A RANDOMIZED COORDINATE DESCENT METHOD WITH VOLUME SAMPLING*

ANTON RODOMANOV† AND DMITRY KROPOTOV‡

Abstract. We analyze the coordinate descent method with a new coordinate selection strategy, called volume sampling. This strategy prescribes selecting subsets of variables of certain size proportionally to the determinants of principal submatrices of the matrix, which bounds the curvature of the objective function. In the particular case when the size of the subsets equals one, volume sampling coincides with the well-known strategy of sampling coordinates proportionally to their Lipschitz constants. For the coordinate descent with volume sampling, we establish the convergence rates for both convex and strongly convex problems. Our theoretical results show that, by increasing the size of the subsets, it is possible to accelerate the method up to the factor which depends on the spectral gap between the corresponding largest eigenvalues of the curvature matrix. Several numerical experiments confirm our theoretical conclusions.

Key words. convex optimization, unconstrained minimization, coordinate descent methods, randomized algorithms, volume sampling, convergence rate

AMS subject classifications. 90C25, 90C06, 68Q25

DOI. 10.1137/19M125532X

1. Introduction. Coordinate descent methods are minimization algorithms that are very popular for solving large-scale optimization problems. The main idea of these algorithms is to successively reduce the value of the objective function along certain subsets of coordinates that are selected at each iteration according to some rule. Coordinate descent has been successfully applied to a number of applications in various areas, such as machine learning, compressed sensing, network problems, etc.

One of the main parameters of a typical coordinate descent method is the *number of coordinates* τ, which are updated at every iteration. When choosing this parameter, one usually faces the following trade-off. On the one hand, the convergence rate of the method becomes faster with the increase of τ, but, on the other hand, each iteration becomes more expensive. Therefore, to obtain a speed-up of the method in terms of the total running time, it is necessary to ensure that the increase in the cost of each iteration is low compared to the increase in the convergence rate. One possible way to achieve this is *parallelization*. This idea has been extensively discussed in the context of parallel coordinate descent [1, 2, 3, 4, 5, 6], where (under certain separability assumptions) the authors show that the convergence rate improves *linearly* in τ, and thus it is possible to achieve a linear speed-up by using τ independent processors instead of one. Note, however, that in the usual *serial regime* (without parallelization) the aforementioned results do not guarantee any decrease in the total running time since each iteration becomes at least τ times more expensive. Clearly, to be able to

†Samsung-HSE Laboratory, National Research University Higher School of Economics, Moscow, 109028, Russia (anton.rodomanov@gmail.com).

‡Samsung-HSE Laboratory, National Research University Higher School of Economics, Moscow, 109028, Russia, and Lomonosov Moscow State University, Moscow, 119991, Russia (dmitry.kropotov@gmail.com).

ensure a speed-up in this regime, one needs some *nonlinear*, in terms of τ, convergence rate estimates.

In this paper, we analyze the coordinate descent method with a new coordinate selection strategy, called *volume sampling*. This strategy prescribes selecting τ-element subsets of variables proportionally to the volumes (or determinants) of principal submatrices of the matrix B, which bounds the curvature of the objective function (see section 2). For the coordinate descent with volume sampling, we establish the worst-case iteration complexity bounds, which have a *nonlinear* dependency on τ. In particular, we show that the increase in τ from τ_1 to τ_2 leads to the improvement of the convergence rate of the method up to the factor of

$$(1.1) \qquad R_\lambda(\tau_1, \tau_2) := \frac{\sum_{i=\tau_1}^n \lambda_i}{\sum_{i=\tau_2}^n \lambda_i},$$

where $\lambda_1 \geq \cdots \geq \lambda_n$ are the eigenvalues of B. Note that $R_\lambda(\tau_1, \tau_2)$ can be arbitrarily big, depending on the *spectral gap* between λ_{τ_1} and λ_{τ_2}.

In addition to this, we also propose a new efficient algorithm for two-element volume sampling from a *sparse* matrix. The preprocessing complexity of this algorithm is of the order of the number of nonzero elements in the matrix, and its sampling complexity is only logarithmic in the dimension.

1.1. Related work. There is a vast literature on coordinate descent methods. Most research in this area is usually focused on the rules for selecting coordinates [7, 8, 9], or on obtaining accelerated [7, 10, 11], parallel [1, 2, 3, 4, 5, 6], proximal [3, 12], and primal-dual [13, 14, 15, 16] variants of the already known methods. For a general overview of the topic, see the recent paper [17] and references therein. Below we discuss just several works that are most closely related to ours.

One of the most influential papers on coordinate descent is [7] by Nesterov, where he proposed a coordinate gradient method (which we will refer to as RCD) with a special *randomized* rule for selecting coordinates. In RCD, each coordinate is sampled with probability proportional to the corresponding coordinate Lipschitz constant. Nesterov then derived the complexity bound for RCD and showed that it can be even better than that of the standard gradient method. The method, which we consider in this work, generalizes RCD in the sense that it coincides with it in the special case when the number of coordinates, selected at each iteration, equals one.

The most relevant work to ours is [16], where the authors propose three different randomized methods for smooth unconstrained minimization. Their Method 1 is exactly the same method that we analyze in this paper, with the only difference that, instead of volume sampling, they consider an *arbitrary* sampling. As a result, the convergence rate of their method is expressed quite abstractly (in terms of the minimal eigenvalue of the expectation of a certain matrix), and, in particular, it is not clear how exactly it depends on the number of coordinates τ used at each iteration. Although the authors provide a particular example of a 3×3 matrix, for which their method should be very efficient, they do not establish any general results. In this regard, our work can be considered a further development of [16], where we establish more interpretable iteration complexity bounds specifically for volume sampling. In addition to that, in [16], the authors work only with the strongly convex problems, while in this paper, we allow the problem to be nonstrongly convex.

Another relevant work to this one is [18], where the authors propose a new randomized optimization method for minimizing quadratic functions, given τ eigenvectors corresponding to the τ smallest eigenvalues of the Hessian. Although the complexity

estimates for this method look similar to ours, there are several key differences. First, the results in [18] show that the increase in the number of coordinates from τ_1 to τ_2 in their method leads to the acceleration rate that depends on the spectral gap between the τ_1st and τ_2nd *smallest* eigenvalues. The acceleration rate for coordinate descent with volume sampling depends, on the contrary, on the spectral gap between the τ_1st and τ_2nd *largest* eigenvalues. Second, their method is not, strictly speaking, a coordinate descent algorithm since it uses eigenvectors as search directions instead of the coordinate directions. Finally, it is also less practical. For example, even in the simplest nontrivial regime $\tau = 1$, their method requires an eigenvector, corresponding to the smallest eigenvalue; the complexity of obtaining such a vector is in general $O(n^3)$. In contrast, the simplest nontrivial choice for coordinate descent with volume sampling is $\tau = 2$, which requires one to perform at the beginning one preprocessing step of $O(n^2)$ (or even $O(n + \mathrm{nnz}(B))$; see section 4.2), and then each iteration of the method takes linear time in the dimension.

Finally, we should mention that, although volume sampling has not been previously considered in the context of coordinate descent methods, it is not a novel concept and has already been known in the literature for some time. To our knowledge, it was first proposed in [19] for the problem of matrix approximation. Later, the same authors developed several efficient exact and approximate methods for doing volume sampling [20] based on the standard linear algebra algorithms. Some other polynomial-time sampling methods and their connection to the theory of Markov chains were considered in [21]. Recently, volume sampling has also been applied to the problem of linear regression [22, 23].

1.2. Contents. This paper is organized as follows. In section 2, we describe the randomized coordinate descent method with volume sampling. In section 3, we present the convergence analysis of this method. We start with an auxiliary sufficient decrease lemma (section 3.1) and then use it to derive the convergence rates for both convex functions (section 3.2) and strongly convex ones (section 3.3). In section 4, we discuss how to generate a random variable according to volume sampling. First, we discuss a simple general approach (section 4.1) and, after this, develop a special algorithm for two-element volume sampling which is suitable for sparse matrices (section 4.2). In section 5, we consider several examples of possible applications: quadratic functions (section 5.1), separable problems (section 5.2), and the smoothing technique (section 5.3). Finally, in section 6, we present the results of several numerical experiments.

1.3. Notation. By $\mathbb{R}^n$ we denote the Euclidean space of all n-dimensional real column vectors with the standard inner product $\langle u, v \rangle := \sum_{i=1}^n u_i v_i$ and the standard Euclidean norm $\|v\| := \langle v, v \rangle^{\frac{1}{2}}$. Given an $n \times n$ real symmetric positive semidefinite matrix B, we also use the seminorm $\|v\|_B := \langle Bv, v \rangle^{\frac{1}{2}}$; recall that $\| \cdot \|_B$ becomes a norm iff B is positive definite.

For $1 \leq \tau \leq n$, by $\binom{[n]}{\tau}$ we denote the collection of all τ-element subsets of $[n] := \{1, \ldots, n\}$. For each $S \in \binom{[n]}{\tau}$, by I_S we denote the $n \times \tau$ matrix obtained from the $n \times n$ identity matrix I by retaining only those columns whose indices are in S; if the dimension n is not specified directly, then it can be determined from the context. For an $n \times n$ matrix B and a subset $S \in \binom{[n]}{\tau}$, by $B_{S \times S}$ we denote the $\tau \times \tau$ principal submatrix located at the intersection of the rows and columns with indices from S (i.e., $B_{S \times S} := I_S^T B I_S$); similarly, for a vector $v \in \mathbb{R}^n$, by v_S we denote the subvector of size τ obtained from v by retaining only the elements with indices from S (i.e., $v_S := I_S^T v$).

Finally, for a square matrix A, by $\mathrm{Adj}(A)$ we denote its adjugate matrix (the transpose of the cofactor matrix).

2. Randomized coordinate descent with volume sampling. Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x),$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a differentiable function. We assume that f is 1-*smooth with respect to the seminorm* $\|\cdot\|_B$ induced by some $n \times n$ real symmetric positive semidefinite matrix B:

$$(2.1) \qquad f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2}\|y - x\|_B^2$$

for all $x, y \in \mathbb{R}^n$ (see section 5 for examples). When f is twice continuously differentiable, one sufficient condition for this is that the Hessian of f is uniformly upper bounded by B.

Remark 2.1. The standard smoothness assumption in the context of coordinate descent methods is slightly different. Typically, it is assumed that, for each $S \in \binom{[n]}{\tau}$, there exists $L_S \geq 0$ (called a coordinate Lipschitz constant) such that

$$(2.2) \qquad f(x + I_S h) \leq f(x) + \langle \nabla f(x)_S, h \rangle + \frac{L_S}{2}\|h\|^2$$

for all $x \in \mathbb{R}^n$ and all $h \in \mathbb{R}^\tau$. Clearly, if f satisfies (2.1), it also satisfies (2.2) for $L_S := \|B_{S \times S}\|$. However, (2.1) and (2.2) are not completely equivalent. For example, if $\tau = 1$ and the function f is twice continuously differentiable, (2.2) requires only the *diagonal* of the Hessian to be uniformly bounded. Nevertheless, for many practical applications, condition (2.1) holds (see section 5).

Let us fix a point $x_0 \in \mathbb{R}^n$ and a τ-element subset of coordinates $S_0 \in \binom{[n]}{\tau}$, where $1 \leq \tau \leq \mathrm{Rank}(B)$. According to (2.1), we have

$$(2.3) \qquad f(x_0 + I_{S_0} h) \leq f(x_0) + \langle \nabla f(x_0)_{S_0}, h \rangle + \frac{1}{2}\|h\|_{B_{S_0 \times S_0}}^2$$

for all $h \in \mathbb{R}^\tau$. A natural idea to obtain an update rule of a coordinate descent algorithm is to minimize the right-hand side of (2.3) in h. It is possible to do so when the matrix $B_{S_0 \times S_0}$ is nondegenerate, and this leads to the following update rule:

$$(2.4) \qquad x_1 := x_0 - I_{S_0}(B_{S_0 \times S_0})^{-1} \nabla f(x_0)_{S_0}.$$

Now it remains to specify the procedure for selecting the coordinates S_0. In view of the above remark, the probability of choosing a degenerate submatrix $B_{S_0 \times S_0}$ should be zero. One sampling scheme that naturally possesses this property is given by the following.

DEFINITION 2.2 (volume sampling). *Let B be an $n \times n$ real symmetric positive semidefinite matrix, let $1 \leq \tau \leq \mathrm{Rank}(B)$, and let S_0 be a random variable taking values in $\binom{[n]}{\tau}$. We say that S_0 is generated according to τ-element volume sampling with respect to B, denoted by $S_0 \sim \mathrm{Vol}_\tau(B)$, if for all $S \in \binom{[n]}{\tau}$ we have*

$$\mathbb{P}(S_0 = S) = \frac{\mathrm{Det}(B_{S \times S})}{\sum_{S' \in \binom{[n]}{\tau}} \mathrm{Det}(B_{S' \times S'})}.$$

Observe that for $\tau = 1$ volume sampling corresponds to picking indices with probabilities proportional to the coordinate Lipschitz constants B_{ii} (diagonal elements of B). Thus, volume sampling in fact generalizes the well-known coordinate Lipschitz constant sampling. We discuss its implementation in section 4.

Combining the update rule (2.4) with the volume sampling of coordinates, we obtain the randomized coordinate descent method with volume sampling (RCDVS); see Algorithm 2.1. Note that for $\tau = 1$, RCDVS coincides with the well-known RCD method from [7].

Algorithm 2.1 RCDVS(f, B, τ, x_0, K).

Require: differentiable function $f : \mathbb{R}^n \to \mathbb{R}$; $n \times n$ real symmetric positive semidefinite matrix B; number of coordinates $1 \leq \tau \leq \mathrm{Rank}(B)$; starting point $x_0 \in \mathbb{R}^n$; number of iterations $K \geq 1$.
for $0 \leq k \leq K - 1$ **do**
  Choose a random subset of coordinates $S_k \sim \mathrm{Vol}_\tau(B)$.
  Set $x_{k+1} := x_k - I_{S_k}(B_{S_k \times S_k})^{-1} \nabla f(x_k)_{S_k}$.
end for

---

3. Convergence analysis. We now turn to analyzing the convergence rate of the RCDVS method. To keep the presentation concise, we only study the convergence rates of expectations, although it is not difficult to establish their high probability counterparts using standard techniques.

3.1. Sufficient decrease lemma. We start with the following simple result, which directly follows from smoothness and does not yet take into account the particular strategy for sampling coordinates:

LEMMA 3.1 (general sufficient decrease lemma). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a 1-smooth function with respect to the seminorm $\| \cdot \|_B$, where B is an $n \times n$ real symmetric positive semidefinite matrix. Let $x_0 \in \mathbb{R}^n$ be deterministic, let $1 \leq \tau \leq \mathrm{Rank}(B)$, let S_0 be a random variable taking values in $\binom{[n]}{\tau}$ such that $B_{S_0 \times S_0}$ is nondegenerate almost surely, and let $x_1 := x_0 - I_{S_0}(B_{S_0 \times S_0})^{-1} \nabla f(x_0)_{S_0}$. Then $f(x_1) \leq f(x_0)$ almost surely, and*

$$(3.1) \qquad f(x_0) - \mathbb{E}f(x_1) \geq \frac{1}{2} \|\nabla f(x_0)\|^2_{\mathbb{E}I_{S_0}(B_{S_0 \times S_0})^{-1}I_{S_0}^T}.$$

In its current form, Lemma 3.1 is not very useful since it involves some general expectation $\mathbb{E}I_{S_0}(B_{S_0 \times S_0})^{-1}I_{S_0}^T$ which is unclear to us how to work. Our task now is to estimate this expectation in a convenient, yet nontrivial way for the particular case $S_0 \sim \mathrm{Vol}_\tau(B)$.

Assume that all $\tau \times \tau$ submatrices of B are nondegenerate, i.e., the τ-element volume sampling has full support (the other case will be considered later). Using Cramer's rule $\mathrm{Det}(B_{S \times S})(B_{S \times S})^{-1} = \mathrm{Adj}(B_{S \times S})$, we can write

$$(3.2) \qquad \mathbb{E}I_{S_0}(B_{S_0 \times S_0})^{-1}I_{S_0}^T = \frac{\sum_{S \in \binom{[n]}{\tau}} I_S \, \mathrm{Adj}(B_{S \times S})I_S^T}{\sum_{S \in \binom{[n]}{\tau}} \mathrm{Det}(B_{S \times S})}.$$

Thus, to estimate the expectation, we need to estimate the following two sums:
  1. the sum of principal minors $\sum_{S \in \binom{[n]}{\tau}} \mathrm{Det}(B_{S \times S})$ and

2. the sum $\sum_{S \in \binom{[n]}{\tau}} I_S \operatorname{Adj}(B_{S \times S}) I_S^T$.

The first sum is rather well known, and a closed form expression for it can be found in many standard textbooks on linear algebra (see, e.g., Chapter 7 in [24]). To present the formula, let us introduce for each $1 \leq m \leq n$ the real *elementary symmetric polynomial* $\sigma_m : \mathbb{R}^n \to \mathbb{R}$ of degree m, defined by

$$\sigma_m(x) := \sum_{1 \leq i_1 < \cdots < i_m \leq n} x_{i_1} \ldots x_{i_m},$$

i.e., the sum of all m-ary products of $x_1, \ldots, x_n$, and put $\sigma_0(x) := 1$ for convenience. The well-known result is the following

LEMMA 3.2 (sum of principal minors). *Let B be an $n \times n$ real symmetric matrix with eigenvalues $\lambda := (\lambda_1, \ldots, \lambda_n)$, where $\lambda_1 \geq \cdots \geq \lambda_n$, and let $1 \leq \tau \leq n$. Then*

$$(3.3) \qquad \sum_{S \in \binom{[n]}{\tau}} \operatorname{Det}(B_{S \times S}) = \sigma_\tau(\lambda).$$

Now we turn to the second sum. To the best of our knowledge, this sum has not been previously considered in the literature. Nevertheless, it turns out that it can also be conveniently expressed in terms of the elementary symmetric polynomials of eigenvalues.

LEMMA 3.3. *Let B be an $n \times n$ real symmetric matrix with eigenvalues $\lambda := (\lambda_1, \ldots, \lambda_n)$, where $\lambda_1 \geq \cdots \geq \lambda_n$, let $B = Q \operatorname{Diag}(\lambda) Q^T$ be its spectral decomposition for some $n \times n$ orthogonal matrix Q, and let $1 \leq \tau \leq n$. Then*

$$(3.4) \qquad \sum_{S \in \binom{[n]}{\tau}} I_S \operatorname{Adj}(B_{S \times S}) I_S^T = Q \operatorname{Diag}(\sigma_{\tau-1}(\lambda_{-1}), \ldots, \sigma_{\tau-1}(\lambda_{-n})) Q^T,$$

where λ_{-i} for each $1 \leq i \leq n$ denotes the vector λ without the ith element.

Let us accept this lemma for now and defer its proof to a separate section (section 3.4).

Using Lemma 3.2 together with Lemma 3.3, we can rewrite (3.2) as follows:

$$(3.5) \qquad \mathbb{E} I_{S_0} (B_{S_0 \times S_0})^{-1} I_{S_0}^T = \frac{Q \operatorname{Diag}(\sigma_{\tau-1}(\lambda_{-1}), \ldots, \sigma_{\tau-1}(\lambda_{-n})) Q^T}{\sigma_\tau(\lambda)}.$$

Thus, we have managed to express the expectation solely in terms of the eigenvectors and eigenvalues of B. However, our new expression for the expectation is still difficult to work with because each elementary symmetric polynomial is in fact a very complex sum. Fortunately, recall that we do not need the expectation itself but only a suitable lower bound for it. To obtain such a bound, it is convenient to introduce the following.

DEFINITION 3.4 (τ-coordinate approximation). *Let B an $n \times n$ real symmetric positive semidefinite matrix with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$, let $1 \leq \tau \leq \operatorname{Rank}(B)$, and let $B = Q \operatorname{Diag}(\lambda_1, \ldots, \lambda_n) Q^T$ be a spectral decomposition of B for some $n \times n$ orthogonal matrix Q. The τ-coordinate approximation of B, denoted by B_τ, is the $n \times n$ real positive semidefinite matrix*

$$(3.6) \qquad B_\tau := Q \operatorname{Diag}(\lambda_1, \ldots, \lambda_\tau, \lambda_\tau, \ldots, \lambda_\tau) Q^T + \sum_{i=\tau+1}^n \lambda_i I.$$

Observe that B_τ is nondegenerate since otherwise $\lambda_\tau = \cdots = \lambda_n = 0$, which, in view of positive semidefiniteness, contradicts the assumption that $\tau \leq \text{Rank}(B)$. Also note that B_τ does not depend on the particular orthogonal matrix Q in the spectral decomposition of B. Indeed, the first term in the definition of B_τ can be written as $Q \text{Diag}(q_\tau(\lambda_1), \ldots, q_\tau(\lambda_n))Q^T$, where $q_\tau : \mathbb{R} \to \mathbb{R}$ is the function $q_\tau(t) := \max\{t, \lambda_\tau\}$. It is well known that such matrices do not depend on the choice of the diagonalizing matrix Q (see, e.g., [24, section 7.3]).

Using the τ-coordinate approximation, we can now lower bound (3.5) as follows.

LEMMA 3.5. *Let B an $n \times n$ real symmetric positive semidefinite matrix with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$, let $1 \leq \tau \leq \text{Rank}(B)$, and let $B = Q \text{Diag}(\lambda_1, \ldots, \lambda_n)Q^T$ be a spectral decomposition of B for some $n \times n$ orthogonal matrix Q. Then*

$$\frac{Q \text{Diag}(\sigma_{\tau-1}(\lambda_{-1}), \ldots, \sigma_{\tau-1}(\lambda_{-n}))Q^T}{\sigma_\tau(\lambda)} \succeq (B_\tau)^{-1}.$$

Proof. Since the eigenvalues are nonnegative, we have

$$\sigma_\tau(\lambda) = \sum_{i_1=1}^{n-\tau+1} \lambda_{i_1} \sum_{i_1+1 \leq i_2 < \cdots < i_\tau \leq n} \lambda_{i_2} \ldots \lambda_{i_\tau} \leq \sigma_{\tau-1}(\lambda_{-1}) \sum_{i=1}^{n-\tau+1} \lambda_i.$$

By the symmetry of elementary symmetric polynomials, this can be strengthened to

$$\sigma_\tau(\lambda) \leq \sigma_{\tau-1}(\lambda_{-1}) \left(\lambda_1 + \sum_{j=\tau+1}^{n} \lambda_j \right),$$

which in turn can be further generalized to

$$\sigma_\tau(\lambda) \leq \sigma_{\tau-1}(\lambda_{-i}) \left(\lambda_{\min\{i,\tau\}} + \sum_{j=\tau+1}^{n} \lambda_j \right)$$

for all $1 \leq i \leq n$. The claim follows. $\square$

To summarize, we have obtained that in the case when volume sampling has full support, we can replace (3.1) with

(3.7)
$$f(x_0) - \mathbb{E}f(x_1) \geq \frac{1}{2}\|\nabla f(x_0)\|^2_{(B_\tau)^{-1}}.$$

It remains to show that exactly the same result holds even when some $\tau \times \tau$ principal submatrices of B are possibly degenerate. In this case, instead of (3.2) we should write more carefully that

$$\mathbb{E}I_{S_0}(B_{S_0 \times S_0})^{-1}I_{S_0}^T = \frac{\sum_{S \in \binom{[n]}{\tau}:\text{Det}(B_{S \times S}) \neq 0} I_S \text{Adj}(B_{S \times S})I_S^T}{\sum_{S \in \binom{[n]}{\tau}} \text{Det}(B_{S \times S})}.$$

Unfortunately, we cannot use Lemma 3.3 anymore because now (3.4) *overestimates* (and not underestimates) the numerator due to the fact that the adjugate to a symmetric positive semidefinite matrix is also symmetric positive semidefinite. However, recall that we are not interested in the numerator itself, but only in how it acts on the gradient $\nabla f(x_0)$. For each $1 \leq \tau \leq n$, define the linear subspace

$$U_\tau(B) := \{u \in \mathbb{R}^n : u_S \in \text{Im}(B_{S \times S}) \text{ for all } S \in \binom{[n]}{\tau} \text{ with } \text{Det}(B_{S \times S}) = 0\},$$

where $\mathrm{Im}(B_{S \times S})$ is the image space of $B_{S \times S}$. Observe that $U_\tau(B) = \mathbb{R}^n$ iff all $\tau \times \tau$ principal submatrices of B are nondegenerate. Our interest in the subspace $U_\tau(B)$ lies in the following observation.

LEMMA 3.6. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a 1-smooth function with respect to the seminorm $\|\cdot\|_B$, where B is an $n \times n$ real symmetric positive semidefinite matrix. If f is bounded from below, then for each $x_0 \in \mathbb{R}^n$ and each $1 \le \tau \le n$, we have $\nabla f(x_0) \in U_\tau(B)$.*

Proof. Let $S \in \binom{[n]}{\tau}$ be such that $\mathrm{Det}(B_{S \times S}) = 0$ (if there is no such S, the claim is vacuously true). Then the kernel $\mathrm{Ker}(B_{S \times S})$ is nontrivial, and hence there exists a nonzero $h \in \mathrm{Ker}(B_{S \times S})$. From smoothness of f with respect to $\|\cdot\|_B$, it follows that $f(x_0 + t I_S h) \le f(x_0) + t \langle \nabla f(x_0)_S, h \rangle$ for all $t \in \mathbb{R}$. Hence, $\nabla f(x_0)_S \in \mathrm{Ker}(B_{S \times S})^\perp = \mathrm{Im}(B_{S \times S})$; otherwise, f is unbounded from below.   □

According to Lemma 3.6 and the above remarks, we are interested only in the action of $\sum_{S \in \binom{[n]}{\tau}: \mathrm{Det}(B_{S \times S}) \ne 0} I_S \mathrm{Adj}(B_{S \times S}) I_S^T$ on the subspace $U_\tau(B)$. But one can easily see that on this subspace it acts exactly as the already studied matrix $\sum_{S \in \binom{[n]}{\tau}} I_S \mathrm{Adj}(B_{S \times S}) I_S^T$, and so the case of degenerate submatrices reduces to that of nondegenerate ones.

Thus, regardless of whether there are degenerate principal submatrices or not, we have proved the following.

LEMMA 3.7 (sufficient decrease lemma for volume sampling). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function which is bounded from below and 1-smooth with respect to the seminorm $\|\cdot\|_B$, where B is an $n \times n$ real symmetric positive semidefinite matrix. Let $x_0 \in \mathbb{R}^n$, let $S_0 \sim \mathrm{Vol}_\tau(B)$ for some $1 \le \tau \le \mathrm{Rank}(B)$, and let $x_1 := x_0 - I_{S_0}(B_{S_0 \times S_0})^{-1} \nabla f(x_0)_{S_0}$. Then $f(x_1) \le f(x_0)$ almost surely, and*

$$(3.8) \qquad f(x_0) - \mathbb{E}f(x_1) \ge \frac{1}{2} \|\nabla f(x_0)\|_{(B_\tau)^{-1}}^2.$$

To finish this section, let us establish the following relations between τ-coordinate approximations, which will be central in the forthcoming convergence analysis.

LEMMA 3.8. *Let B be an $n \times n$ real symmetric positive semidefinite matrix with eigenvalues $\lambda_1 \ge \cdots \ge \lambda_n$, and let $1 \le \tau_1 < \tau_2 \le \mathrm{Rank}(B)$. Then*

$$B_{\tau_2} \preceq B_{\tau_1} \preceq R_\lambda(\tau_1, \tau_2) B_{\tau_2},$$

where $R_\lambda(\tau_1, \tau_2)$ is defined by (1.1).

Proof. Denote $\Sigma_i := \sum_{j=i}^n \lambda_j$ for $1 \le i \le n$. According to (3.6), we have $B_{\tau_1} = Q \mathrm{Diag}(s_1^{(1)}, \ldots, s_n^{(1)}) Q^T$, where

$$s_i^{(1)} := \begin{cases} \lambda_i + \Sigma_{\tau_1 + 1} & \text{if } 1 \le i \le \tau_1, \\ \Sigma_{\tau_1} & \text{if } \tau_1 + 1 \le i \le n. \end{cases}$$

Similarly, $B_{\tau_2} = Q \mathrm{Diag}(s_1^{(2)}, \ldots, s_n^{(2)}) Q^T$, where

$$s_i^{(2)} := \begin{cases} \lambda_i + \Sigma_{\tau_2 + 1} & \text{if } 1 \le i \le \tau_2, \\ \Sigma_{\tau_2} & \text{if } \tau_2 + 1 \le i \le n. \end{cases}$$

Hence, $(B_{\tau_2})^{-1/2} B_{\tau_1} (B_{\tau_2})^{-1/2} = Q \operatorname{Diag}(s_1, \ldots, s_n) Q^T$, where

$$
s_i := \begin{cases} \frac{\lambda_i + \Sigma_{\tau_1 + 1}}{\lambda_i + \Sigma_{\tau_2 + 1}} & \text{if } 1 \leq i \leq \tau_1, \\ \frac{\Sigma_{\tau_1}}{\lambda_i + \Sigma_{\tau_2 + 1}} & \text{if } \tau_1 + 1 \leq i \leq \tau_2, \\ \frac{\Sigma_{\tau_1}}{\Sigma_{\tau_2}} & \text{if } \tau_2 + 1 \leq i \leq n. \end{cases}
$$

To finish the proof, it now remains to show that $s_1, \ldots, s_n$ are bounded from below by 1 and bounded from above by $\frac{\Sigma_{\tau_1}}{\Sigma_{\tau_2}}$.

For $1 \leq i \leq \tau_1$, we have $s_i = 1 + \frac{\Sigma_{\tau_1 + 1} - \Sigma_{\tau_2 + 1}}{\lambda_i + \Sigma_{\tau_2 + 1}}$. Since $\Sigma_{\tau_1 + 1} \geq \Sigma_{\tau_2 + 1} \geq 0$, it follows from this representation that $1 \leq s_1 \leq \cdots \leq s_{\tau_1}$ (recall that $\lambda_1 \geq \cdots \geq \lambda_n \geq 0$). Similarly, for $\tau_1 \leq i \leq \tau_2$ (including τ_1), we have $s_i = \frac{\Sigma_{\tau_1}}{\lambda_i + \Sigma_{\tau_2 + 1}}$, and hence $s_{\tau_1} \leq \cdots \leq s_{\tau_2}$. Finally, $s_{\tau_2} = \cdots = s_n$ (including τ_2). Thus, $1 \leq s_1 \leq \cdots \leq s_n = \frac{\Sigma_{\tau_1}}{\Sigma_{\tau_2}}$. $\quad\square$

3.2. Convex functions. Now we are ready to establish several results on the convergence rate of the RCDVS method. We start with the class of smooth convex functions.

THEOREM 3.9 (convergence rate for convex functions). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function which is 1-smooth with respect to the seminorm $\|\cdot\|_B$, where B is an $n \times n$ real symmetric positive semidefinite matrix. Let x_0 be a deterministic point in $\mathbb{R}^n$, and assume that the sublevel set $L_f(x_0) := \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded. Let $1 \leq \tau \leq \operatorname{Rank}(B)$, $K \geq 1$, and let $(x_k)_{k=1}^K$ be the random points in $\mathbb{R}^n$ generated by $\mathrm{RCDVS}(f, B, \tau, x_0, K)$. Then*

$$
\mathbb{E} f(x_k) - \min f \leq \frac{2 D_\tau^2}{k + 1},
$$

for all $0 \leq k \leq K$, where $D_\tau := \max_{x \in L_f(x_0)} \min_{x^* \in \operatorname{Argmin} f} \|x - x^*\|_{B_\tau}$ is the radius of the sublevel set $L_f(x_0)$ measured in the norm $\|\cdot\|_{B_\tau}$.

Proof. See Appendix A. $\quad\square$

According to Theorem 3.9, for achieving accuracy $\varepsilon > 0$ in terms of the expected value of the objective, one needs the following number of iterations:

$$
K_\tau := \frac{2 D_\tau^2}{\varepsilon}.
$$

In particular, for $\tau = 1$, we have $D_\tau^2 = \operatorname{Tr}(B) D^2$, where D is the radius of the sublevel set $L_f(x_0)$ measured in the standard Euclidean norm; this recovers the already known result for the RCD method [7].

Let us fix $1 \leq \tau_1 < \tau_2 \leq \operatorname{Rank}(B)$ and compare the efficiency estimates of RCDVS with τ_1 coordinates with that of τ_2 coordinates. We obtain that

$$
\frac{K_{\tau_1}}{K_{\tau_2}} = \frac{D_{\tau_1}^2}{D_{\tau_2}^2}.
$$

Thus, we need to compare the quantities $D_{\tau_1}^2$ and $D_{\tau_2}^2$. By Lemma 3.8, we have

$$
\|x - x^*\|_{B_{\tau_2}}^2 \leq \|x - x^*\|_{B_{\tau_1}}^2 \leq R_\lambda(\tau_1, \tau_2) \|x - x^*\|_{B_{\tau_2}}^2
$$

for all $x, x^* \in \mathbb{R}^n$. Hence, by first minimizing in $x^* \in \operatorname{Argmin} f$, and then maximizing in $x \in L_f(x_0)$, we obtain

$$
D_{\tau_2}^2 \leq D_{\tau_1}^2 \leq R_\lambda(\tau_1, \tau_2) D_{\tau_2}^2.
$$

This means that the method with a bigger number of coordinates is always no worse than the corresponding method with a smaller number of coordinates, but it can also be faster up to the ratio (1.1).

3.3. Strongly convex functions. Now let us consider the strongly convex case. For measuring the parameter of strong convexity, it is natural to use the norm $\|\cdot\|_{B_\tau}$. Recall that a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is called *strongly convex* with respect to the norm $\|\cdot\|_{B_\tau}$ if there exists $\mu_\tau > 0$ such that

$$(3.9) \qquad f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu_\tau}{2} \|y - x\|_{B_\tau}^2$$

for all $x, y \in \mathbb{R}^n$. The largest possible value of μ_τ, satisfying (3.9), is called the *modulus* of strong convexity. Observe that if f is additionally 1-smooth with respect to $\|\cdot\|_B$ (see (2.1)), then we must have $\mu_\tau B_\tau \preceq B$. Thus, B cannot be degenerate in this situation.

THEOREM 3.10 (convergence rate for strongly convex functions). *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function which is 1-smooth with respect to the norm $\|\cdot\|_B$, where B is an $n \times n$ real symmetric positive definite matrix, let $1 \leq \tau \leq n$, and let f be strongly convex with respect to the norm $\|\cdot\|_{B_\tau}$ with modulus μ_τ. Let x_0 be a deterministic point in $\mathbb{R}^n$, let $K \geq 1$, and let $(x_k)_{k=1}^K$ be the random points in $\mathbb{R}^n$ generated by $\mathrm{RCDVS}(f, B, \tau, x_0, K)$. Then*

$$(3.10) \qquad \mathbb{E} f(x_k) - \min f \leq (1 - \mu_\tau)^k (f(x_0) - \min f)$$

for all $0 \leq k \leq K$.

Proof. See Appendix B. $\qquad \square$

Since $1 - \mu_\tau \leq e^{-\mu_\tau}$, this means that, given any $\varepsilon > 0$, for achieving accuracy ε in terms of the expected value of the objective, one needs the following number of iterations:

$$K_\tau := \frac{1}{\mu_\tau} \ln \frac{f(x_0) - \min f}{\varepsilon}.$$

For $\tau = 1$, we have $\mu_\tau = \mathrm{Tr}(B)\mu$, where μ is the strong convexity parameter of f in the standard Euclidean norm. This recovers the convergence rate of the RCD method for strongly convex functions from [7].

Similarly to the discussion in section 3.2, let us compare the efficiency estimates for different values of τ. Fix $1 \leq \tau_1 < \tau_2 \leq n$. Then the acceleration rate equals

$$\frac{K_{\tau_1}}{K_{\tau_2}} = \frac{\mu_{\tau_2}}{\mu_{\tau_1}}.$$

Let us compare the constants μ_{τ_1} and μ_{τ_2}. By Lemma 3.8, for all $h \in \mathbb{R}^n$, we have

$$(3.11) \qquad \|h\|_{B_{\tau_2}}^2 \leq \|h\|_{B_{\tau_1}}^2 \leq R_\lambda(\tau_1, \tau_2) \|h\|_{B_{\tau_2}}^2.$$

Hence, by the strong convexity of f in the B_{τ_1}-norm, it follows that

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle \geq \frac{\mu_{\tau_1}}{2} \|y - x\|_{B_{\tau_1}}^2 \geq \frac{\mu_{\tau_1}}{2} \|y - x\|_{B_{\tau_2}}^2$$

for all $x, y \in \mathbb{R}^n$. This means that the modulus of strong convexity of f with respect to $\|\cdot\|_{B_{\tau_2}}$ is at least μ_{τ_1}, i.e., $\mu_{\tau_2} \geq \mu_{\tau_1}$. Similarly, combining the strong convexity of f in the B_{τ_2}-norm with the second inequality in (3.11), we obtain

$$f(y) - f(x) - \langle \nabla f(x), y - x \rangle \geq \frac{\mu_{\tau_2}}{2} \|y - x\|_{B_{\tau_2}}^2 \geq \frac{\mu_{\tau_2}}{2R_\lambda(\tau_1, \tau_2)} \|y - x\|_{B_{\tau_1}}^2$$

for all $x \in \mathbb{R}^n$. This means that $\mu_{\tau_1} \geq \frac{\mu_{\tau_2}}{R_\lambda(\tau_1,\tau_2)}$. Hence, our reasoning shows that

$$\mu_{\tau_1} \leq \mu_{\tau_2} \leq R_\lambda(\tau_1,\tau_2)\mu_{\tau_1}.$$

Thus, we obtain absolutely the same result as in the previous section: the efficiency of the method monotonically improves with τ, and the acceleration factor can reach the ratio (1.1) (for example, one can verify that this is the case for a strictly convex quadratic function).

Remark 3.11. The results presented in this section can be straightforwardly extended from strongly convex functions to a more broader class of *gradient dominated functions of degree* 2, also known as the functions satisfying the *Polyak–Łojasiewicz condition.* For more information and different examples of such functions, see [25]. Note also that recently, in the context of coordinate descent methods, there has appeared an even more general condition, called *generalized error bound property* [4]. However, we do not know whether our results can be extended for this property.

3.4. Proof of Lemma 3.3. In this section, we give the proof of Lemma 3.3 assuming that $n \geq 2$ (otherwise, the claim is trivial). We start with introducing a bit of new notation that will be used only inside this section. For a subset $S \in \binom{[n]}{\tau}$, by I_{-S} we denote the $n \times (n-\tau)$ matrix obtained from the $n \times n$ identity matrix I by removing the columns with indices from S (i.e., $I_{-S} := I_{[n]\setminus S}$). For an $n \times n$ matrix B and a subset $S \in \binom{[n]}{\tau}$, by $B_{-S\times-S}$ we denote the $(n-\tau) \times (n-\tau)$ submatrix obtained from B by removing the rows and columns with indices from S (i.e., $B_{-S\times-S} := I_{-S}^T B I_{-S}$); similarly, for a vector $v \in \mathbb{R}^n$, by v_{-S} we denote the subvector of size $n - \tau$ obtained from v by removing the elements with indices from S (i.e., $v_{-S} := I_{-S}^T v$); for brevity, for each $1 \leq i \leq n$, we also use I_{-i}, $B_{-i\times-i}$, and v_{-i} instead of the more cumbersome $I_{-\{i\}}$, $B_{-\{i\}\times-\{i\}}$, and $v_{-\{i\}}$, respectively.

To prove Lemma 3.3, let us consider the matrix-valued polynomial

$$(3.12) \qquad t \in \mathbb{R} \mapsto P(t) := \mathrm{Adj}(B - tI)$$

and show that the left- and right-hand sides of (3.4) are, up to a constant multiplicative factor, different representations of the $(n-\tau)$th derivative of P at zero.

We start with the easier right-hand side. Using the spectral decomposition $B = Q\,\mathrm{Diag}(\lambda)Q^T$ and the definition of the adjugate matrix, for each $t \in \mathbb{R}$ we readily obtain the following spectral decomposition of P:

$$P(t) = Q\,\mathrm{Diag}(d_1(t),\ldots,d_n(t))Q^T,$$

where $d_i : \mathbb{R} \to \mathbb{R}$ for each $1 \leq i \leq n$ is the polynomial

$$d_i(t) := \prod_{1\leq j\leq n: j\neq i}(\lambda_j(B) - t).$$

Opening the parentheses and grouping the terms by the powers of t, we see that

$$d_i^{(n-\tau)}(0) = (-1)^{n-\tau}(n-\tau)!\sigma_{\tau-1}(\lambda_{-i})$$

for each $1 \leq i \leq n$, and hence

$$(3.13) \qquad P^{(n-\tau)}(0) = (-1)^{n-\tau}(n-\tau)!Q\,\mathrm{Diag}(\sigma_{\tau-1}(\lambda_{-1}),\ldots,\sigma_{\tau-1}(\lambda_{-n}))Q^T.$$

Now we take another approach to calculating the derivative $P^{(n-\tau)}(0)$ by directly differentiating the original expression (3.12). The key inductive step here is the following.

LEMMA 3.12 (inductive step). *Let B be an $n \times n$ ($n \geq 2$) real symmetric matrix, let P be the matrix-valued polynomial* (3.12), *and let $t \in \mathbb{R}$. Then*

$$(3.14) \qquad P'(t) = -\sum_{i=1}^{n} I_{-i} \operatorname{Adj}(B_{-i \times -i} - tI)I_{-i}^{T}.$$

Suppose for the moment that Lemma 3.12 holds, and let $t \in \mathbb{R}$ be arbitrary. Differentiating both sides of (3.14) (each time applying Lemma 3.12 to the matrix $B_{-i \times -i}$) and assuming that $n \geq 3$, we obtain

$$P''(t) = \sum_{i=1}^{n} \sum_{1 \leq j \leq n : j \neq i} I_{-\{i,j\}} \operatorname{Adj}(B_{-\{i,j\} \times -\{i,j\}} - tI)I_{-\{i,j\}}^{T}$$
$$= 2 \sum_{1 \leq i < j \leq n} I_{-\{i,j\}} \operatorname{Adj}(B_{-\{i,j\} \times -\{i,j\}} - tI)I_{-\{i,j\}}^{T}.$$

Similarly, assuming that $n \geq 4$, we have that

$$P'''(t) = -2 \sum_{1 \leq i < j \leq n} \sum_{1 \leq k \leq n : k \neq i,j} I_{-\{i,j,k\}} \operatorname{Adj}(B_{-\{i,j,k\} \times -\{i,j,k\}} - tI)I_{-\{i,j,k\}}^{T}$$
$$= -6 \sum_{1 \leq i < j < k \leq n} I_{-\{i,j,k\}} \operatorname{Adj}(B_{-\{i,j,k\} \times -\{i,j,k\}} - tI)I_{-\{i,j,k\}}^{T},$$

and, more generally (by induction on r), that

$$P^{(r)}(t) = (-1)^{r} r! \sum_{S \in \binom{[n]}{r}} I_{-S} \operatorname{Adj}(B_{-S \times -S} - tI)I_{-S}^{T}$$

for all $0 \leq r \leq n - 1$. In particular,

$$(3.15) \qquad P^{(n-\tau)}(0) = (-1)^{n-\tau}(n-\tau)! \sum_{S \in \binom{[n]}{\tau}} I_{S} \operatorname{Adj}(B_{S \times S})I_{S}^{T}.$$

Equating (3.13) and (3.15), we obtain the claim of Lemma 3.3.

All that remains is to prove Lemma 3.12.

Proof. We begin with a couple of technical simplifications. First, it suffices to prove the claim only for $t = 0$; the general case then follows by replacing B with $B - tI$. Second, we can assume that B and all its $(n-1) \times (n-1)$ principal submatrices $(B_{-1,-1}, \ldots, B_{-n,-n})$ are simultaneously nondegenerate; otherwise, we can replace B with $B + \delta I$ for an arbitrary sufficiently small $\delta > 0$ (such that $B + \delta I$ satisfies the above requirement) and then pass to the limit as $\delta \to 0$ using the continuity of both sides of (3.14) in δ.

Since B is nondegenerate, there exists a sufficiently small neighborhood around zero such that $B - tI$ is nondegenerate for all t from this neighborhood and

$$P(t) = \operatorname{Det}(B - tI)(B - tI)^{-1}$$

by Cramer's rule. Differentiating and denoting $C := B^{-1}$, we obtain

$$P'(0) = -\operatorname{Det}(B)(\operatorname{Tr}(B^{-1})B^{-1} - B^{-2}) = -\operatorname{Det}(B) \sum_{i=1}^{n} (C_{ii}C - c_i c_i^{T}),$$

where $c_1, \ldots, c_n \in \mathbb{R}^n$ are the columns of C. Observe that the ith row and the i-th column of the matrix $C_{ii}C - c_i c_i^T$ for each $1 \leq i \leq n$ consist entirely of zeros, so

$$P'(0) = -\operatorname{Det}(B) \sum_{i=1}^{n} I_{-i}(C_{ii}C_{-i \times -i} - (c_i)_{-i}(c_i)_{-i}^T) I_{-i}^T.$$

Thus, to obtain the claim, it suffices to demonstrate that

$$\operatorname{Det}(B)(C_{ii}C_{-i \times -i} - (c_i)_{-i}(c_i)_{-i}^T) = \operatorname{Adj}(B_{-i \times -i})$$

for all $1 \leq i \leq n$. Replacing B with $P^T B P$ if necessary (where P is the $n \times n$ permutation matrix obtained from the identity matrix by moving the ith column into the end), it is enough to consider only the case $i = n$. Let

$$B = \begin{pmatrix} F & z \\ z^T & \alpha \end{pmatrix},$$

where F is the top-left $(n-1) \times (n-1)$ principal submatrix of B, $z \in \mathbb{R}^{n-1}$ is the right-most column of B with the last element removed, and $\alpha := B_{nn}$ is the element in the lower-right corner. Note that F is nondegenerate as a principal $(n-1) \times (n-1)$ submatrix of B (by the technical assumption made at the very beginning). Using the formula for inverting a block matrix, we obtain that $\alpha - \langle F^{-1}z, z \rangle \neq 0$, and

$$C = B^{-1} = \begin{pmatrix} F^{-1} + \frac{F^{-1}zz^T F^{-1}}{\alpha - \langle F^{-1}z, z \rangle} & -\frac{F^{-1}z}{\alpha - \langle F^{-1}z, z \rangle} \\ -\frac{z^T F^{-1}}{\alpha - \langle F^{-1}z, z \rangle} & \frac{1}{\alpha - \langle F^{-1}z, z \rangle} \end{pmatrix}.$$

In particular, we see that

$$C_{nn}C_{-n \times -n} - (c_n)_{-n}(c_n)_{-n}^T = \frac{F^{-1}}{\alpha - \langle F^{-1}z, z \rangle}.$$

Since by Cramer's rule

$$\operatorname{Adj}(B_{-n \times -n}) = \operatorname{Adj}(F) = \operatorname{Det}(F)F^{-1},$$

it remains to check whether

$$\operatorname{Det}(B) = \operatorname{Det}(F)(\alpha - \langle F^{-1}z, z \rangle).$$

But this is exactly the formula for the determinant of a block matrix. $\square$

4. Implementation of volume sampling. Let B be an $n \times n$ real symmetric positive semidefinite matrix, and let $1 \leq \tau \leq \operatorname{Rank}(B)$. In this section, we discuss how to generate a random variable $S_0 \sim \operatorname{Vol}_\tau(B)$ according to volume sampling.

4.1. General algorithm. Recall that volume sampling is sampling with a finite number of outcomes. Thus, in principle, S_0 can be generated by any general method for generating random variables taking a finite number of values. Let us briefly review one such method which is based on the following result.

PROPOSITION 4.1 (generating a random variable taking a finite number of values). *Let $X := \{x_1, \ldots, x_N\}$ be a finite set, and let $p_1, \ldots, p_N$ be nonnegative numbers such that $\sum_{k=1}^{N} p_k = 1$. For each $1 \leq k \leq N$, let $P_k := \sum_{k'=1}^{k} p_{k'}$. Let u be a random variable uniformly distributed on the interval $(0,1)$, and let $\xi := x_{k_0}$, where $k_0 := \min\{1 \leq k \leq N : u \leq P_k\}$. Then ξ is a well-defined random variable taking values in X such that $\mathbb{P}(\xi = x_k) = p_k$ for all $1 \leq k \leq N$.*

Proof. For convenience, denote $P_0 := 0$. Observe that $0 = P_0 \leq P_1 \leq \cdots \leq P_N = 1$. Thus, $k_0 = k$ for some $1 \leq k \leq N$ iff u belongs to the interval $(P_{k-1}, P_k]$. Since these intervals are disjoint and their union is $(0, 1]$, the variable k_0 is well-defined. Hence, ξ is well-defined, and $\mathbb{P}(\xi = x_k) = \mathbb{P}(k_0 = k) = \mathbb{P}(P_{k-1} < u \leq P_k) = P_k - P_{k-1} = p_k$ for each $1 \leq k \leq N$. □

Proposition 4.1 is in fact a two-stage algorithm for generating a random variable ξ taking values in $\{x_1, \ldots, x_N\}$ given the corresponding list of probabilities $p_1, \ldots, p_N$. At the first stage of this algorithm (called *preprocessing*), we compute the cumulative sums $P_1, \ldots, P_N$. This requires $O(N)$ operations. At the second stage (called *sampling*), we first generate a random variable u uniformly distributed on $(0, 1)$, then compute k_0, and finally output $\xi := x_{k_0}$. Since the cumulative sums $(P_k)_{1 \leq k \leq N}$ are monotonically increasing, one can use binary search for efficiently finding k_0. Thus, the complexity of sampling is just $O(\ln N)$ operations. Note that the preprocessing has to be done only once; after this, one can generate arbitrarily many independent samples using the sampling routine.

In the case of volume sampling, the above procedure looks as follows. At the preprocessing stage, we iterate over all $N = \binom{n}{\tau}$ possible τ-element subsets of $[n]$, computing the corresponding principal minors of B and corresponding cumulative sums. This requires $O(\binom{n}{\tau}\tau^3)$ operations in total assuming that the complexity of calculating a minor of size τ is $O(\tau^3)$. During sampling, we use binary search to find the number k_0 and then return the set $S_0 \in \binom{[n]}{\tau}$ corresponding to k_0. Each sampling thus requires $O(\ln \binom{n}{\tau})$ operations.

Unfortunately, the above $O(\binom{n}{\tau}\tau^3)$ preprocessing time makes the general algorithm impractical for most values of τ. Nevertheless, it is still applicable for several very small values of τ. For example, when $\tau = 1$, the preprocessing time and memory complexities are both $O(n)$, while the sampling time and memory complexities are $O(\ln n)$ and $O(1)$, respectively. Another interesting regime is $\tau = 2$. In this case, the preprocessing time and memory complexities are both $O(n^2)$, while the sampling time and memory complexities are the same as before. Note that in many applications the objective function $f : \mathbb{R}^n \to \mathbb{R}$ has the form $f(x) := \phi(Ax, x)$, where A is a real $m \times n$ matrix, and $\phi : \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}$ is a function that can be computed in time $O(m + n)$ (see section 5 for different examples). In these applications, the $O(n^2)$ memory is comparable to the cost of storing A, and the $O(n^2)$ time is comparable to the cost of one computation of the objective and is often allowable (see also section 4.2 for a possible treatment of sparsity).

While the general procedure described above is not polynomial, we note that there are more specialized methods for volume sampling that are polynomial (e.g., see [20] for an exact algorithm and several efficient approximate ones). However, they are designed for generating just one sample and therefore are not directly suited for using them inside optimization algorithms, where one needs a very fast sampling routine that can be called at each iteration. Perhaps it is possible to modify these methods by properly splitting them into a polynomial preprocessing stage and an independent sampling stage which is much faster, but we have not investigated this direction.

4.2. Two-element volume sampling for sparse matrices. Now suppose that the matrix B is sparse and our goal is to implement two-element volume sampling. The general algorithm described above requires $O(n^2)$ time and $O(n^2)$ memory for preprocessing, which may be too expensive when n is large. In this section, we present a special method that takes into account the sparsity of B and whose preprocessing time and memory complexities are both $O(\mathrm{nnz}(B) + n)$, where $\mathrm{nnz}(B)$ is the number

TABLE 1
Time and memory complexities of 2-element volume sampling for a dense and sparse matrix.

	Preprocessing		Sampling	
	Time	Memory	Time	Memory
Dense	$O(n^2)$	$O(n^2)$	$O(\ln n)$	$O(1)$
Sparse	$O(\mathrm{nnz}(B) + n)$	$O(\mathrm{nnz}(B) + n)$	$O(\ln n)$	$O(1)$

of nonzero elements of B. When B is dense, $\mathrm{nnz}(B) = n^2$, but it can be much smaller than n^2 if B is sufficiently sparse. Once the preprocessing is done, each sampling then has the $O(\ln n)$ time complexity and the $O(1)$ memory complexity, which are exactly the same complexities as those of the general algorithm from the previous section (see Table 1 for a comparison).

Assuming that the matrix B is given in the *compressed sparse row (CSR) format*[1] for each $1 \le i \le n$, we know the r_i indices (possibly zero) $i \le j_1^{(i)} < \cdots < j_{r_i}^{(i)} \le n$ of all nonzero elements in the ith row of B which are located to the right of the diagonal (thus, $\{j_1^{(i)}, \ldots, j_{r_i}^{(i)}\} = \{i \le j \le n : B_{ij} \ne 0\}$), as well as the corresponding values $v_1^{(i)}, \ldots, v_{r_i}^{(i)}$ of these elements (thus, $v_k^{(i)} := B_{ij_k^{(i)}}$ for all $1 \le k \le r_i$). For notational convenience, for each $1 \le i < j \le n; 1 \le k \le r_i$ we also define $j_{r_i+1}^{(i)} := n + 1$ and

$$P_{v,h,t}(i,j,k) := v_1^{(i)}(t_i - t_{j+1}) - h_k^{(i)},$$

where $h^{(1)} \in \mathbb{R}^{r_1}, \ldots, h^{(n-1)} \in \mathbb{R}^{r_{n-1}}$, $t \in \mathbb{R}^{n+1}$ are given, and $h := (h^{(1)}, \ldots, h^{(n-1)})$.

Now the notation has been established and we are ready to present the algorithm. Similarly to the method from the previous section, it is a two-stage procedure that consists of an expensive preprocessing stage (Algorithm 4.1) and a cheap sampling stage (Algorithm 4.2) that can be executed as many times as one wishes once the preprocessing has terminated.

Algorithm 4.1 sparse2vs_preprocess(B).

Require: B: $n \times n$ real symmetric positive semidefinite matrix of rank at least two specified by its CSR-format $(r_1, \ldots, r_n, j^{(1)}, \ldots, j^{(n)}, v^{(1)}, \ldots, v^{(n)})$.

1: For each $1 \le i \le n - 1$, compute $h^{(i)} \in \mathbb{R}^{r_i}$, defined by $h_k^{(i)} := \sum_{k'=1}^{k} (v_{k'}^{(i)})^2$.

2: Compute $t \in \mathbb{R}^{n+1}$, defined by $t_i := \sum_{j=i}^{n} v_1^{(j)}$ (and $t_{n+1} := 0$).

3: Compute $q \in \mathbb{R}^{n-1}$, defined by $q_i := \sum_{i'=1}^{i} P_{v,h,t}(i', n, r_{i'})$.

4: **return** $(h^{(1)}, \ldots, h^{(n-1)}, t, q)$

Clearly, the time and memory complexities of the preprocessing stage are both $O(\mathrm{nnz}(B) + n)$. The sampling stage consists of three successive binary searches over certain subsets of $[n]$ and thus has the $O(\ln n)$ time complexity and the $O(1)$ memory complexity when properly implemented (the function $P_{v,h,t}$ should be computed on the fly inside each binary search).

We now prove the correctness of the algorithm.

---

[1] Strictly speaking, the classical CSR format is different from the one we are describing here. In the classical CSR format, the index vectors $j^{(1)}, \ldots, j^{(n)}$ are concatenated into one large vector; similarly, the value vectors $v^{(1)}, \ldots, v^{(n)}$ are concatenated into one large vector, and instead of the numbers $r_1, \ldots, r_n$ one stores their cumulative sums. Nevertheless, the format we are describing is algorithmically equivalent to the original CSR format and one can be transformed into the other in a straightforward manner without any time or memory overhead.

Algorithm 4.2 sparse2vs_sample$(B, h^{(1)}, \ldots, h^{(n-1)}, t, q)$.

Require: B: $n \times n$ real symmetric positive semidefinite matrix of rank at least two specified by its CSR-format $(r_1, \ldots, r_n, j^{(1)}, \ldots, j^{(n)}, v^{(1)}, \ldots, v^{(n)})$; $h^{(1)} \in \mathbb{R}^{r_1}, \ldots, h^{(n-1)} \in \mathbb{R}^{r_{n-1}}$; $t \in \mathbb{R}^{n+1}$; $q \in \mathbb{R}^{n-1}$.

1: Independently generate random u_1, u_2 uniformly distributed on $(0, 1)$.

2: Find $i_0 := \min\{1 \leq i \leq n - 1 : u_1 \leq \frac{q_i}{q_{n-1}}\}$ using binary search.

3: Find $k_l := \min\{1 \leq k \leq r_{i_0} : u_2 \leq \frac{P_{v,h,t}(i_0, j_{k+1}^{(i_0)} - 1, k)}{P_{v,h,t}(i_0, n, r_{i_0})}\}$ using binary search.

4: Find $j_0 := \min\{j_{k_l}^{(i_0)} \leq j \leq j_{k_l+1}^{(i_0)} - 1 : u_2 \leq \frac{P_{v,h,t}(i_0, j, k_l)}{P_{v,h,t}(i_0, n, r_{i_0})}\}$ using binary search.

5: **return** $\{i_0, j_0\}$

THEOREM 4.2. *Let B be an $n \times n$ real symmetric positive semidefinite matrix with rank at least two. Let $(h^{(1)}, \ldots, h^{(n-1)}, t, q) := \text{sparse2vs\_preprocess}(B)$, and let $S_0 := \text{sparse2vs\_sample}(B, h^{(1)}, \ldots, h^{(n-1)}, t, q)$. Then S_0 is a well-defined random variable distributed according to $\text{Vol}_2(B)$.*

Proof. Observe that, for each $1 \leq i \leq n - 1$ and each $1 \leq k \leq r_i$, we have

$$h_k^{(i)} = \sum_{j=i}^{j_k^{(i)}} B_{ij}^2, \qquad h_{r_i}^{(i)} = \sum_{j=i}^{n} B_{ij}^2, \qquad v_1^{(i)} = B_{ii}, \qquad t_i = \sum_{j=i}^{n} B_{jj}.$$

For each $1 \leq i < j \leq n$, denote

$$P(i, j) := \sum_{j'=i+1}^{j} \text{Det}(B_{\{i,j\} \times \{i,j\}}).$$

Then, for each $1 \leq i \leq n - 1$, it holds that

$$v_1^{(i)} t_i - h_{r_i}^{(i)} = \sum_{j=i}^{n} (B_{ii} B_{jj} - B_{ij}^2) = P(i, n).$$

From Proposition 4.1, it follows that i_0 is a well-defined random variable taking values in $\{1, \ldots, n-1\}$ with probabilities

(4.1)
$$\mathbb{P}(i_0 = i) = \frac{P(i, n)}{\sum_{i'=1}^{n-1} P(i', n)}.$$

Next observe that, for each $1 \leq i \leq n - 1$ and $1 \leq k \leq r_i$, we have

$$P(i, j, k) = B_{ii} \sum_{j'=i}^{j} B_{j'j'} - \sum_{j'=i}^{j_k^{(i)}} B_{ij'}^2 = \sum_{j'=i}^{j} (B_{ii} B_{j'j'} - B_{ij'}^2) = P(i, j)$$

for all $j_k^{(i)} \leq j \leq j_{k+1}^{(i)} - 1$. In particular, $P(i_0, j_{k+1}^{(i_0)} - 1, k) = P(i_0, j_{k+1}^{(i_0)} - 1)$ for all $1 \leq k \leq r_{i_0}$, from which we see that $P(i_0, j_{k+1}^{(i_0)} - 1, k)$ monotonically increases with k (since $j_{k+1}^{(i_0)}$ does so and $P(i_0, j)$ monotonically increases with j) until it reaches $P(i_0, n, r_{i_0}) = P(i_0, n)$ when $k = r_{i_0}$. This shows that k_l is well-defined. Similarly, we can write $P(i_0, j, k_l) = P(i_0, j)$ for all $j_{k_l}^{(i_0)} \leq j \leq j_{k_l+1}^{(i_0)} - 1$, from which it follows

that $P(i_0, j, k_l)$ monotonically increases with j. Combining this with the definition of k_l, which gives

$$\frac{P(i_0, j_{k_l}^{(i_0)})}{P(i, n)} \leq u_2 \leq \frac{P(i_0, j_{k_l+1}^{(i_0)} - 1)}{P(i, n)},$$

we conclude that j_0 is well-defined and in fact

$$j_0 = \min\left\{ i_0 + 1 \leq j \leq n : u_2 \leq \frac{P(i_0, j)}{P(i_0, n)} \right\}.$$

Applying Proposition 4.1 again, we obtain that, conditioned on $i_0 = i$, the random variable j_0 takes values in $\{i + 1, \ldots, n\}$ such that

$$\mathbb{P}(j_0 = j \mid i_0 = i) = \frac{\mathrm{Det}(B_{\{i,j\} \times \{i,j\}})}{P(i, n)}$$

for all $1 \leq i < j \leq n$. Undoing the conditioning, we see that (i_0, j_0) is a well-defined random variable taking values in $\{(i, j) : 1 \leq i < j \leq n\}$ with probabilities

$$\mathbb{P}(i_0 = i; j_0 = j) = \mathbb{P}(i_0 = i)\mathbb{P}(j_0 = j \mid i_0 = i) = \frac{\mathrm{Det}(B_{\{i,j\} \times \{i,j\}})}{\sum_{i'=1}^{n-1} P(i', n)},$$

and the claim follows.   □

5. Examples of applications. Now we consider several examples of objective functions for which it is possible to apply the RCDVS method and discuss different implementation details.

5.1. Quadratic function. Our first example of an objective function is the convex quadratic $f : \mathbb{R}^n \to \mathbb{R}$, defined by

$$f(x) := \frac{1}{2}\langle Ax, x \rangle - \langle b, x \rangle,$$

where A is a given $n \times n$ real symmetric positive semidefinite matrix and $b \in \mathbb{R}^n$. This function is 1-smooth with respect to the seminorm $\|\cdot\|_A$, so one can minimize it by RCDVS with

$$B := A.$$

For doing volume sampling, one can use either the general algorithm from section 4.1 when the matrix A is dense or the special one from section 4.2 when A is sparse.

5.2. Separable problems. The second example gives rise to a whole family of objective functions $f : \mathbb{R}^n \to \mathbb{R}$ that are admissible for the RCDVS method and can be obtained by composing some smooth separable function with a linear mapping:

$$f(x) := \sum_{i=1}^{m} g_i(\langle a_i, x \rangle).$$

Here $a_1, \ldots, a_m \in \mathbb{R}^n$ are given vectors and $g_1, \ldots, g_m : \mathbb{R} \to \mathbb{R}$ are univariate functions such that g_i is L_i-smooth ($L_i \geq 0$) for each $1 \leq i \leq n$, meaning that it is differentiable and satisfies

$$g_i(t) \leq g_i(t_0) + g_i'(t_0)(t - t_0) + \frac{L_i}{2}(t - t_0)^2$$

for all $t, t_0 \in \mathbb{R}$. It is easy to see from the definitions that the resulting function f turns out to be 1-smooth with respect to the seminorm $\|\cdot\|_B$, where

$$(5.1) \qquad B := \sum_{i=1}^{m} L_i a_i a_i^T.$$

Example 5.1 (least squares). Let f be the least squares function

$$f(x) := \frac{1}{2} \sum_{i=1}^{m} (\langle a_i, x \rangle - b_i)^2,$$

where $b_1, \ldots, b_m \in \mathbb{R}$. In this case, $g_i(t) := \frac{1}{2}(t - b_i)^2$ is the quadratic function with $L_i = 1$ for each $1 \leq i \leq m$.

Example 5.2 (logistic regression). Let f be the logistic regression function

$$\sum_{i=1}^{m} \ln(1 + e^{-b_i \langle a_i, x \rangle}),$$

where $b_1, \ldots, b_m \in \{-1, 1\}$. In this case, $g_i(t) := \ln(1 + e^{-b_i t})$ is the logistic function with $L_i = \frac{1}{4}$ for each $1 \leq i \leq m$.

The matrix B can be computed in $O(mn^2)$ operations. If n is sufficiently small, this can be done rather efficiently, and then one can apply RCDVS for several small values of τ (e.g., $\tau = 2, 3$, etc.).

If n is large, one can still use RCDVS provided that the vectors $a_1, \ldots, a_m$ are *sparse*. Indeed, observe that the number of nonzero elements in B is bounded above by $\sum_{i=1}^{m} p_i$, where p_i for each $1 \leq i \leq n$ denotes the number of nonzero elements in a_i. Furthermore, a sparse representation of B (e.g., the commonly used sparse compressed row/column formats) can be obtained in

$$(5.2) \qquad O\left(\sum_{i=1}^{m} p_i^2 + n\right)$$

operations (possibly with some logarithmic terms when a further sorting of indices is needed). After this, one can use the efficient algorithm for sparse two-element volume sampling from section 4.2, whose preprocessing complexity is the same as (5.2). For example, if for all $1 \leq i \leq n$ we have $p_i \leq p$, where p is some sufficiently small integer, then both the computation of B and the preprocessing procedure take

$$O(mp^2 + n)$$

operations, which is only linear in both m and n.

5.3. Smoothing technique. Another interesting and quite rich source of examples comes from the smoothing technique [26, 27], which we now briefly review. Let $g : \mathbb{R}^m \to \mathbb{R}$ be a convex function. By the Fenchel–Moreau theorem, we can write

$$g(y) = \max_{s \in G_*} \{\langle y, s \rangle - g^*(s)\}$$

for all $y \in \mathbb{R}^m$, where $g^* : G_* \to \mathbb{R}$ is the Fenchel conjugate of g with the effective domain G_* (assume that G_* is bounded). Let $\omega^* : \Omega_* \to \mathbb{R}$ be a distance generating

function on G_* with respect to the standard Euclidean norm $\|\cdot\|$ (i.e., a nonnegative closed convex function with domain $\Omega_* \supseteq G_*$ that is 1-strongly convex on G_* with respect to $\|\cdot\|$). Let $\mu > 0$, and let $g_\mu : \mathbb{R}^n \to \mathbb{R}$ be the function

$$g_\mu(y) := \max_{s \in G_*}\{\langle y, s\rangle - g^*(s) - \mu\omega^*(s)\}.$$

It is known that g_μ satisfies $g_\mu(y) \leq g(y) \leq g_\mu(y) + \mu\max_{G_*}\omega^*$ for all $y \in \mathbb{R}^m$, and moreover it is $\frac{1}{\mu}$-smooth with respect to $\|\cdot\|$. Thus, g_μ can be seen as a smooth uniform approximation of g, where the parameter μ controls both the accuracy of approximation and its level of smoothness.

Now let A be an $m \times n$ real matrix, let $b \in \mathbb{R}^n$, and define $f, f_\mu : \mathbb{R}^n \to \mathbb{R}$ by

$$f(x) := g(Ax - b), \qquad f_\mu(x) := g_\mu(Ax - b).$$

It is easy to see that $f_\mu(x) \leq f(x) \leq f_\mu(x) + \mu\max_{G_*}\omega^*$ for all $x \in \mathbb{R}^n$. Furthermore, the function f_μ is $\frac{1}{\mu}$-smooth with respect to the seminorm $\|\cdot\|_{A^T A}$. Thus, the problem of minimizing f can be replaced with the problem of minimizing its smooth approximation f_μ for some carefully chosen value of μ. This latter problem can be solved by RCDVS with

$$B := \frac{1}{\mu}A^T A.$$

This matrix has exactly the same structure as the one from (5.1).

Example 5.3. The function

$$f_\mu(x) := \mu\ln\left(\sum_{i=1}^m e^{(Ax-b)_i/\mu}\right) - \mu\ln m$$

is obtained from $g(y) := \max\{y_1, \ldots, y_m\}$ using the negative entropy function $\omega^*(s) := \sum_{i=1}^m s_i\ln s_i$ with domain $\Omega_* := \{s \in \mathbb{R}^m : \sum_{i=1}^m s_i = 1; s_1, \ldots, s_m \geq 0\}$.

Example 5.4. The function

$$f_\mu(x) := \sum_{i=1}^m H_\mu((Ax - b)_i),$$

where $H_\mu : \mathbb{R} \to \mathbb{R}$ is the *Huber function*

$$H_\mu(t) := \begin{cases} \frac{t^2}{2\mu} & \text{if } |t| \leq \mu, \\ |t| - \frac{\mu}{2} & \text{otherwise,} \end{cases}$$

is obtained from the l^1-norm $g(y) := \|y\|_1$ using the Euclidean distance generating function $\omega^*(s) := \frac{1}{2}\|s\|^2$ with domain $\Omega_* := \mathbb{R}^m$.

Example 5.5. The function

$$f_\mu(x) := \sqrt{\|Ax - b\|^2 + \mu^2} - \mu$$

is obtained from $g(y) := \|y\|$ using the function $\omega^*(s) := 1 - \sqrt{1 - \|s\|^2}$ with domain $\Omega_* := \{s \in \mathbb{R}^n : \|s\| \leq 1\}$.

5.4. Combinations of previous examples. Finally, one can take nonnegative linear combinations of the already considered examples. Indeed, let $f_1, \ldots, f_r : \mathbb{R}^n \to \mathbb{R}$ be functions, where f_i for each $1 \leq i \leq r$ is 1-smooth with respect to an $n \times n$ real symmetric positive semidefinite matrix B_i, and let $\alpha_1, \ldots, \alpha_r > 0$. Then the sum $f := \sum_{i=1}^r \alpha_i f_i$ is 1-smooth with respect to the seminorm $\|\cdot\|_B$ with $B := \sum_{i=1}^r \alpha_i B_i$.

Example 5.6. Let f be the l^2-*regularized logistic regression function*

$$f(x) := \sum_{i=1}^m \ln(1 + e^{-b_i \langle a_i, x \rangle}) + \frac{\gamma}{2} \|x\|^2,$$

where $a_1, \ldots, a_m \in \mathbb{R}^n$, $b_1, \ldots, b_m \in \{-1, 1\}$, $\gamma > 0$. In this case,

$$B := \frac{1}{4} \sum_{i=1}^m a_i a_i^T + \gamma I.$$

6. Numerical experiments. In this section, we investigate the practical behavior of RCDVS and compare it with that of a couple of other already known methods.

The first one is the RCD method. Recall that it is in fact the same method as RCDVS with $\tau = 1$. In comparing RCDVS with RCD, we are interested in investigating how the *actual* acceleration ratio of RCDVS corresponds to our theoretical prediction (see (1.1)):

$$(6.1) \qquad R_\lambda(1, \tau) = \frac{\sum_{i=1}^n \lambda_i}{\sum_{i=\tau}^n \lambda_i}.$$

The difference between the actual acceleration ratio and the theoretical one is that the latter is the ratio of the theoretical upper bounds on the performance of the methods, while the former is the ratio of the real number of iterations performed by the methods on a particular problem instance.

The second one, denoted by SDNA, is Method 1 from [16], which uses so-called τ-nice sampling. As was already discussed in section 1.1, this method is exactly the same method as RCDVS, with the only difference that it uses *uniform* sampling (without replacement) instead of volume sampling. In comparing RCDVS with SDNA, we are interested in seeing how important the sampling strategy is to the performance of the general coordinate descent scheme that we consider in this paper.

6.1. Quadratic function. For the first experiment, we have chosen the convex quadratic function from section 5.1 and have set $\tau = 2$. Our goal is to observe how the behavior of the methods changes when the spectral gap between the two largest eigenvalues of A increases. For this, we construct the matrix A as follows. First, we choose some $\lambda_1 \geq \lambda_2 := 100$ and set $A := \mathrm{Diag}(\lambda_1, \lambda_2, 1, \ldots, 1)$. Then we successively perform 10 random Householder reflections $A \mapsto (I - 2uu^T)A(I - 2uu^T)$ on the rows and columns of A, where each time the direction u is sampled uniformly from the unit sphere in $\mathbb{R}^n$. Observe that, by construction, the eigenvalues of A are exactly $\lambda_1, \lambda_2, 1, \ldots, 1$. Once A is constructed, we set $b := Ax^*$, where x^* is generated from the uniform distribution on the hypercube $[-1, 1]^n$, and run each method from $x_0 := 0$ until the objective value becomes ε-close to the optimal one for $\varepsilon := 0.01$. This procedure is repeated 10 times to take into account the randomness in the data.

The results of the experiment[2] for different values of n and the eigenvalue ratio

---

[2] All experimental results in this paper were obtained on a laptop with the Intel Core i7-8650U CPU (1.90GHz x 8) and 16 GB DDR4 RAM; no parallelism was used. The source code is available online from https://github.com/arodomanov/rcdvs.

TABLE 2
Results for the quadratic function.

Parameters		RCD		SDNA			RCDVS			
n	λ_1/λ_2	It	T	It	Acc	T	It	Acc	%	T
400	4	5	0.0	15	0.3	0.1	2	2	118	0.05
	16	12	0.1	44	0.4	0.2	2	4	105	0.05
	64	37	0.2	92	0.4	0.4	3	11	83	0.05
	256	126	0.4	125	1.2	0.5	3	40	77	0.05
	1,024	499	1.3	137	3.8	0.6	3	132	64	0.06
800	4	8	0.1	7	1.0	0.1	4	2	148	0.19
	16	18	0.2	50	0.4	0.4	5	3	140	0.20
	64	47	0.3	156	0.3	1.1	5	9	115	0.20
	256	163	0.7	356	0.5	2.8	6	27	91	0.21
	1,024	576	2.5	526	1.1	4.3	6	97	84	0.21
1,600	4	18	0.5	22	0.8	1.0	7	2	189	1.03
	16	24	0.5	69	0.4	3.0	9	2	134	1.05
	64	65	0.9	237	0.3	10.1	10	6	125	1.06
	256	202	2.1	794	0.3	35.0	13	14	87	1.17
	1,024	758	6.7	2,034	0.3	86.6	14	48	79	1.20
3,200	4	28	1.7	39	0.8	3.7	15	1	167	4.14
	16	41	2.1	107	0.4	9.7	18	2	151	4.32
	64	81	3.0	428	0.2	39.4	24	3	116	4.75
	256	228	6.0	1,589	0.1	144.7	23	9	113	4.74
	1,024	828	17.1	5,026	0.2	468.0	26	31	97	4.91

λ_1/λ_2 are shown in Table 2. Each column in this table displays the median value of the corresponding statistic: "It" is the total number of iterations (in thousands) taken by the method until its termination; "T" is the corresponding total running time (in seconds); "Acc" is the actual acceleration ratio of the method over RCD in terms of the number of iterations (this ratio may be less than 1 when there is no acceleration); "%" expresses the "Acc" for RCDVS as a percent of the theoretical prediction (6.1).[3] From this table, we can see that the number of iterations for RCD and SDNA grows significantly with the spectral gap between the two largest eigenvalues, while for RCDVS there is almost no growth at all. As a result, RCDVS dramatically outperforms the other two methods both in terms of iterations and total running time, especially for large values of λ_1/λ_2. By inspecting the "Acc" column, we observe that the actual acceleration ratio of RCDVS with respect to RCD monotonically increases with the spectral gap, which is natural. What is more important is that the "%" always takes values around 100, which means that our theoretical prediction (6.1) is quite accurate. SDNA, on the contrary, performs even worse than RCD in most cases.

6.2. Huber function. In the second experiment, we still use $\tau = 2$ but now consider the Huber function from Example 5.4. In contrast to the previous one, this objective is nonstrongly convex.

The design of the experiment is almost the same as before. To generate the matrix A, we choose $\lambda_1 \geq \lambda_2 := 100$, set A to be the $m \times n$ diagonal matrix with elements $\sqrt{\lambda_1/\mu}, \sqrt{\lambda_2/\mu}, \sqrt{1/\mu}, \dots, \sqrt{1/\mu}$, where $\mu := 0.01$, and then successively perform 10 random Householder reflections $A \mapsto (I - 2uu^T)A(I - 2vv^T)$ on the rows and columns of A, where each time u and v are uniformly distributed on the unit spheres in $\mathbb{R}^m$

---

[3]To keep the table concise, we report only the first few significant digits for each statistic. For example, in the first row of the table, the value 2 in the column "It" for RCDVS may actually stand for 2.1, 2.53, or even 2.999.

TABLE 3
Results for the Huber function.

Parameters			RCD		SDNA			RCDVS			
m	n	λ_1/λ_2	It	T	It	Acc	T	It	Acc	%	T
400	800	4	14	0.3	7	1.9	0.2	6	2	131	0.3
		16	34	0.5	8	3.9	0.2	7	4	111	0.3
		64	114	1.6	10	11.4	0.2	8	13	99	0.3
		256	409	6.3	12	34.3	0.3	8	49	94	0.3
		1,024	1,529	21.5	14	118.7	0.3	8	168	81	0.3
800	400	4	13	0.3	20	0.7	0.6	5	2	147	0.2
		16	31	0.7	57	0.6	1.4	6	4	107	0.2
		64	103	2.0	118	0.9	2.8	7	13	96	0.2
		256	337	6.5	165	2.2	4.3	7	44	84	0.2
		1,024	1,402	27.3	184	7.5	4.8	7	172	83	0.2
800	1,600	4	23	0.5	23	0.9	0.9	11	2	143	1.0
		16	45	1.0	22	2.1	0.9	13	3	122	1.1
		64	122	2.9	23	5.9	1.0	14	8	101	1.3
		256	456	9.6	23	18.8	1.0	15	28	98	1.2
		1,024	1,646	33.8	28	57.3	1.1	16	104	91	1.2
1,600	800	4	22	0.9	25	0.8	1.7	10	2	155	0.8
		16	44	1.6	66	0.7	4.6	13	3	120	0.9
		64	119	4.2	184	0.6	11.7	14	8	101	1.0
		256	438	15.2	406	1.0	26.4	14	29	98	1.0
		1,024	1,588	55.1	625	2.6	40.9	15	98	85	1.0

and $\mathbb{R}^n$, respectively. Such a construction ensures that the matrix $B := \frac{1}{\mu} A^T A$ (see section 5.3) has eigenvalues $\lambda_1, \lambda_2, 1, \ldots, 1$ (plus $n-m$ zeros when $m < n$). The vector b, the starting point x_0, and the termination criterion for the methods are absolutely the same as in the previous experiment.

One additional remark should be made in the case when $m < n$. Recall that in this situation B is in fact degenerate, and hence some of its principal submatrices may not be invertible. This does not cause any difficulties for RCD and RCDVS since the probability of choosing a degenerate submatrix in these methods is zero. However, this is not so for SDNA, and thus, strictly speaking, SDNA is not well-defined in the case of a degenerate matrix B. To fix this problem, we use the More–Penrose pseudoinverse instead of the usual inverse in this method. (We have also tried to simply skip the update when a degenerate submatrix has been chosen, but this strategy turned out to work somewhat worse.)

The results of the experiment are shown in Table 3. In the same way as before, we see that RCDVS always significantly outperforms RCD and its actual acceleration ratio is quite close to the theoretical prediction. However, it is interesting that this time SDNA works quite well for problems with $m < n$ (almost comparably to RCDVS). In particular, its number of iterations is almost independent of the spectral gap. Nevertheless, for $n > m$ its behavior is the same as in the previous experiment.

Now let us consider the same problem but with much bigger dimensions. For this, we slightly change the way we construct A. This time, we generate it as a *sparse* matrix using the procedure described above but with sparse directions u and v that are chosen as follows. First, we take some integer $1 \le p \le \min\{m, n\}$ that controls the resulting sparsity level of B. After this, we pick p random uniformly distributed indices and fill the positions corresponding to these indices with a random vector uniformly distributed on the unit sphere in $\mathbb{R}^p$; the rest of the positions are set to zero. Of course, in order to work with a sparse problem, every method has to be properly modified. In particular, we should use the special algorithm from section 4.2

TABLE 4

Results for the Huber function (sparse matrix). For dimensions $8,000 \times 16,000$ and $16,000 \times 8,000$, we use $p := 50$, while for $16,000 \times 32,000$ and $32,000 \times 16,000$ we use $p := 70$.

Parameters			RCD		SDNA			RCDVS			
m	n	λ_1/λ_2	It	T	It	Acc	T	It	Acc	%	T
8,000	16,000	64	352	3.8	2,981	0.1	36.9	125	2	153	1.8
		256	710	7.3	6,326	0.1	77.3	151	4	111	2.1
		1,024	2,328	24.5	12,250	0.2	154.0	169	13	101	2.3
		4,096	8,855	88.3	24,742	0.3	304.2	176	49	95	2.3
		16,384	35,917	363.9	50,049	0.7	624.9	180	200	98	2.5
16,000	8,000	64	332	3.6	1,351	0.2	19.3	136	2	134	1.9
		256	727	7.7	2,654	0.3	37.9	166	4	107	2.2
		1,024	2,306	23.3	5,364	0.5	76.0	170	13	102	2.3
		4,096	8,648	87.1	10,682	0.9	154.2	175	50	98	2.3
		16,384	37,623	378.3	21,313	1.9	303.2	181	214	105	2.4
16,000	32,000	64	518	5.6	5,829	0.1	73.1	232	2	158	3.7
		256	944	9.7	14,751	0.1	184.1	282	3	128	4.2
		1,024	2,642	27.7	32,066	0.1	410.1	352	7	105	5.0
		4,096	9,539	97.2	64,144	0.1	822.1	358	26	98	5.3
		16,384	36,621	367.0	110,638	0.3	1,402.8	367	98	95	5.3
32,000	16,000	64	501	5.7	2,077	0.2	28.7	228	2	156	3.7
		256	918	10.3	4,220	0.2	58.6	301	3	119	4.5
		1,024	2,683	27.4	8,610	0.3	122.3	352	7	101	5.0
		4,096	9,790	99.4	16,909	0.6	239.1	364	27	103	5.1
		16,384	38,055	384.1	34,351	1.1	487.4	375	102	99	5.4

for doing volume sampling in RCDVS.

The results for the bigger dimensions are shown in Table 4. Somewhat surprisingly, the problems with $m < n$ are now even more difficult for SDNA than those with $m > n$. Otherwise, the overall picture is the same as previously.

6.3. Logistic regression. Now we consider the l^2-regularized logistic regression function from Example 5.6, which is very popular in the context of machine learning. The termination criterion for each method is the same as before, with the difference that now the optimal objective value is unknown and we have to calculate it numerically in advance. Nevertheless, this auxiliary computation is needed only for our presentation and does not affect the actual performance of the methods in any way.

We set $\gamma := 1$ since this is a default value of the regularization parameter used in practice. However, instead of generating the data $a_1, \dots, a_m$ and $b_1, \dots, b_m$ artificially as we did before, now we take some real-world data from the LIBSVM website,[4] which is summarized in Table 5. Here m is the number of observations and n is the number of features. The next four columns display the four largest eigenvalues of the matrix $B := \frac{1}{4} \sum_{i=1}^{m} a_i a_i^T + \gamma I$ (see Example 5.6), while the last three columns show the theoretical acceleration ratio (6.1) for the three corresponding values of τ. The main reason why we are presenting this table is to demonstrate that it is not uncommon for real data to have significant spectral gaps between the first largest eigenvalues (although these gaps are not as big as in our previous experiments with artificial data).

For the results of the experiment, see Table 6, where, in contrast to the previous two experiments, we additionally consider several small values of τ for RCDVS and SDNA. We can see that, on the real data, the method SDNA looks much better than previously and in many cases it outperforms RCD. Nevertheless, RCDVS is still a winner, and its actual acceleration rate is usually even faster than predicted by theory.

---

[4]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html

TABLE 5
Real data for logistic regression.

Data	m	n	Top four eigenvalues				TheoryAcc		
							2	3	4
breast cancer	683	10	891	118	41	35	4.0	6.6	8.6
phishing	11,055	68	1,798	109	102	95	2.7	3.1	3.4
a9a	32,561	123	51,184	7,502	4,745	3,695	1.8	2.1	2.3

TABLE 6
Results for logistic regression.

Parameters		RCD		SDNA			RCDVS			
Data	τ	It	T	It	Acc	T	It	Acc	%	T
breast cancer	2	1.8	0.10	0.6	3.2	0.03	0.4	4	101	0.02
	3			0.3	5.4	0.02	0.3	6	96	0.02
	4			0.2	7.2	0.02	0.1	12	148	0.01
phishing	2	11.0	7.12	4.8	2.3	3.74	3.3	3	120	2.80
	3			2.7	4.0	2.36	2.1	5	169	1.69
	4			1.8	6.3	1.93	1.6	6	202	2.26
a9a	2	290.9	949.40	444.1	0.7	1,583	175.6	1	90	613.44
	3			273.3	1.1	1,161	73.8	3	189	309.02
	4			171.8	1.7	839	43.0	6	296	212.13

7. Conclusion. We have analyzed the randomized coordinate descent method with volume sampling (RCDVS) for minimizing a function, which is smooth with respect to some positive semidefinite matrix B. In its iterations, this method randomly selects τ-element subsets of coordinates with probabilities proportional to the determinants of principal submatrices of B. We have shown, both theoretically and empirically, that the increase in τ from τ_1 to τ_2 improves the convergence rate up to the factor (1.1), which depends on the spectral gap between the τ_1st and τ_2nd eigenvalues of B.

However, there are still many important directions for further research:
- *Accelerated method.* In addition to the basic randomized coordinate descent method, there also exists the *accelerated* one [10, 11], where the coordinates are sampled with probabilities proportional to the *square roots* of the diagonal elements of B. Is it possible to accelerate RCDVS in a similar manner, possibly using the square roots of the determinants as probabilities?
- *Constrained and composite optimization.* We have considered only the basic smooth *unconstrained* minimization. However, most optimization methods can often be extended to handling problems involving some *simple constraints* (e.g., box constraints or linear ones), or they can also be extended to working with *composite functions* (when the objective is the sum of a smooth function and a simple convex, possibly nonsmooth function), while still retaining the original convergence rate. Can we generalize RCDVS to these settings?[5]
- *Special volume sampling algorithms or a different kind of sampling.* Although the results that we have obtained are true for *any* value of τ, from the *practical* point of view, currently there is only one choice that is suitable for large scale problems, namely $\tau = 2$ (apart from previously known $\tau = 1$). The problem is that currently there are no algorithms for volume sampling whose preprocessing/sampling complexity is appropriate for large-scale applications

---

[5]Currently, the main problem here is that, to our knowledge, there are no corresponding results even for the case $\tau = 1$, that is, for the RCD method.

(e.g., $O(n^2)$ instead of $O(n^3)$). However, this does not mean that it is not possible to devise such algorithms, especially when the matrix possesses special *structure* (e.g., sparse, banded, low-rank, etc.). Another interesting question is whether volume sampling can be replaced with some other kind of sampling which is more practical but still gives similar results.

Appendix A. Proof of Theorem 3.9.

Proof. Note that Argmin f is nonempty and compact by the Weierstrass theorem (f is continuous as a convex function with an open domain, and the sublevel set $L_f(x_0)$ is bounded by the statement and is closed as the inverse image of a closed set under a continuous mapping). Hence, both min and max in the definition of D_τ are attained and, in particular, D_τ is finite.

Using Lemma 3.7, we obtain $x_k \in L_f(x_0)$ and

$$(\text{A.1}) \qquad \mathbb{E}f(x_k) - \mathbb{E}f(x_{k+1}) \geq \frac{1}{2}\mathbb{E}\|\nabla f(x_k)\|^2_{(B_\tau)^{-1}}$$

for all $k \geq 0$.

Let $k \geq 0$. By the convexity of f and the Cauchy–Schwarz inequality, we have

$$f(x_k) - \min f \leq \langle \nabla f(x_k), x_k - x^* \rangle \leq \|\nabla f(x_k)\|_{(B_\tau)^{-1}}\|x_k - x^*\|_{B_\tau}.$$

Hence, by the definition of D_τ, it follows that

$$f(x_k) - \min f \leq \|\nabla f(x_k)\|_{(B_\tau)^{-1}}D_\tau,$$

from which, by Jensen's inequality, we conclude that

$$(\text{A.2}) \qquad \mathbb{E}\|\nabla f(x_k)\|^2_{(B_\tau)^{-1}} \geq \frac{\mathbb{E}(f(x_k) - \min f)^2}{D_\tau^2} \geq \frac{(\mathbb{E}f(x_k) - \min f)^2}{D_\tau^2}.$$

Combining (A.1) and (A.2) and writing $\delta_k := \mathbb{E}f(x_k) - \min f$, we finally obtain

$$(\text{A.3}) \qquad \delta_k - \delta_{k+1} \geq \frac{\delta_k^2}{2D_\tau^2}$$

for all $k \geq 0$. Now the claim follows by a standard argument. Indeed, we can assume without loss of generality that δ_k is strictly positive for each $k \geq 0$. Then, using (A.3) together with the monotonicity of δ_k, we obtain

$$\frac{1}{\delta_{k+1}} - \frac{1}{\delta_k} = \frac{\delta_k - \delta_{k+1}}{\delta_k \delta_{k+1}} \geq \frac{1}{2D_\tau^2}$$

for all $k \geq 0$. By induction, it follows that

$$\frac{1}{\delta_k} \geq \frac{1}{\delta_0} + \frac{k}{2D_\tau^2} \geq \frac{k+1}{2D_\tau^2}$$

for all $k \geq 0$, where the last inequality is a consequence of (A.3) and the positivity of δ_1. $\qquad \square$

Appendix B. Proof of Theorem 3.10.

Proof. Let $0 \leq k \leq K$. By the same argument as in the proof of Theorem 3.9,

$$\text{(B.1)} \qquad \mathbb{E}f(x_k) - \mathbb{E}f(x_{k+1}) \geq \frac{1}{2}\mathbb{E}\|\nabla f(x_k)\|^2_{(B_\tau)^{-1}}.$$

At the same time, by the strong convexity of f in the norm $\|\cdot\|_{B_\tau}$, we have[6]

$$\text{(B.2)} \qquad \frac{1}{2\mu_\tau}\|\nabla f(x_k)\|^2_{(B_\tau)^{-1}} \geq f(x_k) - \min f.$$

Combining (B.1) and (B.2), we obtain that

$$\mathbb{E}f(x_k) - \mathbb{E}f(x_{k+1}) \geq \mu_\tau(\mathbb{E}f(x_k) - \min f)$$

or, equivalently, after rearranging, that

$$\mathbb{E}f(x_{k+1}) \leq \mathbb{E}f(x_k) - \mu_\tau(\mathbb{E}f(x_k) - \min f).$$

Thus, subtracting $\min f$ from both sides, we get

$$\mathbb{E}f(x_{k+1}) - \min f \leq (1 - \mu_\tau)(\mathbb{E}f(x_k) - \min f).$$

The claim now follows by induction. □

Acknowledgment. The authors are thankful to the anonymous reviewers for their attentive reading and valuable comments.

REFERENCES

[1] P. RICHTÁRIK AND M. TAKÁC, *Parallel coordinate descent methods for big data optimization*, Math. Program., 156 (2016), pp. 433–484.

[2] I. NECOARA, Y. NESTEROV, AND F. GLINEUR, *Random block coordinate descent methods for linearly constrained optimization over networks*, J. Optim. Theory Appl., 173 (2017), pp. 227–254.

[3] O. FERCOQ AND P. RICHTÁRIK, *Optimization in high dimensions via accelerated, parallel, and proximal coordinate descent*, SIAM Rev., 58 (2016), pp. 739–771, https://doi.org/10.1137/16M1085905.

[4] I. NECOARA AND D. CLIPICI, *Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds*, SIAM J. Optim., 26 (2016), pp. 197–226, https://doi.org/10.1137/130950288.

[5] J. BRADLEY, A. KYROLA, D. BICKSON, AND C. GUESTRIN, *Parallel coordinate descent for l1-regularized loss minimization*, in Proceedings of the 28th International Conference on International Conference on Machine Learning (ICML'11), Omnipress, Madison, WI, 2011, pp. 321–328.

[6] Z. PENG, M. YAN, AND W. YIN, *Parallel and distributed sparse optimization*, in Proceedings of the 2013 IEEE Asilomar Conference on Signals, Systems and Computers, 2013, pp. 659–664.

[7] Y. NESTEROV, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM J. Optim., 22 (2012), pp. 341–362, https://doi.org/10.1137/100802001.

[8] A. BECK AND L. TETRUASHVILI, *On the convergence of block coordinate descent type methods*, SIAM J. Optim., 23 (2013), pp. 2037–2060, https://doi.org/10.1137/120887679.

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

---

[6]This is a standard inequality for strongly convex functions and can be easily proved from the definition (3.9) by minimizing both sides in $y \in \mathbb{R}^n$.

[10] Y. NESTEROV AND S. U. STICH, *Efficiency of the accelerated coordinate descent method on structured optimization problems*, SIAM J. Optim., 27 (2017), pp. 110–123, https://doi.org/10.1137/16M1060182.

[11] Z. ALLEN-ZHU, Z. QU, P. RICHTÁRIK, AND Y. YUAN, *Even faster accelerated coordinate descent using non-uniform sampling*, in Proceedings of the International Conference on Machine Learning, 2016, pp. 1110–1119.

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

[13] S. SHALEV-SHWARTZ AND T. ZHANG, *Stochastic dual coordinate ascent methods for regularized loss minimization*, J. Mach. Learn. Res., 14 (2013), pp. 567–599.

[14] 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, https://doi.org/10.1137/141000270.

[15] S. SHALEV-SHWARTZ, *SDCA without duality, regularization, and individual convexity*, in Proceedings of the International Conference on Machine Learning, 2016, pp. 747–754.

[16] Z. QU, P. RICHTÁRIK, M. TAKÁC, AND O. FERCOQ, *SDNA: Stochastic dual Newton ascent for empirical risk minimization*, in Proceedings of the International Conference on Machine Learning, 2016, pp. 1823–1832.

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

[18] D. KOVALEV, P. RICHTÁRIK, E. GORBUNOV, AND E. GASANOV, *Stochastic spectral and conjugate descent methods*, in Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2018, pp. 3362–3371.

[19] A. DESHPANDE, L. RADEMACHER, S. VEMPALA, AND G. WANG, *Matrix approximation and projective clustering via volume sampling*, in Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SIAM, Philadelphia, 2006, pp. 1117–1126.

[20] A. DESHPANDE AND L. RADEMACHER, *Efficient volume sampling for row/column subset selection*, in Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS), 2010, pp. 329–338.

[21] C. LI, S. JEGELKA, AND S. SRA, *Polynomial time algorithms for dual volume sampling*, in Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2017, pp. 5038–5047.

[22] M. DEREZINSKI AND M. WARMUTH, *Reverse iterative volume sampling for linear regression*, J. Mach. Learn. Res., 19 (2018), pp. 853–891.

[23] M. DEREZINSKI, M. WARMUTH, AND D. HSU, *Leveraged volume sampling for linear regression*, in Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, 2018, pp. 2510–2519.

[24] C. D. MEYER, *Matrix Analysis and Applied Linear Algebra*, SIAM, Philadelphia, 2000.

[25] H. KARIMI, J. NUTINI, AND M. SCHMIDT, *Linear convergence of gradient and proximal-gradient methods under the Polyak-Łojasiewicz condition*, in Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, Cham, 2016, pp. 795–811.

[26] Y. NESTEROV, *Smooth minimization of non-smooth functions*, Math. Program., 103 (2005), pp. 127–152.

[27] A. BECK AND M. TEBOULLE, *Smoothing and first order methods: A unified framework*, SIAM J. Optim., 22 (2012), pp. 557–580, https://doi.org/10.1137/100818327.