Efficient Algorithms for High-dimensional Eigenvalue Problems

by

Zhe Wang

Department of Mathematics
Duke University

Date: ____________________________
Approved:

____

Jianfeng Lu, Advisor

____
Xiuyuan Cheng

____
Jonathon Mattingly

____
Weitao Yang

____

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Mathematics
in the Graduate School of
Duke University

2020

ABSTRACT

Efficient Algorithms for High-dimensional Eigenvalue Problems

by

Zhe Wang

Department of Mathematics
Duke University

Date: ______________________________
Approved:

____

Jianfeng Lu, Advisor

____
Xiuyuan Cheng

____
Jonathon Mattingly

____
Weitao Yang

____

An abstract of a dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Department of Mathematics
in the Graduate School of
Duke University

2020

Abstract

The eigenvalue problem is a traditional mathematical problem and has a wide applications. Although there are many algorithms and theories, it is still challenging to solve the leading eigenvalue problem of extreme high dimension. Full configuration interaction (FCI) problem in quantum chemistry is such a problem. This thesis tries to understand some existing algorithms of FCI problem and propose new efficient algorithms for the high-dimensional eigenvalue problem. In more details, we first establish a general framework of inexact power iteration and establish the convergence theorem of full configuration interaction quantum Monte Carlo (FCIQMC) and fast randomized iteration (FRI). Second, we reformulate the leading eigenvalue problem as an optimization problem, then compare the show the convergence of several coordinate descent methods (CDM) to solve the leading eigenvalue problem. Third, we propose a new efficient algorithm named Coordinate descent FCI (CDFCI) based on coordinate descent methods to solve the FCI problem, which produces some state-of-the-art results. Finally, we conduct various numerical experiments to fully test the algorithms.

Contents

List of Figures

List of Tables

Acknowledgements

First, I would like to express my deep and sincere gratitude to my advisor Professor Jianfeng Lu. As my mentor and advisor, Professor Lu has been guiding me in both research and life since I first stepped on the campus of Duke University in 2015 fall. Professor Lu is an extraordinary and profound researcher. He points the direction of my research and his guidance keeps motivating and inspiring me. Discussion with Professor Lu is always enjoyable and fruitful. Professor Lu is also a considerate and reliable friend. Besides academic support, he supports and understands me in many other aspects as well, for example, job searching. I receive full respect and freedom during my PhD. Without Professor Lu, my PhD time would never be so meaningful and colorful.

Second, I would thank my committee members, Professor Jonathan Mattingly, Professor Weitao Yang and Professor Xiuyuan Cheng. The discussions with them help me overcome many difficulties and give me new research directions.

Third, I want to say thank you to Dr. Yingzhou Li. He can not help me more and he is one of my best friends.

I am also grateful to Prof. George Booth, Prof. Jonathan Weare, Prof. Stephen Wright, Prof. Lexing Ying, Prof. Wotao Yin, Prof. Ali Alavi and Dr. Qiming Sun for useful discussion of research with me and my advisor.

I would also like to thank my friends Honglue Shi, Zehua Chen, Hanqing Liu and Xin Zhang for helping me in quantum physics and quamtum chemistry.

Finally, I would like to thankthe U.S. National Science Foundation and the U.S. Department of Energy. My PhD researches are supported by part by National Science Foundation under award OAC-1450280 and DMS-1454939 and by the US Department of Energy via grant DE-SC0019449.

Chapter 1

Introduction

1.1 Leading eigenvalue problem

This thesis focuses on solving the leading eigenvalue problem (LEVP) of extreme high dimensions. Given a symmetric matrix A, the LEVP is defined as

$$Av_1 = \lambda_1 v_1, \tag{1.1}$$

for $A \in \mathbb{R}^{n \times n}$, $A^\top = A$, λ_1 is the largest eigenvalue of A, and $v_1 \in \mathbb{R}^n$ is the corresponding eigenvector (we assume here and in the sequel that the leading eigenvalue is positive and non-degenerate). In general, assume $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n$ are the eigenvalues of A, $\lambda_1 > 0$, and the corresponding orthonormal eigenvectors are $v_1, v_2, v_3, \cdots, v_n$. In the case that several leading eigenpairs are needed, we can combine the methods for LEVP together with deflation technique to obtain all desired eigenpairs.

General edge eigenvalue problems can be easily transformed to the LEVP problem defined in (1.1). If the eigenvalue of the smallest eigenvalue $\lambda_n < 0$ is needed, we can consider $-A$ instead. If the eigenvalue of largest magnitude is needed, we can apply the algorithms to both A and $-A$ and then select the desired one. Another transformation is $I \pm \delta A$, where $\delta > 0$ is a small positive number. For δ small enough, the largest (smallest) eigenvalue is transformed to the largest positive eigenvalue of $I \pm \delta A$ and the eigenspace is invariant. Moreover, $I \pm \delta A$ is positive definite and can be viewed as a perturbation to identity for δ small enough.

Leading eigenvalue(s) problems appear in a wide range of applications, including principal component analysis (PCA), spectral clustering, dimension reduction, electronic structure calculation, quantum many-body problems, etc. As a result, many methods have been developed to address the leading eigenvalue(s) problems, e.g., power method, Lanczos algorithm [26, 44], randomized SVD [30], (Jacobi-)Davidson algorithm [20, 82], local optimal block preconditioned conjugate gradient (LOBPCG) [43], projected preconditioned conjugate gradient (PPCG) [93], orbital minimization method (OMM) [18, 56], Gauss-Newton algorithm [55], etc. In the literature of numerical linear algebra, the eigenvalue problem is a traditional problem and there are plenty of discussions of the algorithms for leading eigenvalue problems. It seems that we already know everything of this problem.

However, the story of the leading eigenvalue problem is far from over. On the one hand, due to the limit of the computational resource, the dimension of the problems we can handle is usually below one million. Most of the traditional iterative methods apply the matrix A every iteration and require many iterations before convergence, where the iteration number usually depends on the condition number of A or the leading eigengap of A. Some other methods conduct the inverse of the matrix every iterations, which is too expensive for large A. On the other hand, leading eigenvalue problems of extreme high dimension ($n \sim 10^{10}$, 10^{20} or even higher) appear in applications, where the traditional algorithms are powerless. Full configuration interaction (FCI) problem in quantum chemistry and quantum physics is such an example of interest. For such a problem, it is infeasible to store the matrix even the vector in the main memory of the computer. Therefore, it is important to understand and develop efficient algorithms for eigenvalue problems of extreme high dimension.

1.2 Full Configuration Interaction

Solving quantum many-body problems of fermions is a well-known challenging problme in quantum chemistry and quantum physics, while full configuration interaction

(FCI) is one of the most accurate formulation of the quantum many-body problem yet the most difficult one to solve. In this section, we will give a brief overview of the FCI problem and the algorithms to solve the FCI problem.

In quantum mechanics, the state of a quantum system is represented by a vector $|\Phi\rangle$ in a Hilbert space $\mathcal{H}$. The Schrödinger equation describes the evolution of the quantum system:

$$i\partial_t \left|\Phi_t\right\rangle = \widehat{H} \left|\Phi_t\right\rangle, \tag{1.2}$$

where $\widehat{H}$ is a Hermitian operator on $\mathcal{H}$, known as the Hamiltonian operator. For a typical electronic many-body Hamiltonian, it has the form of

$$\widehat{H} = \sum_{i=1}^{n_{\text{elec}}} \frac{\hat{p}_i^2}{2m} + \sum_{i=1}^{n_{\text{elec}}} \sum_{I=1}^{n_{\text{atom}}} \frac{Z_I}{\|\hat{r}_i - R_I\|} + \sum_{i=1}^{n_{\text{elec}}} \sum_{j=i+1}^{n_{\text{elec}}} \frac{1}{\|\hat{r}_i - \hat{r}_j\|} + H_{\text{core}}, \tag{1.3}$$

where n_{elec} is the number of electrons; n_{atom} is the number of atoms; m is the mass of electron; Z_I is the charge of the I-th nucleus; $\hat{p}_i$ is the momentum operator of the i-th electron; $\hat{r}_i$ is the position operator of the i-th electron; R_I is the position of the I-th nucleus; and H_{core} is the energy among nuclei. It includes the kinectic energy of electrons, the Coulomb interaction between electrons and nuclei, the Coulomb interaction between electrons and electrons and the Coulomb interaction between nuclei and nuclei.

Given a complete set of spin-orbitals $\{\chi_p\}$, which forms an orthonormal basis of the one-body Hilbert space, the many-body electronic Hamiltonian operator (1.3), under the second quantization, can be written as

$$\widehat{H} = \sum_{p,q} t_{pq} \hat{a}_p^\dagger \hat{a}_q + \frac{1}{2} \sum_{p,r,q,s} v_{prqs} \hat{a}_p^\dagger \hat{a}_r^\dagger \hat{a}_q \hat{a}_s, \tag{1.4}$$

where $\hat{a}_p^\dagger$ and $\hat{a}_p$ denote the creation and annihilation operator of an electron with spin-orbital index p, t_{pq} and v_{prqs} are one- and two-electron integrals respectively.

The problem we are interested is to solve the ground state and the ground state energy of the quantum system, which can be obtained from solving the time-independent Schrödinger equation

$$\widehat{H} \left|\Phi_0\right\rangle = E_0 \left|\Phi_0\right\rangle, \tag{1.5}$$

where E_0 denotes the ground state energy (the smallest eigenvalue) and $\left|\Phi_0\right\rangle$ denotes the corresponding ground state wavefunction. Without loss of generality, we assume that E_0 is negative and non-degenerate, i.e., the eigenvalues of $\widehat{H}$ are given as $E_0 < E_1 \leq E_2 \leq \cdots$. We see that the problem is actually an edge eigenvalue problem. If we consider $A = -H$ or $A = I - \delta H$, it is exactly the LEVP problem defined in (1.1).

However, the eigenvalue problem (1.5) is usually of infinite dimension. To make the problem solvable, further simplification is needed. In FCI formulation, the complete spin-orbital set is truncated to a finite subset $\{\chi_p\}_{p=1}^{n_{\text{orb}}}$, obtained e.g., by Hartree-Fock or Kohn-Sham calculations. The FCI variational space $\mathcal{V}$ is spanned by all possible Slater determinants constructed from $\{\chi_p\}_{p=1}^{n_{\text{orb}}}$, and the dimension of $\mathcal{V}$ is denoted as N_{FCI} also known as the total number of configuration interactions. The ground state wavefunction is then discretized in $\mathcal{V}$, i.e., $\left|\Phi_0\right\rangle \in \mathcal{V} = \operatorname{span}\big\{\left|D_1\right\rangle, \ldots, \left|D_{N_{\text{FCI}}}\right\rangle\big\}$, where $\left|D_1\right\rangle$ denotes the reference determinant, $\left|D_i\right\rangle$ for $1 < i \leq N_{\text{FCI}}$ denotes other Slater determinants constructed from the finite set of spin-orbitals. Correspondingly, the Hamiltonian operator is represented by a many-body Hamiltonian matrix H with its (i,j) entry as $H_{i,j} = \left\langle D_i\right| \widehat{H} \left|D_j\right\rangle$. Let x and z denote coefficient vectors with entry x_i and z_i respectively. The ground state wavefunction can be written as $\left|\Phi_0\right\rangle = \sum_i x_i \left|D_i\right\rangle$. The time-independent Schrödinger equation (1.5) has its matrix representation as,

$$Hx = E_0 x, \tag{1.6}$$

which is known as the FCI eigenvalue problem. The second-quantized Hamiltonian operator as in (1.4) implies that $H_{i,j}$ is nonzero if and only if $\left|D_i\right\rangle$ can be obtained from $|D_j\rangle$ via changing at most two occupied spin-orbitals. Hence, we say that $|D_i\rangle$ is H-connected with $|D_j\rangle$ if $H_{i,j}$ is nonzero. The set of all indices i such that $|D_i\rangle$ is H-connected with $|D_j\rangle$ is called the H-connected index set of j and is denoted as $\mathcal{I}_H(j)$. Since the cardinality of $\mathcal{I}_H(j)$ is much smaller than N_{FCI}, the matrix H is extremely sparse.

The FCI eigenvalue problem is difficult in two aspects: the infamous fermion sign problem and the exponential scaling of the problem size. By the construction of the many-body Hilbert space $\mathcal{H}$, we see that the dimension n is in the scaling of $\binom{n_{\text{orb}}}{n_{\text{elec}}}$. Therefore, even for a small quantum system, the size n can be hugh. For example, for a water molecule H_2O which consists of only 10 electrons, the dimension n is about 4.5×10^8 if 48 spin-orbitals are used. For a slightly larger system, the dimension will be beyond the imagination. So the dimension of FCI eigenvalue problem is extreme high.

However, the FCI problem also has a good property. The magnitudes of coordinates of the ground state eigenvector span widely. Many coordinates are relatively negligible. Hence, although traditional methods, which treat all coordinates evenly, are unfavorable for FCI problems, it is still possible to design novel algorithms adapted to the properties of FCI which can solve the extreme high eigenvalue problem.

In the literature of quantum chemistry and quantum physics, there are many algorithms to solve FCI approximately besides the direct diagonalization of the FCI Hamiltonian matrix [42]. The algorithms can be roughly organized into three groups. The first group proposes an ansatz of the wavefunction vector, which greatly reduces the degree of freedom while captures the structure, just as the neural network ansatz in deep learning or the linear function ansatz in linear regression. Density matrix renormalization group (DMRG) [96, 14, 63] is a representative of this group. DMRG adopts tensor train ansatz in representing the variational wavefunction. DMRG has been routinely applied to study the ground and excited state of strongly correlated π-conjugated molecules and one-dimensional systems [63]. The second group, like full configuration interaction quantum Monte Carlo (FCIQMC) [8, 6, 57], assumes that the ground state variational wavefunction can be represented as the empirical distribution of a large number of stochastic walkers. To reduce the variance of the energy estimator and the required number of walkers, initiator-FCIQMC (iFCIQMC) [16] and semi-stochastic FCIQMC (S-FCIQMC) [70] are developed aiming at a good trade-off between variance and bias. The third group first solves a selected configuration interaction (SCI) problem and then conducts a perturbation calculation. This family of algorithms (SCI+PT), includes the early work on configuration interaction by perturbatively selecting iteration (CIPSI) [36], and more recently, adaptive configuration interaction (ACI) [75], adaptive sampling configuration interaction (ASCI) [91], etc. Heat-bath configuration interaction (HCI) [34] significantly reduces the computational cost of the selected CI phase based on the information from magnitudes of the double excitations. With perturbation, HCI is able to calculate the ground state energy of a strongly correlated all electron chromium dimer up to 1 mHa accuracy in Ahlrichs VDZ basis. More recently, semi-stochastic HCI (SHCI) [79] further accelerates the perturbation phase with a stochastic idea similar to FCIQMC.

1.3 Contributions

Per the previous discussion, we know that there is not enough understanding of the eigenvalue problem of extreme high dimension. In the community of applied mathematics, there are plenty of discussions of algorithms for eigenvalue problems. However, they hardly work for high-dimensional problems. In the community of quantum chemistry and quantum physics, there are many algorithms to approximately solve the FCI problem. However, they are mostly based on the intuition of the background of physics and chemistry, thus lack of understanding in mathematics. We do not even know whether the algorithms converge to the true answer or not. Also, most algorithms rely on the physical property of the problem too much, which make it hard to extend to high-dimensional eigenvalue problems in other communities. There is a huge gap between the community of applied mathematics and the community of quantum chemistry and quantum physics. This is where my PhD research contributes. We try to understand the algorithms proposed by quantum chemists mathematically and try to propose new efficient algorithms for high dimensional eigenvalue problems, especially for FCI eigenvalue problem. In more details, we contribute in the following three aspects.

First, we establish a general framework to understand the recently proposed randomized algorithms for FCI problem. In recent years, following the work of full configuration interaction quantum Monte Carlo (FCIQMC) [8, 16], the idea of using randomized or truncated power method to solve the FCI eigenvalue problem has become quite popular in quantum chemistry literature. However, the mathematical understanding of the randomized algorithms is missing. As we shall see, from the angle of numerical linear algebra, these recent methods can be understood as generalizations of conventional power method when inexact matrix-vector product is used. As a result, the convergence of these methods can be dealt with by a simple extension of the usual proof of convergence of power method. A natural consequence of this understanding is that, to compare the various approaches, the crucial part is to understand the error caused by different strategies of inexact matrix-vector multiplication. Using this insights, we will compare a few of the recently proposed randomized or truncated FCI methods analytically and also numerically.

Second, we dive deeply into the coordinate descent methods (CDM) which could be used to solve the LEVP problem. The LEVP naturally can be rewritten as the following unconstrained optimization problem,

$$\min_{x \in \mathbb{R}^n} \left\| A - xx^\top \right\|_F^2, \tag{1.7}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Based on the eigendecomposition of A, it is easy to verify that $x = \pm\sqrt{\lambda_1} v_1$ are minimizers of (1.7), Therefore, if the optimization problem (1.7) can be solved, the leading eigenpair (λ_1, v_1) can be reconstructed from the minimizer x^*, i.e., $\lambda_1 = \|x^*\|^2$ and $v_1 = \frac{x^*}{\|x^*\|}$. Such an unconstrained optimization problem (1.7) has appeared before for solving eigenvalue problems [48, 55].

To solve the optimization problem (1.7), coordinate descent methods are a class of strong competitors. Recently, as the rise of big data and deep learning, stochastic iterative methods are revived and revisited to reduce the computational cost per iteration, allow aggressive choice of stepsize, and fully adopt the modern computer architecture. Stochastic gradient descent methods and coordinate descent methods (CDMs) are two popular groups of methods among them. Considering CDMs, when single coordinate is updated every iteration, they only access one column of A each iteration and hence cost $O(n)$ operations per-iteration for n being the size of the problem. Meanwhile, the stepsize could be often n times bigger than that in the traditional gradient descent method for convex objective function [61, 71, 97]. Massive parallelization capability is also observed [52, 53, 66, 87, 102], which is achieved mostly via its "asynchronized" property of different coordinates. [1] We develop and analyze methods within the scope of coordinate descent methods addressing the LEVP.

Third, we propose a novel efficient algorithm named Coordinate descent FCI (CD-FCI) for the FCI eigenvalue problem based on the coordinate descent methods[95].

---

[1] Although different coordinates can be updated in an asynchronized way, these stochastic iterative methods still requires synchronization every now and then.

CDFCI belongs to the third group method (SCI+PT) in our classification of FCI algorithms. Our goal is to improve the variational stage of the computation i.e., the selective CI part. Our proposed algorithm, CDFCI, is an adaptive coordinate-wise (i.e., determinant-wise) iterative method. It updates the coefficient of an appreciative determinant each iteration and has the nice feature of visiting determinants with different frequencies proportional to their importance. As not all determinants contribute equally to the ground state wavefunction, CDFCI is able to efficiently capture the important part of the FCI space and obtain a good approximation to the ground state. Figure 1.1 indicates the relation between the updating frequency of determinants and magnitudes of the determinant coefficients of the ground state wavefunction for an all-electron C_2 calculation with cc-pVDZ basis by CDFCI. Coefficients of configuration interaction wavefunction are sorted in a decreasing order based on their magnitudes. Smaller panel shows the results of the largest 200 coefficients. As shown in Figure 1.1, many coefficients are only updated once throughout iterations, which shows the efficiency of the updating strategy in CDFCI. During iterations, CDFCI also compresses those unappreciative determinants through hard thresholding. Our implementation philosophy of CDFCI is to reserve memory resource for storing variational wavefunction as much as possible, hence, the Hamiltonian matrix is evaluated on-the-fly. Eventually, CDFCI is able to capture the binding curve of all-electron N_2 with cc-pVDZ basis up to 6 digits accuracy in one week and compute the ground state energy of all-electron Cr_2 with Ahlrichs VDZ basis to -2086.443565 Ha, which is the state-of-the-art variational result.

1.4 Notations and Organization

Throughout this thesis, we define some common notations, as shown in Table 1.1. These notations will be used without further explanation for the rest of the paper.

Table 1.1: Common notations.

Notation	Explanation
n	the size of the problem
i, j	coordinate index
Ω	a set of coordinate index
k	the size of Ω
ℓ	the iteration index
α, β	real coefficients
λ, λ_1	an eigenvalue and the largest eigenvalue
v, v_1	an eigenvector and the eigenvector associated with λ_1
e_i	indicator vector with one on the i-th entry and zero everywhere else
x, y, z	vectors
$x^{(\ell)}$	the vector of the ℓ-th iteration
x_i	the i-th entry of vector x
x_Ω	a vector with entries indexed by Ω of x
A	the matrix under consideration
I	identity matrix, the size may depend on the context
$A_{i,j}$	the (i, j)-th entry of matrix A
$A_{i,:}, A_{:,j}$	the i-th row and the j-th column of matrix A
$\|\cdot\|_F, \|\cdot\|_2$	Frobenius norm and 2-norm

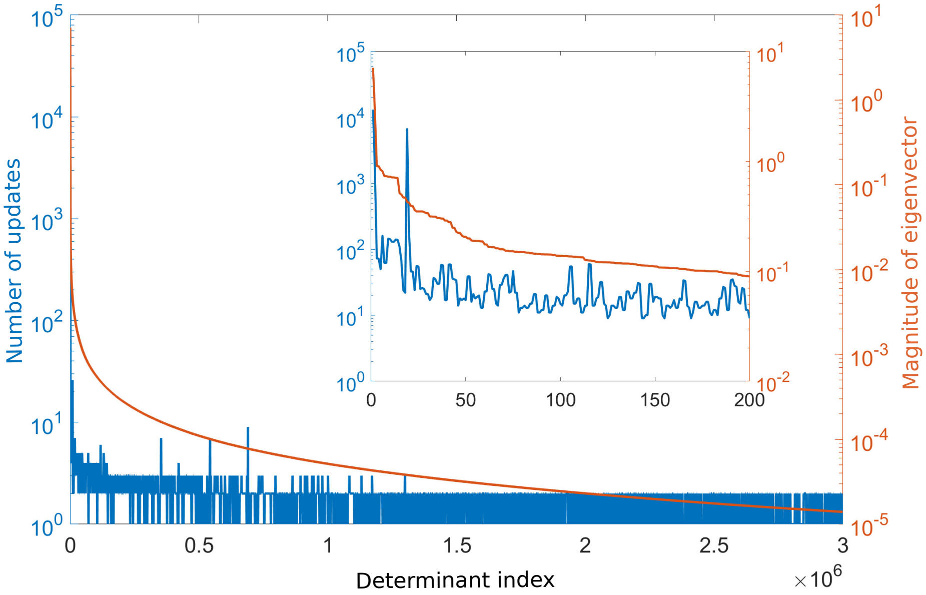


Figure 1.1: Correlation between the updating frequencies and magnitudes of the ground state wavefunction by CDFCI.

Some of the notations in Table 1.1 can be used in a combined way, e.g., $x_i^{(\ell)}$ denotes the i-th entry of the vector x at ℓ-th iteration. The set of coordinate index, Ω, can also be applied to the subscript of a matrix, e.g., $A_{:,\Omega}$ denotes the columns of A with index in Ω. Notice that Frobenius norm and 2-norm are different measure for a matrix, but they are the same measure acting on a vector. Therefore, we will drop the subscript of the norm for vectors, i.e., $\|x\| = \|x\|_2 = \|x\|_F$.

The rest of this thesis is organized as follows. Chapter 2 establishes the framework of inexact power iteration and establish the convergence theorems for several randomized algorithms including full configuration interaction quantum monte carlo (FCIQMC) and fast randomized iteration (FRI). Chapter 3 formulates the LEVP as an optimization problem, and analyzes and compares several coorinate descent methods. Chapter 4 proposed the Coordinate descent FCI for FCI problem. Chapter 5 shows the numerical experiments of the algorithms considered. Chapter 6 is the conclusion.

Chapter 2

Inexact Power Iteration

2.1 Introduction

In this chapter, we propose a general analysis framework for inexact power iteration, which can be used to efficiently solve LEVP. Under the proposed framework, we establish the convergence theorems for several recently proposed randomized algorithms, including the full configuration interaction quantum Monte Carlo (FCIQMC) and the fast randomized iteration (FRI).

To obtain the largest eigenvalue and the corresponding eigenvector, one of the simplest algorithm is standard power iteration, given by

$$z^{(\ell+1)} = Ax^{(\ell)};$$

$$x^{(\ell+1)} = z^{(\ell+1)}/\left\|z^{(\ell+1)}\right\|_2$$

with some initial guess $x^{(0)}$ and iterate till convergence. The algorithm is simple to understand: The matrix multiplication $Ax^{(\ell)}$ amplifies $x^{(\ell)}$ in the leading eigenspace. The convergence of the algorithm is also well known: As long as the initial vector satisfies that $v_1^\top x^{(0)}$ is nonzero and there exists an eigengap ($\lambda_1 > \lambda_2$), the subspace $\operatorname{span} x^{(\ell)}$ converges to the eigenspace $\operatorname{span} v_1$ linearly as $\ell \to \infty$ with rate proportional to λ_2/λ_1.

Since only the convergence of subspace is of interest, the norm of the vector $x^{(\ell)}$ plays no role. Hence the normalization step of power iteration may be omitted

$$x^{(\ell+1)} = Ax^{(\ell)}.$$

This is equivalent with the original power method. Of course, in practical computations, the normalization is important to avoid issues like arithmetic overflow.

Motivated by the recent proposed algorithms in quantum chemistry literature, in this work, we take the view point that we cannot afford (or choose not to perform) the matrix-vector multiplication $z^{(\ell+1)} = Ax^{(\ell)}$ exactly. Among other applications, such a scenario naturally arises when the dimension of the matrix A is extremely large, so that even storage of the vector $z^{(\ell+1)}$ (even in sparse format) is too expensive. For example, this is a common situation for FCI calculations in quantum chemistry since the dimension of the matrix A grows exponentially with respect to the number of electrons in chemical system.

Thus, in power iteration, we would replace the matrix multiplication step by a map

$$z^{(\ell+1)} = F_m(A, x^{(\ell)}). \tag{2.1}$$

Here, given the matrix A and the current iterate $x^{(\ell)}$, the map F_m, either deterministic or stochastic, outputs an approximation of the product $z^{(\ell+1)} \approx Ax^{(\ell)}$. Different choices of F_m corresponds to various recently proposed algorithms, as will be discussed below. We have used the index m to indicate the "complexity" (computational cost) of F_m, the specific meaning depends on the choice of the family of maps. Replacing the matrix-vector multiplication by (2.1), we get Inexact Power Iteration Algorithm 1a and its unnormalized version 1b.

Algorithm 1a: Inexact Power Iteration

```
Initialization: Choose a normalized vector $x^{(0)} \in \mathbb{R}^n$, $\left\|x^{(0)}\right\|_2 = 1$, $v_1^\top x^{(0)} \neq 0$.
for $\ell = 0, 1, 2, \cdots$, while not converged do
    $z^{(\ell+1)} = F_m(A, x^{(\ell)})$;
    $x^{(\ell+1)} = z^{(\ell+1)} / \left\|z^{(\ell+1)}\right\|_2$;
end
```

Algorithm 1b: Inexact Power Iteration without Normalization

```
Initialization: Choose a vector $x^{(0)} \in \mathbb{R}^n$, $v_1^\top x^{(0)} \neq 0$.
for $\ell = 0, 1, 2, \cdots$, while not converged do
    $x^{(\ell+1)} = F_m(A, x^{(\ell)})$;
end
```

Notice that the two versions of inexact power iteration Algorithm 1a and 1b are equivalent if the function $F_m(A, \cdot)$ is homogeneous; we will make this as a standing assumption in our analysis.

Various inexact matrix-vector multiplication has been proposed in the literature for configuration interaction calculations, either deterministic or stochastic, see e.g., earlier attempts in [36, 10, 32, 37, 19, 27, 28, 38, 1], the FCIQMC approach [8, 6, 16, 5, 76], the semi-stochastic approach [70, 4, 33, 79], other stochastic approaches [88, 25, 50], and various deterministic strategies for compressed or truncated representation of the wave functions [72, 73, 58, 22, 41, 104, 91, 54, 74, 106, 105]. In this work, we will focus on two of such strategies: the full configuration-interaction quantum Monte Carlo (FCIQMC) [8, 16] and the fast randomized iteration (FRI) [50]. In some sense, these methods represent two ends of the spectrum of the possibilities; so that the analysis of those can be easily extended to other methodologies. The FCIQMC uses interacting particles to represent the vector $x^{(\ell)}$ and stochastic evolution of particles to represent the action of the matrix A on the vector $x^{(\ell)}$. The FRI on the other hand is based on exact matrix-vector multiplication and stochastic schemes to compress the resulting vectors into sparse ones with given number of nonzero entries. These algorithms will be discussed and analyzed in Section 2.3, following the general analysis framework we establish in Section 2.2.

The rest of the chapter is organized as the following. In Section 2.2, we provide a convergence analysis for a generic class of inexact power iteration. In Section 2.3, we give more details of FCIQMC and FRI and analyze them following the convergence analysis established in Section 2.2.

2.2 General convergence analysis of inexact power iteration

As an advantage of taking a unified framework of various algorithms, the convergence of those can be understood in a fairly generic way, which also facilitates comparison of different proposed strategies. In this section, we establish a general convergence theorem of the inexact power iteration.

The convergence of the iteration to the desired eigenvector will be measured in the angle of the vectors. Recall that the angle between two vectors v and w is given by

$$\theta(v,w) = \arccos\left(\frac{|\langle v,w\rangle|}{\|v\|_2\|w\|_2}\right). \tag{2.2}$$

From the definition, it is obvious that $\theta(v,w) = \theta(av,bw)$ for any vectors v,w and real numbers a,b. In view of this insensitivity of the constant multiple of vectors in the error measure, if the inexact matrix-vector multiplication $F_m(A,x^{(\ell)})$ satisfies the homogeneity assumption below, the two versions of inexact power iterations with or without normalization Algorithm 1a and 1b are equivalent.

Assumption 2.2.1 (Homogeneity)**.**

$$F_m(A,cx) = cF_m(A,x), \tag{2.3}$$

for all vectors $x \in \mathbb{R}^n$ and real number $c \in \mathbb{R}$.

More precisely, if the initial vectors of the two algorithms are the same up to a number $y^{(0)} = c_0 x^{(0)}$, then there exist numbers c_ℓ such that $y^{(\ell)} = c_t x^{(\ell)}$ for all ℓ. Therefore, $\theta(v_1, y^{(\ell)}) = \theta(v_1, x^{(\ell)})$. In the following, when we analyze the algorithm, we will always use $x^{(\ell)}$ for the unnormalized iterate and $y^{(\ell)}$ for the normalized version $y^{(\ell)} = x^{(\ell)}/\|x^{(\ell)}\|_2$.

To analyze the effect of the inexact matrix-vector multiplication, we write $F_m(A, x^{(\ell-1)})$ as a sum of the exact matrix-vector product with an error term

$$x^{(\ell)} = F_m(A, x^{(\ell-1)}) = Ax^{(\ell-1)} + \xi^{(\ell)}, \tag{2.4}$$

where $\xi^{(\ell)}$ is the error of the inexact multiplication at step ℓ, and the dependence on m is suppressed to keep the notation simple. Note that $\xi^{(\ell)}$ can be either deterministic or stochastic depending on the choice of F_m. For example, $\xi^{(\ell)}$ is deterministic for the hard thresholding compression and stochastic for both FCIQMC and FRI methods. While we will proceed viewing $x^{(\ell)}$ as a stochastic process, the results apply to the deterministic case as well.

Denote $\mathcal{F}_\ell = \sigma(x^{(1)}, x^{(2)}, \cdots, x^{(\ell)})$ the σ-algebra generated by $x^{(1)}, x^{(2)}, \cdots, x^{(\ell)}$. We assume that error $\xi^{(\ell)}$ satisfies the following properties. Note that this assumption holds for both FCIQMC and FRI algorithms, as we will prove in Section 2.3.

Assumption 2.2.2. *The error $\xi^{(\ell)}$ in the inexact matrix-vector product* (2.4) *satisfies*

a) Martingale difference sequence condition

$$\mathbb{E}\left(\xi^{(\ell)} \mid \mathcal{F}_{\ell-1}\right) = 0. \tag{2.5}$$

b) Expectation 2-norm bound

$$\mathbb{E}\left(\|\xi^{(\ell)}\|_2^2 \mid \mathcal{F}_{\ell-1}\right) \le C_e \frac{\|A\|_1^2 \|x^{(\ell-1)}\|_1^2}{m}, \tag{2.6}$$

where C_e is a constant that is scale invariant of A (i.e., it does not depend on the norm of A).

c) *Growth of expectation* 1*-norm bound*

$$\mathbb{E}\left(\left\|x^{(\ell)}\right\|_1 \mid \mathcal{F}_{\ell-1}\right) \leq \|A\|_1 \left\|x^{(\ell-1)}\right\|_1. \tag{2.7}$$

A few remarks are in order to help appreciate the Assumption 2.2.2. The martingale difference sequence property is just assumed here for convenience, in fact the convergence result extends to the biased case as we will see in Corollary 2.2.5. The other two assumptions are more essential. Assumption 2.2.2b indicates that the error of the inexact matrix-vector product $F_m(A, x^{(\ell-1)})$ can be controlled by the sparsity of the $x^{(\ell-1)}$, as the 1-norm of $x^{(\ell-1)}$ is a sparsity measure. This is a natural assumption considering that the compression of a vector would be easier if the vector is more sparse. The bound depends proportional to inverse of m, so that one could control the error of the inexact matrix-vector multiplication at the price of increasing the complexity. Note that $1/m$ dependence can be understood as a standard Monte Carlo error scaling. More detailed discussions can be found in Section 2.3 when specific algorithms are analyzed. Assumption 2.2.2c then assumes that the sparsity is not destroyed by the error in the iteration; since otherwise we will lose control of the accuracy of the inexact matrix-vector multiplication.

We now state the convergence theorem for the inexact power iteration Algorithms 1a and 1b. The theorem provides a convergence guarantee with high probability given that the complexity parameter is sufficiently large with the number of iteration steps L chosen properly. Note that the logarithmic dependence of L on the spectral gap λ_1/λ_2 and the error criteria δ and ε is expected from the standard power method. The dependence of m_0, the complexity parameter, on the ratio of the 1-norm and 2-norm of A is due to the competition between the 1- and 2-norm growth of the iterate, where the 1-norm matters for the control of the error of the inexact matrix-vector product.

Theorem 2.2.3. *For the inexact power iteration Algorithm 1b, under Assumption 2.2.2, for any precision* $\varepsilon > 0$ *and small probability* $\delta \in (0,1)$*, there exist time*

$$L = \log(\lambda_1/\lambda_2)^{-1} \log\left(\frac{2\sqrt{2}}{\sqrt{\delta}\varepsilon \cos\theta(v_1, x^{(0)})}\right) \tag{2.8}$$

and measure of complexity

$$m_0 = \frac{4C_e}{\delta\varepsilon^2\big(\cos\theta(v_1, x^{(0)})\big)^2} \frac{\|x^{(0)}\|_1^2}{\|x^{(0)}\|_2^2} L\left(\frac{\|A\|_1}{\|A\|_2}\right)^{2L}, \tag{2.9}$$

such that with probability at least $1 - 2\delta$*, for any* $m \geq m_0$*, it holds*

$$\tan\theta(v_1, x^{(L)}) \leq \varepsilon \tag{2.10}$$

Moreover, if Assumption 2.2.1 is satisfied, the same result holds for Algorithm 1a.

Before proving the theorem, let us collect a few immediate consequences of Assumption 2.2.2. The proof is obvious and will be omitted.

Lemma 2.2.4. *If the error* $\xi^{(\ell)}$ *satisfies Assumption 2.2.2, we have*

a) The error is unbiased

$$\mathbb{E}\,\xi^{(\ell)} = 0. \tag{2.11}$$

b) The error at different step is uncorrelated, in particular

$$\mathbb{E}\,{\xi^{(\ell)}}^\top A^r \xi^{(s)} = 0 \tag{2.12}$$

for any $\ell \neq s$ *and for all non-negative integer* r*.*

c) The expected 2*-norm of the error can be controlled as*

$$\mathbb{E}\left\|\xi^{(\ell)}\right\|_2^2 \le C_e \|A\|_1^{2\ell} \frac{\left\|x^{(0)}\right\|_1^2}{m}. \tag{2.13}$$

Proof of Theorem 2.2.3. From the iteration of Algorithm 1b, we obtain

$$\begin{aligned} x^{(\ell)} &= Ax^{(\ell-1)} + \xi^{(\ell)} \\ &= A^t x^{(0)} + A^{\ell-1}\xi^{(1)} + \cdots + A\xi^{(\ell-1)} + \xi^{(\ell)}. \end{aligned}$$

Since the error $\xi^{(\ell)}$ is unbiased, we have

$$\mathbb{E}\, x^{(\ell)} = A^t x^{(0)}.$$

We now control the variance of $x^{(\ell)}$. Since $\xi^{(\ell)}$ is uncorrelated, we have

$$\mathbb{E}\, {x^{(\ell)}}^\top x^{(\ell)} = {x^{(0)}}^\top A^{2\ell} x^{(0)} + \mathbb{E}\, {\xi^{(1)}}^\top A^{2\ell-2}\xi^{(1)} + \cdots \mathbb{E}\, {\xi^{(\ell-1)}}^\top A^2 \xi^{(\ell-1)} + \mathbb{E}\, {\xi^{(\ell)}}^\top \xi^{(\ell)}.$$

Thus,

$$\mathbb{E}\, {x^{(\ell)}}^\top x^{(\ell)} - \mathbb{E}\, {x^{(\ell)}}^\top \mathbb{E}\, x^{(\ell)} = \mathbb{E}\, {\xi^{(1)}}^\top A^{2\ell-2}\xi^{(1)} + \cdots \mathbb{E}\, {\xi^{(\ell-1)}}^\top A^2 \xi_{\ell-1} + \mathbb{E}\, {\xi^{(\ell)}}^\top \xi^{(t)}.$$

Using Lemma 2.2.4, we estimate

$$\begin{aligned} \left|\mathbb{E}\, {x^{(\ell)}}^\top x^{(\ell)} - \mathbb{E}\, {x^{(\ell)}}^\top \mathbb{E}\, x^{(\ell)}\right| &= \left|\mathbb{E}\, {\xi^{(1)}}^\top A^{2\ell-2}\xi^{(1)} + \cdots + \mathbb{E}\, {\xi^{(t-1)}}^\top A^2 \xi^{(t-1)} + \mathbb{E}\, {\xi^{(\ell)}}^\top \xi^{(\ell)}\right| \\ &\le \mathbb{E}\left|{\xi^{(1)}}^\top A^{2\ell-2}\xi^{(1)}\right| + \cdots + \mathbb{E}\left|{\xi^{(\ell-1)}}^\top A^2 \xi^{(\ell-1)}\right| + \mathbb{E}\left|{\xi^{(\ell)}}^\top \xi^{(\ell)}\right| \\ &\le \lambda_1^{2\ell-2}\mathbb{E}\left|{\xi^{(1)}}^\top \xi^{(1)}\right| + \cdots + \lambda_1^2 \mathbb{E}\left|{\xi^{(\ell-1)}}^\top \xi^{(\ell-1)}\right| + \mathbb{E}\left|{\xi^{(\ell)}}^\top \xi^{(\ell)}\right| \\ &\le C_e \frac{\|A\|_1^2 \left\|x^{(0)}\right\|_1^2}{m}\left(\lambda_1^{2\ell-2} + \cdots + \lambda_1^2 \|A\|_1^{2\ell-4} + \|A\|_1^{2\ell-2}\right) \\ &= C_e \frac{\|A\|_1^2 \left\|x^{(0)}\right\|_1^2}{m} \frac{\|A\|_1^{2\ell} - \|A\|_2^{2\ell}}{\|A\|_1^2 - \|A\|_2^2}, \end{aligned}$$

where in the last step, we used the fact that $\lambda_1 = \|A\|_2$ since λ_1 is the largest eigenvalue. Recall that for symmetric matrix, $\|A\|_1 \geq \|A\|_2$, hence we have

$$\left|\mathbb{E}\, x^{(\ell)^\top} x^{(\ell)} - \mathbb{E}\, x^{(\ell)^\top} \mathbb{E}\, x^{(\ell)}\right| \leq C_e \frac{\left\|x^{(0)}\right\|_1^2}{m}\, \ell \|A\|_1^{2\ell}. \tag{2.14}$$

By analogous arguments for

$$\mathbb{E}\, x^{(\ell)} x^{(\ell)^\top} - \mathbb{E}\, x^{(\ell)} \mathbb{E}\, x^{(\ell)^\top} = \mathbb{E}\, A^{\ell-1} \xi^{(1)} \xi^{(1)^\top} A^{\ell-1} + \cdots + \mathbb{E}\, A \xi^{(\ell-1)} \xi^{(\ell-1)^\top} A + \mathbb{E}\, \xi^{(\ell)} \xi^{(\ell)^\top},$$

we get

$$\left\|\mathbb{E}\, x^{(\ell)} x^{(\ell)^\top} - \mathbb{E}\, x^{(\ell)} \mathbb{E}\, x^{(\ell)^\top}\right\|_2 \leq C_e \frac{\left\|x^{(0)}\right\|_1^2}{m}\, \ell \|A\|_1^{2\ell}. \tag{2.15}$$

Let us now estimate the angle between $x^{(\ell)}$ and v_1 – the eigenvector associated with the largest eigenvalue. By definition,

$$\left(\tan\theta(v_1, x^{(\ell)})\right)^2 = \frac{\left\|x^{(\ell)}\right\|_2^2 - (v_1^\top x^{(\ell)})^2}{(v_1^\top x^{(\ell)})^2}. \tag{2.16}$$

For the denominator, we know the expectation

$$\mathbb{E}\, v_1^\top x^{(\ell)} = v_1^\top A^\ell x^{(0)} = \lambda_1^\ell (v_1^\top x^{(0)}),$$

and the variance

$$\begin{aligned} \delta\left(v_1^\top x^{(\ell)}\right) &= v_1^\top (\mathbb{E}\, x^{(\ell)} x^{(\ell)^\top} - \mathbb{E}\, x^{(\ell)} \mathbb{E}\, x^{(\ell)^\top}) v_1 \\ &\leq \left\|\mathbb{E}\, x^{(\ell)} x^{(\ell)^\top} - \mathbb{E}\, x^{(\ell)} \mathbb{E}\, x^{(\ell)^\top}\right\|_2 \overset{(2.15)}{\leq} C_e \frac{\left\|x^{(0)}\right\|_1^2}{m}\, \ell \|A\|_1^{2\ell}. \end{aligned}$$

The Chebyshev inequality implies that

$$\mathbb{P}\left(\left|v_1^\top x^{(\ell)} - \lambda_1^\ell v_1^\top x^{(0)}\right| \geq \sqrt{\frac{C_e \ell}{m\delta}} \left\|x^{(0)}\right\|_1 \|A\|_1^\ell\right) \leq \delta,$$

and hence, as $\left|\lambda_1^\ell v_1^\top x^{(0)}\right| - \left|v_1^\top x^{(\ell)}\right| \le \left|v_1^\top x^{(\ell)} - \lambda_1^\ell v_1^\top x^{(0)}\right|$,

$$\mathbb{P}\Big(\left|v_1^\top x^{(\ell)}\right| \le \left|\lambda_1^\ell v_1^\top x^{(0)}\right| - \sqrt{\frac{C_e \ell}{m\delta}}\left\|x^{(0)}\right\|_1 \|A\|_1^\ell\Big) \le \delta,$$

or equivalently

$$\mathbb{P}\Big(\left|v_1^\top x^{(\ell)}\right|^2 \le \Big(\left|\lambda_1^\ell v_1^\top x^{(0)}\right| - \sqrt{\frac{C_e \ell}{m\delta}}\left\|x^{(0)}\right\|_1 \|A\|_1^\ell\Big)^2\Big) \le \delta,$$

For the numerator of (2.16), the expectation is

$$\begin{aligned}
\mathbb{E}\left(\left\|x^{(\ell)}\right\|_2^2 - (v_1^\top x^{(\ell)})^2\right) &= \sum_{i=2}^n v_i^\top \mathbb{E}\, x^{(\ell)} {x^{(\ell)}}^\top v_i \\
&= \sum_{i=2}^n v_i^\top (\mathbb{E}\, x^{(\ell)} {x^{(\ell)}}^\top - \mathbb{E}\, x^{(\ell)} \mathbb{E}\, {x^{(\ell)}}^\top) v_i + \sum_{i=2}^n (v_i^\top \mathbb{E}\, x^{(\ell)})^2 \\
&\le \operatorname{tr}\Big(\mathbb{E}\, x^{(\ell)} {x^{(\ell)}}^\top - \mathbb{E}\, x^{(\ell)} \mathbb{E}\, {x^{(\ell)}}^\top\Big) + \lambda_2^{2\ell}\left\|x^{(0)}\right\|_2^2 \\
&= (\mathbb{E}\, {x^{(\ell)}}^\top x^{(\ell)} - \mathbb{E}\, {x^{(\ell)}}^\top \mathbb{E}\, x^{(\ell)}) + \lambda_2^{2\ell}\left\|x^{(0)}\right\|_2^2 \\
&\overset{(2.14)}{\le} C_e \frac{\left\|x^{(0)}\right\|_1^2}{m}\, \ell \|A\|_1^{2\ell} + \lambda_2^{2\ell}\left\|x^{(0)}\right\|_2^2.
\end{aligned}$$

By Markov inequality, for any $\delta \in (0, 1)$

$$\mathbb{P}\left(\left\|x^{(\ell)}\right\|_2^2 - (v_1^\top x^{(\ell)})^2 \ge \frac{1}{\delta}\Big(C_e \frac{\left\|x^{(0)}\right\|_1^2}{m}\, \ell \|A\|_1^{2\ell} + \lambda_2^{2\ell}\left\|x^{(0)}\right\|_2^2\Big)\right) \le \delta.$$

Therefore,

$$\mathbb{P}\left(\big(\tan\theta(v_1, x^{(\ell)})\big)^2 \le \frac{1}{\delta}\frac{C_e \frac{\left\|x^{(0)}\right\|_1^2}{m}\, \ell \|A\|_1^{2\ell} + \lambda_2^{2\ell}\left\|x^{(0)}\right\|_2^2}{\Big(\left|\lambda_1^\ell v_1^\top x^{(0)}\right| - \sqrt{\frac{C_e \ell}{m\delta}}\left\|x^{(0)}\right\|_1 \|A\|_1^\ell\Big)^2}\right) \ge 1 - 2\delta. \tag{2.17}$$

We can explicitly check then with the choices (2.8) and (2.9), for $m \ge m_0$, we have

$$\mathbb{P}\big(\tan\theta(v_1, x^{(L)}) \le \varepsilon\big) \ge 1 - 2\delta,$$

thus the claim of the theorem. □

As mentioned above, it is possible to drop the martingale difference sequence condition in Assumption 2.2.2 and get a similar result. The reason is that the second moment bound (2.6) can be used to control the bias of $\xi^{(t)}$. We state this as the following theorem.

Theorem 2.2.5. *For the inexact power iteration Algorithm 1b, under the Assumptions 2.2.2(b) and 2.2.2(c), for any precision* $\varepsilon > 0$ *and small probability* $\delta \in (0,1)$, *there exist time*

$$L = \log(\lambda_1/\lambda_2)^{-1} \log\left(\frac{4}{\sqrt{\delta}\varepsilon \cos\theta(v_1, x^{(0)})}\right) \tag{2.18}$$

and measure of complexity

$$m_0 = \frac{8C_e}{\delta\varepsilon^2\big(\cos\theta(v_1, x^{(0)})\big)^2} \frac{\big\|x^{(0)}\big\|_1^2}{\big\|x^{(0)}\big\|_2^2} L^2 \left(\frac{\|A\|_1}{\|A\|_2}\right)^{2L}, \tag{2.19}$$

such that with probability at least $1 - 2\delta$, *for any* $m \geq m_0$, *it holds*

$$\tan\theta(v_1, x^{(L)}) \leq \varepsilon \tag{2.20}$$

Moreover, if Assumption 2.2.1 is satisfied, the same result holds for Algorithm 1a.

Proof. Note that

$$\begin{aligned} v_1^\top x^{(\ell)} &= v_1^\top A^\ell x^{(0)} + v_1^\top A^{\ell-1}\xi^{(1)} + \cdots + v_1^\top A\xi^{(\ell-1)} + v_1^\top \xi^{(\ell)} \\ &= \lambda_1^\ell v_1^\top x^{(0)} + \lambda_1^{\ell-1} v_1^\top \xi^{(1)} + \cdots + \lambda_1 v_1^\top \xi^{(\ell-1)} + v_1^\top \xi^{(\ell)}, \end{aligned}$$

so we get

$$\begin{aligned}
\mathbb{E}\,(v_1^\top x^{(\ell)} - \lambda_1^\ell v_1^\top x^{(0)})^2 &= \mathbb{E}\,(\lambda_1^{\ell-1} v_1^\top \xi^{(1)} + \cdots + \lambda_1 v_1^\top \xi^{(\ell-1)} + v_1^\top \xi^{(\ell)})^2 \\
&= \sum_{i,j=1}^{\ell} \lambda_1^{2\ell-i-j} \mathbb{E}\, v_1^\top \xi^{(i)} {\xi^{(j)}}^\top v_1 \\
&\le \sum_{i,j=1}^{\ell} \lambda_1^{2\ell-i-j} \big(\mathbb{E}\, \big\|\xi^{(i)}\big\|_2^2 \mathbb{E}\, \big\|\xi^{(j)}\big\|_2^2\big)^{1/2} \\
&\le C_e \ell^2 \|A\|_1^{2\ell} \frac{\big\|x^{(0)}\big\|_1^2}{m}.
\end{aligned}$$

Moreover,

$$\begin{aligned}
\mathbb{E}\,\big(\big\|x^{(\ell)}\big\|_2^2 - (v_1^\top x^{(\ell)})^2\big)^2 &= \sum_{i=2}^{n} (v_i^\top A^\ell x^{(0)})^2 + 2 \sum_{i=2}^{n} \sum_{b=1}^{\ell} \mathbb{E}\, v_i^\top A^\ell x^{(0)} {\xi^{(b)}}^\top A^{\ell-b} v_i \\
&\quad + \sum_{i=2}^{n} \sum_{a=1}^{\ell} \sum_{b=1}^{\ell} \mathbb{E}\, v_i^\top A^{\ell-a} \xi^{(a)} {\xi^{(b)}}^\top A^{\ell-b} v_i \\
&\le \lambda_2^{2\ell} \big\|x^{(0)}\big\|_2^2 + 2\ell\sqrt{C_e}\lambda_2^\ell \big\|x^{(0)}\big\|_2 \|A\|_1^\ell \frac{\big\|x^{(0)}\big\|_1}{\sqrt{m}} \\
&\quad + \ell^2 C_e \|A\|_1^{2\ell} \frac{\big\|x^{(0)}\big\|_1^2}{m} \\
&\le 2\lambda_2^{2\ell} \big\|x^{(0)}\big\|_2^2 + 2\ell^2 C_e \|A\|_1^{2\ell} \frac{\big\|x^{(0)}\big\|_1^2}{m},
\end{aligned}$$

where the Cauchy-Schwarz inequality is used in the last line. Thus we can again use the Markov inequality to bound both numerator and denominator on the right hand side of (2.16) to obtain the claimed result. □

2.3 Algorithms

In this section, we will review two stochastic power iteration methods recently proposed in the literature: full configuration-interaction quantum Monte Carlo (FCIQMC)

[8] and fast randomized iteration (FRI) [50]. They can be analyzed in the same framework we established in the previous section. In particular, we prove the convergence of the two algorithms using Theorem 2.2.3. We focus on these two methods since in some sense they represent two opposite ends of strategies inexact matrix-vector multiplications. It is possible to combine the ideas and get a zoo of different approaches, which possibly yield better results; and our analysis can be extended to these as well. We will also comment on two variants: iFCIQMC and hard thresholding (HT), closely related to the FCIQMC and FRI approaches.

Without loss of generality, we will assume the matrix A is close to the identity matrix and thus the eigenvalues λ_i are close to 1 (we can always scale and center the original matrix so that this is true).

2.3.1 Full configuration-interaction quantum Monte Carlo

Algorithm Description

FCIQMC is an algorithm originated in quantum chemistry literature to calculate the ground energy of a many-body electron system by a Monte Carlo algorithm for the full configuration-interaction of the many-body Hamiltonian [8].

Let the Hamiltonian be a real symmetric matrix $H \in \mathbb{R}^{n\times n}$ under the Slater determinant basis. To find the ground state (the smallest eigenvector) of H, we write $A = I - \delta H$ for $\delta > 0$ sufficiently small and hence focus on the largest eigenvalue of A; this can be viewed as a first order truncation of the Taylor series of $e^{-\delta H}$. It is also possible to construct other variants of A from H, which we will not go here.

The FCIQMC can be viewed as a stochastic inexact power iteration for finding the largest eigenvector of A, which corresponds more naturally to the unnormalized version of the inexact power iteration (Algorithm 1b). In the algorithm, the vector $x^{(\ell)}$ is not stored as a vector, but represented as a collection of “signed particles”

$\{\alpha^{(\ell,i)}\}_{i=1}^{M_\ell}$, where M_ℓ is the number of signed particles at iteration step ℓ. Each signed particle α has two attributes: location $loc_\alpha \in \{1, 2, \cdots, n\}$ and sign $s_\alpha \in \{1, -1\}$. Denote $e_i \in \mathbb{R}^n$ the standard basis vector with value 1 at its i-th component and 0 at every other component. Then each signed particle α represents a signed unit vector $\alpha = s_\alpha e_{loc_\alpha}$. The vector $x^{(\ell)} \in \mathbb{Z}^n$ is given by the sum of all signed particles at time ℓ:

$$x^{(\ell)} = \sum_{i=1}^{M_\ell} \alpha^{(\ell,i)}. \tag{2.21}$$

With some ambiguity of notation, we refer to both the set of particles and the corresponding vector as $x^{(\ell)}$, connected by (2.21). As we always assume that the particles with opposite signs on the same location are annihilated (see the annihilation step in the algorithm description below), the vector $x^{(\ell)}$ uniquely determines the set of particles.

In FCIQMC, the inexact matrix-vector multiplication $F_m(A, x^{(\ell)})$ consists of three steps of particle evolution: spawning, diagonal death/cloning and annihilation. Write $A = A_d + A_o$ with A_d the diagonal part and A_o the off-diagonal part. The spawning step approximates $A_o x^{(\ell)}$; the diagonal death / cloning step approximates $A_d x^{(\ell)}$, and the annihilation step sums up the results from the previous two steps and approximates the summation $Ax^{(\ell)} = A_o x^{(\ell)} + A_d x^{(\ell)}$. The three steps will be described in more details below.

Spawning. Each signed particle α (we suppress the index of $\alpha^{(\ell,i)}$ to simplify notation) is allowed to spawn a child particle to another location, corresponding to a nonzero component of $A_o\alpha = s_\alpha A_o(:, loc_\alpha)$.[1] The location of spawning is chosen at random, with probability $p_{\text{loc}}(loc \mid loc_\alpha)$, which is chosen in the original FCIQMC algorithm to be uniformly random over all nonzero components of $A_o\alpha$ for some simple

---

[1] MATLAB notation $A(:, loc)$ is used to denote the loc-th column of A.

Hamiltonian H. In general, $p_{\text{loc}}(\cdot \mid loc_\alpha)$ can be more complicated; we refer readers to [8] for more details. In the following of the paper, $p_{\text{loc}}(\cdot \mid loc_\alpha)$ is assumed to be uniform distribution over all nonzero components of $A_o\alpha$, while our analysis can be extended to other choice of $p_{\text{loc}}(\cdot \mid loc_\alpha)$.

Once the location loc is chosen, n' (possibly 0) children particles are spawned with the same sign $s = \operatorname{sgn}(A_o(loc, loc_\alpha)s_\alpha)$ determined by the sign of vector entry $(A_o\alpha)(loc)$ and the particle α. The location loc and number n' are stochastically chosen such that the overall step gives an unbiased estimate of $A_o\alpha$:

$$\mathbb{E}\left(n' s e_{loc} \mid \alpha\right) = A_o\alpha. \tag{2.22}$$

Please refer Algorithm 2a for details.

Algorithm 2a: FCIQMC - Spawning

```
Input  : Set of particles: ${\alpha^{(\ell,i)}}_{i=1,\ldots,M}$, matrix $A$
Output: New set of particles after spawning: ${\alpha^{(\ell+1,j),\text{sp}}}_{j=1,\ldots,M^{\text{sp}}}$
$M^{\text{sp}} = 0$;
for each particle $\alpha \in {\alpha^{(\ell,i)}}_{i=1,\ldots,M}$ do
    Select a spawning location $loc$ with probability $p_{\text{loc}}(loc \mid loc_\alpha)$;
    Determine the expected number of children
        $Q = \frac{|A_o(loc, loc_\alpha)|}{p_{\text{loc}}(loc \mid loc_\alpha)}$;
    Randomly choose the number of children
        $n' = \begin{cases} \lfloor Q \rfloor & \text{w.p. } 1-(Q-\lfloor Q \rfloor) \\ \lfloor Q \rfloor + 1 & \text{w.p. } Q - \lfloor Q \rfloor \end{cases}$
    Assign sign of each children as
      $s = \operatorname{sgn}(A_o(loc, loc_\alpha)s_\alpha) = \operatorname{sgn}\big((A_o\alpha)(loc)\big)$;
    Increase the number of particles $M^{\text{sp}} = M^{\text{sp}} + n'$;
    Add $n'$ particles with location $loc$ and sign $s$ into the spawning set
      ${\alpha^{(\ell+1,j),\text{sp}}}$.
end
```

Diagonal cloning / death. This step represents $A_d x^{(\ell)}$ as a collection of particles in an analogous way to the spawning step. For every signed particle α, we would consider children particles on the location loc_α (i.e., the location of the new particles is chosen to be loc_α) and obtain an unbiased representation

$$\mathbb{E}\left(n' s e_{loc_\alpha} \mid \alpha\right) = A_d \alpha. \tag{2.23}$$

The details can be found in Algorithm 2b, the key steps are similar to Algorithm 2a.

Algorithm 2b: FCIQMC - Diagonal cloning / death

```
Input  : Set of particles: ${\alpha^{(\ell,i)}}_{i=1,\dots,M}$, matrix $A$
Output: New set of particles after diagonal cloning / death:
        ${\alpha^{(\ell+1,j),\,\mathrm{diag}}}_{j=1,\dots,M^{\mathrm{diag}}}$
$M^{\mathrm{diag}} = 0$;
for each particle $\alpha \in \{\alpha^{(\ell,i)}\}_{i=1,\dots,M}$ do
    Determine the expected number of children at $loc_\alpha$
        $Q = |A(loc_\alpha, loc_\alpha)|,$
    Randomly choose the number of children
        $n' = \begin{cases} \lfloor Q \rfloor & \text{w.p.} \quad 1-(Q-\lfloor Q \rfloor) \\ \lfloor Q \rfloor + 1 & \text{w.p.} \quad Q-\lfloor Q \rfloor \end{cases}$
    Assign sign of each children as $s = \mathrm{sgn}\big((A_d\alpha)(loc_\alpha)\big)$;
    Increase the number of particles $M^{\mathrm{diag}} = M^{\mathrm{diag}} + n'$;
    Add $n'$ particles with location $loc_\alpha$ and sign $s$ into the set $\{\alpha^{(\ell+1,j),\mathrm{diag}}\}$.
```

Annihilation. The annihilation step merges the children particles from the previous two steps and remove all pairs of particles with the same location and opposite signs. If we denote $x^{(\ell+1,\mathrm{sp})}$ and $x^{(\ell+1,\mathrm{diag})}$ the corresponding vector representation of the particles are the spawning and diagonal cloning / death steps, the annihilation steps create a collection of particles representing the new vector $x^{(\ell+1)} = x^{(\ell+1,\mathrm{sp})} + x^{(\ell+1,\mathrm{diag})}$. Applying the three steps above to the particles representing $x^{(\ell)}$, we obtain the new set of particles $x^{(\ell+1)}$ at time $\ell+1$. Since by construction

$$\mathbb{E}\left(x^{(\ell+1,\mathrm{sp})} \mid x^{(\ell)}\right) = A_o x^{(\ell)}, \quad \text{and} \quad \mathbb{E}\left(x^{(\ell+1,\mathrm{diag})} \mid x^{(\ell)}\right) = A_d x^{(\ell)},$$

we have on expectation

$$\mathbb{E}\,(x^{(\ell+1)} \mid x^{(\ell)}) = Ax^{(\ell)}. \tag{2.24}$$

In terms of the notations used in the framework of inexact power iteration, $x^{(\ell+1)}$ represented using particles can be viewed as the approximate matrix-vector product $F_m(A, x^{(\ell)})$:

$$F_m(A, x^{(\ell)}) := x^{(\ell+1)} = \sum_{i=1}^{M_{\ell+1}} \alpha^{(\ell+1,i)} = Ax^{(\ell)} + \xi^{(\ell+1)}, \tag{2.25}$$

where $\xi^{(\ell+1)}$ is introduced in the last equality to denote the error from the approximate matrix-vector multiplication through the stochastic particle representation. As we will show in the analysis below, the accuracy of FCIQMC iteration is controlled by the number of particles M_ℓ; and thus it plays the role of the complexity parameter m in our general framework. We would drop the subscript m for F_m in the sequel for FCIQMC, as the complexity parameter is implicit.

Now that we have defined the inexact matrix-vector multiplication $F(A, x^{(\ell)})$ in FCIQMC, we may apply this in the inexact power iteration as Algorithm 1b. However, this can be problematic in practice. Recall that $A = I - \delta H$ is assumed to be a perturbation of identity so its eigenvalue is around 1. If the largest eigenvalue of A is strictly larger than 1, when the signed particles become a good approximation to the leading eigenvector, the number of particles M_ℓ will grow exponentially with rate λ_1, which quickly increases the computational cost and memory requirement. It is also possible (while the probability is tiny) that the number of particles may decrease to 0 due to the randomness.

In practice, it is desirable to have controls on the number of particles to make the algorithm more stable. One such strategy is to introduce a shift $s_\ell \in \mathbb{R}$ and use matrix

$$\widetilde{A} = A + \delta s_\ell I = I - \delta(H - s_\ell I) \tag{2.26}$$

instead of A at the ℓ-th step. Notice that s_ℓ only shifts the eigenvalues while not changing the eigenspace. The shift s_ℓ is adjusted dynamically to control the number of particles. With such shifts, the full FCIQMC algorithm is presented in Algorithm 2.

Algorithm 2: FCIQMC

```
Initialization: $\ell = 0$ and set initial particles $x^{(0)}$.
while $M_\ell \le M^{target}$, the target population do
    // Phase 1:  FCIQMC with fixed shift $s_0$
    Spawning step: Use algorithm 2a with $x^{(\ell)}$ and $A + \delta s_0 I$ to get particle
      set $x^{(\ell+1,\mathrm{sp})}$;
    Diagonal death / cloning step: Use algorithm 2b with $x^{(\ell)}$ and $A + \delta s_0 I$
      to get particle set $x^{(\ell+1,\mathrm{diag})}$;
    Annihilation step to get the particle set of the next time step
      $x^{(\ell+1)} = x^{(\ell+1,\mathrm{sp})} + x^{(\ell+1,\mathrm{diag})}$;
    Update $M_\ell$ and set $\ell = \ell + 1$;
end
Set $s_\ell = s_0$;   // Initialize the dynamic shift
while $\ell < \ell_{\max}$ do
    // Phase 2:  FCIQMC with dynamic shift $s_\ell$
    Spawning step: Use algorithm 2a with $x^{(\ell)}$ and $A + \delta s_\ell I$ to get particle
      set $x^{(\ell+1,\mathrm{sp})}$;
    Diagonal death / cloning step: Use algorithm 2b with $x^{(\ell)}$ and $A + \delta s_\ell I$
      to get particle set $x^{(\ell+1,\mathrm{diag})}$;
    Annihilation step: Merge the two set of particles
      $x^{(\ell+1)} = x^{(\ell+1,\mathrm{sp})} + x^{(\ell+1,\mathrm{diag})}$;
    Update the shift $s_\ell$ as
      $$s_\ell = \begin{cases} s_{\ell-q} - \frac{\eta}{q}\big(\ln M_\ell - \ln M_{\ell-q}\big), & \text{if } \ell = 0 \pmod{q}, \\ s_{\ell-1}, & \text{otherwise}; \end{cases} \qquad (2.27)$$
    Update $M_\ell$ and set $\ell = \ell + 1$;
end
```

The Algorithm 2 contains two phases for different strategies of choosing the shifts and thus controlling the particle population. In Phase 1, the shift is fixed to be s_0, which is chosen such that $|A(i,i) - s_0| \ge 1$ for all i so that the particle number is most likely to grow exponentially till the target population M^{target}. In the second phase, the shift is dynamically adjusted, so to control the growth of the population by a negative feedback loop. The target number of population M^{target} is chosen to be sufficiently large that the variance is small enough to ensure convergence. It plays the role as the 'complexity' m in Theorem 2.2.3. η and q are two parameters to control the fluctuation of number of particles. For the details of the parameter choices, we refer the readers to the original paper on FCIQMC [8] for details.

Energy Estimator. Several estimators can be used to estimate the smallest eigenvalue of H based on the FCIQMC Algorithm 2, which is just a linear transformation of the largest eigenvalue λ_1 of A. One estimator is simply the shift s_ℓ. When the algorithm converges, $x^{(\ell)}$ is approximately proportional to the eigenvector v_1. Since s_ℓ is adjusted to control the number of particles steady, the largest eigenvalue of $A+\delta s_\ell I$ is approximately 1, hence connecting s_ℓ with the desired eigenvalue estimate, cf. (2.26). The other estimator we will consider is the projected energy estimator

$$E_\ell = \frac{v_*^\top H x^{(\ell)}}{v_*^\top x^{(\ell)}}.$$

Here v_* is some fixed vector, for example the Hartree-Fock state of the system. It is clear that when $x^{(\ell)}$ becomes a good approximation of the eigenvector v_1, E_ℓ gives a good estimate of the leading eigenvalue. In the numerical examples, we will focus on the projected energy estimator, since it can be applied to all algorithms we consider in this work (while shift estimator is unique for FCIQMC, in practice, it gives similar results compared to the projected energy estimator).

Convergence Analysis

Since FCIQMC can be viewed as an inexact power iteration as in (2.25), we apply Theorem 2.2.3 to analyze the convergence of FCIQMC. For simplicity, we will focus on the case that the shift is constantly 0, $s_\ell = 0$, since the shift does not affect the eigenvector which is the main focus of Theorem 2.2.3. The probability distribution in the spawning step $p_{\text{loc}}(\cdot \mid loc_\alpha)$ is assumed to be uniform distribution over all the nonzero entries of $A_o(:, loc_\alpha)$. To avoid some degenerate case, we will assume that each diagonal entry of A is non-zero and each column of A has more than 2 nonzero entries (so there is at least one possible location for children particles in the spawning step).

We now check the three conditions in Assumption 2.2.2. The unbiasedness is guaranteed by construction as discussed above for the FCIQMC algorithm, we have

$$\mathbb{E}\left(x^{(\ell+1)} \mid \mathcal{F}_\ell\right) = Ax^{(\ell)}, \tag{2.28}$$

or equivalently, the error $\xi^{(\ell)}$ is a martingale difference sequence:

$$\mathbb{E}\left(\xi^{(\ell+1)} \mid \mathcal{F}_\ell\right) = 0. \tag{2.29}$$

The expectation 2-norm bound is established in the following proposition.

Proposition 2.3.1. *For the inexact matrix-vector multiplication* (2.25) *in FCIQMC Algorithm 2, the error* $\xi^{(\ell)}$ *satisfies*

$$\mathbb{E}\left(\left\|\xi^{(\ell+1)}\right\|_2^2 \mid \mathcal{F}_\ell\right) \le \left(\max_{1\le k\le n}(\|a_k\|_0 - 2)\|a_{o,k}\|_2^2 + \frac{1}{2}\right)\frac{\left\|x^{(\ell)}\right\|_1^2}{M_t}, \tag{2.30}$$

where $a_k = A(:, k)$ *is the* k*-th column vector of* A*, and* $a_{o,k}$ *is the* k*-th column vector of* A_o*, thus* $a_{o,k}$ *equals* a_k *except for the* k*-th entry* $a_{o,k}(k) = 0$*.*

Proof. Since each particle evolves independently,

$$F(A, x^{(\ell)}) = F\Big(A, \sum_{i=1}^{M_\ell} \alpha^{(\ell,i)}\Big) = \sum_{i=1}^{M_\ell} F(A, \alpha^{(\ell,i)}).$$

Moreover $F(A, \alpha_{)}^{(\ell,i)}$ and $F(A, \alpha^{(\ell,j)})$ are independent for $i \neq j$ conditioned on $\mathcal{F}_\ell$.

By construction, $F(A, \alpha^{(\ell,i)})$ is unbiased, *i.e.*,

$$\mathbb{E}\left(F(A, \alpha^{(\ell,i)}) - A\alpha^{(\ell,i)} \mid \mathcal{F}_\ell\right) = 0.$$

Therefore,

$$
\begin{aligned}
\mathbb{E}\left(\left\|\xi^{(\ell+1)}\right\|_2^2 \mid \mathcal{F}_\ell\right) &= \mathbb{E}\left(\left\|\sum_{i=1}^{M_\ell}(F(A,\alpha^{(\ell,i)}) - A\alpha^{(\ell,i)})\right\|_2^2 \mid \mathcal{F}_\ell\right) \\
&= \mathbb{E}\left(\Big(\sum_{i=1}^{M_\ell} F(A,\alpha^{(\ell,i)}) - A\alpha^{(\ell,i)}\Big)^\top \Big(\sum_{j=1}^{M_\ell} F(A,\alpha^{(\ell,j)}) - A\alpha^{(\ell,j)}\Big) \mid \mathcal{F}_\ell\right) \\
&= \sum_{i=1}^{M_\ell} \mathbb{E}\left((F(A,\alpha^{(\ell,i)}) - A\alpha^{(\ell,i)})^\top (F(A,\alpha^{(\ell,i)}) - A\alpha^{(\ell,i)}) \mid \mathcal{F}_\ell\right) \\
&\quad + 2 \sum_{1\le i<j\le M_\ell} \Bigg(\mathbb{E}\left((F(A,\alpha^{(\ell,i)}) - A\alpha^{(\ell,i)})^\top \mid \mathcal{F}_\ell\right) \\
&\quad \times \mathbb{E}\left((F(A,\alpha^{(\ell,j)}) - A\alpha^{(\ell,j)}) \mid \mathcal{F}_\ell\right)\Bigg) \\
&= \sum_{i=1}^{M_\ell} \mathbb{E}\left(\left\|F(A,\alpha^{(\ell,i)}) - A\alpha^{(\ell,i)}\right\|_2^2 \mid \mathcal{F}_\ell\right).
\end{aligned}
$$

Hence, it suffices to consider each particle individually. To simplify the notation, without loss of generality, let us consider a particle with $\alpha^{(\ell,i)} = e_k$ for some k. Since the spawning and diagonal cloning/death steps are independent and unbiased, we have the decomposition

$$
\mathbb{E}\left(\|F(A,e_k) - Ae_k\|_2^2 \mid \mathcal{F}_\ell\right) = \mathbb{E}\left(\|F(A_o,e_k) - A_oe_k\|_2^2 \mid \mathcal{F}_\ell\right) + \mathbb{E}\left(\|F(A_d,e_k) - A_de_k\|_2^2 \mid \mathcal{F}_\ell\right).
$$

For the spawning step, since $A_oe_k = a_{o,k}$, there are $\|a_{o,k}\|_0$ locations to spawn. Remind that $p_{\text{loc}}(\cdot \mid k)$ is assumed to be uniform distribution, so each location is chosen with probability $\frac{1}{\|a_{o,k}\|_0}$. Following the Algorithm 2a, we calculate that

$$
F(A_o,e_k) = \begin{cases}
\lfloor \|a_{o,k}\|_0 \, |a_k(j)| \rfloor \operatorname{sgn}(a_k(j)) e_j, \\
\qquad \text{w.p.} \quad \left(1 - \left(\|a_{o,k}\|_0 \, |a_k(j)| - \lfloor \|a_{o,k}\|_0 \, |a_k(j)| \rfloor\right)\right) / \|a_{o,k}\|_0 \,, \\
\left(\lfloor \|a_{o,k}\|_0 \, |a_k(j)| \rfloor + 1\right) \operatorname{sgn}(a_k(j)) e_j, \\
\qquad \text{w.p.} \quad \left(\|a_{o,k}\|_0 \, |a_k(j)| - \lfloor \|a_{o,k}\|_0 \, |a_k(j)| \rfloor\right) / \|a_{o,k}\|_0 \,,
\end{cases}
$$

for each j such that $a_{o,k}(j) \neq 0$. Straightforward calculation yields

$$\begin{aligned}
\mathbb{E}\, \|F(A_o, e_k) - A_o e_k\|_2^2 &= (\|a_{o,k}\|_0 - 1)\, \|a_{o,k}\|_2^2 \\
&\qquad + \frac{1}{\|a_{o,k}\|_0} \sum_{j,\, a_{o,k}(j)\neq 0} \Big(\big(\|a_{o,k}\|_0\, a_k(j) - \lfloor \|a_{o,k}\|_0\, a_k(j) \rfloor\big) \\
&\qquad \times \Big(1 - \big(\|a_{o,k}\|_0\, a_k(j) - \lfloor \|a_{o,k}\|_0\, a_k(j) \rfloor\big)\Big)\Big) \\
&\leq (\|a_{o,k}\|_0 - 1)\, \|a_{o,k}\|_2^2 + \frac{1}{4}.
\end{aligned}$$

For the diagonal cloning/death step, we have

$$F(A_d, e_k) = \begin{cases} \lfloor |a_k(k)| \rfloor \operatorname{sgn}(a_k(k)) e_k, & \text{w.p.} \quad 1 - \big(|a_k(i)| - \lfloor |a_k(i)| \rfloor\big); \\ (\lfloor |a_k(k)| \rfloor + 1) \operatorname{sgn}(a_k(k)) e_k, & \text{w.p.} \quad |a_k(i)| - \lfloor |a_k(i)| \rfloor. \end{cases}$$

Therefore

$$\mathbb{E}\, (\|F(A_d, e_k) - A_d e_k\|_2^2 \mid \mathcal{F}_t) = \big(|a_k(i)| - \lfloor |a_k(i)| \rfloor\big)\Big(1 - \big(|a_k(i)| - \lfloor |a_k(i)| \rfloor\big)\Big) \leq \frac{1}{4}.$$

Summing up the contribution from the two steps, we arrive at

$$\mathbb{E}\, (\|F(A, e_k) - A e_k\|_2^2 \mid \mathcal{F}_t) \leq (\|a_k\|_0 - 2)\, \|a_{o,k}\|_2^2 + \frac{1}{2},$$

where we used $\|a_{o,k}\|_0 = \|a_k\|_0 - 1$. Thus

$$\mathbb{E}\, (\big\|F(A, x^{(\ell)}) - A x^{(\ell)}\big\|_2^2 \mid \mathcal{F}_\ell) \leq M_\ell \Big(\max_{1\leq k \leq n} (\|a_k\|_0 - 2)\, \|a_{o,k}\|_2^2 + \frac{1}{2} \Big).$$

Since $M_\ell = \big\|x^{(\ell)}\big\|_1$, we can rewrite the above estimate as

$$\mathbb{E}\, (\big\|F(A, x^{(\ell)}) - A x^{(\ell)}\big\|_2^2 \mid \mathcal{F}_\ell) \leq \frac{\|v_\ell\|_1^2}{M_\ell} \Big(\max_{1\leq k \leq n} (\|a_k\|_0 - 2) \|a_{o,k}\|_2^2 + \frac{1}{2} \Big).$$

□

Here we emphasize the important role of the annihilation step in FCIQMC reflected in the error analysis above. Only with the annihilation step is $M_\ell = \left\|x^{(\ell)}\right\|_1$ true, so that the growth of error is controlled as in the last step of the proof. In general, without annihilation, the error will be exponentially larger as $\frac{M_\ell}{\left\|x^{(\ell)}\right\|_1}$ grows exponentially even when $x^{(\ell)}$ is close to the eigenvector v_1. Suppose $x^{(\ell)}$ is approximately v_1. Then $x^{(\ell+1)} \approx \lambda_1 x^{(\ell)}$. Therefore, $\left\|x^{(\ell+1)}\right\|_1 \approx \|A\|_2 \left\|x^{(\ell)}\right\|_1$. However for the number of particles M_ℓ without annihilation, $M_{\ell+1} \approx \||A|\|_2 M_\ell$, where $|A|$ is the entry-wise absolute value of A. To see this, let us denote $x^{(\ell,+)}$ the vector represented by all the particles with positive sign and $-x^{(\ell,-)}$ the vector represented by all the particles with negative sign. Then $x^{(\ell)} = x^{(\ell,+)} - x^{(\ell,-)}$. Denote $\tilde{x}^{(\ell)} = x^{(\ell,+)} + x^{(\ell,-)}$. Then $M_\ell = \left\|\tilde{x}^{(\ell)}\right\|_1$ without annihilation. We can easily check that $\tilde{x}^{(\ell)}$ evolves according to $\tilde{x}^{(\ell+1)} = |A|\tilde{x}^{(\ell)}$. So finally, $\tilde{x}^{(\ell)}$ will converge to the eigenvector of $|A|$, and $M_{\ell+1} \approx \||A|\|_2 M_\ell$. Noticing that $\|A\|_2 \leq \||A|\|_2 \leq \|A\|_1$, we know $\frac{M_\ell}{\left\|x^{(\ell)}\right\|_1}$ grows exponentially at rate $\frac{\||A|\|_2}{\|A\|_2}$ after convergence. Therefore if the number of particles M_ℓ has an upper bound, which is always true in practice due to computational resource constraint, $\left\|x^{(\ell)}\right\|_1$ will decay to zero exponentially, which means the algorithm will not converge to the correct eigenvector. Also comment that if the spawning distribution $p_{\mathrm{loc}}(\cdot \mid loc_\alpha)$ is not exactly uniform distribution, then $\mathbb{E}\left(\|F(A_o, e_k) - A_o e_k\|_2^2 \mid \mathcal{F}_\ell\right)$ will be bound by another constant depending on A_o. Therefore the bound of $\mathbb{E}\left(\left\|\xi^{(\ell+1)}\right\|_2^2 \mid \mathcal{F}_\ell\right)$ in the Proposition will only differ by a constant multiplier.

Compared with Assumption 2.2.2, we observe that the particle number M_ℓ plays the role of the "complexity" parameter. The more particles we have, the smaller the error is. We have the following corollary assuming the particle number is bounded from below by m

Corollary 2.3.2. *If the particle number satisfies $M_\ell \geq m$,*

$$\mathbb{E}\left(\left\|F(A, x^{(\ell)}) - Ax^{(\ell)}\right\|_2^2 \mid \mathcal{F}_\ell, M_\ell \geq m\right) \leq C_e \frac{\|A\|_1^2 \left\|x^{(\ell)}\right\|_1^2}{m}, \tag{2.31}$$

where $C_e = \dfrac{\max_k(\|a_k\|_0 - 2)\|a_{o,k}\|_2^2 + \frac{1}{2}}{\|A\|_1^2}$ *is a parameter scale-invariant of A.*

In summary, FCIQMC satisfies Assumption 2.2.2b, as long as the particle number is not too small. Note that in practice the particle number can be controlled by the dynamic shift s_ℓ to ensure that it does not drop below the required lower bound.

The Assumption 2.2.2c, the growth of expectation 1-norm bound, can also be checked easily, since we have

$$\begin{aligned}
\mathbb{E}\left(\left\|x^{(\ell+1)}\right\|_1 \mid \mathcal{F}_\ell\right) &= \mathbb{E}\left(\left\|F(A, \sum_{i=1}^{M_\ell} \alpha^{(\ell,i)})\right\|_1 \mid \mathcal{F}_\ell\right) = \mathbb{E}\left(\left\|\sum_{i=1}^{M_\ell} F(A, \alpha^{(\ell,i)})\right\|_1 \mid \mathcal{F}_\ell\right) \\
&\leq \sum_{i=1}^{M_\ell} \mathbb{E}\left(\left\|F(A, \alpha^{(\ell,i)})\right\|_1 \mid \mathcal{F}_\ell\right) = \sum_{i=1}^{M_\ell} \left\|A\alpha^{(\ell,i)}\right\|_1 \\
&\leq \sum_{i=1}^{M_\ell} \|A\|_1 \left\|\alpha^{(\ell,i)}\right\|_1 = \|A\|_1 \left\|x^{(\ell)}\right\|_1.
\end{aligned}$$

In conclusion, we have verified the assumptions of Theorem 2.2.3, and thus it can be applied for the convergence and error analysis of FCIQMC.

Remarks on *i*FCIQMC

*i*FCIQMC (initiator FCIQMC) [16] is a modified version of FCIQMC. It can be viewed as a bias-variance tradeoff strategy to reduce the computational cost and error of the FCIQMC approach, by restricting the spawning step.

The n locations are divided into two sets: the initiators L_i and non-initiators L_n with $L_i \cap L_n = \emptyset$, $L_i \cup L_n = \{1, 2, \cdots, n\}$. The rule of *i*FCIQMC is that for any particle α at a non-initiator location $loc_\alpha \in L_n$, it is only allowed to spawn children particles at locations already occupied by some other particles. If α spawns particles to a location unoccupied, then the children particles are discarded. An exception rule is that if at least two particles at non-initiator locations spawn children particles with the same sign at one unoccupied location, then the children particles are kept. There are no restrictions for spawning steps for particles in initiators. In the case that all the locations are initiators $L_n = \emptyset$, iFCIQMC reduces to FCIQMC.

The initiators L_i are chosen at the beginning according to some prior knowledge. The initiators are then updated at each step of iteration. Suppose $n_{i,thre} \in \ltimes$ is a fixed threshold. As soon as the number of particles at a non-initiator location is greater than the threshold $n_{i,thre}$, then the location becomes an initiator. Intuitively, initiators are more important locations for the eigenvector since they are occupied by many particles. The restrictions on the spawning ability of non-initiators reduce the computational cost and the variance of the inexact matrix-vector product while only introducing small bias since there are few particles on non-initiators. Therefore, iFCIQMC can be viewed as a variance control technique for FCIQMC.

2.3.2 Fast Randomized Iteration

In this section, we provide a numerical analysis based on our general framework for the convergence of the fast randomized iteration (FRI), recently proposed in the applied mathematics literature [50], inspired by FCIQMC type algorithms. The basic idea of the FRI method is to first apply the matrix A on the vector of current iterate, and then employ a stochastic compression algorithm to reduce the resulting vector to a sparse representation. The original convergence analysis [50] uses a norm motivated by viewing the vectors as random measures. In comparison, as we have seen in the proof of Theorem 2.2.3, our viewpoint and analysis is closer in spirit to numerical linear algebra, in particular the standard convergence analysis of power method.

Algorithm Description

The fast randomized iteration (FRI) algorithm is based on a choice of random compression function $\Phi_m : \mathbb{R}^n \to \mathbb{R}^n$, which maps a full vector x to a sparse vector $\Phi_m(x)$ with approximately only m nonzero components. The sparsity of $\Phi_m(x)$ reduces the storage cost of the vector and associated computational cost. To combine FRI with the inexact power iteration, define

$$F_m(A, x^{(\ell)}) = \Phi_m(Ax^{(\ell)}) \tag{2.32}$$

in Algorithm 1a and 1b. The error is $\xi^{(\ell+1)} = \Phi_m(Ax^{(\ell)}) - Ax^{(\ell)}$.

Thus the FRI algorithm is completely characterized by the choice of compression function Φ_m, about which we assume the following properties. These are adaptations of the Assumptions 2.2.1 and 2.2.2 in the context of a compression function.

Assumption 2.3.3. *For any vector $x \in \mathbb{R}^n$, the compression function Φ_m satisfies:*

a) Homogeneity: For all $c \in \mathbb{R}$,

$$\Phi_m(cx) = c\Phi_m(x); \tag{2.33}$$

b) Unbiasedness

$$\mathbb{E}\left(\Phi_m(x) \mid x\right) = x; \tag{2.34}$$

c) Variance bound. For some constant C_Φ independent of m and x,

$$\mathbb{E}\left(\|\Phi_m(x) - x\|_2^2 \mid x\right) \le C_\Phi \frac{\|x\|_1^2}{m}; \tag{2.35}$$

d) Expectation 1-norm bound

$$\mathbb{E}\left(\|\Phi_m(x)\|_1 \mid x\right) = \|x\|_1. \tag{2.36}$$

The compression function Φ_m introduced in [50] is as follows. For a given vector $x \in \mathbb{R}^n$, first we sort the entries as $|x(q_1)| \geq |x(q_2)| \geq \cdots \geq |x(q_n)|$, where $q : [n] \to [n]$ is a permutation. The compression function consists of two parts. In the first part, large components of the vector are preserved exactly. Define

$$\tau = \max_{1 \leq i \leq n} \left\{ i : |x(q_i)| \geq \frac{\sum_{j=i}^{n} |x(q_j)|}{m+1-i} \right\},$$

with the convention $\max\{\emptyset\} = 0$, so $0 \leq \tau \leq m$. The compression function keeps the entries $x(q_i)$ for any $1 \leq i \leq \tau$,

$$\big(\Phi_m(x)\big)(q_i) = x(q_i), \qquad \forall i \leq \tau.$$

Note that if $\|x\|_0 \leq m$, all components are 'large' and $\Phi_m(x) = x$, the input vector is kept without compression. The remaining $n - \tau$ components are considered 'small'. Under the compression we only keep a few entries so the resulting vector $\Phi_m(x)$ has about m nonzero entries, as in Algorithm 3; the details are further discussed below.

In the second part of Algorithm 3, the set $B = \{q_{\tau+1}, q_{\tau+2}, \cdots, q_n\}$ consists of the indices of all 'small' components to be compressed. Note that for the integer random variable N_i, $i \in B$, only its expectation $\mathbb{E}\, N_i \in (0, 1)$ is specified, so there is still freedom to choose the probability distribution of $\{N_i\}_{i \in B}$. Here we only discuss independent Bernoulli (which is easy to understand) and systematic sampling (which we use in the numerical examples) approaches, while other choices are possible. Let us focus on the entries in B and define $x' \in \mathbb{R}^n$ such that $x'(i) = x(i)\mathbf{1}_{\{i \in B\}}$. It follows that $\|x'\|_1 = \|x\|_1 - \sum_{i=1}^{\tau} |x(q_i)|$.

For the independent Bernoulli, N_i is independent for each $i \in B$ and follows the Bernoulli distribution as

$$N_i = \begin{cases} 0, & \text{w.p.} \quad 1 - \frac{|x(i)|}{\|x'\|_1/(m-\tau)}; \\ 1, & \text{w.p.} \quad \frac{|x(i)|}{\|v'\|_1/(m-\tau)}. \end{cases}$$

Algorithm 3: FRI - compression function Φ_m

```
Input  : $x \in \mathbb{R}^n$, sparsity parameter $m$
Output: $X = \Phi_m(x) \in \mathbb{R}^n$
// Part 1:  Keep large components
$B = \{1, 2, \cdots, n\}$;
$s = \|x\|_1$;
$i' = \arg\max_{i \in B} |x(i)|$;
$\tau = 0$;
while $|x(i')| \geq \frac{s}{m - \tau}$ do
    $X(i') = x(i')$;
    $s = s - |x(i')|$;
    $\tau = \tau + 1$;
    $B = B \backslash \{i'\}$;
    $i' = \arg\max_{i \in B} |x(i)|$;
end
// Part 2:  Compress small components
for each $i \in B$ do
    Generate nonnegative random integer $\{N_i\}$, such that
        $\mathbb{E}\, N_i = \frac{m - \tau}{s} |x(i)|$;
    $X(i) = \text{sgn}(x(i)) N_i \frac{s}{m - \tau}$;
end
```

Note that the probability is well defined due to the choice of τ. The number of nonzero components of the compressed vector is $\|\Phi_m(x)\|_0 = \tau + \sum_{i \in B} N_i$. From the choice of N_i, $\mathbb{E}\left(\|\Phi_m(x)\|_0 \mid x\right) = m$; so m is the expected sparsity of $\Phi_m(x)$.

Another choice is the systematic sampling [50]: Take a random variable U uniformly distributed in $(0, 1)$. Then for $k = 1, 2, \cdots, m - \tau$, define

$$U_k = \frac{U + k - 1}{m - \tau}.$$

Given $\{q'_1, q'_2, \cdots, q'_{n-\tau}\}$ any permutation of indices in B, define

$$I_k = \max_{1\le i\le n-\tau} \left\{ i : \sum_{j=1}^{i-1} |x(q'_i)| \le U_k \|x'\|_1 < \sum_{j=1}^{i} |x(q'_i)| \right\},$$

then N_i is given by

$$N_i = \begin{cases} 1, & \text{if} \quad i = q'_{I_k} \text{ for some } k, \\ 0, & \text{otherwise.} \end{cases}$$

Notice that by construction, the number of nonzero N_is is exactly $m - \tau$, therefore $\|\Phi_m(x)\|_0 = m$. The N_is generated by systematic sampling is obviously correlated as only one random number U drives the generation. The two approaches will be analyzed in the next section in the framework of inexact power iteration.

Convergence Analysis

We now apply Theorem 2.2.3 to analyze the convergence of the FRI algorithm with either independent Bernoulli or systematic sampling. Notice that we have the immediate result

Lemma 2.3.4. *Assumption 2.3.3 implies Assumptions 2.2.1 and 2.2.2.*

Therefore it suffices to check Assumption 2.3.3 for the compression function Φ_m. Homogeneity is obvious. From the construction of Φ_m, the unbiasedness is guaranteed by the expectation of N_is, no matter which particular distribution is used for N_i.

$$\mathbb{E}\left(\Phi_m(x) \mid x\right) = x. \tag{2.37}$$

The variance bounds are proved in the following lemmas.

Lemma 2.3.5. *For FRI compression with either independent Bernoulli or systematic sampling,*

$$\mathbb{E}\left(\|\Phi_m(x) - x\|_2^2 \mid x\right) \le \frac{\|x'\|_1^2}{m-\tau} \le \frac{\|x\|_1^2}{m}.$$

Moreover, we have the almost sure bound for systematic sampling,

$$\|\Phi_m(x) - x\|_2^2 \leq \frac{2\|x'\|_1^2}{m-\tau} \leq \frac{2\|x\|_1^2}{m}, \quad a.s.$$

It is not possible to obtain an almost sure bound as above for independent Bernoulli, since for example it is possible that all the Bernoulli variables are 1, which gives large error $\|\Phi_m(x) - x\|_2^2 \geq \frac{(N-2m+\tau)\|x'\|_1^2}{(m-\tau)^2}$. This Lemma thus implies the advantage of using the systematic sampling strategy, which in practice gives smaller variance in general. We will only show numerical results using the systematic sampling strategy in the numerical examples later.

Proof. Since large components of x are kept exactly by $\Phi_m(\cdot)$, we have

$$\begin{aligned}\|\Phi_m(x) - x\|_2^2 &= \sum_{i=\tau+1}^{n} (\Phi_m(x)(q_i) - x(q_i))^2 \\ &= \sum_{i=\tau+1}^{n} \big((\Phi_m(x)(q_i))^2 + x(q_i)^2 - 2\Phi_m(x)(q_i)x(q_i)\big).\end{aligned}$$

Take the expectation

$$\begin{aligned}\mathbb{E}\left(\|\Phi_m(x) - x\|_2^2 \mid x\right) =&\mathbb{E}\left(\sum_{i=\tau+1}^{n} (\Phi_m(x)(q_i))^2 \mid x\right) + \sum_{i=\tau+1}^{n} x(q_i)^2 \\ &- 2\sum_{i=\tau+1}^{n} x(q_i)\mathbb{E}\left(\Phi_m(x)(q_i) \mid x\right).\end{aligned}$$

Since both independent Bernoulli and systematic sampling are unbiased,

$$\mathbb{E}\,\Phi_m(x)(q_i) = x(q_i).$$

Moreover, because there are $\sum_{i=\tau+1}^{n} N_i$ number of $\frac{\|x'\|_1}{m-\tau}$ and $n-\tau-\sum_{i=\tau+1}^{n} N_i$ number of 0 in $\{|\Phi_m(x)(q_i)|\}_{i\in B}$, we have

$$\begin{aligned}\mathbb{E}\left(\sum_{i=\tau+1}^{n}(\Phi_m(x)(q_i))^2 \mid x\right) &= \mathbb{E}\left(\mathbb{E}\left(\sum_{i=\tau+1}^{n}(\Phi_m(x)(q_i))^2 \mid \sum_{i=\tau+1}^{n} N_i\right) \mid x\right)\\ &= \frac{\|x'\|_1^2}{(m-\tau)^2}\mathbb{E}\left(\sum_{i=\tau+1}^{n} N_i \mid x\right).\end{aligned}$$

For independent Bernoulli, $\mathbb{E}\left(\sum_{i=\tau+1}^{n} N_i \mid x\right) = m-\tau$ and for systematic sampling, $\sum_{i=\tau+1}^{n} N_i = m-\tau$, so

$$\mathbb{E}\left(\sum_{i=\tau+1}^{n}(\Phi_m(x)(q_i))^2 \mid x\right) = \frac{\|x'\|_1^2}{m-\tau}.$$

Finally,

$$\mathbb{E}\left(\|\Phi_m(x)-x\|_2^2 \mid x\right) = \frac{\|x'\|_1^2}{m-\tau} - \|x'\|_2^2 \le \frac{\|x'\|_1^2}{m-\tau}.$$

We now show that $\frac{\|x'\|_1}{m-\tau} \le \frac{\|x\|_1}{m}$, which follows from the fact that $\frac{\sum_{j=i}^{n}|x(q_j)|}{m+1-i}$ is nonincreasing in i for $i \le \tau$. Indeed, recall from the choice of τ that for $i \le \tau$, $|x(q_i)| \ge \frac{\sum_{j=i}^{n}|x(q_j)|}{m+1-i}$, which is equivalent to

$$\frac{\sum_{j=i}^{n}|x(q_j)|}{m+1-i} \ge \frac{\sum_{j=i+1}^{n}|x(q_j)|}{m-i}.$$

Thus, combined with $\|x'\|_1 \le \|x\|_1$, we arrive at

$$\mathbb{E}\left(\|\Phi_m(x)-x\|_2^2 \mid x\right) \le \frac{\|x'\|_1^2}{m-\tau} \le \frac{\|x\|_1^2}{m}.$$

Next we give the almost sure bound for systematic sampling. Note that if $N_i \ne 0$ for $i \in B$, since $\big(\Phi_m(x)\big)(q_i)$ and $x(q_i)$ have the same sign, we have

$$(\Phi_m(x)(q_i) - x(q_i))^2 \le \Phi_m(q_i)^2 = \frac{\|x'\|_1^2}{(m-\tau)^2}.$$

Since there are exactly $m-\tau$ nonzero N_is, we can estimate

$$\begin{aligned}
\|\Phi_m(x)-x\|_2^2 &= \sum_{i=\tau+1}^{n} \left(\big(\Phi_m(x)\big)(q_i) - x(q_i)\right)^2 \\
&\le \sum_{i=\tau+1}^{n} x(q_i)^2 \mathbf{1}_{N_i=0} + \big(\Phi_m(x)\big)(q_i)^2 \mathbf{1}_{N_i\neq 0} \\
&\le \|x'\|_2^2 + (m-\tau)\frac{\|x'\|_1^2}{(m-\tau)^2} \\
&\le \frac{\|x'\|_1^2}{m-\tau} + \frac{\|x'\|_1^2}{m-\tau} = \frac{2\|x'\|_1^2}{m-\tau} \le \frac{2\|x\|_1^2}{m}.
\end{aligned}$$

□

The expectation 1-norm bound can be easily checked from the definition.

Lemma 2.3.6. *For FRI with independent Bernoulli compression,*

$$\mathbb{E}\left(\|\Phi_m(x)\|_1 \mid x\right) = \|x\|_1.$$

For FRI with systematic sampling compression,

$$\|\Phi_m(x)\|_1 = \|x\|_1, \quad a.s.$$

Therefore, the compression function Φ_m satisfies Assumption 3, and thus the convergence follows Theorem 2.2.3.

Deterministic compression by hard thresholding

Another way to choose the compression function Φ_m is by simple hard thresholding, which means $\Phi_m = \Phi_m^{\mathrm{HT}}$ keeps the m largest entries (in absolute value) and drops the remaining ones. Compared to the previously discussed approaches of compression, the hard thresholding obviously has smaller variance since it is deterministic, as a price to pay, it introduces bias to the inexact matrix-vector multiplication. The bias-variance tradeoff between hard thresholding and FRI type algorithm is similar to that between iFCIQMC and FCIQMC.

Chapter 3

Coordinate Descent Methods

3.1 Introduction

In this section, we consider coordinate descent methods (CDM) for LEVP based on a reformulation of the leading eigenvalue problem as a non-convex optimization problem. The convergence of several coordinate descent methods is analyzed and compared.

As discussed in Section 1.3, the LEVP can be rewritten as the following unconstrained optimization problem,

$$\min_{x \in \mathbb{R}^n} \left\| A - xx^\top \right\|_F^2, \tag{3.1}$$

Throughout this chapter, we denote $f(x) = \left\| A - xx^\top \right\|_F^2$ as the objective function.

This chapter is organized as follows. First we introduce the background of coordinate descent methods in Section 3.1. After the introduction, We analyze the landscape of (3.1) in Section 3.2. Section 3.3, 3.4 and 3.5 follow the same structure, which first introduces or reviews several CDMs, and then conducts the corresponding analysis. Section 3.3 focuses on CDMs with conservative stepsize, whereas Section 3.4 and Section 3.5 focus on greedy and stochastic CDMs respectively. Numerical comparisons between CDMs are discussed in Section 5.2.

The history of coordinate descent methods (CDMs) dates back to the early development of the discipline of optimization [21]. It did not catch much attention until very recently. Readers are referred to the recent surveys [80, 97] and references therein for detailed history of the development and analysis of general CDMs.

Since the survey papers, the area of CDMs is still under fast development, see e.g., [40, 69, 98, 101]. Momentum acceleration can also be combined with the CDMs to further accelerate the convergence [3, 47, 51, 61]. Besides these new development of methodology, new applications adopt CMDs to accelerate computations, including but not limit to the area of imaging processing [31], signal processing [2, 59, 103], wireless communication [99, 100], data science [81, 90, 92], etc. It is worth mentioning that Peng *et al.* [68] discussed coordinate friendly structure, which could be beneficial for more applications.

In terms of designing CDMs for LEVP, Lei *et al.* [48] proposes coordinate-wise power method (CPM) addressing the LEVP. CPM accelerates the traditional power method by the coordinate-wise updating technique. In the same paper, a symmetric greedy coordinate descent (SGCD) method is proposed, which will be reviewed in detail in the following. Wang *et al.* [94] adopts shift-and-invert power method to solve the LEVP, where the inverse of the shifted linear system is addressed by coordinate-wise descent method. Different coordinate updating rules are employed, i.e., Gauss-Southwell-Lipschitz rule (SI-GSL), cyclic rule (SI-Cyclic), and accelerated randomized coordinate descent methods (SI-ACDM). All these methods are proposed as methods calculating the leading eigenvector and outperforms many other method when an accurate upper bound of the leading eigenvalue is given. It is not clear how to obtain such upper bound efficiently in practice however.

A general coordinate-wise descent method (CDM) for the LEVP can be summarised as Algorithm 4. Given a symmetric matrix A and the initial vector $x^{(0)}$. CDM first picks a coordinate j_ℓ according to a specific rule "coordinate-pick($x^{(\ell)}, z^{(\ell)}$)". Then it updates the j_ℓ-th coordinate of $x^{(\ell)}$ with "coordinate-update($x^{(\ell)}, z^{(\ell)}, j_\ell$)" and reaches the vector $x^{(\ell+1)}$ for the next iteration. One key to obtain a good choice of j_ℓ and the update is to get $z^{(\ell)} = Ax^{(\ell)}$ involved in the calculation. Since $x^{(\ell+1)}$

```
Algorithm 4: General coordinate-wise descent method for LEVP
Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$; initial vector $x^{(0)}$.
$z^{(0)} = Ax^{(0)}$
$\ell = 0$
while (not converged) do
    $j_\ell = \text{coordinate-pick}(x^{(\ell)}, z^{(\ell)})$
    $x_j^{(\ell+1)} = \begin{cases} \text{coordinate-update}(x^{(\ell)}, z^{(\ell)}, j_\ell), & j = j_\ell \\ x_j^{(\ell)}, & j \neq j_\ell \end{cases}$
    $z^{(\ell+1)} = z^{(\ell)} + A_{:,j_\ell}\left(x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right)$
    $\ell = \ell + 1$
end while
```

and $x^{(\ell)}$ only differ by a single coordinate, we have an efficient updating expression for $z^{(\ell)}$,

$$z^{(\ell+1)} = Ax^{(\ell+1)} = A\left(x^{(\ell)} + \left(x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right)e_{j_\ell}\right) = z^{(\ell)} + A_{:,j_\ell}\left(x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right).$$

Therefore, updating $z^{(\ell+1)}$ from $z^{(\ell)}$ costs $O(n)$ or less operations. Generally, most coordinate-wise descent methods cost $O(n)$ or less operations per iteration, which is much smaller than traditional iterative method with $O(n^2)$ operations per iteration. Such a gap of the computational cost per iteration enables CDM focusing on the update of more important coordinates throughout iterations. The increase of the number of iterations is also leveraged by the choice of stepsize in the updating. The upper bound for the stepsize with guaranteed convergence in the CDM could be much larger than that in the traditional gradient descent method. Therefore, although CDM requires a little more number of iterations to achieve the convergence criteria, the operations counts and the runtime is much smaller than that of many traditional iterative methods [15, 48].

Throughout this paper, several proposed CDMs together with existing methods will be mentioned many times. In order to reduce the difficulty in remembering all these names, we will follow a systematic name convention. The name of a CDM is

Table 3.1: Short names in name convention of CDMs.

	Short name	Explanation
Type of CDM	CD	plain CDM
	GCD	greedy CDM, the coordinate is pick via a greedy way
	SCD	stochastic CDM, the coordinate is sampled from a probability distribution
Coordinate-pick	Cyc	the coordinate is chosen in a cyclic way
	Uni	the coordinate is sampled uniformly
	Grad	the coordinate is chosen according to the magnitude of the gradient
	LS	the coordinate is chosen according to the exact line-search
Coordinate-update	Grad	the coordinate is updated according to the gradient multiplying a stepsize
	LS	the coordinate is updated according to the exact line-search
	vecLS	the exact line-search is applied to a sparse vector direction

composed of three parts, the type of coordinate-wise descent, coordinate-pick strategy, and coordinate-update strategy. Three parts are separated with hyphens. Table 3.1 defines the short names used in each part. Some of the short name works with specific choice of the type of CDM. For example, Uni can only be combined with SCD. We remark that some of the existing methods in literature are renamed under this system, e.g., the coordinate descent method considered in [21, 35] and [46] are called CD-Cyc-LS and CD-Cyc-Grad respectively; SGCD [48] is renamed as GCD-Grad-LS; etc.

For our contributions, we first highlight them as the following bullet points and then discuss in detail.

- Analyze the landscape of $f(x)$ as (3.1) in detail;

- Derive GCD-LS-LS as the most greedy CDM of $f(x)$ and show that most of the saddle points can be avoided under the method;
- Propose SCD-Grad-vecLS(t) and SCD-Grad-LS(t) as a family of stochastic CDMs of $f(x)$ for t being the sampling power, and the local convergence is proved with a convergence rate monotonically increasing as t increases.

In more details, we first analyze the landscape of the objective function $f(x)$. Through our analysis, we show that, although $f(x)$ is non-convex, $x = \pm\sqrt{\lambda_1}v_1$ are the only two local minima of $f(x)$. Since they are of the same function value, we conclude that all local minima of $f(x)$ are global minima.

Then, we investigate a gradient based CDM, CD-Cyc-Grad. It selects coordinate in a cyclic way and the updating follows the gradient vector restricted to that coordinate multiplying by a fixed stepsize. Thanks to the locality of the gradient updating, we show that CD-Cyc-Grad converges to global minima almost surely for the LEVP optimization problem (3.1).

As many other problems solved by CDMs, exact line search along each coordinate direction can be conducted for (3.1). We further derive that maximum coordinate improvement is achievable in $O(n)$ operations, which leads to a CDM called GCD-LS-LS. Through the analysis of saddle points and the greedy strategy in GCD-LS-LS, we find that many saddle points of the non-convex problem in (3.1) are escapable.

Though greedy method, the GCD-LS-LS, is efficient with single coordinate update per-iteration, they often fail in convergence when multiple coordinates are updated per-iteration. Such a problem can be resolved by introducing stochastic coordinate sampling. The SCD-Grad-vecLS(t) and the SCD-Grad-LS(t) sample several coordinates per-iteration with the probabilities proportional to the t-th power of the gradient vector at current iteration. When the power goes to infinity, the stochastic CDMs become the greedy ones. Further, we analyze the local convergence property for the stochastic CDMs for different sampling power t. The theorem can be applied to show the local convergent property of GCD-Grad-LS and GCD-LS-LS as corollaries. One important message of the theorem is that larger sampling power t leads to faster local convergence rate. Therefore the convergence rate of either GCD-LS-LS or SCD-Grad-LS(t) with $t > 0$ is provably faster or equal to that of SCD-Grad-LS(0) which corresponds to the uniform sampling. However, greedy CDMs or stochastic CDMs with larger t are more difficult to escape from strict saddle points when the objective is non-convex. Therefore, through our analysis and numerical experiments, we recommend SCD-Grad-LS(1) for LEVP.

Although all methods are introduced as a solution to $f(x)$ in (3.1), they, especially SCD-Grad-vecLS(t) and SCD-Grad-LS(t), can be widely extended to other problems. Most of the associated analysis could be extended as well.

All proposed and reviewed methods are tested on synthetic matrices and eigenvalue problems from quantum many-body problems. In all examples, CDMs of (3.1) outperform power method, coordinate-wise power method, and full gradient descent with exact line search. When the matrix is shifted by a big positive number, CDMs of (3.1) converges in the similar number of iterations as in the case without shifting. While power method or coordinate-wise power method, which are sensitive to the shifting, converge much slower. For the matrix from quantum many-body problems, where we know a priori that some of the coordinates of the leading eigenvector is more important than others due to the physical property, all CDMs including CDMs of (3.1) and coordinate-wise power method significantly outperform full vector updating methods. This shows great potential of applying the CDMs to quantum many-body problems.

3.2 Landscape Analysis

This section focuses on the analysis of the variational problem (3.1). More specifically, we analyze the landscape of $f(x)$, especially properties of stationary points in this section. The results show some advantages in working with (3.1), and more importantly, provide insights in designing optimization algorithms.

Stationary points and global minimizers. When working with non-convex problems, the understanding of the landscape of the objective function is crucial for iterative methods. If the objective function is bounded from below and the gradient of the objective function is assumed to be Lipschitz continuous, gradient based iterative methods generally are guaranteed to converge to a stationary point such that $\|\nabla f(x)\| < \epsilon$ within $O(1/\epsilon^2)$ iterations [60]. Without rate, Lee *et al.* [46] show that gradient based iterative methods converge to local minima almost surely if all saddle points are strict saddle points. Adding a random perturbation to the gradient based iterative methods enables the analysis of the convergence to local minima with various rates [23, 39]. However, convergence to global minima in most cases is not guaranteed if the landscape of the objective function is complicated.

In this section, we analyze the landscape of (3.1) and show that every local minimum is a global minimum. For simplicity of presentation, we assume that $f(x)$ is a second order differentiable function (which obviously holds when f is given by (3.1)). Denote the gradient vector and Hessian matrix of $f(x)$ as $\nabla f(x)$ and $\nabla^2 f(x)$ respectively. We give the following definitions.

Definition 3.2.1 (Stationary point)**.** *A point y is a stationary point of f if $\nabla f(y) = 0$.*

Definition 3.2.2 (Strict saddle point)**.** *A point y is a strict saddle point of f if y is a stationary point and $\lambda_{min}(\nabla^2 f(y)) < 0$, where $\lambda_{min}(\nabla^2 f(y))$ is the smallest* *eigenvalue of the Hessian matrix.*[1]

Definition 3.2.3 (Local minimizer)**.** *A point y is a local minimizer of f if there exists an ϵ such that $f(y) \leq f(x)$ for any $\|x - y\| \leq \epsilon$. If further $f(y) < f(x)$ when $x \neq y$, then y is called a strict local minimizer.*

Following these definitions, we explicitly write down the form of the stationary points, strict saddle points and local minimizers of the objective function $f(x)$ in (3.1). The assumptions on the matrix A are summarized in Assumption 3.2.4.

Assumption 3.2.4. *The matrix A is symmetric with eigenvalues and eigenvectors $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_n$ and $v_1, v_2, v_3, \ldots, v_n$ respectively. In addition, the largest eigenvalue is positive, $\lambda_1 > 0$.*

Lemma 3.2.5. *Under Assumption 3.2.4, $x = 0$ is a stationary point of $f(x)$. Other stationary points of $f(x)$ are of the form $\sqrt{\lambda} v$ where λ is a positive eigenvalue of A and $v \in \operatorname{null}\big(\lambda I - A\big)$ with $\|v\| = 1$. In particular, $\pm\sqrt{\lambda_1} v_1$ are both stationary points.*

Proof. The stationary point of $f(x)$ satisfies

$$\nabla f(x) = -4Ax + 4\left(x^\top x\right) x = 0. \tag{3.2}$$

Obviously, $x = 0$ is a stationary point. When $x \neq 0$, we have $\big(x^\top x I - A\big) x = 0$. This implies that $x^\top x I - A$ is singular, i.e., $x^\top x = \lambda$ for λ being a positive eigenvalue of A. When $\lambda = \lambda_1$, we have $x \in \operatorname{null}\big(\lambda_1 I - A\big) = \operatorname{span}\big(v_1\big)$ and $x^\top x = \lambda_1$, which implies $x = \pm\sqrt{\lambda_1} v_1$. When $0 < \lambda < \lambda_1$, we have $x \in \operatorname{null}\big(\lambda I - A\big)$ and $x^\top x = \lambda$. Hence, $x = \sqrt{\lambda} v$ for $v \in \operatorname{null}\big(\lambda I - A\big)$ and $\|v\| = 1$. $\square$

Lemma 3.2.6. *Under Assumption 3.2.4, 0 and $\sqrt{\lambda} v$ with $v \in \operatorname{null}\big(\lambda I - A\big)$ and $\|v\| = 1$ are strict saddle points of $f(x)$, where $\lambda < \lambda_1$ and λ is a positive eigenvalue of A.*

---

[1] This definition of the strict saddle point includes local maximizer as well, which does not cause trouble in minimizing $f(x)$ as in (3.1).

Proof. According to Lemma 3.2.5, 0 and $\sqrt{\lambda}v$ are stationary points of $f(x)$. It suffices to validate the second condition in Definition 3.2.2, i.e.,

$$\lambda_{min}\left(\nabla^2 f(x)\right) = \lambda_{min}\left(-4A + 8xx^\top + 4x^\top x I\right) < 0. \tag{3.3}$$

When $x = 0$, (3.3) is obvious since $\lambda_1 > 0$. When $x = \sqrt{\lambda}v$, we apply $\nabla^2 f(x)$ to v_1 and get

$$\nabla^2 f(x) v_1 = 4(\lambda - \lambda_1) v_1, \tag{3.4}$$

where we have used the orthogonality between v and v_1. Therefore $4(\lambda - \lambda_1)$ is a negative eigenvalue of $\nabla^2 f(x)$ and $\sqrt{\lambda}v$ is a strict saddle point. □

From the proof, we observe that for all the strict saddle points of f, the leading eigenvector v_1 of A is always an unstable direction. This will help us in the convergence proof.

The following theorem follows directly from Lemmas 3.2.5 and 3.2.6.

Theorem 3.2.7. *Under Assumption 3.2.4, local minimizers of $f(x)$ are given by $\pm\sqrt{\lambda_1}v_1$ and both local minimizers of $f(x)$ are global minimizers.*

Proof. Lemma 3.2.5 and 3.2.6 imply that the only possible local minimizers are $\sqrt{\lambda_1}v_1$ and $-\sqrt{\lambda_1}v_1$. We easily check that the objective function values are equal at both $\sqrt{\lambda_1}v_1$ and $-\sqrt{\lambda_1}v_1$:

$$f(\sqrt{\lambda_1}v_1) = f(-\sqrt{\lambda_1}v_1) = \left\|A - \lambda_1 v_1 v_1^\top\right\|_F^2. \tag{3.5}$$

Because $f(x) \geq 0$ is bounded from below and $|f(x)| \to \infty$ as $x \to \infty$, global minimizers exist. Therefore, $\pm\sqrt{\lambda_1}v_1$ are both local minimizers as well as global minimizers. □

Theorem 3.2.7 shows that $f(x)$ has no spurious local minima. If an iterative method converges to a local minimum of $f(x)$, it achieves the global minimum. The remaining obstacle of the global convergence is the (strict) saddle points.

3.3 Coordinate descent method with conservative stepsize

In this section, we discuss CD-Cyc-Grad, whose “coordinate-update” is as the entries of the gradient multiplied by a stepsize. The method has guarantee that for almost all initial values, the iterative procedure converges to the global minimum. While, the choice of stepsize is problem-dependent and conservative. According to our numerical experiments, the number of iterations of these coordinate-wise gradient based methods are too large to be competitive.

Algorithm 5: CD-Cyc-Grad for LEVP

```
Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$; initial vector $x^{(0)}$; stepsize $\gamma$.
$z^{(0)} = Ax^{(0)}$
$\ell = 0$
while (not converged) do
    $j_\ell = (\ell \mod n) + 1$
    $x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} - \gamma\left(-4z_j^{(\ell)} + 4\|x^{(\ell)}\|^2 x_j^{(\ell)}\right), & j = j_\ell \\ x_j^{(\ell)}, & j \neq j_\ell \end{cases}$
    $z^{(\ell+1)} = z^{(\ell)} + A_{:,j_\ell}\left(x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right)$
    $\ell = \ell + 1$
end while
```

3.3.1 CD-Cyc-Grad and SCD-Cyc-Grad

The first coordinate-wise descent method we consider addressing (3.1) is CD-Cyc-Grad, following the name convention in Section 3.1.

CD-Cyc-Grad conducts the “coordinate-pick” step in Algorithm 4 in a cyclic way, i.e.,

$$j_\ell = (\ell \mod n) + 1. \tag{3.6}$$

Hence, all coordinates are picked in a fixed order with almost equal number of updatings throughout iterations. The “coordinate-update” adopts coordinate-wise gradient, that is

$$x_{j_\ell}^{(\ell+1)} = x_{j_\ell}^{(\ell)} - \gamma \nabla_{j_\ell} f(x^{(\ell)}) = x_{j_\ell}^{(\ell)} - \gamma \Big(-4A_{j_\ell,:} x^{(\ell)} + 4 \big\| x^{(\ell)} \big\|^2 x_{j_\ell}^{(\ell)} \Big), \tag{3.7}$$

where $\nabla_{j_\ell} f(x^{(\ell)})$ denotes the j_ℓ-th entry of the gradient of f at $x^{(\ell)}$ and γ is the stepsize. CD-Cyc-Grad is detailed as Algorithm 5, with $z^{(\ell)} = Ax^{(\ell)}$ being adopted in (3.7). The advantage of CD-Cyc-Grad over the full gradient descent (GD) method is mainly that the choice of the stepsize in CD-Cyc-Grad could be much larger than that in GD, which could lead to faster convergence [61].

Here we mention another widely used choice of the "coordinate-pick" called random cyclic, which involves randomness. In the beginning of every n iterations, a random permutation $\Pi^{(\ell)}$ of the indices $1, \ldots, n$ is generated. $\Pi^{(\ell)}$ is a vector of size n and the superscript ℓ denotes the iteration number when the random permutation is generated. Once the random permutation is provided, the following n iteration update the coordinate according to the order in $\Pi^{(\ell)}$. Recently, several works [29, 45, 86] discussed the comparison of the convergence rates for CDMs with cyclic and random strategy of "coordinate-pick" for convex problems.

3.3.2 Global convergence of gradient based coordinate-wise descent method

In this section, we show the global convergence of CD-Cyc-Grad in Theorems 3.3.1. In this paper, global convergence means the iterate points converge to the global minimum as the number of iterations goes to infinity. For simplicity of the argument, here we further restrict the assumption of A such that the positive eigenvalues of A are distinct.

The following theorem establish the global convergence (up to measure 0 set of initial conditions) of the Algorithms 5 without convergence rate.

Theorem 3.3.1. *Let* $R \geq \sqrt{\max_j \|A_{:,j}\|}$ *be a constant and* $\gamma \leq \frac{1}{4(n+4)R^2}$ *be the stepsize. Assume the iteration follows CD-Cyc-Grad as in Algorithm 5 and the iteration starts from the domain* $W_0 = \{x : \|x\|_\infty < R\}$. *The iteration converges to global minima for all* $x^{(0)} \in W_0$ *up to a measure zero set.*

The proof of the theorem can be found in Appendix A.1. The idea of the proof follows the recent work [46]. Comparing the choice of stepsize in Theorem 3.3.1 with the CDM stepsize for some other optimization problems, for example [71, 97], the stepsize γ here is about a fraction of $1/n$ smaller. This is due to the fact that the diagonal of the Hessian of $f(x)$, as in (3.3), is unbounded from above. The choice of R and γ ensures that the iteration stays within W_0, in which the Hessian of f remains bounded. It is worth pointing out that, according to our numerical experiments, when $x^{(0)}$ is set to be close to the boundary of W_0, larger γ leads to divergent iteration. When $x^{(0)}$ is set close to the origin, the stepsize γ can be tuned sightly larger for the our testing cases with some randomly generated matrix A. Although the choice of γ is very restrictive and leads to slow convergence, CD-Cyc-Grad has a guarantee of global convergence up to a measure zero set of initial points.

3.4 Greedy coordinate descent method

In this section, we will review a greedy CDM, GCD-Grad-LS, and present another fully greedy CDM, named GCD-LS-LS. Both greedy methods update the coordinate according to exact line search, and thus the stepsize could be much larger than the conservative choice in CD-Cyc-Grad. GCD-Grad-LS selects coordinate according to the magnitude of the gradient vector and then performs the exact line search along that coordinate, while GCD-LS-LS conducts an exact line search along all coordinate directions and move to the minimizer. We show the advantage of the exact line search in GCD-LS-LS, as it can escape certain saddle points of f, which will be discussed in Section 4.2.

3.4.1 GCD-Grad-LS and GCD-LS-LS

We first review GCD-Grad-LS, proposed in [48]. One of the most widely used greedy strategy in the "coordinate-pick" is the Gauss-Southwell rule [83],

$$j_\ell = \arg\max_{j} \left|\nabla_j f\big(x^{(\ell)}\big)\right|, \tag{3.8}$$

which is the "coordinate-pick" strategy in GCD-Grad-LS. Since the Gauss-Southwell rule selects the coordinate according to the magnitude of the gradient vector, we denote such strategy as "Grad" under the name convention.

Once the coordinate is selected, we solve the minimization problem for the exact line search,

$$\alpha_{j_\ell} = \arg\min_{\alpha} f\big(x^{(\ell)} + \alpha e_{j_\ell}\big). \tag{3.9}$$

Since $h(\alpha) = f\big(x^{(\ell)} + \alpha e_{j_\ell}\big)$ is a quartic polynomial in α, solving the minimization problem is equivalent to find the roots of $h'(\alpha) = 0$. Straightforward calculation shows that,

$$h'(\alpha) = 4\big(\alpha^3 + b_{j_\ell}\alpha^2 + c_{j_\ell}\alpha + d_{j_\ell}\big) = 0, \tag{3.10}$$

where the coefficients are defined as

$$b_{j_\ell} = 3x_{j_\ell}^{(\ell)}, \quad c_{j_\ell} = \big\|x^{(\ell)}\big\|^2 + 2\big(x_{j_\ell}^{(\ell)}\big)^2 - A_{j_\ell,j_\ell}, \quad \text{and} \quad d_{j_\ell} = \big\|x^{(\ell)}\big\|^2 x_{j_\ell}^{(\ell)} - z_{j_\ell}^{(\ell)}. \tag{3.11}$$

We notice that the coefficients of the cubic polynomial requires $O(n)$ operations to compute. Once the coefficients are calculated, solving (3.10) can be done in $O(1)$ operations. There could be multiple roots of (3.10), and we can use the following *root picking strategy* to find the one minimizing $h(\alpha)$:

- If there is only one root of (3.10), then it minimizes $h(\alpha)$.

- If there are two roots of (3.10), according to the property of cubic polynomial, one of them must be single root and the other one is of multiplicity two. The single root minimizes $h(\alpha)$.
- If there are three roots in a row, the middle one is a local maximizer of $h(\alpha)$ and the root further away from the middle one minimizes $h(\alpha)$.

Therefore, with this *root picking strategy* to find the minimizer of $h(\alpha)$, the solution of (3.9) can be achieved in $O(n)$ operations.

Algorithm 6: GCD-Grad-LS for LEVP

```
Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$; initial vector $x^{(0)}$.
$z^{(0)} = Ax^{(0)}$
$\ell = 0$
while (not converged) do
    $\nu = \left\|x^{(\ell)}\right\|^2$
    $j_\ell = \arg\max_j \left|\nu x_j^{(\ell)} - z_j^{(\ell)}\right|$
    $b_{j_\ell} = 3x_{j_\ell}^{(\ell)}$
    $c_{j_\ell} = \nu + 2\big(x_{j_\ell}^{(\ell)}\big)^2 - A_{j_\ell,j_\ell}$
    $d_{j_\ell} = \nu x_{j_\ell}^{(\ell)} - z_{j_\ell}^{(\ell)}$
    Solve $\alpha^3 + b_{j_\ell}\alpha^2 + c_{j_\ell}\alpha + d_{j_\ell} = 0$ with the root picking strategy for $\alpha_{j_\ell}$
    $x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} + \alpha_{j_\ell}, & j = j_\ell \\ x_j^{(\ell)}, & j \neq j_\ell \end{cases}$
    $z^{(\ell+1)} = z^{(\ell)} + A_{:,j_\ell}\alpha_{j_\ell}$
    $\ell = \ell + 1$
end while
```

Algorithm 6 describes the steps of GCD-Grad-LS in detail. The local convergence of GCD-Grad-LS can be established: GCD-Grad-LS can be viewed as a special case of SCD-Grad-LS(t) with $t = \infty$ and $k = 1$, which will be discussed later in Section 3.5.2; We will prove the local convergence of SCD-Grad-LS for all $t \geq 0$, where the local convergence of GCD-Grad-LS is automatically implied.

We observe that in (3.10), all the $O(n)$ computational cost comes from the calculation of $\left\|x^{(\ell)}\right\|^2$. Once $\nu = \left\|x^{(\ell)}\right\|^2$ is pre-calculated, all coefficients in (3.10) can be calculated in $O(1)$ operations and hence (3.9) can be solved in $O(1)$ operations. Therefore, applying this to each coordinate, once ν is pre-calculated, solving (3.9) for all coordinates can be done in $O(n)$ operations. This leads us to investigate the possibility of conducting exact line search along all coordinates,

$$\begin{aligned} & j_\ell, \alpha_{j_\ell} = \arg\min_{j,\alpha} f\left(x^{(\ell)} + \alpha e_j\right) \\ \Longleftrightarrow\ & j_\ell = \arg\min_{j} f\big(x^{(\ell)} + \alpha_j e_j\big) \quad \text{with} \quad \alpha_j = \arg\min_{\alpha} f\big(x^{(\ell)} + \alpha e_j\big). \end{aligned} \tag{3.12}$$

Based on the discussion above, α_j for all $j = 1, \ldots, n$ can be obtained in $O(n)$ operations. Now, we will show that evaluating the difference of $f\big(x^{(\ell)} + \alpha_j e_j\big)$ and $f\big(x^{(\ell)}\big)$ for all j can be done in $O(n)$ operations as well. Therefore the minimization problem of j in the second line of (3.12) can be solved efficiently. Through tedious but straightforward calculation, we obtain

$$\Delta f_j := f\big(x^{(\ell)} + \alpha_j e_j\big) - f\big(x^{(\ell)}\big) = \alpha_j^4 + \frac{4b_j}{3}\alpha_j^3 + 2c_j\alpha_j^2 + 4d_j\alpha_j \tag{3.13}$$

where b_j, c_j and d_j are defined analogous to (3.10). Combining (3.10) and (3.13), we conclude that (3.12) is achievable in $O(n)$ operations. The corresponding method is described in Algorithm 7.

Similar to GCD-Grad-LS, the local convergent of GCD-LS-LS can be established; we will defer the local convergence analysis to the end of Section 3.5.2.

3.4.2 Escapable saddle points using exact line search

This section discusses one advantage in working with the exact line search along all coordinate directions, (3.12), of escaping some of the saddle points. The iteration of

Algorithm 7: GCD-LS-LS for LEVP

```
Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$; initial vector $x^{(0)}$.
$z^{(0)} = Ax^{(0)}$
$\ell = 0$
while (not converged) do
    $\nu = \left\| x^{(\ell)} \right\|^2$
    for $j = 1, 2, \dots, n$ do
        $b_j = 3x_j^{(\ell)}$
        $c_j = \nu + 2\big(x_j^{(\ell)}\big)^2 - A_{j,j}$
        $d_j = \nu x_j^{(\ell)} - z_j^{(\ell)}$
        Solve $\alpha^3 + b_j\alpha^2 + c_j\alpha + d_j = 0$ with the root picking strategy for $\alpha_j$
        $\Delta f_j = \alpha_j^4 + \frac{4b_j}{3}\alpha_j^3 + 2c_j\alpha_j^2 + 4d_j\alpha_j$
    end for
    $j_\ell = \arg\min_j \Delta f_j$
    $x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} + \alpha_j, & j = j_\ell \\ x_j^{(\ell)}, & j \neq j_\ell \end{cases}$
    $z^{(\ell+1)} = z^{(\ell)} + A_{:,j_\ell}\alpha_{j_\ell}$
    $\ell = \ell + 1$
end while
```

GCD-LS-LS can be summarized as

$$\begin{aligned} & x^{(\ell+1)} = x^{(\ell)} + \alpha_\ell e_{j_\ell}, \\ & \text{with } \alpha_\ell, j_\ell = \arg\min_{\alpha, j} f\big(x^{(\ell)} + \alpha e_j\big). \end{aligned} \tag{3.14}$$

Theorem 3.4.1. *Assume Assumption 3.2.4 holds and $x^s \neq 0$ is a strict saddle point associated with eigenvalue $\lambda < \max_i A_{i,i}$. There exists a constant δ_0 such that if $\left\| x^{(0)} - x^s \right\| < \delta_0$, then $f(x^{(1)}) < f(x^s)$.*

Since exact line search in all direction guarantees that $f\big(x^{(\ell+1)}\big) < f\big(x^{(\ell)}\big)$, the theorem guarantees that the iteration will never come back to the neighborhood of the saddle point.

Proof. Without loss of generality, we assume that $A_{1,1} = \max_i A_{i,i}$.

Let $\Delta x = x - x^s$ and $\|\Delta x\| = \delta$. We update x by $\beta_1 e_1$. The update of the objective function is given by

$$f\left(x + \beta_1 e_1\right) = f(x^s) + C_0 + \Delta x^\top C_1 + \Delta x^\top C_2 \Delta x + (\Delta x^\top \Delta x)\Delta x^\top C_3 + (\Delta x^\top \Delta x)^2, \tag{3.15}$$

where

$$\begin{aligned}
&C_0 = \beta_1^2 \left(\beta_1^2 + 4x_1^s \beta_1 + 2\lambda + 4\left(x_1^s\right)^2 - 2A_{1,1}\right) =: q(\beta_1),\\
&C_1 = -4\beta_1 A_{:,1} + 4\beta_1 \lambda e_1 + 8\beta_1 x_1^s x^s + 4\beta_1^2 x^s + 8\beta_1 x_1^s e_1 + 4\beta_1^3 e_1,\\
&C_2 = -2A + 4x^s\left(x^s\right)^\top + 4\beta_1 e_1 e_1^\top + 2\lambda I + 4\beta_1 e_1 \left(x^s\right)^\top + 4\beta_1 x^s e_1^\top + 4\beta_1 x_1^s I + 2\beta_1^2 I,\\
&C_3 = 4x^s + 4\beta_1 e_1.
\end{aligned} \tag{3.16}$$

Applying Cauchy-Schwartz inequality to the last four terms in (3.15), we have

$$f(x + \beta_1 e_1) \le f(x^s) + q(\beta_1) + \|C_1\|\delta + \|C_2\|_2 \delta^2 + \|C_3\|\delta^3 + \delta^4 = f(x^s) + p(\delta), \tag{3.17}$$

where $p(\delta) = q(\beta_1) + \|C_1\|\delta + \|C_2\|_2\delta^2 + \|C_3\|\delta^3 + \delta^4$ is a quartic polynomial in δ with constant coefficient (independent of Δx). Observe that for the polynomial in β

$$q(\beta) = \beta^2\left(\beta^2 + 4x_1^s\beta + 2\lambda + 4\left(x_1^s\right)^2 - 2A_{1,1}\right), \tag{3.18}$$

β^2 is always positive, and the discriminant of the remaining 2-nd order polynomial is $\Delta = 8(A_{1,1} - \lambda) > 0$ by assumption of λ. Thus we can choose β_1 such that $q(\beta_1) < 0$, and hence $p(\delta = 0) = q(\beta_1) < 0$.

By continuity, there exists $\delta_0 > 0$ such that $\forall 0 \le \delta < \delta_0$, $p(\delta) < 0$. Hence, following the greedy coordinate-wise iteration as in (3.14) and $x^{(0)} = x$ for $\|x - x^s\| \le \delta < \delta_0$, we obtain,

$$f\left(x^{(1)}\right) = f\left(x^{(0)} + \alpha_\ell e_{j_\ell}\right) \le f\left(x^{(0)} + \beta_1 e_1\right) < f(x^s). \tag{3.19}$$

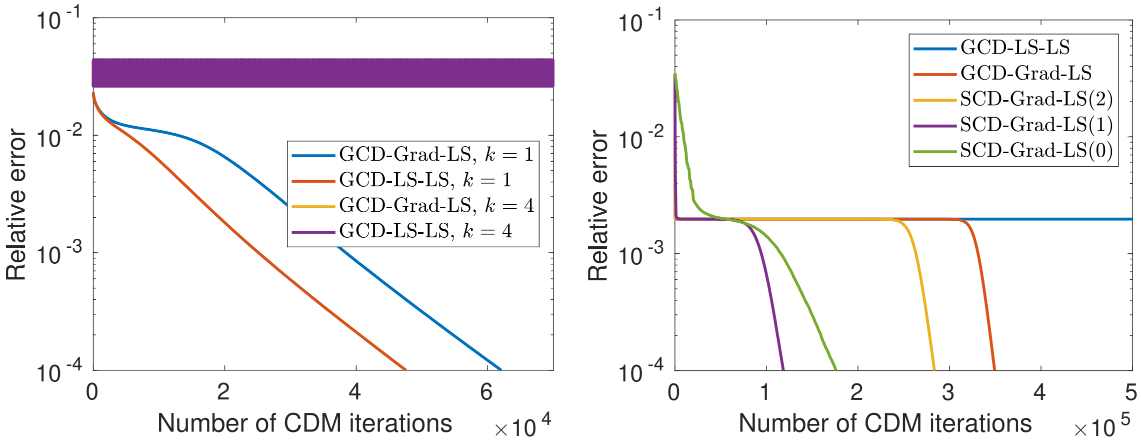
The iteration escapes the strict saddle point x^s in one step. □

To authors' best knowledge, the analysis of global convergence of greedy coordinate-wise descent method is still open for non-convex objective function. Theorem 3.4.1 provides more insights of the behavior of the methods around the saddle points of (3.1). The number of problematic saddle points can be very limited or even zero for a given matrix in practice. For example, combining the result with the Gershgorin circle theorem to locate the eigenvalues can rule out many saddle points. In the literature, there are some analysis of the convergence of coordinate-wise descent method, such as [89]. However, the analysis [89] shows the convergence to coordinate-wise local minima but not the general local minima.

3.5 Stochastic coordinate descent methods

Greedy coordinate-wise descent method, including both GCD-Grad-LS and GCD-LS-LS in Section 3.4, updates a single coordinate every iteration. These methods are beautiful from the theoretical point of view. In practice, these single coordinate methods are not satisfactory on modern computer architecture. Distributed memory super computing cluster, shared big memory machine or even personal laptop have multi-thread parallelism enabled. In order to fully use the computing resources, multi-coordinate updating per iteration is desired for practical usage so that each thread can process a single coordinate simultaneously. The direct extension of the greedy methods to multi-coordinate version simply replaces "coordinate-pick" from the most desired coordinate to the k most desired coordinates; and the "coordinate-update" remains the same for each picked coordinate. However, this change leads to non-convergent iteration. In Figure 3.1, we demonstrate the convergence behavior for solving (3.1) on the matrix A_{108}, which is a random matrix of size 5000 with largest eigenvalue being 108 and other eigenvalue equally distributed in $[1, 100)$. Figure 3.1 (a) demonstrates the non-convergence behavior of the greedy CDMs with $k = 4$.

In this section, we propose another natural extension of the greedy methods, i.e., stochastically sampling multiple coordinates from certain probability distribution and updating each coordinate accordingly. We also provide the analysis of the method for (locally) strongly convex problems. The local convergence analysis of SCD-Uni-LS and GCD-Grad-LS can be viewed as two extreme cases of our analysis. Moreover, the local convergence of GCD-LS-LS also follows as a direct corollary. As will be shown in the analysis part, greedy CDMs are provably outperforms stochastic CDMs when the initial value is in a strongly convex region near the minimizers and single coordinate updating is adopted. If the iteration starts from a non-convex region, as shown in Figure 3.1 (b), stochastic CDMs could converge faster than greedy CDMs. The behavior in the figure will be discussed in more detail at the end of this section.



(a) The non-convergence behavior of greedy CDMs on A_{108} with $k = 4$ comparing with convergent greedy CDMs with $k = 1$. The initial vector is e_1.

(b) The convergence behavior of greedy CDMs and stochastic CDMs on $A_{108} - 100I$. The initial vector has 100 non-zero entries on random coordinates. Note that the only saddle point is the origin.

Figure 3.1: Convergence behavior of greedy coordinate descent methods vs stochastic coordinate descent methods.

3.5.1 SCD-Grad-vecLS and SCD-Grad-LS

The first stochastic coordinate-wise descent method we consider is SCD-Grad-

Algorithm 8: SCD-Grad-vecLS for LEVP

```
Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$; initial vector $x^{(0)}$; probability power $t$;
  number of coordinates $k$.
$z^{(0)} = Ax^{(0)}$
$\ell = 0$
while (not converged) do
    $\nu = \left\|x^{(\ell)}\right\|^2$
    for $j = 1, 2, \dots, n$ do
        $c_j^{(\ell)} = \nu x_j^{(\ell)} - z_j^{(\ell)}$
    end for
    Sample $k$ coordinates with probability proportional to $\left|c^{(\ell)}\right|^t$, denote it
    as $\Omega$
    Solve (3.20) with the root picking strategy for $\alpha_\ell$
    $x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} + \alpha_\ell \nabla_j f\big(x^{(\ell)}\big), & j \in \Omega \\ x_j^{(\ell)}, & j \notin \Omega \end{cases}$
    $z^{(\ell+1)} = z^{(\ell)} + A_{:,\Omega}\big(x_\Omega^{(\ell+1)} - x_\Omega^{(\ell)}\big)$
    $\ell = \ell + 1$
end while
```

vecLS as in Algorithm 8. SCD-Grad-vecLS is a stochastic version of GCD-Grad-LS: Instead of picking the coordinate with largest magnitude in $c_j^{(\ell)}$, we sample k coordinates with probability proportional to $\left|c_j^{(\ell)}\right|^t$ for $t \geq 0$ and the set of sampled indices is denoted as Ω, where as in GCD-Grad-LS in Section 3.4.1, $c^{(\ell)}$ is proportional to the gradient vector, $\nabla f(x^{(\ell)})$. Hence the sampling strategy here is equivalent to sampling with probability proportional to the t-th power of the absolute value of the gradient vector. Through the similar derivation as in Section 3.4, we can show that the exact line search can also be conducted along a given search direction. We denote the line search objective function as $h(\alpha) = f\big(x^{(\ell)} + \alpha v\big) = f\big(x^{(\ell)} + \alpha \sum_{j\in\Omega} \nabla_j f\big(x^{(\ell)}\big) e_j\big)$, where $v = \sum_{j\in\Omega} \nabla_j f\big(x^{(\ell)}\big) e_j$ defines the search direction. Function $h(\alpha)$ is again a quartic polynomial of α. All candidates of optimal α are roots of the cubic polynomial,

$$\frac{\mathrm{d}h}{\mathrm{d}\alpha}(\alpha) = 4\|v_\Omega\|^4\alpha^3 + 12\left(v_\Omega^\top x_\Omega^{(\ell)}\right)\|v_\Omega\|^2\alpha^2$$
$$+ 4\left(\nu\|v_\Omega\|^2 + 2\left(v_\Omega^\top x_\Omega^{(\ell)}\right)^2 - v_\Omega^\top A_{\Omega,\Omega}v_\Omega\right)\alpha + 4\nu\left(v_\Omega^\top x_\Omega^{(\ell)}\right) - 4v_\Omega^\top z_\Omega^{(\ell)} = 0, \tag{3.20}$$

where $\nu = \left\|x^{(\ell)}\right\|^2$. We notice that all coefficients in (3.20), given ν, can be computed in $O(k^2)$ operations for k being the number of indices in Ω. Hence all candidates of the optimal α can be obtained in $O(k^2)$ operations. Analog to Section 3.4, we adopt the *root picking strategy* to find the optimal α which has lowest function value. We conclude that the optimal α is achievable in $O(k^2)$ operations given pre-calculated ν. Algorithm 8 describes the steps in detail.

Algorithm 9: SCD-Grad-LS for LEVP

```
Input: Symmetric matrix $A \in \mathbb{R}^{n\times n}$; initial vector $x^{(0)}$; probability power $t$;
  number of coordinates $k$.
$z^{(0)} = Ax^{(0)}$
$\ell = 0$
while (not converged) do
    $\nu = \left\|x^{(\ell)}\right\|^2$
    for $j = 1, 2, \dots, n$ do
        $c_j^{(\ell)} = \nu x_j^{(\ell)} - z_j^{(\ell)}$
    end for
    Sample $k$ coordinates with probability proportional to $\left|c^{(\ell)}\right|^t$; denote $\Omega$ the
      sampled index set
    for $j \in \Omega$ do
        $p_j = \nu - \left(x_j^{(\ell)}\right)^2 - A_{j,j}$
        $q_j = A_{j,j}x_j^{(\ell)} - z_j^{(\ell)}$
        Solve $\alpha_j^3 + p_j\alpha_j + q_j = 0$ for real $\alpha_j$
    end for
    $x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} + \alpha_j, & j \in \Omega \\ x_j^{(\ell)}, & j \notin \Omega \end{cases}$
    $z^{(\ell+1)} = z^{(\ell)} + A_{:,\Omega}\left(x_\Omega^{(\ell+1)} - x_\Omega^{(\ell)}\right)$
    $\ell = \ell + 1$
end while
```

One drawback of Algorithm 8 is that the exact line search relies on all selected coordinates, which is unsuitable for asynchronous implementation. Here we propose an aggressive SCD method, SCD-Grad-LS, which can be implemented in an asynchronized fashion. The method combines the coordinate picking strategy in SCD-Grad-vecLS and coordinate updating strategy in GCD-Grad-LS. But the updating strategy in SCD-Grad-LS updates each coordinate with the coordinate-wise exact line search independently and can be delayed. Algorithm 9 is the pseudo code of SCD-Grad-LS.

Updating $k > 1$ coordinates independently as in Algorithm 9 does not have guarantee of convergence. When k is large or t is large such that the same coordinate is updated multiple times, we do observe non-convergent behavior of the iteration in practice. However, as shown in the numerical results, when k is relatively small compared to n and $t = 1, 2$, SCD-Grad-LS, on average, requires about a fraction of $1/k$ number of iterations. There exists a fix to guarantee the convergence for any $k > 1$. Instead of updating as $x^{(\ell+1)} = x^{(\ell)} + \sum_{j\in\Omega} \alpha_j e_j$, we update as

$$x^{(\ell+1)} = x^{(\ell)} + \frac{1}{k} \sum_{j\in\Omega} \alpha_j e_j, \tag{3.21}$$

where α_j is the optimal step size in j-th coordinate as in (3.9). Such a change enables convergence but usually increases the iteration number by a factor of k if both cases converge.

In the sampling procedure of stochastic CDMs, there are two ways of sampling, sampling with replacement and sampling without replacement. We claim that when $k \ll n$ and the variance of $\left|c^{(\ell)}\right|^t$ is small, sampling with or without replacement behaves very similarly. While, for $\left|c^{(\ell)}\right|^t$ with large variance or $k \approx n$, these two sampling strategies behave drastically differently. For example, when $t \to \infty$, probability proportional to $\left|c^{(\ell)}\right|^t$ becomes an indicator vector on a single coordinate (assuming non-degeneracy). Sampling with replacement results a set Ω of k same indices, whereas sampling without replacement results a set of k different indices corresponding to the largest k entries in $\left|c^{(\ell)}\right|$. In the analysis below, we prove the local convergence analysis of the Algorithm 8 and Algorithm 9 when $k = 1$ (the two algorithms are equivalent when $k = 1$). Similar but more complicated analysis could be done for $k \geq 1$ when sampling with replacement is adopted and the modified updating strategy (3.21) is adopted in Algorithm 9.

3.5.2 Local convergence of stochastic coordinate descent method

In this section, we analyze the convergence properties of stochastic coordinate-wise descent methods for SCD-Grad-LS. We will present the analysis for a general strongly convex objective function with Lipschitz continuous gradient. The analysis uses the following notations and definitions. We take

$$B^{\pm} = \left\{ y \;\middle|\; \left\| y \mp \sqrt{\lambda_1} v_1 \right\| \leq \frac{1}{30} \frac{\min(2\lambda_1, \lambda_1 - \lambda_2)}{\sqrt{\lambda_1}} \right\} \tag{3.22}$$

as two neighborhoods around global minimizers $\pm\sqrt{\lambda_1} v_1$ respectively.

Definition 3.5.1 (Coordinate-wise Lipschitz continuous). *Let a function $g(x) : \mathbb{S} \mapsto \mathbb{R}$ be continuously differentiable. The gradient of the function $\nabla g(x)$ is coordinate-wise Lipschitz continuous on $\mathbb{S}$ if there exists a positive constant L such that,*

$$|\nabla_i g(x + \alpha e_i) - \nabla_i g(x)| \leq L|\alpha|, \quad \forall\, x, x + \alpha e_i \in \mathbb{S} \text{ and } i = 1, 2, \ldots, n. \tag{3.23}$$

Compared to the usual Lipschitz constant of $\nabla g(x)$, denoted as L_g, we have the relation, $L \leq L_g$. If $g(x)$ is further assumed to be convex, then we have $L_g \leq nL$. If $g(x)$ is twice-differentiable, Definition 3.5.1 is equivalent to $\left|e_i^\top \nabla^2 g(x) e_i\right| \leq L$ for all $i = 1, 2, \ldots, n$ and $x \in \mathbb{S}$. An important consequence of a coordinate-wise Lipschitz continuous function $g(x)$ is that

$$g(x+\alpha e_i) \leq g(x) + \nabla_i g(x)\alpha + \frac{L}{2}\alpha^2, \quad \forall\, x \in \mathbb{S} \text{ and } x+\alpha e_i \in \mathbb{S}. \tag{3.24}$$

The following lemma extends [48, Lemma A.3] to objective function $f(x)$ with matrix A satisfying Assumption 3.2.4.

Lemma 3.5.2. *Function $f(x)$ defined in (3.1) is a continuously differentiable function. Further the gradient function $\nabla f(x)$ is coordinate-wise Lipschitz continuous on either B^+ or B^- with constant $L = 12\lambda_1 + 2\min(2\lambda_1, \lambda_1 - \lambda_2) + 4\max_i |A_{i,i}|$.*

Definition 3.5.3 (strongly $\|\cdot\|_p$-convexity)**.** *Let a function $g(x) : \mathbb{S} \mapsto \mathbb{R}$ be continuously differentiable. It is said to be strongly $\|\cdot\|_p$-convex, if there exists a constant $\mu_p > 0$ such that*

$$g(y) \geq g(x) + \nabla g(x)^\top (y-x) + \frac{\mu_p}{2}\|y-x\|_p^2, \quad \forall x, y \in \mathbb{S}. \tag{3.25}$$

Throughout the paper, we assume $p \geq 1$ in the definition. Definition 3.5.3 is a generalized version of the traditional strong convexity, which corresponds to $p = 2$ case.

Combine (3.25) with the equivalence of different norms in finite dimensional vector space, we obtain for any $p \geq q \geq 1$

$$\begin{aligned} g(y) &\geq g(x) + \nabla g(x)^\top (y-x) + \frac{\mu_p}{2}\|y-x\|_p^2 \\ &\geq g(x) + \nabla g(x)^\top (y-x) + n^{2/p-2/q}\frac{\mu_p}{2}\|y-x\|_q^2, \quad \forall x, y \in \mathbb{S}. \end{aligned} \tag{3.26}$$

Therefore, if $g(x)$ is a strongly $\|\cdot\|_p$-convex function with constant μ_p, then $g(x)$ is a strongly $\|\cdot\|_q$-convex function with constant $\mu_q \geq n^{2/p-2/q}\mu_p$, which is equivalent to $n^{2/q}\mu_q \geq n^{2/p}\mu_p$. On the other hand side, using the inequality of vector norm together with (3.25), we obtain for any $p \geq q \geq 1$,

$$\begin{aligned} g(y) &\geq g(x) + \nabla g(x)^\top (y-x) + \frac{\mu_q}{2}\|y-x\|_q^2 \\ &\geq g(x) + \nabla g(x)^\top (y-x) + \frac{\mu_q}{2}\|y-x\|_p^2, \quad \forall x, y \in \mathbb{S}. \end{aligned} \tag{3.27}$$

Therefore, if $g(x)$ is a strongly $\|\cdot\|_q$-convex function with constant μ_q, then $g(x)$ is a strongly $\|\cdot\|_p$-convex function with constant $\mu_p \geq \mu_q$. Putting two parts together, we have the following inequalities of μ_p and μ_q

$$n^{2/p-2/q}\mu_p \leq \mu_q \leq \mu_p, \tag{3.28}$$

where $p \geq q \geq 1$.

Lemma 3.5.4. *Function $f(x)$ defined in* (3.1) *is strongly $\|\cdot\|_2$-convex on either B^+ or B^- with constant $\mu_2 = 3\min(2\lambda_1, \lambda_1 - \lambda_2)$, and hence, there exists μ_p such that $f(x)$ is strongly $\|\cdot\|_p$-convex for any $p \geq 1$.*

The proof of the strongly $\|\cdot\|_2$-convexity in Lemma 3.5.4 follows an extension of the proof of Lemma A.2 in [48], where the minimum eigenvalue of the Hessian matrix is modified according to the assumption of the matrix A. Combining with (3.28), we have the existence of μ_p for all $p \geq 1$.

B^+ and B^- are two disjoint 2-norm ball around global minimizers. Next we define two sublevel sets D^+ and D^-, contained in B^+ and B^- respectively as

$$D^\pm = \left\{ x \in B^\pm \;\middle|\; f(x) \leq \min_{y \in \partial B^\pm} f(y) \right\}, \tag{3.29}$$

where $\partial B^\pm$ denote the boundary of $B^\pm$. Obviously, two global minimizers lie in $D^\pm$ respectively, i.e., $\pm\sqrt{\lambda_1}v_1 \in D^\pm$. Lemma 3.5.5 shows monotonic decay property of the iteration defined by Algorithm 9 once the iterations falls in $D^\pm$. It also shows that $D^+ \cup D^-$ is a contraction set for the iteration.

Lemma 3.5.5. *Consider function $f(x)$ as defined in* (3.1) *and the iteration follows Algorithm 9 with $k = 1$. For any $x^{(\ell)} \in D^+ \cup D^-$, we have*

$$f\big(x^{(\ell+1)}\big) \leq f\big(x^{(\ell)}\big) - \frac{1}{2L}\left(\nabla_{j_\ell} f\big(x^{(\ell)}\big)\right)^2, \tag{3.30}$$

where j_ℓ is the index of the coordinate being picked. Moreover, we have $x^{(\ell+1)} \in D^+ \cup D^-$.

The proof of Lemma 3.5.5 can be found in Appendix A.2.

In Algorithm 9, we notice that the iteration of $x^{(\ell)}$ in the SCD-Grad-LS samples coordinate j with probability proportional to $\left|\nabla_j f\left(x^{(\ell)}\right)\right|^t$ for some non-negative power t. In the following lemma and theorem, we adopt notation f^* as the minimum of the function and X^* be the set of minimizers, i.e., $X^* = \left\{ \pm\sqrt{\lambda_1} v_1 \right\}$. A distance function between two sets or between a point and a set is defined as, $\operatorname{dist}\big(S_1, S_2\big) = \min_{x \in S_1, y \in S_2} \|x - y\|$.

Lemma 3.5.6. *Consider function $f(x)$ as defined in* (3.1) *and the iteration follows Algorithm 9 with $k = 1$. For any $x^{(\ell)} \in D^+ \cup D^-$,*

$$\mathbb{E}\big[f\big(x^{(\ell+1)}\big) \mid x^{(\ell)}\big] - f^* \leq \left(1 - \frac{\mu_q}{L n^{2-\frac{2}{q}}}\right)\left(f\big(x^{(\ell)}\big) - f^*\right), \tag{3.31}$$

where $q = \frac{t+2}{t+1}$.

The proof of Lemma 3.5.6 can be found in Appendix A.2.

Theorem 3.5.7. *Consider function $f(x)$ as defined in* (3.1) *and the iteration follows Algorithm 9 with $k = 1$. For any $x^{(0)} \in D^+ \cup D^-$,*

$$\mathbb{E}\big[f\big(x^{(\ell)}\big) \mid x^{(0)}\big] - f^* \leq \left(1 - \frac{\mu_q}{L n^{2-\frac{2}{q}}}\right)^\ell \left(f\big(x^{(0)}\big) - f^*\right), \tag{3.32}$$

where $q = \frac{t+2}{t+1}$. *Moreover,*

$$\mathbb{E}\big[\mathrm{dist}\big(x^{(\ell)}, X^*\big)^2 \mid x^{(0)}\big] \leq \frac{2}{\mu_2}\left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)^{\ell} \big(f\big(x^{(0)}\big) - f^*\big)\,. \tag{3.33}$$

The proof of Theorem 3.5.7 can be found in the Appendix A.2. Theorem 3.5.7 is slightly more complicated than Lemma 3.5.6 since Algorithm 9 adopts the exact line search and there are two sublevel sets $D^{\pm}$ for the non-convex objective function $f(x)$. The iteration $x^{(\ell)}$ might jump between the two sets. The monotonicity of the exact line search is the key to extend Lemma 3.5.6 to Theorem 3.5.7.

Remark 3.5.8. *According to Theorem 3.5.7, the convergence rate depends on* $n^{2/q}\mu_q$, *which depends on* q *and hence* t. *The right side of* (3.28) *indicates that, for a problem such that* $\mu_1 = \mu_q = \mu_2$, *the convergence rate of* $q = 1 \Leftrightarrow t = \infty$ *is* n *times larger than that of* $q = 2 \Leftrightarrow t = 0$. *This means that greedy CDM could be potentially* n *times faster than stochastic CDM with uniform sampling. According to the left side of inequality* (3.28), *for* $p \geq q$, *we have* $n^{2/p}\mu_p \leq n^{2/q}\mu_q$, *which means the convergence rate of* q *is equal to or faster than that of* p. *In terms of* t, *we conclude that the convergence rate of smaller* t *is smaller than that of larger* t *and larger* t *potentially leads to faster convergence.*

Remark 3.5.9. *For a fair comparison between CDMs with traditional methods such as gradient descent, we should take* n *iterations of CDM and compare it against a single iteration of gradient descent since one iteration of CDM only updates one coordinate whereas one iteration of gradient descent updates* n *coordinates. Under such a setting, Theorem 3.5.7 implies that*

$$\begin{aligned}\mathbb{E}\big[f\big(x^{(n\ell)}\big) \mid x^{(0)}\big] - f^* &\leq \left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)^{n\ell} \big(f\big(x^{(0)}\big) - f^*\big) \\ &\leq \big(e^{-\frac{\mu_q}{L}n^{\frac{2}{q}-1}}\big)^{\ell} \big(f\big(x^{(0)}\big) - f^*\big)\,,\end{aligned}$$

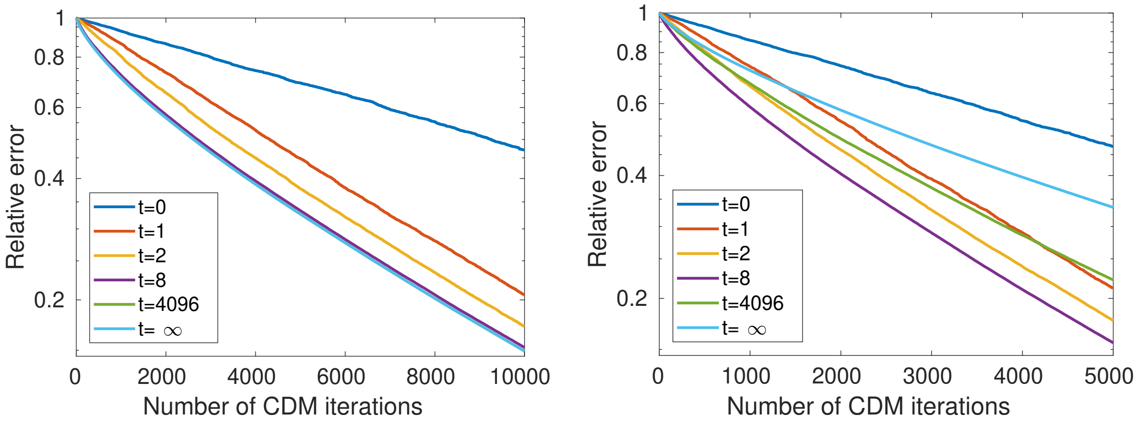
and

$$\mathbb{E}\big[\operatorname{dist}\big(x^{(n\ell)}, X^*\big)^2 \mid x^{(0)}\big] \le \frac{2}{\mu_2}\big(e^{-\frac{\mu_q}{L}n^{\frac{2}{q}-1}}\big)^{\ell}\,\big(f\big(x^{(0)}\big) - f^*\big)\,.$$

For uniform sampling, $t = 0$ *and* $q = 2$*, the decreasing factor is* $e^{-\frac{\mu_q}{L}n^{\frac{2}{q}-1}} = e^{-\frac{\mu_2}{L}}$*, which is independent of* n*. This result implies Algorithm 9 with uniform sampling has the same convergence rate as gradient descent. Combining with Remark 3.5.8, we conclude that Algorithm 9 with arbitrary* $t \ge 0$ *converges at least as fast as gradient descent.*

Remark 3.5.10. *If we extend Definition 3.5.1 to multi-coordinate Lipschitz continuity, and assume* $f(x)$ *is multi-coordinate Lipschitz continuous with* $\widetilde{L}$*, a natural extension of Theorem 3.5.7 would follow. Unfortunately, if the coordinates are sampled independently, in an extreme case when* k *same coordinates are sampled, the convergence rate would be the same as that in Theorem 3.5.7. In the end, the convergence rate would not be improved and could be even worse. This is similar to the argument of CDM vs. full gradient descent method. In practice, when* k *is not too large, and the sampled coordinates are distinct, we do observe* k*-fold speed-up of the iterations. This observation suggests that the convergence rate in Theorem 3.5.7 is not sharp for* $k > 1$*.*

Combining Theorem 3.5.7 together with the monotonicity of μ_q as (3.28) shows that the lower bound of the convergence rate increases monotonically as q decreases. In terms of t, the larger t corresponds to smaller q which leads to larger μ_q for $q = \frac{t+2}{t+1}$. Here we would like to argue that the equality of (3.28) is achievable which demonstrates the power of CDMs with such a sampling strategy. Consider a simple example $f(x) = \|Ax - b\|^2$ for A being a diagonal matrix with diagonal entries $A_{i,i} = 1000$, $i = 1, 2, \ldots, 4999$, $A_{5000,5000} = 1$ and $b = \begin{bmatrix}0 & \cdots & 0 & 1\end{bmatrix}^\top$ being a column vector of size 5000. Through elementary calculations, we can show that



(a) single-coordinate updating (b) double-coordinate updating

Figure 3.2: Convergence behavior of SCD-Grad-Grad method with different choice of t. The axis of iterations on (a) is twice as long as that in (b).

$\nabla^2 f(x) = 2A^2$, $\mu_0 = 2$ and $\mu_\infty \approx 1.99$. Therefore all μ_t lies in the small interval $[1.99, 2]$ and different choice of t would lead to different rate of convergence. We minimize $f(x)$ using the SCD-Grad-Grad method. The step size is fixed to be 10^{-6} and the initial vector is chosen randomly. Figure 3.2 (a) demonstrates the first 10^4 iterations of the SCD-Grad-Grad with different choice of t. The relative error is defined to be $\left(f\left(x^{(\ell)}\right) - f\left(x^*\right)\right)/f\left(x^{(0)}\right)$. Clearly, different choice of t leads to different convergence rates. And we also notice that the most significant improvement appears when increasing t from 0 to 1, which implies that sampling with respect to magnitude of gradient is much better than sampling uniformly. In Figure 3.2 (a), the line with $t = 4096$ and that with $t = \infty$ overlap.

According to either Theorem 3.5.7 or Figure 3.2 (a), it is convincing that for single-coordinate updating CDM, picking the index with largest gradient magnitude should be the optimal strategy. Figure 3.2 (b) investigates the case with double-coordinate updating, i.e., sample two coordinates independently and update two entries simultaneously in every iteration. Comparing Figure 3.2 (a) and (b), we notice for small t, double-coordinate updating strategy almost reduces the required number of iterations by half. As $t \to \infty$, the sampling approaches choosing the coordinate with largest gradient magnitude with probability 1. Sampling two coordinates independently thus results in sampling the same index twice and the iteration behaves similar to the single-coordinate updating CDM. Readers may argue that we should sample without replacement or sampling two indices with two largest gradient magnitudes and updating accordingly. However, when the objective function is non-convex, for example, the objective function $f(x)$ defined in (3.1), the iteration usually stick in the middle of iteration (see Figure 3.1 (a)).

Remark 3.5.11 (Local convergence of greedy coordinate-wise descent method)**.** *The local convergence analysis of SCDMs can be extended to the greedy coordinate-wise descent methods. GCD-Grad-LS as in Algorithm 7 is a special case of SCD-Grad-LS with* $t = \infty$ *and* $k = 1$. [2] *Hence Theorem 3.5.7 proves the local convergence of the GCD-Grad-LS with* $q = 1$.

GCD-LS-LS as in Algorithm 7 conducts optimal coordinate-wise descent every iteration. Therefore, for GCD-LS-LS, Lemma 3.5.5 can be proved for any index of coordinate j *and then Lemma 3.5.6 holds for* $q = 1$. *The modified lemmas lead to the following corollary.*

Corollary 3.5.12. *Consider function* $f(x)$ *as defined in* (3.1) *and the iteration follows Algorithm 7. For any* $x^{(0)} \in D^+ \cup D^-$,

$$\mathbb{E}\big[f\big(x^{(\ell)}\big) \mid x^{(0)}\big] - f^* \leq \left(1 - \frac{\mu_1}{L}\right)^{\ell} \left(f\big(x^{(0)}\big) - f^*\right). \tag{3.34}$$

Moreover,

$$\mathbb{E}\big[\mathrm{dist}\big(x^{(\ell)}, X^*\big)^2 \mid x^{(0)}\big] \leq \frac{2}{\mu_2} \left(1 - \frac{\mu_1}{L}\right)^{\ell} \left(f\big(x^{(0)}\big) - f^*\right). \tag{3.35}$$

---

[2] Previous work [48] provides a local convergence proof for GCD-Grad-LS, which uses the theorem in [61, 62]. Unfortunately, there is a small gap in the proof. The local domain defined in the proof is not a contractive domain of the iteration. Instead, the $D^{\pm}$ defined in (3.29) is local domain of contraction, and can be used to fill in the gap of the proof.

The local convergence of GCD-Grad-LS and GCD-LS-LS are achieveable, while the global convergences are so far still open. Numerically, both methods converge to global minima for most the cases we tested.

Remark 3.5.13 (Convergence behavior in non-convex area)**.** *We demonstrate one example such that the stochastic CDMs outperforms greedy CDMs even all methods are conducting single coordinate updating. We generate a random matrix* $A_{108} - 100I$ *of size 5000, whose the largest eigenvalue is 8 and other eigenvalues are distributed equally on* $[-99, 0)$*. According to Lemma 3.2.6,* 0 *is the only strict saddle point of the problem. If all methods start with a sparse initial vector, in this problem, they all converge to the strict saddle point first, and then escape from it. Figure 3.1 (b) shows the convergence behavior of five methods ordered by the degree of greediness. The horizontal line is the relative error at the strict saddle point 0. As we can see that it takes more time for greedier method to escape from the saddle point. And although SCD-Grad-LS(0) escape from the saddle point fastest, but the overall performance is worse than SCD-Grad-LS(1). This motivates us to apply SCD-Grad-LS with small t in practice. Such methods are almost as efficient as the greedy methods, as shown in Figure 3.2, they are also robust to non-convex objective functions near saddle points.*

Chapter 4

Coordinate Descent Full Configuration Interaction

4.1 Introduction

Solving quantum many-body problem for electrons is a well-known challenging task. While weakly correlated (single-reference) systems can be well approximated using density functional theory and coupled cluster methods such as CCSD(T); strongly corrected (multi-reference) systems remain challenging. The difficulty comes in two aspects: the infamous fermion sign problem and combinatorial scaling of the problem size. Inspired by the discussion of the coordinate descent methods in Chapter 3, we propose an efficient algorithm, named coordinate descent FCI (CDFCI) based on GCD-Grad-LS, to calculate the ground state energy and its corresponding variational wavefunction for both weakly and strongly correlated fermion systems in the framework of full configuration interaction[95].

The rest of the chapter is organized as follows. Section 4.2 presents the CDFCI algorithm. The implementation detail is stated in Section 4.3. In Section 5.3, we demonstrate the efficiency and accuracy of CDFCI via applying it to various molecules including H_2O, C_2, N_2, and Cr_2. Also the binding curve of N_2 is characterized.

4.2 Coordinate descent FCI

As discussed in Section 1.2, we want to solve the ground state and the ground state energy of a quantum system with Hamiltonian

$$\widehat{H} = \sum_{p,q} t_{pq} \hat{a}_p^\dagger \hat{a}_q + \frac{1}{2} \sum_{p,r,q,s} v_{prqs} \hat{a}_p^\dagger \hat{a}_r^\dagger \hat{a}_q \hat{a}_s, \tag{4.1}$$

under second quantization. The ground state $|\Phi_0\rangle$ and the ground state energy E_0 satisfies the time-independent Schrödinger equation

$$\widehat{H} \left|\Phi_0\right\rangle = E_0 \left|\Phi_0\right\rangle, \tag{4.2}$$

where E_0 is the smallest eigenvalue (assume $E_0 < 0$) of $\widehat{H}$.

In FCI formulation, the many-body Hilbert space $\mathcal{H}$ is truncated to a finite-dimensional subspace spanned by all possible Slater determinants

$$\mathcal{V} = \operatorname{span}\big\{\left|D_1\right\rangle, \dots, \left|D_{N_{\mathrm{FCI}}}\right\rangle\big\} \subset \mathcal{H}$$

, which are formed by a finite basis $\{\chi_p\}_{p=1}^{n_{\mathrm{orb}}}$, known as spin-orbitals, of the one-body Hilbert space. Under the basis of the Slater determinants $\{|D_i\rangle\}_{i=1}^{N_{\mathrm{FCI}}}$, the time-independent Schrödinger equation (4.2) has its matrix representation as,

$$Hx = E_0 x, \tag{4.3}$$

known as the FCI eigenvalue problem.

If we define $A = -H$ or $A = I - \delta H$, the FCI eigenvalue problem (4.3) is exactly the leading eigenvalue problem defined in (1.1). However, we will work with H directly throughout this chapter since we aim to solve the particular FCI eigenvalue problem in quantum physics and quantum chemistry. Note in most places, there is just a sign difference.

As the equation (1.7), the FCI eigenvalue problem (4.3) can be reformulated as the following unconstrained non-convex optimization problem,

$$\min_{x \in \mathbb{R}^{N_{\mathrm{FCI}}}} f(x) = \left\| H + x x^\top \right\|_F^2, \tag{4.4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. The gradient of the objective function is

$$\nabla f(x) = 4Hx + 4\left(x^\top x\right) x.$$

As analyzed in Chapter 3, the stationary points are 0 and $\pm\sqrt{-E_i} v_i$ for $E_i < 0$, where v_i is the normalized eigenvector corresponding to E_i. Furthermore, importantly, $\pm\sqrt{-E_0} v_0$ are the only two local minimizers with the same objective value, while other stationary points are saddle points. Thus, solving the optimization problem (4.4) reveals the ground state energy E_0 and the ground state wavefunction coefficient vector v_0. For such an optimization problem, higher order methods converge to global minima efficiently but their per iteration computational costs are too expensive for FCI problems. Hence, only first order methods, like gradient descent methods (GDs), stochastic gradient descent methods (SGDs), and coordinate descent methods (CDMs), are discussed here.

The first order optimization methods applied to (4.4), compared with solving (4.3) using traditional methods, e.g., power method, Davidson method, and Lanczos method, have two main advantages. First, GDs, SGDs, and CDMs do not need any tuning parameters: no diagonal shift is needed to turn the smallest eigenvalue into the largest one in magnitude [49] and the stepsize can be addressed by an exact line search. Second, since no orthonormality constraint appears explicitly in (4.4), solving (4.4) with GDs, SGDs, and CDMs does not need any orthonormalization step. While in traditional methods like Davidson or Lanczos methods, orthonormalization step is needed every a few iterations to avoid numerical instability issue, which would be expensive for FCI problems.

Among the first order optimization methods, CDMs are more suitable for FCI problems. GDs evaluate and update the exact gradient each iteration, which is prohibitively expensive for FCI problems, due to the huge problem dimension. SGDs evaluate and update a stochastic approximation of the full gradient, which is of much cheaper computational cost per iteration. SGDs actually have been applied to FCI problems in an implicit way: as FCIQMC, iFCIQMC, and S-FCIQMC can all be regarded as SGDs applied to a similar objective function as (4.4) with constant stepsizes [57]. SGDs, in general, converge efficiently to a neighborhood of minimizers and then wander around the minimizer due to the stochastic approximation. CDMs select and update a single coefficient each iteration, which is of cheap computational cost. Comparing to GDs, CDMs provably achieve faster convergence rate in terms of the prefactor [49], whereas comparing to SGDs, CDMs are approximately of equal cost per iteration, but is much more stable towards convergence. Further, with a properly designed selecting strategy, CDMs updates different coefficients with different frequencies, taking advantage of the different importance of determinants in FCI problems. Taking these advantages into consideration, we design CDFCI, which is a CDM tailored for FCI problems with compression strategy, to efficiently solve (4.4). This method is described in details below.

4.2.1 Algorithm

The CDFCI algorithm stores two sparse vectors $x^{(\ell)}$ and $z^{(\ell)}$ in the main memory that are contiguously maintained throughout iterations: $x^{(\ell)}$ denotes the computed coefficient vector of the ground state wavefunction in the ℓ-th iteration and $z^{(\ell)}$ is a compressed approximation of $Hx^{(\ell)}$.

Let us first give a sketch of the CDFCI algorithm: In the ℓ-th iteration, CDFCI first finds the $i^{(\ell+1)}$-th determinant with potentially steepest objective function value decrease among all H-connected determinants of $|D_{i^{(\ell)}}\rangle$. Then CDFCI conducts an exact line search to find the optimal update α such that $f(x^{(\ell)}+\alpha e_{i^{(\ell+1)}})$ is minimized and hence the estimator for the ground state energy is reduced, where $e_{i^{(\ell+1)}}$ denotes the indicator vector of $i^{(\ell+1)}$. $z^{(\ell)}$ plays an important role in all above steps. In preparation for next iteration, the vector $z^{(\ell+1)}$ needs to be updated incorporating with $x^{(\ell+1)} = x^{(\ell)} + \alpha e_{i^{(\ell+1)}}$. However, an exact update $z^{(\ell+1)} = z^{(\ell)} + \alpha H_{:,i^{(\ell+1)}}$ could waste limited memory resource on unappreciative determinants. Our compression step updates coefficients only when they do not cost extra memory resource or if they are significantly large in magnitudes. Although the compression step introduces error along the iterations, as we will show, we can still calculate the Rayleigh quotient corresponding to $x^{(\ell+1)}$ exactly, which is used as the ground state energy estimator in CDFCI.

In the following, we will discuss each part of CDFCI in detail and then conclude this section with a pseudo-code for the algorithm.

Determinant-select and coefficient-update

Determinant-select is the first step in each iteration. Assume that the ℓ-th iteration updates determinant $|D_{i^{(\ell)}}\rangle$ and results in a coefficient vector $x^{(\ell)}$. We select the determinant to be updated at the current iteration, $|D_{i^{(\ell+1)}}\rangle$, according to local information at $x^{(\ell)}$. In order to decrease the objective function value to the largest extent, we could select the determinant with the largest magnitude of the approximated gradient at $x^{(\ell)}$, i.e., $|D_{i^{(\ell+1)}}\rangle$ with

$$i^{(\ell+1)} = \arg\max_j \left| 4z_j^{(\ell)} + 4\left(\left(x^{(\ell)}\right)^\top x^{(\ell)}\right) x_j^{(\ell)} \right|,$$

where $z^{(\ell)}$ is a compressed approximation of $Hx^{(\ell)}$. However, the above strategy requires checking each j (i.e., all determinants), which is prohibitive for even moderate size problems. Hence, instead of checking all determinants, CDFCI only checks the H-connected determinants of $|D_{i^{(\ell)}}\rangle$, i.e.,

$$i^{(\ell+1)} = \operatorname*{arg\,max}_{j \in \mathcal{I}_H(i^{(\ell)})} \left| 4z_j^{(\ell)} + 4\left(\left(x^{(\ell)}\right)^\top x^{(\ell)} \right) x_j^{(\ell)} \right|. \tag{4.5}$$

Since our compression strategy introduced later in Section 4.2.1 truncates unappreciative determinants, the expression in (4.5) remains a good approximation of the exact gradient at $x^{(\ell)}$. Empirically, such a gradient-based determinant-select strategy outperforms other perturbation-based determinant-select strategies as used in other SCI algorithms (see Section 5.3 for details).

Once the $i^{(\ell+1)}$-th determinant is selected, CDFCI determines the stepsize by the line search along that direction so to decrease the objective function value by the largest amount. Denoting the update as α, the line search can be formulated as

$$\alpha = \operatorname*{arg\,min}_{\widetilde{\alpha} \in \mathbb{R}} f(x^{(\ell)} + \widetilde{\alpha} e_{i^{(\ell+1)}}). \tag{4.6}$$

Since $h(\widetilde{\alpha}) = f(x^{(\ell)} + \widetilde{\alpha} e_{i^{(\ell+1)}})$ is a quartic polynomial in $\widetilde{\alpha}$, solving the minimization problem (4.6) is equivalent to finding roots of $h'(\widetilde{\alpha})$ – the derivative of $h(\widetilde{\alpha})$. If $h'(\widetilde{\alpha})$ has a unique root, then the root is the minimizer. If $h'(\widetilde{\alpha})$ has two roots, then the one with multiplicity one is the minimizer. If $h'(\widetilde{\alpha})$ has three roots, then the one further away from the middle one is the minimizer. Given the update α, we can easily update $x^{(\ell)}$ as

$$x_i^{(\ell+1)} = \begin{cases} x_i^{(\ell)}, & i \neq i^{(\ell+1)}; \\ x_i^{(\ell)} + \alpha, & i = i^{(\ell+1)}. \end{cases} \tag{4.7}$$

In CDFCI, we also need to maintain $z^{(\ell+1)} \approx Hx^{(\ell+1)}$ for future determinant-select steps. Since only one coefficient is updated in $x^{(\ell)}$, the corresponding $z^{(\ell)}$ can be updated accordingly as,

$$z^{(\ell+1)} \approx Hx^{(\ell+1)} \approx z^{(\ell)} + \alpha H_{:,i^{(\ell+1)}}. \tag{4.8}$$

Therefore, each update step requires evaluation of all H-connections from $|D_{i^{(\ell+1)}}\rangle$. Besides the update from $x^{(\ell+1)}$, we also recalculate the current $i^{(\ell+1)}$-th entry in $z^{(\ell+1)}$ to guarantee the correctness and increase the numerical stability of our algorithm. Such a recalculation could improve the accuracy of the determinant-select (4.5) and line search (4.6) in the following iterations, and also provide an accurate Rayleigh quotient as the estimator of the ground state energy as in (4.11). We argue this correction comes for free in addition to (4.8), since

$$\begin{aligned} z^{(\ell+1)}_{i^{(\ell+1)}} &= H_{i^{(\ell+1)},:} x^{(\ell+1)} \\ &= \sum_{j \in \mathcal{I}_H(i^{(\ell+1)})} (H_{j,i^{(\ell+1)}})^* x^{(\ell+1)}_j, \end{aligned} \tag{4.9}$$

where $(H_{j,i^{(\ell+1)}})^*$ denotes the complex conjugate of $H_{j,i^{(\ell+1)}}$, which has already been evaluated when updating (4.8).

Coefficient compression

Since CDFCI initializes $x^{(0)}$ with the reference determinant $|D_1\rangle$ and $z^{(0)} = Hx^{(0)}$, the coefficient of the reference determinant in $z^{(0)}$ is nonzero and the reference determinant is in $z^{(0)}$. In later iterations, CDFCI is designed to follow one rule: if a determinant is in $x^{(\ell)}$, then it is in $z^{(\ell)}$ as well. Under this rule, if a determinant $|D_j\rangle$ is neither in $x^{(\ell)}$ nor in $z^{(\ell)}$, according to (4.5), this determinant has zero value therein and will not be selected. Hence (4.5) selects either a new determinant not in $x^{(\ell)}$ but in $z^{(\ell)}$ or an old determinant already in both $x^{(\ell)}$ and $z^{(\ell)}$. In CDFCI, thus, the compression strategy compresses only the unappreciative determinants in $z^{(\ell)}$ to control the computation and memory cost, which in turn restricts the growth of the coefficient vector $x^{(\ell)}$.

Detailed compression strategy is as follows. When a coefficient $\alpha H_{j,i^{(\ell+1)}}$ is added to $b_j^{(\ell)}$, we use a predefined tolerance ε to compress the update. If the j-th determinant is already selected before, then $\alpha H_{j,i^{(\ell+1)}}$ is added to $z_j^{(\ell)}$ without any compression. If the j-th determinant has not been selected in $z^{(\ell)}$, but the update is quantitatively large, i.e., $\left|\alpha H_{j,i^{(\ell+1)}}\right| > \varepsilon$, it indicates that the j-th determinant is appreciable and the j-th determinant will be added to $z^{(\ell)}$ with the coefficient $\alpha H_{j,i^{(\ell+1)}}$. Otherwise, the update is truncated, i.e., the coefficient in $z^{(\ell)}$ remains 0. The described compression strategy is deterministic and satisfies the rule that determinants with nonzero coefficients in $x^{(\ell)}$ are in $z^{(\ell)}$ as well. For molecules, such a deterministic strategy outperforms other strategies including stochastic compression schemes [50] due to its effectiveness and cheap cost.

Energy estimation

Although the vector $z^{(\ell+1)}$ is compressed, we emphasize that the Rayleigh quotient $r(x) = \frac{x^\top Hx}{x^\top x}$ can be maintained accurately for $x^{(\ell+1)}$, which is used in CDFCI as the estimator of the ground state energy. First, the squared norm of $x^{(\ell)}$ can be updated up to numerical error, i.e.,

$$\left(x^{(\ell+1)}\right)^\top x^{(\ell+1)} = \left(x^{(\ell)}\right)^\top x^{(\ell)} + 2\alpha c_{i^{(\ell+1)}}^{(\ell)} + \alpha^2. \tag{4.10}$$

An exact update can be computed for the numerator of the Rayleigh quotient as well, i.e.,

$$\begin{aligned}\left(x^{(\ell+1)}\right)^\top Hx^{(\ell+1)} = & \left(x^{(\ell)}\right)^\top Hx^{(\ell)} \\ & + 2\alpha z_{i^{(\ell+1)}}^{(\ell+1)} - \alpha^2 H_{i^{(\ell+1)},i^{(\ell+1)}},\end{aligned} \tag{4.11}$$

where $z_{i^{(\ell+1)}}^{(\ell+1)}$ is recalculated accurately as discussed in Section 4.2.1 around (4.9). Hence this update is accurate. The Rayleigh quotient of the updated variational wavefunction, $r(x^{(\ell+1)})$ is the ratio of two accurately cumulated quantity and hence accurate. Both in theoretical and numerical results, we observed that the Rayleigh quotient is much more accurate than the projected energy estimator [8, 6, 16, 70], which is $\frac{z_1^{(\ell+1)}}{x_1^{(\ell+1)}}$ in our notation.

Stopping criteria can be tricky for all iterative methods, including DMRG, FCIQMC, HCI, SHCI, etc, and is also the case for CDFCI. Here we propose three suggestions. As for many iterative methods, we can stop the iteration if the updated value is small. For CDFCI, it is suggested to monitor the cumulated updated values across a few iterations as the stopping criteria. Another stopping criteria is based on the change of the Rayleigh quotient. Usually, we observe monotone decay of the Rayleigh quotient before iteration converges. Therefore, we can stop the algorithm if the decay of the Rayleigh quotient after a few iterations is small. The third suggestion is based on the ratio of the number of nonzero coefficients in z and x. When the algorithm converges, this ratio converges to 1. When the ratio is close to 1, the error introduced by the compression slows down the convergence significantly. Hence more iterations do not make much accuracy improvement. Mixed use of these stopping criteria is suggested in practice.

We conclude this section with a pseudo-code for CDFCI:

1. Initialize $x^{(0)}$ by the reference state $|D_1\rangle$ with coefficient being 1, initialize $z^{(0)} = Hx^{(0)}$, and initialize $\ell = 0$.
2. Select a determinant with the largest gradient magnitude according to (4.5). Denote the selected determinant as $|D_{i^{(\ell+1)}}\rangle$.
3. Solve a cubic polynomial equation to obtain the optimal update α for the selected determinant. Update the $i^{(\ell+1)}$-th coefficient as (4.7).

4. Update $z_j^{(\ell+1)} = z_j^{(\ell)} + \alpha H_{j,i^{(\ell+1)}}$ if the j-th determinant is already selected in $z^{(\ell)}$. Otherwise, add new determinant $|D_j\rangle$ to $z^{(\ell+1)}$ with coefficient $\alpha H_{j,i^{(\ell+1)}}$ if $\left|\alpha H_{j,i^{(\ell+1)}}\right| > \varepsilon$. Exactly reevaluate $z_{i^{(\ell+1)}}^{(\ell+1)}$ as (4.9).

5. Update $\left(x^{(\ell+1)}\right)^\top x^{(\ell+1)}$ and $\left(x^{(\ell+1)}\right)^\top Hx^{(\ell+1)}$ as (4.10) and (4.11) respectively. Calculate the exact Rayleigh quotient for $x^{(\ell+1)}$.

6. Repeat 2-5 with $\ell \leftarrow \ell + 1$ until some stopping criteria is achieved.

4.3 Implementation and complexity

We now give some implementation details of the algorithm, focusing on the computationally expensive parts and the numerical stability issues. In the end of this section, a per iteration complexity analysis is conducted.

The indices of Slater determinants are encoded in the way that coincides with that in the second quantization. Suppose there are n_{orb} spin-orbitals in the FCI discretization, and n_{elec} electrons in the system. Then a Slater determinant is encoded as an n_{orb}-bit binary string, with each bit representing a spin-orbital. The spin-orbital is occupied if the corresponding bit is 1 and unoccupied if the bit is 0. The n_{orb}-bit binary string is stored as an array of 64-bit integers. Thus, $\lceil \frac{n_{\text{orb}}}{64} \rceil$ integers are needed to represent the index of a determinant.

We now focus on the implementation detail of the determinant-update step, as it dominates the runtime. Since the vectors z and x are sparse and compressed in the algorithm, their entries cannot be contiguously stored in memory. For CDFCI, we have tried two different data structure implementations for the combined vector (since the indices of nonzero coefficients of x are contained in z, these two vector are stored together in a single data structure): red-black tree and hash table [17].

Red-black tree is a memory compact representation of the vector: Given that z at current iteration has n nonzero coefficients, red-black tree requires $O(n)$ memory. Inserting, updating and deleting a nonzero coefficient to this red-black tree cost $O(\log n)$ operations. The drawback is that each nonzero coefficient is a node on the tree and hence requires extra memory to store pointers, which turns out to be more expensive comparing to the hash table.

In CDFCI, therefore, we prefer to use a fixed-size open addressing hash table. The hash function mapping a configuration string to an array index is chosen as

$$\mathrm{Hash}(d) = s \cdot d \pmod{p}, \tag{4.12}$$

where the size of the hash table is chosen to be a large prime number p; d is the vector of $\lceil \frac{n_{\mathrm{orb}}}{64} \rceil$ integers with bits representing the configuration of the determinant; and s is a fixed vector of the same length as d with entries randomly chosen from $[0, p-1]$ during the CDFCI initialization step. In our current implementation, for each execution of the algorithm, we allocate an array of size approaching machine memory limit for the hash table, which could be modified to enable dynamic resizing feature in order to be memory compact. Inserting, updating and deleting a nonzero coefficient in hash table cost $O(1)$ operations on average; while in the worst case, when the table is almost full, inserting and deleting operation would cost $O(p)$ operations. In order to avoid such inefficient scenarios, we limit the load factor below 80%. In practice, these settings of hash table work well and significantly outperform red-black tree. All the numerical results in this paper are produced with hash table.

Besides the expensive data accessing step, the computational expensive step is the evaluation of $H_{:,i^{(\ell+1)}}$. Let $N_H = \max_i |\mathcal{I}_H(i)|$ be the maximum number of nonzero entries in columns of the Hamiltonian matrix. Although $N_H \ll N_{\mathrm{FCI}}$, N_H still scales as $O(n_{\mathrm{elec}}^2 n_{\mathrm{orb}}^2)$. The computational cost for evaluating each entry $H_{i,j}$ also depends moderately on n_{elec}. CDFCI uses an efficient Fortran implemented open source quantum chemistry code HANDE-QMC as backend for the evaluation of Hamiltonian entries.

Shared memory parallelism based on OpenMP is used in our implementation. For each iteration, the double excitation calculation is the bottleneck for the evaluation $H_{:,i^{(\ell+1)}}$, which is embarrassingly parallelized with OpenMP. In terms of runtime, accessing a nonzero coefficient of z and x is also expensive due to the lack of memory continuity. Therefore, we also parallelize the access to $z_{\mathcal{I}_H(i^{(\ell+1)})}$ and $x_{\mathcal{I}_H(i^{(\ell+1)})}$ with OpenMP. Due to the possible collision of the hash function of open addressing, we partition the hash array into 2000 blocks and set locker for each block, such that no two threads can access the same block simultaneously. Increasing the number of lockers would reduce the idling time of threads but would increase the memory cost. We did not try to optimize the number of blocks.

Last point on implementation focuses on the numerical stability of the Rayleigh quotient. Different from other iterative methods, CDFCI updates one determinant per iteration. Hence, for large systems, the number of iteration could easily go beyond 10^8. For cumulated quantities such as $x^\top x$ and $x^\top Hx$, the value is updated at least 10^8 times, hence the accumulated numerical error could pollute the chemical accuracy, and thus careful treatment is needed. In our implementation, we use a quadruple-precision floating point for $x^\top x$ and $x^\top Hx$ such that the relative error is at most 10^{-16} unless the number of iteration exceeds 10^{16}.

Let us remark that our current implementation of CDFCI is by no means optimal. The bottleneck of the current implementation is the naïve hash table. Random access to the main memory is expensive since the cache hierarchy is not fully adapted. An optimized hash table may improve the performance by a big constant.

To conclude the section, let us conduct a leading order per iteration complexity analysis for CDFCI. In determinant-select step, since all $z_i^{(\ell)}$ and $x_i^{(\ell)}$ for $i \in \mathcal{I}_H(i^{(\ell+1)})$ have been accessed in the previous iteration, $i^{(\ell+1)}$ can be computed without paying the cost of accessing the data structure of z and x. Hence the leading cost is $O(N_H)$ with a small prefactor. Line search and updating x cost $O(1)$ operations and are hence negligible. Updating z is the most expensive step throughout the algorithm. It requires evaluating $O(N_H)$ entries of Hamiltonian matrix and accessing z and x $O(N_H)$ times. This step costs $O(N_H)$ operation with a prefactor being the Hamiltonian per entry evaluation cost plus the averaged data structure accessing cost. In our implementation, the compression step is combined with the updating step. Once $H_{:,i^{(\ell+1)}}$ has been evaluated, $z_i^{(\ell)}$ and $x_i^{(\ell)}$ for $i \in \mathcal{I}_H(i^{(\ell+1)})$ are accessed, then exact update of $z_{i^{(\ell+1)}}^{(\ell+1)}$, cumulative updates of $x^\top x$ and $x^\top Hx$ cost $O(N_H)$ operations with a small prefactor. Overall, CDFCI costs $O(N_H)$ operations per iteration with the prefactor dominated by the computation cost of one Hamiltonian entry and the averaged access cost of the data structure. The memory cost of CDFCI is dominated by the cost of allocating the data structure.

Chapter 5

Numerical Results

5.1 Comparison of FCIQMC and FRI

In this section, we give some numerical tests of the FCIQMC and FRI algorithms, and their variance iFCIQMC and Hard Thresholding to compare the performance. The numerical problem is to compute the ground energy of a Hamiltonian H for a quantum system. As discussed before, we define $A = I - \delta H$ for δ small so the problem is equivalent to find the largest eigenvalue of A. We will test these methods with two types of model systems: the 2D fermionic Hubbard model and small chemical molecules under the full CI discretization. The Hamiltonians for these have the same structure. In our test examples, we choose the total spin $S^{\text{tot}} = 0$. Therefore the dimension of the space is $\binom{N^{\text{orb}}}{N^{\text{elec}}/2}^2$, neglecting other constraints like symmetry. The dimension grows exponentially as N^{orb} and N^{elec} grows. Here we summarize the system in our numerical tests in the Table 5.1:

Table 5.1: Test systems for FCIQMC and FRI

System	N^{orb}	N^{elec}	dimension	HF energy	Ground energy
4×4 Hubbard	16	10	1.2×10^6	-17.7500	-19.5809
Ne, aug-cc-pVDZ	23	10	1.4×10^8	-128.4963	-128.7114
H_2O, cc-pVDZ	24	10	4.5×10^8	-76.0240	-76.2419

The exact ground energy of the Hubbard model and Ne are computed using exact power iteration, and the ground energy of H_2O is from the paper [64]. We use HANDE-QMC[85] (`http://www.hande.org.uk/`), an open source stochastic quantum chemistry program written in Fortran, for FCIQMC and iFCIQMC calculation. FRI and HT subroutines are implemented in Fortran based on HANDE-QMC. The

Hamiltonian of the Hubbard model is included in the HANDE-QMC package and the Hamiltonians of Ne and H_2O are calculated using RHF (restricted Hartree Fock) by Psi4 (`http://www.psicode.org/`), an open source ab initio electronic structure package. The code to generate the entries of Hamiltonian H is the same for all four algorithms, so the comparison among algorithms is fair in terms of computational time. The four algorithms are tested on a computer with 6 Core Xeon CPU at 3.5GHz and 64GB RAM.

Note that our comparison is mostly for illustrative purpose and should not be taken as benchmark tests for the various algorithms especially for large scale calculations, which would depend on parallel implementation, hardware infrastructure, etc. On the other hand, even for small problems, the numerical results still offer some suggestions on further development of inexact power iteration based solvers for many-body quantum systems.

5.1.1 Hubbard Model

The Hubbard model is a standard model used in condensed matter physics, which describes interacting particles on a lattice. In real space, the Hubbard Hamiltonian is

$$\widehat{H} = -\sum_{\langle r,r'\rangle,\sigma} \hat{c}^{\dagger}_{r,\sigma}\hat{c}_{r',\sigma} + U\sum_{r}\hat{n}_{r\uparrow}\hat{n}_{r\downarrow}, \tag{5.1}$$

where we have scale the hopping parameter to be 1 and so the on-site repulsion parameter U gives the ratio of interaction strength relative to the kinetic energy. We choose an intermediate interaction strength $U = 4$ in our test.

In the d dimensional Hubbard Hamiltonian (5.1), r is a d-dimensional vector representing a site in the lattice, $\langle r, r'\rangle$ means r and r' are the nearest neighbor, and σ takes values of $\uparrow$ and $\downarrow$, which is the spin of the electron. $\hat{c}_{r,\sigma}$ and $\hat{c}^{\dagger}_{r,\sigma}$ are the annihilation and creation operator of electrons at site r with spin σ. They satisfy the commutation relations

$$\{\hat{c}_{r,\sigma}, \hat{c}^{\dagger}_{r',\sigma'}\} = \delta_{r,r'}\delta_{\sigma,\sigma'}, \qquad \{\hat{c}_{r,\sigma}, \hat{c}_{r',\sigma'}\} = 0, \quad \text{and} \quad \{\hat{c}^{\dagger}_{r,\sigma}, \hat{c}^{\dagger}_{r',\sigma'}\} = 0,$$

where $\{A, B\} = AB + BA$ is the anti-commutator. $\hat{n}_{r,\sigma}$ is the number operator and defined as $\hat{n}_{r,\sigma} = \hat{c}^{\dagger}_{r,\sigma}\hat{c}_{r,\sigma}$. We will consider Hubbard model on a finite 2D lattice with periodic boundary condition.

When the interaction strength U is small, it is better to work in the momentum space instead of the real space, since the planewaves are the eigenfunctions of the kinetic part of the Hamiltonian. The annihilation operator in momentum space is $\hat{c}_{k,\sigma} = \frac{1}{\sqrt{N^{\text{orb}}}} \sum_r e^{ik\cdot r}\hat{c}_{r,\sigma}$, where $k = (k_1, k_2)$ is the wave number and N^{orb} is the total number of orbitals or sites. The Hubbard Hamiltonian in momentum space is then

$$\widehat{H} = \sum_{k,\sigma} \varepsilon(k)\hat{n}_{k,\sigma} + \frac{U}{N^{\text{orb}}} \sum_{k,p,q} \hat{c}^{\dagger}_{p-q,\uparrow}\hat{c}^{\dagger}_{k+q,\downarrow}\hat{c}_{k,\downarrow}\hat{c}_{p,\uparrow}, \tag{5.2}$$

where $\varepsilon(k) = -2\sum_{i=1}^{2} \cos(k_i)$.

Written as a matrix, the Hubbard Hamiltonian in the momentum space is just a real symmetric matrix with diagonal entries $\varepsilon(k)$ and off-diagonal either 0 or $\pm\frac{U}{N^{\text{orb}}}$. For inexact power iteration, we take $A = I - \delta H$ with $\delta = 0.01$. In our numerical test, we will use the projected energy estimator for the smallest eigenvalue of H; the projected vector v_* is chosen to be the Hartree-Fock state. The initial iteration of all methods is also chosen as the Hartree-Fock state (a vector whose only nonzero entry is at the Slater determinant corresponding to the Hartree-Fock ground state of the system).

Figure 5.1 plots the error of projected energy of each iteration versus wall-clock time (first 1500 seconds) for a typical realization. The error is defined as the difference between the projected energy estimate and the exact ground energy. The complexity parameters of the algorithms are shown in Table 5.2, which are chosen such that FRI and FCIQMC use about the same amount of memory (e.g., the particle number in FCIQMC is roughly equal to the non-zero entries of the matrix-vector product in FRI or HT before compression), and also chosen so large that all the algorithms converge. The time per iteration listed in Table 5.2 is averaged over several realizations and is used in the Figure 5.1.

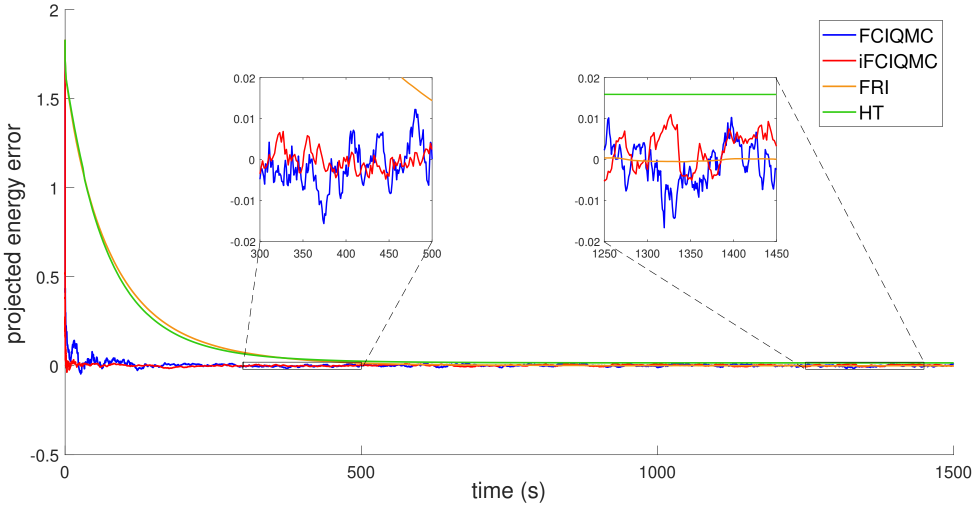


Figure 5.1: Convergence of the projected energy with respect to time for System 1, a 4×4 Hubbard model for comparison between FCIQMC and FRI.

Table 5.2: Parameters and numerical results for System 1, a 4×4 Hubbard Model for comparison between FCIQMC and FRI.

	m	$\|Av_t\|_0$	avg. error	std.	MSE
FCIQMC	1.7×10^6	-	4.4×10^{-4}	3.0×10^{-4}	2.6×10^{-7}
iFCIQMC	1.7×10^6	-	3.2×10^{-4}	2.2×10^{-4}	1.8×10^{-7}
FRI	3.0×10^4	9.4×10^5	1.2×10^{-4}	6.1×10^{-5}	2.8×10^{-8}
HT	3.0×10^4	7.2×10^5	1.6×10^{-2}	-	2.5×10^{-4}

	τ_{auto}	compr. error	time/iter.(s)
FCIQMC	14.1	4.6×10^{-2}	1.1
iFCIQMC	13.8	2.6×10^{-2}	0.91
FRI	13.7	1.6×10^{-1}	3.6
HT	-	4.5×10^{-3}	3.5

As shown in Figure 5.1, all four algorithms converge to result close to the exact eigenvalue and the estimated value from each iteration stays around the eigenvalue for a long time. FCIQMC and iFCIQMC take much less time to converge, thanks to their much lower-cost inexact matrix-vector multiplication compared to FRI and HT, but the variance is also larger. In terms of iteration number, the convergence of the four algorithms is similar, which can be understood from our analysis since it is the same eigenvalue gap of the Hamiltonian that drives the convergence. As we mentioned already, per iteration, the FCIQMC and iFCIQMC is much cheaper in comparison. The reason is that FRI and HT need to access all nonzero elements of A for each column associated with a non-zero entry in the current iterate (for multiplying A with the sparse vector), while FCIQMC and iFCIQMC just need to randomly pick some, without accessing the others. The number of non-zero entries per row is large and accessing elements of A is quite expensive for FCI type problems. More quantitatively, we see in Table 5.2 that for a sparse vector of 3×10^4 non-zero entries in FRI, after multiplication by A before compression, the number of non-zero entries increases to roughly 10^6. Thus for this problem, on average, each column has about 40 nonzero entries that FRI needs to access, while FCIQMC algorithm only needs access of few entries after the random choice.

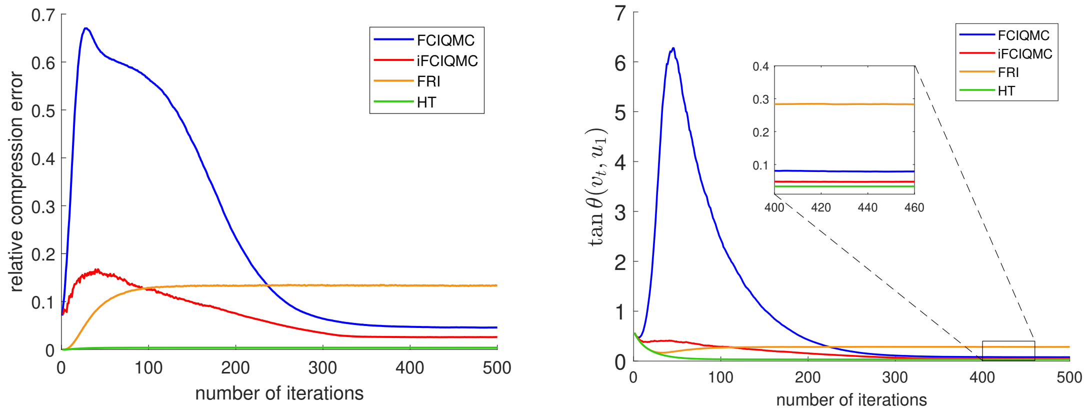
After convergence, the projected energy of FCIQMC and iFCIQMC fluctuate around the exact ground state energy. Although iFCIQMC is biased, the bias is not large for the current problem, while the variance is smaller than FCIQMC. So iFCIQMC is an effective strategy for bias-variance trade-off. The projected energy of FRI also varies around the true energy, and the variance is much smaller than FCIQMC or iFCIQMC. HT is deterministic and the projected energy shows no variance. However the bias is also quite visible.

We can average the projected energy over the path to get a better estimate.

The variance of the estimator will decay to zero as we include longer time period in the average. Thus, due to unbiasedness, the error of FCIQMC and FRI can be made smaller if we run for long enough. In Table 5.2, we give more quantitative comparison of the results of the algorithms. The quantities in the table are defined as below

avg. error	$\displaystyle \frac{1}{w}\sum_{i=i_0}^{i_0+w-1} \left\lvert E_i - E^{\text{true}} \right\rvert$
std.	$\displaystyle \sqrt{\frac{1}{w-1}\sum_{i=i_0}^{i_0+w-1}\left(E_i - \frac{1}{w}\sum_{j=i_0}^{i_0+w-1} E_i\right)^2}\sqrt{\frac{1+2\tau_{\text{auto}}}{W}}$
MSE	$\text{avg. error}^2 + \text{std.}^2$
τ_{auto}	$\displaystyle \sum_{t=1}^{w-1} \frac{\frac{1}{w-1}\sum_{i=i_0}^{i_0+w-t-1}\left(E_i - \frac{1}{w}\sum_{j=i_0}^{i_0+w-1} E_i\right)\left(E_{i+t} - \frac{1}{w}\sum_{j=i_0}^{i_0+w-1} E_i\right)}{\frac{1}{w-1}\sum_{i=i_0}^{i_0+w-1}\left(E_i - \frac{1}{w}\sum_{j=i_0}^{i_0+w-1} E_i\right)^2}$
compr. error	$\displaystyle \frac{1}{w}\sum_{i=i_0}^{i_0+w-1} \frac{\lVert \xi^{(i+1)} \rVert_2}{\lVert Ax^{(i)} \rVert_2}$

Here E^{true} is the true ground energy obtained by exact power iteration, i_0 is a burn-in parameter and w is the window size of the average. For FCIQMC and iFCIQMC, $w = 1600$ and $i_0 = 2400$. For FRI and HT, $w = 400$ and $i_0 = 600$. The numerical tests show that the quantities above are insensitive to the choice of w and i_0, as long as the algorithms indeed converge after i_0 steps and the window size w is not too small. τ_{auto} is the integrated autocorrelation time and W is the number of iterations averaged. The std. is short for the standard deviation of the sample mean $\bar{E}^{(W)}$ defined as $\bar{E}^{(W)} = \frac{1}{W}\sum_{i=i_0}^{i_0+W-1} E_i$. Since the time cost per iteration of different algorithms is quite different, to make a fair comparison, we take $W = \frac{10000}{\text{time per iter.}}$ for each algorithm. It gives the standard error of the sample mean if we run each algorithm for 10000 seconds after convergence. The mean square error (MSE) is simply defined to incorporate the variance and bias together.



(a) Relative compression error $\|\xi_{t+1}\|_2/\|Av_t\|_2$ as a function of iteration steps.

(b) Angle between the iterate and the exact ground state $\tan\theta(v_t, u_1)$ as a function of iteration steps.

Figure 5.2: Convergence of error and ground state of 4×4 Hubbard model for comparison between FCIQMC and FRI.

To further obtain insights of the interplay between the error per step of inexact power iteration and the convergence, we plot in Figure 5.2 the relative compression error and the tangent of the angle between v_t and the exact eigenvector u_1. We observe that that FRI and HT reach convergence after about 100 steps and FCIQMC and iFCIQMC converge after about 350 steps; the more steps of FCIQMC and iFCIQMC are related with the first phase of the algorithm where the particle number is exponentially growing. This can be seen from Figure 5.2(left) as the huge error growth of the initial stage of the iterations. Only when the particle number reaches a certain level, the compression error becomes small and the power iteration convergence kicks in.

After convergence, FRI has the largest compression error and HT has the smallest. The compression error of iFCIQMC is also smaller than the one of FCIQMC. It is reasonable since HT and iFCIQMC reduce variance and thus compression error compared with the fully stochastic FRI and FCIQMC. As shown in Figure 5.2, in this example with the parameter choice, FCIQMC has smaller compression error than FRI; and the larger the compression error is, the further v_t is away from the true eigenvector u_1. This agrees with the theoretical results we obtain in Theorem 2.2.3, because $\tan\theta(v_t, u_1)$ is controlled by the error ξ_t at each step.

We remark that the $\tan\theta(v_t, u_1)$ error measure does not directly translate to the error of the projected energy estimator using say the Hartree-Fock state. In fact, we observe in Figure 5.1 and Table 5.2 that per iteration, the projected energy estimated by FRI is smaller than FCIQMC and iFCIQMC. As an explanation, in our parameter regime, the exact ground state has a large overlap with the Hartree-Fock state, so in FRI, that component is kept unchanged in the compression, while for FCIQMC and iFCIQMC, the stochastic error is more uniformly distributed over all the entries. This behavior seems more problem dependent though, as we will see in the chemical molecular examples that the MSE of FRI become comparable with FCIQMC.

5.1.2 Molecules

We also tested the four algorithms for some molecule examples. The FCI Hamiltonian is obtained by a Hartree-Fock calculations in a chosen chemical basis (for single-particle Hilbert space), such as cc-PVDZ. We choose Ne and H_2O at equilibrium geometry as examples, which is described in Table 5.1. The time step is taken as $\delta = 0.01$.

The convergence of projected energy error versus wall-clock time is shown in Figure 5.3 and Figure 5.4 respectively. The parameter choice of the algorithms and more quantitative comparison are shown in Table 5.3 and Table 5.4. The four algorithms also work well for molecule systems. The convergence behavior is similar to the Hubbard case.

The complexity parameter m needed to achieve convergence depends on the system. The ratio m/N of Ne is smaller than H_2O. The time cost of FRI and HT is much larger than FCIQMC and iFCIQMC, because they require the exact matrix-vector multiplication Av_t, which is still expensive although v_t is sparse. Unlike the Hubbard case where FRI gives much smaller error, the MSE of FRI is similar to FCIQMC and iFCIQMC in these cases.

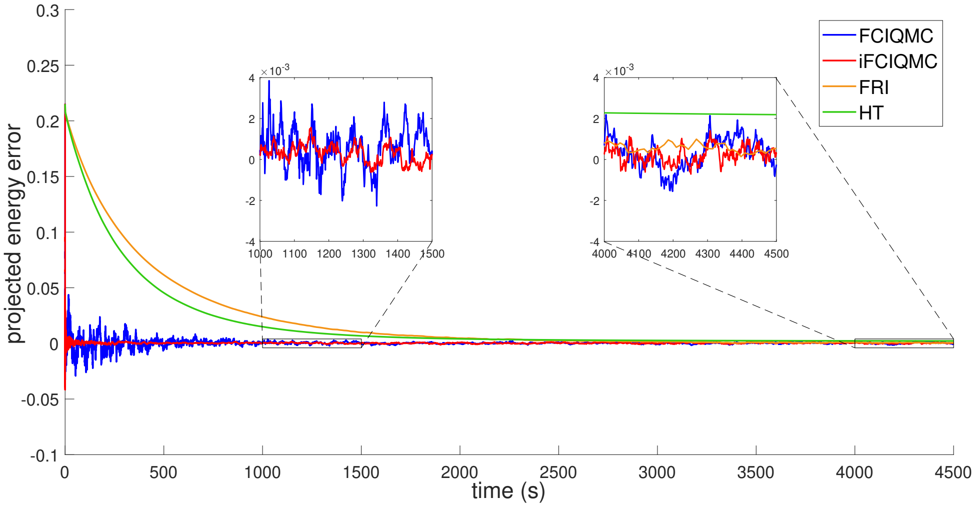


Figure 5.3: Convergence of the projected energy with respect to time for Ne in aug-cc-pVDZ basis for comparison between FCIQMC and FRI

Table 5.3: Comparison of FCIQMC and FRI for Ne in aug-cc-pVDZ basis

	m	$\|Av_t\|_0$	avg. error	std.	MSE
FCIQMC	1.8×10^6	-	1.1×10^{-4}	3.8×10^{-5}	1.8×10^{-8}
iFCIQMC	1.8×10^6	-	7.7×10^{-5}	2.4×10^{-5}	9.6×10^{-9}
FRI	1.0×10^4	7.9×10^6	8.0×10^{-5}	4.8×10^{-5}	1.1×10^{-8}
HT	1.0×10^4	3.3×10^6	1.7×10^{-3}	-	2.8×10^{-6}

	τ_{auto}	time/iter.(s)
FCIQMC	12.4	1.5
iFCIQMC	15.3	1.1
FRI	12.3	11.6
HT	-	7.9

In summary, the numerical examples show that the FCIQMC, FRI and their

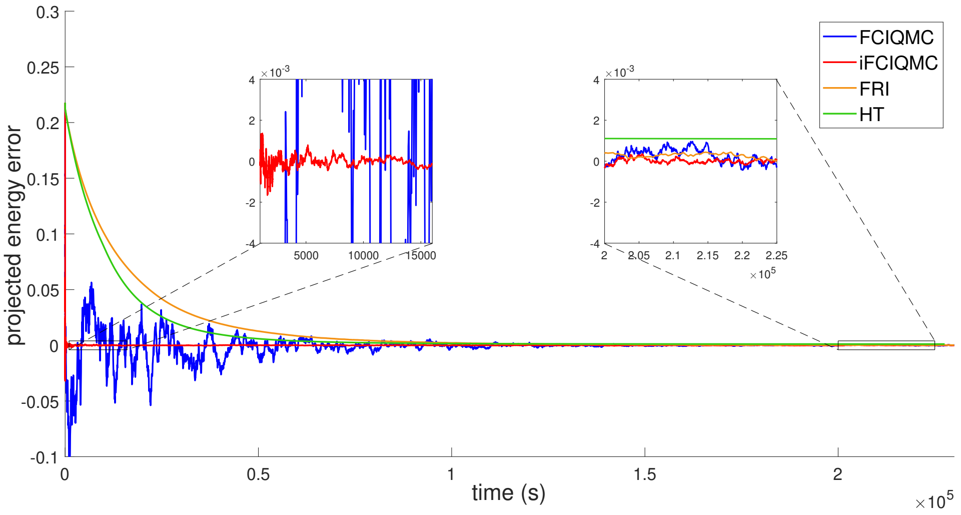


Figure 5.4: Convergence of the projected energy with respect to time for H_2O in cc-pVDZ basis for comparison of FCIQMC and FRI

variants can achieve convergence using much less memory and computational time compared to the standard power iteration. The stochastic algorithms FCIQMC, iFCIQMC and FRI give better estimates than the deterministic method HT in general. The numerical test also points out directions to further improve these inexact power iterations, including variance and memory cost reduction of the inexact matrix-vector multiplication and efficient parallel implementation to overcome the memory bottleneck. These will be leaved for future works.

5.2 Comparison of CDMs

In this section, we perform numerical tests of the coordinate descent methods on two different kinds of matrices. The first numerical test calculates the largest eigenvalue of a random symmetric dense matrix, where we have control of the difficulty of the problems. The leading eigenvector is evenly distributed among all the entries. The second test calculates the ground energy and ground state of the two dimensional (2D) Hubbard model, which is one of our target applications for the quantum many-body

Table 5.4: Comparison of FCIQMC and FRI for H_2O in cc-pVDZ basis

	m	$\|Av_t\|_0$	avg. error	std.	MSE
FCIQMC	6.0×10^7	-	4.1×10^{-5}	1.7×10^{-4}	2.1×10^{-9}
iFCIQMC	6.0×10^7	-	1.4×10^{-5}	5.3×10^{-5}	2.7×10^{-10}
FRI	1.2×10^5	1.6×10^8	2.2×10^{-5}	1.2×10^{-4}	8.9×10^{-10}
HT	1.2×10^5	3.4×10^7	1.1×10^{-3}	-	1.2×10^{-6}

	τ_{auto}	time/iter.(s)
FCIQMC	25.6	54.1
iFCIQMC	119	36.6
FRI	12.8	379.3
HT	-	227.0

calculation. The ground state problem in quantum many-body system is equivalent to find the smallest eigenvalue of a huge sparse symmetric matrix, with each coordinate representing the coefficient of a Slater determinant. Although the eigenvector in this example is dense, it is also ‘sparse’ in some sense, which means a great majority of the entries is close to zero. Without eigenvector compression, for the quantum many-body calculation, the state-of-the-art algorithms are in fact variants of power iterations. Hence in this paper, we compare coordinate-wise methods against power method as it is a simple baseline algorithm. Due to the nature of the problem, it is anticipated that CDMs will have much better performance over the traditional power method.

Three quantities are used to measure the convergence of the methods: the square root of the relative error of the objective function value (3.1), the relative error of the eigenvalue and the tangent of the angle between $x^{(\ell)}$ and the eigenvector. The square root of the relative error of the objective function value is defined as

$$\epsilon_{\text{obj}} = \sqrt{\frac{f\left(x^{(\ell)}\right) - f^*}{f^*}}, \tag{5.3}$$

where f^* denotes the minimum of $f(x)$. Since f^* is on the scale of λ_1^2, we use the square root of the relative error of the objective function which is on the same scale as λ_1. To estimate the eigenvalue, we use

$$E = \frac{x_*^\top A x^{(\ell)}}{x_*^\top x^{(\ell)}}, \tag{5.4}$$

where x_* is the reference vector that overlaps with the eigenvector v_1. In our test, x_* is a unit vector and will be specified later. This estimator is referred as the projected energy in the context of quantum chemistry. Thus the relative error is define by

$$\epsilon_{\text{energy}} = \frac{|E - \lambda_1|}{|\lambda_1|}. \tag{5.5}$$

Last,

$$\epsilon_{\tan} = \tan\theta(v^{(\ell)}, v_1) \tag{5.6}$$

is used to measure the convergence of the eigenvector.

The number of matrix column evaluation is used to measure the efficiency of the methods. There are several reasons why the number of matrix column evaluation is a good measure. First it is independent of the computation environment. Second we assume the evaluation of matrix column is the most expensive step. It is usually true in quantum many-body calculation in chemistry (full configuration interaction method) because the matrix is too large to be stored, the evaluation has to be on-th-fly and the evaluation of each entry is relatively expensive. Third, it can be used to compare with other related methods.

All methods are implemented and tested in MATLAB R2017b. The exact eigenvalue and eigenvector of each problem are calculated by the "*eigs*" function in Matlab with high precision.

Table 5.5: Performance of various CDMs for A_{108}

Method	k	Min Iter	Med Iter	Max Iter	Total Col Access
PM	5000	-	135	-	675000
CPM	1	-	309085	-	309085
CPM	4	-	75829	-	303316
CPM	16	-	18809	-	300944
SI-GSL(λ_1)	1	-	260000	-	260000
SI-GSL($1.5\lambda_1$)	1	-	1002283	-	1002283
Grad-vecLS	5000	-	80	-	400000
GCD-Grad-LS	1	-	109751	-	109751
GCD-LS-LS	1	-	100464	-	100464
SCD-Grad-vecLS(0)	4	86054	94837	128467	379348
SCD-Grad-vecLS(0)	16	21409	24066	30850	385056
SCD-Grad-vecLS(1)	4	36081	40866	52279	163464
SCD-Grad-vecLS(1)	16	9225	10205	11934	163280
SCD-Grad-vecLS(2)	4	29867	33538	45480	134152
SCD-Grad-vecLS(2)	16	7417	8334	11410	133344
SCD-Grad-LS(0)	1	342945	377783	483842	377783
SCD-Grad-LS(0)	4	86241	96074	150090	384296
SCD-Grad-LS(0)	16	21952	24487	31313	391792
SCD-Grad-LS(1)	1	146161	166415	223471	166415
SCD-Grad-LS(1)	4	38034	41773	53257	167092
SCD-Grad-LS(1)	16	9207	10479	15063	167664
SCD-Grad-LS(2)	1	120411	136468	177161	136468
SCD-Grad-LS(2)	4	30200	34091	44993	136364
SCD-Grad-LS(2)	16	7583	8631	11415	138096

5.2.1 Dense random matrices

We first show the advantage of the coordinate-wise descent methods over power method on dense random matrices. All matrices tested in this section involve symmetric matrices of size 5000 with random eigenvectors, i.e.,

$$A_{\lambda_1} = Q \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_{5000} \end{bmatrix} Q^\top, \tag{5.7}$$

where $\lambda_2, \ldots, \lambda_{5000}$ are equally distributed on $[1, 100)$ and Q is a random unitary matrix generated by a QR factorization of a random matrix with each entry being Gaussian random number. We use λ_1 to control the difficulty of the problem. In particular, we tested three matrices A_{108}, A_{101}, and $A_{108} + 1000I$. The last one is a shifted version of A_{108}. It is obvious that $A_{108} + 1000I$ is more difficult than A_{108} for power methods. For the optimization problem (3.1), it is also more difficult since the landscape of the objective function becomes steeper. While, in practice, CDMs for the optimization problem is less sensitive to the shift. In this section, all results are reported in terms of number of column accesses, converging to 10^{-6} under the measure of ϵ_{obj}. For non-stochastic methods, we report the number of column accesses from a single run and reported in the column named "Med Iter". While, for stochastic methods, we report the minimum, medium, and maximum numbers of iterations from 100 runs. In all tables, the column named "Total Col Access" provides the total number of matrix column evaluation, which is calculated as the product of number of coordinates updating each iteration, i.e., k, and the medium number of iterations. The initial vector of CDM is always chosen to be the unit vector e_1, with 1 in the first coordinate and 0 in all others.

Among the methods compared, PM and Grad-vecLS with $k = 5000$ update all coordinates per iteration. All other methods are coordinate-wise methods and all of them show speedup. PM converges the slowest. Although the iteration number is small, the cost of each iteration is huge. CPM reduces the cost by half on average, which is a big improvement. Grad-vecLS with $k = 5000$ is equivalent to gradient descent with exact line search. Since it updates at the full gradient direction, the cost is also large. SI-GSL(λ_1) and SI-GSL($1.5\lambda_1$) adopt λ_1 and $1.5\lambda_1$ as the estimation of the leading eigenvalue respectively, and other parameters in SI-GSL are set to be the underlying true values. SI-GSL(λ_1) in general performs compariable to GCD-Grad-

Table 5.6: Performance of various CDMs for A_{101}

Method	k	Min Iter	Med Iter	Max Iter	Total Col Access
PM	5000	-	839	-	4195000
CPM	1	-	1899296	-	1899296
CPM	4	-	471779	-	1887116
CPM	16	-	125525	-	2008400
SI-GSL(λ_1)	1	-	460000	-	460000
SI-GSL($1.5\lambda_1$)	1	-	3972548	-	3972548
Grad-vecLS	5000	-	480	-	2400000
GCD-Grad-LS	1	-	726093	-	726093
GCD-LS-LS	1	-	554521	-	554521
SCD-Grad-vecLS(0)	4	444954	557394	788291	2229576
SCD-Grad-vecLS(0)	16	118259	140177	178029	2242832
SCD-Grad-vecLS(1)	4	207804	253035	389396	1012140
SCD-Grad-vecLS(1)	16	53497	62959	84697	1007344
SCD-Grad-vecLS(2)	4	173455	206857	367199	827428
SCD-Grad-vecLS(2)	16	43618	51701	82524	827216
SCD-Grad-LS(0)	1	1896835	2208812	3279223	2208812
SCD-Grad-LS(0)	4	483657	571571	832675	2286284
SCD-Grad-LS(0)	16	119192	139788	203427	2236608
SCD-Grad-LS(1)	1	871362	1022169	1481621	1022169
SCD-Grad-LS(1)	4	206500	262616	402321	1050464
SCD-Grad-LS(1)	16	54106	63607	94236	1017712
SCD-Grad-LS(2)	1	717007	820728	1205225	820728
SCD-Grad-LS(2)	4	176141	209155	288910	836620
SCD-Grad-LS(2)	16	43237	51768	75115	828288

LS and GCD-LS-LS, while SI-GSL($1.5\lambda_1$) is siganificantly slower than all optimization based CDMs. The performance of SI-GSL sensitively depends on the provided a priori estimation of the leading eigenvalue, which makes it impractical for our target application settings. The greedy methods GCD-Grad-LS and GCD-LS-LS speed up significantly. They are about six times faster than PM for A_{108} and A_{101}. While, for $A_{108} + 1000I$, the greedy methods is about 70 times faster. The two greedy methods themselves have similar cost. For the stochastic methods, we use gradient information to pick the coordinates and use vecLS or LS to update. Different probability power t and number of coordinates updated k are tested. With the same parameters, SCD-

Table 5.7: Performance of various CDMs for $A_{108} + 1000I$

Method	k	Min Iter	Med Iter	Max Iter	Total Col Access
PM	5000	-	1214	-	6070000
CPM	1	-	3338202	-	3338202
CPM	4	-	853724	-	3414896
CPM	16	-	214645	-	3434320
SI-GSL(λ_1)	1	-	140000	-	140000
SI-GSL($1.5\lambda_1$)	1	-	1409635	-	1409635
Grad-vecLS	5000	-	875	-	4375000
GCD-Grad-LS	1	-	92532	-	92532
GCD-LS-LS	1	-	102098	-	102098
SCD-Grad-vecLS(0)	4	78212	92682	126872	370728
SCD-Grad-vecLS(0)	16	18632	21257	30375	340112
SCD-Grad-vecLS(1)	4	31107	34023	48125	136092
SCD-Grad-vecLS(1)	16	7888	8570	10915	137120
SCD-Grad-vecLS(2)	4	25047	27733	37547	110932
SCD-Grad-vecLS(2)	16	6111	7000	9788	112000
SCD-Grad-LS(0)	1	336724	404492	486490	404492
SCD-Grad-LS(0)	4	88807	100736	127954	402944
SCD-Grad-LS(0)	16	22687	25715	34403	411440
SCD-Grad-LS(1)	1	125623	139462	192680	139462
SCD-Grad-LS(1)	4	31205	34731	54808	138924
SCD-Grad-LS(1)	16	8081	9524	13275	152384
SCD-Grad-LS(2)	1	101177	110757	147548	110757
SCD-Grad-LS(2)	4	25399	31354	52048	125416
SCD-Grad-LS(2)	16	do not converge			

Grad-vecLS and SCD-Grad-LS share the similar performance. For $t = 0$, which is equivalent to uniform sampling, the number of column accesses are almost the same as the steepest gradient descent. It is reasonable since uniform sampling does not pick the important coordinate and should behave similar with the full gradient. For $t = 1$ and $t = 2$, the speedup is obvious. The performance for different t agrees with Theorem 3.5.7, since larger t shows faster convergence. Another thing to notice is that the cost is not influenced by the different choices of k for CPM and SCD. Therefore if we do the real asynchronized version, more speedup can be gained. Moreover, for stochastic CDMs, the variance is not large.

Comparing the performance for A_{101} and A_{108}, the numbers of column accesses for A_{101} are approximately six times as many as for A_{108} for all methods. This behavior is anticipated. Comparing A_{108} to $A_{108} + 1000I$, we see that PM, CPM and Grad-vecLS slow down significantly by the shift, since the condition number ($\frac{\lambda_1}{\lambda_1 - \lambda_2}$ for PM and the condition number of the Hessian for gradient descent) increases. However, all SI-GSL, GCD and SCD methods converge at almost the same speed, or even a little faster.

5.2.2 Hubbard models

In this test, we calculate the ground state energy and ground state of the 2D Hubbard model. The fermion Hubbard model is the same as introduced in Section 5.1.1.

To study the performance of methods, we test several 2D Hubbard models with different sizes. Here we report the results of two Hubbard models: both of them are on the 4×4 lattice with periodic boundary condition. The hopping strength is $t = 1$ and the interaction strength is $U = 4$. The first example contains 6 electrons, with 3 spin up and 3 spin down, whereas the second example contains 10 electrons, with 5 spin up and 5 spin down. The properties of the Hamiltonians are summarized in Table 5.8. The nonzero off-diagonal entries of H take value $\pm\frac{U}{N^{\text{orb}}} = \pm 0.25$, and the diagonal entries distribute in $(-20, 30)$ with bell-like shape. From the table we see that the Hamiltonians are indeed sparse. The smallest eigenvalue (Eigenvalues Min) is the ground energy we are to compute, thus the difference between the smallest one and the second smallest one is the eigengap. Since the spectra of H of the two examples live in the interval $(-100, 100)$, the matrix $A = 100I - H$ is used for calculation such that the smallest eigenvalue of H becomes the largest one of A and A is positive definite. Both the initial vector and the reference vector of the energy estimator are chosen to be $10\, e_{\text{HF}}$ for all calculation, this amounts to an initial guess of eigenvector e_{HF} with eigenvalue $10^2 = 100$ on the same scale as the diagonal of A (due to the $100I$ shift). Notice that the reference vector has only one nonzero entry, so computing the projected energy is cheap.

Table 5.8: Properties of the 2D Hubbard Hamiltonian for numerical comparison of CDMs

H	Dim	nnz per col			Eigenvalues		
		Min	Med	Max	Min	2nd Min	Max
6 electrons	1.96×10^4	100	102	112	-14.90	-14.55	20.26
10 electrons	1.19×10^6	196	202	240	-19.58	-17.08	32.73

For the first example, the 4 × 4 Hubbard model with 6 electrons, we run the methods until the relative error of the objective function value $e_{\mathrm{obj}} < 10^{-6}$. The result is shown in Table 5.9.

Table 5.9: Performance of various CDMs for 4 × 4 Hubbard model with 6 electrons

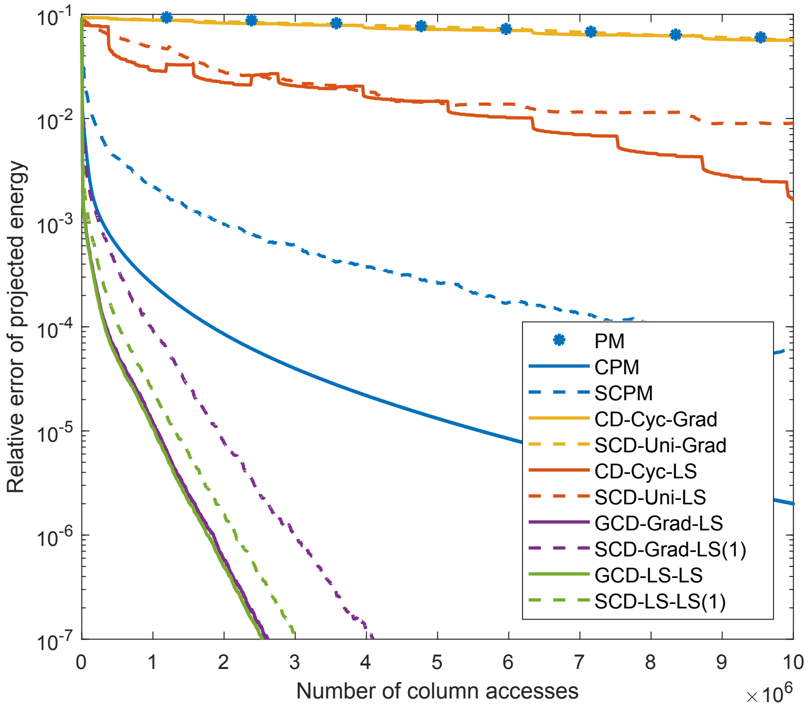
Method	k	Min Iter	Med Iter	Max Iter	Total Col Access
PM	19600	-	2255	-	44198000
CPM	1	-	456503	-	456503
CPM	16	-	29293	-	468688
Grad-vecLS	19600	-	361	-	7075600
GCD-Grad-LS	1	-	31997	-	31997
GCD-LS-LS	1	-	30996	-	30996
SCD-Grad-vecLS(1)	4	36966	38763	40224	155052
SCD-Grad-vecLS(1)	16	11490	12115	12644	193840
SCD-Grad-vecLS(1)	32	6283	6642	6991	212544
SCD-Grad-vecLS(2)	4	18934	19260	19613	77040
SCD-Grad-vecLS(2)	16	6438	6547	6652	104752
SCD-Grad-vecLS(2)	32	3880	3946	4002	126272
SCD-Grad-LS(1)	1	117261	120613	123744	120613
SCD-Grad-LS(1)	2	58536	60470	62209	120940
SCD-Grad-LS(1)	4	28716	30152	30918	120608
SCD-Grad-LS(1)	8	13053	15352	20197	122816
SCD-Grad-LS(2)	1	47603	48136	48802	48136
SCD-Grad-LS(2)	2	23789	24059	24399	48118
SCD-Grad-LS(2)	4	do not converge			
SCD-Grad-LS(2)	8	do not converge			

In Table 5.9, the second column k is the number of coordinates updated in one iteration. Each stochastic method runs for 100 times and we report the number of iterations and matrix column accesses, similar to the previous tests.

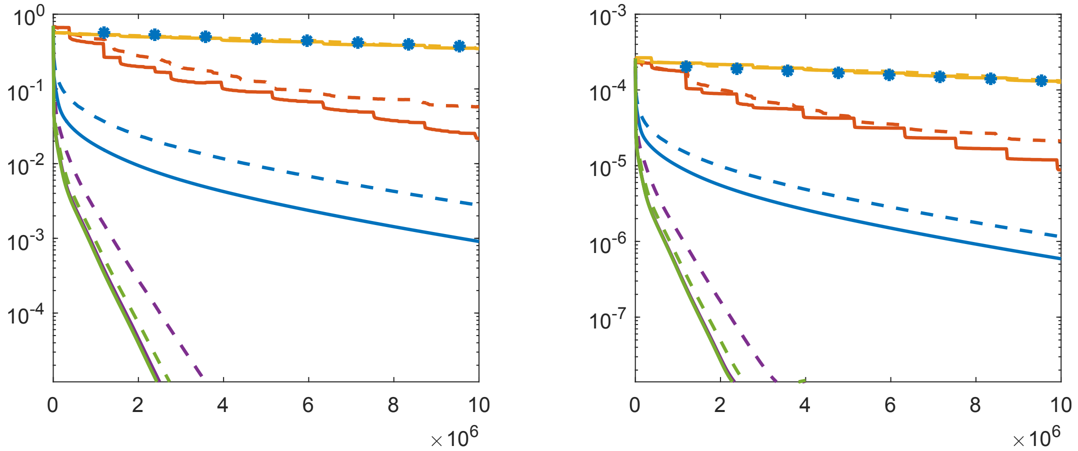
Since the dimension of the Hubbard model is larger and the eigenvector v_1 is sparser than that in the first dense matrix example, the advantage of the coordinate-wise methods is more significant. They are 10^2 to 10^3 faster than PM. GCD-Grad-LS and GCD-LS-LS are still the fastest one. Stochastic CDMs also outperform PM and CPM. Comparing with the greedy CDMs, they are about 3-5 times slower, which is more than that in the previous tests.

For the second model with 10 electrons in the 4×4 grid, we also compare with other methods, such as SCPM (stochastic version of CPM), SCD-Uni-Grad (which samples coordinate uniformly and updates the coordinate as the gradient multiplying a fixed stepsize), CD-Cyc-LS (which picks the coordinate in a cyclic way and updates the coordinate using line search), and SCD-Uni-LS (i.e., SCD-Grad-LS(0)). The step size, if used, is chosen to be 2. This step size is larger than that in Theorem 3.3.1, but numerically the iteration converges. All methods besides PM update single coordinate per iteration and running up to 10^7 column accesses or desired accuracy. Stochastic CDMs sample coordinate with respect to the magnitude of the gradient, i.e., $t = 1$. The convergence of the square root of the relative error of objective function value ϵ_{obj}, eigenvalue estimator ϵ_{energy} and eigenvector estimator $\epsilon_{\tan}$ are plotted in Figure 5.5.

From Figure 5.5, we see that the convergence of the three error measures share a similar pattern, thus the convergence of one quantity should imply the convergence of the other two in practice. The convergence of all methods tested consist of one or two stages: a possible fast decay followed by linear convergence. We know that both power method (PM) and gradient descent (Grad) converge linearly. In this figure there are only 8 iterates of PM since each iteration needs the full access of



(a) ϵ_{energy}, Relative error of projected energy



(b) $\epsilon_{\text{tangent}}$, Tangent of angle between $v^{(\ell)}$ and v_1

(c) ϵ_{obj}, Relative error of objective function value

Figure 5.5: Convergence behavior of CDMs for the 4×4 Hubbard model with 10 electrons. $k = 1$ for all methods except PM.

the matrix. CD-Cyc-Grad and SCD-Uni-Grad seem to behave similarly with slow linear decay, which is reasonable since they treat each coordinate equally and should behave similarly as the gradient descent method. CD-Cyc-LS and SCD-Uni-LS also decay linearly, but with a faster rate. This is because updating coordinates by Grad uses the same step size for all coordinates and all iterates, which has to be small thus not optimal. LS always gives the fastest local decay for different coordinates and different iterates. This argument can be confirmed by the sudden-decay-behavior of CD-Cyc-LS. When CD-Cyc-LS updates the important coordinate, it will give a large 'step size' and the error drops down rapidly.

All the other methods decay very fast at the beginning, and the common feature is that they pick the coordinates for updating according to their importance, instead of treating them equally. This justifies the motivation of coordinate descent methods. Within 10^5 column accesses, equivalent to $^1/_{12}$ iteration of PM, the projected energy reaches 10^{-2} to 10^{-4} accuracy. This is really amazing, and agrees with the result of the previous examples. GCD-Grad-LS and GCD-LS-LS behave almost the same and they converge fastest both at the initial stage and the linear convergence stage. The projected energy reaches 10^{-8} accuracy when PM only iterates 3 times. SCD-LS-LS and SCD-Grad-LS also converge fast. The convergence rates of CPM and SCPM are not as fast as SCD-LS-LS and SCG-Grad-LS, but they are still much better than PM.

Comparing between the stochastic CDMs and the greedy CDMs, greedy methods always converge a little faster, such as CD-Cyc-LS versus SCD-Uni-LS and GCD-Grad-LS versus SCD-Grad-LS. This agrees with our theoretical results and previous examples. If more than 1 coordinate is updated at each iteration, greedy CDMs often get stuck but stochastic CDMs are much less likely. Another thing to mention is that in a small neighborhood of the minimum, updating by LS is almost the same as updating by Grad with some step size depending on the Hessian of the minimum. Thus, updating by Grad could perform as well as LS ideally if there exists a step size which both guarantees the convergence and also converges fast. On the other hand size, LS choose a optimal step size every iteration.

5.3 Benchmark of CDFCI

In this section, we perform a sequence of numerical experiments to demonstrate the efficiency of CDFCI. First, we compare the performance of CDFCI, Heat-bath CI (HCI), DMRG and iS-FCIQMC (FCIQMC with initiator and semi-stochastic adaptation) on H_2O, C_2 and N_2 under cc-pVDZ basis. Then, we benchmark the binding curve of nitrogen dimer under cc-pVDZ basis using CDFCI up to 10^{-3} mHa accuracy. Finally, we use CDFCI to calculate the ground state energy of chromium dimer Cr_2 under the Ahlrichs VDZ Basis at $r = 1.5$Å, which is a well-known challenging task due to the strong correlation.

In all experiments, the orbitals and integrals are calculated via restricted Hartree Fock (RHF) in Psi4[67] package. All the reported energies are in Hartree (Ha) but the length unit is in either Bohr radius (a_0) or ångström (Å) due to different configurations in the references.

5.3.1 Numerical results of H_2O, C_2, and N_2

We first compare the performance of CDFCI with other algorithms, HCI, DMRG and iS-FCIQMC. In this paper, we choose iS-FCIQMC instead of FCIQMC or iFCIQMC because it balances well among bias, variance and runtime. CDFCI is implemented as stated in Section 4.3 with the on-the-fly Hamiltonian elements evaluation interfaced from Hande-QMC[85, 84]. HCI adopts the original implementation in Dice [34]; DMRG adopts the widely used implementation in Block[12, 11, 24, 78, 63]; iS-

FCIQMC adopts the implementation in NECI code [7]. All programs are compiled by Intel compiler 19.0.144 with `-O3` option. MPI and OpenMP support are disabled for all programs in this section. All the tests in this section are produced on a machine with Intel Xeon CPU E5-1650 v2 @ 3.50GHz and 64GB memory. For all algorithms, the Hartree-Fock state is used as the initial wavefunction. Most parameters in CDFCI, HCI, DMRG, and iS-FCIQMC will be clearly stated in the later content. We run HCI with different truncation threshold ε_1, DMRG with different maximum bond dimension $\max M$ and iS-FCIQMC with different target population m. Any unspecified parameter is set to be the default value. Besides the specified or default parameters, for iS-FCIQMC, the time step is optimized by NECI; the initiator truncation threshold is 3 and the size of the deterministic space is 1000. For all the tests reported in Table 5.11, Table 5.12, and Table 5.13, iS-FCIQMC runs for 10000 steps and the energy is estimated by the block analysis in NECI . In all algorithms, the energy is reported without any perturbation or extrapolation post-calculation. Variational energy (Rayleigh quotient) is reported for CDFCI, HCI and DMRG, while average projected energy is reported for iS-FCIQMC. Since iS-FCIQMC is a stochastic method, we also report one significant digit of the Standard Error of the Mean (SEM) in the parenthesis and the SEM is of the same accuracy level as the last digit of the average. SEM is estimated as the sample standard deviation divided by the square root of the sample size. Also Root Mean Square Error (RMSE) of the average estimator is adopted as the error of iS-FCIQMC, which is defined as $\sqrt{\text{standard error}^2 + \text{bias}^2}$. We emphasize that similar perturbation calculation as in HCI can also be applied to CDFCI and the comparison is left as future work.

In this section, we test the four algorithms on three molecules: H_2O with OH bond length $1.84345a_0$ and HOH bond angle 110.6° [65, 8], C_2 with bond length 1.24253Å [34, 77] and N_2 with bond length $2.118a_0$ [13]. The properties of the systems are summarized in Table 5.10, where the reference ground state energy is calculated by CDFCI to a high precision.

Table 5.10: Properties of test molecule systems for CDFCI. HF energy and GS energy denote Hartree-Fock energy and ground state energy respectively.

Molecules	Basis	Electrons	Orbitals	Dimension	HF Energy	GS Energy
H_2O	cc-pVDZ	10	24	4.53×10^8	-76.0240386	-76.2418601
C_2	cc-pVDZ	12	28	1.77×10^{10}	-75.4168820	-75.7319604
N_2	cc-pVDZ	14	28	1.75×10^{11}	-108.9493779	-109.2821727

H_2O molecule

Table 5.11 and Figure 5.6 illustrate numerical results for H_2O. In Table 5.11, we report the detailed results and the corresponding used parameters. For CDFCI, we run the test once and report the wall time to reach the accuracy. For other tests, each row corresponds to one test. We also report the wall clock time for CDFCI to reach the same accuracy as well as the ratio of the wall time of the method over the wall time of CDFCI in the last two columns. iS-FCIQMC runs for 10000 iterations and reports the average projected energy and the standard error (in parenthesis) through block analysis in the energy column. RMSE is reported as the error of iS-FCIQMC. Figure 5.6 plots the convergence of the energy against the wall-clock time based on the results in Table 5.11. For iS-FCIQMC, we run another test with $m = 50000$ for longer time and plot the curve of the projected energy as well as the cumulative average of energy starting at iteration 5000.

From Table 5.11 and Figure 5.6, we shall see that all algorithms reach chemical accuracy efficiently. CDFCI has a good performance in general. The energy drops quickly to the level of 0.1 mHa accuracy at the beginning. Then it has a slower but steady linear decay. This behavior proves the rationale behind CDFCI: contributions of different determinants to the FCI energy vary a lot, especially in the early stage

Table 5.11: Convergence of ground state energy of H_2O to benchmark CDFCI.

Algorithm	Parameter	Energy	Error	Time(s)	CDFCI Time(s)	CDFCI Ratio
CDFCI	$\varepsilon = 0$	-76.2318601	1.0×10^{-2}	3.7	-	
		-76.2408601	1.0×10^{-3}	96.2	-	
		-76.2417601	1.0×10^{-4}	592.5	-	
		-76.2418501	1.0×10^{-5}	2780.0	-	
		-76.2418591	1.0×10^{-6}	9569.5	-	
		-76.2418600	1.0×10^{-7}	25227.5	-	
		-76.2418601	1.0×10^{-8}	54242.2	-	
HCI	$\varepsilon_1 = 1.0 \times 10^{-4}$	-76.2412891	5.7×10^{-4}	58.4	156.3	0.37x
	$\varepsilon_1 = 2.0 \times 10^{-5}$	-76.2417533	1.1×10^{-4}	312.9	565.3	0.55x
	$\varepsilon_1 = 1.0 \times 10^{-5}$	-76.2418109	4.9×10^{-5}	593.5	993.1	0.60x
	$\varepsilon_1 = 5.0 \times 10^{-6}$	-76.2418402	2.0×10^{-5}	1148.3	1823.2	0.63x
DMRG	$\max M = 500$	-76.2418170	4.3×10^{-5}	1731	1089.7	1.59x
	$\max M = 1000$	-76.2418557	4.4×10^{-6}	5224	4435.7	1.18x
	$\max M = 2000$	-76.2418596	4.5×10^{-7}	17839	13802.6	1.29x
	$\max M = 4000$	-76.2418599	1.7×10^{-7}	77023	20585.9	3.74x
iS-FCIQMC	$m = 10000$	-76.2418(3)	2.7×10^{-4}	222.4	277.2	0.80x
	$m = 50000$	-76.24197(8)	1.4×10^{-4}	1009.0	470.5	2.14x
	$m = 100000$	-76.24181(5)	7.1×10^{-5}	1942.8	762.1	2.55x
	$m = 500000$	-76.24181(3)	5.9×10^{-5}	9074.3	875.2	10.37x

of iterations. Since CDFCI always updates the “best” determinant at each iteration, it is able to reach high accuracy with only a few iterations.

For small molecules like H_2O, HCI is also an excellent algorithm and costs less time to achieve the same accuracy comparing to CDFCI. It shows that the determinant selecting strategy used in HCI, which relies on decaying property of Hamiltonian entries, is also quite efficient for molecules. For example, when $\varepsilon_1 = 5.0 \times 10^{-6}$, HCI uses only 3006594 determinants to reach 2.0×10^{-5} Ha accuracy, whereas CDFCI uses 1823176 determinants, which is about 60% determinants used by HCI to achieve the same accuracy.

The speedup of HCI over CDFCI is due to the different implementation strate-

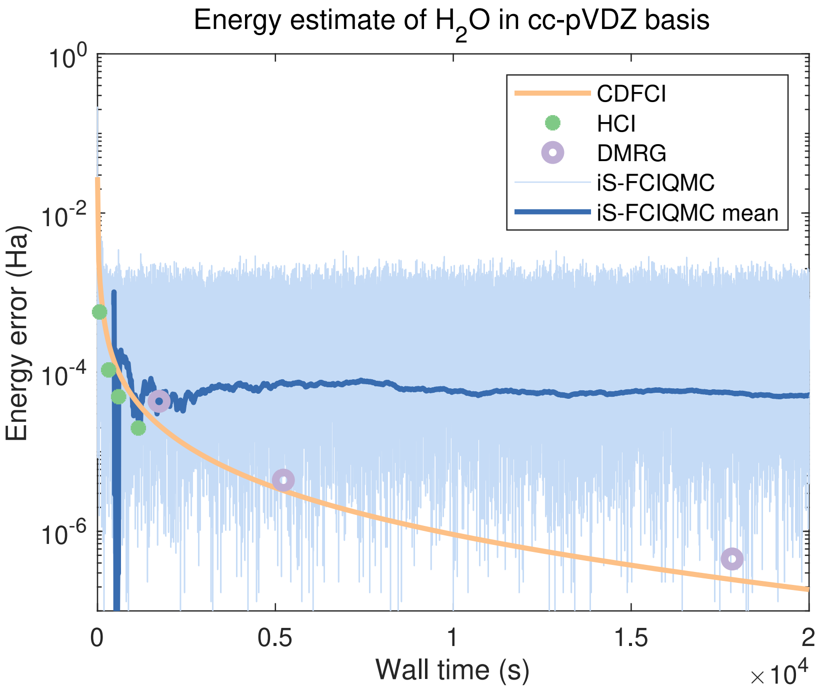


Figure 5.6: Convergence of ground state energy of H_2O against time to benchmark CDFCI. Each point or curve represents one test as in Table 5.11.

gies of the algorithms. The implementation of HCI in DICE stores the submatrix of the Hamiltonian with respect to the selected determinants in the main memory, and reuses them for inner Davidson iterations. Both the submatrix and the vector are stored and accessed in contiguous memory. Therefore, two advantages of the implementation come into play: one-time evaluation of Hamiltonian entries and efficient usage of memory hierarchy. However, the disadvantage is also obvious: huge memory cost for the submatrix. In Table 5.11 and Figure 5.6, we do not report results for smaller ε_1 because DICE reaches the memory limit. The high memory cost is also the reason why the variational stage of HCI does not perform good for chromium dimer (see Section 5.3.3). As a comparison, CDFCI uses a different philosophy in the implementation. CDFCI calculates the Hamiltonian entries on-the-fly, which saves all memory for the coefficient vector, and stores the coefficients in a hash table. Hence much more coefficients can be used to represent the ground state but paying the cost of repeated evaluation of Hamiltonian entries and limited usage of memory hierarchy.

DMRG also achieves high accuracy in reasonable time with small memory cost. But it is always slower than CDFCI and HCI for H_2O.

iS-FCIQMC is very efficient for small number of walkers and iterations. It is able to achieve reasonable accuracy in a short time. In Figure 5.6 we see the convergence behavior of iS-FCIQMC projected energy. It can reach accuracy level of 1 mHa very efficiently, but hard to converge to higher accuracy due to the slow convergence of Monte Carlo and the bias introduced by the initiator approximation. It could be possible to use more walkers to reduce the variance and bias. However, as shown in Table 5.11, moderate increase of the amount of walkers does not change the convergence behavior.

Carbon dimer and nitrogen dimer

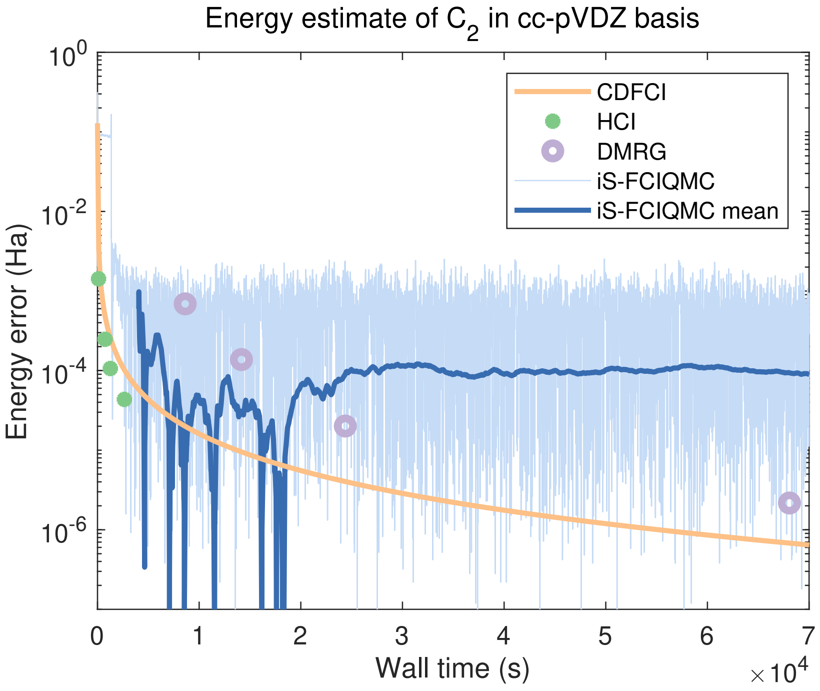


Figure 5.7: Convergence of ground state energy of C_2 against time to benchmark CDFCI. Each point or curve represents one test as in Table 5.12.

Table 5.12: Convergence of ground state energy of C_2 to benchmark CDFCI.

Algorithm	Parameter	Energy	Error	Time(s)	CDFCI Time(s)	CDFCI Ratio
CDFCI	$\varepsilon = 3.0 \times 10^{-8}$	-75.7219604	1.0×10^{-2}	49.0	-	
		-75.7309604	1.0×10^{-3}	388.2	-	
		-75.7318604	1.0×10^{-4}	2687.3	-	
		-75.7319504	1.0×10^{-5}	13717.6	-	
		-75.7319594	1.0×10^{-6}	55210.2	-	
HCI	$\varepsilon_1 = 1.0 \times 10^{-4}$	-75.7305361	1.4×10^{-3}	100.9	277.8	0.36x
	$\varepsilon_1 = 2.0 \times 10^{-5}$	-75.7317130	2.5×10^{-4}	745.0	1319.1	0.56x
	$\varepsilon_1 = 1.0 \times 10^{-5}$	-75.7318541	1.1×10^{-4}	1261.8	2565.2	0.49x
	$\varepsilon_1 = 5.0 \times 10^{-6}$	-75.7319170	4.4×10^{-5}	2644.3	4989.8	0.53x
DMRG	$\max M = 500$	-75.7312704	6.9×10^{-4}	8624	544.4	15.84x
	$\max M = 1000$	-75.7318227	1.4×10^{-4}	14163	2102.9	6.73x
	$\max M = 2000$	-75.7319403	2.0×10^{-5}	24377	8582.9	2.84x
	$\max M = 4000$	-75.7319583	2.2×10^{-6}	68071	35435	1.92x
iS-FCIQMC	$m = 10000$	-75.729(1)	2.9×10^{-3}	229.5	140.8	1.63x
	$m = 50000$	-75.7301(5)	1.9×10^{-3}	1041.4	212.9	4.89x
	$m = 100000$	-75.7314(4)	7.0×10^{-4}	2038.2	539.8	3.78x
	$m = 500000$	-75.7320(1)	1.5×10^{-4}	9604.2	2003.2	4.79x
	$m = 1000000$	-75.7318(1)	2.1×10^{-4}	18644.8	1497.4	12.45x

Carbon dimer C_2 and nitrogen dimer N_2 are more challenging than H_2O molecule since their correlation is stronger and the dimension N_{FCI} is higher. The results of C_2 are reported in Table 5.12 and Figure 5.7; and the results of N_2 are reported in Table 5.13 and Figure 5.8. For iS-FCIQMC, the projected energy and its cumulative average from iteration 5000 are plotted with target population $m = 500000$ in for both C_2 and N_2 in Figure 5.7 and Figure 5.8. And in Table 5.12 and Table 5.13, iS-FCIQMC runs for 10000 iterations and reports the average projected energy and standard error (in parenthesis) through the block analysis in the energy column. RMSE is reported as the error of iS-FCIQMC.

In general, C_2 costs more time than H_2O to converge to a fixed accuracy and N_2 costs more time than C_2, which agrees with their system complexities.

Table 5.13: Convergence of ground state energy of N_2 to benchmark CDFCI.

Algorithm	Parameter	Energy	Error	Time(s)	CDFCI Time(s)	 Ratio
CDFCI	$\varepsilon = 5.0 \times 10^{-7}$	-109.2721727	1.0×10^{-2}	33.4	-	
		-109.2811727	1.0×10^{-3}	752.6	-	
		-109.2820727	1.0×10^{-4}	7892.6	-	
		-109.2821627	1.0×10^{-5}	49862.6	-	
HCI	$\varepsilon_1 = 1.0 \times 10^{-4}$	-109.2805259	1.7×10^{-3}	100.7	427.1	0.24x
	$\varepsilon_1 = 2.0 \times 10^{-5}$	-109.2817822	3.9×10^{-4}	730.9	2107.4	0.35x
	$\varepsilon_1 = 1.0 \times 10^{-5}$	-109.2819787	1.9×10^{-4}	1335.2	4266.8	0.31x
	$\varepsilon_1 = 5.0 \times 10^{-6}$	-109.2820857	8.7×10^{-5}	3330.0	8920.3	0.37x
DMRG	$\max M = 500$	-109.2809830	1.2×10^{-3}	9936	619.6	16.04x
	$\max M = 1000$	-109.2818757	3.0×10^{-4}	17647	2806.7	6.29x
	$\max M = 2000$	-109.2821098	6.3×10^{-5}	37549	11857.1	3.12x
	$\max M = 4000$	-109.2821632	9.5×10^{-6}	85703	51574.7	1.66x
iS-FCIQMC	$m = 10000$	-109.2818(4)	5.2×10^{-4}	235.7	1556.8	0.15x
	$m = 50000$	-109.2822(3)	2.6×10^{-4}	1068.9	3161.5	0.34x
	$m = 100000$	-109.2818(2)	4.0×10^{-4}	2090.5	2077.2	1.01x
	$m = 500000$	-109.2822(1)	1.2×10^{-4}	9839.3	6832.6	1.44x
	$m = 1000000$	-109.28214(5)	5.0×10^{-5}	18959.7	14510.0	1.31x

CDFCI shows similar convergence pattern for C_2 and N_2, with fast decay at the beginning followed by a slower but steady linear decay. It takes only several minutes to reach the chemical accuracy. Therefore, CDFCI is consistently efficient for different systems with different correlation strength.

HCI also shows similar convergence behavior for C_2 and N_2. It converges to chemical accuracy the fastest among tested algorithms. However, HCI can not converge to higher accuracy due to the memory limit of the implementation. Comparing to CDFCI in terms of the number of operations, however, CDFCI uses less operations and determinants than HCI to the same accuracy level, as in the case of H_2O.

DMRG also performs similar for C_2 and N_2 but is significantly slower than CDFCI and HCI. One reason is that DMRG needs more iterations to converge due to the

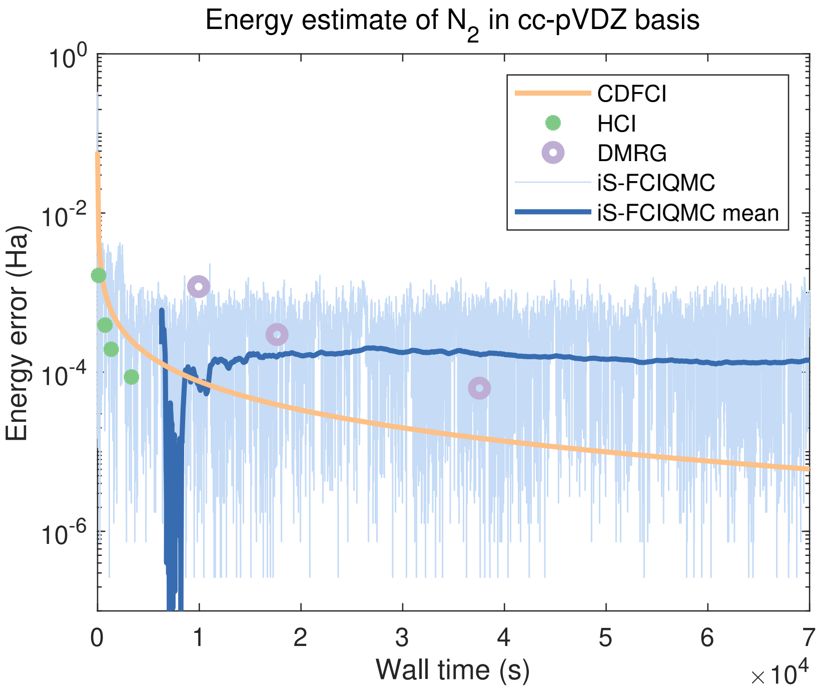


Figure 5.8: Convergence of ground state energy of N_2 against time to benchmark CDFCI. Each point or curve represents one test as in Table 5.13.

strong correlation. In this sense, determinant selecting algorithms are less affected by the correlation strength than DMRG.

iS-FCIQMC, however, has a different behavior for C_2 and N_2. While N_2 is significantly harder than C_2 for CDFCI, HCI and DMRG, iS-FCIQMC reaches a higher accuracy for N_2 within 10000 iterations, as shown by the data in Table 5.12 and Table 5.13. We point out that iS-FCIQMC performs quite well for N_2, as it can reach 10^{-4} Ha error in a short time with only $m = 10000$ or $m = 50000$ walkers, whereas CDFCI takes more time to converge since the dimension of N_2 is higher than H_2O and C_2. iS-FCIQMC seems to be less influenced by the increase of dimensionality.

In conclusion, as shown in both Section 5.3.1 and Section 5.3.1, CDFCI is efficient for both weakly-correlated and strongly-correlated systems. It can achieve chemical accuracy efficiently and is able to achieve higher accuracy in all tested molecules. HCI costs more operations than CDFCI to achieve the same accuracy but costs less time due to the different philosophies in implementations. Comparing to CDFCI and HCI, DMRG is less efficient for strongly-correlated systems. While similar as CDFCI, DMRG can also achieve much higher accuracy than the chemical accuracy. iS-FCIQMC is also efficient to reach chemical accuracy with a few walkers in short time for the testing molecules, but it may need much more time and walkers to reach higher accuracy.

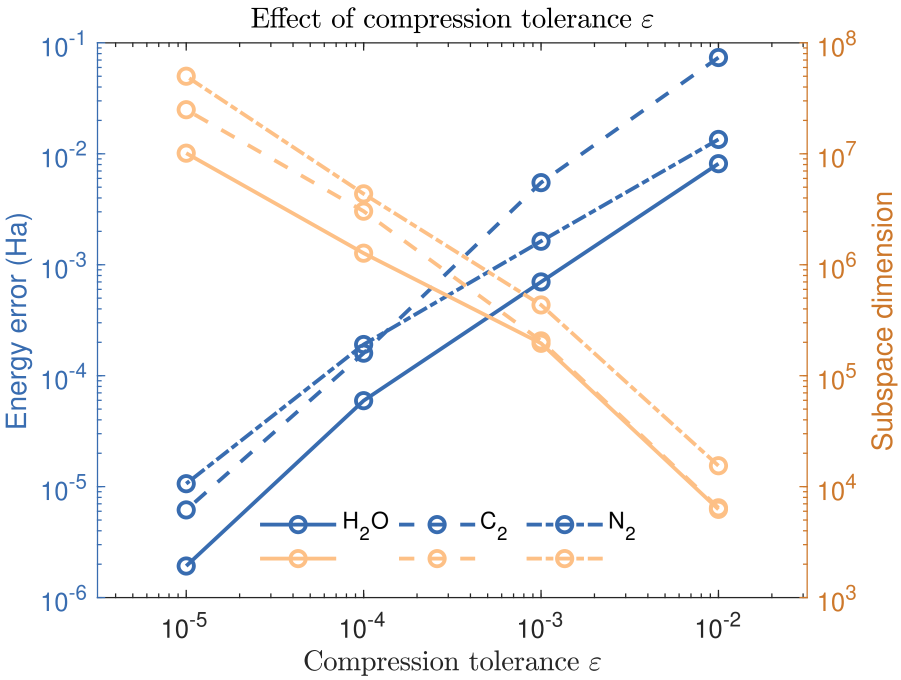


Figure 5.9: Error of energy and the number of nonzeros entries in the vector z after convergence for different compression tolerance ε in CDFCI.

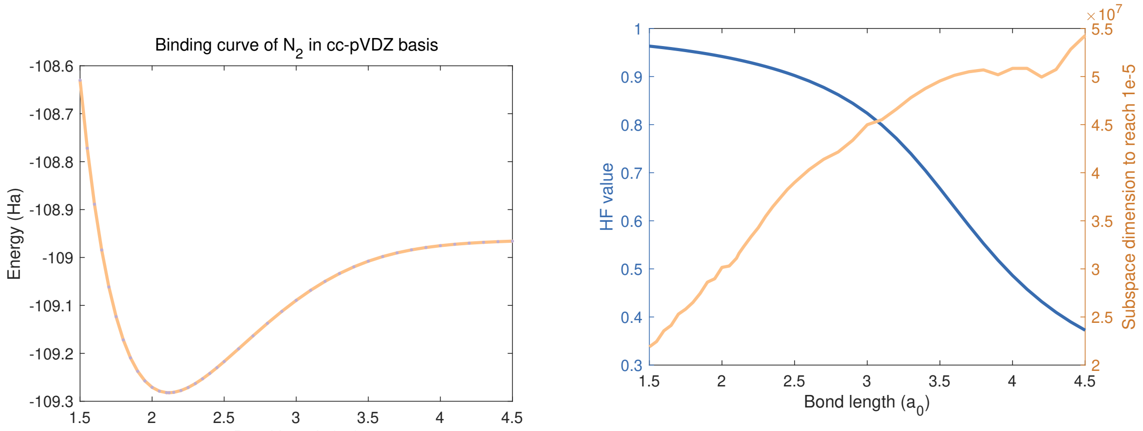
In terms of the usability, CDFCI, HCI, and DMRG only have one single parameter to be tuned, whereas iS-FCIQMC has more parameters. The proper parameter in CDFCI can be revealed in a few minutes, judging from whether the stabilized vector z properly utilizes the given amount of memory. By choosing ε, we can easily balance between accuracy and memory cost, as shown in Figure 5.9. We conclude that CDFCI is an easy-to-use efficient algorithm for FCI problems.

5.3.2 Binding curve of N_2

In this section, we benchmark the all-electron nitrogen binding curve using CDFCI under the Dunning's cc-pVDZ basis. The nitrogen binding curve is a well-known difficult problem. When the nitrogen atoms are stretched away from the equilibrium geometry, Hartree-Fock theory no longer gives a good approximation and the system becomes multi-referenced due to the triple bond between the atoms. DMRG and coupled cluster theory, e.g., CCSD, CCSD(T), CCSDT, etc., have been tested on this problem, but only on 6 geometry configurations [13]. Here we show that CDFCI is capable to efficiently benchmark the all-electron nitrogen binding curve on a very fine grid of bond length and the variational energy converges to at least 10^{-3} mHa accuracy in each configuration.

In this problem, there are 14 electrons and 28 orbitals, and the dimension of the FCI space is about $N_{\text{FCI}} \approx 1.75 \times 10^{11}$. In all configurations, $\varepsilon = 10^{-6}$ is used in CDFCI for truncation. Here we use the same computing environment as in Section 5.3.1 but with OpenMP enabled with 5 threads. Each configuration on the bind curve results take roughly one day to achieve the 10^{-3} mHa accuracy. Figure 5.10 shows the binding curve and Table 5.16 in Section 5.3.4 list all converged variational energies for every configuration in the figure. In Table 5.14, we compare selected results obtained from CDFCI with that from other algorithms reported in Ref. [13].

Several remarks are in order regarding the benchmark results. First, Figure 5.10 demonstrates a smooth standard shape binding curve. Different from the carbon binding curve [34], no jump is observed in our benchmark results, where D_{2h} symmetry is used for all configurations. Second, Table 5.14 shows that CDFCI gives the lowest energy. CDFCI energy is accurate beyond the level of 10^{-3} mHa whereas DMRG is accurate up to 10^{-2} mHa. The DMRG results in Table 5.14 are taken from previous work [13], which agree with the results obtained in Section 5.3.1. Other algo-



(a) Energy of N_2

(b) Hartree-Fock value $x_{\mathrm{HF}}/\|x\|$ and the number of nonzeros of x to reach 10^{-5} Ha energy error for CDFCI

Figure 5.10: Binding curve and the corresponding Hartree-Fock values and number of nonzeros in x of N_2 under cc-pVDZ basis computed by CDFCI.

rithms such as CCSDTQ are much less accurate. Third, the more the nitrogen dimer molecule is stretched from the equilibrium, the more determinants and iterations are needed for CDFCI to converge to 10^{-3} mHa accuracy. This is because Hartree-Fock theory only works well near equilibrium configuration. However, the number of determinants and iterations do not increase significantly, which shows the efficiency of CDFCI again. Other algorithms become less accurate for larger stretching distance.

5.3.3 All electron chromium dimer calculation

Chromium dimer is hard to compute due to its strong correlation. We calculate the all-electron molecule using CDFCI under the Ahlrichs VDZ basis with radius $r = 1.5$Å. There are 48 electrons and 42 orbitals, and the dimension of the FCI space is about 2×10^{22}. Many methods have been applied to this problem including DMRG [63] and HCI [34]. Table 5.15 summarizes all results, including our CDFCI results and others from literature [63, 34].

In this paper, we only consider variational ground state energy without any

Table 5.14: Nitrogen molecule ground state energy using CDFCI, DMRG (max M = 4000) and couple cluster theories. *Slant digits* indicate inaccurate digits.

Bond Length	2.118a_0	2.4a_0	2.7a_0
CDFCI	-109.282173	-109.241908	-109.163600
DMRG	-109.2821*57*	-109.2418*86*	-109.1635*72*
CCSD	-109.2*67626*	-109.2*19794*	-109.1*31491*
CCSDTQ	-109.281*943*	-109.241*321*	-109.16*2264*
MRCISD	-109.27*5356*	-109.2*34925*	-109.15*6473*
MRCCSD	-109.28*0646*	-109.24*0362*	-109.16*1969*
Bond Length	**3.0a_0**	**3.6a_0**	**4.2a_0**
CDFCI	-109.089405	-108.998083	-108.970132
DMRG	-109.0893*80*	-108.9980*52*	-108.9700*90*
CCSD	-109.0*52884*	-108.9*75885*	-108.9*60244*
CCSDTQ	-109.08*6502*	-108.99*3736*	-108.96*8124*
MRCISD	-109.08*2149*	-108.99*0759*	-108.9*63070*
MRCCSD	-109.08*7613*	-108.99*5885*	-108.96*7865*

Table 5.15: Energy of Cr$_2$ by CDFCI, DMRG, HCI and coupled cluster theories. CDFCI produces the state-of-the-art variational energy.

Algorithm	Energy (Ha)
HCI (variational)	−2086.*384*
CCSD(T)	−2086.4*22229*
CCSDTQ	−2086.4*30244*
DMRG (max M = 8000)	−2086.44*3334*
CDFCI	−2086.443*565*
HCI (perturbed)	−2086.444*04*
DMRG (extrapolated)	−2086.444784

perturbation or extrapolation. Regarding the variational ground energy, CDFCI achieves lowest energy among all algorithms in one month running time on a machine with Intel Xeon CPU E5-1650 v3 @ 3.50GHz and 128GB memory. Both DMRG ($\max M = 8000$) and CDFCI achieve the chemical accuracy if the energy of DMRG (extrapolated) is regarded as the ground truth. But HCI(variational) and coupled cluster theory cannot achieve chemical accuracy. HCI converges to -2086.384 Ha in about eight minutes [34], whereas CDFCI reaches the same accuracy in about twenty minutes, although different computing environments are used. As the dimension of the Hamiltonian becomes larger and the system becomes more correlated, HCI can only afford storing the submatrix in the main memory with very limited number of determinants and such a limited number cannot achieve higher accuracy in the variational phase. With perturbation phase enabled, HCI can achieve accuracy less than 1 mHa. Similar perturbation phase can be adapted to CDFCI to further boost the accuracy or extend the applicability of our algorithm to larger systems.

5.3.4 N_2 Binding Curve Data

Figure 5.10 plots the binding curve of nitrogen dimer in cc-pVDZ basis with data given in Table 5.16. The bond length of nitrogen dimer in equilibrium geometry is $2.118a_0$. Table 5.16 list variational energies of nitrogen dimer produced by CDFCI with bond lengths smaller and larger than $2.118a_0$ respectively. In both tables, CDFCI used $\varepsilon = 10^{-6}$ as the truncation threshold.

The bond lengths are selected through the following two steps. CDFCI first calculates energies for a vector of bond lengths linearly spaced between and including $1.50a_0$ and $4.50a_0$ with gap $0.10a_0$. Then, according to the initial rough binding curve, another vector of bond lengths is added to smooth out the curve. These added bond lengths are in the sharp changing range around the equilibrium setting.

Table 5.16: Energy of nitrogen dimer with different bond lengthsThe energy refers to variational ground state energy calculated by CDFCI with $\varepsilon = 10^{-6}$.

Bond length (a_0)	Energy (Ha)
1.50	−108.6300476
1.55	−108.7719968
1.60	−108.8888460
1.65	−108.9843136
1.70	−109.0615754
1.75	−109.1233484
1.80	−109.1719641
1.85	−109.2094264
1.90	−109.2374578
1.95	−109.2575411
2.00	−109.2709530
2.05	−109.2787896
2.10	−109.2819938
2.118	−109.2821727
2.118	−109.2821727
2.15	−109.2813737
2.20	−109.2776211
2.25	−109.2713283
2.30	−109.2630013
2.35	−109.2530718
2.40	−109.2419079
2.45	−109.2298228
2.50	−109.2170830
2.60	−109.1905077
2.70	−109.1635998
2.80	−109.1373583
2.90	−109.1124729
3.00	−109.0894053
3.10	−109.0684502
3.20	−109.0497787
3.30	−109.0334619
3.40	−109.0194835
3.50	−109.0077466
3.60	−108.9980829
3.70	−108.9902691
3.80	−108.9840499
3.90	−108.9791625
4.00	−108.9753572
4.10	−108.9724102
4.20	−108.9701316
4.30	−108.9683664
4.40	−108.9669909
4.50	−108.9659102

Chapter 6

Conclusion

Eigenvalue problems of extreme high dimension are a class of challenging problem which traditional theories and algorithms are powerless. However, they +t+++++++++are important in many applications, for example, the full configuration interaction problem in quantum chemistry and quantum physics. In this thesis which summarizes my Ph.D. research, we push the boundary of solving this challenging problem a little further.

First, to understand the recent proposed randomized algorithms such as full configuration interaction quantum Monte Carlo (FCIQMC) and fast randomized interaction (FRI), we establish a framework of inexact power iteration which fits the randomized algorithms. Convergence theorems are proved and comparison between the algorithms are discussed based on both the convergence theory and the numerical experiments. We find that inexact power iteration such as FCIQMC and FRI perform much better than the traditional power iteration since they put more weight on the important coordinates.

Second, inspired by the inexact power iteration, we analyze coordinate descent methods which share the advantages of inexact power iterations and have potential tobe efficient for the high-dimensional eigenvalue problems. We formulate the eigenvalue problem as an optimization problem and analyze the landscape of the non-convex objective function $f(x)$ of LEVP as (3.1) and conclude that all local minima of $f(x)$ are global minima. We then investigate CD-Cyc-Grad and prove the global convergence of CD-Cyc-Grad on $f(x)$ almost surely. Through the derivation of the exact line search along a coordinate, GCD-LS-LS is presented as the most greedy

CDM and avoids many saddle points of $f(x)$. Finally we propose SCD-Grad-vecLS(t) and SCD-Grad-LS(t). The local convergence analysis for these stochastic CDMs shows that convergence rate of either GCD-LS-LS or SCD-Grad-LS(t) with $t > 0$ is provably faster or equal to that of SCD-Uni-LS. And one example as in Figure 3.1 (b), demonstrates the robustness of the SCD-Grad-LS(t) with small t. Therefore we recommend SCD-Grad-LS(1) as an efficient and robust CDM for LEVP. Numerical results agree with our analysis. The performance of CDMs on a protypical quantum many-body problem of the Hubbard model shows that CDMs has great potential for quantum many-body calculations.

Finally, based on the analysis of coordinate descent methods, we propose a novel efficient algorithm coordinate descent FCI (CDFCI). The proposed CDFCI is an easy-to-use, accurate, and efficient algorithm for full configuration interaction eigenvalue problems of quantum many-body systems, especially for strongly correlated systems. The only tuning parameter in CDFCI, ε, controls the trade-off between memory cost and accuracy. Given the fixed amount of memory, the "close-to-optimal" ε can be determined within a few minutes without waiting for convergent results. Hence, we believe that CDFCI is one of the most easy-to-use algorithms among competitors. Besides the user friendly property, CDFCI performs competitively with many other methods, including heat-bath configuration interaction (HCI), density matrix renormalization group (DMRG), and full configuration interaction quantum Monte Carlo with initiator and semi-stochastic adaptation (iS-FCIQMC). The CDFCI can give the state-of-art results for many strongly correlated FCI problems.

There are several directions worth exploring for future works. Momentum acceleration for coordinate descent [3, 47, 51, 61] can be combined with the coordinate descent methods to potentially accelerate the convergence at least in the local convex area. Also it is of interest to pursue the asynchronous implementation of the coordinate descent methods and prove the convergence property in that setting. Another possible future direction is to employ the sparsity of the matrix into the method such that per-iteration cost could be further reduced.

For CDFCI, we can apply CDFCI to current examples with larger basis sets and other more challenging systems; add perturbation stage to further improve the accuracy; and, parallelize CDFCI in a distributed-memory setting. The earlier two can be accomplished directly, while the massive distributed-memory parallelization of CDFCI requires modification of the greedy determinant-select strategy. Hence FCIQMC type methods currently have some advantage in that regard. As discussed in Chapter 3, with a stochastic variant of the current determinant-select strategy, the asynchronized feature of coordinate descent methods can be enabled. [1] Massive distributed-memory parallelized CDFCI is expected to achieve good performance. Besides these, we are also exploring (semi-)stochastic CDFCI to improve the parallelizability of the algorithm, and further accelerating the initial iterations. Replacing the current hash function with a more efficient one to fully utilize the memory hierarchy is also under investigation. It is also interesting to design an auto-tuning procedure for "close-to-optimal" ε to remove the only tuning parameter in the algorithm.

Beyond ground state computation, CDFCI is also suitable for excited state computation. The extension of the optimization problem (4.4) to low-lying k excited states can be achieved without orthogonality constraint, i.e.,

$$\min_{x \in \mathbb{R}^{N_{\mathrm{FCI}} \times k}} f(x) = \left\| H + xx^\top \right\|_F^2. \tag{6.1}$$

This is favorable as it removes the expensive orthogonalization step for FCI wavefunctions during iterations. Hence extending CDFCI to solve low-lying excited states

---

[1] While it is called asynchronized parallelization in coordinate descent methods, communication is still needed after every several iterations. Hence the embarrassing parallelization of Monte Carlo methods, as in FCIQMC type methods, is of better scalability.

is another promising future direction to be explored.

Appendix A

Proofs of Theorems

A.1 Proof of Theorem 3.3.1

In order to show the global convergence of the CD-Cyc-Grad, we introduce a few definitions and terminologies here. Let χ^s be a set of strict saddle points of $f(x)$, i.e., χ^s is characterized by Lemma 3.2.6. Let $g_j(x)$ be the "coordinate-updating" in Algorithm 5, mapping from $x^{(\ell)}$ to $x^{(\ell+1)}$ with coordinate j, i.e.,

$$x^{(\ell+1)} = g_j(x^{(\ell)}). \tag{A.1}$$

When n steps of updating are composed together, we denote the composed mapping as

$$g = g_n \circ g_{n-1} \circ \cdots \circ g_1. \tag{A.2}$$

The corresponding iteration then is

$$x^{((k+1)n)} = g(x^{(kn)}), \qquad k = 0, 1, 2, \ldots. \tag{A.3}$$

We first establish a few lemmas for the proof of Theorem 3.3.1.

Lemma A.1.1. *Let* $R \geq \sqrt{\max_j \|A_{:,j}\|}$ *be a constant,* $\gamma \leq \frac{1}{4(n+4)R^2}$ *be the stepsize, and* $W_0 = \{x : \|x\|_\infty < R\}$ *be the set of initial vectors. For any* $x^{(0)} \in W_0$, *and any coordinate index* j, *the following iteration is still in* W_0, *i.e.,*

$$x^{(1)} = g_j(x^{(0)}) = x^{(0)} - \gamma \left(-4z_j^{(0)} + 4\|x^{(0)}\|^2 x_j^{(0)} \right) e_j \in W_0, \tag{A.4}$$

where g_j *is defined as* (A.1).

Proof. Since $x^{(0)}$ and $x^{(1)}$ only differ by one entry from each other, in order to validate that $x^{(1)} \in W_0$, it is sufficient to show that

$$\left| x_j^{(0)} - \gamma \left(-4z_j^{(0)} + 4\big\|x^{(0)}\big\|^2 x_j^{(0)} \right) \right| < R. \tag{A.5}$$

In the rest of the proof, we drop the superscript (0) to simplify the notations. We denote the expression in the absolute value in (A.5) as

$$h\big(x_j\big) = -4\gamma x_j^3 + \left(1 + 4\gamma A_{j,j} - 4\gamma \sum_{i \neq j} x_i^2 \right) x_j + 4\gamma \sum_{i \neq j} A_{i,j} x_i, \tag{A.6}$$

which is a cubic polynomial. Showing (A.5) is equivalent to show that $-R < h\big(x\big) < R$ for any $x \in (-R, R)$. The local maximizer and minimizer of $h(x)$ are the roots of $h'(x)$,

$$x^{\pm} = \pm \sqrt{\frac{1}{12\gamma} + \frac{A_{j,j}}{3} - \frac{\sum_{i \neq j} x_i^2}{3}}, \tag{A.7}$$

where we have used the fact that $1 + 4\gamma A_{j,j} - 4\gamma \sum_{i \neq j} x_i^2 \geq 0$, which can be verified using the constraints on R and γ. Again, by the constraints, we obtain

$$\big|x^{\pm}\big| \geq \sqrt{R^2 + \frac{R^2 + A_{j,j}}{3} + \frac{nR^2 - \sum_{i \neq j} x_i^2}{3}} > R, \tag{A.8}$$

which means the local maximizer and minimizer are beyond the interval $(-R, R)$. The leading coefficient of $h(x)$ is $-4\gamma < 0$. Therefore, showing $-R < h(x) < R$ for any $x \in (-R, R)$ can be implied by $h(R) < R$ and $h(-R) > -R$.

The inequality $h(R) < R$ can be shown as

$$\begin{aligned} h(R) &= R + 4\gamma \left(\sum_i A_{i,j} x_i - R \sum_i x_i^2 \right) \leq R + 4\gamma \left(\|A_{:,j}\| \|x\| - R\|x\|^2 \right) \\ &\leq R + 4\gamma \|x\| \left(\|A_{:,j}\| - R^2 \right) < R, \end{aligned} \tag{A.9}$$

where the first inequality adopts Hölder's inequality and the second inequality is due to the fact that $\|x\| \geq R$. The inequality $h(-R) > -R$ can be shown analogously. Hence we proved the lemma. $\square$

Lemma A.1.2. *For any $x \in W_0$,*

$$\max_{i,j} \left| e_i^\top \nabla^2 f(x) e_j \right| < 4(n+3)R^2, \tag{A.10}$$

where W_0 and R are as defined in Lemma A.1.1, f is the objective function as in (3.1).

Proof. For any $x \in W_0$ and any $i, j = 1, 2, \ldots, n$, we have,

$$\left| e_i^\top \nabla^2 f(x) e_j \right| = \left| -4A_{i,j} + 8x_i x_j + 4\|x\|^2 \right| < 4|A_{i,j}| + 8R^2 + 4nR^2 < 4(n+3)R^2. \tag{A.11}$$

$\square$

One direct consequence of Lemma A.1.2 is that for any consecutive iterations in either CD-Cyc-Grad, we have the following relation,

$$f\left(x^{(\ell+1)}\right) \leq f\left(x^{(\ell)}\right) + \nabla_{j_\ell} f\left(x^{(\ell)}\right)\left(x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right) + \frac{L}{2}\left|x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right|^2 \tag{A.12}$$

where j_ℓ is the chosen coordinate at the ℓ-th iteration and $L = 4(n+3)R^2$ is the Lipschitz constant for the gradient of f. To simplify the presentation below, we will use L instead of the explicit expression.

Lemma A.1.3. *Let $R \geq \sqrt{\max_j \|A_{:,j}\|}$ be a constant, $\gamma \leq \frac{1}{4(n+4)R^2}$ be the stepsize, and $W_0 = \{\, x \mid \|x\|_\infty < R \,\}$ be the set of initial vectors. For any $x^{(0)} \in W_0$, and the iteration follows either CD-Cyc-Grad, we have*

$$\lim_{\ell \to \infty} \left\| \nabla f\left(x^{(\ell)}\right) \right\| = 0. \tag{A.13}$$

Proof. Substituting the updating expression in Algorithm 5 into (A.12), we obtain,

$$f\big(x^{(\ell+1)}\big) \leq f\big(x^{(\ell)}\big) - \gamma\big(\nabla_{j_\ell} f\big(x^{(\ell)}\big)\big)^2 + \frac{L\gamma^2}{2}\big(\nabla_{j_\ell} f\big(x^{(\ell)}\big)\big)^2. \tag{A.14}$$

Since $1 - \frac{\gamma L}{2} > 0$, we have for any ℓ

$$\big(\nabla_{j_\ell} f\big(x^{(\ell)}\big)\big)^2 \leq \frac{1}{\gamma\big(1-\frac{\gamma L}{2}\big)}\big(f\big(x^{(\ell)}\big) - f\big(x^{(\ell+1)}\big)\big). \tag{A.15}$$

Summing over all ℓ from 0 to $T = nK$ for any big integer K, we have

$$\begin{aligned}\sum_{k=0}^{K}\sum_{\ell=nk}^{n(k+1)-1}\big(\nabla_{j_\ell} f\big(x^{(\ell)}\big)\big)^2 \leq& \frac{1}{\gamma\big(1-\frac{\gamma L}{2}\big)}\big(f\big(x^{(0)}\big) - f\big(x^{(T)}\big)\big)\\ \leq& \frac{1}{\gamma\big(1-\frac{\gamma L}{2}\big)}\big(f\big(x^{(0)}\big) - f^*\big).\end{aligned} \tag{A.16}$$

where f^* denotes the minimum of the objective function. Hence this means $\lim_{\ell\to\infty} \nabla_{j_\ell} f\big(x^{(\ell)}\big) = 0$. The limit is equivalent to say that for any δ_0, there exists a constant K_0 such that for any $k \geq K_0$, we have,

$$\big|\nabla_{j_\ell} f\big(x^{(\ell)}\big)\big| \leq \delta_0, \qquad \ell = kn, \ldots, (k+1)n - 1. \tag{A.17}$$

Let ℓ_1 and ℓ_2 be two iterations within kn and $(k+1)n - 1$, we first conduct a loose Lipschitz bound on the coordinate-wise gradient, without loss of generality, we assume $\ell_1 \leq \ell_2$,

$$\begin{aligned}\Big|\nabla_{j_{\ell_1}} f\big(x^{(\ell_1)}\big) - \nabla_{j_{\ell_1}} f\big(x^{(\ell_2)}\big)\Big| \leq& \sum_{\ell=\ell_1}^{\ell_2-1}\Big|\nabla_{j_{\ell_1}} f\big(x^{(\ell)}\big) - \nabla_{j_{\ell_1}} f\big(x^{(\ell+1)}\big)\Big|\\ \leq& L\sum_{\ell=\ell_1}^{\ell_2-1}\big|x^{(\ell)} - x^{(\ell+1)}\big|\\ \leq& L\sum_{\ell=\ell_1}^{\ell_2-1}\big|\gamma\nabla_{j_\ell} f\big(x^{(\ell)}\big)\big| \leq n\delta_0,\end{aligned} \tag{A.18}$$

where the last inequality is due to (A.17) and $\gamma L \leq 1$. Let ℓ_0 be an iteration within kn and $(k+1)n-1$. And another important point in Algorithm 5 is that $\{j_\ell\}_{\ell=nk}^{(k+1)n-1}$ is $\{1,2,\ldots,n\}$. Next, we would show that the square norm of the full gradient also converges to zero,

$$\begin{aligned}
\left\|\nabla f\left(x^{(\ell_0)}\right)\right\|^2 =& \sum_{j=1}^{n}\left(\nabla_j f\left(x^{(\ell_0)}\right)\right)^2 \\
=& \sum_{\ell=nk}^{(k+1)n-1}\left(\nabla_{j_\ell} f\left(x^{(\ell_0)}\right)-\nabla_{j_\ell} f\left(x^{(\ell)}\right)+\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2 \\
\leq& \sum_{\ell=nk}^{(k+1)n-1}\Big(\left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2+2\delta_0\left|\nabla_{j_\ell} f\left(x^{(\ell_0)}\right)-\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right| \\
&+\left|\nabla_{j_\ell} f\left(x^{(\ell_0)}\right)-\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right|^2\Big) \\
\leq& (n+2n^2+n^3)\delta_0^2.
\end{aligned} \tag{A.19}$$

Since δ_0 can be arbitary small, therefore, we conclude that $\lim_{\ell\to\infty}\left\|\nabla f\left(x^{(\ell)}\right)\right\|=0$.

Here, the initial vector of the method must be chosen in the W_0 to guarantee the bounded Lipschitz condition on the gradient of f and iteration stays within W_0. □

Proof of Theorem 3.3.1. Since $R \geq \sqrt{\max_j \|A_{:,j}\|}$ and $\gamma \leq \frac{1}{4(n+4)R^2}$, Lemma A.1.1 states that for any $x \in W_0$ and j, we have $g_j(x) \in W_0$. Hence we have, for any $x \in W_0$, $g(x) \in W_0$, where g is defined as (A.2). Further, as stated by Lemma A.1.2, f has bounded Lipschitz coordinate gradient in W_0 and the stepsize γ obeys $\gamma \leq \frac{1}{4(n+4)R^2} < \frac{1}{4(n+3)R^2}$. Proposition 4 in [46] shows that under these conditions, $\det(Dg(x)) \neq 0$. Corollary 1 in [46] shows that $\mu\left(\left\{\, x^{(0)} \;\middle|\; \lim_{k\to\infty} g^k(x^{(0)}) \in \chi^s \,\right\}\right) = 0$. Combining with the conclusion of Lemma A.1.3, we obtain the conclusion of Theorem 3.3.1.

□

A.2 Proof of local convergence of SCD methods

Proof of Lemma 3.5.5. For any $x^{(\ell)} \in D^+ \subset B^+$ and fixed index j, $f\big(x^{(\ell)}\big)$ is a quartic monic polynomial of $x_j^{(\ell)}$, denoted as $p(x_j^{(\ell)})$. Since $f\big(x^{(\ell)}\big) = p(x_j^{(\ell)})$, $\mathcal{I} = \{y : p(y) \leq f\big(x^{(\ell)}\big)\}$ is a non-empty set.

When $p'(x_j^{(\ell)}) = \nabla_j f\big(x^{(\ell)}\big) = 0$, we obtain,

$$f\big(x^{(\ell+1)}\big) \leq f\big(x^{(\ell)}\big) - \frac{1}{2L}\left(\nabla_j f\big(x^{(\ell)}\big)\right)^2. \tag{A.20}$$

When $p'(x_j^{(\ell)}) = \nabla_j f\big(x^{(\ell)}\big) < 0$, there exists an interval $[a, b]$ with $a < b$ such that $p(a) = p(b) = f\big(x^{(\ell)}\big)$ and $p(y) < f\big(x^{(\ell)}\big)$ for all $y \in (a, b)$. It can be further confirmed that $a = x_j^{(\ell)}$. By mean value theorem, there exists at least one number $c \in (a, b)$ such that $p'(c) = 0$. If there are two, let c denote the smaller one so that $p'(y) < 0$ for $y \in (a, c)$. Applying the mean value theorem one more time, we have,

$$0 < -\nabla_j f\big(x^{(\ell)}\big) = -p'(a) = p'(c) - p'(a) = (c - a)p''(\xi) \leq (c - a)L, \tag{A.21}$$

which means

$$a - \frac{\nabla_j f\big(x^{(\ell)}\big)}{L} = x_j^{(\ell)} - \frac{\nabla_j f\big(x^{(\ell)}\big)}{L} \in (a, c). \tag{A.22}$$

Equation (A.22) implies $x^{(\ell)} - \frac{1}{L}\nabla_j f\big(x^{(\ell)}\big)e_j \in D^+$. Following the Lipschitz condition of $\nabla_j f$ in D^+, we obtain,

$$\begin{aligned} f\big(x^{(\ell+1)}\big) \leq& f\big(x^{(\ell)} - \frac{1}{L}\nabla_j f\big(x^{(\ell)}\big)e_j\big) \\ \leq& f\big(x^{(\ell)}\big) - \frac{1}{L}\left(\nabla_j f\big(x^{(\ell)}\big)\right)^2 + \frac{L}{2}\left(\frac{1}{L}\nabla_j f\big(x^{(\ell)}\big)\right)^2 \\ =& f\big(x^{(\ell)}\big) - \frac{1}{2L}\left(\nabla_j f\big(x^{(\ell)}\big)\right)^2. \end{aligned} \tag{A.23}$$

Similarly, equation (A.23) can be obtain when $p'(x_j^{(\ell)}) = \nabla_j f\big(x^{(\ell)}\big) > 0$ as the case of $p'(x_j^{(\ell)}) = \nabla_j f\big(x^{(\ell)}\big) < 0$.

Analogically, we can show the same conditions with the same constants hold for f in D^-.

Once we have $f\left(x^{(\ell+1)}\right) \leq f\left(x^{(\ell)}\right) - \frac{1}{2L}\left(\nabla_j f\left(x^{(\ell)}\right)\right)^2$, it is straightforward to show that $x^{(\ell+1)} \in D^+ \cup D^-$. If $x^{(\ell+1)} \notin D^+ \cup D^-$ then there exist more than two minimizers of $f(x)$ which violates Theorem 3.2.7. Hence, $x^{(\ell+1)} \in D^+ \cup D^-$. □

Proof of Lemma 3.5.6. By Lemma 3.5.5, we have,

$$f\left(x^{(\ell+1)}\right) \leq f\left(x^{(\ell)}\right) - \frac{1}{2L}\left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2, \tag{A.24}$$

where j_ℓ denotes the index picked at ℓ-th iteration. Subtracting f^* from both side of (A.24) and taking the conditional expectation, we obtain,

$$\begin{aligned}
\mathbb{E}\left[f\left(x^{(\ell+1)}\right) \mid x^{(\ell)}\right] - f^* \leq& f\left(x^{(\ell)}\right) - f^* - \frac{1}{2L}\mathbb{E}\left[\left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2 \mid x^{(\ell)}\right] \\
=& f\left(x^{(\ell)}\right) - f^* - \frac{1}{2L}\sum_{j=1}^{n} \frac{\left|\nabla_j f\left(x^{(\ell)}\right)\right|^{t+2}}{\left\|\nabla f\left(x^{(\ell)}\right)\right\|_t^t} \\
=& f\left(x^{(\ell)}\right) - f^* - \frac{1}{2L}\frac{\left\|\nabla f\left(x^{(\ell)}\right)\right\|_{t+2}^t}{\left\|\nabla f\left(x^{(\ell)}\right)\right\|_t^t}\left\|\nabla f\left(x^{(\ell)}\right)\right\|_{t+2}^2 \\
\leq& f\left(x^{(\ell)}\right) - f^* - \frac{1}{2Ln^{\frac{2}{t+2}}}\left\|\nabla f\left(x^{(\ell)}\right)\right\|_{t+2}^2,
\end{aligned} \tag{A.25}$$

where the first equality uses the probability p_j of the sampling procedure and the last equality is due to elementary vector norm inequality.

Next, we will bound the last term in (A.25) with the strongly $\|\cdot\|_{\frac{t+2}{t+1}}$-convex property of $f(x)$. Minimizing both side of (3.25) with respect to y and substituting $x = x^{(\ell)}$, we have,

$$\begin{aligned}
f^* \geq& f\left(x^{(\ell)}\right) - \sup_y \left\{-\nabla f\left(x^{(\ell)}\right)^\top \left(y - x^{(\ell)}\right) - \frac{\mu_q}{2}\left\|y - x^{(\ell)}\right\|_{\frac{t+2}{t+1}}^2\right\} \\
\geq& f\left(x^{(\ell)}\right) - \frac{1}{2\mu_q}\left\|\nabla f\left(x^{(\ell)}\right)\right\|_{t+2}^2,
\end{aligned} \tag{A.26}$$

where $q = \frac{t+2}{t+1}$ and the last supreme term in the first line is a conjugate of function $\frac{\mu_q}{2}\|\cdot\|^2_{\frac{t+2}{t+1}}$ and the result is given by Example 3.27 in [9], which is $\frac{1}{2\mu_q}\|\cdot\|^2_{t+2}$. Here $\|\cdot\|_{t+2}$ is the dual norm of $\|\cdot\|_{\frac{t+2}{t+1}}$.

Substituting (A.26) into (A.25), we obtain,

$$\mathbb{E}\big[f\big(x^{(\ell+1)}\big) \mid x^{(\ell)}\big] - f^* \le \big(1 - \frac{\mu_q}{Ln^{2-2\frac{t+1}{t+2}}}\big)\big(f\big(x^{(\ell)}\big) - f^*\big) = \big(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\big)\big(f\big(x^{(\ell)}\big) - f^*\big). \tag{A.27}$$

□

Proof of Theorem 3.5.7. By Lemma 3.5.5, we know that if $x^{(0)} \in D^+ \cup D^-$, then $x^{(\ell)} \in D^+ \cup D^-$ for all ℓ. Hence Lemma 3.5.6 holds for all $x^{(\ell)} \in D^+ \cup D^-$. We take expectation condition on the sigma algebra of $x^{(0)}$,

$$\begin{aligned}\mathbb{E}\big[f\big(x^{(\ell)}\big) \mid x^{(0)}\big] - f^* \le& \big(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\big)\left(\mathbb{E}\big[f\big(x^{(\ell-1)}\big) \mid x^{(0)}\big] - f^*\right) \\ \le& \cdots \le \big(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\big)^{\ell}\left(f\big(x^{(0)}\big) - f^*\right),\end{aligned} \tag{A.28}$$

where we adopt property of conditional expectation.

Assume $x^{(\ell)} \in D^+$ and $x^* = \sqrt{\lambda_1}v_1 \in D^+$. Due to the strongly convexity of $f(x)$, we have

$$\begin{aligned}\mathbb{E}\big[\mathrm{dist}\big(x^{(\ell)}, X^*\big)^2 \mid x^{(0)}\big] \le& \mathbb{E}\big[\big\|x^{(\ell)} - x^*\big\|^2 \mid x^{(0)}\big] \\ \le& \frac{2}{\mu_2}\mathbb{E}\big[f\big(x^{(\ell)}\big) - f(x^*) \mid x^{(0)}\big] \\ \le& \frac{2}{\mu_2}\big(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\big)^{\ell}\left(f\big(x^{(0)}\big) - f^*\right).\end{aligned} \tag{A.29}$$

If $x^{(\ell)} \in D^-$, then we choose $x^* = -\sqrt{\lambda_1}v_1 \in D^-$ and the conclusion follows as above. □

Bibliography

[1] Abrams, M. L. and David Sherrill, C. (2005). Important configurations in configuration interaction and coupled-cluster wave functions. *Chem. Phys. Lett.*, 412:121–124.

[2] Alaee Kerahroodi, M., Aubry, A., De Maio, A., Naghsh, M. M., and Modarres-Hashemi, M. (2017). A coordinate-descent framework to design low PSL/ISL sequences. *IEEE Trans. Signal Process.*, 65(22):5942–5956.

[3] Allen-Zhu, Z., Qu, Z., Richtárik, P., and Yuan, Y. (2016). Even faster accelerated coordinate descent using non-uniform sampling. In *Proc. 33rd Int. Conf. Mach. Learn.*, pages 1–9.

[4] Blunt, N. S., Smart, S. D., Kersten, J. A. F., Spencer, J. S., Booth, G. H., and Alavi, A. (2015). Semi-stochastic full configuration interaction quantum Monte Carlo: Developments and application. *J. Chem. Phys.*, 142(18):184107.

[5] Booth, G. H., Cleland, D., Thom, A. J. W., and Alavi, A. (2011). Breaking the carbon dimer: The challenges of multiple bond dissociation with full configuration interaction quantum Monte Carlo methods. *J. Chem. Phys.*, 135(8):084104.

[6] Booth, G. H., Grüneis, A., Kresse, G., and Alavi, A. (2013). Towards an exact description of electronic wavefunctions in real solids. *Nature*, 493:365–370.

[7] Booth, G. H., Smart, S. D., and Alavi, A. (2014). Linear-scaling and parallelisable algorithms for stochastic quantum chemistry. *Mol. Phys.*, 112(14):1855–1869.

[8] Booth, G. H., Thom, A. J. W., and Alavi, A. (2009). Fermion Monte Carlo without fixed nodes: A game of life, death, and annihilation in Slater determinant space. *J. Chem. Phys.*, 131(5):054106.

[9] Boyd, S. and Vandenberghe, L. (2004). *Convex optimization.* Cambridge University Press, Cambridge.

[10] Buenker, R. J. and Peyerimhoff, S. D. (1974). Individualized configuration selection in CI calculations with subsequent energy extrapolation. *Theor. Chim. Acta.*, 35(1):33–58.

[11] Chan, G. K.-L. (2004). An algorithm for large scale density matrix renormalization group calculations. *J. Chem. Phys.*, 120(7):3172–3178.

[12] Chan, G. K.-L. and Head-Gordon, M. (2002). Highly correlated calculations with a polynomial cost algorithm: A study of the density matrix renormalization group. *J. Chem. Phys.*, 116(11):4462–4476.

[13] Chan, G. K.-L., Kállay, M., and Gauss, J. (2004). State-of-the-art density matrix

renormalization group and coupled cluster theory studies of the nitrogen binding curve. *J. Chem. Phys.*, 121(13):6110–6116.

[14] Chan, G. K.-L. and Sharma, S. (2011). The density matrix renormalization group in quantum chemistry. *Annu. Rev. Phys. Chem.*, 62(1):465–481.

[15] Chow, Y. T., Wu, T., and Yin, W. (2017). Cyclic coordinate-update algorithms for fixed-point problems: Analysis and applications. *SIAM J. Sci. Comput.*, 39(4):A1280–A1300.

[16] Cleland, D., Booth, G. H., and Alavi, A. (2010). Communications: Survival of the fittest: Accelerating convergence in full configuration-interaction quantum Monte Carlo. *J. Chem. Phys.*, 132(4):041103.

[17] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2009). *Introduction to algorithms*. The MIT Press, 3rd edition.

[18] Corsetti, F. (2014). The orbital minimization method for electronic structure calculations with finite-range atomic basis sets. *Comput. Phys. Commun.*, 185(3):873–883.

[19] Daudey, J.-P., Heully, J.-L., and Malrieu, J.-P. (1993). Size-consistent self-consistent truncated or selected configuration interaction. *J. Chem. Phys.*, 99(2):1240–1254.

[20] Davidson, E. R. (1975). The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comput. Phys.*, 17(1):87–94.

[21] D'esopo, D. A. (1959). A convex programming procedure. *Nav. Res. Logist. Q.*, 6(1):33–42.

[22] Evangelista, F. A. (2014). Adaptive multiconfigurational wave functions. *J. Chem. Phys.*, 140(12):124114.

[23] Ge, R., Huang, F., Jin, C., and Yuan, Y. (2015). Escaping from saddle points – online stochastic gradient for tensor decomposition. In *J. Mach. Learn. Res.*, volume 40, pages 1–46.

[24] Ghosh, D., Hachmann, J., Yanai, T., and Chan, G. K.-L. (2008). Orbital optimization in the density matrix renormalization group, with applications to polyenes and β-carotene. *J. Chem. Phys.*, 128(14):144117.

[25] Giner, E., Scemama, A., and Caffarel, M. (2013). Using perturbatively selected configuration interaction in quantum Monte Carlo calculations. *Can. J. Chem.*, 91(9):879–885.

[26] Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. The Johns Hopkins University Press, 4th edition.

[27] Greer, J. C. (1995). Estimating full configuration interaction limits from a Monte Carlo selection of the expansion space. *J. Chem. Phys.*, 103(5):1821–1828.

[28] Greer, J. C. (1998). Monte Carlo Configuration Interaction. *J. Comput. Phys*, 146(1):181–202.

[29] Gurbuzbalaban, M., Ozdaglar, A., Parrilo, P. A., and Vanli, N. (2017). When cyclic coordinate descent outperforms randomized coordinate descent. In *Adv. Neural Inf. Process. Syst. 30*, pages 6999–7007.

[30] Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.*, 53(2):217–288.

[31] Han, J., Song, K. S., Kim, J., and Kang, M. G. (2018). Permuted coordinate-wise optimizations applied to Lp-regularized image deconvolution. *IEEE Trans. Image Process.*, 27(7):3556–3570.

[32] Harrison, R. J. (1991). Approximating full configuration interaction with selected configuration interaction and perturbation theory. *J. Chem. Phys.*, 94(7):5021–5031.

[33] Holmes, A. A., Changlani, H. J., and Umrigar, C. J. (2016a). Efficient heat-bath sampling in Fock space. *J. Chem. Theory Comput.*, 12:1561–1571.

[34] Holmes, A. A., Tubman, N. M., and Umrigar, C. J. (2016b). Heat-bath configuration interaction: An efficient selected configuration interaction algorithm inspired by heat-bath sampling. *J. Chem. Theory Comput.*, 12(8):3674–3680.

[35] Hong, M., Wang, X., Razaviyayn, M., and Luo, Z.-Q. (2017). Iteration complexity analysis of block coordinate descent methods. *Math. Program.*, 163(1-2):85–114.

[36] Huron, B., Malrieu, J. P., and Rancurel, P. (1973). Iterative perturbation calculations of ground and excited state energies from multiconfigurational zeroth-order wavefunctions. *J. Chem. Phys.*, 58(12):5745–5759.

[37] Illas, F., Rubio, J., Ricart, J. M., and Bagus, P. S. (1991). Selected versus complete configuration interaction expansions. *J. Chem. Phys.*, 95(3):1877–1883.

[38] Ivanic, J. (2003). Direct configuration interaction and multiconfigurational self-consistent-field method for multiple active spaces with variable occupations. I. Method. *J. Chem. Phys.*, 119(18):9364–9376.

[39] Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. (2017). How to

escape saddle points efficiently. In *Proc. 34th Int. Conf. Mach. Learn.*, volume 70 of *Proceedings of Machine Learning Research*, pages 1724–1732, International Convention Centre, Sydney, Australia. PMLR.

[40] Johnson, T. B. and Guestrin, C. (2017). StingyCD: Safely avoiding wasteful updates in coordinate descent. In *Proc. 34th Int. Conf. Mach. Learn.*, pages 1752–1760.

[41] Knowles, P. J. (2015). Compressive sampling in configuration interaction wavefunctions. *Mol. Phys.*, 113(13-14):1655–1660.

[42] Knowles, P. J. and Handy, N. C. (1984). A new determinant-based full configuration interaction method. *Chem. Phys. Lett.*, 111(4-5):315–321.

[43] Knyazev, A. V. (2001). Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput.*, 23(2):517–541.

[44] Lanczos, C. (1950). An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand. (1934).*, 45(4):255.

[45] Lee, C.-p. and Wright, S. J. (2016). Random permutations fix a worst case for cyclic coordinate descent. *IMA J. Numer. Anal.*

[46] Lee, J. D., Panageas, I., Piliouras, G., Simchowitz, M., Jordan, M. I., and Recht, B. (2019). First-order methods almost always avoid strict saddle points. *Math. Program.*, 176(1-2):311–337.

[47] Lee, Y. T. and Sidford, A. (2013). Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems. In *2013 IEEE 54th Annu. Symp. Found. Comput. Sci.*, pages 147–156. IEEE.

[48] Lei, Q., Zhong, K., and Dhillon, I. S. (2016). Coordinate-wise power method. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Adv. Neural Inf. Process. Syst. 29*, pages 2064–2072. Curran Associates, Inc.

[49] Li, Y., Lu, J., and Wang, Z. (2019). Coordinate-wise descent methods for leading eigenvalue problem. *SIAM J. Sci. Comput.*, 41(4):A2681–A2716.

[50] Lim, L.-H. and Weare, J. (2017). Fast randomized iteration: Diffusion Monte Carlo through the lens of numerical linear algebra. *SIAM Rev.*, 59(3):547–587.

[51] Lin, Q., Lu, Z., and Xiao, L. (2015). An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization. *SIAM J. Optim.*, 25:2244–2273.

[52] Liu, J. and Wright, S. J. (2015). Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM J. Optim.*, 25(1):351–376.

[53] Liu, J., Wright, S. J., Ré, C., Bittorf, V., and Sridhar, S. (2015a). An asynchronous parallel stochastic coordinate descent algorithm. *J. Mach. Learn. Res.*, 16:285–322.

[54] Liu, W. and Hoffmann, M. R. (2016). iCI: Iterative CI toward full CI. *J. Chem. Theory Comput.*, 12(3):1169–1178.

[55] Liu, X., Wen, Z., and Zhang, Y. (2015b). An Efficient Gauss–Newton Algorithm for Symmetric Low-Rank Product Matrix Approximations. *SIAM J. Optim.*, 25(3):1571–1608.

[56] Lu, J. and Thicke, K. (2017). Orbital minimization method with l1 regularization. *J. Comput. Phys.*, 336:87–103.

[57] Lu, J. and Wang, Z. (2020). The full configuration interaction quantum Monte Carlo method through the lens of inexact power iteration. *SIAM J. Sci. Comput.*, 42(1):B1–B29.

[58] Ma, D., Li Manni, G., and Gagliardi, L. (2011). The generalized active space concept in multiconfigurational self-consistent field methods. *J. Chem. Phys.*, 135(4):044128.

[59] Mitchell, A. E., Smith, G. E., Bell, K. L., and Rangaswamy, M. (2016). Coordinate descent for cognitve radar adaptation. In *2016 CIE Int. Conf. Radar*, pages 1–5. IEEE.

[60] Nesterov, Y. (2004). *Introductory lectures on convex optimization: a basic course.* Springer, 1st edition.

[61] Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM J. Optim.*, 22(2):341–362.

[62] Nutini, J., Schmidt, M., Laradji, I. H., Friedlander, M., and Koepke, H. (2015). Coordinate descent converges faster with the Gauss-Southwell rule than random selection. In *Proc. 32nd Int. Conf. Mach. Learn.*, pages 1632–1641.

[63] Olivares-Amaya, R., Hu, W., Nakatani, N., Sharma, S., Yang, J., and Chan, G. K.-L. (2015). The ab-initio density matrix renormalization group in practice. *J. Chem. Phys.*, 142(3):034102.

[64] Olsen, J. (1996). Full configuration-interaction and state of the art correlation calculations on water in a valence double-zeta basis with polarization functions. *J. Chem. Phys.*, 104(20):8007.

[65] Olsen, J., Jørgensen, P., Koch, H., Balkova, A., and Bartlett, R. J. (1998). Full configuration-interaction and state of the art correlation calculations on water in a valence double-zeta basis with polarization functions. *J. Chem. Phys.*, 104(20):8007.

[66] Parnell, T., Dünner, C., Atasu, K., Sifalakis, M., and Pozidis, H. (2020). Tera-scale coordinate descent on GPUs. *Futur. Gener. Comput. Syst.*, 108:1173–1191.

[67] Parrish, R. M., Burns, L. A., Smith, D. G. A., Simmonett, A. C., DePrince, A. E., Hohenstein, E. G., Bozkaya, U., Sokolov, A. Y., Di Remigio, R., Richard, R. M., Gonthier, J. F., James, A. M., McAlexander, H. R., Kumar, A., Saitow, M., Wang, X., Pritchard, B. P., Verma, P., Schaefer, H. F., Patkowski, K., King, R. A., Valeev, E. F., Evangelista, F. A., Turney, J. M., Crawford, T. D., and Sherrill, C. D. (2017). Psi4 1.1: An open-source electronic structure program emphasizing automation, advanced libraries, and interoperability. *J. Chem. Theory Comput.*, 13(7):3185–3197.

[68] Peng, Z., Wu, T., Xu, Y., Yan, M., and Yin, W. (2016). Coordinate-friendly structures, algorithms and applications. *Ann. Math. Sci. Appl.*, 1(1):57–119.

[69] Perekrestenko, D., Cevher, V., and Jaggi, M. (2017). Faster coordinate descent via adaptive importance sampling. In *Proc. Mach. Learn. Res.*, volume 54, pages 869–877.

[70] Petruzielo, F. R., Holmes, A. A., Changlani, H. J., Nightingale, M. P., and Umrigar, C. J. (2012). Semistochastic projector Monte Carlo method. *Phys. Rev. Lett.*, 109(23):230201.

[71] Richtárik, P. and Takáč, M. (2014). Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Math. Program.*, 144(1-2):1–38.

[72] Rolik, Z., Szabados, Á., and Surján, P. R. (2008). A sparse matrix based full-configuration interaction algorithm. *J. Chem. Phys.*, 128(14):144101.

[73] Roth, R. (2009). Importance truncation for large-scale configuration interaction approaches. *Phys. Rev. C*, 79(6):064324.

[74] Schriber, J. B. and Evangelista, F. A. (2016). Communication: An adaptive configuration interaction approach for strongly correlated electrons with tunable accuracy. *J. Chem. Phys.*, 144(16):161106.

[75] Schriber, J. B. and Evangelista, F. A. (2017). Adaptive configuration interaction for computing challenging electronic excited states with tunable accuracy. *J. Chem. Theory Comput.*, 13(11):5354–5366.

[76] Schwarz, L. R., Alavi, A., and Booth, G. H. (2017). Projector Quantum Monte

Carlo Method for Nonlinear Wave Functions. *Phys. Rev. Lett.*, 118(17):176403.

[77] Sharma, S. and Alavi, A. (2015). Multireference linearized coupled cluster theory for strongly correlated systems using matrix product states. *J. Chem. Phys.*, 143(10):102815.

[78] Sharma, S. and Chan, G. K.-L. (2012). Spin-adapted density matrix renormalization group algorithms for quantum chemistry. *J. Chem. Phys.*, 136(12):124121.

[79] Sharma, S., Holmes, A. A., Jeanmairet, G., Alavi, A., and Umrigar, C. J. (2017). Semistochastic Heat-Bath Configuration Interaction Method: Selected Configuration Interaction with Semistochastic Perturbation Theory. *J. Chem. Theory Comput.*, 13(4):1595–1604.

[80] Shi, H.-J. M., Tu, S., Xu, Y., and Yin, W. (2016). A primer on coordinate descent algorithms. http://arxiv.org/abs/1610.00040.

[81] Shi, Q., Sun, H., Lu, S., Hong, M., and Razaviyayn, M. (2017). Inexact block coordinate descent methods for symmetric nonnegative matrix factorization. *IEEE Trans. Signal Process.*, 65(22):5995–6008.

[82] Sleijpen, G. L. G. and Van der Vorst, H. A. (1996). A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17(2):401–425.

[83] Southwell, R. V. (2010). *Relaxation methods in engineering science – a treatise on approximate computation.* Pickard Press.

[84] Spencer, J. S., Blunt, N. S., Choi, S., Etrych, J., Filip, M.-A., Foulkes, W. M. C., Franklin, R. S. T., Handley, W. J., Malone, F. D., Neufeld, V. A., Di Remigio, R., Rogers, T. W., Scott, C. J. C., Shepherd, J. J., Vigor, W. A., Weston, J., Xu, R., and Thom, A. J. (2019). The HANDE-QMC project: open-source stochastic quantum chemistry from the ground state up. *J. Chem. Theory Comput.*, 15(3):1728–1742.

[85] Spencer, J. S., Blunt, N. S., Vigor, W. A., Malone, F. D., Foulkes, W. M. C., Shepherd, J. J., and Thom, A. J. W. (2015). Open-Source Development Experiences in Scientific Software: The HANDE Quantum Monte Carlo Project. *J. Open Res. Softw*, 3(1).

[86] Sun, R. and Ye, Y. (2019). Worst-case complexity of cyclic coordinate descent: $O(n^2)$ gap with randomized version. *Math. Program.*, pages 1–34.

[87] Tappenden, R., Takáč, M., and Richtárik, P. (2018). On the complexity of parallel coordinate descent. *Optim. Methods Softw.*, 33(2):372–395.

[88] Ten-no, S. (2013). Stochastic determination of effective Hamiltonian for the full

configuration interaction solution of quasi-degenerate electronic states. *J. Chem. Phys.*, 138(16):164126.

[89] Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494.

[90] Tu, S., Roelofs, R., Venkataraman, S., and Recht, B. (2016). Large scale kernel learning using block coordinate descent. http://arxiv.org/abs/1602.05310.

[91] Tubman, N. M., Lee, J., Takeshita, T. Y., Head-Gordon, M., and Whaley, K. B. (2016). A deterministic alternative to the full configuration interaction quantum Monte Carlo method. *J. Chem. Phys.*, 145(4):044112.

[92] Vandaele, A., Gillis, N., Lei, Q., Zhong, K., and Dhillon, I. (2016). Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization. *IEEE Trans. Signal Process.*, 64(21):5571–5584.

[93] Vecharynski, E., Yang, C., and Pask, J. E. (2015). A projected preconditioned conjugate gradient algorithm for computing many extreme eigenpairs of a Hermitian matrix. *J. Comput. Phys.*, 290:73–89.

[94] Wang, J., Wang, W., Garber, D., and Srebro, N. (2018). Efficient coordinate-wise leading eigenvector computation. In *Proc. Algorithmic Learn. Theory, PMLR*, volume 83, pages 806–820.

[95] Wang, Z., Li, Y., and Lu, J. (2019). Coordinate Descent Full Configuration Interaction. *J. Chem. Theory Comput.*, 15(6):3558–3569.

[96] White, S. R. and Martin, R. L. (1999). Ab initio quantum chemistry using the density matrix renormalization group. *J. Chem. Phys.*, 110(9):4127.

[97] Wright, S. J. (2015). Coordinate descent algorithms. *Math. Program.*, 151(1):3–34.

[98] Wright, S. J. and Lee, C.-p. (2020). Analyzing random permutations for cyclic coordinate descent. *Math. Comput.*, 89(325):2217–2248.

[99] Wu, M., Dick, C., Cavallaro, J. R., and Studer, C. (2016a). FPGA design of a coordinate descent data detector for large-scale MU-MIMO. In *2016 IEEE Int. Symp. Circuits Syst.*, pages 1894–1897. IEEE.

[100] Wu, M., Dick, C., Cavallaro, J. R., and Studer, C. (2016b). High-throughput data detection for massive MU-MIMO-OFDM using coordinate descent. *IEEE Trans. Circuits Syst. I Regul. Pap.*, 63(12):2357–2367.

[101] Xu, Y. (2018). Hybrid Jacobian and Gauss–Seidel proximal block coordinate update methods for linearly constrained convex programming. *SIAM J. Optim.*,

28(1):646–670.

[102] You, Y., Lian, X., Liu, J., Yu, H.-F., Dhillon, I. S., Demmel, J., and Hsieh, C.-J. (2016). Asynchronous parallel greedy coordinate descent. In *Adv. Neural Inf. Process. Syst. 29*, pages 4682–4690. Curran Associates, Inc.

[103] Zeng, W. J. and So, H. C. (2020). Coordinate descent algorithms for phase retrieval. *Signal Process.*, 169:107418.

[104] Zhang, T. and Evangelista, F. A. (2016). A Deterministic Projector Configuration Interaction Approach for the Ground State of Quantum Many-Body Systems. *J. Chem. Theory Comput.*, 12(9):4326–4337.

[105] Zimmerman, P. M. (2017a). Incremental full configuration interaction. *J. Chem. Phys.*, 146(10):104102.

[106] Zimmerman, P. M. (2017b). Strong correlation in incremental full configuration interaction. *J. Chem. Phys.*, 146(22):224104.