 7.   for $i = j+1, j+2, \ldots, n$
 8.     if $(i,j) \in \mathcal{Z}$
 9.       $r(i,j) = v(i)/r(j,j)$
10.     else
11.       $r(i,j) = 0$
12.     end if
13.   end for
14. end for
```

The following rules are often used to define the sparsity set $\mathcal{Z}$:

1. *Fixed sparsity pattern (FSP)*: the IC factor R has a prescribed sparsity pattern $\mathcal{Z}$. A popular sparsity pattern is that of the original matrix A, i.e., $\mathcal{Z} = \{(i,j) : i \geq j \text{ and } a_{ij} \neq 0\}$.
2. *Drop small elements (DSE)*: the small elements of R are dropped. It is controlled by a drop threshold σ. In this case, the sparsity pattern $\mathcal{Z}$ of R and the number of fill-ins in R are unknown in advance.
3. *Fixed number of non-zero elements per column.* It is similar to the DSE rule except that the maximum number of non-zero elements in each column of R is fixed.

The existence of IC factorization (4.6), for an arbitrary sparsity set $\mathcal{Z}$, is proven only for special classes of matrices [43, 44, 53]. For a general SPD matrix A, the non-zero elements introduced into the error matrix E could result in the loss of positive-definiteness of the matrix $A - E$, and the IC factorization does not exist.

Table 7: IC with DSE, $(N, L, t, \beta) = (16 \times 16, 80, 1, 10)$.

U	0	2	4	6
$\sigma = 10^{-6}$	18.97/0.07	19.17/0.14	19.09/0.30	19.07/0.48
10^{-5}	13.20/0.11	16.92/0.20	16.41/0.80	pbd
10^{-4}	2.95/0.17	5.72/0.58	pbd	pbd
10^{-3}	pbd	pbd	pbd	pbd
10^{-2}	0.01/0.16	pbd	pbd	pbd
10^{-1}	pbd	pbd	pbd	pbd

HQMC application. Table 7 shows numerical results of the IC preconditioner R with sparsity $\mathcal{Z}$ defined by the DSE rule. The reported data is an average of successful solutions over 10 trials with the left-looking IC implementation. The table records the CPU time with different drop tolerance values σ. The first number in each cell is the time for constructing the preconditioner R and the second number is for the PCG iteration. In the table, "pbd" stands for the pivot breakdown to indicate that all of 10 trials failed due to the pivot breakdowns.

We see that the IC factorization encounters the pivot breakdown frequently, except with an extremely small drop tolerance σ. Small drop tolerance leads to high memory and CPU time costs, and becomes impractical for large systems.

Note that the right-looking IC algorithm is mathematically equivalent to its left-looking algorithm, it also encounters the same pivot breakdown. It clearly indicates that the IC preconditioner R is neither robust nor efficient for the HQMC simulation.

4.4.2 Modified IC

To avoid the pivot breakdown, one attempt is to first make a small perturbation of A, say by simple diagonal perturbation, and then compute its IC factorization:

$$A + \alpha D_A = RR^T + S + S^T, \tag{4.13}$$

where $D_A = \text{diag}(A)$ and the scalar α is prescribed [43]. R is referred to as an IC$_p$ preconditioner.

If the shift α is chosen such that $A + \alpha D_A$ is diagonally dominant, then it is provable that the IC factorization (4.13) exists [43]. Table 8 records the performance of the left-looking IC$_p$ implementation, where the shift α is chosen such that $A + \alpha D_A$ is diagonally dominant. In the table, "nnz$_r(R)$" is the average number of nonzero elements per row of R, "Mem(R)" is the memory in MB for storing R in the CSC (Compressed

Table 8: IC_p/diag.dom., $\sigma = 0.001$, $(N, L, t, \beta) = (48 \times 48, 80, 1, 10)$.

U	0	1	2	3	4	5	6
α	1.29	4.15	7.58	12.37	19.70	26.14	38.80
$\text{nnz}_r(R)$	25.29	23.35	21.99	20.90	20.01	19.30	25.01
Mem(R)	57	52	49	47	45	43	56
Wksp.	22	20	19	18	18	17	20
Itrs.	94	357	882	2620	16645	68938	102500
P-time	1.40	1.22	1.13	1.06	1.01	0.97	0.93
S-time	5.03	18.21	43.48	125.58	782.96	3178.58	4621.06
T-time	6.44	19.43	44.60	126.64	783.96	3179.54	4621.98

Table 9: IC_p, $\sigma = 0.001$, $(N, L, t, \beta) = (48 \times 48, 80, 1, 10)$.

U	0	1	2	3	4	5	6
α (fixed)	0.005	0.005	0.005	0.005	0.005	0.005	0.005
$\text{nnz}_r(R)$	22.31	24.43	24.74	24.89	24.96	24.99	25.01
Mem(R)	50	55	55	56	56	56	56
Wksp.	18	19	20	20	20	20	20
Itrs.	14	32	72	190	1087	3795	5400
P-time	1.23	1.29	1.30	1.31	1.31	1.31	1.32
S-time	0.65	1.57	3.47	9.17	52.49	183.15	286.15
T-time	1.87	2.86	4.77	10.48	53.49	184.76	287.47

Sparse Column) format, "Wksp." is the required workspace in MB, and "Itrs." is the total number of PCG iterations.

By Table 8, we see that with the choice of the shift α such that $A + \alpha D_A$ is diagonally dominant, the pivot breakdown is avoided. However, the quality of the resulting preconditioner R is poor. In practice, we observed that good performance can often be achieved with a much smaller shift α. Although $A + \alpha D_A$ is not diagonally dominant, the IC factorization still exists. Table 9 records significant improvements of the IC_p preconditioner R computed with the fixed shift $\alpha = 0.005$.

There is no general strategy for an optimal choice of the shift α. It is computed by a trial-and-error approach in PETSc [47].

4.5 Robust incomplete Cholesky preconditioners

In the IC factorizations (4.6) and (4.13), the discarded elements of R are simply moved to the error matrix E. As we have seen, this may result in the loss of the positive definiteness of the matrix $A - E$ and subsequently lead to the pivot breakdown. To avoid this, the error matrix E needs to be taken into account. It should be updated dynamically during the construction of an IC factorization such that the matrix $A - E$ is preserved to be SPD. Specifically, we want to have an IC factorization algorithm that computes a nonsingular lower triangular matrix R of an arbitrarily prescribed sparsity pattern $\mathcal{Z}$ satisfying

$$\begin{cases} \quad A = RR^T + E, \\ \text{s.t. } A - E > 0. \end{cases} \tag{4.14}$$

In the rest of this section, we will discuss several approaches to construct such an IC factor R satisfying (4.14). The resulting preconditioner R is referred to as a *robust incomplete Cholesky (RIC)* preconditioner.

4.5.1 RIC1

A sufficient condition for the existence of the factorization (4.14) is to ensure that the error matrix $-E$ is symmetric semi-positive definite, $-E = -E^T \geq 0$. For doing so, let us write

$$E = S - D + S^T,$$

where S is strictly lower-triangular and D is diagonal, then a robust IC preconditioner should satisfy

$$\begin{cases} \quad A = RR^T + S - D + S^T \\ \text{s.t. } -(S - D + S^T) \geq 0. \end{cases} \tag{4.15}$$

The factorization is referred to as version 1 of RIC, or RIC1 in short. RIC1 was first studied in [31, 39].

Note that in the IC factorization (4.6), $D = 0$ and $E = S + S^T$. In the modified IC factorization (4.13), the diagonal matrix D is prescribed $D = -\alpha D_A$ and the error matrix $E = S - \alpha D_A + S^T$. Now, in the RIC1 factorization, D will be dynamically assigned and updated during the process to satisfy the condition $-(S - D + S^T) \geq 0$.

The RIC1 factorization can be computed by using the following partition and factorization:

$$\begin{bmatrix} a_{11} & \widehat{a}^T \\ \widehat{a} & A_1 \end{bmatrix} = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & C_1 \end{bmatrix} \begin{bmatrix} r_{11} & \widehat{r}^T \\ 0 & I \end{bmatrix} + \begin{bmatrix} -d_1 & \widehat{s}^T \\ \widehat{s} & -D_1 \end{bmatrix}, \tag{4.16}$$

where $C_1 = A_1 + D_1 - \widehat{r}\widehat{r}^T$.

By the first column of the both sides of the factorization, we have

$$\begin{aligned} a_{11} &= r_{11}^2 - d_1 \\ \widehat{a} &= \widehat{r}\, r_{11} + \widehat{s}. \end{aligned}$$

It suggests that we first compute the vector $v = \widehat{r}\, r_{11} + \widehat{s}$ based on the sparsity set $\mathcal{Z}$:

$$\text{for } i \le n-1, \qquad \begin{cases} v_i = \widehat{a}_i,\ \widehat{s}_i = 0, & \text{if } (i+1,1) \in \mathcal{Z}, \\ v_i = 0,\ \ \widehat{s}_i = \widehat{a}_i, & \text{otherwise.} \end{cases} \tag{4.17}$$

To ensure $-E = -(S - D + S^T) \ge 0$, if there is a discarded element $\widehat{a}_i$ assigned to $\widehat{s}_i$, the diagonal element d_1 and the ith diagonal element $d_i^{(1)}$ of D_1 are updated

$$d_1 := d_1 + \delta_1, \quad d_i^{(1)} := d_i^{(1)} + \delta_i, \tag{4.18}$$

where δ_1 and δ_i are chosen such that $\delta_1, \delta_i > 0$ and $\delta_1\delta_i = \widehat{s}_i^2$. Subsequently, the element r_{11} is determined by

$$r_{11} = \sqrt{a_{11} + d_1},$$

and the vector $\widehat{r}$ is set by

$$\widehat{r} = \frac{1}{r_{11}} v.$$

If we have an RIC1 factorization of the $(n-1) \times (n-1)$ matrix C_1:

$$C_1 = R_1 R_1^T + S_1 - \widehat{D}_1 + S_1^T, \tag{4.19}$$

then the RIC1 factorization (4.15) of A is given by

$$R = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & R_1 \end{bmatrix}, \quad D = \begin{bmatrix} d_1 & 0 \\ 0 & D_1 + \widehat{D}_1 \end{bmatrix}, \quad S = \begin{bmatrix} 0 & 0 \\ \widehat{s} & S_1 \end{bmatrix}.$$

In other words, the RIC1 factorization can be obtained through the repeated applications of (4.16) on (4.19).

The following algorithm computes the RIC1 factor R in the right-looking fashion, where the sparsity $\mathcal{Z}$ is controlled by a prescribed drop tolerance σ. In the algorithm, δ_i and δ_j are chosen so that they result in a same factor of increase in the corresponding diagonal elements.

```
RIC1 (RIGHT-LOOKING VERSION)
1. $R = \text{lower}(A)$
2. $d(1:n) = 0$
3. for  $j = 1, 2, \ldots, n$
4.    for  $i = j+1, j+2, \ldots, n$
5.       $\tau = |r(i,j)|/[(r(i,i) + d(i))(r(j,j) + d(j))]^{1/2}$
6.       if  $\tau \le \sigma$
7.          $r(i,j) = 0$
```

```
 8.         $d(i) := d(i) + \tau(r(i,i) + d(i))$
 9.         $d(j) := d(j) + \tau(r(j,j) + d(j))$
10.       end if
11.    end for
12.    $r(j,j) := \sqrt{r(j,j) + d(j)}$
13.    $r(j+1:n,j) := r(j+1:n,j)/r(j,j)$
14.    for $k = j+1, j+2, \ldots, n$
15.       $r(k:n,k) := r(k:n,k) - r(k,j)r(k:n,j)$
16.    end for
17. end for
```

The RIC1 factorization can also be computed by a left-looking algorithm. By comparing the jth column of the factorization (4.15), we have

$$a_{jj} = \sum_{k=1}^{j} r_{jk}^2 - d_j,$$

$$a_{ij} = \sum_{k=1}^{j} r_{jk} r_{ik} + s_{ij}, \quad i = j+1, \ldots, n$$

This says that

$$r_{jj}^2 - d_j = a_{jj} - \sum_{k=1}^{j-1} r_{jk}^2,$$

$$r_{jj} r_{ij} + s_{ij} = a_{ij} - \sum_{k=1}^{j-1} r_{jk} r_{ik}, \quad i = j+1, \ldots, n.$$

Thus, to compute the jth column of R, one first computes

$$v = a_j - \sum_{k=1}^{j-1} r_{jk}\, r_k,$$

and then imposes the sparsity:

$$\text{for } i \geq j+1, \qquad \begin{cases} s_{ij} = 0, & \text{if } (i,j) \in \mathcal{Z}, \\ s_{ij} = v_i,\ v_i = 0, & \text{otherwise.} \end{cases} \tag{4.20}$$

To ensure $-E = -(S - D + S^T) \geq 0$, if there is a discarded element assigned to s_{ij}, the corresponding diagonal elements d_i and d_j are updated

$$d_i := d_i + \delta_i,\ d_j := d_j + \delta_j, \tag{4.21}$$

where δ_i and δ_j are chosen such that $\delta_i, \delta_j > 0$ and $\delta_i \delta_j = s_{ij}^2$. Initially, all d_i are set to be zero.

Subsequently, the jth column r_j of R is given by

$$\begin{aligned} r_{jj} &= \sqrt{a_{jj} + d_j}, \\ r_{ij} &= v_i / r_{jj}, \quad i = j+1, \ldots, n \end{aligned}$$

The following algorithm computes the RIC1 factor R in the left-looking fashion, where the sparsity is controlled by a drop tolerance σ.

```
    RIC1 (LEFT-LOOKING VERSION)
 1. $d(1:n) = 0$
 2. for $j = 1, 2, \ldots, n$
 3.    $v(j:n) = a(j:n, j)$
 4.    for $k = 1, 2, \ldots, j-1$
 5.       $v(j:n) := v(j:n) - r(j,k) r(j:n,k)$
 6.    end for
 7.    for $i = j+1, j+2, \ldots, n$
 8.       $\tau = |v(i)| / [(a(i,i) + d(i))(a(j,j) + d(j))]^{1/2}$
 9.       if $\tau \le \sigma$
10.          $v(i) = 0$
11.          $d(i) := d(i) + \tau(a(i,i) + d(i))$
12.          $d(j) := d(j) + \tau(a(j,j) + d(j))$
13.       end if
14.    end for
15.    $r(j,j) = \sqrt{a(j,j) + d(j)}$
16.    $r(j+1:n, j) = v(j+1:n) / r(j,j)$
17. end for
```

The computational cost of RIC1 is only slightly higher than the IC preconditioner (4.6). To assess the quality of the RIC1 preconditioner R, we note that the norm of the residue

$$R^{-1} A R^{-T} - I = R^{-1}(S - D + S^T) R^{-T} = R^{-1} E R^{-T} \tag{4.22}$$

could be amplified by a factor of $\|R^{-1}\|^2$ of the error matrix E. When a large number of diagonal updates are necessary, some elements of D could be large. Consequently, the residue norm is large and R is a poor preconditioner.

HQMC application. Table 10 records the performance of the RIC1 preconditioner computed by the left-looking implementation. The drop threshold is set to be $\sigma = 0.003$. With this drop threshold, the resulting RIC1 preconditioners R is are about the same sparsity as the IC$_p$ preconditioners reported in Table 9 of section 4.4.

Table 10: RIC1/left-looking, $\sigma = 0.003$, $(N, L, t, \beta) = (48 \times 48, 80, 1, 10)$.

U	0	1	2	3	4	5	6
$\text{nnz}_r(R)$	22.01	24.28	24.45	24.43	24.33	24.21	24.07
Mem(R)	49	54	55	55	55	54	54
Wksp.	18	19	20	19	19	19	19
Itrs.	20	57	127	342	1990	7530	11460
P-time	1.83	1.93	1.93	1.93	1.92	1.90	1.89
S-time	1.00	3.02	6.69	17.95	104.12	393.60	596.00
T-time	2.82	4.96	8.62	19.88	106.03	395.51	597.90

We note that the quality of the RIC1 preconditioner in terms of the number of PCG iterations is worse than the IC_p preconditioner. This is due to the fact that it is necessary to have a large diagonal matrix D to guarantee the semi-positive definiteness of the error matrix $-E = -(S - D + S^T)$. On the other hand, the RIC1 factorization is provably robust and does not breakdown. In the following sections, we will discuss how to improve the quality of the RIC1 preconditioner.

The right-looking implementation requires the updating of the unfactorized block, i.e., forming the matrix C_1 in (4.16). It causes significant computational overhead. It is less efficient than the left-looking implementation. Therefore, we only present the performance data for the left-looking implementation.

4.5.2 RIC2

One way to improve the quality of the RIC1 preconditioner is by setting the error matrix E as $E = RF^T + FR^T$, where F is strictly lower-triangular. This was proposed in [51]. In this scheme, we compute an IC factorization of the form

$$A = RR^T + RF^T + FR^T. \tag{4.23}$$

Note that the factorization can be equivalently written as

$$A + FF^T = (R + F)(R + F)^T.$$

Hence, the existence of R is guaranteed. With the factorization (4.23), the residue becomes

$$R^{-1}AR^{-T} - I = FR^{-T} + R^{-1}F^T.$$

The residue norm could be amplified at most by the factor of $\|R^{-1}\|$ of the error matrix F, instead of $\|R^{-1}\|^2$ in the RIC1 factorization. We refer (4.23) as version 2 of RIC factorization, or RIC2 in short.

The RIC2 factorization (4.23) can be constructed by using the following partition and factorization:

$$A = \begin{bmatrix} a_{11} & \widehat{a}^T \\ \widehat{a} & A_1 \end{bmatrix} = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & C_1 \end{bmatrix} \begin{bmatrix} r_{11} & \widehat{r}^T \\ 0 & I \end{bmatrix} + \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & 0 \end{bmatrix} \begin{bmatrix} 0 & \widehat{f}^T \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \widehat{f} & 0 \end{bmatrix} \begin{bmatrix} r_{11} & \widehat{r}^T \\ 0 & 0 \end{bmatrix} \tag{4.24}$$

where $C_1 = A_1 - \widehat{r}\widehat{r}^T - \widehat{r}\widehat{f}^T - \widehat{f}\widehat{r}^T$.

By the first column of the both sides of the factorization, we have

$$a_{11} = r_{11}^2,$$
$$\widehat{a} = \widehat{r}\,r_{11} + \widehat{f}\,r_{11}.$$

Hence, we have

$$r_{11} = \sqrt{a_{11}}.$$

The vectors $\widehat{r}$ and $\widehat{f}$ are computed based on the sparsity set $\mathcal{Z}$:

$$\text{for } i \le n-1,$$
$$\begin{cases} \widehat{r}_i = \widehat{a}_i / r_{11}, \ \widehat{f}_i = 0, & \text{if } (i,1) \in \mathcal{Z}, \\ \widehat{f}_i = 0, \quad \widehat{f}_i = \widehat{a}_i / r_{11}, & \text{otherwise.} \end{cases}$$

If an RIC2 factorization of the $(n-1) \times (n-1)$ matrix C_1 is given by

$$C_1 = R_1 R_1^T + R_1 F_1^T + F_1 R_1^T, \tag{4.25}$$

then the RIC2 factorization of A is given by

$$R = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & R_1 \end{bmatrix}, \quad F = \begin{bmatrix} 0 & 0 \\ \widehat{f} & F_1 \end{bmatrix}.$$

Hence the RIC2 factorization can be computed by the repeated applications of (4.24) on (4.25).

The following algorithm computes the RIC2 factor R in the right-looking implementation. The sparsity $\mathcal{Z}$ is controlled by dropping small elements of R with the drop tolerance σ.

```
   RIC2 (RIGHT-LOOKING VERSION)
1. $R = \text{lower}(A)$
2. for $j = 1, 2, \ldots, n$
3.   $r(j,j) := \sqrt{r(j,j)}$
4.   for $i = j+1, j+2, \ldots, n$
5.     if $|a(i,j)|/r(j,j) > \sigma$
6.       $r(i,j) := r(i,j)/r(j,j)$
```

```
 7.          $f(i,j) = 0$
 8.       else
 9.          $r(i,j) = 0$
10.          $f(i,j) = r(i,j)/r(j,j)$
11.       end if
12.    end for
13.    for  $k = j+1, j+2, \ldots, n$
14.       $r(k:n,k) := r(k:n,k) - r(k,j)\, r(k:n,j)$
15.       $r(k:n,k) := r(k:n,k) - r(k,j)\, f(k:n,j)$
16.       $r(k:n,k) := r(k:n,k) - f(k,j)\, r(k:n,j)$
17.    end for
18. end for
```

The RIC2 factorization can also be computed by a left-looking algorithm. By comparing the jth column in the factorization (4.23), we have

$$a_j = \sum_{k=1}^{j} (r_{jk} r_k + r_{jk} f_k + f_{jk} r_k). \tag{4.26}$$

This says that

$$r_{jj}(r_j + f_j) = a_j - \sum_{k=1}^{j-1} (r_{jk} r_k + r_{jk} f_k + f_{jk} r_k).$$

Thus, to compute the jth column of R, one first computes

$$v = a_j - \sum_{k=1}^{j-1} (r_{jk} r_k + r_{jk} f_k + f_{jk} r_k). \tag{4.27}$$

Then, the jth diagonal entry of R is given by

$$r_{jj} = \sqrt{v_j},$$

and the rest of the non-zero elements in r_j and f_j are computed based on the sparsity set $\mathcal{Z}$:

for $i \geq j+1$,

$$\begin{cases} r_{ij} = v_i / r_{jj},\ f_{ij} = 0, & \text{if } (i,j) \in \mathcal{Z}, \\ r_{ij} = 0, \qquad f_{ij} = v(i)/r_{ii}, & \text{otherwise.} \end{cases}$$

The following algorithm computes the RIC2 factor R in the left-looking fashion, where the sparsity of R is controlled by a drop tolerance σ.

```
RIC2 (LEFT-LOOKING VERSION)
1. for  $j = 1, 2, \ldots, n$
```

Table 11: RIC2/left, $\sigma = 0.012$, $(N, L, t, \beta) = (16 \times 16, 80, 1, 10)$.

U	0	1	2	3	4	5	6
Mem(R)	5	5	5	5	5	5	5
Wksp.	129	129	129	129	127	127	127
Iters.	16	36	73	194	344	453	539
P-time	1.79	1.86	1.88	1.90	1.90	1.90	1.88
S-time	0.12	0.27	0.57	1.52	2.70	3.57	4.24
T-time	1.91	2.13	2.45	3.42	4.60	5.47	6.11

```
2.     $v(j:n) = a(j:n,j)$
3.     for  $k = 1, 2, \dots, j-1$
4.        $v(j:n) := v(j:n) - r(j,k)r(j:n,k)$
5.        $v(j:n) := v(j:n) - r(j,k)f(j:n,k)$
6.        $v(j:n) := v(j:n) - f(j,k)r(j:n,k)$
7.     end for
8.     $r(j,j) = \sqrt{v(j)}$
9.     for  $i = j+1, j+2, \dots, n$
10.       if  $|v(i)|/r(j,j) > \sigma$
11.          $r(i,j) = v(i)/r(j,j)$
12.          $f(i,j) = 0$
13.       else
14.          $r(i,j) = 0$
15.          $f(i,j) = v(i)/r(j,j)$
16.       end if
17.    end for
18. end for
```

Note that in the above algorithm, the columns $f_1, f_2, \dots, f_{j-1}$ of the matrix F are required to compute r_j.

HQMC application. Since the left-looking implementation of RIC2 needs to store the entire error matrix F, it requires a large amount of workspace. For example, Table 11 shows that the workspace is about 128MB. It runs out of core memory for $N = 48 \times 48$ and $L = 80$. The right-looking implementation reduces the workspace by a factor of more than 10 but with a significant increase of the CPU time as shown in Table 12. Note that the left-looking and right-looking implementations of RIC2 produce the same preconditioner R. Therefore, the storage requirement for R and the number of the PCG iterations are the same.

Table 12: RIC2/right, $\sigma = 0.012$, $(N, L, t, \beta) = (16 \times 16, 80, 1, 10)$.

U	0	1	2	3	4	5	6
Mem(R)	5	5	5	5	5	5	5
Wksp.	11	11	11	11	11	11	10
Iters.	16	36	73	194	344	453	539
P-time	19.29	19.29	19.31	19.31	19.34	19.28	19.19
S-time	0.12	0.27	0.57	1.52	2.70	3.57	4.24
T-time	19.43	19.58	19.90	20.84	22.06	22.87	23.45

4.5.3 RIC3

One way to reduce the large workspace requirement of the RIC2 factorization (4.23) is to impose additional sparsity of the error matrix F with a secondary drop threshold. This was proposed in [40]. It begins with setting the error matrix E as

$$E = RF^T + FR^T + S - D + S^T,$$

where S is a strictly lower-triangular and represents the discarded elements from F, and D is diagonal. It means that we compute a preconditioner R satisfying

$$\begin{cases} \quad A = RR^T + RF^T + FR^T + S - D + S^T, \\ \text{s.t. } -(S - D + S^T) > 0. \end{cases} \tag{4.28}$$

The sparsity of R and F are controlled by the primary and secondary drop thresholds σ_1 and σ_2, respectively. Similar to the RIC1 factorization (4.15), the diagonal elements D is dynamically updated such that $-(S - D + S^T) \geq 0$. As a result, the robustness of the factorization is guaranteed. We called this as version 3 of the RIC factorization, or RIC3 in short.

With the factorization (4.28), the residue becomes

$$R^{-1}AR^{-T} - I = FR^{-T} + R^{-1}F^T + R^{-1}(S - D - S^T)R^{-T}.$$

Therefore, we see that the residue norm is amplified by a factor of $\|R^{-1}\|$ on the primary error $\|F\| = O(\sigma_1)$, and a factor of $\|R^{-1}\|^2$ on the secondary error $\|S - D - S^T\| = O(\sigma_2)$. The RIC3 factorization will be able to preserve at least the same quality of the RIC2 preconditioner R as long as σ_2 is small enough.

The RIC3 factorization (4.28) can be constructed by using the following partition and factorization:

$$A = \begin{bmatrix} a_{11} & \widehat{a}^T \\ \widehat{a} & A_1 \end{bmatrix} = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & C_1 \end{bmatrix} \begin{bmatrix} r_{11} & \widehat{r}^T \\ 0 & I \end{bmatrix} + \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & 0 \end{bmatrix} \begin{bmatrix} 0 & \widehat{f}^T \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \widehat{f} & 0 \end{bmatrix} \begin{bmatrix} r_{11} & \widehat{r}^T \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} -d_1 & \widehat{s}^T \\ \widehat{s} & -D_1 \end{bmatrix} \tag{4.29}$$

where $C_1 = A_1 + D_1 - \widehat{r}\widehat{r}^T - \widehat{r}\widehat{f} - \widehat{f}\widehat{r}^T$.

By the first column of the both sides of the factorization, we have

$$a_{11} = r_{11}^2 - d_1,$$
$$\widehat{a} = \widehat{r}\, r_{11} + \widehat{f}\, r_{11} + \widehat{s}.$$

It suggests that we first compute the vector $v = (\widehat{r} + \widehat{f})r_{11}$ by imposing the sparsity constraint with the secondary drop tolerance σ_2, i.e.,

$$\text{for } i \leq n-1, \quad \begin{cases} v_i = \widehat{a}_i,\ \widehat{s}_i = 0, & \text{if } \tau > \sigma_2, \\ v_i = 0,\ \ \widehat{s}_i = \widehat{a}_i, & \text{otherwise,} \end{cases}$$

where

$$\tau = \left[\frac{\widehat{a}_i^2}{(a_{11} + d_1)(a_{ii}^{(1)} + d_i^{(1)})} \right]^{1/2},$$

and $a_{ii}^{(1)}$ and $d_i^{(1)}$ denote the ith diagonal elements of A_1 and D_1, respectively.

To ensure $-(S - D + S^T) \geq 0$, if a discarded element $\widehat{a}_i$ is assigned to the position $\widehat{s}_i$, the corresponding diagonal element d_1 and $d_i^{(1)}$ are updated,

$$d_1 := d_1 + \delta_1, \quad d_i^{(1)} := d_i^{(1)} + \delta_i,$$

where δ_1 and δ_i are chosen such that $\delta_1, \delta_i > 0$ and $\delta_1\delta_i = \widehat{s}_i^2$. Initially, all d_i are set to be zero.

Subsequently, the entry r_{11} of R is given by

$$r_{11} = \sqrt{a_{11} + d_1}.$$

Finally, vectors $\widehat{r}$ and $\widehat{f}$ are computed by imposing the primary sparsity constraint on v with the drop threshold σ_1, i.e.,

$$\text{for } i < n-1, \quad \begin{cases} \widehat{r}_i = v_i/r_{11},\ \widehat{f}_i = 0, & \text{if } |v_i|/r_{11} > \sigma_1, \\ \widehat{r}_i = 0, \qquad\quad \widehat{f}_i = v_i/r_{11}, & \text{otherwise.} \end{cases}$$

Note that the vectors $\widehat{r}$ and $\widehat{f}$ are structurally "orthogonal", i.e., $\widehat{r}_i f_i = 0$ for all i.

If we have an RIC3 factorization of the $(n-1) \times (n-1)$ matrix C_1,

$$C_1 = R_1 R_1^T + R_1 F_1^T + F_1 R_1^T + S_1 - \widehat{D}_1 + S_1^T, \tag{4.30}$$

then the RIC3 factorization is given by

$$R = \begin{bmatrix} r_{11} & 0 \\ \widehat{r} & R_1 \end{bmatrix}, \ F = \begin{bmatrix} 0 & 0 \\ \widehat{f} & F_1 \end{bmatrix}, \ S = \begin{bmatrix} 0 & 0 \\ \widehat{s} & S_1 \end{bmatrix}, \ D = \begin{bmatrix} d_1 & 0 \\ 0 & D_1 + \widehat{D}_1 \end{bmatrix}.$$

Thus, the RIC3 factorization can be computed by the repeated application of (4.29) on (4.30).

The following algorithm computes the RIC3 factor R in the right-looking fashion, where δ_i and δ_j are chosen in the same way as in the RIC1 algorithms for the same increasing of the diagonal elements of D.

```
RIC3 (RIGHT-LOOKING VERSION)
 1. $R = \mathrm{lower}(A)$
 2. $d(1:n) = 0$
 3. for  $j = 1, 2, \ldots, n$
 4.    for  $i = j+1, j+2, \ldots, n$
 5.       $\tau = |r(i,j)|/[(a(i,i) + d_i)(a(j,j) + d(j))]^{1/2}$
 6.       if  $\tau \le \sigma_2$
 7.          $r(i,j) = 0$
 8.          $d(i) := d(i) + \tau(a(i,i) + d(i))$
 9.          $d(j) := d(j) + \tau(a(j,j) + d(j))$
10.       end if
11.    end for
12.    $r(j,j) = \sqrt{a(j,j) + d(j)}$
13.    for  $i = j+1, j+2, \ldots, n$
14.       if  $|r(i,j)|/r(j,j) > \sigma_1$
15.          $r(i,j) := r(i,j)/r(j,j)$
16.          $f(i,j) = 0$
17.       else
18.          $r(i,j) = 0$
19.          $f(i,j) = r(i,j)/r(j,j)$
20.       end if
21.    end for
22.    for  $k = j+1, j+2, \ldots, n$
23.       $r(k:n,k) := r(k:n,k) - r(k,j)r(k:n,j)$
24.       $r(k:n,k) := r(k:n,k) - r(k,j)f(k:n,j)$
25.       $r(k:n,k) := r(k:n,k) - f(k,j)r(k:n,j)$
26.    end for
27. end for
```

We remark that in the previous right-looking algorithm, the jth column f_j of F can be discarded after it is used to update the remaining column $r_{j+1}, \ldots, r_n$.

The RIC3 factorization can also be computed by a left-looking algorithm. By comparing the jth column in the factorization (4.28), we have

$$\begin{aligned} a_{jj} &= \sum_{k=1}^{j} r_{jk}^2 - d_{jj}, \\ a_{ij} &= s_{ij} + \sum_{k=1}^{i} (r_{jk} r_{ik} + r_{jk} f_{ik} + f_{jk} r_{ik}), \quad i = j+1, j+2, \ldots, n. \end{aligned}$$

This says that

$$r_{jj}^2 + d_{jj} = a_{jj} - \sum_{k=1}^{j-1} r_{jk}^2,$$

and

$$r_{jj}(r_{ij} + f_{ik}) + s_{ij} = a_{ij} - \sum_{k=1}^{j-1} (r_{jk}(r_{ik} + f_{ik}) + f_{jk} r_{ik}), \quad i = j+1, \ldots, n.$$

Therefore, to compute the jth column of R, one first computes the vector

$$v = a_j - \sum_{k=1}^{j-1} (r_{jk}(r_k + f_k) + f_{jk} r_k).$$

Then the sparsity of v is imposed with the secondary drop threshold σ_2, i.e.

$$\text{for } i \geq j+1, \quad \begin{cases} s_{ij} = 0, & \text{if } \tau > \sigma_2, \\ s_{ij} = v_i,\ v_i = 0, & \text{otherwise,} \end{cases}$$

where

$$\tau = \left[\frac{v_i^2}{(a_{ii} + d_i)(a_{jj} + d_j)} \right]^{1/2}.$$

To ensure $-(S - D + S^T) \geq 0$, if a discarded element $\widehat{a}_i$ is entered into the position $\widehat{s}_i$, the diagonal elements d_i and d_j are updated,

$$d_i = d_i + \delta_i, \quad d_j = d_j + \delta_j,$$

where δ_i and δ_j are chosen such that $\delta_i, \delta_j > 0$ and $\delta_i \delta_j = s_{ij}^2$. Initially, all d_i are set to be zero.

Subsequently, the jth diagonal entry of R is given by

$$r_{jj} = \sqrt{v_j + d_j},$$

and the rest of non-zero elements in r_j and f_j are computed by imposing the primary sparsity constraint on v with the primary drop threshold σ_1, i.e.

$$\text{for } i \geq j+1,$$

$$\begin{cases} r_{ij} = v_i/r_{jj},\ f_{ij} = 0, & \text{if } |v_i|/r_{jj} > \sigma_1 \\ r_{ij} = 0, \qquad f_{ij} = v_i/r_{jj}, & \text{otherwise.} \end{cases}$$

The following algorithm computes the RIC3 factor R in the left-looking fashion.

```
RIC3 (LEFT-LOOKING VERSION)
 1. $d(1:n) = 0$
 2. for  $j = 1, 2, \ldots, n$
 3.    $v(j:n) = a(j:n, j)$
 4.    for  $k = 1, 2, \ldots, k-1$
 5.       $v(j:n) := v(j:n) - r(j,k)r(j:n,k)$
 6.       $v(j:n) := v(j:n) - r(j,k)f(j:n,k)$
 7.       $v(j:n) := v(j:n) - f(j,k)r(j:n,k)$
 8.    end for
 9.    for  $i = j+1, j+2, \ldots, n$
10.       $\tau = |v(i)|/[(a(i,i) + d(i))(a(j,j) + d(j))]^{1/2}$
11.       if  $\tau \leq \sigma_2$
12.          $v(i) = 0$
13.          $d(i) := d(i) + \tau(a(i,i) + d(i))$
14.          $d(j) := d(j) + \tau(a(j,j) + d(j))$
15.       end if
16.    end for
17.    $r(j,j) = \sqrt{v(j) + d(j)}$
18.    for  $i = j+1, j+2, \ldots, n$
19.       if  $|v(i)|/r(j,j) > \sigma_1$
20.          $r(i,j) = v(i)/r(j,j)$
21.          $f(i,j) = 0$
22.       else
23.          $r(i,j) = 0$
24.          $f(i,j) = v(i)/r(j,j)$
25.       end if
26.    end for
27. end for
```

Note that in the above algorithm, the columns $f_1, f_2, \ldots, f_{j-1}$ are needed to compute the jth column r_j.

Table 13: RIC3/left, $\sigma_1 = 0.005$, $\sigma_2 = 0.00025$, $(N, L, t, \beta) = (48 \times 48, 80, 1, 10)$.

U	0	1	2	3	4	5	6
$\mathrm{nnz}_r(R)$	25.84	27.24	26.98	26.73	26.53	26.35	26.20
$\mathrm{nnz}_r(F)$	42.65	51.84	49.04	46.90	45.21	43.85	42.69
Mem(R)	58	61	60	60	59	59	59
Wksp.	200	236	226	217	211	206	201
Itrs.	12	29	66	106	1026	3683	5412
P-time	5.54	6.36	6.07	5.84	5.65	5.51	5.46
S-time	0.60	1.49	3.33	9.22	51.12	182.24	296.24
T-time	6.13	7.86	9.40	15.05	56.77	188.05	301.71

HQMC application. Table 13 shows the numerical results of the RIC3 preconditioner computed by the left-looking implementation. The drop thresholds are set to be $\sigma_1 = 0.005$ and $\sigma_2 = 0.00025$. With these drop thresholds, the RIC3 preconditioners R are of about the same sparsity as the IC_p and RIC1 preconditioners presented Tables 9 and 10, respectively.

The RIC3 factorization introduces smaller diagonal updates D and results a preconditioner of better quality than the RIC1 preconditioner. Even though the quality of the IC_p preconditioner for the particular choice of the shift reported in Table 9 is as good as the RIC3 preconditioner, the robustness of the IC_p factorization is not guaranteed, and the quality strongly depends on the choice of the shift α.

The right-looking algorithm is not competitive. Similar to the RIC2 implementations, although the right-looking implementation reduces the workspace requirement, it significantly increases the CPU time.

4.6 Performance evaluation

The numerical results presented so far in this section indicate that the IC_p and RIC3 preconditioners are the most competitive ones for solving the HQMC linear system (4.1). In this section, we focus on these two preconditioners and evaluate their performance for solving HQMC linear systems (4.1) with respect to the length-scale parameter N and energy-scale parameter U. The rest of parameters of the linear systems are $(L, t, \beta, \mu) = (80, 1, 10, 0)$. The IC_p preconditioners are computed with the diagonal shift $\alpha = 10^{-3}$ and the drop tolerance $\sigma = 10^{-3}$. On the other hand, the RIC3 preconditioners are computed with the drop tolerances $\sigma_1 = 10^{-2}$ and $\sigma_2 = 10^{-3}$.

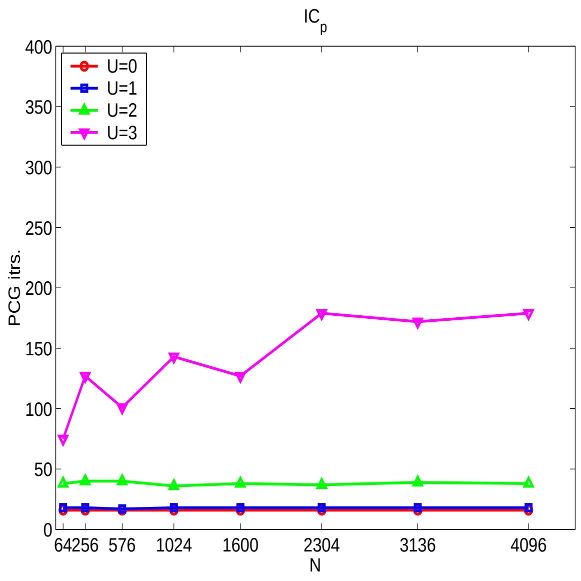

Figure 4.7: Number of PCG iterations using IC$_p$ preconditioner, $U = 0, 1, 2, 3$.

4.6.1 Moderately interacting systems

Figures 4.7 and 4.8 show the performance of the PCG solvers using IC$_p$ and RIC3 preconditioners for moderate interacting systems, namely $U = 0, 1, 2, 3$. These plots show that as lattice size N increases, the numbers of PCG iterations are essentially constants for $U = 0, 1, 2$ and only increases slightly for $U = 3$.

The number of PCG iterations indicates the linear-scaling of PCG solver with respect to the lattice size N. Figures 4.9 and 4.10 show the CPU elapsed time. The black dashed lines indicate the linear-scaling for $U = 1$ and 3. The CPU time at $N = 40 \times 40$ is used as the reference point.

To summarize, the quality of the IC$_p$ and RIC3 preconditioners are comparable. The IC$_p$ preconditioner is slightly more efficient than the RIC3 preconditioner in terms of the total CPU elapsed time. We should note that even though the pivot breakdown did not occur with the shift $\alpha = \sigma$, the IC$_p$ factorization is not provable robust.

4.6.2 Strongly interacting systems

For strongly interacting systems, namely $U \geq 4$, the number of PCG iterations grows rapidly as the lattice sizes N increasing as shown in

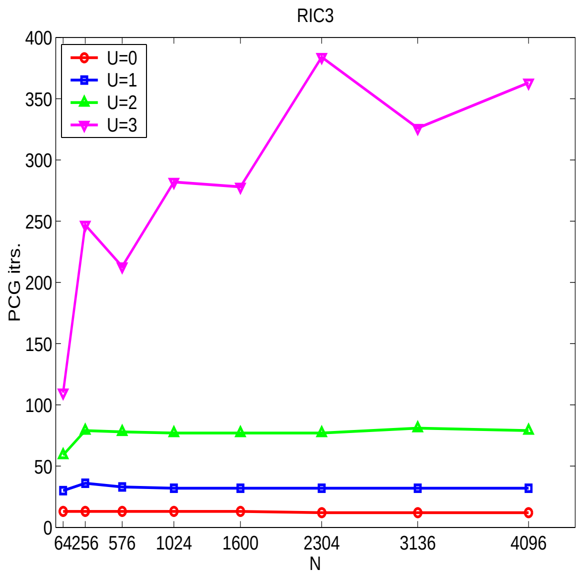

Figure 4.8: Number of PCG iterations using RIC3 preconditioner, $U = 0, 1, 2, 3$.

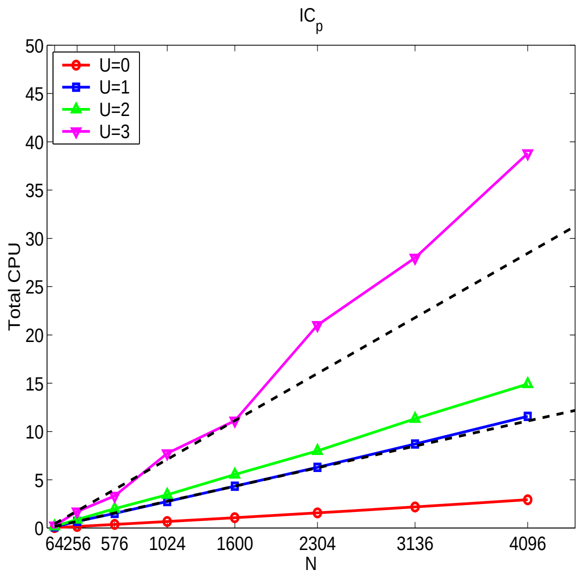

Figure 4.9: CPU time of PCG using IC$_p$ preconditioner, $U = 0, 1, 2, 3$.

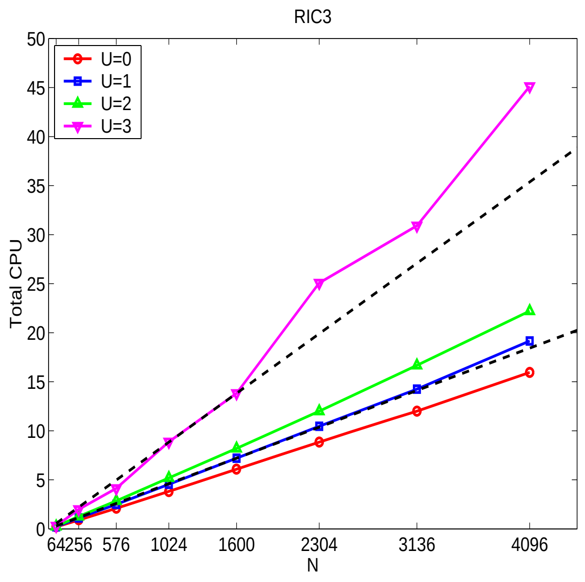

Figure 4.10: CPU time of PCG using RIC3 preconditioner, $U = 0, 1, 2, 3$.

Figures 4.11 and 4.12. The CPU elapsed time are shown in Figures 4.13 and 4.14. As we see that the RIC3 preconditioner slightly outperforms the IC$_p$ preconditioner. However, for both preconditioners, the total CPU time of the PCG solver scales at the order of N^2. The dashed line indicates the desired linear-scaling for $U = 4$. The CPU time at $N = 40 \times 40$ is used as the reference point.

To summarize, for strongly-interacting systems, the linear equations (4.1) are ill-conditioned. It remains an open problem whether there is a preconditioning technique to achieve a linear-scaling PCG solver for the strongly-interacting systems.

Remark 4.1. We have observed that for strongly-interacting systems, the residual norm stagnates after initial rapid decline. Figure 4.15 shows the relative residual norm of the PCG iteration for $(N, L, U, t, \beta, \mu) = (32 \times 32, 80, 6, 1, 10, 0)$.

The plateau is largely due to the the slow decay of the components of the residual vector associated with the small eigenvalues of the preconditioned matrix $R^{-1}AR^{-T}$. Several techniques have been proposed to deflate these components from the residual vector as a way to avoid the plateau of the convergence, see [32, 36, 35, 45, 46] and references within. It remains to be studied about the applicability of these techniques to our HQMC applications.

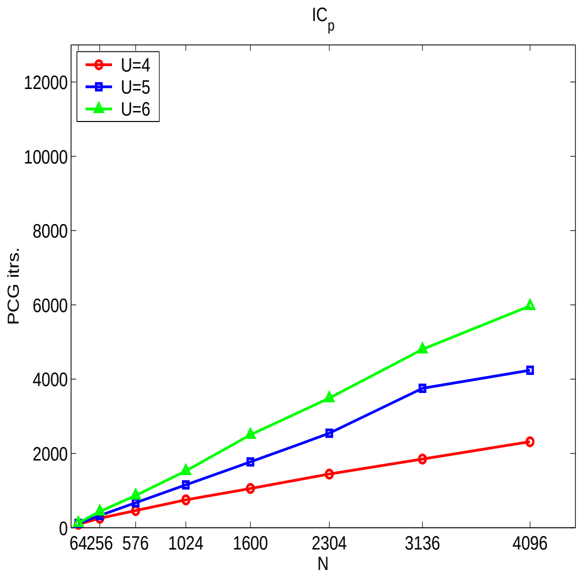

Figure 4.11: Number of PCG iterations using IC$_p$ preconditioner, $U = 4, 5, 6$.

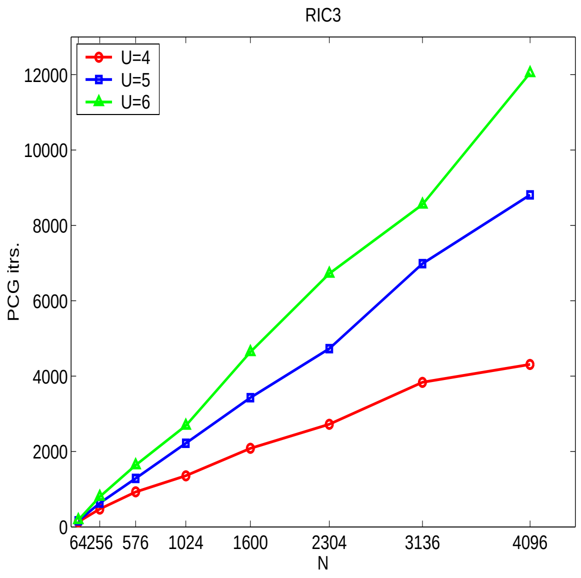

Figure 4.12: Number of PCG iterations using RIC3 preconditioner, $U = 4, 5, 6$.

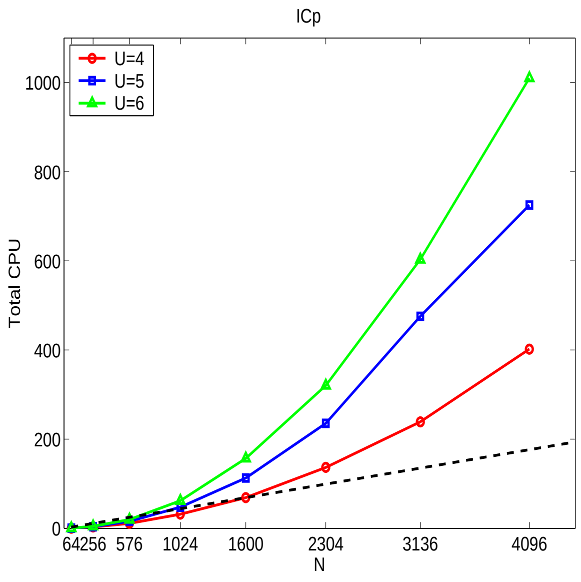

Figure 4.13: CPU time of PCG using IC$_p$ preconditioner, $U = 4, 5, 6$.

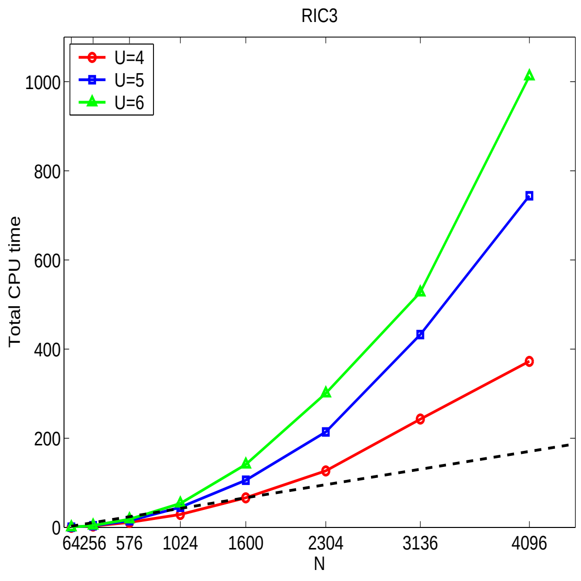

Figure 4.14: CPU time of PCG using RIC3 preconditioner, $U = 4, 5, 6$.

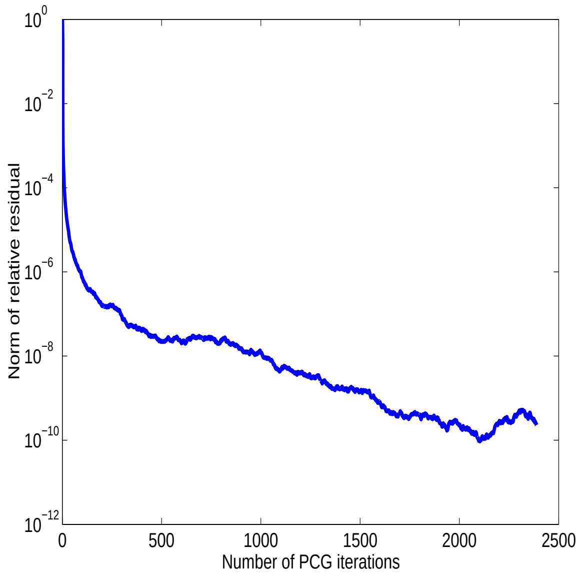

Figure 4.15: Relative residual norms of PCG iterations using RIC3 preconditioner.

Appendix A. Updating algorithm in DQMC

In this appendix, we discuss the single-state MC updating algorithm to provide a fast means to compute the Metropolis ratio in DQMC described in Section 1.2.2.

A.1 Rank-one updates

Consider matrices M_1 and M_2 of the forms

$$M_1 = I + FV_1 \quad \text{and} \quad M_2 = I + FV_2,$$

where F is a given matrix. V_1 and V_2 are diagonal and nonsingular, and moreover, they differ only at the (1,1)-element, i.e.,

$$V_1^{-1}V_2 = I + \alpha_1 e_1 e_1^T,$$

where

$$\alpha_1 = \frac{V_2(1,1)}{V_1(1,1)} - 1,$$

and e_1 is the first column of the identity matrix I.

It is easy to see that M_2 is a rank-one update of M_1:

$$\begin{aligned} M_2 &= I + FV_1 + FV_1(V_1^{-1}V_2 - I) \\ &= M_1 + \alpha_1(M_1 - I)e_1e_1^T \\ &= M_1\left[I + \alpha_1(I - M_1^{-1})e_1e_1^T\right]. \end{aligned}$$

The ratio of the determinants of the matrices M_1 and M_2 is immediately given by[13]

$$r_1 = \frac{\det[M_2]}{\det[M_1]} = 1 + \alpha_1(1 - e_1^T M_1^{-1} e_1). \tag{4.31}$$

Therefore, computing the ratio r_1 is essentially about computing the (1,1)-element of the inverse of the matrix M_1.

By Sherman-Morrison-Woodbury formula,[14] the inverse of the matrix M_2 is a rank-one update of M_1^{-1}:

$$\begin{aligned} M_2^{-1} &= \left[I - \frac{\alpha_1}{r_1}(I - M_1^{-1})e_1e_1^T\right] M_1^{-1} \\ &= M_1^{-1} - \left(\frac{\alpha_1}{r_1}\right) u_1 w_1^T, \end{aligned} \tag{4.32}$$

where

$$u_1 = (I - M_1^{-1})e_1, \quad w_1 = M_1^{-T}e_1.$$

Now, let us consider a sequence of matrices M_{i+1} generated by rank-one updates

$$M_{i+1} = I + FV_{i+1}$$

for $i = 1, 2, \ldots, n-1$, where

$$V_i^{-1}V_{i+1} = I + \alpha_i e_i e_i^T, \quad \alpha_i = \frac{V_{i+1}(i,i)}{V_i(i,i)} - 1.$$

Then by equation (4.31), we immediately have

$$r_i = \frac{\det[M_{i+1}]}{\det[M_i]} = 1 + \alpha_i(1 - e_i^T M_i^{-1} e_i),$$

and

$$M_{i+1}^{-1} = M_i^{-1} - \left(\frac{\alpha_i}{r_i}\right) u_i w_i^T,$$

where $u_i = (I - M_i^{-1})e_i$ and $w_i = M_i^{-T}e_i$.

[13]Here we use the fact that $\det[I + xy^T] = 1 + y^T x$ for any two column vectors x and y.

[14]$(A + UV^T)^{-1} = A^{-1} - A^{-1}(I + V^T A^{-1} U)^{-1} U^T A^{-1}$.

Denote

$$U_k = [u_1, u_2, \cdots, u_{k-1}] \quad \text{and} \quad W = [w_1, w_2, \cdots, w_{k-1}].$$

then it is easy to see that the inverse of M_k can be written as a rank-$(k-1)$ update of M_1^{-1}:

$$M_k^{-1} = M_1^{-1} - U_{k-1} D_k W_{k-1}^T,$$

where $D_k = \operatorname{diag}(\frac{\alpha_1}{r_1}, \frac{\alpha_2}{r_2}, \ldots, \frac{\alpha_{k-1}}{r_{k-1}})$.

Numerical stability of the rank updating procedure have been examined in [54] and [55].

A.2 Metropolis ratio and Green's function computations

As we discussed in section 1.2.2 of the DQMC simulation, it is necessary to repeatedly compute the Metropolis ratio

$$r = \frac{\det[M_+(h')]\det[M_-(h')]}{\det[M_+(h)]\det[M_-(h)]},$$

for configurations $h = (h_1, h_2, \ldots, h_L)$ and $h' = (h'_1, h'_2, \ldots, h'_L)$, where $M_\sigma(h)$ is defined in (1.18), namely

$$M_\sigma(h) = I + B_{L,\sigma}(h_L) B_{L-1,\sigma}(h_{L-1}) \cdots B_{2,\sigma}(h_2) B_{1,\sigma}(h_1).$$

The Green's function associated with the configuration h is defined as

$$G^\sigma(h) = M_\sigma^{-1}(h).$$

In the DQMC simulation, the elements of configurations h' and h are the same except at a specific imaginary time slice ℓ and spatial site i,

$$h'_{\ell,i} = -h_{\ell,i}.$$

It says that the configuration h' is obtained by a simple flipping at the site (ℓ, i).

The Monte-Carlo updates run in the double-loop for $\ell = 1, 2, \ldots, L$ and $i = 1, 2, \ldots, N$. Let us start with the imaginary time slice $\ell = 1$:

- At the spatial site $i = 1$:

  $$h'_{1,1} = -h_{1,1}.$$

  By the relationship between $M_\sigma(h')$ and $M_\sigma(h)$ and equation (4.31), one can derive that the Metropolis ratio r_{11} is given by

  $$r_{11} = d_+ d_-, \tag{4.33}$$

where for $\sigma = \pm$,

$$\begin{aligned} d_\sigma &= 1 + \alpha_{1,\sigma}\left(1 - e_1^T M_\sigma^{-1}(h) e_1\right) \\ &= 1 + \alpha_{1,\sigma}\left(1 - G_{11}^\sigma(h)\right), \end{aligned}$$

and

$$\alpha_{1,\sigma} = e^{-2\sigma\nu h_{1,1}} - 1.$$

Therefore, the gist of computing the Metropolis ratio r_{11} is to compute the $(1,1)$-element of the inverse of the matrix $M_\sigma(h)$. If the Green's function $G^\sigma(h)$ has been computed explicitly in advance, then it is essentially *free* to compute the ratio r_{11}.

In the DQMC simulation, if the proposed h' is accepted, then by the equality (4.32), the Green's function $G^\sigma(h)$ is updated by a rank-one matrix:

$$G^\sigma(h) \leftarrow G^\sigma(h) - \frac{\alpha_{1,\sigma}}{r_{11}} u_\sigma w_\sigma^T.$$

where

$$u_\sigma = (I - G^\sigma(h)) e_1 \quad \text{and} \quad w_\sigma = (G^\sigma(h))^T e_1.$$

- At the spatial site $i = 2$:

$$h'_{1,2} = -h_{1,2}.$$

By a similar derivation as for the previous case, we have

$$r_{12} = d_+ d_-, \tag{4.34}$$

where for $\sigma = \pm$,

$$d_\sigma = 1 + \alpha_{2,\sigma}\left(1 - G_{12}^\sigma(h)\right), \quad \alpha_{2,\sigma} = e^{-2\sigma h_{1,2}} - 1.$$

Correspondingly, if necessary, the Green's function is updated by the rank-one matrix

$$G^\sigma(h) \leftarrow G^\sigma(h) - \frac{\alpha_{2,\sigma}}{r_{12}} u_\sigma w_\sigma^T.$$

where

$$u_\sigma = (I - G^\sigma(h)) e_2 \quad \text{and} \quad w_\sigma = (G^\sigma(h))^T e_2.$$

- In general, for $i = 3, 4, \ldots, N$, we can immediately see that the same procedure can be used for computing the Metropolis ratios r_{1i} and updating the Green's functions.

Remark 4.2. For high performance computing, one may delay the update of the Green's functions to lead to a block high rank update instead of rank-one update. This is called a "delayed update" technique.

Now, let us consider how to do the DQMC configuration update for the time slice $\ell = 2$. We first notice that the matrices $M_\sigma(h)$ and $M_\sigma(h')$ can be rewritten as

$$\begin{aligned} M_\sigma(h) &= B_{1,\sigma}^{-1}(h_1)\widehat{M}_\sigma(h)B_{1,\sigma}(h_1) \\ M_\sigma(h') &= B_{1,\sigma}^{-1}(h'_1)\widehat{M}_\sigma(h')B_{1,\sigma}(h'_1) \end{aligned}$$

where

$$\begin{aligned} \widehat{M}_\sigma(h) &= I + B_{1,\sigma}(h_1)B_{L,\sigma}(h_L)B_{L-1,\sigma}(h_{L-1})\cdots B_{2,\sigma}(h_2) \\ \widehat{M}_\sigma(h') &= I + B_{1,\sigma}(h'_1)B_{L,\sigma}(h'_L)B_{L-1,\sigma}(h'_{L-1})\cdots B_{2,\sigma}(h'_2). \end{aligned}$$

The Metropolis ratios r_{2i} corresponding to the time slice $\ell = 2$ can be written as

$$r_{2i} = \frac{\det[M_+(h')]\det[M_-(h')]}{\det[M_+(h)]\det[M_-(h)]} = \frac{\det[\widehat{M}_+(h')]\det[\widehat{M}_-(h')]}{\det[\widehat{M}_+(h)]\det[\widehat{M}_-(h)]}.$$

and the associated Green's functions are given by "wrapping":

$$\widehat{G}^\sigma(h) \leftarrow B_{1,\sigma}^{-1}(h_1)G^\sigma(h)B_{1,\sigma}(h_1).$$

As a result of the wrapping, the configurations h_2 and h'_2 associated with the time slice $\ell = 2$ appear at the same location of the matrices $\widehat{M}_\sigma(h)$ and $\widehat{M}_\sigma(h')$ as the configurations h_1 and h'_1 at the time slice $\ell = 1$. Therefore, we can use the same formulation as for the time slice $\ell = 1$ to compute the Metropolis ratios r_{2i} and update the associated Green's functions.

For $\ell \geq 3$, it is clear that we can repeat the wrapping trick to compute the Metropolis ratios $r_{\ell i}$ and updating the associated Green's functions.

Remark 4.3. By the discussion, see that the main computing cost of computing the Metropolis ratios $r_{\ell i}$ is on the Green's function updating. It costs $2N^2$ flops for each update. The total cost of one sweep through all $N \times L$ Hubbard-Stratonovich variables h is $2N^3L$. An important issue is about numerical stability and efficiency of computation, updating and wrapping of the Green's functions. A QR decomposition with partial pivoting based method is currently used in the DQMC implementation [11].

Appendix B Particle-hole transformation

In this appendix, we present an algebraic derivation for the so-called *particle-hole transformation*.

B.1 Algebraic identities

We first present a few algebraic identities.

Lemma 4.1. For any nonsingular matrix A,

$$(I + A^{-1})^{-1} = I - (I + A)^{-1}.$$

PROOF. By straightforward verification. □

Lemma 4.2. Let the matrices A_ℓ be symmetric and nonsingular for $\ell = 1, 2, \cdots, m$, then

$$(I + A_m^{-1} A_{m-1}^{-1} \cdots A_1^{-1})^{-1} = I - (I + A_m A_{m-1} \cdots A_1)^{-T}.$$

PROOF. By straightforward verification. □

Theorem 4.1. For any square matrices A and B, if there exists a nonsingular matrix Π such that

$$\Pi A + A\Pi = 0 \quad \text{and} \quad \Pi B - B\Pi = 0,$$

namely, Π anti-commutes with A and commutes with B. Then we have

$$(I + e^{A-B})^{-1} = I - \Pi^{-1}(I + e^{A+B})^{-1}\Pi \tag{4.35}$$

and

$$\det\left[I + e^{A-B}\right] = e^{\mathrm{Tr}(A-B)} \det\left[I + e^{A+B}\right]. \tag{4.36}$$

PROOF. First, we prove the inverse identity (4.35),

$$\begin{aligned}(I + e^{A-B})^{-1} &= I - (I + e^{-A+B})^{-1} = I - (I + e^{\Pi^{-1}(A+B)\Pi})^{-1} \\ &= I - (I + \Pi^{-1}e^{A+B}\Pi)^{-1} = I - \Pi^{-1}(I + e^{A+B})^{-1}\Pi.\end{aligned}$$

Now, let us prove the determinant identity (4.36). Note that

$$\begin{aligned}I + e^{A-B} &= e^{A-B}(I + e^{-(A-B)}) = e^{A-B}(I + e^{-A+B}) \\ &= e^{A-B}(I + e^{\Pi^{-1}A\Pi + \Pi^{-1}B\Pi}) = e^{A-B}(I + \Pi^{-1}e^{A+B}\Pi) \\ &= e^{A-B}\Pi^{-1}(I + e^{A+B})\Pi.\end{aligned}$$

Hence, we have

$$\begin{aligned}\det\left[I + e^{A-B}\right] &= \det[e^{A-B}] \cdot \det[\Pi^{-1}] \cdot \det[I + e^{A+B}] \cdot \det[\Pi] \\ &= e^{\mathrm{Tr}(A-B)} \det[I + e^{A+B}].\end{aligned}$$

For the last equality, we used the identity $\det e^W = e^{\mathrm{Tr} W}$ for any square matrix W. □

The following theorem gives the relations of the inverses and determinants of the matrices $I + e^A e^{-B}$ and $I + e^A e^B$.

Theorem 4.2. For symmetric matrices A and B, if there exists a nonsingular matrix Π such that

$$\Pi A + A\Pi = 0 \quad \text{and} \quad \Pi B - B\Pi = 0.$$

Then we have

$$(I + e^A e^{-B})^{-1} = I - \Pi^{-T}(I + e^A e^B)^{-T}\Pi^T \tag{4.37}$$

and

$$\det[I + e^A e^{-B}] = e^{\text{Tr}(A-B)} \det[I + e^A e^B] \tag{4.38}$$

Proof. Similar to the proof of Theorem 4.1. □

The following two theorems are the generalization of Theorem 4.2.

Theorem 4.3. Let $M_\sigma = I + e^A e^{\sigma B_k} e^A e^{\sigma B_{k-1}} \cdots e^A e^{\sigma B_1}$, where A and $\{B_\ell\}$ are symmetric, $\sigma = +, -$. If there exists a nonsingular matrix Π that anti-commutes with A and commutes with B_ℓ, i.e.,

$$\Pi A + A\Pi = 0 \quad \text{and} \quad \Pi B_\ell - B_\ell \Pi = 0 \quad \text{for } \ell = 1, 2, \ldots, k.$$

Then we have

$$M_-^{-1} = I - \Pi^{-T} M_+^{-T} \Pi^T \tag{4.39}$$

and

$$\det[M_-] = e^{k\text{Tr}(A) - \sum_{\ell=1}^k \text{Tr}(B_\ell)} \det[M_+] \tag{4.40}$$

Theorem 4.4. Let A and B be symmetric matrices and W be a nonsingular matrix. If there exists a nonsingular matrix Π such that it anti-commutes with A and commutes with B, i.e.,

$$\Pi A + A\Pi = 0 \quad \text{and} \quad \Pi B - B\Pi = 0$$

and furthermore, it satisfies the identity

$$\Pi = W \Pi W^T.$$

Then

$$(I + e^A e^{-B} W)^{-1} = I - \Pi^{-T}(I + e^A e^B W)^{-T}\Pi^T \tag{4.41}$$

and

$$\det[I + e^A e^{-B} W] = e^{\text{Tr}(A-B)} \cdot \det[W] \cdot \det[I + e^A e^B W]. \tag{4.42}$$

B.2 Particle-hole transformation in DQMC

For the simplest 1-D lattice of N_x sites:

$$K_x = \begin{bmatrix} 0 & 1 & & & 1 \\ 1 & 0 & 1 & & \\ & 1 & 0 & 1 & \\ & & \ddots & \ddots & \ddots \\ 1 & & & 1 & 0 \end{bmatrix}_{N_x \times N_x}.$$

and $N_x \times N_x$ diagonal matrices V_ℓ for $\ell = 1, 2, \ldots, L$, if N_x is even, the matrix

$$\Pi_x = \mathrm{diag}(1, -1, 1, -1, \ldots, 1, -1)$$

anti-commutes with K_x and commutes with V_ℓ:

$$\Pi_x K_x + K_x \Pi_x = 0$$

and

$$\Pi_x V_\ell - V_\ell \Pi_x = 0 \quad \text{for} \quad \ell = 1, 2, \ldots, L.$$

Then by Theorem 4.3, the determinants of the matrices M_- and M_+ satisfy the relation

$$\det[M_-] = e^{-\sum_{\ell=1}^{L} \mathrm{Tr}(V_\ell)} \det[M_+].$$

For the Green's functions:

$$G^\sigma = M_\sigma^{-1} = \left(I + e^{\Delta\tau t K_x} e^{\sigma V_L} e^{\Delta\tau t K_x} e^{\sigma V_{L-1}} \cdots e^{\Delta\tau t K_x} e^{\sigma V_1}\right)^{-1}$$

where $\sigma = +$ or $-$, we have

$$G^- = I - \Pi_x (G^+)^T \Pi_x.$$

This is referred to as the *particle-hole transformation* in the condensed matter physics literature because it can be viewed as a change of operators $c_{i\downarrow} \to c_{i\downarrow}^\dagger$.

For a 2-D rectangle lattice with $N_x \times N_y$ sites:

$$K = K_x \otimes I + I \otimes K_y.$$

and $N_x N_y \times N_x N_y$ diagonal matrices V_ℓ for $\ell = 1, 2, \ldots, L$, if N_x and N_y are even, the matrix

$$\Pi = \Pi_x \otimes \Pi_y$$

anti-commutes with K and commutes with V_ℓ:

$$\Pi K + K \Pi = 0$$

and

$$\Pi V_\ell - V_\ell \Pi = 0 \quad \text{for} \quad \ell = 1, 2, \ldots, L.$$

Then by Theorem 4.3, we have

$$\det[M_-] = e^{-\sum_{\ell=1}^{L} \mathrm{Tr}(V_\ell)} \det[M_+].$$

This is the identity used for the equation (1.26). For the Green's functions, we have

$$G^\sigma = M_\sigma^{-1} = \left(I + e^{\Delta\tau t K} e^{\sigma V_L} e^{\Delta\tau t K} e^{\sigma V_{L-1}} \cdots e^{\Delta\tau t K} e^{\sigma V_1}\right)^{-1}$$

where $\sigma = +$ (spin up) or $-$ (spin down), we have

$$G^- = I - \Pi (G^+)^T \Pi.$$

This is the *particle-hole transformation* for the 2D rectangle lattice.

B.3 Particle-hole transformation in the HQMC

In the HQMC, we consider the matrix M_σ of the form

$$\begin{aligned} M_\sigma &= \begin{bmatrix} I & & & & e^{\Delta\tau t} K e^{\sigma V_1} \\ -e^{\Delta\tau t K} e^{\sigma V_2} & I & & & \\ & -e^{\Delta\tau t K} e^{\sigma V_2} & I & & \\ & & \ddots & \ddots & \\ & & & -e^{\Delta\tau t} K e^{\sigma V_L} & I \end{bmatrix} \\ &= I + e^{A} e^{\sigma D} P, \end{aligned}$$

where $A = \mathrm{diag}(\Delta\tau t K, \Delta\tau t K, \ldots, \Delta\tau t K)$ and $D = \mathrm{diag}(V_1, V_2, \ldots, V_L)$ and

$$P = \begin{bmatrix} 0 & & & & I \\ -I & 0 & & & \\ & -I & 0 & & \\ & & \ddots & \ddots & \\ & & & -I & 0 \end{bmatrix}.$$

Note that $\det[P] = 1$. It can be verified that for the 1-D or 2-D rectangle lattice, i.e., $K = K_x$ or $K = K_x \otimes I + I \otimes K_y$ as defined in B.2, the matrix

$$\Pi = I \otimes \Pi_x \quad \text{(1-D)}$$

or

$$\Pi = I \times \Pi_x \otimes P_y \quad \text{(2-D)}$$

anti-commutes with A and commutes with D, i.e.,

$$\Pi A + A \Pi = 0, \quad \Pi D - D \Pi = 0.$$

Furthermore, it satisfies

$$\Pi = P \Pi P^T.$$

Then by Theorem 4.4, the determinants of M_+ and M_- are related by

$$\det[M_-] = e^{-\sum_{\ell=1}^{L} \operatorname{Tr}(V_\ell)} \cdot \det[M_+].$$

and the Green's functions $G^\sigma = M_\sigma^{-1}$ satisfy the relation

$$G^- = I - \Pi (G^+)^T \Pi.$$

Remark 4.4. Besides the 1-D and 2-D rectangle lattices, namely the lattice structure matrices K_x and K as defined in B.2, are there other types of lattices (and associated structure matrices K) such that we can apply Theorems 4.4 to establish the relationships between the inverses and determinants in the DQMC? It is known that for the honeycomb lattices, it is true, but for the triangle lattices, it is false. A similar question is also valid for the HQMC. Indeed, it works on any "bipartite" lattice, i.e., any geometry in which sites divides into two disjoint sets $\mathcal{A}$ and $\mathcal{B}$ and K connects sites in $\mathcal{A}$ and $\mathcal{B}$ only.

B.4 Some identities of matrix exponentials

1. In general, $e^{A+B} \neq e^A e^B$, and $e^A e^B \neq e^B e^A$.
2. If A and B commute, namely $AB = BA$, then $e^{A+B} = e^A e^B = e^B e^A$.
3. $(e^A)^{-1} = e^{-A}$
4. $e^{P^{-1}AP} = P^{-1} e^A P$
5. $(e^A)^H = e^{A^H}$ for every square matrix A
   e^A is Hermitian if A is Hermitian
   e^A is unitary if A is skew-Hermitian
6. $\det e^A = e^{\operatorname{Tr} A}$ for every square matrix A
7. $e^{A \otimes I + I \otimes B} = e^A \otimes e^B$

Acknowledgments

This paper first assembled for the lecture notes used in the Shanghai Summer School of Mathematics held in 2006. We are exceedingly grateful to Professor Tatsien Li (Li Daqian) of Fudan University and Professor Thomas Yizhou Hou of California Institute of Technology for inviting us to present this work at this summer school and providing us an opportunity to put the materials together in the first place.

References

[1] V. I. Arnold. Mathematical Methods of Classical Mechanics, second edition. Springer-Verlag, New York, 1989.

[2] R. Blankenbecler, D. J. Scalapino and R. L. Sugar. Monte Carlo calculations of coupled Boson-fermion systems I. Phys. Rev. D, 24(1981), pp.2278-2286.

[3] R. P. Feynman and A. R. Hibbs. Quantum Mechanics and Path Integrals. McGraw-Hill, New York, 1965

[4] E. Hairer, C. Lubich and G. Wanner. Geometric numerical integration illustrated by the Störmer-Verlet method. Acta Numerica, 12(2003), pp.399-450.

[5] J. E. Hirsch. Two-dimensional Hubbard model: numerical simulation study. Phy. Rev. B, 31(1985), pp.4403-4419.

[6] J. E. Hirsch. Hubbard-Stratonovich transformation for fermion lattice models. Phy. Rev. B, 28(1983), pp.4059-4061.

[7] J. E. Hirsch. Erratum: Discrete Hubbard-Stratonovich transformation for fermion lattice models. Phy. Rev. B, 29(1984), p.4159.

[8] J. Hubbard. Electron correlations in narrow energy bands. Proc. Roy. Soc. London, A, 276(1963), pp.238-257.

[9] J. Hubbard. Electron correlations in narrow energy bands III: an Improved solution. Proc. Roy. Soc. London, A, 281(1964), pp.401-419.

[10] Jun S. Liu. Monte Carlo Strategies in Scientific Computing. Springer, 2001.

[11] E. Y. Loh Jr. and J. E. Gubernatis. Stable numerical simulations of models of interacting electrons. In Electronic Phase Transition edited by W. Hanks and Yu. V. Kopaev. Elsevier Science Publishers B.V., 1992, pp.177–235.

[12] R. K. Pathria. Statistical Mechanics, second edition. Elsevier, 2001.

[13] R. Schumann and E. Heiner. Transformations of the Hubbard interaction to quadratic forms. Phy. Let. A, 134(1988), 202-204.

[14] D. J. Scalapino and R. L. Sugar. Monte Carlo calculations of coupled Boson-fermion systems II. Phys. Rev. B, 24(1981), pp.4295-4308.

[15] R. T. Scalettar, D. J. Scalapino, R. L. Sugar, and D. Toussaint. Hybrid molecular-dynamics algorithm for the numerical simulation of many-electron systems. Phy. Rev. B, 36(1987), pp.8632-8640.

[16] R. T. Scalettar, D. J. Scalapino and R. L. Sugar. New algorithm for the numerical simulation of fermions. Phy. Rev. B, 34(1986), pp.7911-7917.

[17] J. P. Wallington and J. F. Annett. Discrete symmetries and transformations of the Hubbard model. Phy. Rev. B, 58(1998), pp.1218-1221.

[18] T. Hanaguri, C. Lupien, Y. Kohsaka, D.-H. Lee, M. Azuma, M. Takono, H. Takagi and J. C. Davis. A 'checkerboard' electronic crystal state in lightly hole-doped $Ca_{2-x}Na_xCuO_2CI_2$. Nature, 430(2004), pp.1001-1005.

[19] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. SIAM Review, 45(2003), pp.3-49.

[20] R. S. Varga. *Matrix iterative analysis.* Prentice-Hall, Englewood Cliffs, 1962. 2nd ed., Springer, Berlin/Heidelberg, 2000.

[21] S. R. White and D. J. Scalapino. Density matrix renormalization group study of the striped phase in the 2D t-J model. Phys. Rev. Lett. 80, pp.1272–1275, (1998).

[22] B. L. Buzbee, G. H. Golub, and C. W. Nielson. On direct methods for solving Poisson's equations. *SIAM J.Numer. Anal.*, 7(1970), pp.627–656.

[23] G. Fairweather and I. Gladwell. Algorithms for almost block diagonal linear systems. *SIAM Review*, 46(2004), pp.49–58.

[24] N. Higham, Accuracy and Stability of Numerical Algorithms (Second Edition). *SIAM*, 2002

[25] B. Philippe, Y. Saad and W. J. Stewart. Numerical methods in Markov chain modeling. *Operations Research*, 40(1992), pp.1156–1179.

[26] W. J. Stewart. *Introduction to the numerical solution of Markov chains.* Princeton University Press, 1994.

[27] G. J. Tee. An application of p-cyclic matrices for solving periodic parabolic problems. *Numer. Math.*, 6(1964), pp.142–159.

[28] U. M. Ascher, R. M. M. Mattheij and R. D. Russell. *Numerical solution of boundary value problems for ordinary differential equations.* Prentice-Hall, Englewood Cliffs, 1988.

[29] S. J. Wright. Stable parallel algorithms for two-point boundary value problems. *SIAM J. Sci. Statist. Comput.*, 13(1):742–764, 1992.

[30] S. J. Wright. A collection of problems for which gaussian elimination with partial pivoting is unstable. *SIAM J. Sci. Statist. comput.*, 14(1993), pp.231–238.

[31] M. Ajiz and A. Jennings. A robust incomplete choleski-conjugate gradient algorithm. *Inter. J. Numer. Meth. Engrg.*, 20(1984), pp.949–966.

[32] M. Arioli and and D. Ruiz. A Chevyshev-based two-stage iterative method as an alternative to the direct solution of linear systems. *Technical Report, RAL-TR-2002-021, Rutherford Appleton Laboratory.* 2000

[33] O. Axelsson and L. Y. Kolotilina. Diagonally compensated reduction and related preconditioning methods. *Numer. Linear Algebra Appl.*, 1(1994), pp.155–177.

[34] Z. Bai and R. T. Scalettar. private communication, 2005

[35] M. Bollhoefer and V. Mehrmann. A New approach to algebraic multilevel methods based on sparse approximate inverses. Preprint, Numerische Simulation auf Massiv Parallelen Rechnern. (1999).

[36] B. Capentieri, I. S. Duff, and L. Giraud. A class of spectral two-level preconditioners. *SIAM J. Scient. Comp.*, 25(2003), pp.749–765.

[37] V. Eijkhout. On the existence problem of incomplete factorization methods. LAPACK Working Note 144, UT-CS-99-435, Computer Science Department, University of Tennessee. 1991

[38] G. H. Golub and C. F. van Loan. *Matrix Computations*, third edition. Johns Hopkins University Press, 1996.

[39] A. Jennings and G. M. Malik. Partial elimination. *J. Inst. Math. Appl.*, 20(1977), pp.307–316.

[40] I. E. Kaporin. High quality preconditioning of a general symmetric positive definite matrix based on its $u^t u + u^t r + r^t u$-decomposition. *Numer. Lin. Alg. Appl.*, 5(1998), pp.483–509.

[41] D. S. Kershaw. The incomplete Cholesky conjugate gradient method for the iterative solution of systems of linear equations. *J. of Comp. Phys.*, 26(1978), pp.43–65.

[42] R. Lancaze, A. Morel, B. Petersson and J. Schroper. An investigation of the 2D attractive Hubbard model. *Eur. Phys. J. B.*, 2(1998), pp.509–523.

[43] T. A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comp.*, 150(1980), pp.473–497.

[44] J. Meijerinkand and H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix. *Math. Comp.*, 137 (1977), pp.134–155.

[45] R. A. Nicolaides. Deflation of conjugate gradients with application to boundary value problems. *SIAM J. Numer. Anal.*, 24 (1987), pp.355–365.

[46] A. Padiy, O. Axelsson and B. Polman. Generalized augmented matrix preconditioning approach and its application to iterative solution of ill-conditioned algebraic systems. *SIAM J. Matrix Anal. Appl.*, 22 (200), pp.793–818.

[47] PETSc: Portable, Extensible toolkit for scientific computation. http://www-unix.mcs.anl.gov/petsc/petsc-2/.

[48] R. T. Scalettar, D. J. Scalapino and R. L. Sugar. A new algorithm for numerical simulation of fermions. *Phys. Rev. B,* 34(1986), pp.7911–7917.

[49] R. T. Scalettar, D. J. Scalapino, R. L. Sugar and D. Tousaint. A hybrid-molecular dynamics algorithm for the numerical simulation of many electron systems. *Phys. Rev. B*, 36(1987), pp.8632–8641.

[50] R. B. Schnabel and E. Eskow. A new modified Cholesky factorization. *SIAM J. Sci. Comput.*, 11(1990), pp.1136–1158.

[51] M. Tismenetsky. A new preconditioning technique for solving large sparse linear systems. *Lin. Alg. Appl.*, 154–156(1991), pp.331–353.

[52] H. A. van der Vorst. Iterative solution methods for certain sparse linear systems with a non-symmetric matrix arising from PDE-problems. *J. of Comp. Phys.*, 44(1981), pp.1–19.

[53] R. S. Varga, E. B. Saff and V. Mehrmann. Incomplete factorizations of matrices and connections with H-matrices. *SIAM J. Numer. Anal.*, 17(1980), pp.787–793.

[54] G. W. Stewart. Modifying pivot elements in Gaussian elimination. *Math. Comp.*, 28(1974), pp.537–542.

[55] E. L. Yip. A note on the stability of solving a rank-p modification of a linear system by the Sherman-Morrison-Woodbury formula. *SIAM J. Sci. Stat. Comput.*, 7(1986), pp.507-513.