
Lecture Notes on Advances of Numerical Methods for Multi-Scale Quantum Simulations

ZHAOJUN BAI
WENBIN CHEN
RICHARD SCALETTAR
ICHIKARO YAMAZAKI

Version 06.06.12

Preface

One of the core problems in materials science is how do the interactions between electrons in a solid give rise to properties like magnetism, superconductivity, and metal-insulator transitions? Our ability to solve this central question in quantum statistical mechanics 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 lecture, we begin with an introduction of Hubbard Hamiltonian and quantum Monte Carlo simulations. The Hubbard Hamiltonian is a simple and effective model that has successfully captured many of the qualitative features of materials. We then present numerical methods for the computational kernels of quantum Monte Carlo simulations, Parts of the lecture notes contents some of our recent work.

This lecture note, assembled in a limited time, is still a work in progress. Your comments and suggestions are welcome.

Our research was partially supported by the National Science Foundation ITR Grant DMR-0313390.

Zhaojun Bai

`bai@cs.ucdavis.edu`

Wenbin Chen

`wbchen@fudan.edu.cn`

Richard Scalettar

`scalettari@physics.ucdavis.edu`

Ichitaro Yamazaki

`yamazaki@cs.ucdavis.edu`

UC Davis and Fudan

June 12, 2006

Contents

1	Hubbard model and QMC simulations	1
1.1	Introduction	1
1.2	Hubbard model	1
1.2.1	Hubbard model when no hopping	3
1.2.2	Hubbard model when no Coulomb interaction	5
1.3	Approximation of the partition function using discrete Hubbard-Stratonovich transformation	8
1.4	Determinant QMC	11
1.4.1	Physical measurements	12
1.5	Approximation of the partition function using continuous Hubbard-Stratonovich transformation	13
1.6	Hybrid QMC	16
1.7	Physical measurements	20
1.7.1	Equal-time Green's function \bar{G}	21
1.7.2	Two-particle Green's function	21
1.7.3	Equal-time density-density correlations	21
1.7.4	Equal-time spin-spin correlations	22
1.7.5	Staggered Susceptibility	23
1.8	Pseudo-code of HQMC	23
2	Hubbard matrix analysis	26
2.1	Hubbard matrices	26
2.2	Basic Properties	29
2.3	Eigen-distribution	33
2.4	Condition numbers	36
2.5	Condition number of $M^{(k)}$	38
3	Self-adaptive direct linear system solvers	43
3.1	Introduction	43
3.2	Block cyclic reduction	44
3.3	Block orthogonal factorization method	47
3.4	A hybrid method	49
3.5	Self-adaptive reduction factor k	49
3.6	Numerical experiments	51

4	Preconditioned iterative solvers	57
4.1	Introduction	57
4.2	Cholesky factorization	58
4.3	Incomplete Cholesky factorization	59
4.4	Robust Incomplete Cholesky preconditioners	63
4.4.1	RIC1	64
4.4.2	RIC2.	67
4.4.3	RIC3	69
4.5	Numerical results	71
5	The Monte-Carlo method	78
5.1	Introduction	78
5.2	Monte Carlo simulation by Markov chain	83
5.3	MCMC for multi-dimensional integrals	86
5.4	γ -Metropolis algorithm in the DQMC	88

Lecture 1

Hubbard model and QMC simulations

1.1 Introduction

The Hubbard model is a simple and 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) simulation to study many-electron systems. Subsequent lectures will describe computational kernels of the QMC simulation.

1.2 Hubbard model

The Hubbard model is defined by the Hamiltonian:

$$\mathcal{H} = \mathcal{H}_K + \mathcal{H}_\nu + \mathcal{H}_V,$$

where \mathcal{H}_K , \mathcal{H}_ν and \mathcal{H}_V stand for kinetic energy, chemical energy and potential energy, respectively, and are defined as the followings:

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

and

- i and j label the spatial sites of the lattice,
- the operators $c_{i\sigma}^\dagger$ and $c_{i\sigma}$ are the creation and annihilation operators for electrons located on the i th lattice site with z component of spin $\sigma = +(\text{up})$ or $-(\text{down})$, respectively.
- The operators $n_{i\sigma} = c_{i\sigma}^\dagger c_{i\sigma}$ are the number operators which count the number of electrons of spin σ on site i .

- The hopping parameter t controls the kinetic energy of the electrons, and is determined by the overlap of atomic wave functions on neighboring sites,
- $\langle i, j \rangle$ represents the electrons only hopping to nearest neighboring sites.
- The term $Un_{i+}n_{i-}$ represents an energy cost U if the site i has two electrons on it (nonzero only if both n_{i+} and $n_{i-} = 1$), it therefore describes a local repulsion between electrons.

The expected value of a physical observable \mathcal{E} , such as the density-density correlation, spin-spin correlation or the magnetic susceptibility χ (see section 1.7), is given by

$$\langle \mathcal{E} \rangle = \text{Tr} \left(\mathcal{E} \frac{e^{-\beta \mathcal{H}}}{Z} \right), \quad (1.2.1)$$

where Z is the so-called partition function and is defined by

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

and $\beta = \frac{1}{k_B T}$ is proportional to the inverse of the temperature T and k_B is Boltzmann's constant. Note that "Tr" is a trace over the Hilbert space describing all the possible occupation states of the lattice, i.e.,

$$\text{Tr}(e^{-\beta \mathcal{H}}) = \sum_i \langle \psi_i | e^{-\beta \mathcal{H}} | \psi_i \rangle \quad (1.2.2)$$

where $|\psi_i\rangle$ are any orthonormal basis of the Hilbert space. The trace does not depend on the choice of the basis. One possible basis choice is the "occupation number basis" described below.

In a classical problem where $H = E$ is the energy, a real number, then $\exp(-\beta E)/Z$ is the probability. In a quantum mechanics problem, as we shall see, we will need to recast the operator $\exp(-\beta H)$ in to a number. The "path integral" method to do this was introduced by Feynman.

Remark 1 The creation operators $c_{i\sigma}^\dagger$ and the annihilation operators $c_{i\sigma}$ have the anticommutation relations:

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

where the anticommutator of two operators $\{A, B\}$ is defined to be $AB + BA$.

If we choose $j = l$ 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. 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 2 According to Pauli principle, for electronic particles, there are four states at every site:

$$\begin{aligned} |\cdot\rangle & \text{ no particle,} \\ |\uparrow\rangle & \text{ one particle with spin up,} \\ |\downarrow\rangle & \text{ one particle with spin down,} \\ |\uparrow\downarrow\rangle & \text{ two particles with different spin directions.} \end{aligned}$$

Therefore the dimension of the Hilbert space is 4^N .

A different notation is also used to describe the states if the spin-direction can be omitted:

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

The actions of the basic operators on the states are (here we omit the i, σ indices):

$$\begin{aligned} c : \quad |0\rangle = 0, \quad |1\rangle = |0\rangle, \\ c^\dagger : \quad |0\rangle = |1\rangle, \quad |1\rangle = 0, \\ n : \quad |0\rangle = 0, \quad |1\rangle = |1\rangle. \end{aligned} \tag{1.2.3}$$

The last equation of (1.2.3) shows that the states $|0\rangle$ and $|1\rangle$ are the eigen-states of the operator n .

These eigenenergies immediately illustrate one key aspect of the physics of the Hubbard model: The single occupied states $|\uparrow\rangle$ and $|\downarrow\rangle$ are lower in energy by $U/2$ (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 3 The operator $n_\uparrow n_\downarrow$ describes the potential energy of the two electrons with different spin directions if the two electrons are at some site:

$$\begin{aligned} n_\uparrow n_\downarrow : \quad |\cdot\rangle = 0, \quad |\uparrow\rangle = 0, \\ \quad \quad \quad |\downarrow\rangle = 0, \quad |\uparrow\downarrow\rangle = |\uparrow\downarrow\rangle. \end{aligned} \tag{1.2.4}$$

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

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

therefore if there is one particle on the $i+1$ th site, and no particle on the i th site, the operator $c_i^\dagger c_{i+1}$ will destroy the particle on the $i+1$ th site and create one particle on the i th site, so we say the electron hops from site $i+1$ to site i after the action of the operator $c_i^\dagger c_{i+1}$.

1.2.1 Hubbard model when no hopping

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

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

Note that the states $|0\rangle, |\uparrow\rangle, |\downarrow\rangle, |\uparrow\downarrow\rangle$ are eigen-states of the operator n_σ , and therefore also the eigen-states of the Hamiltonian \mathcal{H} :

$$\begin{aligned}\mathcal{H}|0\rangle &= \frac{U}{4}|0\rangle, & \mathcal{H}|\uparrow\rangle &= \frac{U}{4}|\uparrow\rangle - \left(\mu + \frac{U}{2}\right)|\uparrow\rangle, \\ \mathcal{H}|\downarrow\rangle &= \frac{U}{4}|\downarrow\rangle - \left(\mu + \frac{U}{2}\right)|\downarrow\rangle, & \mathcal{H}|\uparrow\downarrow\rangle &= \frac{U}{4}|\uparrow\downarrow\rangle - 2\mu|\uparrow\downarrow\rangle.\end{aligned}$$

Note the eigen-states are orthogonal and normalized, and based on the eigen-states $\psi_i = |0\rangle, |\uparrow\rangle, |\downarrow\rangle, |\uparrow\downarrow\rangle$, the Hamiltonian \mathcal{H} can be diagonalized :

$$\mathcal{H} \longrightarrow (\langle\psi_i|\mathcal{H}|\psi_j\rangle) = \begin{bmatrix} \frac{U}{4} & & & \\ & \frac{U}{4} - \left(\mu + \frac{U}{2}\right) & & \\ & & \frac{U}{4} - \left(\mu + \frac{U}{2}\right) & \\ & & & \frac{U}{4} - 2\mu \end{bmatrix}. \quad (1.2.5)$$

Hence the operator $e^{-\beta\mathcal{H}}$ can also be diagonalized:

$$e^{-\beta\mathcal{H}} \longrightarrow e^{-\frac{U\beta}{4}} \text{diag} \left(1, e^{\beta(U/2+\mu)}, e^{\beta(U/2+\mu)}, e^{2\mu\beta} \right). \quad (1.2.6)$$

The partition function Z is computed by

$$Z = \text{Tr}(e^{-\beta\mathcal{H}}) = \sum_i \langle\psi_i|e^{-\beta\mathcal{H}}|\psi_i\rangle = e^{-\frac{U\beta}{4}} \left(1 + 2e^{(\frac{U}{2}+\mu)\beta} + e^{2\mu\beta} \right). \quad (1.2.7)$$

In this occupation number basis, the operators $\mathcal{H}e^{-\beta\mathcal{H}}, n_\uparrow e^{-\beta\mathcal{H}}, n_\downarrow e^{-\beta\mathcal{H}}$ and $n_\uparrow n_\downarrow e^{-\beta\mathcal{H}}$ are

$$\begin{aligned}\mathcal{H}e^{-\beta\mathcal{H}} &\longrightarrow e^{-\frac{U\beta}{4}} \text{diag} \left(\frac{U}{4}, \left(-\mu - \frac{U}{4}\right)e^{\beta(U/2+\mu)}, \left(-\mu - \frac{U}{4}\right)e^{\beta(U/2+\mu)}, \left(\frac{U}{4} - 2\mu\right)e^{2\mu\beta} \right) \\ n_\uparrow e^{-\beta\mathcal{H}} &= e^{-\beta\mathcal{H}} \{0, 1, 0, 1\} \longrightarrow e^{-\frac{U\beta}{4}} \text{diag} \left(0, e^{\beta(U/2+\mu)}, 0, e^{2\mu\beta} \right) \\ n_\downarrow e^{-\beta\mathcal{H}} &= e^{-\beta\mathcal{H}} \{0, 0, 1, 1\} \longrightarrow e^{-\frac{U\beta}{4}} \text{diag} \left(0, 0, e^{\beta(U/2+\mu)}, e^{2\mu\beta} \right) \\ n_\uparrow n_\downarrow e^{-\beta\mathcal{H}} &= e^{-\beta\mathcal{H}} \{0, 0, 0, 1\} \longrightarrow e^{-\frac{U\beta}{4}} \text{diag} \left(0, 0, 0, e^{2\mu\beta} \right)\end{aligned}$$

Therefore, the trace of the operators are given by the following:

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

By the definition, the following physical observables 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:

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

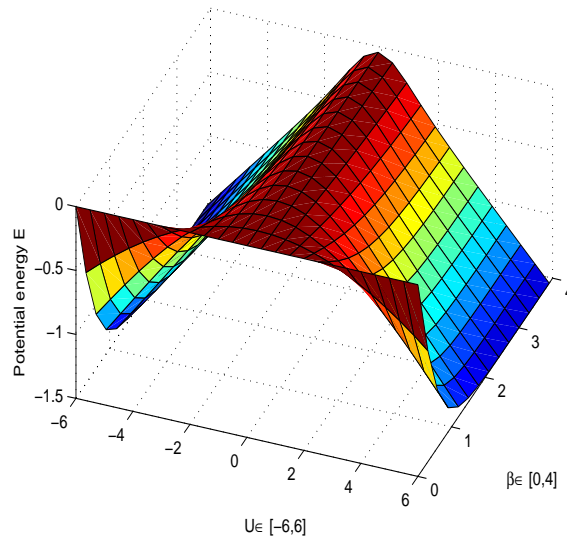


Figure 1.1: Potential energy E for $t = 0, \mu = 0$

In particular, when there is no chemical potential, i.e., $\mu = 0, \rho = 1$. One refers to this as “half-filling” because the density is one-half the maximal possible value.

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

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

In particular, when there is no chemical potential, i.e., $\mu = 0, \rho = 1$. $E = \frac{U}{4} - \frac{U}{2(1+e^{-\frac{U}{2}})}$.

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

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

In particular, when there is no chemical potential, i.e., $\mu = 0$, $\langle n_{\uparrow}n_{\downarrow} \rangle = \frac{1}{2(1+e^{\frac{U}{2}})}$. Note that as U or $\beta = 1/T$ increase the double occupancy goes to zero.

1.2.2 Hubbard model when no Coulomb interaction

When there is no Coulomb interaction ($U = 0$), the two spin spaces are independent of each other (\mathcal{H} breaks into two separate pieces in which \uparrow and \downarrow terms never occur together). In this

case we can consider each spin separately. If the spin is omitted, the Hamiltonian \mathcal{H} becomes:

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

Since the number operator $n_i = c_i^\dagger c_i$, the Hamiltonian \mathcal{H} can be rewritten as a bilinear form:

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

where I is the identity matrix, the matrix K describes the hopping relationship $\langle i, j \rangle$, and

$$\vec{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_N \end{bmatrix}, \quad \vec{c}^\dagger = (c_1^\dagger, c_2^\dagger, \dots, c_N^\dagger).$$

For one dimensional (1D) lattice of N_x sites ,

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

This form of K incorporates “periodic boundary conditions (pbc)” in which sites 1 and N_x are connected by t . The use of pbc reduce finite size effects.

For two dimensional (2D) rectangle lattice of $N_x \times N_y$ sites:

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

where I_x and I_y are identity matrices with dimension N_x and N_y .

The matrix K has the eigen-decomposition:

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

where $\Lambda = \text{diag}(\lambda_k)$ is a diagonal matrix, and λ_k are the eigenvalues of the matrix K :

$$\begin{aligned} \lambda_k &= 2 \cos \theta_{kx}, & \text{for 1D lattice} \\ \lambda_k &= 2(\cos \theta_{kx} + \cos \theta_{ky}), & \text{for 2D lattice} \end{aligned}$$

where $\theta_{kx} = \frac{2k_x \pi}{N_x}$, $k_x = 0, 1, \dots, N_x - 1$ and $\theta_{ky} = \frac{2k_y \pi}{N_y}$, $k_y = 0, 1, \dots, N_y - 1$.

Let $\vec{c} = F \vec{c}$, $\vec{c}^\dagger = (F \vec{c})^\dagger$, then the Hamiltonian \mathcal{H} can also be diagonalized:

$$\mathcal{H} = \vec{c}^\dagger (-t\Lambda - \mu I) \vec{c} = \sum_k \epsilon_k \tilde{n}_k, \quad (1.2.8)$$

where $\epsilon_k \equiv -t\lambda_k - \mu$, and \tilde{n}_k is a new number operator: $\tilde{n}_k = \vec{c}_k^\dagger \vec{c}_k$.

Remark 4 *It can be shown that the operators 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, but the interaction term in the Hubbard model is fairly complex when written in momentum space.*

Therefore, the partition function Z can be written in the form

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

Subsequently, we have following expressions for physical observables of interest.

1. the density ρ (per site average occupation) means the average occupation of the each sites:

$$\rho = \langle n \rangle = \frac{1}{N} \sum_{i=1}^N \langle n_{i\uparrow} + n_{i\downarrow} \rangle = \sum_k (1 + e^{\beta\epsilon_k})^{-1}. \quad (1.2.9)$$

2. The energy E is:

$$E = \langle \mathcal{H} \rangle = \frac{1}{N} \sum_k \frac{\epsilon_k + \mu}{e^{\beta\epsilon_k} + 1}. \quad (1.2.10)$$

For the sake of completeness, we will also write down here the expression for the ‘Green’s function’ in the $U = 0$ case. This quantity plays a key role in the discussion of the matrices arising in the simulation which follows.

$$G_{l,n} = \langle c_l c_n^\dagger \rangle = \frac{1}{N} \sum_k e^{ik \cdot (n-l)} (1 - f_k). \quad (1.2.11)$$

Here $f_k = 1/[1 + e^{\beta(\epsilon_\sigma(k) - \mu)}]$. Notice that G is just a function of the difference $n - l$. This is a consequence of the fact that the Hamiltonian is translationally invariant, that is, with periodic boundary conditions, 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 $k = (\pi, \pi)$.)

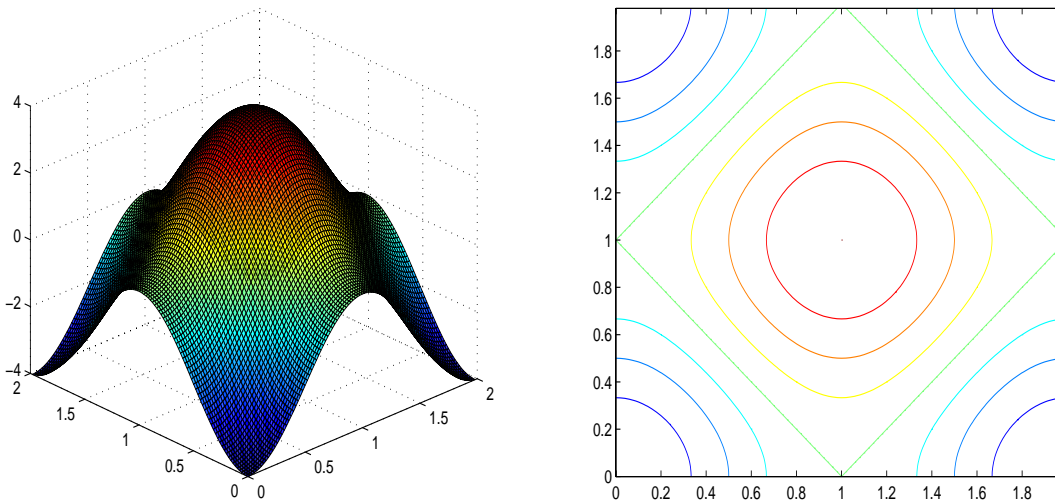


Figure 1.2: Left: ϵ_k for $U = 0$ and $\mu = 0$. Right: the contour plot of ϵ_k

1.3 Approximation of the partition function using discrete Hubbard-Stratonovich transformation

For simplicity, we will consider the chemical potential $\mu = 0$ which corresponds to the important half-filled-band case. 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.

Since the operators \mathcal{H}_K and \mathcal{H}_V do not commute, we apply the Trotter decomposition, i.e., divide β into L smaller subintervals $\tau = \frac{\beta}{L}$. Then the partition function Z is approximated by

$$Z = \text{Tr} \left(e^{-\beta \mathcal{H}} \right) = \text{Tr} \left(\prod_{\ell=1}^L e^{-\tau \mathcal{H}} \right) = \text{Tr} \left(\prod_{\ell=1}^L e^{-\tau \mathcal{H}_K} e^{-\tau \mathcal{H}_V} \right) + O(\tau^2). \quad (1.3.12)$$

Note that $e^{-\tau \mathcal{H}_K}$ are quadratic in the fermion operators.

For each factor of the L terms $e^{-\tau \mathcal{H}_V}$, we introduce N Hubbard-Stratonovich variables (The collection of all these variables is often called the ‘‘Hubbard-Stratonovich field’’), one for each of the spatial sites:

$$\begin{aligned} e^{-\tau \mathcal{H}_V} &= e^{-U\tau \sum_{i=1}^N (n_{i+} - \frac{1}{2})(n_{i-} - \frac{1}{2})} \\ &= \prod_{i=1}^N e^{-U\tau (n_{i+} - \frac{1}{2})(n_{i-} - \frac{1}{2})} \\ &= \prod_{i=1}^N \left(\frac{1}{2} e^{-\frac{U\tau}{4}} \sum_{h_i = \pm 1} e^{\nu h_i (n_{i+} - n_{i-})} \right). \end{aligned} \quad (1.3.13)$$

Here for the last equality, we have used the following discrete Hubbard-Stratonovich transformation, which replaces the the interaction (potential energy) terms $n_{i+} n_{i-} = c_{i+}^\dagger c_{i+} c_{i-}^\dagger c_{i-}$ by

quadratic ones.

Lemma 5 (*Discrete Hubbard-Stratonovich transformation, [1, 2]*). *If $U > 0$, then*

$$e^{-U(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})} = C_1 \sum_{h_i=\pm 1} e^{\nu h_i(n_{i+}-n_{i-})}, \quad (1.3.14)$$

where $\cosh \nu = e^{\frac{U}{2}}$ and the constant $C_1 = \frac{1}{2}e^{-\frac{U}{4}}$.

PROOF. On every site i , the particles have four states: $|\cdot\rangle, |\uparrow\rangle, |\downarrow\rangle$ and $|\uparrow\downarrow\rangle$, and the following lists the actions of the operators $(n_{i+} - \frac{1}{2})(n_{i-} - \frac{1}{2})$ and $(n_{i+} - n_{i-})$.

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

Now for the operator $e^{-U(n_{i+}-\frac{1}{2})(n_{i-}-\frac{1}{2})}$:

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

and

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

And for the operator $\frac{1}{2}e^{-\frac{U}{4}} \sum_{h_i=\pm 1} e^{\nu h_i(n_{i+}-n_{i-})}$:

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

and

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

Therefore if we let

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

then the discrete Hubbard-Stratonovich transformation holds. \square

Remark 6

- Note that in the proof, U is required to be positive, otherwise no real number ν exists such that $\cosh \nu = e^{\frac{U}{2}}$.
- For $U < 0$, the Hubbard model is called the attractive Hubbard model. A similar discrete Hubbard-stratonovich transformation also exists, see [3, 4].
- Even for the discrete version, the transformation is not unique. There exists a class of transformations, see [6, 10].

Let us continue to rewrite the term $e^{-\tau\mathcal{H}_V}$ from the expression (1.3.13). For the sake of simplicity in expression, consider $N = 2$,

$$\begin{aligned} e^{-\tau\mathcal{H}_V} &= (C_1)^2 \left(\sum_{h_1=\pm 1} e^{\nu h_1(n_{1+}-n_{1-})} \right) \left(\sum_{h_2=\pm 1} e^{\nu h_2(n_{2+}-n_{2-})} \right) \\ &= (C_1)^2 \sum_{h_1=\pm 1, h_2=\pm 1} e^{\sum_{i=1}^2 \nu h_i(n_{i+}-n_{i-})} \\ &\equiv (C_1)^2 \text{Tr}_h e^{\sum_{i=1}^2 \nu h_i(n_{i+}-n_{i-})}. \end{aligned}$$

In the last equation, a new notation Tr_h is introduced, which represents the sum for different $h_i = \pm 1$.

By the definition of the notation Tr_h , for general N , we have

$$e^{-\tau\mathcal{H}_V} = (C_1)^N \text{Tr}_h e^{\sum_{i=1}^N \nu h_i(n_{i+}-n_{i-})} \quad (1.3.15)$$

$$= (C_1)^N \text{Tr}_h \left(e^{\sum_{i=1}^N \nu h_i n_{i+}} e^{\sum_{i=1}^N -\nu h_i n_{i-}} \right) \quad (1.3.16)$$

$$= (C_1)^N \text{Tr}_h (e^{\mathcal{H}_{V_+}} e^{\mathcal{H}_{V_-}}), \quad (1.3.17)$$

where the operators \mathcal{H}_{V_+} and \mathcal{H}_{V_-} correspond to spin-up and spin-down, respectively.

Therefore in every time slice ℓ , the operator $e^{-\tau\mathcal{H}_V}$ can be transformed as quadratic forms:

$$e^{-\tau\mathcal{H}_V} = (C_1)^N \text{Tr}_{h_\ell} (e^{\mathcal{H}_{V_{\ell+}}} e^{\mathcal{H}_{V_{\ell-}}}). \quad (1.3.18)$$

Denote

$$V_\ell(h_\ell) = \begin{bmatrix} \nu h_{\ell,1} & & \\ & \ddots & \\ & & \nu h_{\ell,N} \end{bmatrix}.$$

Then the operators $\mathcal{H}_{V_{\ell+}}$ and $\mathcal{H}_{V_{\ell-}}$ can be written as

$$\mathcal{H}_{V_{\ell+}} = \sum_{i=1}^N \nu h_{\ell,i} n_{i+} = \vec{c}_+^\dagger V_\ell(h_\ell) \vec{c}_+,$$

$$\mathcal{H}_{V_{\ell-}} = -\sum_{i=1}^N \nu h_{\ell,i} n_{i-} = -\vec{c}_-^\dagger V_\ell(h_\ell) \vec{c}_-,$$

Note that the Hubbard-Stratonovich variable $h_{\ell,i}$ has two indices, space i and ‘‘imaginary-time’’ ℓ .

Finally, after interchanging the traces and applying the identity in (1.3.21), Z is rewritten as

$$Z = (C_1)^{NL} \text{Tr}_h \text{Tr} \left(\prod_{\ell=1}^L e^{-\tau\mathcal{H}_{K_+}} e^{\mathcal{H}_{V_{\ell+}}} \right) \left(\prod_{\ell=1}^L e^{-\tau\mathcal{H}_{K_-}} e^{\mathcal{H}_{V_{\ell-}}} \right) \quad (1.3.19)$$

$$= (C_1)^{NL} \text{Tr}_h \det M_+(h) \det M_-(h). \quad (1.3.20)$$

where $M_\sigma(h) = I + B_{L,\sigma}B_{L-1,\sigma}\cdots B_{1,\sigma}$ and $\sigma = \pm$. The $N \times N$ matrix $B_{\ell,\pm} = e^{t\tau K}e^{\pm V_\ell}$ corresponds to the operator $e^{-\tau\mathcal{H}_K}e^{-\tau\mathcal{H}_V}$. Note that the error in the Trotter decomposition (1.3.12) is omitted.

Note that the equality is based on Hirsch's argument[2]:

if every \mathcal{H}_ℓ is in quadratic form:

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

then

$$Z = \text{Tr}(e^{-\mathcal{H}_1(\sigma)}e^{-\mathcal{H}_2(\sigma)}\dots e^{-\mathcal{H}_L(\sigma)}) = \det(I+e^{-H_1(\sigma)}e^{-H_2(\sigma)}\dots e^{-H_L(\sigma)}). \quad (1.3.21)$$

Remark 7 When $U = 0$, $M_\sigma(h)$ are constant matrices which does not depend on the configuration h , and the Trotter decomposition are exact.

Remark 8 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.

1.4 Determinant QMC

At this stage, the evaluation of the partition function Z has been reduced to a classical monte carlo problem: sum over the possible configurations of the real variables $h = \{h_{\ell,i}\}$ with the Boltzmann weight (the product of the two "fermion determinants):

$$P(h) = \frac{1}{Z_h} \det M_+(h) \det M_-(h),$$

where the partition function Z_h is

$$Z_h = \text{Tr}_h \det M_+(h) \det M_-(h).$$

By continuously updating the configuration h and computing the determinants of $M_\sigma(h)$, the determinant QMC (DQMC) method can be developed.

Before we discuss such a DQMC method, one thing should be understood: How to update a new configuration h' form an old configuration h . The answer for this question is to use the Metropolis-Hasting algorithm, for example see [1, p.111]. Alternatively, one simpler strategy is to try to flip $h_{\ell,i}$ only on one given site (ℓ, i) :

$$h'_{\ell,i} = -h_{\ell,i}, \quad (1.4.22)$$

and then accept or reject the new configuration.

The following is an outline of the DQMC simulation [1].

DQMC

1. Choose an initial configuration $h^{(0)}$. Let $i = 1$, $\ell = 1$.
2. Given current configuration $h^{(n)}$,
 - (a) Try the new configuration h' by single spin-flip sampling: $h'_{\ell,i} = -h_{\ell,i}$.
 - (b) Generate one random number $r \sim \text{Uniform}[0, 1]$ and update

$$h^{(n+1)} = \begin{cases} h', & \text{if } r \leq \min \left\{ 1, \frac{\det(M_+(h')M_-(h'))}{\det(M_+(h)M_-(h))} \right\} \\ h^{(n)}, & \text{otherwise.} \end{cases} \quad (1.4.23)$$

- (c) Go the next site $(\ell, \text{mod}(i+1, N) + 1)$ and repeat updating, and if $i = N$, then $\ell = \text{mod}(\ell + 1, L) + 1$.
 3. At certain measurement step, physical measurements are obtained from the elements of the inverse of $M_\sigma(h^{(n)})$.
-

Remark 9 *By the only one site update at every Monte Carlo step, the block matrices B_i are the same except the rank-one updating for $i = \ell$. This simple observation is very important. Based on this observation, it is possible to compute the Metropolis ratio $\frac{\det(M_+(h')M_-(h'))}{\det(M_+(h)M_-(h))}$ and the inverse of $M_\sigma(h)$ efficiently.*

1.4.1 Physical measurements

By sampling the configurations h , many important physical measurements can be obtained from the single-particle Green's function $G_{ij} = \langle c_i c_j^\dagger \rangle$. Hirsch has shown that the single-particle Green's function can be computed from the inverse of $M(h)$ (here we omit the spin parameter σ):

$$G_{ij} = (M(h))_{ij}^{-1}.$$

By applying Wick's theorem, the two-particle Green's function can be obtained from one-particle Green's function:

$$G_{ijkl} = \langle c_i^\dagger c_j \rangle \langle c_k^\dagger c_l \rangle + \langle c_i^\dagger c_l \rangle \langle c_j^\dagger c_k \rangle.$$

Then at every measurement step, the physical observation can be computed from the Green's function. For example, by using anticommutation relationship:

$$c_i^\dagger c_i = 1 - c_i c_i^\dagger,$$

the density $\rho = \langle n \rangle$ can be obtained by

$$\begin{aligned} \langle n \rangle &= \langle n_\uparrow \rangle + \langle n_\downarrow \rangle = \frac{1}{N} \sum_{i=1}^N (\langle n_{i\uparrow} \rangle + \langle n_{i\downarrow} \rangle) \\ &= \frac{1}{N} \sum_{i=1}^N \left(2 - \langle c_{i\uparrow} c_{i\uparrow}^\dagger \rangle - \langle c_{i\downarrow} c_{i\downarrow}^\dagger \rangle \right) \\ &= 2 - \frac{1}{N} \sum_{i=1}^N (G_{ii+} + G_{ii-}). \end{aligned}$$

Therefore if we sample the configurations N_0 times: $h^{(n)}, j = 1, 2, \dots, N_0$, then the density ρ is estimated by

$$\rho = \langle n \rangle = 2 - \frac{1}{NN_0} \sum_{j=1}^{N_0} \sum_{i=1}^N \left(\left(M_+(h^{(n)}) \right)_{ii}^{-1} + \left(M_-(h^{(n)}) \right)_{ii}^{-1} \right).$$

Note that the dimension of the configuration space is 2^{NL} , therefore, it is impossible to sample directly or to use a simple Monte Carlo method.

1.5 Approximation of the partition function using continuous Hubbard-Stratonovich transformation

The discrete Hubbard-Stratonovich transformation is the most efficient way to do DQMC in its standard form. However, in order to formulate new ways to do DQMC, it is necessary to have variables which are continuous.

The procedure summarized in Section 1.4 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 determinants and inverses of matrices which have a dimension the spatial size of the system, N . Thus the algorithm scales as N^3 . In practice, this means simulations are limited to a few hundred sites. In order to try to circumvent this bottleneck and develop an algorithm which scales better with N we will need to reformulate our problem in a number of ways:

1. Replace the discrete Hubbard Stratonovich field be a continuous one.
2. Express the determinant of the dense N dimensional matrices $M_\sigma(h)$ as Gaussian integrals over larger (NL dimensional) sparse matrices. We shall now describe each of these steps in detail.

Instead of using discrete Hubbard-Stratonovich transformation to approximate the partition function Z , one can also introduce a continuous Hubbard-Stratonovich variable $x_{i,l}$ at each lattice site i and time slice l and then perform the continuous Hubbard-Stratonovich transformation as described in the following lemma. The continuous Hubbard-Stratonovich transformation is based on the Stratonovich identity: for any scalar $a > 0$

$$e^{\frac{1}{2}a^2} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-\frac{1}{2}x^2 - xa} dx.$$

Lemma 10 (*continuous Hubbard-Stratonovich transformation*) For $U > 0$, we have

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

where constant $C_2 = \frac{\frac{1}{2}e^{-\frac{U\tau}{4}}}{\pi^{\frac{1}{2}}}$.

PROOF. First we can verify that

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

Note that $(n_\uparrow - n_\downarrow)^2$ and $n_\uparrow - n_\downarrow$ can be diagonalized based on the eigen-states of the operators n_σ , then the Stratonovich identity still holds if we replace the scalar α by the operator $n_\uparrow - n_\downarrow$:

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

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

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

Combining the above equations, we obtain the continuous Hubbard-Stratonovich identity. \square

Returning to the approximation of the partition function Z , by the continuous Hubbard-Stratonovich identity, we have

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

where the operators $\mathcal{H}_{V_{\ell+}}$ and $\mathcal{H}_{V_{\ell-}}$ correspond to spin-up and spin-down respectively:

$$\begin{aligned} \mathcal{H}_{V_{\ell+}} &= \tau \sum_i (2U)^{\frac{1}{2}} x_{\ell,i} n_{i+} = \vec{c}_+^\dagger V_\ell(h_\ell) \vec{c}_+, \\ \mathcal{H}_{V_{\ell-}} &= -\tau \sum_i (2U)^{\frac{1}{2}} x_{\ell,i} n_{i-} = -\vec{c}_-^\dagger V_\ell(h_\ell) \vec{c}_-, \end{aligned}$$

and the diagonal matrix

$$V_\ell(x_\ell) = (2U)^{\frac{1}{2}} \text{diag}(x_{\ell,1}, x_{\ell,2}, \dots, x_{\ell,N}).$$

Denote $S_B(x) = \tau \sum_{\ell,i} x_{\ell,i}^2$, then similar to the derivation in section 1.3 (see equation (1.3.19)), by the Trotter approximation:

$$\begin{aligned} Z &= \text{Tr} \left(\prod_{\ell=1}^L e^{-\tau \mathcal{H}_K} e^{-\tau \mathcal{H}_V} \right) \\ &= (C_2)^{NL} \int [\delta x] e^{-S_B(x)} \text{Tr} \left(\prod_{\ell=1}^L e^{\mathcal{H}_{\ell+}} e^{\mathcal{H}_{\ell-}} \right) \text{Tr} \left(\prod_{\ell=1}^L e^{\mathcal{H}_{\ell-}} e^{\mathcal{H}_{\ell+}} \right) \\ &= (C_2)^{NL} \int [\delta x] e^{-S_B(x)} \det \left(I + \prod_{\ell=1}^L e^{t\tau K} e^{\tau V_\ell(x_\ell)} \right) \det \left(I + \prod_{\ell=1}^L e^{t\tau K} e^{-\tau V_\ell(x_\ell)} \right) \\ &= (C_2)^{NL} \int [\delta x] e^{-S_B(x)} \det M_+(x) \det M_-(x), \end{aligned}$$

where the submatrices $B_{\ell,\sigma}(x_\ell) = e^{t\tau K} e^{-\sigma\tau V_\ell(x_\ell)}$, and the matrices M_σ are:

$$M_\sigma(x) = I + B_{L,\sigma}(x_L)B_{L-1,\sigma}(x_{L-1})\cdots B_{1,\sigma}(x_1), \quad \sigma = \pm. \quad (1.5.24)$$

Therefore the partition function Z is approximated by:

$$Z = (C_2)^{NL} \int [\delta x] e^{-S_B(x)} \det M_+(x) \det M_-(x). \quad (1.5.25)$$

By particle-hole transformation¹,

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

Then the product of determinants $e^{-S_B(x)} \det M_+(x) \det M_-(x)$ is positive definite, so that it can be used as a Boltzmann weight, and the probability of the configuration is given by

$$P(x) = \frac{1}{Z_x} e^{-S_B(x)} \det M_+(x) \det M_-(x)$$

where the partition function $Z_x = \int [\delta x] e^{-S_B(x)} \det M_+(x) \det M_-(x)$.

We can replace the determinant in the Boltzmann weight $P(x)$ by using the following two facts:

1. Let

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

then

$$\det(M) = \det(I + B_L B_{L-1} \cdots B_1). \quad (1.5.26)$$

The proof can be easily derived based on the following lemma.

Lemma 11 *If a matrix A is a 2×2 block matrix,*

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

then

$$|A| = |A_{22}| |F_{11}| = |A_{11}| |F_{22}|,$$

where $F_{11} = A_{11} - A_{12} A_{22}^{-1} A_{21}$, $F_{22} = A_{22} - A_{21} A_{11}^{-1} A_{12}$.

2. If one matrix A is symmetric and positive definite, then

$$\int e^{-v^T A^{-1} v} dv = (\sqrt{\pi})^{\dim(v)} \det(A^{\frac{1}{2}}),$$

¹See the note on an algebraic version of the particle-hole transformation

Therefore, by introducing two auxiliary scalar fields Φ_+ and Φ_- , we have

$$\det M_\sigma(x) = \det \left((M_\sigma^T(x)M_\sigma(x))^{\frac{1}{2}} \right) = \pi^{-\frac{NL}{2}} \int e^{-\Phi_\sigma (M_\sigma^T(x)M_\sigma(x))^{-1} \Phi_\sigma}, \quad (1.5.27)$$

where $\sigma = \pm$.

Denote $O_\sigma = M_\sigma^T M_\sigma$, combining (1.5.25) and (1.5.27), an approximation of the partition function Z is expressed by the following form

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

where $V(x, \Phi_\sigma) = S_B(x) + \Phi_+^T O_+^{-1} \Phi_+ + \Phi_-^T O_-^{-1} \Phi_-$.

1.6 Hybrid QMC

Now we are in a position to consider how to move the configuration x which satisfies the distribution:

$$P(x, \Phi_\sigma) \propto e^{-V(x, \Phi_\sigma)}.$$

Similar to the DQMC method, one can try to move $\{x_{\ell,i}\}$ at every site i and imaginary time ℓ at every MC step. An alternative popular way is to move the entire configuration x by adding some Gaussian noise

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

where Z_t follows the standard Gaussian distribution. This method is called Langevin-Euler moves, for example, see [1, p.192]. It has the advantage of changing all x simultaneously.

Spurred by the popularity of the molecular dynamics method, Scalettar et al [8] proposed a hybrid method to move x by combining Monte carlo method and molecular dynamics method. Like the Langevin method, all the x change together in a single step.

In the molecular dynamics procedures, another auxiliary field $p = \{p_{\ell,i}\}$ is introduced, based on the identity

$$\int_{-\infty}^{\infty} dp_{\ell,i} e^{-p_{\ell,i}^2} = \sqrt{\pi}.$$

In this case, the partition function Z can be rewritten as

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

The field configurations $\{x, p, \Phi_\sigma\}$ obey a probability distribution proportional to $\exp(-H)$:

$$P(x, p, \Phi_+, \Phi_-) = \frac{1}{Z_H} e^{-H(x, p, \Phi_+, \Phi_-)},$$

where the partition function Z_H is defined as:

$$Z_H = \int [\delta x \delta p \delta \Phi_+ \delta \Phi_-] e^{-H(x, p, \Phi_\sigma)}.$$

Now we move current configuration (x, p, Φ_σ) to a new configuration (x', p', Φ'_σ) by molecular dynamics method. One such an approach is

1. Generate two vectors of Gaussian random number R_σ , each component of which has a probability distribution proportional to $\exp(-R_{i,\sigma}^2)$, and define

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

2. Fix the fields Φ_σ , then move p to a new position, and regard the current (x, p) as an initial value $(x(0), p(0))$, and move $(x(t), p(t))$ to a new configuration $(x(T), p(T))$:

- (a) Set each $p_{\ell,i}$ equal to a Gaussian random number with probability distribution proportional to $\exp(-p_{i,\ell}^2)$.
- (b) Regard (x, p) as an initial values, then move to a new configuration $(x(T), p(T))$ by some MD procedures:

$$\begin{aligned} \dot{p}_{\ell,i} &= -\frac{\partial H}{\partial x_{\ell,i}} = -\frac{\partial V}{\partial x_{\ell,i}}, \\ \dot{x}_{\ell,i} &= \frac{\partial H}{\partial p_{\ell,i}} = 2p_{\ell,i} \end{aligned}$$

Remark 12

1. The fields Φ_σ and p are auxiliary fields, here we first fixe the fields Φ_σ , vary the fields p , and move (x, p) together. It is possible to use different moving order.
2. We can take the measurement step outside the loop or inside the loop, just depends on whether $x(0)$ and $x(T)$ are far away enough.
3. From Liouville's theorem, we move along a trajectory in which both H and the differential volume element in phase space are constant and the system is equilibrium, that is, if the probability distribution of the fields is given by

$$P(x, p, \Phi_\sigma) = \frac{1}{Z_H} e^{-H(x, p, \Phi_\sigma)},$$

then the molecular dynamics steps will keep it in equilibrium.

The leap-frog method is a simple and efficient numerical method to move the configuration $(x(0), p(0))$ to $(x(T), p(T))$ according to the Hamilton's equations:

$$\begin{aligned}\frac{\partial p_{\ell,i}}{\partial t} &= -\frac{\partial V}{\partial x_{\ell,i}}, \\ \frac{\partial x_{\ell,i}}{\partial t} &= 2p_{\ell,i}.\end{aligned}$$

The leap-frog method goes as the follows:

$$x_{\ell,i}(t + \Delta t) - x_{\ell,i}(t) = \int_t^{t+\Delta t} dt' 2p_{\ell,i}(t') = 2p_{\ell,i}(t + \frac{1}{2}\Delta t)\Delta t + O(\Delta t^3) \quad (1.6.28)$$

$$p_{\ell,i}(t + \frac{3}{2}\Delta t) - p_{\ell,i}(t + \frac{1}{2}\Delta t) = -\int_{t+\frac{1}{2}\Delta t}^{t+\frac{3}{2}\Delta t} dt' \frac{\partial V(t')}{\partial x_{\ell,i}} = -\frac{\partial V(t + \Delta t)}{\partial x_{\ell,i}}\Delta t + O(\Delta t^3). \quad (1.6.29)$$

Note that at the first step, we need to update the moment $p_{\ell,i}$ at half step:

$$p_{\ell,i}(t + \frac{1}{2}\Delta t) = p_{\ell,i}(t) - \frac{\partial V(t)}{\partial x_{\ell,i}} \frac{1}{2}\Delta t + O(\Delta t^2). \quad (1.6.30)$$

The approximation is the leap-frog method if we omit the high order term.

Remark 13 *The leap frog method has another equivalent formula:*

$$\begin{aligned}x(t + \Delta t) &= x(t) + \Delta t p(t) - \frac{\Delta t^2}{2} \frac{\partial V}{\partial x}(t), \\ p(t + \Delta t) &= p(t) - \frac{\Delta t}{2} \left(\frac{\partial V}{\partial x}(t) + \frac{\partial V}{\partial x}(t + \Delta t) \right).\end{aligned}$$

For large time steps Δt , the discrete error of Leap-frog method becomes large and $e^{-H(x,p,\Phi_\sigma)}$ is not constant, then a Metropolis acceptance-rejection step is needed.

In the limit of small Δt a molecular dynamics step keeps H constant. One can view this as certain type of Monte Carlo move in which the energy does not change. The Metropolis algorithm tells us such a move should be accepted with probability $p = 1$. Unfortunately, we have to use finite Δt and so H is not precisely constant. In order to keep the simulation from being biased by this fact, we need to add an acceptance-rejection step with $\exp(-\Delta H)$.

Now we are ready to present a hybrid quantum monte carlo (HQMC) method.

HQMC

1. Generate two vectors of Gaussian random number R_σ , each component of which has a probability distribution proportional to $\exp(-R_{i,\sigma}^2)$, and a heat-bath updating of Φ_σ is made:

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

2. Set each $p_{\ell,i}$ equal to a Gaussian random number with probability distribution proportional to $\exp(-p_{i,l}^2)$.
3. Fix Φ_σ , and regard current (x, p) as initial values, then update (x, p) by leap-frog method:
 - (a) p is evolved through a time $\frac{1}{2}\Delta t$:

$$p_{\ell,i}(t + \frac{1}{2}\Delta t) = p_{\ell,i}(t) - \frac{\partial V(t)}{\partial x_{\ell,i}} \frac{1}{2}\Delta t. \quad (1.6.31)$$

- (b) Next, of order $1/\Delta t$ molecular-dynamics steps are made using Eqs. (1.6.28) and (1.6.29):

$$x_{\ell,i}(t + \Delta t) - x_{\ell,i}(t) = 2p_{\ell,i}(t + \frac{1}{2}\Delta t)\Delta t, \quad (1.6.32)$$

$$p_{\ell,i}(t + \frac{3}{2}\Delta t) - p_{\ell,i}(t + \frac{1}{2}\Delta t) = -\frac{\partial V(t + \Delta t)}{\partial x_{\ell,i}} \Delta t. \quad (1.6.33)$$

4. Acceptance-rejection: Generate one random number $r \sim \text{Uniform}[0, 1]$ and update

$$(x(T), p(T)) = \begin{cases} (x(T), p(T)), & \text{if } r \leq \min \left\{ 1, \frac{e^{-H(x(T), p(T), \Phi_\sigma)}}{e^{-H(x(0), p(0), \Phi_\sigma)}} \right\} \\ (x(0), p(0)), & \text{otherwise.} \end{cases} \quad (1.6.34)$$

5. Repeat step 2 and step 3 N_{MD} times, then Goto step 1.

Remark 14 *The Langevin-Euler update is equivalent to a single-step hybrid Monte Carlo move[1]. The molecular dynamics and Langevin 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. We are using these because we do not want to change the fields one at a time with monte carlo.*

The computation of $\frac{\partial V}{\partial x}$: For simplicity, here we denote $K = e^{t\tau K}$ and $V_\ell = e^{-\sigma\tau V_\ell}$. In order to obtain $\frac{\partial V}{\partial x_{i,l}}$, $\frac{\partial \Phi^T O^{-1} \Phi}{\partial x_{\ell,i}}$ should be computed. Let $X = O^{-1} \Phi$,

$$\frac{\partial \Phi^T O^{-1} \Phi}{\partial x_{\ell,i}} = -\Phi^T O^{-1} \frac{\partial O}{\partial x_{\ell,i}} O^{-1} \Phi = -X^T \frac{\partial O}{\partial x_{\ell,i}} X.$$

Since $O = M^T M$, it follows

$$X^T \frac{\partial O}{\partial x_{\ell,i}} X = 2(MX)^T \frac{\partial M}{\partial x_{\ell,i}} X.$$

Note that M can be written as

$$M = I - K_{[L]} \text{diag}(V_\ell) P,$$

where

$$K_{[L]} = \text{diag}(K, K, \dots, K), \quad P = \begin{bmatrix} 0 & & & -I \\ I & 0 & & \\ & \ddots & \ddots & \\ & & I & 0 \end{bmatrix}.$$

Since only $v_{\ell,i}$ depends on $x_{\ell,i}$, we have

$$\frac{\partial \text{diag}(V_\ell)}{\partial x_{\ell,i}} = \text{diag}(0, \dots, 0, \frac{\partial v_{\ell,i}}{\partial x_{\ell,i}}, 0, \dots, 0).$$

Therefore

$$-X^T \frac{\partial O}{\partial x_{\ell,i}} X = 2 \frac{\partial v_{\ell,i}}{\partial x_{\ell,i}} (K_{[L]}^T M X)_{i,\ell} (P X)_{\ell,i}.$$

Note that $V(x) = S_B(x) + \sum_\sigma \Phi_\sigma O_\sigma^{-1} \Phi_\sigma$ and $v_{\ell,i,\sigma} = \exp(-\sigma\tau(2U)^{\frac{1}{2}}x_{\ell,i})$, then

$$\begin{aligned} \frac{\partial V}{\partial x_{\ell,i}} &= 2\tau x_{\ell,i} - 2(2U)^{\frac{1}{2}}\tau v_{\ell,i,+} (K_{[L]}^T M_+ O_+^{-1} \Phi_+)_{\ell,i} (P O_+^{-1} \Phi_+)_{\ell+1,i} \\ &\quad + 2(2U)^{\frac{1}{2}}\tau v_{\ell,i,-} (K_{[L]}^T M_- O_-^{-1} \Phi_-)_{i,\ell} (P O_-^{-1} \Phi_-)_{\ell+1,i}. \end{aligned}$$

1.7 Physical measurements

In this section, we formulate interesting physical measurement quantities which can be obtained from the equal-time single-particle Green's function $G_{ij} = \langle c_i c_j^\dagger \rangle$ and unequal-time single-particle Green's function $G(\ell_1, i; \ell_2, j) = \langle c_i(\ell_1) c_j^\dagger(\ell_2) \rangle$.

Hirsch proved that the equal time Green's function

$$\langle c_i c_j^\dagger \rangle = (I + B_L B_{L-1} \cdots B_1)_{ij}^{-1},$$

and the non-equal time Green's function

$$\langle c_i(\ell_1) c_j^\dagger(\ell_2) \rangle = (B_{\ell_1} B_{\ell_1-1} \cdots B_{\ell_2+1} (I + B_{\ell_2} \cdots B_1 B_L \cdots B_{\ell_2+1})^{-1})_{ij}.$$

At every measurement steps in the HQMC simulation, the equal-time Green's function G_{ij} can be obtained from the diagonal block of M^{-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^{-1} .

Remark 15 *The inverse of M is known explicitly, and of the form*

$$M^{-1} = W^{-1} Z,$$

where

$$W = \begin{bmatrix} I + B_1 B_L \cdots B_2 & & & \\ & I + B_2 B_1 B_L \cdots B_3 & & \\ & & \ddots & \\ & & & I + B_L B_{L-1} \cdots B_1 \end{bmatrix}.$$

and

$$Z = \begin{bmatrix} I & -B_1 B_L \cdots B_3 & -B_1 B_L \cdots B_4 & \cdots & -B_1 B_L & -B_1 \\ B_2 & I & -B_2 B_1 B_L \cdots B_4 & \cdots & -B_2 B_1 B_L & -B_2 B_1 \\ B_3 B_2 & B_3 & I & \cdots & -B_3 B_2 B_1 B_L & -B_3 B_2 B_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ B_{L-1} \cdots B_2 & B_{L-1} \cdots B_3 & B_{L-1} \cdots B_4 & \cdots & I & -B_{L-1} \cdots B_2 B_1 \\ B_L \cdots B_2 & B_L \cdots B_3 & B_L \cdots B_4 & \cdots & B_L & I \end{bmatrix},$$

In other words, the (ℓ_1, ℓ_2) block submatrix of M^{-1} is:

$$(M^{-1})_{\ell_1, \ell_2} = (I + B_{\ell_1} \cdots B_1 B_L \cdots B_{\ell_1+1})^{-1} Z_{\ell_1 \ell_2},$$

where

$$Z_{\ell_1 \ell_2} = \begin{cases} B_{\ell_1} B_{\ell_1-1} \cdots B_{\ell_2+1}, & \ell_1 > \ell_2 \\ I, & \ell_1 = \ell_2 \\ -B_{\ell_1} \cdots B_1 B_L \cdots B_{\ell_2+1}, & \ell_1 < \ell_2 \end{cases}.$$

where $B_{\ell_2} = B_1$ if $\ell_2 > L$.

1.7.1 Equal-time Green's function \bar{G}

The Green's function is translationally invariant. It only depends on the $k = i - j$ and not i and j separately (see (1.2.11)). By averaging over all the equivalent values, the equal-time Green's function \bar{G} will give us a better estimate (small error bars).

The equal-time Green's function \bar{G} is defined by: for $0 \leq k_x \leq \frac{N_x}{2}$ and $0 \leq k_y \leq \frac{N_y}{2}$,

$$\bar{G}(kx, ky) = \frac{1}{2N_x N_y x_{fac} y_{fac}} \sum_{|i_x - j_x| = k_x} \sum_{|i_y - j_y| = k_y} (G_{\uparrow}(i, j) + G_{\downarrow}(i, j)),$$

where the factors x_{fac} and y_{fac} are defined as

$$x_{fac} = \begin{cases} 1 & \text{if } k_x = 0, \frac{N_x}{2} \\ 2 & \text{otherwise.} \end{cases} \quad y_{fac} = \begin{cases} 1 & \text{if } k_y = 0, \frac{N_y}{2} \\ 2 & \text{otherwise.} \end{cases}$$

1.7.2 Two-particle Green's function

By using Wick's theorem, if $\alpha \neq \beta \neq \delta \neq \gamma$, then

$$\langle c_{\gamma}^{\dagger} c_{\delta}^{\dagger} c_{\beta} c_{\alpha} \rangle = \langle c_{\gamma}^{\dagger} c_{\alpha} \rangle \langle c_{\delta}^{\dagger} c_{\beta} \rangle - \langle c_{\gamma}^{\dagger} c_{\beta} \rangle \langle c_{\delta}^{\dagger} c_{\alpha} \rangle.$$

1.7.3 Equal-time density-density correlations

The "up-up" density-density correlation is defined as

$$(n_{i\uparrow} n_{j\uparrow} + n_{i\downarrow} n_{j\downarrow})/2.$$

If $i = j$

$$(n_{i\uparrow} n_{j\uparrow} + n_{i\downarrow} n_{j\downarrow})/2 = (n_{i\uparrow} + n_{i\downarrow})/2,$$

and if $i \neq j$,

$$\langle n_i n_j \rangle = \langle n_i \rangle \langle n_j \rangle - \langle c_j c_i^\dagger \rangle \langle c_i c_j^\dagger \rangle,$$

therefore the correlation is

$$\begin{aligned} (n_{i\uparrow} n_{j\uparrow} + n_{i\downarrow} n_{j\downarrow})/2 &= ((1 - G_\uparrow(i, i))(1 - G_\uparrow(j, j)) + (1 - G_\downarrow(i, i))(1 - G_\downarrow(j, j))) \\ &\quad - G_\uparrow(j, i)G_\uparrow(i, j) - G_\downarrow(j, i)G_\downarrow(i, j))/2 \end{aligned}$$

The ‘up-down’ density correlation is defined as

$$n_{i\uparrow} n_{j\downarrow},$$

which can be computed be

$$\langle n_{i\uparrow} n_{j\downarrow} \rangle = (1 - G_\uparrow(i, i))(1 - G_\downarrow(j, j)).$$

1.7.4 Equal-time spin-spin correlations

The spin-spin correlations measured in the z and in the x direction are

$$\begin{aligned} C_{ij}^z &= (c_i^\dagger \sigma_z c_i)(c_j^\dagger \sigma_z c_j) = \langle (n_{i\uparrow} - n_{i\downarrow})(n_{j\uparrow} - n_{j\downarrow}) \rangle, \\ C_{ij}^x &= (c_i^\dagger \sigma_x c_i)(c_j^\dagger \sigma_x c_j) = \langle (c_{i\uparrow}^\dagger c_{i\downarrow} + c_{i\downarrow}^\dagger c_{i\uparrow})(c_{j\uparrow}^\dagger c_{j\downarrow} + c_{j\downarrow}^\dagger c_{j\uparrow}) \rangle, \end{aligned}$$

where $c_i = (c_{i\uparrow} \quad c_{i\downarrow})$, and

$$\sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

and

$$\langle (n_{i\uparrow} - n_{i\downarrow})(n_{j\uparrow} - n_{j\downarrow}) \rangle = \langle n_{i\uparrow} n_{j\uparrow} \rangle + \langle n_{i\downarrow} n_{j\downarrow} \rangle - \langle n_{i\uparrow} \rangle \langle n_{j\downarrow} \rangle - \langle n_{i\downarrow} \rangle \langle n_{j\uparrow} \rangle.$$

If $i = j$, since

$$\langle n_{i\uparrow} n_{j\uparrow} \rangle = \langle n_{i\uparrow}^2 \rangle = \langle n_{i\uparrow} \rangle,$$

then

$$\langle (n_{i\uparrow} - n_{i\downarrow})^2 \rangle = \langle n_{i\uparrow} \rangle + \langle n_{i\downarrow} \rangle - 2 \langle n_{i\downarrow} \rangle \langle n_{i\uparrow} \rangle.$$

If $i \neq j$, then

$$\begin{aligned} \langle (n_{i\uparrow} - n_{i\downarrow})(n_{j\uparrow} - n_{j\downarrow}) \rangle &= (\langle n_{i\uparrow} \rangle - \langle n_{i\downarrow} \rangle)(\langle n_{j\uparrow} \rangle - \langle n_{j\downarrow} \rangle) \\ &\quad - \langle c_{j\uparrow} c_{i\uparrow}^\dagger \rangle \langle c_{i\uparrow} c_{j\uparrow}^\dagger \rangle - \langle c_{j\downarrow} c_{i\downarrow}^\dagger \rangle \langle c_{i\downarrow} c_{j\downarrow}^\dagger \rangle. \end{aligned}$$

Therefore the spin-spin correlations in the z direction are computed as

$$C_{ij}^z = \begin{cases} 2 - G_\uparrow(i, i) - G_\downarrow(i, i) - 2(1 - G_\downarrow(i, i))(1 - G_\uparrow(i, i)), & i = j. \\ (G_\uparrow(i, i) - G_\downarrow(i, i))(G_\uparrow(j, j) - G_\downarrow(j, j)) - G_\uparrow(j, i)G_\uparrow(i, j) - G_\downarrow(j, i)G_\downarrow(i, j), & i \neq j \end{cases}$$

For the spin-spin correlations in the x direction,

$$C_{ij}^x = \langle c_{i\uparrow}^\dagger c_{i\downarrow} c_{j\uparrow}^\dagger c_{j\downarrow} + c_{i\uparrow}^\dagger c_{i\downarrow} c_{j\downarrow}^\dagger c_{j\uparrow} + c_{i\downarrow}^\dagger c_{i\uparrow} c_{j\uparrow}^\dagger c_{j\downarrow} + c_{i\downarrow}^\dagger c_{i\uparrow} c_{j\downarrow}^\dagger c_{j\uparrow} \rangle.$$

We will prove that the first and fourth terms on the right hand are zeros, then if $i \neq j$,

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

If $i = j$,

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

1.7.5 Staggered Susceptibility

$$\begin{aligned}\chi &= \frac{1}{V_s} \int_0^\beta d\tau \sum_{i,j} (-1)^{i+j} \langle [c_i^\dagger(\tau) \sigma_z c_i(\tau)] \times [c_j^\dagger(0) \sigma_z c_j(0)] \rangle \\ &= \frac{1}{V_s} \int_0^\beta d\tau \sum_{i,j} (-1)^{i+j} \langle [n_{i\uparrow}(\tau) - n_{i\downarrow}(\tau)] \times [n_{j\uparrow}(0) - n_{j\downarrow}(0)] \rangle.\end{aligned}$$

Here we use the usual z component of the magnetization. Because of rotational invariance, we can use the x or y components. It turns out that the transverse (x or y) components provide a significantly less noisy estimator for χ :

$$\chi = \frac{1}{V_s} \int_0^\beta d\tau \sum_{i,j} (-1)^{i+j} \langle [c_i^\dagger(\tau) \sigma_x c_i(\tau)] [c_j^\dagger(0) \sigma_x c_j(0)] \rangle. \quad (1.7.35)$$

We will define

$$\chi(q, \omega) = \sum_l e^{iq \cdot l} \sum_\tau e^{i\omega\tau} \chi(l, \tau). \quad (1.7.36)$$

1.8 Pseudo-code of HQMC

The following is a pseudo-code of a full HMQC we have implemented.

HQMC

1. Initial step:
 - Generate Gaussian random numbers $R_{i,l,\sigma}$, and compute $\Phi_\sigma = M_\sigma^T R_\sigma$;
 - Setup initial value of $x_{i,l}^{(0)}$.
2. MC step, this step will run MAXKMC times
 - (a) MD step (`MDx.f`)
 - i. Generate Gaussian random numbers $p_{i,l}^0$;
 - ii. Initial step: $p_{i,l}^{(\frac{1}{2})} = p_{i,l}^{(0)} - \frac{1}{2} \frac{\partial V(x^{(0)})}{\partial x_{i,l}^{(0)}} \Delta t$
 - iii. MD will run $\frac{C}{\Delta t}$ steps:
 - A. $x_{i,l}^{(1)} - x_{i,l}^{(0)} = 2p_{i,l}^{(\frac{1}{2})} \Delta t$,
 - B. $p_{i,l}^{(\frac{3}{2})} - p_{i,l}^{(\frac{1}{2})} = -\frac{\partial V(x^{(1)})}{\partial x_{i,l}^{(1)}} \Delta t$
 - (b) Move $x^{(0)}$ to a new configuration $x^{(1)}$, compute $e^{-H(x,p,\Phi_\sigma)}$ (`hamiltonian.f`)

$$H(x, p, \Phi_\sigma) = p^T p + S_B(x) + \Phi_+ O_+^{-1} \Phi_+ + \Phi_- O_-^{-1} \Phi_-.$$

Using Metropolis algorithm to decide whether we accept the configuration $x^{(1)}$: If $x^{(1)}$ is accepted, then $x^{(0)} = x^{(1)}$.

- (c) If measurement is needed, enter measure step.
 - (d) Goto step 2.
3. Measure step:
- (a) Get X_σ from $M_\sigma X_\sigma = R_\sigma$
 - (b) Obtain the unbiased estimate $\hat{M}_{i,j,\sigma}^{-1} = 2X_{i,\sigma}R_{j\sigma}$
 - (c) Get the statistical error bar.
 - (d) Update $R_{i,l}$
 - (e) $\Phi_\sigma = M_\sigma^T R_\sigma$.
-

Remark 16 *The force term $\frac{\partial V(x)}{\partial x}$ is computed by the subroutine `PVPXdownx.f` as described in section 6.*

Bibliography

- [1] R. Blankenbecler, D. J. Scalapino and R. L. Sugar, Phys. Rev. D, 24(1981), 2278-
- [2] J. E. Hirsch, Two-dimensional Hubbard model: Numerical simulation study, Physical Review B, v31,N7(1985), 4403-4419.
- [3] J. E. Hirsch, Hubbard-stratonovich transformation for fermion lattice models, PRB, 28(7)(1983),4059-4061.
- [4] J. E. Hirsch, Erratum: Discrete Hubbard-Stratonovich transformation for fermion lattice models, PRB,29(7)(1984),4159.
- [5] Jun S. Liu, Monte Carlo strategies in scientific computing, Spring series in statistics, 2001.
- [6] R. Schumann and E. Heiner, Transformations of the Hubbard interaction to quadratic forms, Physics Letters A, 134(3)(1988),202-204.
- [7] D. J. Scalapino and R. L. Sugar, Phys.Rev. B, 24(1981),4295-
- [8] R. T. Scalettar, D. J. Scalapino, R. L. Sugar, and D. Toussaint, Hybrid molecular-dynamics algorithm for the numerical simulation of many-electron systems, Physical Review B, V36, N16(1987),8632-8640.
- [9] R. T. Scalettar, D. J. Scalapino and R. L. Sugar, New algorithm for the numerical simulation of fermions, Physical Review B, 34(1986),7911-7917.
- [10] J. P. Wallington and J. F. Annett, Discrete symmetries and transformations of the Hubbard model, PRB, 58(3),1998,1218-1221.

Lecture 2

Hubbard matrix analysis

Before achieving our goals of developing robust and efficient algorithmic techniques and high performance software for QMC simulation, we need to understand the mathematical and numerical properties of the underlying matrices, such as eigenvalue distribution and condition number. In this lecture, we study the dynamics and transitional behavior of these properties as functions of the multiscale parameters.

2.1 Hubbard matrices

The Hubbard matrices M introduced in Lecture 1 are block L -cyclic matrices of the form

$$M = \begin{bmatrix} I & & & & B_1 \\ -B_2 & I & & & \\ & -B_3 & I & & \\ & & \ddots & \ddots & \\ & & & -B_L & I \end{bmatrix}, \quad (2.1.1)$$

where

- I is an $N \times N$ unit matrix, N is the spatial size of system, $N = N_x \times N_y$. N_x is the dimension of x -spatial lattice, and N_y is the dimension of y -spatial lattice.
- For $\ell = 1, 2, \dots, L$, matrices B_ℓ are $N \times N$ and are of the form

$$B_\ell = e^{t\tau K} e^{V_\ell}. \quad (2.1.2)$$

- t is hopping parameters, and $\tau = \frac{\beta}{L}$.
- The matrix K is defined by

$$K = I_y \otimes K_x + K_y \otimes I_x, \quad (2.1.3)$$

where I_x and I_y are identity matrices with dimension N_x and N_y and

$$K_x, K_y = \begin{bmatrix} 0 & 1 & & & 1 \\ 1 & 0 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & & 1 & 0 & 1 \\ 1 & & & & 1 & 0 \end{bmatrix}. \quad (2.1.4)$$

- For $\ell = 1, 2, \dots, L$, matrices V_ℓ are diagonal

$$V_\ell = \mu \cdot \text{diag}(h_{\ell,1}, h_{\ell,2}, \dots, h_{\ell,N}), \quad (2.1.5)$$

where ν is a parameters defined by $\cosh \nu = e^{\frac{U\tau}{2}}$, and $h_{\ell,j}$ are random variables. In the DQMC, $h_{\ell,i} = 1$ or -1 with equal probability (two-point distribution). In the HQMC, $h_{\ell,i}$ are given from a molecular-dynamics (MD). Note that by the Taylor expansion, $\nu = \sqrt{\tau U} + \frac{(\tau U)^{\frac{3}{2}}}{12} + \mathcal{O}((\tau U)^2)$.

The matrix M is referred as a *multiscale* matrix since the dimension and properties of M are characterized by multiple length and energy parameters

- Length parameters: N and L
 - $N = N_x \times N_y$ is the 2D spatial size. It measures the number of electrons being simulated.
 - L is the number of blocks, it is set by the inverse of the temperature.
- 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
- The parameter connecting length and energy scales: τ
 - $\tau = \beta/L$,
 - τ is an imaginary-time parameter, a measure of the accuracy of the Trotter decomposition.

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.

The following parameter ranges are of practical interests:

- integer $N = N_x \times N_y = 4 \times 4, 8 \times 8, 16 \times 16, 32 \times 32$
- integer $L = 8, 16, \dots, 80, \dots, 160, \dots, 320$,
- real number $t = 1$
- real number $U \in [0, 6]$
- real number $\beta \in (0, 20]$
- real number $\tau = 1/8, 1/16, 1/32$.

In summary, the key features of these matrices are

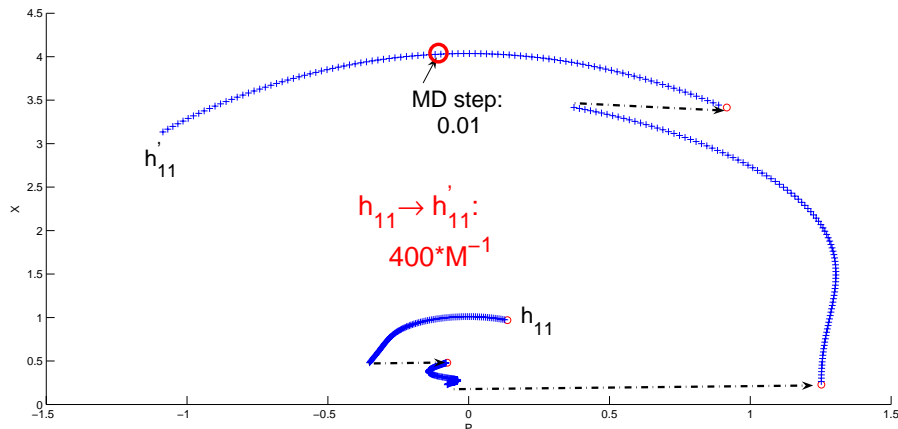


Figure 2.1: A typical MD trajectory.

- M incorporates multiple structural scales: The inverse temperature β determines the number of blocks $L = \beta/\tau$, where τ is a discretization stepsize. Typically $L = 10$ to 10^2 . The dimension of the individual blocks is set by N the number of spatial sites. In current 2D simulations $N = N_x \times N_y \propto 10^2$. Thus the total dimension of the M currently being studied is 10^4 . Our goal is to extend this by an order of magnitude to 10^5 .
- M incorporates multiple energy scales: The parameter t which enters K determines the kinetic energy of the electrons, and the interaction energy scale U enters V_ℓ .
- M is a function of a collection of NL variables, the Hubbard-Stratonovich field $h_{\ell i}$. The role of the simulation is to determine the configurations of these variables which make large contributions to operator expectation values, and then to sum over those configurations. Therefore, the associated matrix computation problems, such as $\det(M)$ and $(M^T M)^{-1}b$, need to be solved 10^4 to 10^5 times in a full simulation, see Figure 2.1

The numerical linear algebra problems which enter quantum simulations are the following:

1. Solution of $M^T M x = b$ is needed in a molecular dynamics step.
2. Computation of specific elements of the inverse $(M^{-1})_{ij}$. These determine all the physical observables: energy, density, magnetic moments, etc.
3. Computation of $\det(M)$, the probability of the configuration $h_{\ell i}$.
4. Computation of $\frac{\det(\widehat{M})}{\det(M)}$, where \widehat{M} is a low-rank update of M is needed in the accept/reject decision of a monte carlo move involving a change to a small number of $h_{\ell i}$.

The computational challenge is to increase the spatial dimension $N = N_x \times N_y$ from $O(10^2)$ to $O(10^3)$, that is, to do a 1000 electron QMC simulation. Such an increase would have a tremendous impact on our understanding of strongly interacting materials because it

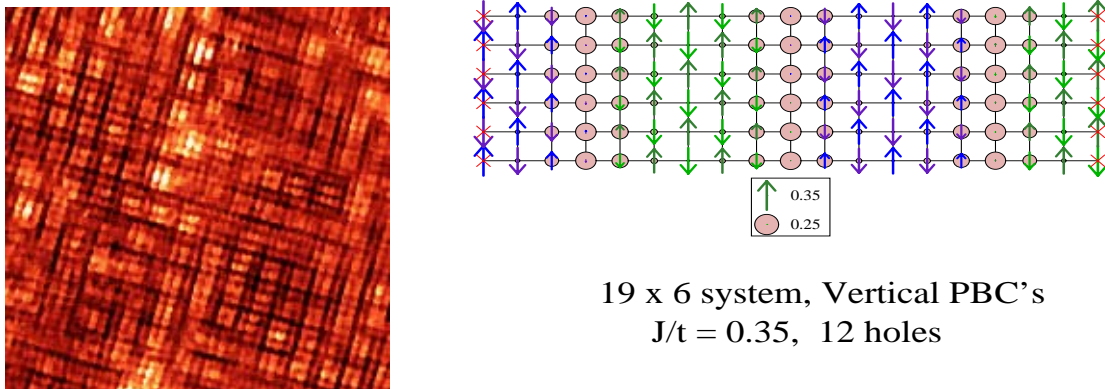


Figure 2.2: Left: Conductance map of a 'checkerboard' electronic crystal state in lightly hole-doped $\text{Ca}_{2-x}\text{Na}_x\text{CuO}_22\text{Cl}_2$ obtained using scanning tunneling microscopy. Ref: T. Hanaguri *etal.* Nature 430, 1001 (2004). Right: The complex stripe structure arising from removing electrons from the filling of one electron per site in the Hubbard model. The arrows and their sizes indicate the magnetic moments present on the sites, while the circles indicate the density of holes. The holes form (vertical) stripes separated by regions with strong antiferromagnetic correlations. Ref: S.R. White *etal.*, Phys. Rev. Lett. 80, 1272(1998).

would allow for the first time the simulation of systems incorporating a reasonable number of the mesoscopic structures, such as the checkerboards and stripes illustrated in Figure 2.2.

2.2 Basic Properties

In this section, we exploit basic properties of the Hubbard matrix M .

1. The matrix M can be compactly written as

$$M = I - \text{diag}(B_1, B_2, \dots, B_L)P,$$

where P is a block matrix of the form

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

2. We have the identity

$$\det(M) = \det(I + B_L B_{L-1} \cdots B_1). \quad (2.2.6)$$

PROOF. If a matrix A is a 2×2 block matrix,

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

then

$$\det(A) = \det(A_{11}) \det(A_{22} - A_{21}A_{11}^{-1}A_{12}). \quad (2.2.7)$$

Therefore if M is regarded as a 2×2 block matrix A , and let A_{11} is the identity block I at the first row of blocks and the first column of blocks, then from (2.2.7), we have

$$\det(M) = \det \begin{bmatrix} I & & & & & B_2 B_1 \\ -B_3 & I & & & & \\ & -B_4 & I & & & \\ & & & \ddots & & \\ & & & & -B_L & I \end{bmatrix}.$$

Repeatedly using (2.2.7), we obtain the equation (2.2.6). \square

3. The inverse of M is known explicitly,

$$M^{-1} = W^{-1}Z, \quad (2.2.8)$$

where

$$W = \text{diag}(I + B_1 B_L \cdots B_2, I + B_2 B_1 B_L \cdots B_3, \dots, I + B_L B_{L-1} \cdots B_1)$$

and

$$Z = \begin{bmatrix} I & -B_1 B_L \cdots B_3 & -B_1 B_L \cdots B_4 & \cdots & -B_1 B_L & -B_1 \\ B_2 & I & -B_2 B_1 B_L \cdots B_4 & \cdots & -B_2 B_1 B_L & -B_2 B_1 \\ B_3 B_2 & B_3 & I & \cdots & -B_3 B_2 B_1 B_L & -B_3 B_2 B_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ B_{L-1} \cdots B_2 & B_{L-1} \cdots B_3 & B_{L-1} \cdots B_4 & \cdots & I & -B_{L-1} \cdots B_2 B_1 \\ B_L \cdots B_2 & B_L \cdots B_3 & B_L \cdots B_4 & \cdots & B_L & I \end{bmatrix}.$$

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

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

where

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

Note that $B_j = B_1$ if $j > L$.

PROOF: By direct verification that $MM^{-1} = I$.

4. The eigenvalues of K can be written as

$$\lambda(K) = 2(\cos \theta_x + \cos \theta_y) \equiv \epsilon_{x,y}, \quad (2.2.9)$$

where

$$\begin{aligned} \theta_x &= \frac{2k_x \pi}{N_x}, & \text{for } k_x = 1, 2, \dots, N_x \\ \theta_y &= \frac{2k_y \pi}{N_y}, & \text{for } k_y = 1, 2, \dots, N_y. \end{aligned}$$

and the corresponding eigenvector of K is $v_y \otimes v_x$, where

$$\begin{aligned} v_x &= \frac{1}{\sqrt{N_x}} [1, e^{i\theta_x}, e^{i2\theta_x}, \dots, e^{i(N_x-1)\theta_x}]^T, \\ v_y &= \frac{1}{\sqrt{N_y}} [1, e^{i\theta_y}, e^{i2\theta_y}, \dots, e^{i(N_y-1)\theta_y}]^T. \end{aligned}$$

PROOF: By direct verification.

5. The computation of the exponential matrix $B = e^{t\tau K}$

By the definition of K , the matrix exponential $e^{t\tau K}$ can be written as the product of two exponential matrices:

$$B = e^{t\tau K} = (I_y \otimes e^{t\tau K_x})(e^{t\tau K_y} \otimes I_x) = e^{t\tau K_y} \otimes e^{t\tau K_x}. \quad (2.2.10)$$

By the eigendecomposition of K_x and K_y , we can use the FFT to compute B . The computational complexity of formulating the matrix B explicitly is $O(N^2)$. The complexity of matrix-vector multiplication is $O(N(\log N_x + \log N_y))$.

Alternatively, we use a so-called *checkerboard approximation* to construct the matrix B . For simplicity, assume that N_x is even, then K_x can be decomposed as

$$K_x = K_x^{(1)} + K_x^{(2)}, \quad (2.2.11)$$

where

$$K_x^{(1)} = \begin{bmatrix} D & & & \\ & D & & \\ & & \ddots & \\ & & & D \end{bmatrix}, \quad K_x^{(2)} = \begin{bmatrix} 0 & & & 1 \\ & D & & \\ & & \ddots & \\ 1 & & & D \\ & & & & 0 \end{bmatrix}. \quad (2.2.12)$$

where D is the 2×2 matrix,

$$D = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Then for any constant $\alpha \neq 0$, $e^{\alpha D}$ is given by

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

Therefore, we have

$$e^{\alpha K_x^{(1)}} = \begin{bmatrix} e^{\alpha D} & & & \\ & e^{\alpha D} & & \\ & & \ddots & \\ & & & e^{\alpha D} \end{bmatrix},$$

and

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

Note that $K_x^{(1)}$ and $K_x^{(2)}$ are not commutative, therefore, we can use the approximation

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

The exponential matrix $e^{\alpha K_y}$ can be approximated in the same way.

As a result, the matrix B has the approximation

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

In practice, the matrix $B = e^{t\tau K}$ is expressed by

$$B = (e^{t\tau K_y^{(1)}} e^{t\tau K_y^{(2)}}) \otimes (e^{t\tau K_x^{(1)}} e^{t\tau K_x^{(2)}}). \quad (2.2.13)$$

There are 16 nonzero elements in every row and every column of the matrix B .

For any vector x with dimension $N_x N_y$ and matrix X with $x = \text{vec}(X)$,

$$Bx = \text{vec}(e^{t\tau K_x^{(2)}} e^{t\tau K_x^{(1)}} X e^{t\tau K_y^{(1)}} e^{t\tau K_y^{(2)}}). \quad (2.2.14)$$

If $\cosh \alpha$ and $\sinh \alpha$ are computed in advance, the cost to (approximately) construct the whole matrix B is $16N$. The cost of matrix-vector multiplication Bx is $12N$,

The computational cost can be further reduced by rewriting the block $e^{\alpha D}$ as

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

By this trick, the computational cost of the matrix-vector multiplication Bx is $9N$.

A few remarks in order:

- The checkerboard method is an approximation method, which can only be used when τ is small enough.
- The approximation of B is not symmetric. It is easy to have a symmetric approximation of B :

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

In this symmetric version, there are 36 nonzero elements in every row and every column.

The cost to construct a symmetric approximation of the matrix B is $36N$.

- By the approximation (2.2.13) of B , the matrix M is sparse with 17 nonzero element in every row and every column.
- The checkerboard method is particularly useful for different hopping t_{ij} at different lattices, i.e., t is not a constant.

2.3 Eigen-distribution

The study of eigenvalues of a cyclic matrix can be tracked back to the work of Frobenius (1912), Romanovsky (1936) and Varga (1962) [7]. The following theorem characterizes the eigenvalue distribution of the block L -cyclic matrix M .

Theorem. 17 *The eigenvalues of M are given by*

$$\lambda(M) = 1 - \lambda(B_L \cdots B_2 B_1)^{\frac{1}{L}} e^{i \frac{(2\ell+1)\pi}{L}}, \quad 0 \leq \ell \leq L-1, \quad (2.3.15)$$

where $\lambda(B_L \cdots B_2 B_1)$ is an eigenvalue of the matrix $B_L \cdots B_2 B_1$.

PROOF: Note that $(I - M)^L$ is a block diagonal matrix

$$(I - M)^L = \text{diag}(-B_1 B_L \cdots B_2, -B_2 B_1 B_L \cdots B_3, \dots, -B_L B_{L-1} \cdots B_1)$$

Note that the diagonal blocks are the cyclically permuted products of the matrices B_1, B_2, \dots, B_L , and therefore, they have the same eigenvalues:

$$\lambda((I - M)^L) = -\lambda(B_L B_{L-1} \cdots B_1).$$

Therefore we have

$$1 - \lambda(M) = \lambda(I - M) = \lambda(B_L B_{L-1} \cdots B_1)^{\frac{1}{L}} e^{i \frac{(2\ell+1)\pi}{L}}, \quad 0 \leq \ell \leq L-1.$$

The proof of the theorem is complete. ■

When $U = 0$, $B_1 = B_2 = \cdots = B_L = B = e^{t\tau K}$. Moreover, the eigenvalues of K are known, see (2.2.9). Immediately, we have the following result.

Theorem. 18 *When $U = 0$, eigenvalues of M are*

$$\lambda(M) = 1 - e^{t\tau \epsilon_{x,y}} e^{i \frac{(2\ell+1)\pi}{L}} i \quad \text{for } 0 \leq \ell \leq L-1. \quad (2.3.16)$$

Furthermore,

$$\max |1 - \lambda(M)| = e^{4t\tau} \quad \text{and} \quad \min |1 - \lambda(M)| = e^{-4t\tau}. \quad (2.3.17)$$

where $\epsilon_{x,y}$ are the eigenvalues of K as defined in (2.2.9).

Figure 2.3 shows the eigenvalue distributions of the matrices M for $U = 0$ (left) and $U = 6$ (right), where $N = 4 \times 4$, $L = 8$, $\beta = 1$, $t = 1$.

When $U = 0$, we can use Theorem 18 to interpret. The eigendistribution has a ring structure, centered at $(1, 0)$. On every ring there are $L = 8$ circles. For $N_x = N_y = 4$, $\epsilon_{x,y} = 2(\cos \theta_x + \cos \theta_y)$ only have 5 different values. There are 40 circles, but the dimension of the matrix M is 128, each circle stands for multiple eigenvalues.

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

1. Lattice size N : Figure 2.4 (Left) shows that when N increases, there are more (blue) points on the ray crossing the ring. The reason is there are more combination in $\epsilon_{x,y}$ for $N = 16 \times 16$ than for $N = 4 \times 4$.

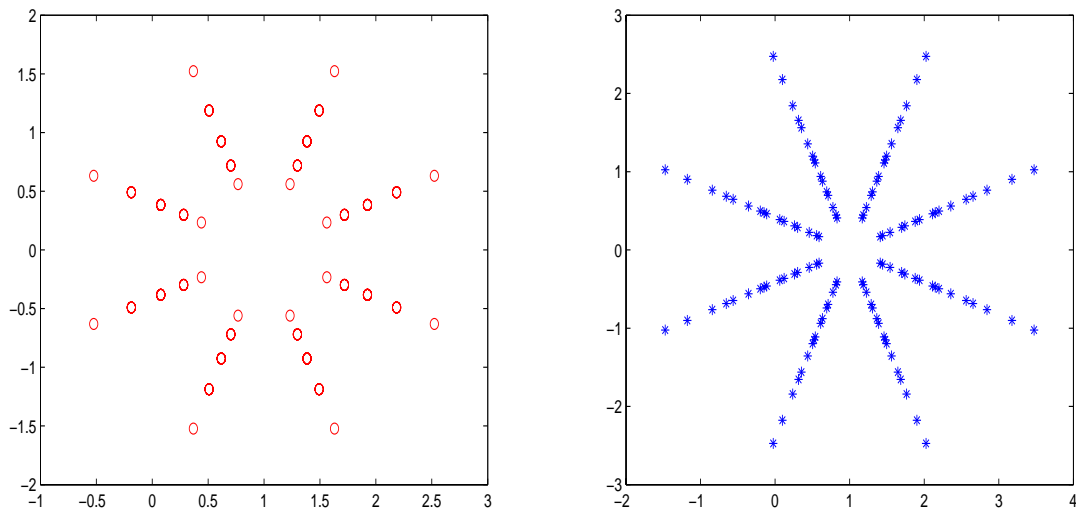


Figure 2.3: The eigenvalues distributions of M for $U = 0$ (left) and $U = 6$ (right), with $N = 4 \times 4$, $L = 8$, $\beta = 1$ and $t = 1$.

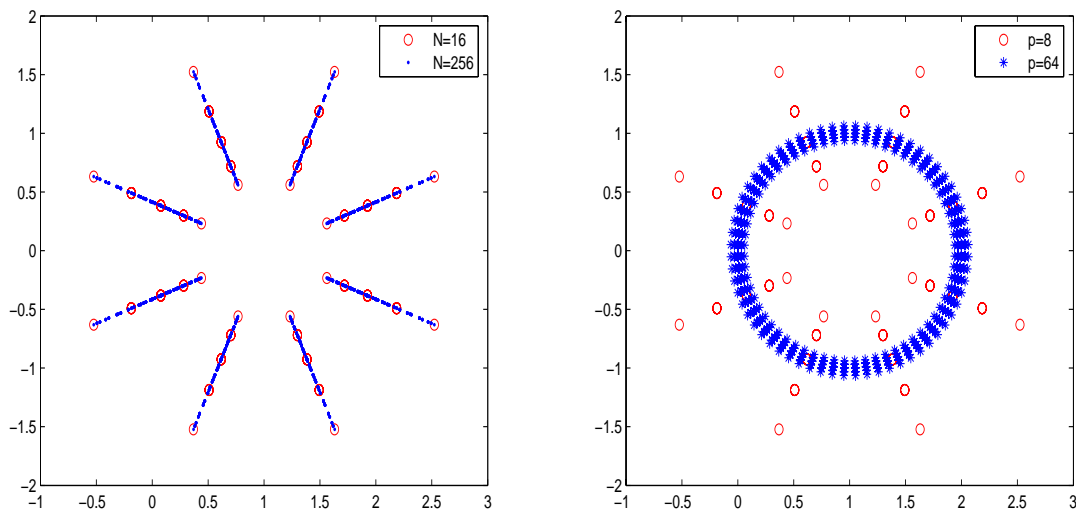


Figure 2.4: Left: The eigenvalue distribution of M for different lattice sizes N : $N = 4 \times 4$ (circle) and $N = 16 \times 16$ (dot), Other parameters are set as $U = 0$, $L = 8$ and $t = 1$. Right: The eigenvalue distribution of M for different number of blocks L : $L = 8$ (circle) and $L = 64$ (asterisk), where $N = 4 \times 4$, $U = 0$ and $t = 1$.

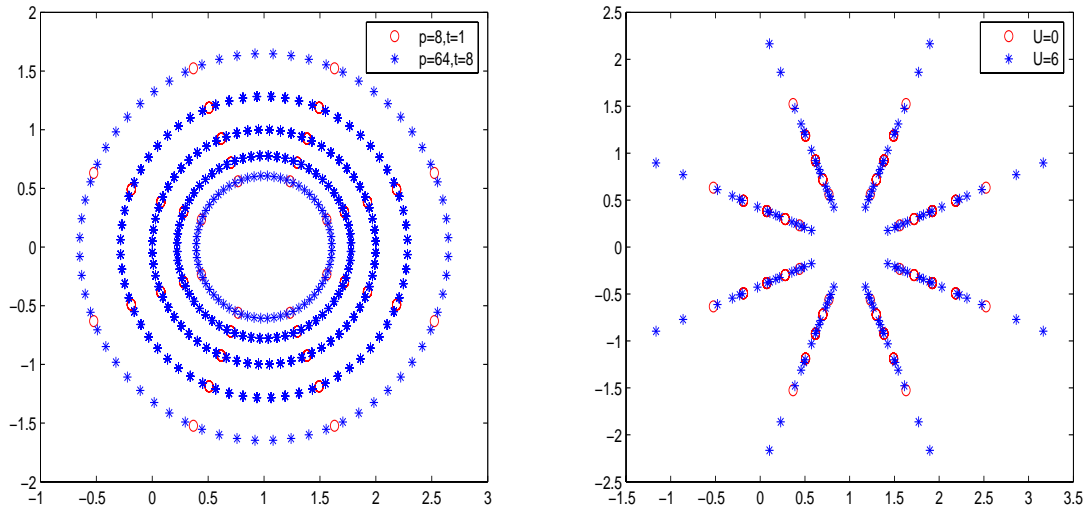


Figure 2.5: Left: The eigenvalues of M for different number of blocks L and t : $L = 8, t = 1$ (circle) and $L = 64, t = 8$ (asterisk), where $N = 4 \times 4$ and $U = 0$. Right: The eigenvalue distribution of M for different energy scales U : $U = 0$ (circle) and $U = 6$ (asterisk). $N = 4 \times 4, L = 8, t = 1$.

2. Block number L : Figure 2.4 (Right) shows for different block number L . Observe (1) number of rings increase, (2) the points on the same rings are equal to L , (3) rings shrink inward.
3. Block number L and t : Figure 2.5 (Left) shows for different block number L and t . The rings do not shrink. The points on the same rings are equal to L .

Theorem 18 can be used to explain the spectral distribution. The points on the one ring are equal to L , so the points on one ring will increase with the block size L . At the same time, since $\tau = \frac{1}{L}$, the range of $|1 - \lambda(M)|$ is $[e^{-\frac{4t}{L}} e^{\frac{4t}{L}}]$, the range will shrink when p increases. But if we fix $\frac{t}{L}$, the range will still keep same, this is the case of left of Figure 2.5 for $L = 8, t = 1$ and $L = 64, t = 8$.

4. Potential energy scale U : For the potential energy scale U (Right of Figure 2.5). Compared with $U = 0$, the range of eigenvalues are wide and the distribution of the eigenvalues are also more diverse.
5. Kinetic energy scale t : For different t , there are same “ring” structure, but the range of the eigenvalues on same ray are change significantly: From our Theorem 18, the ranges of eigenvalues on the same ray are (rounded to four digits):

$$[\min |1 - \lambda(M)|, \max |1 - \lambda(M)|] = \begin{cases} [0.6045, 1.6487], & t = 1, \\ [0.0183, 54.5982], & t = 8. \end{cases}$$

The numerical results exactly fit in Theorem 18.

2.4 Condition numbers

When $U = 0$, $B_1 = B_2 = \dots = B_L = B = e^{t\tau K}$, and B is symmetric.

Lemma 19 *When $U = 0$, the eigenvalues of $M^T M$ are*

$$\lambda_{k,\ell}(M^T M) = 1 + 2\lambda_k(B) \cos \theta_\ell + (\lambda_k(B))^2, \quad (2.4.18)$$

where θ_ℓ is defined as

$$\theta_\ell = \frac{(2\ell + 1)\pi}{L}, \quad \ell = 0, 1, \dots, L - 1.$$

PROOF: For any real number a , define a matrix A

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

The eigenvalues of the matrix $A(a)$ is

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

Note that for any eigenvalue $\lambda(B)$ of B , the corresponding eigenvalues of $M^T M$ are equal to $\lambda(A(\lambda(B)))$, therefore the eigenvalues of $M^T M$ are given by (2.4.18). \blacksquare

Note that for any real number a ,

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

Therefore, we have the following inequalities

$$\max \lambda(M^T M) \leq (1 + \max |\lambda(B)|)^2 \quad \text{and} \quad \min \lambda(M^T M) \geq \sin^2 \frac{\pi}{L}.$$

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

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

and

$$\|M^{-1}\| = \frac{1}{\min \lambda(M^T M)^{\frac{1}{2}}} \leq \frac{1}{\sin \frac{\pi}{L}}. \quad (2.4.19)$$

Note that $B = e^{t\tau K}$ and $\lambda_{\max}(K) = 4$, we have the following theorem.

Theorem. 20 *When $U = 0$, the condition number of M is bounded by*

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

Example: for $t\beta = 20$ and $L = 160$, $\kappa(M) \approx 424$.

When $U \neq 0$, by the expression $M = I - \text{diag}(B_\ell)P$, we have a bound of the norm of M

$$\|M\| \leq 1 + \max_\ell \|B_\ell\| \|P\| = 1 + \max_\ell \|B_\ell\| \leq 1 + e^{4t\tau + \nu}. \quad (2.4.20)$$

To bound $\|M^{-1}\|$, we first consider when U is small. In this situation, the matrix M can be viewed as a perturbation of M at $U = 0$. We have the following result.

Theorem. 21 *If U is small such that*

$$e^\nu < 1 + \sin \frac{\pi}{L}, \quad (2.4.21)$$

then

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

PROOF: M can be expanded at $U = 0$:

$$M(U) = M(0) + \text{diag}(e^{t\tau K} - B_\ell)P.$$

Note that $\|P\| = 1$, then if $\|M(t, 0)^{-1} \text{diag}(e^{t\tau K} - B_\ell)\| < 1$, we have

$$\|M(t, U)^{-1}\| \leq \frac{\|M(t, 0)^{-1}\|}{1 - \|M(t, 0)^{-1} \text{diag}(e^{t\tau K} - B_\ell)\|}. \quad (2.4.22)$$

Note that (see the proof when $U = 0$)

$$\|M(t, 0)^{-1} \text{diag}(e^{t\tau K})\| \leq \frac{1}{\sin \frac{\pi}{L}}.$$

Therefore

$$\|M(t, 0)^{-1} \text{diag}(e^{t\tau K} - B_\ell)\| \leq \|M(t, 0)^{-1} \text{diag}(e^{t\tau K})\| \|\text{diag}(I - e^{V_\ell})\| \leq \frac{e^\nu - 1}{\sin \frac{\pi}{L}}.$$

Now, if $\frac{e^\nu - 1}{\sin \frac{\pi}{L}} < 1$, i.e., $e^\nu < 1 + \sin \frac{\pi}{L}$, by (2.4.22) and (2.4.19), we have

$$\|M^{-1}\| \leq \frac{\frac{1}{\sin \frac{\pi}{L}}}{1 - \frac{e^\nu - 1}{\sin \frac{\pi}{L}}} = \frac{1}{\sin \frac{\pi}{L} + 1 - e^\nu}. \quad (2.4.23)$$

This ends the proof. ■

Note that the Taylor expansion of ν gives the expression

$$\nu = \sqrt{U\tau} + \frac{(U\tau)^{\frac{3}{2}}}{12} + O(U^2\tau^2). \quad (2.4.24)$$

Then to the first-order approximation, the condition (2.4.21) is

$$\sqrt{U} \leq \frac{\pi}{\beta} \sqrt{\tau} + O(\tau), \quad \text{or} \quad U \leq \frac{\pi^2}{\beta^2} \tau + O(\tau^{\frac{3}{2}}). \quad (2.4.25)$$

Therefore, to the first order approximation, we have a bound on the condition number of M :

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

A few remarks in order:

- If U is small enough, M is well-conditioned, namely, $\kappa(M) = \mathcal{O}(L)$, the same behavior in the case where $U = 0$.
- The condition number of M strongly depends on the approximation $\nu \approx \sqrt{U\tau}$. And if $\nu \approx U\tau$, the requirement (2.4.25) for U will be relaxed as: $U\beta \leq \pi$.
- Since $\beta = L\tau$, the constraint for U is equivalent to $U\beta L \leq \pi$. We will show that the constraint is reasonable if we require that the condition number of M increases slowly.

In general, it is still an open problem to give a rigorous sharp upper bound $\kappa(M)$ for a general $U \neq 0$?

Figure 2.6 shows the average condition numbers of M for 100 H-S field configurations (uniform two-point distribution) as a function of L for $U = 2, 4, 6$. The figure illustrates two key points concerning the transition from well-conditioned to ill-conditioned behavior. (1) When $U \neq 0$, 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. The first observation tells us the parameter L is critical to the difficult of our numerical linear algebra solvers. The second suggests that widely varying condition number might be encountered in the course of a simulation, and therefore that a solver might need to have the ability to adopt different solution strategies on the fly.

2.5 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 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

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

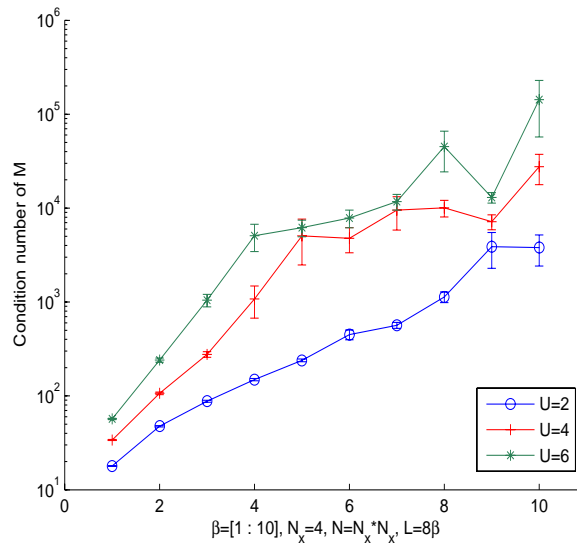


Figure 2.6: The condition number of M for $\beta = [1 : 10]$, $N = 16$, $L = 8\beta$, $t = 1$ and $U = 2, 4, 6$.

First we have two results:

1. Note that the block $B_\ell^{(k)}$ has the following bound:

$$\|B_\ell^{(k)}\| \leq e^{(4t\tau+\nu)k}.$$

Since the matrix $M^{(k)}$ has the same block cyclic structure as M , therefore, we have

$$\|M^{(k)}\| \leq 1 + e^{(4t\tau+\nu)k} \leq c e^{(4t\tau+\nu)k} \quad (2.5.26)$$

where c is a constant.

2. The inverse of $M^{(k)}$ is a “submatrix” of the inverse of M . Specifically, since M and $M^{(k)}$ have the same block structure, by the expression (2.2.8) of M^{-1} , we have

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

where

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

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

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

Obviously $(M^{(k)})_{i,j}^{-1} = M_{ik,jk}^{-1}$. If $i = L^{(k)}$,

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

which is equal to $M_{L,jk}^{-1}$. Therefore, $(M^{(k)})^{-1}$ is a “submatrix” of M^{-1} .

Based on this observation, we have

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

Returning to bound the condition number of the matrix $M^{(k)}$. We distinguish the cases where $U = 0$ and $U \neq 0$.

When $U = 0$. First, consider the case when the reduction factor $k = L$, i.e, the matrix M is reduced to a single block

$$M^{(L)} = I + B_L \cdots B_2 B_1 = I + B \cdots B B = I + B^L = I + (e^{t\tau K})^L = I + e^{t\beta K}.$$

Then the condition number of $M^{(L)}$ is given by the eigendecomposition of the matrix K :

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

Note that $M^{(L)}$ is extremely ill-conditioned when β is large.

When the reduction factor $k < L$, we have the following result, which can be proved in a similar way as the proof of Lemma 19.

Lemma 22 *If $U = 0$ and $L_k = \frac{L}{k}$ is an integer, then we have*

$$\kappa(M^{(k)}) \leq \frac{1 + e^{4t\tau k}}{\sin \frac{\pi}{L_k}}. \quad (2.5.27)$$

Figure 2.7 shows condition numbers of $M^{(k)}$ with respect to the reduction factor k when $U = 0$. The computational and estimated results fit well.

The bound seems still true even for L/k is not an integer as shown, although a rigorous bound and its proof have not been completed.

When $U \neq 0$. In general, $\kappa(M^{(k)})$ is bounded by

$$\kappa(M^{(k)}) = \|M^{(k)}\| \|(M^{(k)})^{-1}\| \leq \|M^{(k)}\| \|M^{-1}\| \leq \frac{\|M^{(k)}\|}{\|M\|} \cdot \kappa(M). \quad (2.5.28)$$

By (2.5.26), we have the bound of $\kappa(M^{(k)})$.

Lemma 23 *For $U \neq 0$, and $k < L$, we have*

$$\kappa(M^{(k)}) \leq ce^{k(4t\tau+\nu)} \kappa(M). \quad (2.5.29)$$

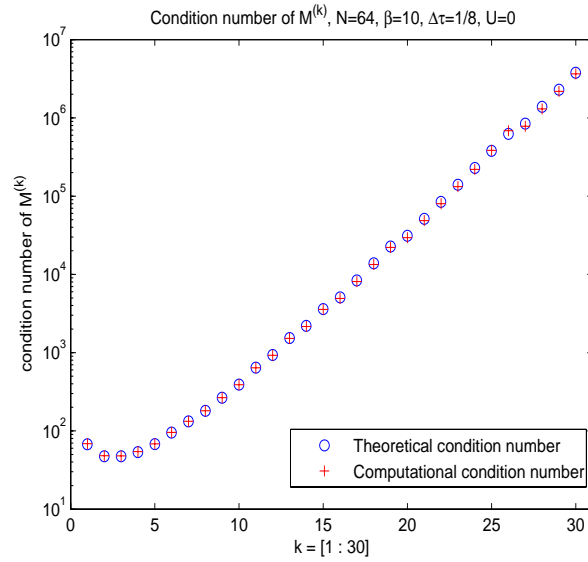
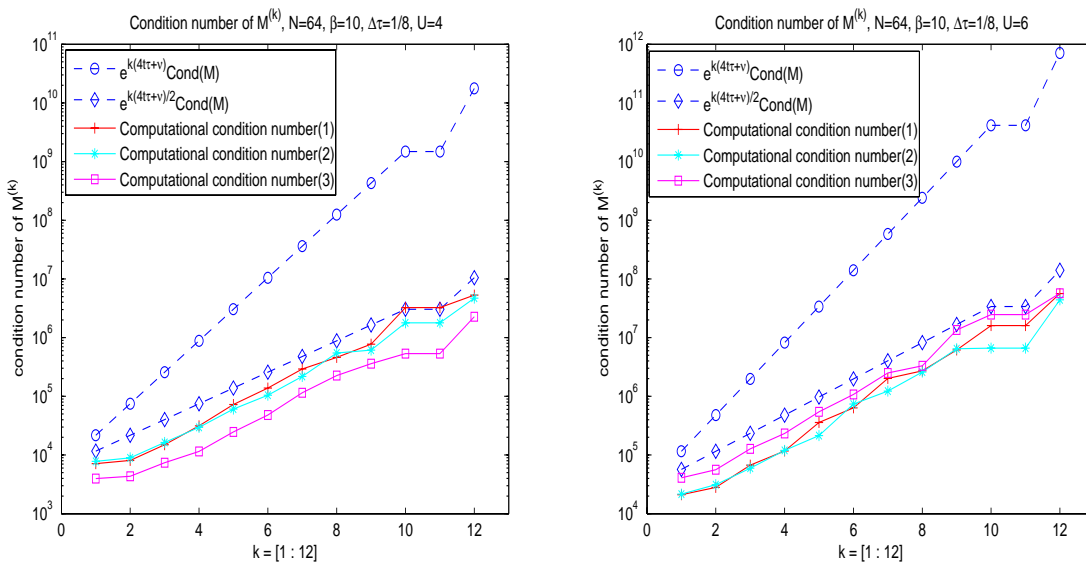
Figure 2.7: The condition number of $M^{(k)}$ for $U = 0$ Figure 2.8: The condition number of $M^{(k)}$ for $U = 4$ and $U = 6$

Figure 2.8 shows the condition numbers of a few sample matrices $M^{(k)}$ (solid lines), and the upper bound (2.5.29) (circle dashed line). The condition number $\kappa(M)$ of M uses the mean of the condition numbers of the sample matrices M . Clearly, the bound (2.5.29) is overestimated due to the over-estimation of the norm of $M^{(k)}$.

By Figure 2.8, we see that the condition number of $M^{(k)}$ is closer to the behavior of $e^{\frac{k}{2}(4t\tau+\nu)}\kappa(M)$ (diamond dashed line). This is a subject of further study.

Lecture 3

Self-adaptive direct linear system solvers

3.1 Introduction

In this lecture, we consider the computational kernel of the QMC simulations: solving the linear system of equations

$$Mx = b, \tag{3.1.1}$$

where the coefficient matrix M is the Hubbard matrix as defined in Lecture 2. One of main challenges in the multiscale QMC simulation is to develop algorithmic techniques and paradigms that can robustly and efficiently solve the linear system of equations with underlying multiscale coefficient matrices in a self-adapting fashion to achieve a required simulation accuracy.

The Hubbard matrix M exhibits the form of a so-called block p -cyclic consistently ordered matrix [7]. p -cyclic matrices arise in a number of important contexts in applied mathematics, including numerical solution of boundary value problems for ordinary differential equations [6], finite-difference equations for the steady-state solution of a parabolic equation with periodic boundary conditions [5], and computing the stationary solution of Markov chains with periodic graph structure [4].

It is known that the block Gaussian elimination with and without pivoting for solving p -cyclic linear systems can be numerically unstable, similar to the case of multiple shooting method for solving two-point boundary value problems [9, 2] and Markov chain modeling [3].

Block cyclic reduction [1] is a powerful idea to solve such p -cyclic system. However, a full block cyclic reduction is applicable only for small energy scales, namely, $U \leq 1$, due to the emerging of ill-conditioning of the reduced system. A stable p -cyclic linear system solver is based on the structural orthogonal factorization [8, 2]. Unfortunately, the costs of memory requirements and flops is prohibitively expensive when the length scales N and L increase.

To take advantage of significant reduction of memory requirement and floating point computations in the block cyclic reduction and numerical stability of the orthogonal factorization method, and to carefully examine the accuracy needs in our quantum monte carlo simulation, in this lecture, we present a hybrid method, which we simply call a *Self-Adaptive Block cyclic reduction Orthogonal factorization method*, or SABO method for short.

3.2 Block cyclic reduction

A-factor-of-four block cyclic reduction (BCR) of the following 16×16 block linear system

$$\begin{bmatrix} I & & & & & & & & & & & & & & & B_1 \\ -B_2 & I & & & & & & & & & & & & & & \\ & -B_3 & I & & & & & & & & & & & & & \\ & & -B_4 & I & & & & & & & & & & & & \\ & & & -B_4 & I & & & & & & & & & & & \\ & & & & \ddots & \ddots & & & & & & & & & & \\ & & & & & -B_{15} & I & & & & & & & & & \\ & & & & & & -B_{16} & I & & & & & & & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_{15} \\ x_{16} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_{15} \\ b_{16} \end{bmatrix}$$

leads to a block 4-cycle linear system of the same form

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

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 furthermore,

$$\begin{aligned} B_1^{(4)} &= B_4B_3B_2B_1 & b_1^{(4)} &= b_4 + B_4b_3 + B_4B_3b_2 + B_4B_3B_2b_1 \\ B_2^{(4)} &= B_8B_7B_6B_5 & b_2^{(4)} &= b_8 + B_8b_7 + B_8B_7b_6 + B_8B_7B_6b_5 \\ B_3^{(4)} &= B_{12}B_{11}B_{10}B_9 & b_3^{(4)} &= b_{12} + B_{12}b_{11} + B_{12}B_{11}b_{10} + B_{12}B_{11}B_{10}b_9 \\ B_4^{(4)} &= B_{16}B_{15}B_{14}B_{13} & b_4^{(4)} &= b_{16} + B_{16}b_{15} + B_{16}B_{15}b_{14} + B_{16}B_{15}B_{14}b_{13} \end{aligned}$$

Note that the number of equations of the reduced system has been reduced by a factor of 4.

After the vector $x^{(4)}$ is computed, i.e., the block components x_4, x_8, x_{12} and x_{16} of the solution x , the rest of block components of x can be computed by a mixed forward-backward substitution:

- 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 back and forward substitutions is to minimize the propagation of the computational errors.

In general, for an integer $k \leq L$, a-factor-of- k BCR leads to a block $L^{(k)}$ -cycle linear system,

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

where $M^{(k)}$ is a block L_k -cycle matrix:

$$M^{(k)} = \begin{bmatrix} I & & & & B_1^{(k)} \\ -B_2^{(k)} & I & & & \\ & -B_3^{(k)} & I & & \\ & & \ddots & \ddots & \\ & & & -B_{L_k}^{(k)} & I \end{bmatrix}, \quad (3.2.2)$$

with

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

The number of blocks is $L_k = \lceil \frac{L}{k} \rceil$. The corresponding $x^{(k)}$ and $b^{(k)}$ are

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

In particular, when $k = L$, $L_k = 1$, then we have

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

where $M^{(L)} = I + B_L B_{L-1} \cdots B_1$ and $b^{(L)} = b_L + \sum_{t=1}^{L-1} B_L \cdots B_{t+1} b_t$.

The reduced system (3.2.2) can be derived by using a block Gaussian elimination of the original system (3.1.1). Writing the matrix M 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 $k \times k$ block matrices defined as

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

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 $\widehat{D} = \text{diag}(D_1, D_2, \dots, D_{L_k})$, then

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

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

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

Therefore, $M^{(k)}$ is a submatrix of $\widehat{D}^{-1}M$. There exists a matrix Π , such that

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

where the matrix Π 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 $I_{NL \times NL}$.

After the solution vector $x^{(k)}$ of the reduced system is computed, the the rest of block components of x_i are obtained by a forward-back substitution, that is to say, when the index t is less than $\frac{k}{2}$, the components of block x_{ik+t} are obtained by forward substitution and otherwise by backward substitution.

1. Let $\xi = [k, 2k, \dots, (L_k - 1)k, L]$
2. For $j = 1, 2, \dots, L_k$
 - (a) $x_{\xi(j)} = x_j^{(k)}$

(b) forward substitution

For $\ell = \xi(j-1) + 1 : \xi(j-1) + \lceil \frac{1}{2}(\xi(j) - \xi(j-1) - 1) \rceil$ with $\xi(0) = 0$:

If $\ell = 1$

$$x_1 = b_1 - B_1 x_L$$

else

$$x_\ell = b_\ell + B_\ell x_{\ell-1}$$

(c) back substitution

For $\ell = \xi(j) - 1 : -1 : \xi(j) - \lfloor \frac{1}{2}(\xi(j) - \xi(j-1) - 1) \rfloor$,

$$x_\ell = B_{\ell+1}^{-1}(x_{\ell+1} - b_{\ell+1}).$$

3.3 Block orthogonal factorization method

Comparing with Gaussian elimination and block cyclic reduction, the block orthogonal factorization (BOF) method presented in this section is more expensive, but it is numerically backward stable.

By multiplying a sequence of orthogonal transformation matrices Q_i , M is transformed to an upper triangular, namely,

$$Q_{L-1}^T \cdots Q_2^T Q_1^T M = R, \quad (3.3.3)$$

where

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

and diagonal blocks $R_{\ell,\ell}$ are upper triangular. The orthogonal matrix Q_ℓ are defined as

$$Q_\ell = \begin{bmatrix} I & & & & \\ & \ddots & & & \\ & & Q_{11}^{(\ell)} & Q_{12}^{(\ell)} & \\ & & Q_{21}^{(\ell)} & Q_{22}^{(\ell)} & \\ & & & \ddots & \\ & & & & I \end{bmatrix}, \quad (3.3.4)$$

and the 2 by 2 diagonal block is the orthogonal factor of the QR decomposition:

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

where $\widetilde{M}_{\ell,\ell}$ are defined as

- for $\ell = 1$, $\widetilde{M}_{\ell,\ell} = I$

- for $\ell = 2, 3, \dots, L - 2$, $\widetilde{M}_{\ell,\ell} = (Q_{22}^{(\ell-1)})^T$

For the last step $\ell = L - 1$, we use the QR decomposition:

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

The following is a pseudo-code the block orthogonal factorization method to solve the system (3.1.1).

BOF method

1. Set $R_{1,L} = B_1$ and $c_1 = b_1$.
2. For $\ell = 1, 2, \dots, L - 2$,
 - (a) Compute the QR decomposition
$$\begin{bmatrix} M_{\ell,\ell} \\ M_{\ell+1,\ell} \end{bmatrix} = \begin{bmatrix} Q_{11}^{(\ell)} & Q_{12}^{(\ell)} \\ Q_{21}^{(\ell)} & Q_{22}^{(\ell)} \end{bmatrix} \begin{bmatrix} R_{\ell\ell} \\ 0 \end{bmatrix}.$$
 - (b) Compute $\begin{bmatrix} R_{\ell,\ell+1} \\ M_{\ell+1,\ell+1} \end{bmatrix} := \begin{bmatrix} Q_{11}^{(\ell)} & Q_{12}^{(\ell)} \\ Q_{21}^{(\ell)} & Q_{22}^{(\ell)} \end{bmatrix}^T \begin{bmatrix} 0 \\ M_{\ell+1,\ell+1} \end{bmatrix}$
 - (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) Compute $\begin{bmatrix} c_\ell \\ c_{\ell+1} \end{bmatrix} := \begin{bmatrix} Q_{11}^{(\ell)} & Q_{12}^{(\ell)} \\ Q_{21}^{(\ell)} & Q_{22}^{(\ell)} \end{bmatrix}^T \begin{bmatrix} c_\ell \\ b_{\ell+1} \end{bmatrix}.$
3. For $\ell = L - 1$,
 - (a) Compute the QR decomposition
$$\begin{bmatrix} M_{L-1,L-1} & R_{L-1,L} \\ M_{L,L-1} & M_{L,L} \end{bmatrix} = \begin{bmatrix} Q_{11}^{(L-1)} & Q_{12}^{(L-1)} \\ Q_{21}^{(L-1)} & Q_{22}^{(L-1)} \end{bmatrix} \begin{bmatrix} R_{L-1,L-1} & R_{L-1,L} \\ 0 & R_{L,L} \end{bmatrix}.$$
 - (b) Compute $\begin{bmatrix} c_{L-1} \\ c_L \end{bmatrix} := \begin{bmatrix} Q_{11}^{(L-1)} & Q_{12}^{(L-1)} \\ Q_{21}^{(L-1)} & Q_{22}^{(L-1)} \end{bmatrix}^T \begin{bmatrix} c_{L-1} \\ b_L \end{bmatrix}$
4. Solve the block triangular system $Rx = c$
 - (a) Solve $R_{L,L}x_L = c_L$ 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 $\ell = L - 2, L - 3, \dots, 1$, solve
$$R_{\ell\ell}x_\ell = c_\ell - R_{\ell,\ell+1}x_{\ell+1} - R_{\ell L}x_L$$
for x_ℓ .

Computational cost of the BOF method is $15N^3L$. The memory requirement of the R -factor is $3N^2L$.

3.4 A hybrid method

To take advantage of significant reduction of memory requirement and floating point computations in the BCR method and numerical stability of the BOF method, and to carefully examine the accuracy needs in our QMC simulation, we can use a hybrid method as the following:

Step 1. Perform a factor k block cyclic reduction:

$$Mx = b \implies M^{(k)}x^{(k)} = b^{(k)}.$$

Namely, the initial block L -cyclic system is cyclically reduced to a block $L^{(k)}$ -cyclic system, where $L^{(k)} = \frac{L}{k}$. Note that the conditioning number of $M^{(k)}$ increases with the increase of the reduction factor k .

Step 2. Solve the reduced cyclic system by the block structural orthogonal factorization

$$Q_{\frac{L}{k}-1}^T \cdots Q_1^T M^{(k)} = R.$$

In this way, the memory and computational costs are effectively reduced by a factor of k comparing to the original system.

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

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

We use both forward and back substitutions to minimize the propagation of errors induced at the steps 1 and 2.

Figure 3.1 shows the procedure of the method for a 16-block cyclic system with a reduction factor $k = 4$. We see that by Step 1, the order of $M^{(k)}$ is reduced by a factor of k , therefore, it is desirable that the larger k , the better. The computational cost is reduced from $O(N^3 L)$ to $O(N^3 \frac{L}{k})$, an effective factor k speedup. However, the condition number of $M^{(k)}$ increases when k increases, which means the accuracy of the computed solution decreases. Therefore, the critical question turns to how to find a reduction factor k , such that the computed solution has the required accuracy for the application. Such a reduction factor k should be determined in a *self-adapting* fashion with respect to the changes of underlying problem length and energy scales.

3.5 Self-adaptive reduction factor k

The goal of this section is to determine the reduction parameter k in a self-adapting fashion with respect to the changes of underlying parameters.

First, based on the well-known result of error analysis of the linear system, we know that after the solution $x^{(k)}$ of the reduced system obtained by the block orthogonal factorization method, its relative error is governed by $\kappa(M^{(k)})\epsilon$, where ϵ is the machine precision.

We now consider the propagation of the error the computed solution $x^{(k)}$ in the back and forward substitutions. For example, the computed x_L is

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

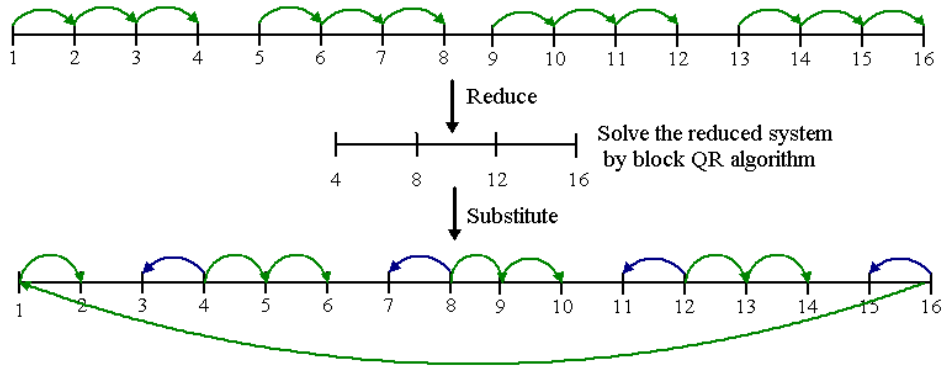


Figure 3.1: Scheme map of the block cyclic reduction and orthogonal factorization method for a 16-cyclic system with reduction factor $k = 4$.

where

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

Then by the forward substitution to compute x_1 , this error could be amplified to $\|B_1\|\kappa(M^{(k)})\epsilon$ in the computed x_1 ,

$$\hat{x}_1 = b_1 - B_1\hat{x}_L = b_1 - B_1(x_L + \delta x_L) = b_1 - B_1x_L + B_1\delta x_L = x_1 + \delta x_1,$$

where

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

Similarly, the relative error of the computed solution x_2 is bounded by $\|B_2\|\|B_1\|\kappa(M^{(k)})\epsilon$.

At the end of forward substitution, the relative error of the computed solution $x_{\frac{k}{2}}$ is bounded by $\|B_{\frac{k}{2}}\| \cdots \|B_2\| \|B_1\| \kappa(M^{(k)})\epsilon$.

In a summary, the errors in all computed components x_ℓ of the solution x are bounded by

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

By the upper bound (2.5.29) of the condition number $\kappa(M^{(k)})$ of the matrix $M^{(k)}$, we have

$$\frac{\|\delta x_\ell\|}{\|x_\ell\|} \leq ce^{\frac{1}{2}k(4t\tau+\nu)} \cdot e^{k(4t\tau+\nu)} \kappa(M)\epsilon = ce^{\frac{3}{2}k(4t\tau+\nu)} \kappa(M)\epsilon.$$

On the other hand, a user desired accuracy of the solution vector x can be specified by

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

Combining these analyses, for a desired accuracy “tol” of the solution vector x with machine precision ϵ , then a reduction factor k can be *self-adaptively* determined by

$$k = \left\lceil \frac{\frac{2}{3} \ln(\text{tol}/\epsilon)}{4t\tau + \nu} \right\rceil.$$

Note that when $U = 0$, $\nu = 0$. Otherwise, ν is given by (2.4.24).

Here we drop the factor of $\ln \kappa(M)$ in deciding reduction factor k , the reason is that as we discussed in section 2.4, $\kappa(M)$ grows slowly and is not large in the range of parameters of interest.

The following table shows the reduction factor k for $U = 0$ to 6, where $t = 1$, $\tau = \frac{1}{8}$, $\text{tol} = 10^{-8}$ and $\epsilon = 10^{-16}$.

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

In practice, to balance the number of the matrices B_ℓ involved in the product $B_\ell^{(k)}$, after k is computed as above, then we compute

$$L_k = \left\lceil \frac{L}{k} \right\rceil.$$

The final k is adjusted as

$$k = \left\lceil \frac{L}{L_k} \right\rceil \quad (3.5.5)$$

In summary, we have the following self-adapting block cyclic reduction orthogonal factorization method, SABO in short, to solve the linear system (3.1.1) to satisfy the user-specified relative accuracy tol .

SABO method

1. Determine the reduction factor k by (3.5.5),
2. Reduce (M, b) to $(M^{(k)}, b^{(k)})$ by BCR,
3. Solve the reduced system $M^{(k)}x^{(k)} = b^{(k)}$ by the BOF method.
4. Use forward and back substitutions to compute the remaining solution components.

3.6 Numerical experiments

In all experiments, it is required that the self-adapting direct solver has the relative accuracy at the order of $\sqrt{\epsilon}$:

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

Performance data are from an Intel Itanium2 machine, with 1.5GHZ CPU and 2GB core memory.

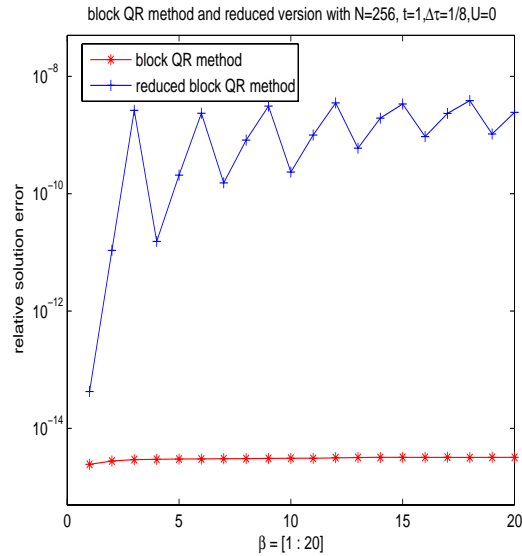


Figure 3.2: The relative error of the solution. $t = 1, U = 0, \beta = [1 : 20], \tau = 1/8, L = 8\beta$.

Experiment 1. In this experiment, we focus on the issues of robustness and stability when $U = 0$.

- The left plot of Figure 3.2 shows (1) the relative error of solutions by the BOF factorization method is at the order 10^{-15} . It indicates that the linear systems is well-conditioned and the BOF is backward stable. (2) the relative errors by the SABO solver are under 10^{-8} , as we requested.
- The CPU timing and other data are reported in Table 3.1.

The reduction factor k : $\beta \leq 3$, the SABO solver reduces the original system all the way to one block $M^{(L)}$. Since the flops of one QR decomposition is $\frac{4}{3}N^3$, the final flops is $\frac{4}{3}N^3 + 13N^2L$. For example, if $N = 256, L = 8$, the speedup will be larger than 60.

For large β , the SABO solver reduces to 6 or 7 blocks. For example, when $\beta = 20, L = 160$, it reduces to $L_k = \lfloor \frac{160}{23} \rfloor + 1 = 7$ with $k = 23$.

Experiment 2. In this experiment, we focus on the robustness and stability of the SABO solver when $U \neq 0$.

As shown in Figure 3.3, the solver works well for $\beta \in (0, 20]$ and $U \in [0, 6]$. The relative solution errors are all under 10^{-8} as specified. The SABO solver keeps good stability for varied β and U with $t = 1$ and $\tau = \frac{1}{8}$.

Experiment 3. In this experiment, we examine computational efficiency with respect to the parameter L .

β	$L = 8\beta$	k	$L^{(k)}$	BOF (sec)	SABO (sec)	speedup (\times)
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

Table 3.1: Reduction factors k , CPU timing etc. $t = 1, U = 0, N = 256, \tau = \frac{1}{8}, L = \beta/\tau = 8\beta, \beta = [1 : 20]$.

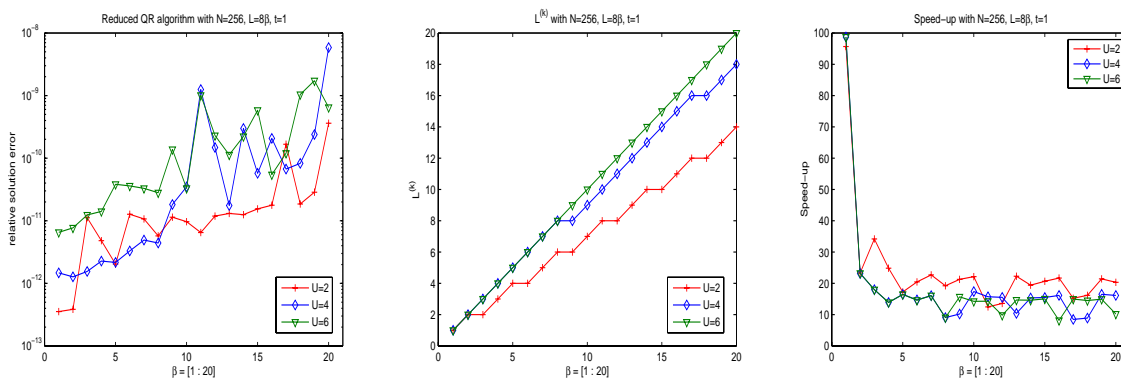


Figure 3.3: $U \neq 0$: Left: the relative error of the solution. Middle: L_k . Right: Speedup. Other parameters $t = 1, \beta = [1 : 20], \tau = 1/8, L = \frac{\beta}{\tau} = 8\beta$.

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

Table 3.2: CPU Timing with parameters: $\beta = [1 : 20]$, $t = 1$, $U = 6$, $N = 256$, $\tau = [\frac{1}{8}, \frac{1}{16}, \frac{1}{32}]$, and accuracy 10^{-8} . The dimensions of the matrices M vary from 2,048 ($\beta = 1, \tau = \frac{1}{8}$) to 163,840 ($\beta = 20, \tau = \frac{1}{32}$).

We have shown that the SABO solver is faster than the BOF method for fixed τ . The SABO solver is more efficient for smaller τ , see Table 3.2. For large energy scale parameters t , β and U , smaller τ are needed in order to get enough accuracy in approximation errors, such as the error of Trotter decomposition. Smaller τ implies larger L , $L = \frac{\beta}{\tau}$. For the SABO solver, smaller τ also implies larger reduction factor k . L increases linearly with $\frac{1}{\tau}$, but L_k only increases slowly.

Note that the flops of the BOF method is $O(15N^3L)$, and the flops of the SABO solver is $O(15N^3L_k)$. Therefore the SABO is more efficient for smaller τ , see Table 3.2 and Figure 3.4.

Experiment 4. In this experiment, we focus on the memory requirement with respect to the increase of the parameter N .

The memory requirement of the BOF method is $3N^2L = 3N_x^4L$. If $N_x = 32$, the memory storage of one $N \times N$ matrix is 8M bytes. Therefore for the machine with 2G memory, L cannot be larger than 85.

When $N_x = 32$, every $N \times N$ matrix needs 8M memory storage. If 1.5GB memory is available, then $L < 63$, which means for $\tau = \frac{1}{8}$, $\beta < 8$. Therefore, the BOF method will stop for $\beta \geq 8$. However, the SABO solver works for $\tau = \frac{1}{32}$, $U = 6$ and $\beta = [0 : 10]$, see Figure 3.3.

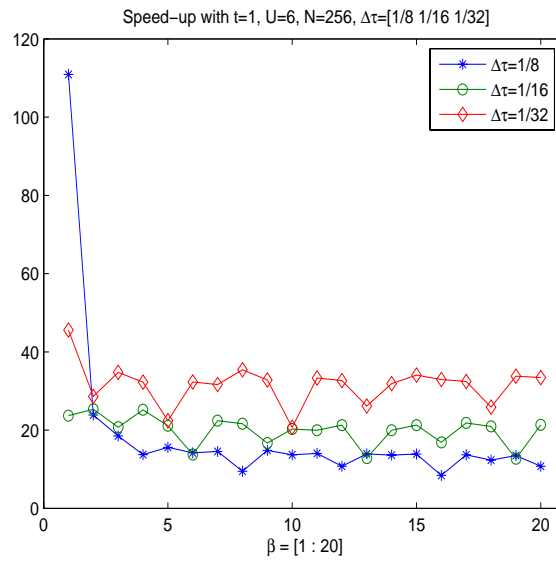


Figure 3.4: The speed up for different β and τ .

β	L	k	$L^{(k)}$	BOF	SABO	Speedup (\times)
1	8	8	1	148	2.10	70
2	16	8	2	322	17.8	18
3	24	8	3	509	40.1	12.7
4	32	8	4	689	64.5	10.6
5	40	8	5	875	88.6	9.8
6	48	8	6	1060	110	9.6
7	56	8	7	1250	131	9.5
8	64	8	8	out of memory	150	
9	72	8	9	out of memory	172	
10	80	8	10	out of memory	200	

Table 3.3: CPU Timing, where parameters $\beta = [1 : 10]$, $t = 1$, $U = 6$, $N = 1024$, $\tau = \frac{1}{8}$.

Bibliography

- [1] B.L. Buzbee, G.H. Golub, and C.W. Nielson. On direct methods for solving poisson's equations. *SIAM J.Numer. Anal.*, 7:627–656, 1970.
- [2] G. Fairweather and I. Gladwell. Algorithms for almost block diagonal linear systems. *SIAM Review*, 46(1):49–58, 2004.
- [3] B. Philippe and Y. Saad W.J. Stewart. Numerical methods in markov chain modeling. *Operations research*, 40(6):1156–1179, 1992.
- [4] W. J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.
- [5] G.J. Tee. An application of p -cyclic matrices for solving periodic parabolic problems. *Numerische Mathematik*, 6:142–159, 1964.
- [6] 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.
- [7] R.S. Varga. *Matrix iterative analysis*. Prentice-Hall, Englewood Cliffs, 1962. 2nd ed., Springer, Berlin/Heidelberg,2000.
- [8] S.J. Wright. Stable parallel algorithms for two-point boundary value problems. *SIAM J. Sci. Statist. Comput.*, 13(1):742–764, 1992.
- [9] S.J. Wright. A collection of problems for which gaussian elimination with partial pivoting is unstable. *SIAM J. Sci. Statist. comput.*, 14(1):231–238, 1993.

Lecture 4

Preconditioned iterative solvers

4.1 Introduction

In this lecture, we consider the preconditioned iterative methods for the solution of the linear system of equations of the form

$$M^T M x = b, \tag{4.1.1}$$

where M is the Hubbard matrix as introduced in Lectures 1 and 2. The linear system of such form is one of the computational kernels as we discussed in Lecture 1.

The popular GMRES, QMR and Bi-CGSTAB (without preconditioning) iterative methods have been considered for solving the couple linear systems $M^T y = b$ and $M x = y$. Preliminary study has found that these methods suffer from the slow convergence rate and erratic convergence behaviors. On the other hand, the convergence rate of CG is slow, however, it is robust in the sense that its residual errors decrease smoothly. Therefore, we decided to focus on the study of CG method [4] to solve the linear system of equations (4.1.1) directly.

For the rest of this lecture, we denote

$$A = M^T M.$$

The dimension of A is denoted as $n \times n$.

The convergence rate of CG is typically improved by a proper preconditioner R , which symmetrically preconditions the linear system of equations (4.1.1) such that

$$R^{-1} A R^{-T} \cdot R^T x = R^{-1} b, \tag{4.1.2}$$

where R is constructed to ideally satisfy the following three conditions:

- 1) The cost of constructing R is affordable.
- 2) $R R^T$ is a good approximation of A .
- 3) The application of R , namely, solving $R z = r$ for z , is not expensive.

Since $A = M^T M$, the matrix R is considered as a good preconditioner, if the matrix $M R^{-T}$ nearly orthogonal.

There are a number of approaches suggested to improve the convergence rate of the CG method by conditioning the linear system (4.1.1) with the matrix whose energy-scale parameter

$U = 0$ [3, 10]. The construction of this type of preconditioners requires only a small amount of computation. However, they are poor quality in term of the number of PCG iterations and computational costs. In this lecture, we focus on incomplete Cholesky (IC) factorizations of A and its variants to construct a preconditioner R .

We begin with a review of the Cholesky factorization, and then extend to incomplete Cholesky (IC) factorizations and robust incomplete Cholesky (RIC) factorizations.

4.2 Cholesky factorization

We consider Cholesky factorization of $n \times n$ symmetric positive definite (SPD) matrix matrix A ,

$$A = RR^T, \quad (4.2.3)$$

where R is an $n \times n$ lower-triangular matrix.

By the i th column of the factorization (4.2.3), we have

$$a_i = \sum_{j=1}^i r_j(i)r_j. \quad (4.2.4)$$

Equivalently, we have

$$r_i(i)r_i = a_i - \sum_{j=1}^{i-1} r_j(i)r_j. \quad (4.2.5)$$

Hence, to compute the i th column r_i of R , one first updates $a_i(i:n)$:

$$a_i(i:n) = a_i(i:n) - \sum_{j=1}^{i-1} r_j(i)r_j(i:n). \quad (4.2.6)$$

and then computes

$$\begin{aligned} r_i(i) &= \sqrt{a_i(i)}, \\ r_i(i+1:n) &= a_i(i+1:n)/r_i(i). \end{aligned}$$

Since the i th column r_i is computed based on the gaxpy operations (4.2.6), this procedure to compute Cholesky factor R is called **gaxpy** Cholesky in [4]. It is also referred as *left-looking* implementation since the i th column r_i is computed by updating with the previous columns r_1, \dots, r_{i-1} of R , which are on the left of r_i . The following is the pseudocode:

```

RIGHT-LOOKING CHOLESKY
for  $i = 1, \dots, n$  do
  for  $j = 1, \dots, i - 1$  do
     $a_i = a_i - r_j(i)r_j$ 
  end for
   $r_i(i) = \sqrt{a_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
     $r_i(j) = a_i(j)/r_i(i)$ 
  end for
end for

```

An alternative formulation of computing R is to initialize $A_0 = A$, and then recursively computes the i th column r_i of R based on the factorization:

$$A_{i-1} = \begin{bmatrix} \alpha_i & b_i^T \\ b_i & B_i \end{bmatrix} = \begin{bmatrix} \beta_i & 0 \\ b_i/\beta_i & I_{n-i} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A_i \end{bmatrix} \begin{bmatrix} \beta_i & b_i^T/\beta_i \\ 0 & I_{n-i} \end{bmatrix},$$

where A_{i-1} is an $(n-i+1) \times (n-i+1)$ matrix, B_i is an $(n-i) \times (n-i)$ matrix, b_i is an $(n-i)$ -length vector, $\beta_i = \sqrt{\alpha_i}$, and

$$A_i = B_i - \frac{b_i b_i^T}{\alpha_i}. \quad (4.2.7)$$

Hence, the matrix A_i is updated as soon as r_i is computed, where

$$\begin{aligned} r_i(i) &= \beta_i, \\ r_i(i+1:n) &= b_i/\beta_i. \end{aligned}$$

This procedure to compute Cholesky factor R is called **outer-product** Cholesky in [4] because the formulation of R is based on the repeated application of outer-product updates (4.2.7). It is also called *right-looking* Cholesky because at the i th iteration, the i th column r_i of R is used to update the remaining $(n-i+1) \times (n-i+1)$ matrix A_i , which are on the right of r_i . The following is the pseudocode:

```

RIGHT-LOOKING CHOLESKY
for  $i = 1, \dots, n$  do
   $r_i(i) = \sqrt{a_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
     $r_i(j) = a_i(j)/r_i(i)$ 
  end for
  for  $j = i + 1, \dots, n$  do
     $a_j = a_j - r_i(j)r_i$ 
  end for
end for

```

When the matrix M is preconditioned with Cholesky factor R , the preconditioned matrix MR^{-T} becomes orthogonal, and $R^{-1}AR^{-T}$ is an identity matrix. PCG for the solution of the linear system of equation (4.1.2) converges in a single iteration. Figure 4.1 shows that the eigenvalues of MR^{-T} are on the unit circle. However, both the memory requirement and computational cost of R are intractable for practical use.

4.3 Incomplete Cholesky factorization

To reduce the computational and storage costs of Cholesky factorization, we can use an incomplete Cholesky (IC) factorization:

$$A - E = RR^T, \quad (4.3.8)$$

where R is a lower-triangular matrix, and E is a symmetric error matrix. The goal is that by imposing a certain sparsity of R , it becomes a tractable cost (memory and flops) to construct the factor R , assume that E is small and R is a good approximation of Cholesky factor.

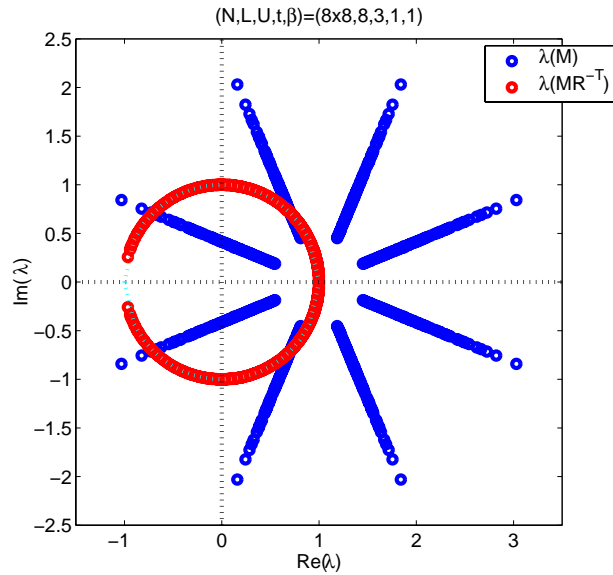


Figure 4.1: Eigenvalue distribution of M and MR^{-T} , where R is the Cholesky factor of $M^T M$.

By the i th column of equation (4.3.8), we have

$$a_i - e_i = \sum_{j=1}^i r_j(i)r_j. \quad (4.3.9)$$

Equivalently, it can be cast as

$$r_i(i)r_i - e_i = a_i - \sum_{j=1}^{i-1} r_j(i)r_j.$$

Thus, similar to left-looking Cholesky factorization, the i th column r_i can be computed by first updating the i th column a_i with the previously computed columns r_1, \dots, r_{i-1} ,

$$a_i(i:n) = a_i(i:n) - \sum_{j=i}^{i-1} r_j(i)r_j(i:n). \quad (4.3.10)$$

Then, the pivot of the i th column r_i of the incomplete Cholesky factor R is given by

$$r_i(i) = \sqrt{a_i(i)}.$$

The rest elements of r_i is computed by imposing a sparsity condition, i.e., for $j > i$,

$$\begin{cases} r_i(j) = a_i(j)/r_i(i), & s_i(j) = 0 & \text{if a sparsity constraint is satisfied,} \\ r_i(j) = 0, & s_i(j) = a_i(j) & \text{otherwise,} \end{cases}$$

where $E = S^T + S$, and S is a strictly lower-triangular matrix. The following is a pseudocode.

```

LEFT-LOOKING INCOMPLETE CHOLESKY
for  $i = 1, \dots, n$  do
  for  $j = 1, \dots, k - 1$  do
     $a_i = a_i - r_j(i)r_j$ 
  end for
   $r_i(i) = \sqrt{a_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
    if a sparsity constraint is satisfied then
       $r_i(j) = a_i(j)/r_i(i)$ 
    end if
  end for
end for

```

Alternatively, there is a right-looking IC factorization. The procedure first initializes $A_0 = A$, and then recursively computes the i th column r_i based on the partitioning of A_i such that

$$A_{i-1} = \begin{bmatrix} \alpha_i & b_i^T \\ b_i & B_i \end{bmatrix} = \begin{bmatrix} \beta_i & 0 \\ h_i/\beta_i & I_{n-i} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A_i \end{bmatrix} \begin{bmatrix} \beta_i & h_i^T/\beta_i \\ 0 & I_{n-i} \end{bmatrix} + \begin{bmatrix} 0 & g_i^T \\ g_i & 0_{n-i} \end{bmatrix}, \quad (4.3.11)$$

where A_{i-1} is an $(n - i + 1) \times (n - i + 1)$ matrix, B_i is an $(n - i) \times (n - i)$ matrix, h_i and g_i are $(n - i)$ -length vector, and $\beta_i = \sqrt{\alpha_i}$.

The vectors h_i and g_i are computed as the following: for $j \leq n - i$,

$$\begin{cases} h_i(j) = b_i(j), & g_i(j) = 0, & \text{if a sparsity constraint is satisfied,} \\ h_i(j) = 0, & g_i(j) = b_i(j), & \text{otherwise,} \end{cases}$$

and the matrix A_i is computed by the rank-one update:

$$A_i = B_i - \frac{h_i h_i^T}{\alpha_i}.$$

Subsequently, the i th column of R is given by

$$\begin{aligned} r_i(i) &= \beta_i, \\ r_i(i + 1 : n) &= h_i/\beta_i, \quad s_i(i + 1 : n) = g_i. \end{aligned}$$

The following is the pseudocode.

```

RIGHT-LOOKING INCOMPLETE CHOLESKY
for  $i = 1, \dots, n$  do
   $r_i(i) = \sqrt{a_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
    if a sparsity constraint is satisfied then
       $r_i(j) = a_i(j)/r_i(i)$ 
    end if
  end for
  for  $j = i + 1, \dots, n$  do
     $a_j = a_j - r_i(j)r_i$ 
  end for
end for

```

Note that a sparsity constraint should be such that the factor R and S do not have a non-zero entry at common locations, i.e., they are structurally orthogonal,

$$\text{diag}(R^T S) = 0.$$

The following sparsity constraints are commonly used.

1. *Fixed sparsity pattern (SP)*, namely, the factor R has a prescribed sparsity pattern $\mathcal{L} = \{(i, j), \}$. If $(i, j) \in \mathcal{L}$, $r_i(j) \neq 0$. A popular sparsity pattern \mathcal{L} is based on the known sparsity pattern of the original A : $\mathcal{L} = \{(i, j) : a_i(j) \neq 0\}$. i.e., $e_i(j) = 0$ if $(i, j) \in \mathcal{L}$. The following is the pseudocode.

```

LEFT-LOOKING IC - SP
for  $i = 1, \dots, n$  do
  for  $j = 1, \dots, k - 1$  do
    for  $k : (i, k) \in \mathcal{L}$  do
       $a_i(k) = a_i(k) - r_j(i)r_j(k)$ 
    end for
  end for
   $r_i(i) = \sqrt{a_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
    if  $(i, j) \in \mathcal{L}$  then
       $r_i(j) = a_i(j)/r_i(i)$ 
    end if
  end for
end for

```

```

RIGHT-LOOKING IC - SP
for  $i = 1, \dots, n$  do
   $r_i(i) = \sqrt{a_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
    if  $(i, j) \in \mathcal{L}$  then
       $r_i(j) = a_i(j)/r_i(i)$ 
    end if
  end for
  for  $j = i + 1, \dots, n$  do
    for  $k : (j, k) \in \mathcal{L}$  do
       $a_j(k) = a_j(k) - r_i(j)r_i(k)$ 
    end for
  end for
end for

```

Note that $e_i(j)$ at positions $(i, j) \notin \mathcal{L}$ can be large, and the resulting R may not be a good approximation of Cholesky factor.

2. *Dropping small entries (DSE)*. Another approach is to discard the non-zero entries of R with their magnitude less than a specified dropping threshold τ . This constraint guarantees that the relative magnitude of non-zero entries in the error matrix is less than τ .

The following is the pseudocode with the dropping tolerance threshold τ :

```

LEFT-LOOKING IC - DSE
for  $i = 1, \dots, n$  do
  for  $j = 1, \dots, i - 1$  do
     $a_i = a_i - r_j(i)r_j$ 
  end for
   $r_i(i) = \sqrt{a_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
    if  $|a_i(j)| > \tau$  then
       $r_i(j) = a_i(j)/r_i(i)$ 
    end if
  end for
end for

```

```

RIGHT-LOOKING IC - DSE
for  $i = 1, \dots, n$  do
   $r_i(i) = \sqrt{a_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
    if  $|a_i(j)| > \tau$  then
       $r_i(j) = a_i(j)/r_i(i)$ 
    end if
  end for
  for  $j = i + 1, \dots, n$  do
     $a_j = a_j - r_i(j)r_i$ 
  end for
end for

```

Note that in this approach, the sparsity pattern of R is not known prior to the construction of R . All the non-zero entries of R need to be computed to compare with the threshold, and the computational cost of constructing R can be high. Furthermore, the number of fill-ins, thus the memory required to store R , cannot be easily predicted.

3. *Fixed number of entries (FNE)*. This is a strategy to fix the number of entries allowed in each columns of R (by keeping the m non-zero entries with the largest magnitudes). Thus, the memory requirement is controlled. The following is the pseudocode.

<pre> LEFT-LOOKING IC – FNE for $i = 1, \dots, n$ do for $j = 1, \dots, i - 1$ do $a_i = a_i - r_j(i)r_j$ end for $r_i(i) = \sqrt{a_i(i)}$ for $j = i + 1, \dots, n$ do if $a_i(j)$ is among the m largest in a_i then $r_i(j) = a_i(j)/r_i(i)$ end if end for end for </pre>	<pre> RIGHT-LOOKING IC – FNE for $i = 1, \dots, n$ do $r_i(i) = \sqrt{a_i(i)}$ for $j = i + 1, \dots, n$ do if $a_i(j)$ is among the m largest in a_i then $r_i(j) = a_i(j)/r_i(i)$ end if end for for $j = i + 1, \dots, n$ do $a_j = a_j - r_i(j)r_i$ end for end for </pre>
--	---

Note that if the distribution of the non-zero entries in R is irregular, i.e., some columns have many large entries while some columns have many small entries, then the magnitude of non-zero entries in the error matrix can be large. Also, the computation of R can be expensive because all the non-zero entries in R need to be computed so that the non-zero entries with largest magnitudes can be located.

Note that in all IC factorizations, when non-zero entries are discarded from R , the operations with the non-zero entries are eliminated from the remaining procedure to compute R , and they do not have to be stored. Thus, imposing a sparsity on R reduces the cost to compute R in term of both storage and computation.

Unfortunately, IC factorization can fail due to a pivot breakdown, namely, the operation $r_i(i) = \sqrt{a_i(i)}$ is invalid. This indicates that when the non-zero entries introduced into E result in the loss of positive-definiteness in $A - E$. The existence of IC factorization is theoretically proven only for some special classes of matrices, see [7, 6, 12].

4.4 Robust Incomplete Cholesky preconditioners

The primary goal of a Robust Incomplete Cholesky (RIC) factorization is to avoid the pivot breakdown by imposing the matrix $A - E$ in (4.3.8) remaining to be SPD

$$A - E = A^T - E^T > 0. \quad (4.4.12)$$

In this section, we will discuss several approaches.

4.4.1 RIC1

One simple approach to satisfy the condition (4.4.12) is by simple diagonal updates of E . Specifically, by writing the error matrix $E = D + S + S^T$, we construct R based on a factorization of the form

$$A = RR^T + E = RR^T + D + S + S^T, \quad (4.4.13)$$

where R is a lower-triangular matrix, D is a diagonal matrix, and S is a strictly lower-triangular matrix.

To impose condition (4.4.12), decomposition (4.4.13) is constructed so that the error matrix $-E$ is symmetric and positive semi-definite (SPSD),

$$-E = -E^T = -D - S - S^T \geq 0.$$

There are two approaches to construct the diagonal matrix D : *static* or *dynamic*. A static approach computes D prior to the construction of decomposition (4.4.13), while a dynamic approach computes D during the construction of the decomposition. An example of static approach include global shifting of diagonal entries of A where all diagonal entries of A are shifted by a same factor [6]. Another example of static approach is to set all positive off-diagonal elements of A to zero and to add the values to the corresponding diagonal elements [2].

An example of dynamic approach is local shifting of diagonal elements of A , where the non-positive pivots are replaced by some positive values [6]. An alternative approach updates the corresponding diagonal entries of D for every new non-zero entries introduced into S [1]. As an example to compute decomposition (4.4.13), we present this dynamic approach proposed in [1].

By the i th column of equation (4.4.13), we have

$$a_i(i) = \sum_{j=1}^i r_j(i)^2 + d_i(i), \quad (4.4.14)$$

$$a_i(i+1:n) = \sum_{j=1}^i r_j(i)r_j(i+1:n) + s_i(i+1:n). \quad (4.4.15)$$

These two equations can be rearranged such that

$$r_i(i)^2 + d_i(i) = a_i(i) - \sum_{j=1}^{i-1} r_j(i)^2,$$

and

$$r_i(i)r_i(i+1:n) + s_i(i+1:n) = a_i(i+1:n) - \sum_{j=1}^{i-1} r_j(i)r_j(i+1:n).$$

Thus, similar to left-looking formulation of IC factorization, the i th column r_i can be computed by first updating the i th column a_i with the previously computed columns r_1, \dots, r_{i-1} ,

$$a_i(i:n) = a_i(i:n) - \sum_{j=1}^{i-1} r_j(i)r_j(i:n).$$

Then, the i th column s_i is computed by imposing a sparsity on $a_i(i+1:n)$ with a dropping threshold σ_1 so that, for $i+1 \leq j \leq n$,

$$\begin{cases} a_i(j) = a_i(j), & s_i(j) = 0, & \text{if } \tau_{ij} > \sigma_1, \\ a_i(j) = 0, & s_i(j) = a_i(j), & \text{otherwise,} \end{cases}$$

where¹

$$\tau_{ij} = \left[\frac{a_i(j)^2}{(a_i(i) + d_i(i))(a_j(j) + d_j(j))} \right]^{1/2}.$$

To impose the positive semi-definiteness of $-E$, the corresponding diagonal entries $d_i(i)$ and $d_j(j)$ are updated for every discarded non-zero element $s_i(j)$,

$$d_i(i) = d_i(i) - \delta_i, \quad d_j(j) = d_j(j) - \delta_j,$$

where δ_i and δ_j are chosen such that $\delta_i, \delta_j > 0$ and $\delta_i \delta_j = s_i(j)^2$. Finally, the diagonal element is

$$r_i(i) = \sqrt{a_i(i) + d_i(i)},$$

and the off-diagonal elements are

$$r_i(i+1:n) = a_i(i+1:n)/r_i(i).$$

The following is the pseudocode.

```

RIC1 - LEFT-LOOKING
for  $i = 1, \dots, n$  do
  for  $j = 1, \dots, k-1$  do
     $a_i = a_i - r_j(i)r_j$ 
  end for
  for  $j = i+1, \dots, n$  do
     $\tau_{ij} = a_i(j)/[(a_i(i) + d_i(i))(a_j(j) + d_j(j))]^{1/2}$ 
    if  $\tau_{ij} \leq \sigma_1$  then
       $a_i(j) = 0$ 
       $d_i(i) = d_i(i) + \delta_i$ 
       $d_j(j) = d_j(j) + \delta_j$ 
    end if
  end for
   $r_i(i) = \sqrt{a_i(i) + d_i(i)}$ 
  for  $j = i+1, \dots, n$  do
     $r_i(j) = a_i(j)/r_i(i)$ 
  end for
end for

```

¹This is used in the paper by Ajiz and Jennings. We're considering to use alternate constraint test $\tau_{ij} = a_i(j)/\sqrt{a_i(i) + d_i(i)}$. This seems to follow our presentation and give a similar performance.

The IC factorization (4.4.13) can be also computed in a right-looking fashion. This is based on the partitioning of the matrix A_{i-1} as in equation (4.3.11). Here, the h_i and g_i are computed so that, for $j < n - i$,

$$\begin{cases} h_i(j) = b_i(j), & g_i(j) = 0, & \text{if } \tau_{ij} > \sigma_1, \\ h_i(j) = 0, & g_i(j) = b_i(j), & \text{otherwise.} \end{cases}$$

where

$$\tau_{ij} = \left[\frac{b_i(j)^2}{(a_i(i) + d_i(i))(a_{i+j}(i+j) + d_j(i+j))} \right]^{1/2}.$$

For every discarded non-zero element $g_i(j)$, the corresponding diagonal entries $d_i(i)$ and $d_j(i+j)$ are updated,

$$d_i(i) = d_i(i) - \delta_i, \quad d_j(i+j) = d_j(i+j) - \delta_{i+j},$$

where δ_i and δ_{i+j} are chosen such that $\delta_i, \delta_{i+j} > 0$ and $\delta_i \delta_{i+j} = g_i(j)^2$. Thus, the partitioning of A_{i-1} becomes

$$A_{i-1} = \begin{bmatrix} \alpha_i & b_i^T \\ b_i & B_i \end{bmatrix} = \begin{bmatrix} \beta_i & 0 \\ h_i/\beta_i & I_{n-i} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A_i \end{bmatrix} \begin{bmatrix} \beta_i & h_i^T/\beta_i \\ 0 & I_{n-i} \end{bmatrix} + \begin{bmatrix} -d_i(i) & g_i^T \\ g_i & 0_{n-i} \end{bmatrix}, \quad (4.4.16)$$

where $\beta_i = \sqrt{\alpha_i + d_i(i)}$ and $A_i = B_i - \frac{h_i h_i^T}{\beta_i^2}$. Finally, the i th column of the IC factor R is given by

$$\begin{aligned} r_i(i) &= \beta_i, \\ r_i(i+1:n) &= h_i/r_i(i), \end{aligned}$$

Correspondingly, $s_i(i+1:n) = g_i$.

The following is the pseudocode.

```

RIC1 - RIGHT-LOOKING
for  $i = 1, \dots, n$  do
  for  $j = i + 1, \dots, n$  do
    if  $\tau_{ij} \leq \sigma_2$  then
       $a_i(j) = 0$ 
       $d_i(i) = d_i(i) + \delta_i$ 
       $d_j(j) = d_j(j) + \delta_j$ 
    end if
  end for
   $r_i(i) = \sqrt{a_i(i) + d_i(i)}$ 
  for  $j = i + 1, \dots, n$  do
     $r_i(j) = r_i(j)/r_i(i)$ 
  end for
  for  $j = i + 1, \dots, n$  do
     $a_j = a_j - r_i(j)r_i$ 
  end for
end for

```

The existence of the factorization (4.4.13) can be proven by showing that for any n -length vector v , we have

$$v^T(-E)v = \sum_{i,j \text{ s.t. } s_i(j) \neq 0} (\sqrt{\delta_i}v(i) - \sqrt{\delta_j}v(j))^2 \geq 0,$$

Both left-looking and right-looking implementations guarantee that $-E$ is SPSD and that $A - E$ is SPD. Thus, this factorization is robust, and is called as RIC in version 1, or RIC1 in short.

Note that RIC1 is based on simple diagonal updates of E , the construction of R is computationally efficient.

To measure the quality of RIC1 preconditioner R , we note that the norm of the residue

$$R^{-1}AR^{-T} - I = -R^{-1}(D + S + S^T)R^{-T} = -R^{-1}ER^{-T}$$

can be amplified by the factor of $\|R^{-1}\|^2$ of the norm of the error matrix E . When approximating an ill-conditioned matrix A , especially for strong interacting energy scale ($U = 5, 6$), the norm $\|R^{-1}\|$ is large, and the resulting R is often a poor preconditioner.

If many non-zero entries of R need to be dropped, then RIC1 proposed by Ajiz and Jennings performs large number of diagonal updates, and the resulting preconditioner may not be a good approximation of Cholesky factor. In this situation, other approaches such as the global shifting of diagonal elements proposed by Manteuffel may perform better.

4.4.2 RIC2.

In [11], it is proposed that to improve the quality of R , one can construct R based on a factorization of the form

$$A = RR^T + E = RR^T + RF^T + FR^T, \quad (4.4.17)$$

where R is a lower-triangular matrix. The error matrix is written as $E = RF^T + FR^T$, F is a strictly lower-triangular matrix. Similar to IC factorization, a sparsity is imposed on R to reduce the computational cost.

By the i th column of equation (4.4.17), we have

$$a_i = \sum_{j=1}^{i-1} (r_j(i)r_j + r_j(i)f_j + f_j(i)r_j). \quad (4.4.18)$$

It can be rearranged such that

$$r_i(i)t_i = a_i - \sum_{j=1}^{i-1} (r_j(i)t_j + f_j(i)r_j),$$

where $t_j = r_j + f_j$. Thus, after the columns r_1, \dots, r_{i-1} and f_1, \dots, f_{i-1} are computed, the i th columns r_i and f_i can be computed by first updating the i th column a_i ,

$$a_i(i:n) = a_i(i:n) - \sum_{j=1}^{i-1} (r_j(i)t_j(i:n) + f_j(i)r_j(i:n)).$$

Then, the pivot element of r_i is

$$r_i(i) = \sqrt{a_i(i)},$$

and the off-diagonal elements of r_i are computed by imposing a sparsity on $a_i(i+1:n)$ with a dropping threshold σ_1

$$\begin{cases} r_i(j) = a_i(j)/r_i(i), & f_i(j) = 0, & \text{if } |a_i(j)|/r_i(i) > \sigma_1, \\ r_i(j) = 0, & f_i(j) = a_i(j)/r_i(i), & \text{otherwise.} \end{cases}$$

for $j \geq i+1$, This is an implementation in a *left-looking* fashion.

Alternatively, the factorization (4.4.17) can be constructed in a *right-looking* fashion. The procedure first initializes $A_1 = A$, then recursively computes the i th columns r_i and f_i based on the partitioning of the matrix A_{i-1} ,

$$A_{i-1} = \begin{bmatrix} \alpha_i & b_i^T \\ b_i & B_i \end{bmatrix} = \begin{bmatrix} \beta_i & 0 \\ h_i + g_i & I_{n-i} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A_i \end{bmatrix} \begin{bmatrix} \beta_i & h_i^T + g_i^T \\ 0 & I_{n-i} \end{bmatrix} - \begin{bmatrix} 0 & 0 \\ 0 & f_i f_i^T \end{bmatrix},$$

where A_{i-1} is an $(n-i+1) \times (n-i+1)$ matrix, B_i is an $(n-i) \times (n-i)$ matrix, b_i , h_i , and g_i are $(n-i)$ -length vectors, and $\beta_i = \sqrt{\alpha_i}$. Here, h_i and g_i are computed so that

$$\begin{cases} h_i(j) = b_i(j)/\beta_i, & g_i(j) = 0, & \text{if } |b_i(j)|/\beta_i > \sigma_1, \\ h_i(j) = 0, & g_i(j) = b_i(j)/\beta_i, & \text{otherwise,} \end{cases}$$

and

$$A_{i+1} = B_i - h_i h_i^T - h_i g_i^T - g_i h_i^T. \quad (4.4.19)$$

The i th column of the RIC factor R is given by

$$\begin{aligned} r_i(i) &= \beta_i, \\ r_i(i+1:n) &= h_i \end{aligned}$$

Correspondingly, $f_i(i+1:n) = g_i$.

The existence of decomposition (4.4.17) can be proven based on the observation that the matrix A_i in (4.4.19) can be written as

$$A_i = B_i - \frac{b_i b_i^T}{\alpha_i} + g_i g_i^T.$$

Since $B_i - \frac{b_i b_i^T}{\alpha_i} > 0$ when A_{i-1} is SPD, and $g_i g_i^T \geq 0$ for any real vector f_i , the matrix A_i is SPD. As a result, the diagonal entry α_{i+1} of the SPD matrix A_i is always positive, the pivot breakdown is avoided. The factorization (4.4.17) is thus robust, and it is called RIC - version 2, or RIC2 in short.

RIC2 often results in a good preconditioner R because the norm of the residue

$$R^{-1} A R^{-T} - I = -F R^{-T} - R^{-1} F^T$$

is amplified at most by the norm of $\|R^{-1}\|$ of the norm of the F .

Note that when $\sigma_1 = 0$ thus $F = 0$, factorization (4.4.17) becomes Cholesky factorization. When $\sigma_1 > 0$ and small non-zero entries are discarded from $r_i(i+1:n)$ into $f_i(i+1:n)$, then the cost of computing the decomposition is reduced by the cost of updating the remaining columns of A with the outer-product of f_i . However, the computation of RIC2 is generally still expensive due to the large number of fill-ins in $f_i(i+1:n)$.

4.4.3 RIC3

It is observed that many of the non-zero entries in F in factorization (4.4.17) have magnitude much smaller than the dropping threshold σ_1 . Therefore in [5], it is proposed to impose a sparsity on F with a secondary dropping threshold $\sigma_2 \ll \sigma_1$. Specifically, one constructs a preconditioner R based on the factorization of the form

$$A = RR^T + E = RR^T + RF^T + FR^T + D + S + S^T, \quad (4.4.20)$$

where R is a lower-triangular matrix, the error matrix $-E = -RF^T - FR^T - D - S - S^T$, F and S are strictly lower-triangular matrices, and D is a diagonal matrix.

By the i th column of (4.4.20), we have

$$a_i(i) = \sum_{j=1}^i (r_j(i))^2 + 2f_j(i)r_j(i) + d_i(i), \quad (4.4.21)$$

and

$$a_i(i+1:n) = s_i(i+1:n) + \sum_{j=1}^i (r_j(i)t_j(i+1:n) + f_j(i)r_j(i+1:n)), \quad (4.4.22)$$

where $t_j = r_j + f_j$. Similar to RIC2, we can have a SPD matrix $A - RF^T - FR^T$,

$$A - RF^T - FR^T > 0.$$

To have the positive semi-definiteness of the error matrix $-E$, one want the secondary error matrix $-E_2 = -D - S - S^T$ is SPSD,

$$-E_2 = -E_2^T = -D - S - S^T \geq 0.$$

This can be done by using the dynamic approach proposed by Ajiz and Jennings in [1].

Specifically, equations (4.4.21) and (4.4.22) can be rearranged such that

$$r_i(i)^2 + d_i(i) = a_i(i) - \sum_{j=1}^{i-1} (r_j(i))^2 + 2f_j(i)r_j(i),$$

and

$$r_i(i)t_i(i+1:n) + s_i(i+1:n) = a_i(i+1:n) - \sum_{j=1}^{i-1} (r_j(i)t_j(i+1:n) + f_j(i)r_j(i+1:n)).$$

Thus, similar to left-looking formulation of IC factorization, the i th columns r_i and f_i can be computed by first updating the i th column a_i with the previous columns,

$$a_i(i+1:n) = a_i(i+1:n) - \sum_{j=1}^{i-1} (r_j(i)t_j(i+1:n) + f_j(i)r_j(i+1:n)).$$

Then, the i th column s_i is computed by imposing a sparsity on $a_i(i+1:n)$ with the secondary dropping threshold σ_2 . Thus, for $j > i+1$,

$$\begin{cases} a_i(j) = a_i(j), & s_i(j) = 0 & \text{if } \tau_{ij} > \sigma_2, \\ a_i(j) = 0, & s_i(j) = a_i(j) & \text{otherwise,} \end{cases}$$

where

$$\tau_{ij} = \left[\frac{a_i(j)^2}{(a_i(i) + d_i(i))(a_j(j) + d_j(j))} \right]^{1/2}.$$

To impose $-E_2$ to be SPSD, the corresponding diagonal entries $d_i(i)$ and $d_j(j)$ are updated for every discarded non-zero element $s_i(j)$,

$$d_i(i) = d_i(i) - \delta_i, \quad d_j(j) = d_j(j) - \delta_j,$$

where δ_i and δ_j are chosen such that $\delta_i, \delta_j > 0$ and $\delta_i \delta_j = s_i(j)^2$. Then, the pivot element of the i column r_i of R is

$$r_i(i) = \sqrt{a_i(i) + d_i(i)},$$

and the remaining elements are computed by imposing the primary sparsity constraint on $a_i(i+1:n)$ with a dropping threshold σ_1

$$\begin{cases} r_i(j) = a_i(j)/r_i(i), & f_i(j) = 0, & \text{if } |a_i(j)|/r_i(i) > \sigma_1 \\ r_i(j) = 0, & f_i(j) = a_i(j)/r_i(i), & \text{otherwise.} \end{cases}$$

for $j \geq i+1$. This gives an implementation of the RIC3 in a *left-looking* fashion.

The factorization (4.4.20) can be also constructed in a *right-looking* fashion. The procedure first initializes $A_1 = A$, and recursively computes the i th columns r_i and f_i based on the partitioning of A_i in equation (4.3.11). Here, the h_i and g_i are computed so that

$$\begin{cases} h_i(j) = b_i(j), & g_i(j) = 0, & \text{if } \tau_{ij} > \sigma_1, \\ h_i(j) = 0, & g_i(j) = b_i(j), & \text{otherwise,} \end{cases}$$

where

$$\tau_{ij} = \left[\frac{b_i(j)^2}{(a_i(i) + d_i(i))(a_{i+j}(i+j) + d_j(i+j))} \right]^{1/2}.$$

For every discarded non-zero element $g_i(j)$, the corresponding diagonal entries $d_i(i)$ and $d_j(i+j)$ are updated,

$$d_i(i) = d_i(i) - \delta_i, \quad d_j(i+j) = d_j(i+j) - \delta_{i+j},$$

where δ_i and δ_j are chosen such that $\delta_i, \delta_j > 0$ and $\delta_i \delta_j = s_i(j)^2$. Thus, the partitioning of A_i becomes the one in equation (4.4.16). Furthermore, a sparsity is imposed on h_i such that the partitioning becomes

$$A_i = \begin{bmatrix} \alpha_i & b_i^T \\ b_i & B_i \end{bmatrix} = \begin{bmatrix} \beta_i & 0 \\ v_i + w_i & I_{n-i} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & A_{i+1} \end{bmatrix} \begin{bmatrix} \beta_i & v_i^T + w_i^T \\ 0 & I_{n-i} \end{bmatrix} + \begin{bmatrix} -d_i(i) & g_i^T \\ g_i & w_i w_i^T \end{bmatrix},$$

where $\beta_i = \sqrt{\alpha_i - d_i(i)}$. Here, v_i and w_i are computed such that

$$\begin{cases} v_i(j) = h_i(j)/\beta_i, & w_i(j) = 0, & \text{if } |h_i(j)|/\beta_i > \sigma_1, \\ v_i(j) = 0, & w_i(j) = h_i(j)/\beta_i, & \text{otherwise,} \end{cases}$$

and

$$A_{i+1} = B_i - v_i v_i^T - v_i w_i^T - w_i v_i^T. \quad (4.4.23)$$

Finally, the i -th columns of the RIC3 preconditioner R is given by

$$r_i(i) = \beta_i, r_i(i+1:n) = v_i$$

Correspondingly, and $f_i(i+1:n) = w_i$ and $s_i(i+1:n) = g_i$.

By construction, it is guaranteed that

$$A - RF^T - FR^T > 0, \quad \text{and} \quad -D - S - S^T \geq 0,$$

and $A - E$ is SPD, where $E = RF^T + FR^T + D + S + S^T$. Therefore, this factorization is robust, and it is named as RIC - version 3, or RIC3 in short.

RIC3 often results in a good preconditioner R when $\|S\|$ and $\|D\|$ are small enough, i.e., $\|S\|, \|D\| \leq \|F\|/\|R^{-1}\|$, and the norm of the residue

$$R^{-1}AR^{-T} - I = -FR^{-T} - R^{-1}F^T - R^{-1}(D + S + S^T)R^{-T}$$

is amplified at most by the factor of about $\|R^{-1}\|$.

Since sparsity is imposed on both R and F , the computational cost of RIC3 is reduced significantly from that of RIC2 decomposition.

4.5 Numerical results

In this section, we present numerical results on the performance of RIC1, RIC2 and RIC3 preconditioners discussed in this lecture. For testing purpose, the right-hand side vectors b of the linear systems are chosen so that the entries of solution vector x have uniformly distributed random values between 0 and 1, and the initial vector of PCG is the vector with all zero entries. The stopping criterion used for PCG iteration is $\|x_k - x\|_2/\|x\|_2 < 10^{-3}$. The numerical results reported are the average of five solutions.

The numerical experiments are performed on an HP Itanium 2 workstation with 1GHz CPU and 2GB main memory. All algorithms are implemented in FORTRAN. We used the fortran compiler `ifort` in Intel Math Kernel library and the optimization option `-O3` for the code compilation.

For RIC1, we present the dynamic approach proposed by Ajiz and Jennigs, RIC1_d and the static approach proposed by Manteuffel, RIC1_s. The fixed dropping thresholds $\sigma_1 = 10^{-2}$ and $\sigma_2 = 10^{-4}$ are used for RIC3L. For the rest of preconditioners, σ_1 is chosen so that R with similar density are obtained. For RIC1_s, the diagonal elements are shifted by the factor of σ_1 . For RIC3, the diagonal elements are shifted with the factor of $2 \times \sigma_2$, that often stabilizes the factorization and improves the quality of R [5]. The choices of dropping thresholds are not optimal in term of the total solution time, but it gives a good idea of the performances of the preconditioners.

For each preconditioner, we report

- the memory required for storing the preconditioner in CSC format (R)
- the memory requirement for the intermediate workspace (W),

	R (MB)	W (MB)	Itrs	P-time (s)	S-time (s)	T-time (s)
JACO	--	--	7,043	--	77.28	77.28
RIC1L _d	20.70	6.90	527	1.47	18.33	19.81
RIC1L _s	20.70	6.90	281	0.79	9.54	10.33
RIC2L						
RIC3R	20.64	34.48	242	22.86	8.27	31.14
RIC3L	20.64	161.18	242	5.24	8.46	13.71

Table 4.1: Performance data, $U = 3$

	R (MB)	W (MB)	Itrs	P-time (s)	S-time (s)	T-time (s)
JACO	--	--	7,043	--	77.28	77.28
RIC1L _d	20.19	6.73	7,750	1.43	260.22	261.65
RIC1L _s	20.61	6.87	3,642	0.79	125.97	126.77
RIC2L						
RIC3R	20.53	34.50	2,557	19.94	86.59	106.55
RIC3L	20.53	140.89	2,836	4.62	96.38	101.00

Table 4.2: Performance data, $U = 6$

- the number of PCG iterations required for solution convergence (Itrs),
- the CPU time for constructing R (P-time),
- the CPU time for the PCG iterations (S-time),
- the total CPU time (T-time).

Tables 4.1 and 4.2 report these performance data for $U = 3$ and $U = 6$, while the rest of parameters are fixed at $(N, L, t, \beta, \mu) = (32 \times 32, 80, 1, 10, 0)$.

Figure 4.2 compares the total CPU time required to solve the linear system of equations with RIC1L and RIC3L preconditioners.

By Tables 4.1 and 4.2 and Figure 4.2, we see that

- For small U , i.e., $U = 0, 1, 2$, or 3 , the linear system is relatively well-conditioned, RIC1L_s outperforms RIC3L in term of total CPU time, due to the computational efficiency of RIC1L.
- However, for large U , i.e., $U = 4, 5, 6$, the linear system is ill-conditioned. RIC3L outperforms RIC1L_s in term of total CPU time because higher quality of RIC3 preconditioner outweighs the extra cost spent in computing the preconditioner.

In addition, we have observed the linear-scaling complexity of the preconditioned iterative solver when the interaction energies of the system are moderate, i.e., $U = 0, 1, 2, 3$. Figure 4.3 and 4.4 show that the number of PCG iterations grows slowly with N when RIC1L_s and RIC3L preconditioners are used, respectively.

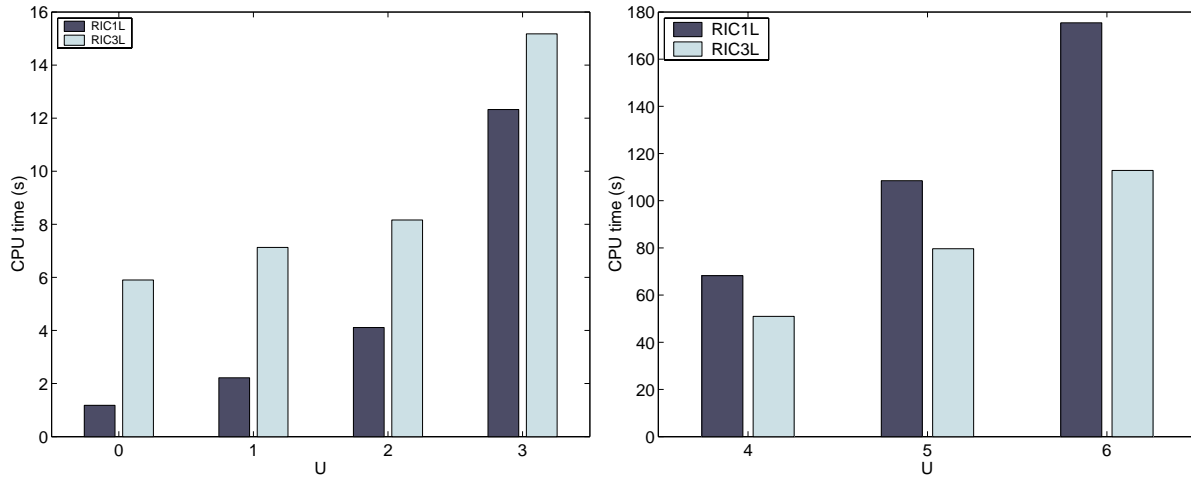


Figure 4.2: Comparison of the speed of RIC1 and RIC3 preconditioners. Left: for small U . Right: for large U . where $N = 32 \times 32$ and $L = 80$.

The right plots in Figure 4.5 and Figure 4.6 show that total CPU time required for the solution of the linear system of equations (4.1.1) scales linearly in term of N with these preconditioners.

For the system with strong interaction energies, i.e, $U = 4, 5, 6$, Figure 4.5 and 4.6 show that the number of PCG iterations required to achieve the desired solution accuracy grows linearly with N . However, the total CPU timing becomes more than linear in term of N .

It is still one crucial target to seek a preconditioner which can exhibit linear scaling behavior when the interaction energies are high. It is necessary to have a further understanding the convergence behavior of PCG under different length and energy scales. We have observed that for large energy scale, the PCG stagnates after some initial rapid decline. This stagnation of PCG convergence (plateau) is often due to the slow convergence of components of residual vectors associated with small eigenvalues, see Figure 4.7. Several techniques have been proposed to deflate these components from the residual vector so that the plateau of convergence can be avoided [8, 9].

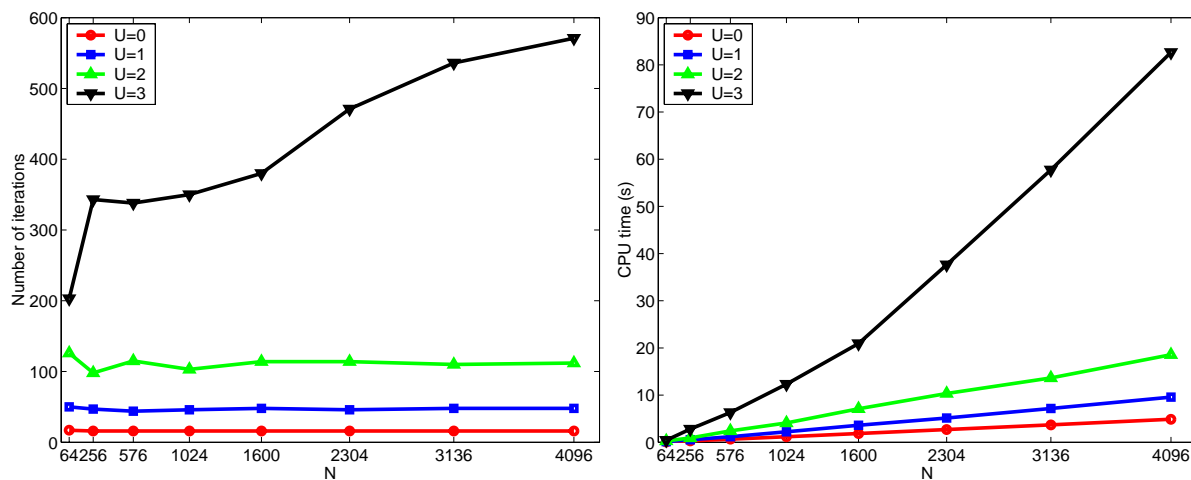


Figure 4.3: RIC1L₈: number of PCG iterations (left) and total CPU time (right) when $L = 80$.

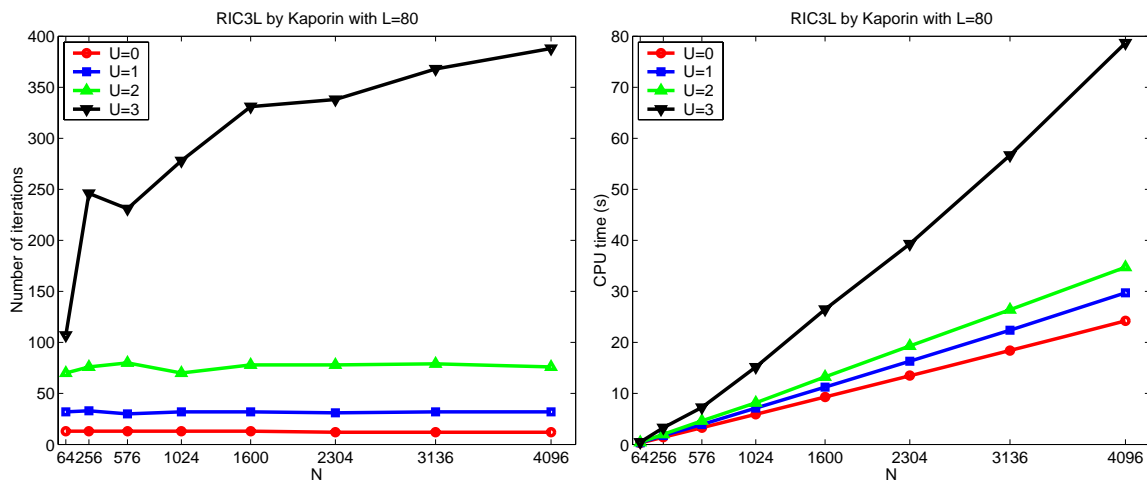


Figure 4.4: RIC3L: number of PCG iterations (left) and total CPU time (right) when $L = 80$.

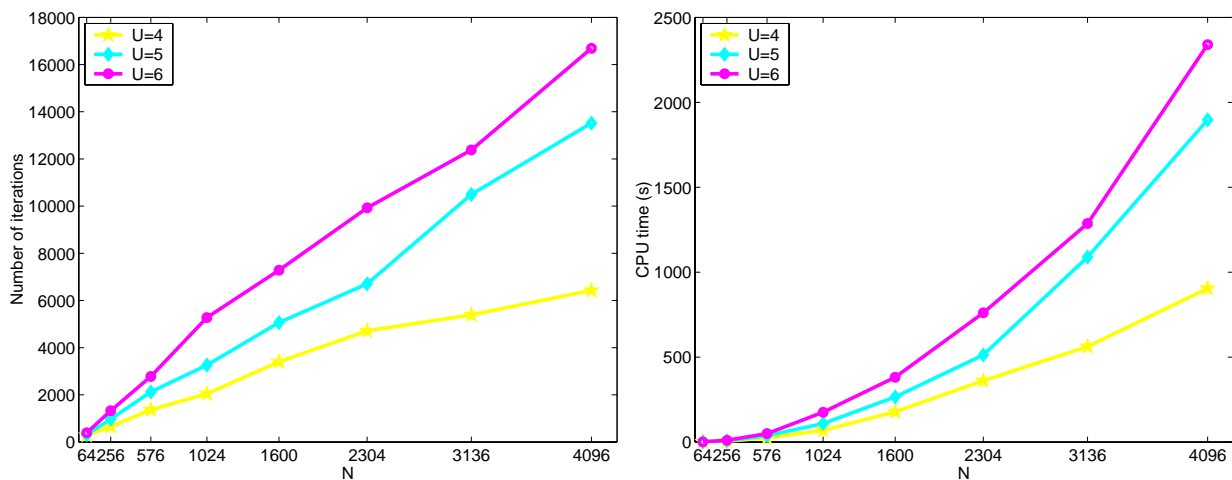


Figure 4.5: Number of PCG iterations (Left) and total CPU time (Right) for the solution of the linear system when $L = 80$. RIC1L_s preconditioner is used.

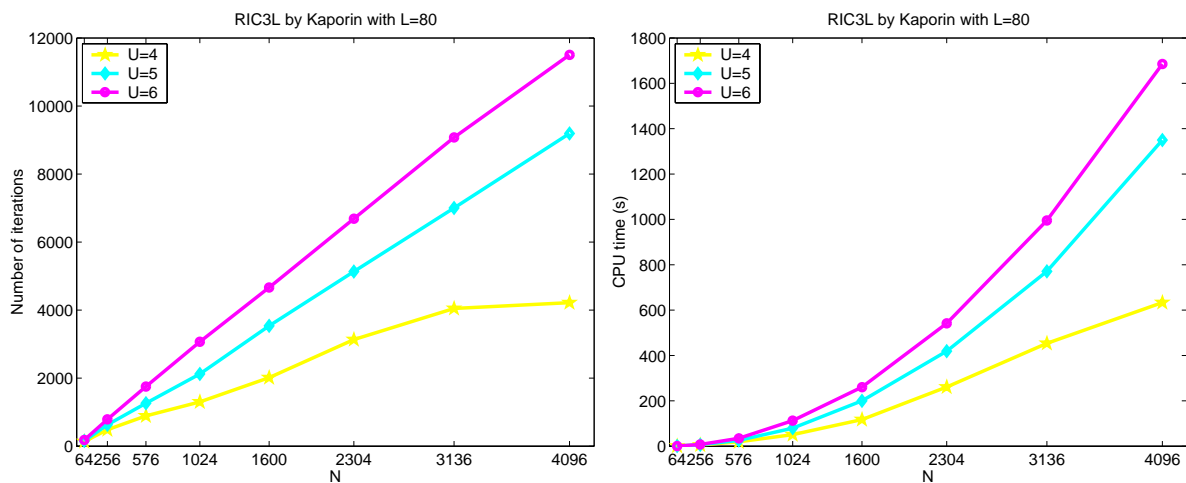


Figure 4.6: Number of PCG iterations (Left) and total CPU time (Right) for the solution of the linear system when $L = 80$. RIC3L preconditioner is used.

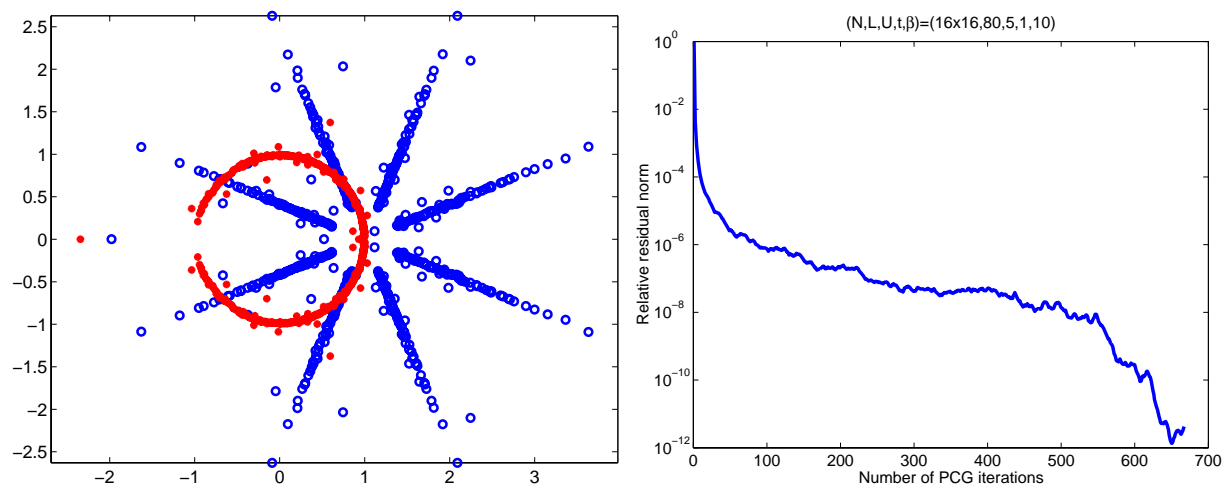


Figure 4.7: Left: eigenvalue distribution of the original matrix M and RIC3L preconditioned matrix MR^{-1} when $U = 6$. Right: initially, the relative residual norm of the PCG drops rapidly to reach to 10^{-6} in less than 100 iterations. However, it requires more than additional 500 iterations to drop to 10^{-8} .

Bibliography

- [1] AJIZ, M., AND JENNINGS, A. A robust incomplete choleski-conjugate gradient algorithm. *Internat. J. Numer. Methods Engrg.* 20, 5 (1984), 949–966.
- [2] AXELSSON, O., AND KOLOTILINA, L. Y. Diagonally compensated reduction and related preconditioning methods. *Numer. Linear Algebra Appl.* 1 (1994), 155–177.
- [3] BAI, Z., AND SCALETTAR, R. T. Itr: Advances of core numerical linear algebra techniques for quantum simulation in solid state physics. NSF proposal.
- [4] GOLUB, G. H., AND LOAN, C. F. V. *Matrix Computations*, 3 ed. Johns Hopkins University Press, Baltimore and London, 1996.
- [5] KAPORIN, I. E. High quality preconditioning of a general symmetric positive definite matrix based on its $u^t u + u^t r + r^t u$ -decomposition. *Numer. Linear Algebra Appl.* 5 (1998), 483–509.
- [6] MANTEUFFEL, T. A. An incomplete factorization technique for positive definite linear systems. *Mathematics of Computation* 34, 150 (1980), 473–497.
- [7] MEIJERINKAND, J., AND VAN DER VORST, H. A. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix. *Math. Comput* 31, 137 (1977), 134–155.
- [8] R. A. Nicolaides. Deflation of conjugate gradients with application to boundary value problems. *SIAM J. Numer. Anal.*, 24:355–365, 1987.
- [9] 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.*, 22:793–818, 2000.
- [10] SCALETTAR, R. T., SCALAPINO, D. J., AND SUGAR, R. L. A new algorithm for numerical simulation of fermions. *Phys. Rev. B, Condens Matter* 34 (1986), 7911–7917.
- [11] TISMENETSKY, M. A new preconditioning technique for solving large sparse linear systems. *Linear Algebra Appl.* 154–156 (1991), 331–353.
- [12] VARGA, R. S., SAFF, E. B., AND MEHRMANN, V. Incomplete factorizations of matrices and connections with h-matrices. *SIAM J. Numer. Anal.* 17 (1980), 787–793.

Lecture 5

The Monte-Carlo method

The Monte Carlo method is often referred to as a “computer experiment”. In this lecture, we discuss basic concepts of Monte Carlo simulation to understand how does it work in the QMC simulation described in Lecture 1.

5.1 Introduction

Let us begin with a simple toy example. Let x be a scalar, and $H(x)$ be defined by

$$H(x) = x^2.$$

We want to estimate¹ the integrals

$$I(n) = \frac{1}{Z} \int_{-\infty}^{\infty} x^n e^{-H(x)} dx, \quad n = 1, 2, \dots, \quad (5.1.1)$$

where Z is a constant

$$Z = \int_{-\infty}^{\infty} e^{-H(x)} dx = \sqrt{\pi}.$$

It is clear that the integrals $I(n)$ can be written as expectations:

$$I(n) = E[X^n], \quad (5.1.2)$$

where X is a random variable with probability density function (pdf) e^{-x^2}/Z .

One can approximate $I(n)$ by the Riemann sum. First, by restricting to the interval $[-a, a]$, the integrals $I(n)$ can be estimated by $I_a(n)$:

$$I_a(n) = \frac{1}{Z(a)} \int_{-a}^a x^n e^{-H(x)} dx,$$

where $Z(a) = \int_{-a}^a e^{-H(x)} dx$.

¹In fact, for this simple case, the integrals are known exactly

$$I(n) = \begin{cases} 0, & n = 2k + 1, \\ \frac{(2k-1)!!!}{2^k}, & n = 2k, \end{cases} \quad k = 1, 2, \dots$$

n	0	2	4	6	8
$I(n)$	1	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{15}{8}$	$\frac{105}{16}$
$I_a(n)$	1.0000	0.5000	0.7500	1.8750	6.5625
$I_a(n) - I(n)$	1.5e-12	4.0e-11	1.0e-9	2.7e-8	7.1e-7

Table 5.1: The integral $I(n)$ and its approximation $I_a(n)$, $a = 5$

Second, by taking the points $x_i = -a + \frac{2ai}{N}$ for $i = 0, 1, \dots, N$ and dividing the interval $[-a, a]$ into N subinterval $[x_i, x_{i+1}]$, then one can use the following Riemann sum $I_{a,N}(n)$ to approximate the integral $I_a(n)$ with order $O(\frac{1}{N})$:

$$I_a(n) \approx I_{a,N}(n) = \frac{2a}{N} \frac{1}{Z(a)} \sum_{i=1}^N (x_i)^n e^{-H(x_i)}.$$

The number $a = 5$ is a nice choice since $x^n e^{-x^2}$ decay rapidly, see Table 5.1.

Now let us discuss two simple sampling methods: uniformly sampling and Gaussian sampling.

If we draw independent and identically distributed (i.i.d.) random samples $x^{(i)}$ uniformly from $[-a, a]$. Then the probability $p(x)$ of $x = x^{(i)}$ is

$$p(x) = \begin{cases} \frac{1}{2a}, & x \in [-a, a], \\ 0, & \text{otherwise} \end{cases}$$

and an approximate to $I_a(n)$ can be obtained as

$$\begin{aligned} I_{N,U}(n) &= \frac{2a}{N} \frac{1}{Z(a)} \sum_{i=1}^N (x^{(i)})^n e^{-(x^{(i)})^2} \\ &= \frac{1}{N} \frac{1}{Z(a)} \sum_{i=1}^N \frac{(x^{(i)})^n e^{-(x^{(i)})^2}}{p(x^{(i)})}. \end{aligned}$$

Remark 24 If we sample $x^{(i)}$ uniformly on the subinterval $[x_i, x_{i+1}]$, then the Monte Carlo method can be regarded as the Riemann sum. It is an example of stratified sampling.

Alternatively, by (5.1.2), one can also draw random samples $x = x^{(i)}$ by the *Gaussian distribution*:

$$p(x) = \frac{e^{-x^2}}{Z}.$$

Then an approximation of the integral $I(n)$ can be obtained as the expectation value under the distribution $p(x)$

$$I_{N,G}(n) = \frac{1}{N} \sum_{i=1}^N (x^{(i)})^n.$$

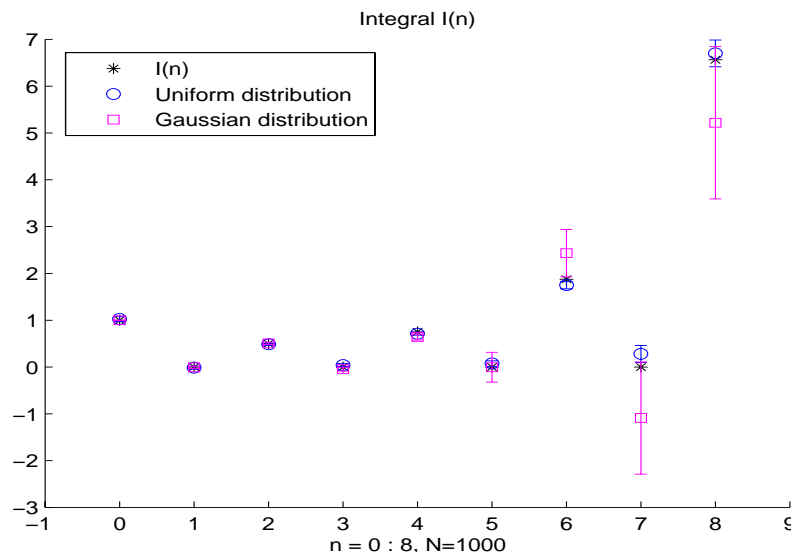


Figure 5.1: Simple Monte Carlo method for one dimensional integral

Figure 5.1 shows that both samplings are good approximation, however with different error bars. Their convergence rates are slower than the Riemann sum. This phenomenon can be explained by central limit theorem. For Gaussian sampling, the random variable $x^{(i)}$ is an i.i.d. sequence, and let the variance $\sigma^2(n) = \text{var}((x^{(i)})^n)$. Then the central limit theorem tells us that for a large N :

$$\frac{I_{N,G}(n) - I(n)}{\sqrt{\frac{\sigma^2(n)}{N}}} \sim N(0, 1) \quad \text{approximately.}$$

Therefore the "error term" of the monte carlo approximation is $O(\sqrt{\frac{\sigma^2}{N}})$, which is large than the approximation order $O(\frac{1}{N})$ of the Riemann sum. However, the convergence rate of the monte carlo approximation does not depend on the dimensionality of $x^{(i)}$, therefore for high dimension problem, the Monte Carlo method becomes attractive.

Remark 25 One can use the sample variance $S_N^2(n)$ to estimate the variance $\sigma^2(n)$:

$$S_N^2(n) = \frac{\sum_{i=1}^N ((x^{(i)})^n - I_{N,G}(n))^2}{N - 1}.$$

Then $E(S_N^2(n)) = \sigma^2(n)$ and $S_N^2(n)$ converge to $\sigma^2(n)$ with probability 1. The sample variance $\sqrt{S_N^2(n)/N}$ can be used as an error bar of the expectation (as shown in Figure 5.1). When N is sufficiently large, the 95% confidence interval for the expectation is:

$$I_{N,G}(n) \pm 1.96 \sqrt{S_N^2(n)/N}.$$

From Figure 5.1, the approximates by uniformly sampling and Gaussian sampling are different, especially the error bars (the variances). Since the convergence rate of the Monte Carlo method depends on the variance $\frac{\sigma}{\sqrt{N}}$, it is highly desired to reduce the variance σ^2 . This leads to so-called importance sampling principle.

Writing in a general notation, the goal of our simple example is to estimate

$$I = \int f(x)p(x)dx = E_p[f(X)],$$

where $X \sim p(x)$. By sampling X_i based on the $p(x)$, an estimator of I is

$$I_{N,p} = \frac{1}{N} \sum_{i=1}^N f(X_i)$$

with the variance

$$\text{var}(I_{N,p}) = \frac{1}{N} \left[\int f^2(x)p(x)dx - I^2 \right].$$

Choosing another p.d.f $\pi(x)$, and writing

$$I = \int \frac{f(x)p(x)}{\pi(x)} \pi(x)dx = E_\pi \left[\frac{f(Y)p(Y)}{\pi(Y)} \right],$$

where $Y \sim \rho$, then an importance sampling estimator based on ρ is

$$I_{N,\rho} = \frac{1}{N} \sum_{i=1}^N \frac{f(Y_i)p(Y_i)}{\rho(Y_i)},$$

where Y_i are i.i.d. with p.d.f. ρ . Its variance is

$$\text{var}(I_{N,\rho}) = \frac{1}{N} \left[\int \frac{f^2(x)p^2(x)}{\pi(x)} dx - I^2 \right].$$

Thus one can try to choose a proper p.d.f. $\rho(x)$ to reduce the variance.

To end this section, we briefly consider multidimensional integrals. Let H be a symmetric and positive definite matrix, and x be a column vector. Define function $H(x)$

$$H(x) = x^T H x.$$

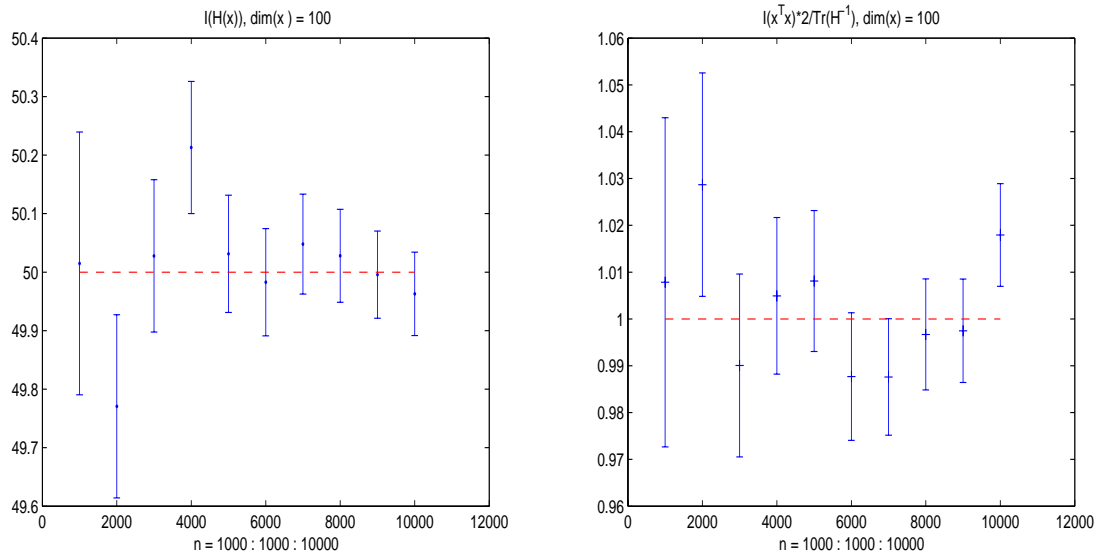
Consider the integral

$$I(x^T x) = \frac{1}{Z} \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_i x^T x e^{-H(x)} dx,$$

where

$$Z = \int e^{-H(x)} dx.$$

Figure 5.2: Simple Monte Carlo method for multidimensional integral



Remark 26 *The integral can be computed analytically:*

$$I(x^T x) = \frac{1}{2} \text{Tr}(H^{-1})$$

with

$$Z = \sqrt{\frac{\pi^{\dim(x)}}{\det(H)}},$$

Remark 27 *The following integral is independent of the matrix H :*

$$I(H) = \frac{1}{Z} \underbrace{\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty}}_i H(x) e^{-H(x)} dx = \frac{\dim(x)}{2},$$

The Riemann sum becomes intractable for high dimensional integrals, For example, if the dimension of the vector is 100 and only we only take 10 sample points in every direction, then the Riemann sum will need totally 10^{100} points. However, the Monte Carlo method can still be effectively used. By sampling X_i based on the the pdf $p(x) = \frac{1}{Z} e^{-H(x)}$, approximations of $I(x^T x)$ and $I(H(x))$ can be obtained, respectively

$$I_N(x^T x) = \frac{1}{N} \sum_{i=1}^N X_i^T X_i,$$

and

$$I_N(H(x)) = \frac{1}{N} \sum_{i=1}^N H(X_i).$$

Figure 5.2 shows that both two integrals can be efficiently approximated.

Remark 28 In any dimension, the Monte Carlo error goes as $\frac{1}{\sqrt{N}}$ where N is the number of points sampled. Also the time spent in the calculation is proportional to N , that is $T = cN$. Thus error = $\frac{1}{\sqrt{N}}$. On the other hand, if one is doing a Riemann sum by, for example, the trapezoid rule, the error goes as δ^2 where δ is the distance between the sampling points. In one dimension, the cpu time is $T = \frac{A}{\delta}$, so $\epsilon = \frac{1}{T^2}$. Clearly in one dimension the error goes down much faster with T ($1/T^2$ versus $1/\sqrt{T}$) using a Riemann sum method. But in dimension d , the Riemann cpu time is $T = \frac{A}{\delta^d}$, and hence error = $\frac{1}{T^{2/d}}$. If $d > 4$ the monte carlo $1/\sqrt{T}$ decays faster with T .

5.2 Monte Carlo simulation by Markov chain

In simple Monte Carlo methods, the samples $x^{(i)}$ are independent and identically distributed (i.i.d.). The requirement of independence is not necessary. The i th sample $x^{(i)}$ can be chosen to depend on the old samplings $x^{(i-1)}, x^{(i-2)}, \dots$. In particular, if the n th sample $x^{(n)}$ only depends on the previous $(n-1)$ th sample $x^{(n-1)}$:

$$x^{(0)} \rightarrow x^{(1)} \rightarrow \dots \rightarrow x^{(n-1)} \rightarrow x^{(n)} \rightarrow x^{(n+1)} \rightarrow \dots,$$

then all samples $x^{(i)}$ ($i = 0, 1, 2, \dots$) form a chain, it is called *Markov chain*, then a Monte Carlo simulation is called Markov chain Monte Carlo (MCMC) method. The MCMC method is often more efficient than simple Monte Carlo method described in the previous section.

The question now is how to move the sample from $x^{(n-1)}$ to $x^{(n)}$, that is to say, a transition rule should be made. Let us define the n -step transition function of a Markov chain as

$$P^{(n)} = (p_{xy}^{(n)})$$

with

$$p_{xy}^{(n)} = P(x^{(n)} = y | x^{(0)} = x).$$

Chapman and Kolmogorov have proven that

$$P^{(m+n)} = P^{(m)}P^{(n)}, \quad (5.2.3)$$

where $P = P^{(1)}$.

In MCMC method, one hopes that the transition rule will drive the samples x to arrive at a stationary state and the system will be in equilibrium. This requirement is often guaranteed in the practical MCMC methods and explained by the following convergence results.

If the state space of the Markov chain is finite and the chain is irreducible and aperiodic, then for any given initial state x , there exists a distribution $\pi(y)$ such that

$$\lim_{n \rightarrow \infty} P_{xy}^{(n)} = \pi(y).$$

Moreover from by the Chapman-Kolmogorov identity (5.2.3), the row vector $\{\pi(x)\}$ is the the left eigenvector of the matrix P :

$$\pi(x)P(x, y) = \pi(y).$$

This equation shows that if the state x is in equilibrium, after one transition step, the new state y will be still in equilibrium. This condition can be used to check whether the Markov chain arrives at stationary state, that is to say, whether the MCMC method converges.

Remark 29

- A Markov chain is irreducible if the chain can eventually get from each state to every other state, that is for every x and y there exists a $k > 0$ (depending on x and y) such that $P_{xy}^{(k)} > 0$.
- An irreducible Markov chain has period D if D is the greatest common divisor of $\{k \geq 1 : p_{xx}^{(k)} > 0\}$ for some state x . A Chain is called aperiodic if its period is 1.

In MCMC methods, the basic idea to move the configuration is by two-step moving strategy: try and adjust (or predict and correct)

$$x^{(n)} \xrightarrow[\text{try}]{T(x^{(n)}, y)} y \xrightarrow[\text{adjust}]{\text{Metropolis}} x^{(n+1)}$$

where

1. In the first step, try to move $x^{(n)}$ towards a new position y according to the trial transition rule (or trial proposal) $T(x^{(n)}, y)$.
2. In the second step, adjust y by $x^{(n+1)}$ to drive $x^{(n+1)}$ to the stationary state.

For any symmetric proposal function $T(x, y)$: $T(x, y) = T(y, x)$, Metropolis et al (1953) introduced famous acceptance-rejection strategy to adjust the movement:

Generate a random number $U \sim \text{Uniform}[0, 1]$. Accept y as $x^{(n+1)}$ if

$$U \leq r(x^{(n)}, y), \quad \text{where } r(x, y) = \frac{\pi(y)}{\pi(x)},$$

otherwise, reject the movement, and go back to old position: $x^{(n+1)} = x^{(n)}$.

Hasting extended the idea to nonsymmetric proposal function $T(x, y)$ by using a modifying the Metropolis ratio:

$$r(x, y) = \frac{\pi(y)T(y, x)}{\pi(x)T(x, y)}.$$

Now we introduce some basic trial proposals $T(x, y)$. More can be found in [1].

- *Flipping*. If the state x is a d -dimension vector: $x = \{x_1, \dots, x_s, \dots, x_d\}$, and every components x_s takes two possible values $\{t, -t\}$. By flipping, only one component is updated in every MC step:

$$y_s = -x_s^{(n)} \quad \text{and} \quad y_i = x_i^{(n)}, \quad i \neq s.$$

Flipping is used in DQMC.

- *Random-walk.* By random-walk, y is obtained by perturbing $x^{(n)}$:

$$y = x^{(n)} + \epsilon_n,$$

where ϵ_n is independent and identically distributed for different n . Typically, ϵ_n can be a spherically symmetric distribution. For example, by Gaussian distribution, $x^{(n)}$ is updated by

$$y = x^{(n)} + \sigma\epsilon, \quad \epsilon \sim N(0, I).$$

Here the parameters σ is controlled by the user.

- *Langevin-Euler moves.* In many physical applications, an energy function $H(x)$ is often defined. Then by Langevin-Euler moves, a procedure for updating $x^{(n)}$ is

$$y = x^{(n)} - \frac{h}{2} \frac{\partial H}{\partial x}(x^{(n)}) + \sqrt{h}\epsilon^{(n)},$$

where $\epsilon^{(n)}$ follows a standard Gaussian distribution. Note that if let $\sqrt{h} = dt$, then the Langevin updating can be regarded as the Euler discretization of the following stochastic differential equation:

$$dx_t = -\frac{1}{2} \frac{\partial H(x)}{\partial x} dt + dW_t, \quad (5.2.4)$$

where W_t is the standard Brownian motion, and the solution of the stochastic follows the target distribution $\pi(x) = \frac{e^{-H(x)}}{Z}$.

To understand why does the Metropolis algorithm work, we we need to do is to check the identity

$$\pi(x)P(x, y) = \pi(y). \quad (5.2.5)$$

By the Metropolis-Hasting algorithm, the new Markov chain has transition probabilities

$$p_{xy} = \begin{cases} T(x, y) \min\{1, r(x, y)\}, & \text{if } y \neq x \\ 1 - \sum_{z: z \neq x} T(x, z) \min\{1, r(x, z)\}, & \text{if } z = x \end{cases}$$

and is called the Metropolis chain for $\pi(x)$ with proposal matrix T . Note that here the diagonal elements are chosen such that the column sum of the transition matrix P equals to 1.

A easy-to-check but more restrictive condition than (5.2.5) is the so-called *detailed balance* condition:

$$\pi(x)P(x, y) = \pi(y)P(y, x).$$

Chains that satisfy the detailed balance condition are called *reversible*, since the detailed balance shows that the probability from the state x to the state y is same as from the state y to the state x .

Proposition 30 Consider an irreducible Markov chain with discrete state space \mathcal{S} . Assume that there exists positive number $\pi_i, i \in \mathcal{S}$, such that $\sum_i \pi_i = 1$ and

$$\pi_i p_{ij} = \pi_j p_{ji}, \quad \text{for every } i, j \in \mathcal{S},$$

then $\pi = (\pi_i)_{i \in \mathcal{S}}$ is the equilibrium distribution.

PROOF. Note that $\sum_i p_{j,i} = 1$, then by detailed balance,

$$(\pi P)_j = \sum_i \pi_i p_{ij} = \sum_i \pi_j p_{j,i} = \pi_j \sum_i p_{j,i} = \pi_j. \quad \square$$

For Metropolis-Hasting algorithm, $P(x, y)$ has been written down explicitly: for $x \neq y$

$$P(x, y) = T(x, y) \min\{1, r(x, y)\}. \quad (5.2.6)$$

Hence

$$\pi(x)P(x, y) = \min\{\pi(x)T(x, y), \pi(y)T(y, x)\},$$

which is a symmetric function in x and y . Thus, the detailed balance condition is satisfied.

The Metropolis algorithm is not unique way to generate the Markov chain, Barker (1965) suggested another acceptance function:

$$r_B(x, y) = \frac{\pi(y)T(y, x)}{\pi(y)T(y, x) + \pi(x)T(x, y)} = \frac{r(x, y)}{1 + r(x, y)},$$

and corresponding transition ($x \neq y$)

$$P(x, y) = \pi(y) \frac{T(x, y)T(y, x)}{\pi(y)T(y, x) + \pi(x)T(x, y)}.$$

When the trial proposal $T(x, y)$ is symmetric: $T(x, y) = T(y, x)$, then

$$r_B(x, y) = \frac{\pi(y)}{\pi(x) + \pi(y)}.$$

This method is often called "heat-bath" in the field of computational physics.

5.3 MCMC for multi-dimensional integrals

Now we use the MCMC method to compute the multi-dimensional integrals, treated by the simple Monte Carlo method in section 1. Here we use the random-walk:

$$y = x^{(n)} + \sigma R^{-1} \epsilon, \quad \epsilon \sim N(0, I).$$

where the upper triangle matrix R is the Cholesky factorization of the positive matrix H :

$$H = R^T R,$$

and R is computed before the MCMC simulation. The parameter σ is taken as 0.085 which is chosen such that the acceptance ratio is near $\frac{1}{2}$. For multi-dimensional integral problems, the acceptance ratio is sensitive to the parameter σ and the final results is better in $\frac{1}{2}$ acceptance ratio than others.

In the MCMC method, we should to determine when the system enter equilibrium. For example, Figure 5.3 records the values $(x^{(n)})^T H x^{(n)}$ for every state $x^{(n)}$, $n = 1, \dots, 2000$. It shows that states are in stationary state around $n = 500$. Then the integral $I(H)$ is computed as following:

$$I_{N,p}(H) = \frac{1}{N-p} \sum_{i=p+1}^N (x^{(i)})^T H x^{(i)}, \quad (5.3.7)$$

and p is set as 500.

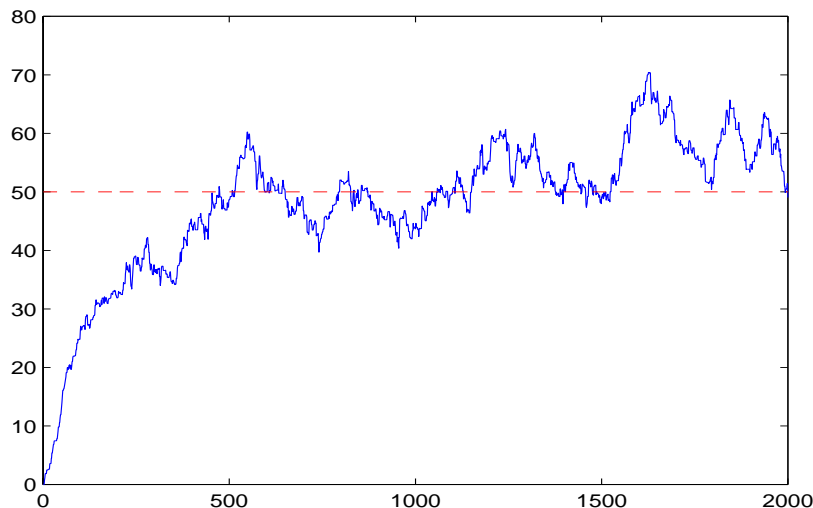


Figure 5.3: $(x^{(n)})^T H x^{(n)}$ for $n=1:2000$

Remark 31 *The first 500 steps in Figure 5.3 are often called the "warm-up" or "equilibration time" and that they are analogous to the time one would wait in an experiment between putting a sample in a cold chamber and starting to take data, because one would want to give the sample a chance to cool off to the temperature of the chamber.*

Figure 5.3 also shows that the state $x^{(n)}$ and the states $x^{(n+i)}, i = 1, \dots, k$ are dependent: states which are above the exact value of 50 tend to stay above 50 for a while, and ones below tend to stay below. Therefore the method of batch means is used to estimate means from a stationary sequence: Choose a "bin size" M , and average the data for x over each of the $L = \frac{N}{M}$ bins to create L "binned measurements" m_j (we should point out that N is the total number of measurements $N - p$):

$$m_j = \frac{1}{M} \sum_{i=M*(j-1)+1}^{M*j} x^{(i)}.$$

If M is sufficiently large, the new variables m_j are approximately independent, and denote $\sigma^2(m)$ the variance of the variables m_j , we use $\sqrt{\frac{\sigma^2(m)}{L}}$ as error bar. The formula for error bars

$$\sigma^2 = \frac{\langle x^2 \rangle - \langle x \rangle^2}{N - 1}$$

assumes the measurements are independent. If they are not, then we need to bin.

Figure 5.4 shows that the results with batch means are better than without the bin.

More details can be found in [3] and [2, Chap.5]

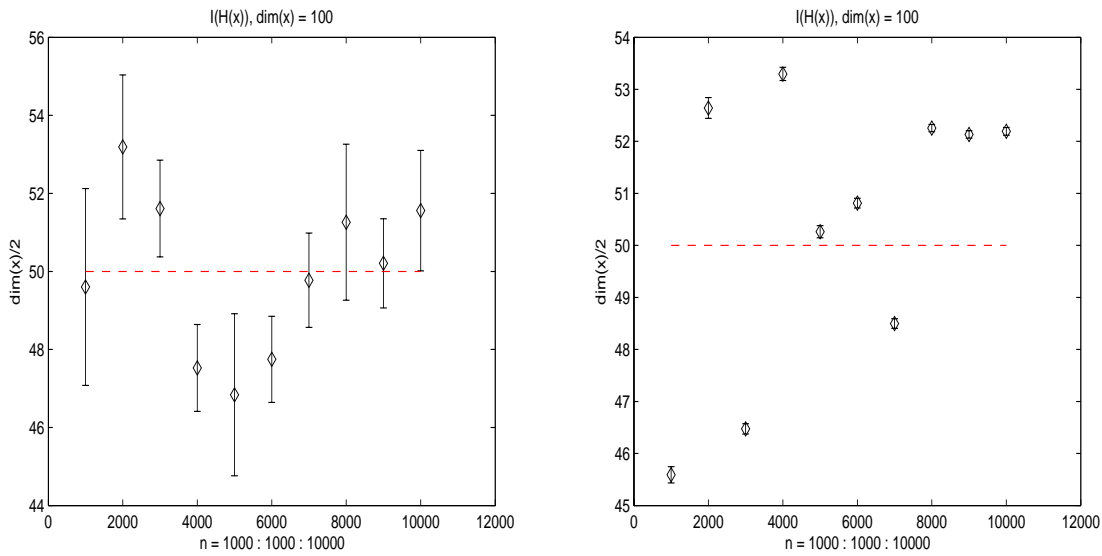


Figure 5.4: $I_{N,p}(H)$ with bin ($L=10$ in left figure) and without bin (right figure)

5.4 γ -Metropolis algorithm in the DQMC

In this section, we discuss how the Metropolis algorithm and detailed balance used in the determinant QMC. In the current version of the determinant QMC code, the random variable $x = \{S(i, l)\}$, $1 \leq i \leq N$, $1 \leq l \leq L$, and $S_{i,l} = 1$ or -1 . Define

$$M_\sigma(x) = I + e^K e^{V_\sigma(L)} \dots e^K e^{V_\sigma(1)},$$

where $K, V_\sigma(l)$ are $N \times N$ matrices, $V_\sigma(l)$ are diagonal matrices such that

$$V_\uparrow(l) = \text{diag}(e^{\lambda S(i,l)})$$

and

$$V_\downarrow(l) = \text{diag}(e^{-\lambda S(i,l)}).$$

The object distribution under investigation can be written as

$$\pi(x) = \frac{\det M_\uparrow(x) \det M_\downarrow(x)}{Z},$$

where Z is the partition function defined as

$$Z = \sum_x \det M_\uparrow(x) \det M_\downarrow(x).$$

Now we use the simplest probability transition function $T(x, y)$:

$$S(i, l) = -S(i, l).$$

Define Metropolis ratio as

$$r = \frac{\det M_{\uparrow}(y) \det M_{\downarrow}(y)}{\det M_{\uparrow}(x) \det M_{\downarrow}(x)}.$$

Since $T(x, y)$ is symmetric function (in fact there is no random factor there), then the Metropolis-Hasting algorithm is just the Metropolis algorithm with flipping:

Metropolis algorithm. Given current state $x^{(t)} = \{S(i, l)\}$:

- Try new state y by flipping one site by $S(i, l) = -S(i, l)$, and compute the Metropolis ratio

$$r = \frac{\det M_{\uparrow}(y) \det M_{\downarrow}(y)}{\det M_{\uparrow}(x) \det M_{\downarrow}(x)}.$$

- Draw $U \sim \text{Uniform}[0, 1]$ and update

$$x^{(t+1)} = \begin{cases} y, & \text{if } U \leq \min\{1, r\} \\ x^{(t)}, & \text{otherwise.} \end{cases}$$

If we change to the Barker's heat-bath algorithm, just let $r = \frac{r}{1+r}$, in this case, then the algorithm is the "heat bath" DQMC method.

Now let us rewrite the DQMC method as currently implemented in DQMC code, and call it γ -Metropolis algorithm.

γ -Metropolis algorithm. Given initial state x and initial parameter γ .

- For $l = 1, \dots, L, i = 1, \dots, N$, do Metropolis algorithm on site (i, l)
 1. Flip state variable by $S(i, l) = -S(i, l)$, and compute the Metropolis ratio

$$r = \frac{\det M_{\uparrow}(y) \det M_{\downarrow}(y)}{\det M_{\uparrow}(x) \det M_{\downarrow}(x)}.$$

2. Compute γ -Metropolis ratio

$$r_{\gamma}(x, y) = \begin{cases} \frac{r}{1+\gamma r}, & \text{if } r \leq 1 \\ \frac{r}{\gamma+r}, & \text{otherwise.} \end{cases}$$

3. Draw $U \sim \text{Uniform}[0, 1]$ and update

$$x = \begin{cases} y, & \text{if } U \leq r_{\gamma}(x, y) \\ x, & \text{otherwise.} \end{cases}$$

- Update γ if average acceptance ratio $\eta \not\approx 0.5$,

$$\gamma = \begin{cases} 1 & \text{if } \gamma + (\eta - 0.5) > 1, \\ \gamma + (\eta - 0.5) & \text{if } 0 \leq \gamma + (\eta - 0.5) \leq 1, \\ 0, & \text{if } \gamma + (\eta - 0.5) < 0, \end{cases}$$

where the accepting ratio η is defined by the ratio the accept steps of total steps.

Remark 32 *The motivation of introducing the parameter γ is to try to get the $\frac{1}{2}$ accepting ratio, $\frac{1}{2}$ acceptance ratio is regarded as better than other ratios. $\frac{1}{2}$ accepting ratio means that half of all random variables $S(i, l)$ will be changed, since in the expression of $M_\sigma(x)$, every $S(i, l)$ has equal probability to take $+1$ and -1 , so naturally after NL step Metropolis updating, half of them are expected to be changed.*

We now check whether the actual transition function $P(x, y)$ satisfies the detailed balance. Note that in this case, the Metropolis ratio can also be rewritten as

$$r = \frac{\pi(y)}{\pi(x)},$$

and the actual transition function $P(x, y)$ is equal to $r_\gamma(x, y)$. If $\pi(y) \geq \pi(x)$, then $r > 1$, and

$$P(x, y) = \frac{r}{\gamma + r}, \quad P(y, x) = \frac{r}{1 + \gamma r}.$$

Then $P(x, y)$ satisfies the "detailed balance":

$$\pi(x)P(x, y) = \frac{\pi(x)\pi(y)}{\pi(x) + \gamma\pi(y)} = \pi(y)P(y, x).$$

Similarly, it can be shown that for $\pi(x) \leq \pi(y)$, the "detailed balance" still holds.

Bibliography

- [1] Jun S. Liu, Monte Carlo Strategies in Scientific Computing, Springer-Verlag, 2001.
- [2] Neal Madras, Lectures on Monte Carlo Methods, AMS, 2002.
- [3] Richard Scalettar, Lecture notes for the Monte Carlo method.