COORDINATEWISE DESCENT METHODS FOR LEADING EIGENVALUE PROBLEM*

YINGZHOU LI†, JIANFENG LU†‡, AND ZHE WANG†

Abstract. Leading eigenvalue problems for large scale matrices arise in many applications. Coordinatewise descent methods are considered in this work for such problems based on a reformulation of the leading eigenvalue problem as a nonconvex optimization problem. The convergence of several coordinatewise methods is analyzed and compared. Numerical examples of applications to quantum many-body problems demonstrate the efficiency and provide benchmarks of the proposed coordinatewise descent methods.

Key words. stochastic iterative method, coordinatewise descent method, leading eigenvalue problem

AMS subject classifications. 65F15, 90C06, 90C26

DOI. 10.1137/18M1202505

1. Introduction. This paper focuses on solving the leading eigenvalue problem (LEVP) via coordinatewise descent methods (CDMs). Given a symmetric matrix A, the LEVP is defined as

$$Ax = \lambda x, \tag{1}$$

where $A \in \mathbb{R}^{n \times n}$, $A^\top = A$, λ is the largest eigenvalue of A, and $x \in \mathbb{R}^n$ is the corresponding eigenvector (we assume here and in what follows that the leading eigenvalue is positive and nondegenerate). In the case that several leading eigenpairs are needed, we can combine the methods for LEVP together with the deflation technique to obtain all desired eigenpairs. If the eigenvalue of largest magnitude is needed, we can apply the proposed methods to both A and $-A$ and then select the desired one.

Leading eigenvalue(s) problems appear in a wide range of applications, including principal component analysis, 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., the power method, the Lanczos algorithm [9, 17], randomized SVD [11], the (Jacobi–)Davidson algorithm [6, 37], local optimal block preconditioned conjugate gradient [16], projected preconditioned conjugate gradient [45], the orbital minimization method [5, 26], the Gauss–Newton algorithm [25], etc. However, most of those 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 iteration, which is too expensive for large A. In this paper, we focus on designing algorithms for solving LEVP of extremely large size

†Department of Mathematics, Duke University, Durham, NC 27708 (yingzhou.li@duke.edu, zhe.wang3@duke.edu).

‡Department of Chemistry and Department of Physics, Duke University, Durham, NC 27708 (jian feng@math.duke.edu).

and expensive matrix evaluation. Full configuration interaction (FCI) from quantum many-body problems is one of the examples in such a setting. In FCI calculation [40, pp. 350–353], the size of the Hamiltonian matrix grows factorially as the number of electrons and orbitals in the system increases. For such a large dimensional problem, it is infeasible to store the matrix in the main memory and matrix entry should be computed on-the-fly. Thus the evaluation of the matrix entry is expensive. Another property of FCI is that the magnitudes of coordinates of the leading eigenvector span widely. Many coordinates are relatively negligible. Hence traditional methods, which treat all coordinates evenly, are unfavorable for FCI problems. Novel algorithms adapted to the properties of FCI are desired. Although the designed algorithms in this paper can be applied to other problems, FCI will be regarded as the target application.

The LEVP naturally can be rewritten as the following unconstrained optimization problem:

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

where $\|\cdot\|_F$ denotes the Frobenius norm. Throughout this paper, we denote $f(x) = \left\| A - xx^\top \right\|_F^2$ as the objective function. Based on the eigendecomposition of A, it is easy to verify that $x = \pm\sqrt{\lambda_1}v_1$ are minimizers of (2), where λ_1 is the largest eigenvalue of A and v_1 is the corresponding eigenvector with $\|v_1\| = 1$. Therefore, if the optimization problem (2) 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 (2) has appeared before for solving eigenvalue problems [21, 25]. In this paper, we solve the LEVP problem (2) with algorithms adapted to FCI problems.

Recently, with the rise of big data and deep learning, stochastic iterative methods have been 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 CDMs are two popular groups of methods among them. Considering CDMs, when a 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 often be n times bigger than that in the traditional gradient descent method for the convex objective function [29, 34, 47]. Massive parallelization capability is also observed [23, 24, 31, 41, 52], which is achieved mostly via its "asynchronized" property of different coordinates.[1] In this work, we develop and analyze methods within the scope of CDMs addressing the LEVP.

1.1. Coordinatewise descent method. The history of CDMs dates back to the early development of the discipline of optimization [7]. It did not catch much attention until very recently. Readers are referred to the recent surveys [35, 47] and references therein for a detailed history of the development and analysis of general CDMs. Since the survey papers, the area of CDMs remains under fast development; see, e.g., [15, 33, 48, 51]. Momentum acceleration can also be combined with the CDMs to further accelerate the convergence [2, 20, 22, 29]. Besides these new developments of methodology, new applications adopt CDMs to accelerate computations, including but not limited to the areas of imaging processing [12], signal processing [1, 27, 53],

---

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

wireless communication [49, 50], and data science [36, 43, 44]. It is worth mentioning that Peng et al. [32] discussed coordinate friendly structure, which could be beneficial for more applications.

In terms of designing CDMs for LEVP, Lei, Zhong, and Dhillon [21] proposed the coordinatewise power method (CPM) addressing the LEVP. CPM accelerates the traditional power method by the coordinatewise 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. [46] adopt the shift-and-invert power method to solve the LEVP, where the inverse of the shifted linear system is addressed by the CDM. Different coordinate updating rules are employed, i.e., the Gauss–Southwell–Lipschitz rule (SI-GSL), the cyclic rule, and the accelerated randomized coordinate descent methods. All these methods are proposed as methods calculating the leading eigenvector and outperform many other methods when an accurate upper bound of the leading eigenvalue is given. It is not clear how to obtain such an upper bound efficiently in practice, however.

A general CDM for the LEVP can be summarized as in Algorithm 1. 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_ℓth 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 obtaining a good choice of j_ℓ and the update is to get $z^{(\ell)} = Ax^{(\ell)}$ involved in the calculation. Since $x^{(\ell+1)}$ and $x^{(\ell)}$ differ by only 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 fewer operations. Generally, most CDMs cost $O(n)$ or fewer operations per iteration, which is much smaller than a traditional iterative method with $O(n^2)$ operations per iteration. Such a gap of the computational cost per iteration enables CDM to focus 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 few more iterations to achieve the convergence criteria, the operations counts and the runtime is much smaller than that of many traditional iterative methods [4, 21].

Algorithm 1. General CDM for LEVP.

Input: Symmetric matrix $A \in \mathbb{R}^{n \times n}$; initial vector $x^{(0)}$.

1: $z^{(0)} = Ax^{(0)}$

2: $\ell = 0$

3: **while** (not converged) **do**

4:    $j_\ell = \text{coordinate-pick}(x^{(\ell)}, z^{(\ell)})$

5:    $x_j^{(\ell+1)} = \begin{cases} \text{coordinate-update}(x^{(\ell)}, z^{(\ell)}, j_\ell), & j = j_\ell, \\ x_j^{(\ell)}, & j \neq j_\ell \end{cases}$

6:    $z^{(\ell+1)} = z^{(\ell)} + A_{:,j_\ell}\left(x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right)$

7:    $\ell = \ell + 1$

8: **end while**

TABLE 1
Short names in name convention.

	Short name	Explanation
Type of CDM	CD	plain CDM
	GCD	greedy CDM, the coordinate is picked in 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

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 composed of three parts: the type of coordinatewise descent, the coordinate-pick strategy, and the coordinate-update strategy. The three parts are separated with hyphens. Table 1 defines the short names used in each part. Some of the short names work with the 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 the literature are renamed under this system, e.g., the coordinate descent methods considered in [7, 13] and [19] are called CD-Cyc-LS and CD-Cyc-Grad, respectively, and SGCD [21] is renamed as GCD-Grad-LS.

1.2. Contribution. We first highlight our contribution as follows and then discuss in detail:

- Analyze the landscape of $f(x)$ as (2) 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 detail, we first analyze the landscape of the objective function $f(x)$. Through our analysis, we show that, although $f(x)$ is nonconvex, $x = \pm\sqrt{\lambda_1}v_1$ 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 a 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 (2).

As with many other problems solved by CDMs, an exact line search along each coordinate direction can be conducted for (2). 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 nonconvex problem in (2) are escapable.

Though the greedy method, the GCD-LS-LS, is efficient with a 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 the 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 convergence 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 a 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 nonconvex. 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 (2), 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 (2) outperform the power method, the CPM, and full gradient descent with exact line search. When the matrix is shifted by a big positive number, CDMs of (2) converge in a similar number of iterations as in the case without shifting, while the power method or CPM, 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 are more important than others due to the physical property, all CDMs including CDMs of (2) and the CPM significantly outperform full vector updating methods. This shows great potential of applying the CDMs to quantum many-body problems.

1.3. Organization. The rest of the paper is organized as follows. We introduce notation and analyze the landscape of (2) in section 2. Sections 3, 4, and 5 follow the same structure, which first introduces or reviews several CDMs, and then conducts the corresponding analysis. Section 3 focuses on CDMs with conservative stepsize, whereas sections 4 and 5 focus on greedy and stochastic CDMs, respectively. The numerical results are given in section 6, followed by the conclusion and discussion in section 7.

2. Landscape analysis. This section focuses on the analysis of the variational problem (2). 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 (2), and more importantly, provide insights in designing optimization algorithms.

Notation. Before the detailed analysis, we define some common notation, as shown in Table 2. This notation will be used without further explanation for the rest of the paper.

Some of the notation in Table 2 can be used in a combined way, e.g., $x_i^{(\ell)}$ denotes the ith entry of the vector x at the ℓth iteration. The set of coordinate indices, 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 the Frobenius norm and the 2-norm are different measures

TABLE 2
Common notation.

Notation	Explanation
n	the size of the problem
i, j	coordinate index
Ω	a set of coordinate indices
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 ith entry and zero everywhere else
x, y, z	vectors
$x^{(\ell)}$	the vector of the ℓth iteration
x_i	the ith 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 ith row and the jth column of matrix A
$\|\cdot\|_F, \|\cdot\|_2$	Frobenius norm and 2-norm

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., $\|x\| = \|x\|_2 = \|x\|_F$.

Stationary points and global minimizers. When working with nonconvex 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 [28]. Without rate, Lee et al. [19] 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 [8, 14]. 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 (2) 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 (2)). 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 2.1 (stationary point). *A point y is a stationary point of f if $\nabla f(y) = 0$.*

DEFINITION 2.2 (strict saddle point). *A point y is a strict saddle point of f if y is a stationary point and $\lambda_{min}\left(\nabla^2 f(y)\right) < 0$, where $\lambda_{min}\left(\nabla^2 f(y)\right)$ is the smallest eigenvalue of the Hessian matrix.[2]*

DEFINITION 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.*

---

[2]This definition of the strict saddle point includes local maximizer as well, which does not cause trouble in minimizing $f(x)$ as in (2).

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 (2). The assumptions on the matrix A are summarized in Assumption A.

Assumption A. 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 2.4. *Under Assumption* A, $x = 0$ *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* $v \in \mathrm{null}(\lambda I - A)$ *with* $\|v\| = 1$. *In particular,* $\pm\sqrt{\lambda_1}v_1$ *are both stationary points.*

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

$$(3) \qquad \nabla f(x) = -4Ax + 4\left(x^\top x\right)x = 0.$$

Obviously, $x = 0$ is a stationary point. When $x \neq 0$, we have $\left(x^\top x I - A\right)x = 0$. 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 $x \in \mathrm{null}(\lambda_1 I - A) = \mathrm{span}(v_1)$ and $x^\top x = \lambda_1$, which implies $x = \pm\sqrt{\lambda_1}v_1$. When $0 < \lambda < \lambda_1$, we have $x \in \mathrm{null}(\lambda I - A)$ and $x^\top x = \lambda$. Hence, $x = \sqrt{\lambda}v$ for $v \in \mathrm{null}(\lambda I - A)$ and $\|v\| = 1$. $\qquad\square$

LEMMA 2.5. *Under Assumption* A, 0 *and* $\sqrt{\lambda}v$ *with* $v \in \mathrm{null}(\lambda I - A)$ *and* $\|v\| = 1$ *are strict saddle points of* $f(x)$, *where* $\lambda < \lambda_1$ *and* λ *is a positive eigenvalue of* A.

Proof. According to Lemma 2.4, 0 and $\sqrt{\lambda}v$ are stationary points of $f(x)$. It suffices to validate the second condition in Definition 2.2, i.e.,

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

When $x = 0$, (4) is obvious since $\lambda_1 > 0$. When $x = \sqrt{\lambda}v$, we apply $\nabla^2 f(x)$ to v_1 and get

$$(5) \qquad \nabla^2 f(x)v_1 = 4(\lambda - \lambda_1)v_1,$$

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. $\qquad\square$

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 2.4 and 2.5.

THEOREM 1. *Under Assumption* A, *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. Lemmas 2.4 and 2.5 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 $-\sqrt{\lambda_1}v_1$:

$$(6) \qquad f\left(\sqrt{\lambda_1}v_1\right) = f\left(-\sqrt{\lambda_1}v_1\right) = \left\|A - \lambda_1 v_1 v_1^\top\right\|_F^2.$$

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. $\qquad\square$

Theorem 1 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. Coordinatewise 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 guarantees 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 coordinatewise gradient based methods is too large to be competitive.

3.1. CD-Cyc-Grad and SCD-Cyc-Grad. The first CDM we consider addressing (2) is CD-Cyc-Grad, following the name convention in section 1.1.

CD-Cyc-Grad conducts the "coordinate-pick" step in Algorithm 1 in a cyclic way, i.e.,

$$(7) \qquad j_\ell = (\ell \mod n) + 1.$$

Hence, all coordinates are picked in a fixed order with an almost equal number of updatings throughout iterations. The "coordinate-update" adopts the coordinatewise gradient, that is,

$$(8) \qquad x_{j_\ell}^{(\ell+1)} = x_{j_\ell}^{(\ell)} - \gamma \nabla_{j_\ell} f\left(x^{(\ell)}\right) = x_{j_\ell}^{(\ell)} - \gamma \left(-4A_{j_\ell,:}x^{(\ell)} + 4\left\|x^{(\ell)}\right\|^2 x_{j_\ell}^{(\ell)}\right),$$

where $\nabla_{j_\ell} f(x^{(\ell)})$ denotes the j_ℓth entry of the gradient of f at $x^{(\ell)}$ and γ is the stepsize. CD-Cyc-Grad is detailed as Algorithm 2, with $z^{(\ell)} = Ax^{(\ell)}$ being adopted in (8). The advantage of CD-Cyc-Grad over the full gradient descent method is mainly that the choice of the stepsize in CD-Cyc-Grad could be much larger than that in gradient descent, which could lead to faster convergence [29].

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 iterations update the coordinate according to the order in $\Pi^{(\ell)}$. Recently, several works [10, 18, 39] discussed the comparison of the convergence rates for CDMs with cyclic and random strategy of "coordinate-pick" for convex problems.

Algorithm 2. CD-Cyc-Grad for LEVP.

Input: Symmetric matrix $A \in \mathbb{R}^{n \times n}$; initial vector $x^{(0)}$; stepsize γ.

1: $z^{(0)} = Ax^{(0)}$

2: $\ell = 0$

3: **while** (not converged) **do**

4: $\quad j_\ell = (\ell \mod n) + 1$

5: $\quad x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} - \gamma \left(-4z_j^{(\ell)} + 4\|x^{(\ell)}\|^2 x_j^{(\ell)}\right), & j = j_\ell \\ x_j^{(\ell)}, & j \neq j_\ell \end{cases}$

6: $\quad z^{(\ell+1)} = z^{(\ell)} + A_{:,j_\ell}\left(x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right)$

7: $\quad \ell = \ell + 1$

8: **end while**

3.2. Global convergence of gradient based coordinatewise descent method. In this section, we show the global convergence of CD-Cyc-Grad in Theorem 2. 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 establishes the global convergence (up to measure 0 set of initial conditions) of Algorithm 2 without convergence rate.

THEOREM 2. *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 2 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. The idea of the proof follows the recent work [19]. Comparing the choice of stepsize in Theorem 2 with the CDM stepsize for some other optimization problems, for example, [34, 47], 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 (4), 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 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.

4. Greedy coordinatewise 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 a 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 moves 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.

4.1. GCD-Grad-LS and GCD-LS-LS. We first review GCD-Grad-LS, proposed in [21]. One of the most widely used greedy strategies in the "coordinate-pick" is the Gauss–Southwell rule [38],

$$(9) \qquad j_\ell = \arg \max_j \left| \nabla_j f\left(x^{(\ell)}\right) \right|,$$

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,

$$(10) \qquad \alpha_{j_\ell} = \arg \min_\alpha f\left(x^{(\ell)} + \alpha e_{j_\ell}\right).$$

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

(11) $$h'(\alpha) = 4\left(\alpha^3 + b_{j_\ell}\alpha^2 + c_{j_\ell}\alpha + d_{j_\ell}\right) = 0,$$

where the coefficients are defined as

(12)
$$b_{j_\ell} = 3x_{j_\ell}^{(\ell)}, \quad c_{j_\ell} = \left\|x^{(\ell)}\right\|^2 + 2\left(x_{j_\ell}^{(\ell)}\right)^2 - A_{j_\ell,j_\ell}, \quad \text{and} \quad d_{j_\ell} = \left\|x^{(\ell)}\right\|^2 x_{j_\ell}^{(\ell)} - z_{j_\ell}^{(\ell)}.$$

We notice that the coefficients of the cubic polynomial requires $O(n)$ operations to compute. Once the coefficients are calculated, solving (11) can be done in $O(1)$ operations. There could be multiple roots of (11), and we can use the following *root picking strategy* to find the one minimizing $h(\alpha)$:

- If there is only one root of (11), then it minimizes $h(\alpha)$.
- If there are two roots of (11), according to the property of cubic polynomial, one of them must be a 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 (10) can be achieved in $O(n)$ operations.

Algorithm 3 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 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 (11), all the $O(n)$ computational cost comes from the calculation of $\|x^{(\ell)}\|^2$. Once $\nu = \|x^{(\ell)}\|^2$ is precalculated, all coefficients in (11) can

---

Algorithm 3. GCD-Grad-LS for LEVP.

Input: Symmetric matrix $A \in \mathbb{R}^{n \times n}$; initial vector $x^{(0)}$.

1: $z^{(0)} = Ax^{(0)}$
2: $\ell = 0$
3: **while** (not converged) **do**
4: $\quad \nu = \left\|x^{(\ell)}\right\|^2$
5: $\quad j_\ell = \arg\max_j \left|\nu x_j^{(\ell)} - z_j^{(\ell)}\right|$
6: $\quad b_{j_\ell} = 3x_{j_\ell}^{(\ell)}$
7: $\quad c_{j_\ell} = \nu + 2\left(x_{j_\ell}^{(\ell)}\right)^2 - A_{j_\ell,j_\ell}$
8: $\quad d_{j_\ell} = \nu x_{j_\ell}^{(\ell)} - z_{j_\ell}^{(\ell)}$
9: $\quad$ Solve $\alpha^3 + b_{j_\ell}\alpha^2 + c_{j_\ell}\alpha + d_{j_\ell} = 0$ with the *root picking strategy* for α_{j_ℓ}
10: $\quad x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} + \alpha_{j_\ell}, & j = j_\ell \\ x_j^{(\ell)}, & j \neq j_\ell \end{cases}$
11: $\quad z^{(\ell+1)} = z^{(\ell)} + A_{:,j_\ell}\alpha_{j_\ell}$
12: $\quad \ell = \ell + 1$
13: **end while**

---

be calculated in $O(1)$ operations and hence (10) can be solved in $O(1)$ operations. Therefore, applying this to each coordinate, once ν is precalculated, solving (10) for all coordinates can be done in $O(n)$ operations. This leads us to investigate the possibility of conducting an exact line search along all coordinates,

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

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 $f\left(x^{(\ell)} + \alpha_j e_j\right)$ and $f\left(x^{(\ell)}\right)$ for all j can be done in $O(n)$ operations as well. Therefore the minimization problem of j in the second line of (13) can be solved efficiently. Through tedious but straightforward calculation, we obtain

$$
(14) \qquad \Delta f_j := f\left(x^{(\ell)} + \alpha_j e_j\right) - f\left(x^{(\ell)}\right) = \alpha_j^4 + \frac{4b_j}{3}\alpha_j^3 + 2c_j\alpha_j^2 + 4d_j\alpha_j,
$$

where b_j, c_j, and d_j are defined analogous to (11). Combining (11) and (14), we conclude that (13) is achievable in $O(n)$ operations. The corresponding method is described in Algorithm 4.

Algorithm 4. GCD-LS-LS for LEVP.

Input: Symmetric matrix $A \in \mathbb{R}^{n \times n}$; initial vector $x^{(0)}$.

1: $z^{(0)} = Ax^{(0)}$

2: $\ell = 0$

3: **while** (not converged) **do**

4: $\quad \nu = \left\|x^{(\ell)}\right\|^2$

5: $\quad$ **for** $j = 1, 2, \ldots, n$ **do**

6: $\qquad b_j = 3x_j^{(\ell)}$

7: $\qquad c_j = \nu + 2\left(x_j^{(\ell)}\right)^2 - A_{j,j}$

8: $\qquad d_j = \nu x_j^{(\ell)} - z_j^{(\ell)}$

9: $\qquad$ Solve $\alpha^3 + b_j\alpha^2 + c_j\alpha + d_j = 0$ with the *root picking strategy* for α_j

10: $\qquad \Delta f_j = \alpha_j^4 + \frac{4b_j}{3}\alpha_j^3 + 2c_j\alpha_j^2 + 4d_j\alpha_j$

11: $\quad$ **end for**

12: $\quad j_\ell = \arg\min_j \Delta f_j$

13: $\quad x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} + \alpha_j, & j = j_\ell \\ x_j^{(\ell)}, & j \neq j_\ell \end{cases}$

14: $\quad z^{(\ell+1)} = z^{(\ell)} + A_{:,j_\ell}\alpha_{j_\ell}$

15: $\quad \ell = \ell + 1$

16: **end while**

Similar to GCD-Grad-LS, the local convergence of GCD-LS-LS can be established; we will defer the local convergence analysis to the end of section 5.2.

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, (13), of escaping some of the saddle points. The iteration of GCD-LS-LS can be summarized as

$$x^{(\ell+1)} = x^{(\ell)} + \alpha_\ell e_{j_\ell}$$

(15)
$$\text{with } \alpha_\ell, j_\ell = \arg\min_{\alpha,j} f\left(x^{(\ell)} + \alpha e_j\right).$$

THEOREM 3. *Assume Assumption* A *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* $\|x^{(0)} - x^s\| < \delta_0$, *then* $f(x^{(1)}) < f(x^s)$.

Since an exact line search in all directions guarantees that $f\left(x^{(\ell+1)}\right) < f\left(x^{(\ell)}\right)$, 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

(16)
$$f(x + \beta_1 e_1) = f(x^s) + C_0 + \Delta x^\top C_1 + \Delta x^\top C_2 \Delta x + (\Delta x^\top \Delta x)\Delta x^\top C_3 + (\Delta x^\top \Delta x)^2,$$

where

(17)
$$C_0 = \beta_1^2 \left(\beta_1^2 + 4x_1^s \beta_1 + 2\lambda + 4(x_1^s)^2 - 2A_{1,1}\right) =: q(\beta_1),$$
$$C_1 = -4\beta_1 A_{:,1} + 4\beta_1 \lambda e_1 + 8\beta_1 x_1^s x^s + 4\beta_1^2 x^s + 8\beta_1 x_1^s e_1 + 4\beta_1^3 e_1,$$
$$C_2 = -2A + 4x^s (x^s)^\top + 4\beta_1 e_1 e_1^\top + 2\lambda I + 4\beta_1 e_1 (x^s)^\top + 4\beta_1 x^s e_1^\top + 4\beta_1 x_1^s I + 2\beta_1^2 I,$$
$$C_3 = 4x^s + 4\beta_1 e_1.$$

Applying the Cauchy–Schwarz inequality to the last four terms in (16), we have

(18)
$$f(x + \beta_1 e_1) \leq f(x^s) + q(\beta_1) + \|C_1\|\delta + \|C_2\|_2 \delta^2 + \|C_3\|\delta^3 + \delta^4 = f(x^s) + p(\delta),$$

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

(19)
$$q(\beta) = \beta^2 \left(\beta^2 + 4x_1^s \beta + 2\lambda + 4(x_1^s)^2 - 2A_{1,1}\right),$$

β^2 is always positive, and the discriminant of the remaining second 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 for all $0 \leq \delta < \delta_0$, $p(\delta) < 0$. Hence, following the greedy coordinatewise iteration as in (15) and $x^{(0)} = x$ for $\|x - x^s\| \leq \delta < \delta_0$, we obtain

(20)
$$f(x^{(1)}) = f(x^{(0)} + \alpha_\ell e_{j_\ell}) \leq f(x^{(0)} + \beta_1 e_1) < f(x^s).$$

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

To the authors' best knowledge, the analysis of global convergence of the greedy CDM is still open for the nonconvex objective function. Theorem 3 provides more insights on the behavior of the methods around the saddle points of (2). 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

is some analysis of the convergence of the CDM, such as [42]. However, the analysis [42] shows the convergence to coordinatewise local minima but not the general local minima.

5. Stochastic coordinatewise descent methods. The greedy CDM, including both GCD-Grad-LS and GCD-LS-LS in section 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 supercomputing clusters, shared big memory machines, or even personal laptops have multithread parallelism enabled. In order to fully use the computing resources, multicoordinate 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 the multicoordinate 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 nonconvergent iteration. Figure 1(a) demonstrates the nonconvergence 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 a 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 provably outperform 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 nonconvex region, as shown in Figure 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.
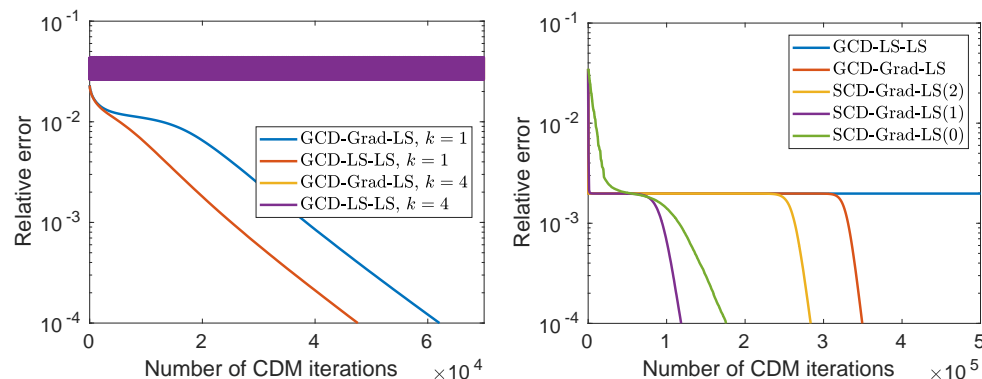
(a) The nonconvergence behavior of greedy CDMs on A_{108} with $k = 4$ comparing with convergent greedy CDMs with $k = 1$. The initial vector is e_1.

(b) The convergence behavior of greedy CDMs and stochastic CDMs on $A_{108} - 100I$. The initial vector has 100 nonzero entries on random coordinates.

FIG. 1. *Convergence behavior of greedy coordinatewise methods versus stochastic coordinatewise methods. Both* (a) *and* (b) *demonstrate the convergence behavior for solving* (2) *and the matrix* A_{108} *is a random matrix of size* 5000 *with the largest eigenvalue being* 108 *and other eigenvalues equally distributed in* $[1, 100)$. *Note that we shift the matrix in* (b) *as* $A_{108} - 100I$ *so that the only saddle point is the origin.*

5.1. SCD-Grad-vecLS and SCD-Grad-LS. The first stochastic CDM we consider is SCD-Grad-vecLS as in Algorithm 5. 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 $|c_j^{(\ell)}|^t$ for $t \geq 0$ and the set of sampled indices is denoted as Ω, where as in GCD-Grad-LS in section 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 tth power of the absolute value of the gradient vector. Through the similar derivation as in section 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 $h(\alpha) = f(x^{(\ell)} + \alpha v) = f(x^{(\ell)} + \alpha \sum_{j \in \Omega} \nabla_j f(x^{(\ell)}) e_j)$, where $v = \sum_{j \in \Omega} \nabla_j f(x^{(\ell)}) e_j$ 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,

(21)
$$
\frac{\mathrm{d}h}{\mathrm{d}\alpha}(\alpha) = 4\|v_\Omega\|^4 \alpha^3 + 12 \left(v_\Omega^\top x_\Omega^{(\ell)} \right) \|v_\Omega\|^2 \alpha^2
$$
$$
+ 4 \left(\nu\|v_\Omega\|^2 + 2 \left(v_\Omega^\top x_\Omega^{(\ell)} \right)^2 - v_\Omega^\top A_{\Omega,\Omega} v_\Omega \right) \alpha + 4\nu \left(v_\Omega^\top x_\Omega^{(\ell)} \right) - 4 v_\Omega^\top z_\Omega^{(\ell)} = 0,
$$

where $\nu = \|x^{(\ell)}\|^2$. We notice that all coefficients in (21), 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. Analogous to section 4, we adopt the *root picking strategy* to find the optimal α which has the lowest function value. We conclude that the optimal α is achievable in $O(k^2)$ operations given precalculated ν. Algorithm 5 describes the steps in detail.

One drawback of Algorithm 5 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

---

Algorithm 5. 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.

1: $z^{(0)} = Ax^{(0)}$
2: $\ell = 0$
3: **while** (not converged) **do**
4: 　　$\nu = \|x^{(\ell)}\|^2$
5: 　　**for** $j = 1, 2, \ldots, n$ **do**
6: 　　　$c_j^{(\ell)} = \nu x_j^{(\ell)} - z_j^{(\ell)}$
7: 　　**end for**
8: 　　Sample k coordinates with probability proportional to $|c^{(\ell)}|^t$, denote it as Ω
9: 　　Solve (21) with the *root picking strategy* for α_ℓ
10: 　　$x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} + \alpha_\ell \nabla_j f\left(x^{(\ell)}\right), & j \in \Omega \\ x_j^{(\ell)}, & j \notin \Omega \end{cases}$
11: 　　$z^{(\ell+1)} = z^{(\ell)} + A_{:,\Omega}\left(x_\Omega^{(\ell+1)} - x_\Omega^{(\ell)}\right)$
12: 　　$\ell = \ell + 1$
13: **end while**

Algorithm 6. 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.

1: $z^{(0)} = Ax^{(0)}$

2: $\ell = 0$

3: **while** (not converged) **do**

4:     $\nu = \left\| x^{(\ell)} \right\|^2$

5:     **for** $j = 1, 2, \ldots, n$ **do**

6:         $c_j^{(\ell)} = \nu x_j^{(\ell)} - z_j^{(\ell)}$

7:     **end for**

8:     Sample k coordinates with probability proportional to $\left| c^{(\ell)} \right|^t$; denote Ω the sampled index set

9:     **for** $j \in \Omega$ **do**

10:         $p_j = \nu - \left(x_j^{(\ell)} \right)^2 - A_{j,j}$

11:         $q_j = A_{j,j} x_j^{(\ell)} - z_j^{(\ell)}$

12:         Solve $\alpha_j^3 + p_j \alpha_j + q_j = 0$ for real α_j

13:     **end for**

14:     $x_j^{(\ell+1)} = \begin{cases} x_j^{(\ell)} + \alpha_j, & j \in \Omega \\ x_j^{(\ell)}, & j \notin \Omega \end{cases}$

15:     $z^{(\ell+1)} = z^{(\ell)} + A_{:,\Omega} \left(x_\Omega^{(\ell+1)} - x_\Omega^{(\ell)} \right)$

16:     $\ell = \ell + 1$

17: **end while**

SCD-Grad-vecLS and the coordinate updating strategy in GCD-Grad-LS. But the updating strategy in SCD-Grad-LS updates each coordinate with the coordinatewise exact line search independently and can be delayed. Algorithm 6 is the pseudocode of SCD-Grad-LS.

Updating $k > 1$ coordinates independently as in Algorithm 6 does not guarantee convergence. When k is large or t is large such that the same coordinate is updated multiple times, we do observe nonconvergent 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 $1/k$ 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

$$(22) \qquad x^{(\ell+1)} = x^{(\ell)} + \frac{1}{k} \sum_{j \in \Omega} \alpha_j e_j,$$

where α_j is the optimal stepsize in the jth coordinate as in (10). 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 $|c^{(\ell)}|^t$ is small, sampling with or without replacement behaves very similarly, while for $|c^{(\ell)}|^t$ with large variance or $k \approx n$, these two sampling strategies behave drastically differently. For example, when $t \to \infty$, probability proportional to $|c^{(\ell)}|^t$ becomes an indicator vector on a single coordinate (assuming nondegeneracy). Sampling with replacement results in a set Ω of k same

indices, whereas sampling without replacement results in 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 Algorithms 5 and 6 when $k = 1$ (the two algorithms are equivalent when $k = 1$). A similar but more complicated analysis could be done for $k \geq 1$ when sampling with replacement is adopted and the modified updating strategy (22) is adopted in Algorithm 6.

5.2. Local convergence of stochastic coordinatewise descent method. In this section, we analyze the convergence properties of stochastic CDMs 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 notation and definitions. We take

$$(23) \qquad B^{\pm} = \left\{ y \,\middle|\, \left\| y \mp \sqrt{\lambda_1} v_1 \right\| \leq \frac{1}{30} \frac{\min(2\lambda_1, \lambda_1 - \lambda_2)}{\sqrt{\lambda_1}} \right\}$$

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

DEFINITION 5.1 (coordinatewise Lipschitz continuous). *Let a function $g(x)$: $\mathbb{S} \mapsto \mathbb{R}$ be continuously differentiable. The gradient of the function $\nabla g(x)$ is coordinatewise Lipschitz continuous on $\mathbb{S}$ if there exists a positive constant L such that*

$$(24) \qquad |\nabla_i g(x + \alpha e_i) - \nabla_i g(x)| \leq L|\alpha| \quad \forall x, x + \alpha e_i \in \mathbb{S} \text{ and } i = 1, 2, \ldots, n.$$

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 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 coordinatewise Lipschitz continuous function $g(x)$ is that

$$(25) \qquad g(x + \alpha e_i) \leq g(x) + \nabla_i g(x)\alpha + \frac{L}{2}\alpha^2 \quad \forall x \in \mathbb{S} \text{ and } x + \alpha e_i \in \mathbb{S}.$$

The following lemma extends [21, Lemma A.3] to objective function $f(x)$ with matrix A satisfying Assumption A.

LEMMA 5.2. *Function $f(x)$ defined in (2) is a continuously differentiable function. Further the gradient function $\nabla f(x)$ is coordinatewise 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 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*

$$(26) \qquad g(y) \geq g(x) + \nabla g(x)^\top (y - x) + \frac{\mu_p}{2} \|y - x\|_p^2 \quad \forall x, y \in \mathbb{S}.$$

Throughout the paper, we assume $p \geq 1$ in the definition. Definition 5.3 is a generalized version of the traditional strong convexity, which corresponds to the $p = 2$ case.

Combining (26) with the equivalence of different norms in finite dimensional vector space, we obtain for any $p \geq q \geq 1$

$$(27) \qquad \begin{aligned} g(y) &\geq g(x) + \nabla g(x)^\top (y - x) + \frac{\mu_p}{2} \|y - x\|_p^2 \\ &\geq g(x) + \nabla g(x)^\top (y - x) + n^{2/p - 2/q} \frac{\mu_p}{2} \|y - x\|_q^2 \quad \forall x, y \in \mathbb{S}. \end{aligned}$$

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, using the inequality of vector norm together with (26), we obtain for any $p \geq q \geq 1$,

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

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:

$$
(29) \quad n^{2/p-2/q}\mu_p \leq \mu_q \leq \mu_p,
$$

where $p \geq q \geq 1$.

LEMMA 5.4. *Function $f(x)$ defined in (2) 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 5.4 follows an extension of the proof of Lemma A.2 in [21], where the minimum eigenvalue of the Hessian matrix is modified according to the assumption of the matrix A. Combining with (29), we have the existence of μ_p for all $p \geq 1$.

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

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

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 5.5 shows the monotonic decay property of the iteration defined by Algorithm 6 once the iterations falls in $D^\pm$. It also shows that $D^+ \cup D^-$ is a contraction set for the iteration.

LEMMA 5.5. *Consider function $f(x)$ as defined in (2) and the iteration follows Algorithm 6 with $k = 1$. For any $x^{(\ell)} \in D^+ \cup D^-$, we have*

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

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 5.5 can be found in Appendix B.

In Algorithm 6, 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 nonnegative power t. In the following lemma and theorem, we adopt notation f^* as the minimum of the function and X^* as 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 $\mathrm{dist}(S_1, S_2) = \min_{x \in S_1, y \in S_2} \|x - y\|$.

LEMMA 5.6. *Consider function $f(x)$ as defined in* (2) *and the iteration follows Algorithm 6 with $k = 1$. For any $x^{(\ell)} \in D^+ \cup D^-$,*

$$(32) \qquad \mathbb{E}\left[f\left(x^{(\ell+1)}\right)\Big|x^{(\ell)}\right] - f^* \le \left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)\left(f(x^{(\ell)}) - f^*\right),$$

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

The proof of Lemma 5.6 can be found in Appendix B.

THEOREM 4. *Consider function $f(x)$ as defined in* (2) *and the iteration follows Algorithm 6 with $k = 1$. For any $x^{(0)} \in D^+ \cup D^-$,*

$$(33) \qquad \mathbb{E}\left[f(x^{(\ell)})\Big|x^{(0)}\right] - f^* \le \left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)^\ell \left(f(x^{(0)}) - f^*\right),$$

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

$$(34) \qquad \mathbb{E}\left[\operatorname{dist}\left(x^{(\ell)}, X^*\right)^2 \Big|x^{(0)}\right] \le \frac{2}{\mu_2}\left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)^\ell \left(f(x^{(0)}) - f^*\right).$$

The proof of Theorem 4 can be found in Appendix B. Theorem 4 is slightly more complicated than Lemma 5.6 since Algorithm 6 adopts the exact line search and there are two sublevel sets $D^\pm$ for the nonconvex 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 5.6 to Theorem 4.

Remark 5.7. According to Theorem 4, the convergence rate depends on $n^{2/q}\mu_q$, which depends on q and hence t. The right side of (29) 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 the greedy CDM could be potentially n times faster than the stochastic CDM with uniform sampling. According to the left side of inequality (29), for $p \ge q$, we have $n^{2/p}\mu_p \le n^{2/q}\mu_q$, 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 5.8. For a fair comparison between CDMs with traditional methods such as gradient descent, we should take n iterations of CDM and compare them against a single iteration of gradient descent since one iteration of CDM updates only one coordinate whereas one iteration of gradient descent updates n coordinates. Under such a setting, Theorem 4 implies that

$$\mathbb{E}\left[f(x^{(n\ell)})\Big|x^{(0)}\right] - f^* \le \left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)^{n\ell}\left(f(x^{(0)}) - f^*\right)$$

$$\le \left(e^{-\frac{\mu_q}{L}n^{\frac{2}{q}-1}}\right)^\ell \left(f(x^{(0)}) - f^*\right)$$

and

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

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 6 with uniform sampling has the same convergence rate as gradient descent. Combining with Remark 5.7, we conclude that Algorithm 6 with arbitrary $t \geq 0$ converges at least as fast as gradient descent.

Remark 5.9. If we extend Definition 5.1 to multicoordinate Lipschitz continuity and assume $f(x)$ is multicoordinate Lipschitz continuous with $\widetilde{L}$, a natural extension of Theorem 4 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 4. In the end, the convergence rate would not be improved and could be even worse. This is similar to the argument of CDM versus the full gradient descent method. In practice, when k is not too large, and the sampled coordinates are distinct, we do observe k-fold speedup of the iterations. This observation suggests that the convergence rate in Theorem 4 is not sharp for $k > 1$.

Combining Theorem 4 together with the monotonicity of μ_q as (29) 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 (29) 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 $\nabla^2 f(x) = 2A^2$, $\mu_0 = 2$, and $\mu_\infty \approx 1.99$. Therefore all μ_t lies in the small interval $[1.99, 2]$ and a different choice of t would lead to a different rate of convergence. We minimize $f(x)$ using the SCD-Grad-Grad method. The stepsize is fixed to be 10^{-6} and the initial vector is chosen randomly. Figure 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, a 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 2(a), the line with $t = 4096$ and that with $t = \infty$ overlap.
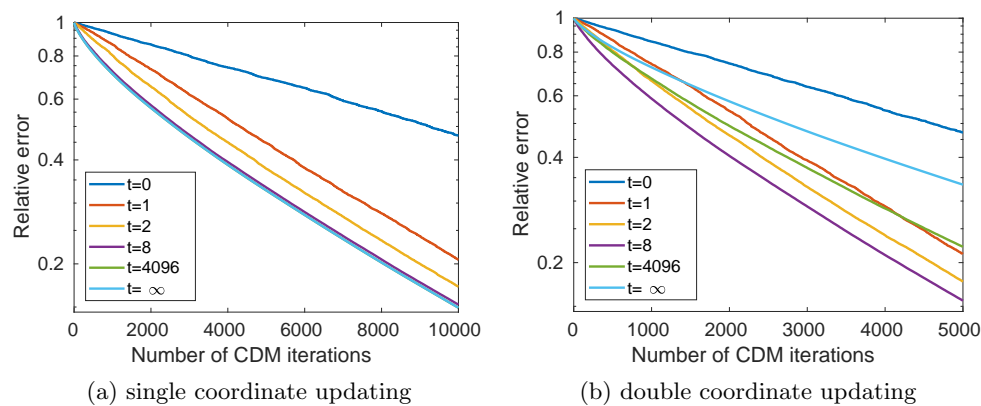

(a) single coordinate updating   (b) double coordinate updating

FIG. 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).

According to either Theorem 4 or Figure 2(a), it is convincing that for single-coordinate updating CDM, picking the index with the largest gradient magnitude should be the optimal strategy. Figure 2(b) investigates the case with double coordinate updating, i.e., sample two coordinates independently and update two entries simultaneously in every iteration. Comparing Figures 2(a) and (b), we notice for small t, the double coordinate updating strategy almost reduces the required number of iterations by half. As $t \to \infty$, the sampling approaches choosing the coordinate with the largest gradient magnitude with probability 1. Sampling two coordinates independently thus results in sampling the same index twice and the iteration behaves similarly 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 nonconvex, for example, the objective function $f(x)$ defined in (2), the iteration usually sticks in the middle of the iteration (see Figure 1(a)).

Remark 5.10 (local convergence of greedy CDM). The local convergence analysis of stochastic CDMs can be extended to the greedy CDMs. GCD-Grad-LS as in Algorithm 4 is a special case of SCD-Grad-LS with $t = \infty$ and $k = 1$.[3] Hence Theorem 4 proves the local convergence of the GCD-Grad-LS with $q = 1$.

GCD-LS-LS as in Algorithm 4 conducts optimal coordinatewise descent every iteration. Therefore, for GCD-LS-LS, Lemma 5.5 can be proved for any index of coordinate j and then Lemma 5.6 holds for $q = 1$. The modified lemmas lead to the following corollary.

COROLLARY 5.11. *Consider function* $f(x)$ *as defined in* (2) *and the iteration follows Algorithm* 4. *For any* $x^{(0)} \in D^+ \cup D^-$,

$$(35) \qquad \mathbb{E}\left[f\big(x^{(\ell)}\big) \Big| x^{(0)} \right] - f^* \leq \left(1 - \frac{\mu_1}{L} \right)^\ell \left(f\big(x^{(0)}\big) - f^* \right).$$

Moreover,

$$(36) \qquad \mathbb{E}\left[\mathrm{dist}\left(x^{(\ell)}, X^* \right)^2 \Big| x^{(0)} \right] \leq \frac{2}{\mu_2} \left(1 - \frac{\mu_1}{L} \right)^\ell \left(f\big(x^{(0)}\big) - f^* \right).$$

The local convergences 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 of the cases we tested.

Remark 5.12 (convergence behavior in nonconvex area). We demonstrate one example such that the stochastic CDMs outperform greedy CDMs even when all methods are conducting single coordinate updating. We generate a random matrix $A_{108} - 100I$ of size 5000, whose largest eigenvalue is 8 and other eigenvalues are distributed equally on $[-99, 0)$. According to Lemma 2.5, 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 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, it takes more time for the greedier method to escape from the saddle point. And

---

[3]Previous work [21] provides a local convergence proof for GCD-Grad-LS, which uses the theorem in [29, 30]. 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 (30) is local domain of contraction and can be used to fill in the gap of the proof.

although SCD-Grad-LS(0) escapes from the saddle point fastest, 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 2. They are also robust to nonconvex objective functions near saddle points.

6. Numerical results. In this section, we perform numerical tests of the methods mentioned in previous sections 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 calculation. The ground state problem in a quantum many-body system is equivalent to finding 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 are 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 coordinatewise methods against the 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 (2), 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

$$(37) \qquad \epsilon_{\mathrm{obj}} = \sqrt{\frac{f\left(x^{(\ell)}\right) - f^*}{f^*}},$$

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

$$(38) \qquad E = \frac{x_*^\top A x^{(\ell)}}{x_*^\top x^{(\ell)}},$$

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 to as the projected energy in the context of quantum chemistry. Thus the relative error is define by

$$(39) \qquad \epsilon_{\mathrm{energy}} = \frac{|E - \lambda_1|}{|\lambda_1|}.$$

Last,

$$(40) \qquad \epsilon_{\mathrm{tan}} = \tan \theta \left(v^{(\ell)}, v_1\right)$$

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 (FCI method) because the matrix is

TABLE 3
Performance of various algorithms for A_{108}.

Method	k	Min Iter Num	Med Iter Num	Max Iter Num	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

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.

6.1. Dense random matrices. We first show the advantage of the CDMs over the power method on dense random matrices. All matrices tested in this section involve symmetric matrices of size 5000 with random eigenvectors, i.e.,

$$(41) \qquad A_{\lambda_1} = Q \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_{5000} \end{bmatrix} Q^\top,$$

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 a 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 $A_{108} + 1000I$. And the numerical results are reported in Tables 3, 4, and 5, respectively. 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 (2), it is also more difficult since the landscape of the objective function becomes steeper, while, in practice, CDMs for the optimization problem are less sensitive to the shift. In this section, all results are reported in terms

Table 4
Performance of various algorithms for A_{101}.

Method	k	Min Iter Num	Med Iter Num	Max Iter Num	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

of number of column accesses, converging to 10^{-6} under the measure of ϵ_{obj}. For nonstochastic methods, we report the number of column accesses from a single run in the column named "Med Iter Num," 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 evaluations, which is calculated as the product of the 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 coordinatewise 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 comparably to GCD-Grad-LS and GCD-LS-LS, while SI-GSL($1.5\lambda_1$) is significantly 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 are about 70 times faster. The two greedy methods themselves have similar cost. For the stochastic methods, we use gradient information

TABLE 5
Performance of various algorithms for $A_{108} + 1000I$.

Method	k	Min Iter Num	Med Iter Num	Max Iter Num	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		

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-Grad-vecLS and SCD-Grad-LS share a similar performance. For $t = 0$, which is equivalent to uniform sampling, the number of column accesses is almost the same as the steepest gradient descent. It is reasonable since uniform sampling does not pick the important coordinate and should behave similarly to the full gradient. For $t = 1$ and $t = 2$, the speedup is obvious. The performance for different t agrees with Theorem 4, 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.

6.2. Hubbard models. In this test, we calculate the ground energy and ground state of the 2D Hubbard model. The fermion Hubbard model is widely used in condensed matter physics, which models interacting fermion particles in a lattice with the nearest neighbor hopping and on-site interaction. It is defined by the Hamiltonian in the second quantization notation:

$$(42) \qquad \hat{H} = -t \sum_{\langle r,r' \rangle, \sigma} \hat{c}_{r,\sigma}^\dagger \hat{c}_{r',\sigma} + U \sum_r \hat{n}_{r\uparrow} \hat{n}_{r\downarrow},$$

where t is the hopping strength and U is the on-site interaction strength. r, r' are indexes of sites in the lattice, and $\langle r, r' \rangle$ means r and r' are the nearest neighbor. The spin σ takes value of $\uparrow$ and $\downarrow$. The operators $\hat{c}_{r,\sigma}^\dagger$ and $\hat{c}_{r,\sigma}$ are creation and annihilation operators, which create or destroy a fermion at site r with spin σ. They satisfy the canonical anticommutation relation:

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

where $\{A, B\} = AB + BA$ is the anticommutator. $\hat{n}_{r,\sigma} = \hat{c}_{r,\sigma}^\dagger \hat{c}_{r,\sigma}$ is the number operator.

When the interaction is not strong, we usually work in the momentum space, because in the momentum space, the hopping term in the Hamiltonian is diagonalized and the ground state is close to the Hatree–Fock (HF) state. Mathematically speaking, the normalized leading eigenvector v_1 is close to the unit vector e_{HF} whose nonzero entry indicates the HF state. The Hamiltonian in momentum space can be written as

$$(43) \qquad \hat{H} = t \sum_{k,\sigma} \varepsilon(k) \hat{n}_{k,\sigma} + \frac{U}{N^{\mathrm{orb}}} \sum_{k,p,q} \hat{c}_{p-q,\uparrow}^\dagger \hat{c}_{k+q,\downarrow}^\dagger \hat{c}_{k,\downarrow} \hat{c}_{p,\uparrow},$$

where $\varepsilon(k) = -2(\cos(k_1) + \cos(k_2))$ and N^{orb} is the number of orbitals. In momentum space, k is the wave number and the creation and annihilation operator can be obtained by the transform $\hat{c}_{k,\sigma} = \frac{1}{\sqrt{N^{\mathrm{orb}}}} \sum_r e^{ik \cdot r} \hat{c}_{r,\sigma}$.

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 6. The nonzero off-diagonal entries of H take value $\pm \frac{U}{N^{\mathrm{orb}}} = \pm 0.25$, and the diagonal entries distribute in $(-20, 30)$ with a 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 $10 \, e_{\mathrm{HF}}$ for all calculations; 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 6
Properties of the Hubbard Hamiltonian.

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

TABLE 7
Performance of various algorithms for 4×4 Hubbard model with 6 electrons.

| Method	k	Min Iter Num	Med Iter Num	Max Iter Num	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		

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 7.

In Table 7, 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. Stochastic CDMs also outperform PM and CPM. Comparing with the greedy CDMs, they are about 3 to 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 stepsize, if used, is chosen to be 2. This stepsize is larger than that in Theorem 2, but numerically the iteration converges. All methods besides PM update a single coordinate per iteration and run up to 10^7 column accesses or desired accuracy. Stochastic CDMs sample coordinates with respect to the magnitude of the gradient, i.e., $t = 1$. The convergences of the square root of the relative error of objective function value ϵ_{obj}, eigenvalue estimator $\epsilon_{\mathrm{energy}}$, and eigenvector estimator ϵ_{tan} are plotted in Figure 3.

From Figure 3, we see that the convergences 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 consists of one or two stages: a possible fast decay followed by linear convergence. We know that
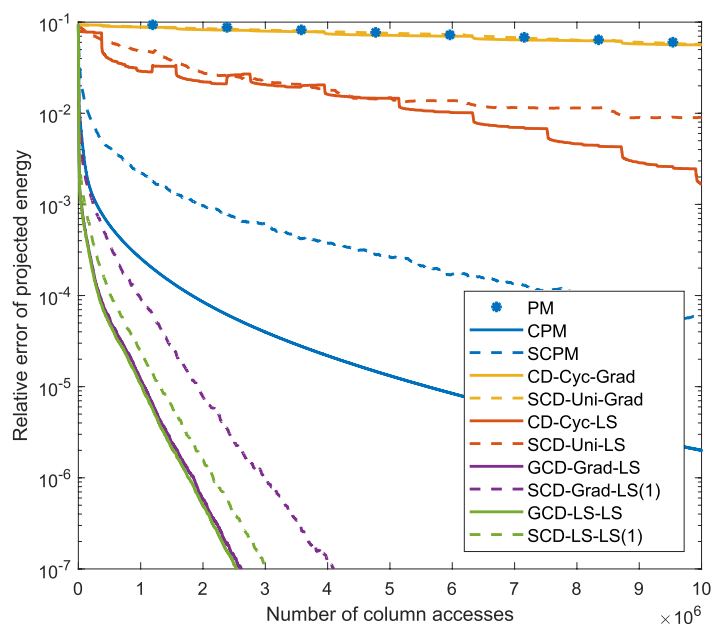
(a) ϵ_{energy}, Relative error of projected energy
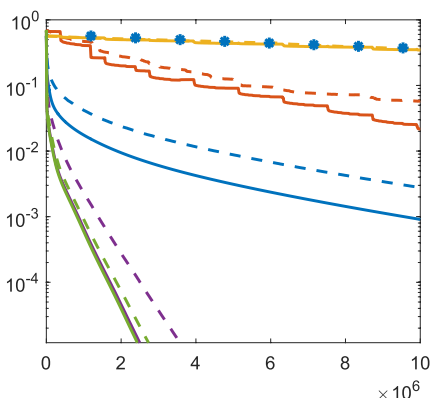
(b) $\epsilon_{\text{tangent}}$, Tangent of angle between $v^{(\ell)}$ and v_1
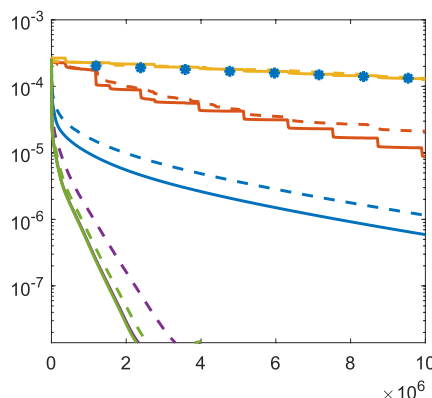
(c) ϵ_{obj}, Relative error of objective function value

FIG. 3. *Convergence behavior of the methods for the* 4×4 *Hubbard model with* 10 *electrons.* $k = 1$ *for all methods except PM.*

both 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 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 stepsize for all coordinates and all iterates, which has to be small and thus is 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 stepsize 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 CDMs. 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 iterates only three 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 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 one 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 minimizer, updating by LS is almost the same as updating by Grad with some stepsize depending on the Hessian at the minimizer. Thus, updating by Grad could perform as well as LS ideally if there exists a stepsize which both guarantees the convergence and converges fast. On the other hand, LS chooses an optimal stepsize every iteration.

7. Conclusion. We analyze the landscape of the nonconvex objective function $f(x)$ of LEVP as (2) 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 the convergence rate of either GCD-LS-LS or SCD-Grad-LS(t) with $t > 0$ is provably faster than or equal to that of SCD-Uni-LS. And one example as in Figure 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 have great potential for quantum many-body calculations.

There are several directions worth exploring for future works. Momentum acceleration for coordinate descent [2, 20, 22, 29] can be combined with the proposed 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 proposed 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. Last but not least, the proposed methods should be extensively tested on other quantum many-body calculations and more other areas whose computational bottlenecks are the LEVP.

Appendix A. Proof of Theorem 2. 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 2.5. Let $g_j(x)$ be the "coordinate-updating" in Algorithm 2, mapping from $x^{(\ell)}$ to $x^{(\ell+1)}$ with coordinate j, i.e.,

$$(44) \qquad x^{(\ell+1)} = g_j\left(x^{(\ell)}\right).$$

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

$$(45) \qquad g = g_n \circ g_{n-1} \circ \cdots \circ g_1.$$

The corresponding iteration then is

$$(46) \qquad x^{((k+1)n)} = g\left(x^{(kn)}\right), \qquad k = 0, 1, 2, \dots.$$

We first establish a few lemmas for the proof of Theorem 2.

LEMMA A.1. *Let* $R \geq \sqrt{\max_j \|A_{\cdot,j}\|}$ *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.,*

$$(47) \qquad x^{(1)} = g_j\left(x^{(0)}\right) = x^{(0)} - \gamma\left(-4z_j^{(0)} + 4\left\|x^{(0)}\right\|^2 x_j^{(0)}\right) e_j \in W_0,$$

where g_j *is defined as* (44).

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

$$(48) \qquad \left| x_j^{(0)} - \gamma\left(-4z_j^{(0)} + 4\left\|x^{(0)}\right\|^2 x_j^{(0)}\right) \right| < R.$$

In the rest of the proof, we drop the superscript (0) to simplify the notation. We denote the expression in the absolute value in (48) as

$$(49) \qquad h\left(x_j\right) = -4\gamma x_j^3 + \left(1 + 4\gamma A_{j,j} - 4\gamma \sum_{i \neq j} x_i^2\right) x_j + 4\gamma \sum_{i \neq j} A_{i,j} x_i,$$

which is a cubic polynomial. Showing (48) is equivalent to showing that $-R < h(x) < R$ for any $x \in (-R, R)$. The local maximizer and minimizer of $h(x)$ are the roots of $h'(x)$,

$$(50) \qquad x^\pm = \pm\sqrt{\frac{1}{12\gamma} + \frac{A_{j,j}}{3} - \frac{\sum_{i \neq j} x_i^2}{3}},$$

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

$$(51) \qquad \left|x^\pm\right| \geq \sqrt{R^2 + \frac{R^2 + A_{j,j}}{3} + \frac{nR^2 - \sum_{i \neq j} x_i^2}{3}} > R,$$

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

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

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. □

LEMMA A.2. *For any* $x \in W_0$,

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

where W_0 *and* R *are as defined in Lemma* A.1, *and* f *is the objective function as in* (2).

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

$$
\left| e_i^\top \nabla^2 f(x) e_j \right| = \left| -4A_{i,j} + 8x_i x_j + 4\|x\|^2 \right| < 4|A_{i,j}| + 8R^2 + 4nR^2 < 4(n+3)R^2.
$$
(54) □

One direct consequence of Lemma A.2 is that for any consecutive iterations in either CD-Cyc-Grad, we have the relation

$$
(55) \qquad f\left(x^{(\ell+1)}\right) \leq f\left(x^{(\ell)}\right) + \nabla_{j_\ell} f\left(x^{(\ell)}\right) \left(x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)}\right) + \frac{L}{2} \left| x_{j_\ell}^{(\ell+1)} - x_{j_\ell}^{(\ell)} \right|^2,
$$

where j_ℓ is the chosen coordinate at the ℓth 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.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*

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

Proof. Substituting the updating expression in Algorithm 2 into (55), we obtain

$$
(57) \qquad f\left(x^{(\ell+1)}\right) \leq f\left(x^{(\ell)}\right) - \gamma \left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2 + \frac{L\gamma^2}{2} \left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2.
$$

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

$$
(58) \qquad \left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2 \leq \frac{1}{\gamma\left(1 - \frac{\gamma L}{2}\right)} \left(f\left(x^{(\ell)}\right) - f\left(x^{(\ell+1)}\right)\right).
$$

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

$$
\begin{aligned}
(59) \quad \sum_{k=0}^{K} \sum_{\ell=nk}^{n(k+1)-1} \left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2 &\leq \frac{1}{\gamma\left(1 - \frac{\gamma L}{2}\right)} \left(f\left(x^{(0)}\right) - f\left(x^{(T)}\right)\right) \\
&\leq \frac{1}{\gamma\left(1 - \frac{\gamma L}{2}\right)} \left(f\left(x^{(0)}\right) - f^*\right),
\end{aligned}
$$

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

$$(60) \qquad \left|\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right| \leq \delta_0, \qquad \ell = kn, \ldots, (k+1)n - 1.$$

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 coordinatewise gradient, and without loss of generality, we assume $\ell_1 \leq \ell_2$,

$$
\begin{aligned}
\left|\nabla_{j_{\ell_1}} f\left(x^{(\ell_1)}\right) - \nabla_{j_{\ell_1}} f\left(x^{(\ell_2)}\right)\right| &\leq \sum_{\ell=\ell_1}^{\ell_2-1} \left|\nabla_{j_{\ell_1}} f\left(x^{(\ell)}\right) - \nabla_{j_{\ell_1}} f\left(x^{(\ell+1)}\right)\right| \\
&\leq L \sum_{\ell=\ell_1}^{\ell_2-1} \left|x^{(\ell)} - x^{(\ell+1)}\right| \\
&\leq L \sum_{\ell=\ell_1}^{\ell_2-1} \left|\gamma \nabla_{j_\ell} f\left(x^{(\ell)}\right)\right| \leq n\delta_0,
\end{aligned}
$$
(61)

where the last inequality is due to (60) and $\gamma L \leq 1$. Let ℓ_0 be an iteration within kn and $(k+1)n - 1$. Another important point in Algorithm 2 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,

$$
\begin{aligned}
\left\|\nabla f\left(x^{(\ell_0)}\right)\right\|^2 &= \sum_{j=1}^{n} \left(\nabla_j f\left(x^{(\ell_0)}\right)\right)^2 \\
&= \sum_{\ell=nk}^{(k+1)n-1} \left(\nabla_{j_\ell} f\left(x^{(\ell_0)}\right) - \nabla_{j_\ell} f\left(x^{(\ell)}\right) + \nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2 \\
&\leq \sum_{\ell=nk}^{(k+1)n-1} \left(\left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2 + 2\delta_0 \left|\nabla_{j_\ell} f\left(x^{(\ell_0)}\right) - \nabla_{j_\ell} f\left(x^{(\ell)}\right)\right| \right. \\
&\qquad \left. + \left|\nabla_{j_\ell} f\left(x^{(\ell_0)}\right) - \nabla_{j_\ell} f\left(x^{(\ell)}\right)\right|^2\right) \\
&\leq \left(n + 2n^2 + n^3\right) \delta_0^2.
\end{aligned}
$$
(62)

Since δ_0 can be arbitrarily 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 2. Since $R \geq \sqrt{\max_j \|A_{:,j}\|}$ and $\gamma \leq \frac{1}{4(n+4)R^2}$, Lemma A.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 (45). Further, as stated by Lemma A.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 [19] shows that under these conditions, $\det(Dg(x)) \neq 0$. Corollary 1 in [19] shows that $\mu\left(\left\{ x^{(0)} \mid \lim_{k \to \infty} g^k(x^{(0)}) \in \chi^s \right\}\right) = 0$. Combining with the conclusion of Lemma A.3, we obtain the conclusion of Theorem 2. □

Appendix B. Proof of local convergence.

Proof of Lemma 5.5. For any $x^{(\ell)} \in D^+ \subset B^+$ and fixed index j, $f\left(x^{(\ell)}\right)$ is a quartic monic polynomial of $x_j^{(\ell)}$, denoted as $p(x_j^{(\ell)})$. Since $f\left(x^{(\ell)}\right) = p(x_j^{(\ell)})$, $\mathcal{I} = \{y : p(y) \leq f\left(x^{(\ell)}\right)\}$ is a nonempty set.

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

$$(63) \qquad f\left(x^{(\ell+1)}\right) \leq f\left(x^{(\ell)}\right) - \frac{1}{2L}\left(\nabla_j f\left(x^{(\ell)}\right)\right)^2.$$

When $p'(x_j^{(\ell)}) = \nabla_j f\left(x^{(\ell)}\right) < 0$, there exists an interval $[a,b]$ with $a < b$ such that $p(a) = p(b) = f\left(x^{(\ell)}\right)$ and $p(y) < f\left(x^{(\ell)}\right)$ for all $y \in (a,b)$. It can be further confirmed that $a = x_j^{(\ell)}$. By the 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

$$(64) \qquad 0 < -\nabla_j f\left(x^{(\ell)}\right) = -p'(a) = p'(c) - p'(a) = (c-a)p''(\xi) \leq (c-a)L,$$

which means

$$(65) \qquad a - \frac{\nabla_j f\left(x^{(\ell)}\right)}{L} = x_j^{(\ell)} - \frac{\nabla_j f\left(x^{(\ell)}\right)}{L} \in (a,c).$$

Equation (65) implies $x^{(\ell)} - \frac{1}{L}\nabla_j f\left(x^{(\ell)}\right)e_j \in D^+$. Following the Lipschitz condition of $\nabla_j f$ in D^+, we obtain

$$(66) \qquad \begin{aligned} f\left(x^{(\ell+1)}\right) &\leq f\left(x^{(\ell)} - \frac{1}{L}\nabla_j f\left(x^{(\ell)}\right)e_j\right) \\ &\leq f\left(x^{(\ell)}\right) - \frac{1}{L}\left(\nabla_j f\left(x^{(\ell)}\right)\right)^2 + \frac{L}{2}\left(\frac{1}{L}\nabla_j f\left(x^{(\ell)}\right)\right)^2 \\ &= f\left(x^{(\ell)}\right) - \frac{1}{2L}\left(\nabla_j f\left(x^{(\ell)}\right)\right)^2. \end{aligned}$$

Similarly, (66) can be obtain when $p'(x_j^{(\ell)}) = \nabla_j f\left(x^{(\ell)}\right) > 0$ as the case of $p'(x_j^{(\ell)}) = \nabla_j f\left(x^{(\ell)}\right) < 0$.

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 1. Hence, $x^{(\ell+1)} \in D^+ \cup D^-$. $\qquad\square$

Proof of Lemma 5.6. By Lemma 5.5, we have

$$(67) \qquad f\left(x^{(\ell+1)}\right) \leq f\left(x^{(\ell)}\right) - \frac{1}{2L}\left(\nabla_{j_\ell} f\left(x^{(\ell)}\right)\right)^2,$$

where j_ℓ denotes the index picked at the ℓth iteration. Subtracting f^* from both sides of (67) and taking the conditional expectation, we obtain

$$\mathbb{E}\left[f\left(x^{(\ell+1)}\right)\Big|x^{(\ell)}\right] - f^* \le f\left(x^{(\ell)}\right) - f^* - \frac{1}{2L}\mathbb{E}\left[\left(\nabla_{j_\ell}f\left(x^{(\ell)}\right)\right)^2\Big|x^{(\ell)}\right]$$

(68)
$$= f\left(x^{(\ell)}\right) - f^* - \frac{1}{2L}\sum_{j=1}^{n}\frac{\left|\nabla_j f\left(x^{(\ell)}\right)\right|^{t+2}}{\left\|\nabla f\left(x^{(\ell)}\right)\right\|_t^t}$$

$$= f\left(x^{(\ell)}\right) - f^* - \frac{1}{2L}\frac{\left\|\nabla f\left(x^{(\ell)}\right)\right\|_{t+2}^t}{\left\|\nabla f\left(x^{(\ell)}\right)\right\|_t^t}\left\|\nabla f\left(x^{(\ell)}\right)\right\|_{t+2}^2$$

$$\le f\left(x^{(\ell)}\right) - f^* - \frac{1}{2Ln^{\frac{2}{t+2}}}\left\|\nabla f\left(x^{(\ell)}\right)\right\|_{t+2}^2,$$

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

Next, we will bound the last term in (68) with the strongly $\|\cdot\|_{\frac{t+2}{t+1}}$-convex property of $f(x)$. Minimizing both sides of (26) with respect to y and substituting $x = x^{(\ell)}$, we have

(69)
$$f^* \ge f\left(x^{(\ell)}\right) - \sup_{y}\left\{-\nabla f\left(x^{(\ell)}\right)^\top\left(y - x^{(\ell)}\right) - \frac{\mu_q}{2}\left\|y - x^{(\ell)}\right\|_{\frac{t+2}{t+1}}^2\right\}$$

$$\ge f\left(x^{(\ell)}\right) - \frac{1}{2\mu_q}\left\|\nabla f\left(x^{(\ell)}\right)\right\|_{t+2}^2,$$

where $q = \frac{t+2}{t+1}$ and the last supreme term in the first line is a conjugate of function $\frac{\mu_q}{2}\|\cdot\|_{\frac{t+2}{t+1}}^2$ and the result is given by Example 3.27 in [3], which is $\frac{1}{2\mu_q}\|\cdot\|_{t+2}^2$. Here $\|\cdot\|_{t+2}$ is the dual norm of $\|\cdot\|_{\frac{t+2}{t+1}}$.

Substituting (69) into (68), we obtain

(70)
$$\mathbb{E}\left[f\left(x^{(\ell+1)}\right)\Big|x^{(\ell)}\right] - f^* \le \left(1 - \frac{\mu_q}{Ln^{2-2\frac{t+1}{t+2}}}\right)\left(f\left(x^{(\ell)}\right) - f^*\right)$$

$$= \left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)\left(f\left(x^{(\ell)}\right) - f^*\right). \qquad \square$$

Proof of Theorem 4. By Lemma 5.5, we know that if $x^{(0)} \in D^+ \cup D^-$, then $x^{(\ell)} \in D^+ \cup D^-$ for all ℓ. Hence Lemma 5.6 holds for all $x^{(\ell)} \in D^+ \cup D^-$. We take the expectation conditioned on the sigma algebra of $x^{(0)}$,

(71)
$$\mathbb{E}\left[f\left(x^{(\ell)}\right)\Big|x^{(0)}\right] - f^* \le \left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)\left(\mathbb{E}\left[f\left(x^{(\ell-1)}\right)\Big|x^{(0)}\right] - f^*\right)$$

$$\le \cdots \le \left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)^\ell\left(f\left(x^{(0)}\right) - f^*\right),$$

where we adopt the property of conditional expectation.

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

$$
\begin{aligned}
\mathbb{E}\left[\operatorname{dist}\left(x^{(\ell)}, X^*\right)^2 \Big| x^{(0)}\right] &\leq \mathbb{E}\left[\left\|x^{(\ell)} - x^*\right\|^2 \Big| x^{(0)}\right] \\
&\leq \frac{2}{\mu_2} \mathbb{E}\left[f\left(x^{(\ell)}\right) - f(x^*) \Big| x^{(0)}\right] \\
&\leq \frac{2}{\mu_2}\left(1 - \frac{\mu_q}{Ln^{2-\frac{2}{q}}}\right)^\ell \left(f\left(x^{(0)}\right) - f^*\right).
\end{aligned}
$$

(72)

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

Acknowledgments. We thank Stephen J. Wright, Weitao Yang, and Wotao Yin for helpful discussions.

REFERENCES

[1] M. ALAEE KERAHROODI, A. AUBRY, A. DE MAIO, M. M. NAGHSH, AND M. MODARRES-HASHEMI, *A coordinate-descent framework to design low PSL/ISL sequences*, IEEE Trans. Signal Process., 65 (2017), pp. 5942–5956.

[2] Z. ALLEN-ZHU, Z. QU, P. RICHTÁRIK, AND Y. YUAN, *Even faster accelerated coordinate descent using non-uniform sampling*, in Proceedings of the 33rd International Conference on Machine Learning, 2016, pp. 1–9.

[3] S. BOYD AND L. VANDENBERGHE, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.

[4] Y. T. CHOW, T. WU, AND W. YIN, *Cyclic coordinate-update algorithms for fixed-point problems: Analysis and applications*, SIAM J. Sci. Comput., 39 (2017), pp. A1280–A1300.

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

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

[7] D. A. D'ESOPO, *A convex programming procedure*, Nav. Res. Logist. Q., 6 (1959), pp. 33–42.

[8] R. GE, F. HUANG, C. JIN, AND Y. YUAN, *Escaping from saddle points—online stochastic gradient for tensor decomposition*, J. Mach. Learn. Res., 40 (2015), pp. 1–46.

[9] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 4th ed., Johns Hopkins University Press, Baltimore, MD, 2013.

[10] M. GURBUZBALABAN, A. OZDAGLAR, P. A. PARRILO, AND N. VANLI, *When cyclic coordinate descent outperforms randomized coordinate descent*, Adv. Neural Inf. Process. Syst. 30, (2017), pp. 6999–7007.

[11] N. HALKO, P.-G. MARTINSSON, AND J. A. TROPP, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Rev., 53 (2011), pp. 217–288.

[12] J. HAN, K. S. SONG, J. KIM, AND M. G. KANG, *Permuted coordinate-wise optimizations applied to Lp-regularized image deconvolution*, IEEE Trans. Image Process., 27 (2018), pp. 3556–3570.

[13] M. HONG, X. WANG, M. RAZAVIYAYN, AND Z.-Q. LUO, *Iteration complexity analysis of block coordinate descent methods*, Math. Program., 163 (2017), pp. 85–114.

[14] C. JIN, R. GE, P. NETRAPALLI, S. M. KAKADE, AND M. I. JORDAN, *How to escape saddle points efficiently*, in Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR, 2017, pp. 1724–1732.

[15] T. B. JOHNSON AND C. GUESTRIN, *StingyCD: Safely avoiding wasteful updates in coordinate descent*, in Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1752–1760.

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

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

[18] C.-P. LEE AND S. J. WRIGHT, *Random permutations fix a worst case for cyclic coordinate descent*, IMA J. Numer. Anal. (2016), dry 040.

[19] J. D. LEE, I. PANAGEAS, G. PILIOURAS, M. SIMCHOWITZ, M. I. JORDAN, AND B. RECHT, *First-Order Methods Almost Always Avoid Saddle Points*, arXiv:1710.07406, 2017.

[20] Y. T. LEE AND A. SIDFORD, *Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems*, in Proceedings of the IEEE 54th Annual Symposium on the Foundations of Computer Science, 2013, pp. 147–156.

[21] Q. LEI, K. ZHONG, AND I. S. DHILLON, *Coordinate-wise power method*, Adv. Neural Inf. Process. Syst., 29 (2016), pp. 2064–2072.

[22] Q. LIN, Z. LU, AND L. XIAO, *An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization*, SIAM J. Optim., 25 (2015), pp. 2244–2273.

[23] J. LIU, AND S. J. WRIGHT, *Asynchronous stochastic coordinate descent: Parallelism and convergence properties*, SIAM J. Optim., 25 (2015), pp. 351–376.

[24] J. LIU, S. J. WRIGHT, C. RÉ, V. BITTORF, AND S. SRIDHAR, *An asynchronous parallel stochastic coordinate descent algorithm*, J. Mach. Learn. Res., 16 (2015a), pp. 285–322.

[25] X. LIU, Z. WEN, AND Y. ZHANG, *An efficient Gauss–Newton algorithm for symmetric low-rank product matrix approximations*, SIAM J. Optim., 25 (2015), pp. 1571–1608.

[26] J. LU AND K. THICKE, *Orbital minimization method with l1 regularization*, J. Comput. Phys., 336 (2017), pp. 87–103.

[27] A. E. MITCHELL, G. E. SMITH, K. L. BELL, AND M. RANGASWAMY, *Coordinate descent for cognitive radar adaptation*, in CIE International Conference of Radar Proceedings, 2016, pp. 1–5.

[28] Y. NESTEROV, *Introductory Lectures on Convex Optimization: A Basic Course*, Springer, New York, 2004.

[29] Y. NESTEROV, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM J. Optim., 22 (2012), pp. 341–362.

[30] J. NUTINI, M. SCHMIDT, I. H. LARADJI, M. FRIEDLANDER, AND H. KOEPKE, *Coordinate descent converges faster with the Gauss-Southwell rule than random selection*, in Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 1632–1641.

[31] T. PARNELL, C. DÜNNER, K. ATASU, M. SIFALAKIS, AND H. POZIDIS, *Tera-scale coordinate descent on GPUs*, Futur. Gener. Comput. Syst. (2018).

[32] Z. PENG, T. WU, Y. XU, M. YAN, AND W. YIN, *Coordinate-friendly structures, algorithms and applications*, Ann. Math. Sci. Appl., 1 (2016), pp. 57–119.

[33] D. PEREKRESTENKO, V. CEVHER, AND M. JAGGI, *Faster coordinate descent via adaptive importance sampling*, Proc. Mach. Learn. Res., 54 (2017), pp. 869–877.

[34] P. RICHTÁRIK, AND M. TAKÁČ, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Math. Program., 144 (2014), pp. 1–38.

[35] H.-J. M. SHI, S. TU, Y. XU, AND W. YIN, *A Primer on Coordinate Descent Algorithms*, arXiv:1610.00040, 2016.

[36] Q. SHI, H. SUN, S. LU, M. HONG, AND M. RAZAVIYAYN, *Inexact block coordinate descent methods for symmetric nonnegative matrix factorization*, IEEE Trans. Signal Process., 65 (2017), pp. 5995–6008.

[37] G. L. G. SLEIJPEN AND H. A. VAN DER VORST, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[38] R. V. SOUTHWELL, *Relaxation Methods in Engineering Science: A Treatise on Approximate Computation*, Pickard Press, 2010.

[39] R. SUN AND Y. YE, *Worst-Case Complexity of Cyclic Coordinate Descent: $O(n^2)$ Gap with Randomized Version*, arXiv:1604.07130, 2016.

[40] A. SZABO AND N. S. OSTLUND, *Modern Quantum Chemistry*, Dover, New York, 1996.

[41] R. TAPPENDEN, M. TAKÁČ, AND P. RICHTÁRIK, *On the complexity of parallel coordinate descent*, Optim. Methods Softw., 33 (2018), pp. 372–395.

[42] P. TSENG, *Convergence of a block coordinate descent method for nondifferentiable minimization*, J. Optim. Theory Appl., 109 (2001), pp. 475–494.

[43] S. TU, R. ROELOFS, S. VENKATARAMAN, AND B. RECHT, *Large Scale Kernel Learning Using Block Coordinate Descent*, arXiv:1602.05310, 2016.

[44] A. VANDAELE, N. GILLIS, Q. LEI, K. ZHONG, AND I. DHILLON, *Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization*, IEEE Trans. Signal Process., 64 (2016), pp. 5571–5584.

[45] E. VECHARYNSKI, C. YANG, AND J. E. PASK, *A projected preconditioned conjugate gradient algorithm for computing many extreme eigenpairs of a Hermitian matrix*, J. Comput. Phys., 290 (2015), pp. 73–89.

[46] J. WANG, W. WANG, D. GARBER, AND N. SREBRO, *Efficient coordinate-wise leading eigenvector computation*, Proc. Mach. Learn. Res., 83 (2018), pp. 806–820.

[47] S. J. WRIGHT, *Coordinate descent algorithms*, Math. Program., 151 (2015), pp. 3–34.

[48] S. J. WRIGHT AND C.-P. LEE, *Analyzing Random Permutations for Cyclic Coordinate Descent*, arXiv:1706.00908, 2017.

[49] M. Wu, C. Dick, J. R. Cavallaro, and C. Studer, *FPGA design of a coordinate descent data detector for large-scale MU-MIMO*, in Proceedings of the IEEE International Symposium on Circuits and Systems, 2016, pp. 1894–1897.

[50] M. Wu, C. Dick, J. R. Cavallaro, and C. Studer, *High-throughput data detection for massive MU-MIMO-OFDM using coordinate descent*, IEEE Trans. Circuits Syst. I Regul. Pap., 63 (2016), pp. 2357–2367.

[51] Y. Xu, *Hybrid Jacobian and Gauss–Seidel proximal block coordinate update methods for linearly constrained convex programming*, SIAM J. Optim., 28 (2018), pp. 646–670.

[52] Y. You, X. Lian, J. Liu, H.-F. Yu, I. S. Dhillon, J. Demmel, and C.-J. Hsieh, *Asynchronous parallel greedy coordinate descent*, Adv. Neural Inf. Process. Syst. 29 (2016), pp. 4682–4690.

[53] W.-J. Zeng and H. C. So, *Coordinate Descent Algorithms for Phase Retrieval*, arXiv:1706.03474, 2017.