Journal of Computational Mathematics
Vol.36, No.3, 2018, 404–425.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1708-m2017-0113

PARALLEL STOCHASTIC NEWTON METHOD*

Mojmír Mutný
Department of Informatics, ETH Zürich, Zürich, Switzerland
Email: mojmir.mutny@inf.ethz.ch
Peter Richtárik
School of Mathematics, University of Edinburgh, Edinburgh ETH9 3FD, UK and
Computer, Electrical and Mathematical Sciences & Engineering Department, KAUST, Saudi Arabia
Email: peter.richtarik@ed.ac.uk, peter.richtarik@kaust.edu.sa

Abstract

We propose a parallel stochastic Newton method (PSN) for minimizing unconstrained smooth convex functions. We analyze the method in the strongly convex case, and give conditions under which acceleration can be expected when compared to its serial counterpart. We show how PSN can be applied to the large quadratic function minimization in general, and empirical risk minimization problems. We demonstrate the practical efficiency of the method through numerical experiments and models of simple matrix classes.

Mathematics subject classification: 65K05, 65Y05, 68W10, 68W20.
Key words: optimization, parallel methods, Newton's method, stochastic algorithms.

1. Introduction

This work presents a novel parallel algorithm for minimizing a strongly convex function without constraints. This work is motivated by the possibility of better leveraging the structure in surrogate approximation, and the need for efficient optimization methods of high dimensional functions. The age of "Big Data" demands efficient algorithms to solve optimization problems that arise, for example, in fitting of large statistical models or large systems of equations. These new demands define open questions in algorithm design that make previously efficient algorithms obsolete.

For example, in this context, classical second order methods such as Newton method are not applicable as the inversion step of the algorithm is too costly ($O(n^3)$) to be performed in big data settings. Due to this reason, first-order algorithms enjoy huge popularity in the field of practicing optimizers, mainly in the field of machine learning. Recent years have shown that randomization and use of second-order information can lead to better convergence properties of algorithms. A prime example of this utilization are coordinate methods; to mention a few: [3, 14, 18, 20]. Another school, more traditionally grouped under term second-order, has seen a plethora of algorithms in recent year with modified LBFGS [6,8] methods to sub-sampled Newton methods [2, 11, 21–23], which coincide with the direction of this work.

In the current trend, computations are increasingly becoming parallelized, and the increase in performance is usually achieved by including more computing units solving a problem in parallel. Such architectures demand an efficient design of parallel algorithms that are able to exploit the parallel nature of computing clusters. An effort has been undertaken to provide

* Received May 3, 2017 / Revised version received July 4, 2017 / Accepted August 7, 2017 / Published online March 28, 2018 /

theoretical certificates on convergence of parallel optimization algorithms, to name a few, [19, 20], or from class of stochastic methods [16, 21, 29].

We chose to extend an existing algorithm that utilizes curvature information, called SDNA [13], which improves on standard coordinate methods such as SDCA [24] (of which parallel versions exist [20], [17]), and present theoretical certificates on parallelization efficiency of this algorithm along with analysis of special matrix classes. These analyses hint to better theoretical and practical than parallel coordinate descent method (PCDM) [20].

We apply the algorithm to general quadratic problems that arise in finite differences, and further we focus on big data application in machine learning, namely, Empirical Risk Minimization (ERM)

$$\min_{w\in\mathbb{R}^d}\left[P(w)=\frac{1}{n}\sum_{i=1}^{n}\phi_i(a_i^\top w)+\lambda g(w)\right], \tag{1.1}$$

which fits many of the statistical estimation models such as Ridge Regression. We present modified PSN for this type of problems that track dual and primal variables and gives ability to asses duality gap.

1.1. Contributions

The main contribution of this paper is the *design of a novel parallel algorithm* and its subsequent *novel theoretical analysis*. In the case of a smooth objective function, we present convergence analysis with proofs. The method in its simple serial case reduces to variants of algorithms introduced in [13] or [24]. For a different parallelization strategy in the case of convex quadratic optimization, see [21].

We identify parameters of the problem that determine its parallelizability and analyze them in special cases. To do this, we generalize two classes of quadratic optimization problems parametrized by one parameter and analytically calculate the convergence rates for them.

This work utilizes the research on sampling analyzed in paper [12], and is contrasted mainly with another parallel algorithm - parallel coordinate method (PCDM) analyzed in [18]. Furthermore, it generalizes further the class of coordinate methods beyond the generalization of blocks. In this work, the sampled blocks of the over-approximation are not fixed and can overlap. The choice of sampling leading to non-overlapping and fixed blocks has been analyzed previously in [4, 9] and mainly in [17].

1.2. Notation

Vectors. In this work, we use the convention that vectors in $\mathbb{R}^n$ are labeled with lowercase Latin letters. By $e_1, e_2, \dots e_n$ we denote the standard basis vectors in $\mathbb{R}^n$. The ith element of a vector $x \in \mathbb{R}^n$ therefore is $x_i = e_i^\top x$. The standard Euclidean inner product between vectors in $\mathbb{R}^n$ is given by $\langle x, y\rangle := x^\top y = \sum_{i=1}^n x_i y_i$.

Matrices. We use the convention that matrices in $\mathbb{R}^{n\times n}$ are labeled with uppercase bold Latin letters. By $\mathbf{I}$ we denote the identity matrix in $\mathbb{R}^{n\times n}$. The diagonal matrix with vector $w \in \mathbb{R}^n$ on the diagonal is denoted by $\mathbf{D}(w)$. We write $\mathbf{M} \succeq 0$ (resp. $\mathbf{M} \succ 0$) to indicate that $\mathbf{M}$ is symmetric positive semi-definite (resp. symmetric positive definite). Elements of a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ are denoted in the natural way: $\mathbf{A}_{ij} := e_i^\top \mathbf{A} e_j$.

Sampling a Matrix. Let S be a non-empty subset of $[n] := \{1, 2, \ldots, n\}$. We let $\mathbf{I}_{:S}$ be the $n \times |S|$ matrix composed of columns $i \in S$ of the $n \times n$ identity matrix $\mathbf{I}$. Note that $\mathbf{I}_{:S}^\top \mathbf{I}_{:S}$ is the $|S| \times |S|$ identity matrix. Given an invertible matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$, we can extract its principal $|S| \times |S|$ sub-matrix corresponding to the rows and columns indexed by S by

$$\mathbf{M}_{SS} := \mathbf{I}_{:S}^\top \mathbf{M} \mathbf{I}_{:S}. \tag{1.2}$$

It will be also convenient to define

$$\mathbf{M}_S := \mathbf{I}_{:S} \mathbf{M}_{SS} \mathbf{I}_{:S}^\top \tag{1.3}$$

and

$$(\mathbf{M}_S)^{-1} := \mathbf{I}_{:S} (\mathbf{M}_{SS})^{-1} \mathbf{I}_{:S}^\top \tag{1.4}$$

as we shall use these matrices often. Notice that $\mathbf{M}_S$ is the $n \times n$ matrix obtained from $\mathbf{M}$ by retaining elements $\mathbf{M}_{ij}$ for $i \in S$ and $j \in S$; and all the other elements set to zero. On the other hand, $(\mathbf{M}_S)^{-1}$ is obtained from $\mathbf{M}$ by zeroing out the elements corresponding to $i, j \notin S$ and inverting, "in place", the $|S| \times |S|$ matrix composed of elements $i, j \in S$.

Additionally, for any vector $h \in \mathbb{R}^n$ and $\emptyset \neq S \subseteq [n]$ we define $h_S \in \mathbb{R}^n$ by

$$h_S := \mathbf{I}_{:S} \mathbf{I}_{S:} h = \sum_{i \in S} h_i e_i. \tag{1.5}$$

That is, h_S is obtained from h by zeroing out elements $i \notin S$.

1.3. Randomly sampled sub-matrices

In this section we introduce some basic notation which will be needed throughout the paper, following the convention established in [12].

A *sampling*, denoted $\hat{S}$, is a random set-valued mapping with values being subsets of $[n] := \{1, \ldots, n\}$. With each sampling we associate a *probability matrix*, $\mathbf{P} = \mathbf{P}(\hat{S})$, defined via

$$\mathbf{P}_{ij} := \mathbb{P}(i \in \hat{S} \text{ and } j \in \hat{S}), \qquad i, j \in [n]. \tag{1.6}$$

We drop the index $\hat{S}$ if it is clear from the context what sampling is being considered. Further, let

$$p_i := \mathbf{P}_{ii} = \mathbb{P}(i \in \hat{S}), \qquad i \in [n]. \tag{1.7}$$

A sampling $\hat{S}$ for which $p_i > 0$ for all $i \in [n]$ is called *proper*. It is easy to see that the probability matrix $\mathbf{P}(\hat{S})$ does not uniquely determine the underlying sampling $\hat{S}$. For any matrix $\mathbf{M} \in \mathbb{R}^{n\times n}$, by $\mathbf{M}_{\hat{S}}$ we denote the random variable that selects out the elements of $\mathbf{M}$ according to $\hat{S}$, as defined in (1.3).

2. Main Assumptions and Random Matrix Sampling

We start this section with three assumptions that concern our objective function.

2.1. Assumptions

In the following lines we present three main assumptions on the problem structure and machinery at hand used to solve it.

Assumption 2.1 (Smoothness) *There exists a symmetric positive definite matrix* $\mathbf{M} \in \mathbb{R}^{n\times n}$ *such that* $\forall x, h \in \mathbb{R}^n$,

$$f(x+h) \leq f(x) + \langle \nabla f(x), h\rangle + \frac{1}{2}\langle h, \mathbf{M}h\rangle . \tag{2.1}$$

Assumption 2.2 (Strong Convexity) *There exists a symmetric positive definite matrix* $\mathbf{G} \in \mathbb{R}^{n\times n}$ *such that* $\forall x, h \in \mathbb{R}^n$,

$$f(x) + \langle \nabla f(x), h\rangle + \frac{1}{2}\langle h, \mathbf{G}h\rangle \leq f(x+h). \tag{2.2}$$

Minimizing (2.2) on both sides in h gives

$$f(x) - f(x^*) \leq \frac{1}{2}\langle \nabla f(x), \mathbf{G}^{-1}\nabla f(x)\rangle , \tag{2.3}$$

where x^* denotes the (necessarily unique) minimizer of f. Also note that clearly

$$\mathbf{G} \preceq \mathbf{M}, \tag{2.4}$$

with equality if and only if f is a quadratic.

2.2. Samplings

We begin by defining samplings used in this work and by exposing the differences among them. Previous papers such as [12, 14, 18] focused on arbitrary samplings. In this work we will focus on subset of possible *proper* samplings, which can be easily implemented in practice. However, for the sake of completeness, we will define other samplings as well.

Definition 2.1.

1. *τ-nice sampling picks subsets of $[n]$ with cardinality τ, uniformly at random.*
2. *τ-list sampling picks subsets of $[n]$ with cardinality τ, uniformly at random with constraint that subsets have to contain successive elements modulo n.*
3. *Parallel (τ,c)-nice sampling performs c independent τ-nice samplings with replacement. The independent sets are denoted $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_c$.*
4. *Parallel (τ,c)-list sampling performs c independent τ-list samplings with replacement. The independent sets are denoted $\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_c$.*
5. *Parallel (τ, c)-non-overlapping sampling performs one τc-nice sampling on a master node. Subsequently, the set of size $c\tau$ is partitioned to c sets and distributed to worker nodes.*

Remark 2.1. The difference between parallel (c, τ)-nice sampling and standard $c\tau$-nice sampling can be visualized by looking at what part of matrix influences a single iteration. For example, suppose that $n = 5$, $c = 2$ and $\tau = 2$.

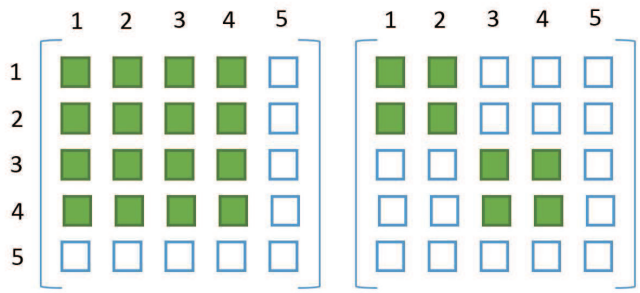

Fig. 2.1. On the left side we see a sampling of a matrix of size 5×5 with τ-nice sampling ($\tau = 4$). Similarly, on the right we see sampling with (c, τ)-nice parallel sampling with $\tau = 2$ and $c = 2$. Namely, here the two sets sampled are $\{\{1, 2\}, \{3, 4\}\}$. We can see that the serial sampling accesses more information in a single iteration even when the number of coordinates updated at each iteration is the same.

Both samplings sample $c\tau$ coordinates; in this case 4. Let the coordinates sampled with $c\tau$-nice sampling be $\{1, 2, 3, 4\}$. Further, the parallel sampling samples two sets at random (let this be $\{1, 2\}$ and $\{3, 4\}$ for this discussion) and distributes it to the worker nodes, whereas with serial τ-nice all four coordinates are handled by the single master node.

In Figure 2.1 one can see that with τ-nice more information is used, however a bigger matrix has to be inverted leading to greater computational costs.

Assumption 2.3 (Independence of Serial Samplings) *The random sets $\{S_k\}_{k\geq 0}$ are:*

1. *independent and identically distributed,*
2. *proper, and*
3. *non-vacuous (i.e., $\mathbb{P}(S_k = \emptyset) = 0$).*

Assumption 2.4 (Independence of Parallel Samplings) *For the random sets of sets $\{\{S_1^k, S_2^k, \ldots, S_c^k\}\}_{k\geq 0}$ holds that:*

1. *The sets are independent and identically distributed.*
2. *S_i^k are identically and independently distributed for all i*
3. *S_i^k is proper for all i and k*
4. *S_i^k is non-vacuous for all i and k; i.e. $\mathbb{P}(S_i^k = \emptyset) = 0$.*

3. The Algorithm

In this section we first briefly review the (serial) stochastic Newton method in Section 3.1, then describe our parallel extension in Section 3.2, and finally formulate the main iteration complexity result in Section 3.3.

3.1. Serial algorithm for smooth functions

The stochastic Newton method (SN) method was first proposed and analyzed in [13]. The method was introduced to solve problem in $\min_{x\in\mathbb{R}^n} f(x)$, where f is smooth and strongly

convex. It is an iterative method given by

$$x^{k+1} = x^k + h^k, \tag{3.1}$$

where

$$h^k = -(\mathbf{M}_{S_k})^{-1}\nabla f(x^k). \tag{3.2}$$

Note that if $\mathbf{M} = \mathbf{G}$, then f is a convex quadratic with Hessian $\mathbf{M}$, and the above method in each iteration performs a Newton step in a random subspace of $\mathbb{R}^n$ generated by the random set S_k. In particular, this is the random subspace spanned by the unit basis vectors e_i where $i \in S_k$. This is done in order to avoid working directly in the n dimension space $\mathbb{R}^n$ as it is assumed that n is very large.

An alternative to the above would be to aim at the full Newton step

$$h^k = -\mathbf{M}^{-1}\nabla f(x^k),$$

but to only apply a few steps of some iterative method to approximate it. This class of methods is known as *truncated Newton methods.* However, this approach does not avoid the dimension n: even when a few steps of some iterative method are taken, such a method still works directly in $\mathbb{R}^n$.

It has been shown that the stochastic Newton method (3.1)-(3.2) converges to optimum given that smoothness and strong convexity assumptions hold. In particular, the following iteration complexity theorem holds.

Theorem 3.1 (Qu, Richtárik, Takáč and Fercoq [13]) *Let Assumptions* 2.1*,* 2.2 *and* 2.4 *be satisfied. Let* $\{x^k\}_{k\geq 0}$ *be a sequence of random vectors produced by the Serial Method and let* x^* *be optimum of function* f*. Then,*

$$\mathbb{E}\left[f(x^{k+1}) - f(x^*)\right] \leq (1-\sigma_1)\mathbb{E}\left[f(x^k) - f(x^*)\right], \tag{3.3}$$

where

$$\sigma_1 := \lambda_{\min}\left(\mathbf{G}^{1/2}\mathbb{E}\left[\left(\mathbf{M}_{\hat{S}}\right)^{-1}\right]\mathbf{G}^{1/2}\right). \tag{3.4}$$

The complexity of the algorithm depends on a parameter denoted σ_1, which is a complicated scalar parameter loosely related to the condition number of a matrix $\mathbf{M}$. More intuitive analysis of this parameter is presented in [13], and for special classes of problems, further in this work.

3.2. Parallel formulation for smooth functions

In this section we introduce a parallel extension of the serial method. We follow the same general iterative scheme defined by (3.1), and thus our method differs only by definition of the update rule which we define as

$$h^k = -\frac{1}{b}\sum_{i=1}^{c}\left(\mathbf{M}_{S_i^k}\right)^{-1}\nabla f(x^k). \tag{3.5}$$

The index k is the iteration counter and the index i labels the subsets of $[n]$, S_i^k, at each iteration. The method depends on the parameter b, which we shall comment on in the next sections on convergence analysis. The parameter b cannot admit any values to guarantee convergence. We show in the next section that, as long as b is bigger than some threshold value b^*, the parallel method is, in a certain precise sense, superior to the serial method.

Algorithm 3.1 PSN: Parallel Stochastic Newton Method

Parameters: sampling $\hat{S}$; data matrix $\mathbf{M}$; aggregation parameter b
Initialization: Pick $x^0 \in \mathbb{R}^n$
for $k = 0, 1, 2, \ldots$ **do**
  for $i = 1, \ldots, c$ in parallel **do**
    Independently generate a random set $\hat{S}_i^k \sim \hat{S}$
    $h_i^k \leftarrow \left(\mathbf{M}_{\hat{S}_i^k}\right)^{-1} \nabla f(x^k)$
  end for
  $x^{k+1} \leftarrow x^k - \frac{1}{b}\sum_{i=1}^c h_i^k$
end for

3.3. Main convergence result

We are now ready to present the main theoretical result of this paper.

Theorem 3.2. *Let Assumptions* 2.1, 2.2 *and* 2.4 *be satisfied. Assume that the aggregation parameter b satisfies*

$$b \geq (c-1)\lambda\theta + 1, \tag{3.6}$$

where

$$\lambda := \lambda_{\max}\left(\mathbf{G}^{-1/2}\mathbf{M}\mathbf{G}^{-1/2}\right), \tag{3.7}$$

$$\theta := \lambda_{\max}\left(\mathbf{G}^{1/2}\mathbb{E}\left[\left(\mathbf{M}_{\hat{S}}\right)^{-1}\right]\mathbf{G}^{1/2}\right). \tag{3.8}$$

Then the random iterates $\{x^k\}$ produced by the PSN method (Algorithm 3.1*) satisfy*

$$\mathbb{E}\left[f(x^{k+1}) - f(x^*)\right] \leq (1-\sigma_p)\mathbb{E}\left[f(x^k) - f(x^*)\right] \tag{3.9}$$

where

$$\sigma_p := \frac{c\sigma_1}{b} = \frac{c\sigma_1}{1+\theta\lambda(c-1)}. \tag{3.10}$$

When $c = 1$, we recover Theorem 3.1. Also note that when $\mathbf{M} = \mathbf{G}$ (quadratic cost function), then $\theta = 1$, and hence $b \leq c$. Consequently, $\sigma_p \geq \sigma_1$. We now have the following corollary:

Corollary 3.1 (of Theorem 3.1) [1] *Let Assumptions* 2.1, 2.2 *and* 2.3 *be satisfied then for non-overlapping parallel (τ, c)-nice sampling, we achieve a linear speedup in c with respect to the τ-nice sampling.*

Proof. Let $\hat{S}_i$ denote the partitioning of the set of sets $\hat{S}$ where $|\hat{S}_i| = \tau$ according to the definition of the non-overlapping sampling. Due to non-overlapping property, up to a permutation of the rows, the matrix $\mathbf{M}_{\hat{S}}$ is block diagonal. Hence, $(\mathbf{M}_{\hat{S}})^{-1} = (\sum_{i=1}^c \mathbf{M}_{\hat{S}_i})^{-1} = \sum_{i=1}^c (\mathbf{M}_{\hat{S}_i})^{-1}$, and therefore

$$\sigma_p = \lambda_{\min}\left(\mathbf{G}^{1/2}\mathbb{E}\left[\left(\mathbf{M}_{\hat{S}}\right)^{-1}\right]\mathbf{G}^{1/2}\right) = c\lambda_{\min}\left(\mathbf{G}^{1/2}\mathbb{E}\left[\left(\mathbf{M}_{\hat{S}_i}\right)^{-1}\right]\mathbf{G}^{1/2}\right) = c\sigma_1.$$

This completes the proof. □

[1] This fact was pointed to the authors by Sai Praneeth Reddy Karimireddy, for which they are most thankful.

4. Complexity Analysis

In order to develop a sound complexity analysis and prove Theorem 3.2, we need to develop a series of lemmas dealing with expectations of matrix minors and relations of constants defined in (3.8) and (3.10).

4.1. Sampling lemmas

Lemma 4.1 (Tower property for expectation of matrix minors) *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be symmetric, and $x \in \mathbb{R}^n$. If $\hat{S}_1, \hat{S}_2 \sim \hat{S}$ are i.i.d. samplings, then,*

$$\mathbb{E}\left[\langle \mathbf{A}\mathbf{A}_{\hat{S}_1}x, \mathbf{A}_{\hat{S}_2}x\rangle\right] = \langle \mathbf{A}\mathbb{E}[\mathbf{A}_{\hat{S}}]x, \mathbb{E}[\mathbf{A}_{\hat{S}}]x\rangle\,. \tag{4.1}$$

Proof. We write

$$\begin{aligned}\mathbb{E}[\langle \mathbf{A}\mathbf{A}_{\hat{S}_1}x, \mathbf{A}_{\hat{S}_2}x\rangle] &= \mathbb{E}[x^\top \mathbf{A}_{\hat{S}_1}\mathbf{A}\mathbf{A}_{\hat{S}_2}x] = x^\top \mathbb{E}[\mathbf{A}_{\hat{S}_1}\mathbf{A}\mathbf{A}_{\hat{S}_2}]x \\ &= x^\top \mathbb{E}\left[\mathbb{E}\left[\mathbf{A}_{\hat{S}_1}\mathbf{A}\mathbf{A}_{\hat{S}_2} \mid \hat{S}_2\right]\right]x = x^\top \mathbb{E}\left[\mathbb{E}[\mathbf{A}_{\hat{S}_1}]\mathbf{A}\mathbf{A}_{\hat{S}_2}\right]x \\ &= x^\top \mathbb{E}[\mathbf{A}_{\hat{S}}]\mathbf{A}\mathbb{E}[\mathbf{A}_{\hat{S}}]x = \langle \mathbf{A}\mathbb{E}[\mathbf{A}_{\hat{S}}]x, \mathbb{E}[\mathbf{A}_{\hat{S}}]x\rangle\,.\end{aligned}$$

In the second step we use linearity of expectation, combined with linearity of the mapping $\mathbf{X} \to x^\top \mathbf{X}x$. In the third step we use the tower property. In the fourth step, we use linearity of expectation and independence. In the fifth step we use linearity of expectation again, combined with the assumption that $\hat{S}_1$ and $\hat{S}_2$ have the same distribution as $\hat{S}$. □

Lemma 4.2. *Assume that the block matrix*

$$\mathbf{Q} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{pmatrix} \tag{4.2}$$

is positive definite. Then the matrix defined as

$$\mathbf{K} = \mathbf{A}^{-1}\mathbf{B}(\tilde{\mathbf{A}})^{-1}\mathbf{B}^\top \mathbf{A}^{-1}, \tag{4.3}$$

where $\tilde{\mathbf{A}}$ denotes the Shur complement of $\mathbf{A}$, is positive semi-definite.

Proof. By [28] Notion 7.4 we know that Shur complement is positive semi-definite if $\mathbf{C}$ is non-singular. As $\mathbf{A}$ is positive definite, then $\mathbf{C}$ is non-singular. Thus, $\mathbf{K}$ must be positive semi-definite as well.

Lemma 4.3 (Zhang [28]) *Let $\mathbf{M} \succ 0$, and S be a subset of $[n]$. Then*

$$(\mathbf{M}_S)^{-1} \preceq (\mathbf{M}^{-1})_S. \tag{4.4}$$

Lemma 4.4. *Let S' and S be two samplings such that $S' \subset S \subset [n]$. Also let $\mathbf{X} \in \mathbb{R}^{n\times n}$ be a positive definite matrix. Then*

$$(\mathbf{X}_{S'})^{-1} \preceq (\mathbf{X}_S)^{-1}. \tag{4.5}$$

Proof. Letting $\mathbf{Y} = \mathbf{X}_{SS}$, we can write

$$\mathbf{X}_{S'} = \mathbf{I}_{:S'}(\mathbf{I}_{S'S}\mathbf{Y}\mathbf{I}_{SS'})\mathbf{I}_{S':} = \mathbf{Y}_{S'}.$$

By Lemma 4.3 and positive definiteness, we get

$$(\mathbf{X}_{S'})^{-1} = (\mathbf{Y}_{S'})^{-1} \overset{(4.4)}{\preceq} (\mathbf{Y}^{-1})_{S'} \overset{\text{pos. def.}}{\preceq} (\mathbf{Y}^{-1})_S = (\mathbf{X}_S)^{-1}.$$ □

4.2. Parallelization parameters

Lemma 4.5. *The following estimates hold:*

$$0 \leq \sigma_1 \leq \theta \leq 1. \tag{4.6}$$

Proof. The proof of the first inequality follows by noting that σ_1 is the smallest eigenvalue of a positive definite matrix; the second is by the definition of θ and σ_1. The last inequality follows from the convexity of $\lambda_{\max}$ operator and Jensen's inequality, namely,

$$\mathbb{E}[\lambda_{\max}(\mathbf{X})] \geq \lambda_{\max}(\mathbb{E}[\mathbf{X}]). \tag{4.7}$$

Consequently,

$$\begin{aligned}\theta &\overset{(3.8)}{=} \lambda_{\max}(\mathbf{G}^{1/2}\mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]\mathbf{G}^{1/2}) \overset{(4.4)}{\leq} \mathbb{E}[\lambda_{\max}(\mathbf{G}^{1/2}(\mathbf{M}^{-1})_{\hat{S}}\mathbf{G}^{1/2})] \\ &\overset{(4.5)}{\leq} \mathbb{E}[\lambda_{\max}(\mathbf{G}^{1/2}(\mathbf{M}^{-1})\mathbf{G}^{1/2})] = \mathbb{E}\left[\frac{1}{\lambda_{\min}(\mathbf{G}^{-1/2}(\mathbf{M})\mathbf{G}^{-1/2})}\right] \overset{(2.4)}{\leq} 1\end{aligned} \tag{4.8}$$

This completes the proof. □

Proposition 4.1 (Bound on θ for τ-list samplings) *Let $\mathbf{M} = \mathbf{G}$ as in (2.1) and $\hat{S}$ be parallel (τ, c)-list sampling, then*

$$\theta \leq \frac{\tau}{n}\operatorname{cond}(\mathbf{M}). \tag{4.9}$$

Proof. It follows that

$$\begin{aligned}\theta &\overset{(3.8)}{=} \lambda_{max}(\mathbf{M}^{1/2}\mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]\mathbf{M}^{1/2}) \\ &= \max_{\|x\|=1} \langle \mathbf{M}^{1/2}x, \mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]\mathbf{M}^{1/2}x\rangle \\ &= \max_{\|\mathbf{M}^{-1/2}y\|=1} \langle y, \mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]y\rangle \\ &\overset{\text{Def. 4}}{=} \max_{\|M^{-1/2}y\|=1} \frac{1}{n}\sum_{i=1}^{n} \langle y, (\mathbf{M}_{S_i})^{-1}y\rangle \\ &= \max_{\|M^{-1/2}y\|=1} \frac{1}{n}\sum_{i=1}^{n} \langle \mathbf{I}_{S:}y, (\mathbf{M}_{S_iS_i})^{-1}\mathbf{I}_{S:}y\rangle \\ &\overset{(4.5)}{\leq} \max_{\|M^{-1/2}y\|=1} \frac{1}{n}\sum_{i=1}^{n} \lambda_{max}((\mathbf{M})^{-1})\,\|\mathbf{I}_{S:}y\|^2 \\ &= \frac{\tau}{n}\lambda_{max}(\mathbf{M}^{-1}) \max_{\|M^{-1/2}y\|=1} \|y\|^2 \\ &= \frac{\tau}{n}\operatorname{cond}(\mathbf{M}).\end{aligned}$$

This completes the proof. □

Proposition 4.1 can be useful for classes of problems where we have information about the condition number of the matrix $\mathbf{M}$ in Assumption 2.1, and $n \gg \operatorname{cond}(\mathbf{M})$. In these circumstance, the bound can be used as a good proxy for θ. A prime example of such problem class are banded Toeplitz matrices where condition number is usually independent of n but rather depends on the band width.

5. Analysis and Comparison with Existing Methods

5.1. Theoretical comparison PSN with PCDM

Parallel coordinate descend method (PCDM) is a powerful parallel optimization algorithm analyzed in [20] and [17]. We would like to compare the performance of this method to the current parallel method. The complexity of PCDM can be expressed using the language of this paper and the paper [13] by,

$$\sigma_3 = \lambda_{\min}(\mathbf{G}^{1/2}\mathbf{D}(v^{-1})\mathbf{D}(p)\mathbf{G}^{1/2}), \tag{5.1}$$

where v and $p \in \mathbb{R}^d$. Determining v for a given problem with matrix $\mathbf{M} = \mathbf{A}^\top\mathbf{A}$ is subject of [12] and p is defined in (1.7).

To have a fair comparison while using parallel (τ, c)-nice sampling for PSN, we compare the algorithm with (τc)-nice sampling for PCDM such that access the same number of coordinates is maintained. Also, the assumptions of the PCDM algorithm are a little different in what we assumed here. For example, in [20], they assume that matrix $\mathbf{M}$ in the quadratic over-approximation has a decomposition such that $\mathbf{M} = \mathbf{A}^\top\mathbf{A}$. The the condition for v in expression (5.1) can be deduced, for example, from basic considerations where $v_i = \lambda_{\max}(\mathbf{A}^\top\mathbf{A})$ for all i, which defines

$$\sigma_b := \lambda_{\min}(\mathbf{G}^{1/2}\mathbf{D}(p)\mathbf{G}^{1/2})/\lambda_{\max}(\mathbf{A}^\top\mathbf{A}). \tag{5.2}$$

Alternatively, we can use structural sparsity with [12, Proposition 5.1] and can calculate the vector v as follows

$$v_i = \sum_{j=1}^{m}\left(1 + \frac{(|J_j| - 1)(\tau c - 1)}{\max(n-1, 1)}\right)\mathbf{A}_{ji} \tag{5.3}$$

where J_j denotes $J_j := \{i \in [n] \mid \mathbf{A}_{ji} \neq 0\}$. As we do not have the decomposition at disposal currently, let us assume that $\mathbf{A}$ is fully dense, and thus $|J_j| = n \ \forall j$. In this setting, v_i reduces to $v_i = \tau c\mathbf{M}_{ii}$, and $p_i = \frac{\tau c}{n}$ for (τ, c)-nice sampling. Thus we can see that in fact σ_3 does depend on τc at all, and we do not gain any theoretical speedup in convergence rate. This exemplifies the theoretical contribution made in [13] and this paper, which show the improvement in convergence rates without sparsity patterns as σ_1 and σ_p depend on τ or c even in the fully dense scenario. To illustrate this, we calculate the value of σ_p for a random matrix $(n = 400)$ in the Figure 5.1.

Table 5.1: The theoretical convergence rates for quadratic problems with $\mathbf{A}^\top\mathbf{A} = \mathbf{M}$, where $\mathbf{A}$ was 400×400 matrix. Entries of the matrix $\mathbf{A}$ were sampled from a standard normal distribution. The basic approach with the largest eigenvalue σ_b works better only when τc is very big, however given the difficulty of inverting large matrices, this approach is often impractical.

c (cores)	τ	τc	$\sigma_b \times 10^{-6}$	$\sigma_3 \times 10^{-6}$	$\sigma_p \times 10^{-6}$
1	3	3	0.0015	0.0019	0.0058
2	3	6	0.0029	0.0019	0.0113
4	3	12	0.0058	0.0019	0.0214
8	3	24	0.0117	0.0019	0.0386
16	3	48	0.0233	0.0019	0.0646
32	3	96	0.0466	0.0019	0.0975
64	3	192	0.0933	0.0019	0.1308
128	3	384	0.1866	0.0019	0.1578

Table 5.2: The theoretical convergence rates for quadratic problems with $\mathbf{A}^\top\mathbf{A} = \mathbf{M}$ where $\mathbf{A}$ was 400×400 matrix. The matrix $\mathbf{A}$ has also sparse structure where only approximately one third of the elements were non-zero. The entries were sample from a standard normal distribution.

c (cores)	τ	τc	$\sigma_b \times 10^{-6}$	$\sigma_3 \times 10^{-6}$	$\sigma_p \times 10^{-6}$
1	3	3	0.0604	0.1744	0.2679
2	3	6	0.1208	0.2305	0.5199
4	3	12	0.2416	0.2747	0.9818
8	3	24	0.4833	0.3038	1.7664
16	3	48	0.9666	0.3208	2.9418
32	3	96	1.9332	0.3300	4.4086
64	3	192	3.8663	0.3348	5.8727
128	3	384	7.7326	0.3373	7.0420

However, one has to bear in mind that we assumed that the matrix $\mathbf{A}$ was fully dense, which is the worst case scenario for the PCDM algorithm. Unfortunately, as there is not an easy way to present a comprehensive theoretical comparison, we only present one based on generated random matrices which have either dense or sparse structure. In an experiment with sparse matrix (only one third of the elements are present) and using (5.3), we arrived at the results summarized in Table 5.2. These values can be almost directly compared as the cost of inversion of 3×3 matrix is too small to influence the cost of one iteration significantly. The instruction level parallelism can be utilized to greater extent as more computation is done with the same amount of information leading to better operational intensity [27].

5.2. ρ-matrix analysis

In the following analysis, we compute the convergence rates for the serial and parallel algorithm, applied to a specific problem, exactly. We minimize the quadratic function $\frac{1}{2}x^\top\mathbf{M}x$ where $\mathbf{M}$ has a special structure that we call ρ-matrix. The reason for introducing this special type of problem is that for this problem an analytical expression for σ_1 and θ could be found and intuition about behavior of these theoretical constants can be illustrated fully. Since the parallel speedup depends only on parameter θ we can predict the theoretical speedup for this class of matrices easily. The theoretical speedup is presented in Figure 5.1.

This matrix class tries to represent fully dense matrices that occur in machine learning, where speedup can be achieved without the sparsity assumption as done in the previous work. Problems with matrices $\mathbf{M}$ which have fully dense structure occur, for example, in Gaussian process regression where kernel matrices serve the purpose of $\mathbf{M}$. Specifically, similar structure to ρ-matrix occurs when each element is defined via shift invariant covariance functions [15]. A classical example of such covariance function is squared exponential kernel

$$k(x, y) = \exp\left(-\|x - y\|^2 / 2\right), \tag{5.4}$$

where $x, y \in \mathbb{R}^d$. In this case, the kernel matrix (symmetric positive-definite) has has a unit diagonal with off-diagonal elements bounded above by 1.

Definition 5.1. *For $0 < \rho < 1$ we define ρ-matrix to be a square matrix with the following*

structure:

$$\mathbf{M} = \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{pmatrix}.$$

Proposition 5.1. *Suppose we have a ρ-matrix $\mathbf{M}$ with $0 < \rho < 1$ as a parameter. Then σ_1 and θ can be expressed as functions of n and τ for τ-nice sampling as follows*

$$\sigma_1 = (1-\rho)\left(1 - \frac{(\tau-1)B(\tau)}{(n-1)A(\tau)}\right)\frac{\tau A(\tau)}{n}, \tag{5.5}$$

$$\theta = (n\rho - \rho + 1)\left(n\frac{(\tau-1)B(\tau)}{(n-1)A(\tau)} - \frac{(\tau-1)B(\tau)}{(n-1)A(\tau)} + 1\right)\frac{\tau A(\tau)}{n} \tag{5.6}$$

where

$$A(\tau) := \frac{((\tau-2)\rho+1)}{(1-\rho)((\tau-1)\rho+1)}, \tag{5.7}$$

$$B(\tau) := -\frac{\rho}{(1-\rho)((\tau-1)\rho+1)}. \tag{5.8}$$

Proof. There are n eigenvalues of the matrix $\mathbf{M}$; $n-1$ of them are degenerate and equal to $1-\rho$. This can be verified by plugging the suitable eigenvector which has only two non-zero elements $-1, 1$. The last eigenvalue can be computed using the trace of the matrix. $\lambda = n - (1-\rho)(n-1)$. After solving for this eigenvalue, we immediately see it is the biggest one, thus, $\lambda_{\max}(\mathbf{M}) = n\rho - \rho + 1$.

Given a τ-nice sampling with τ as parameter, the matrix that is sub-sampled is a ρ-matrix of the size $\tau \times \tau$. Due to the symmetry of the problem, any subset of τ coordinates slices out the same matrix. Additionally, the cofactor matrices for diagonal elements are all equal, and the cofactor matrices for off-diagonal elements are all equal as well.

As the entries in the inverse of the matrix depend on the determinant of the cofactor matrix only, all diagonal and non-diagonal elements of the inverse are equal. In addition, taking expectation does preserve this structure as the τ-nice sampling is uniform. Thus, let us denote the value of these two entries in diagonal and off-diagonal in the sampled inverted matrix as $A(\tau)$ and $B(\tau)$ respectively.

Therefore, the value of $\mathbb{E}[\mathbf{M}_{\hat{S}}^{-1}] = \frac{\tau}{n}\mathbf{I}A(\tau) + \frac{\tau(\tau-1)}{n(n-1)}(\mathbf{S}-\mathbf{I})B(\tau)$, where $\mathbf{S}$ is matrix full of ones. To determine the values of A and B, we compute inverse for a given τ using Cramer's rule (using matrix minors)

$$\mathbf{M}^{-1} = \frac{1}{\det(\mathbf{M})}\mathbf{C}^\top, \tag{5.9}$$

where $\mathbf{C}$ is matrix of cofactors.

The trick here is that we need to look only at two cofactors due to the structure of the matrix. So for $A(\tau)$ on diagonal we take $\mathbf{C}_{11}$ minor. We adopt notation where we use subscript of $\mathbf{M}$ to denote the size of the sub-matrix of ρ-matrix $\mathbf{M}$. We calculate determinant as a product of the known eigenvalues discussed earlier.

$$\begin{aligned} A(\tau) = (\mathbf{M}_\tau^{-1})_{11} &\overset{(5.9)}{=} \frac{\det(\mathbf{M}_{\tau-1})}{\det(\mathbf{M}_\tau)} \\ &= \frac{(1-\rho)^{(\tau-2)}((\tau-2)\rho+1)}{(1-\rho)^{(\tau-1)}((\tau-1)\rho+1)} = \frac{((\tau-2)\rho+1)}{(1-\rho)((\tau-1)\rho+1)} \end{aligned}$$

When we look at $B(\tau)$ the minor of the matrix that determines any off-diagonal is a matrix which contains only ρ's at first row and continues with other rows as in matrix $\mathbf{M}_\tau$. Let us denote such type of matrix $\mathbf{N}_\tau$, where the subscript τ symbolizes the size of the matrix again. For $\det(\mathbf{N}_\tau)$, we arrive at the following recurrence relation by using Laplace decomposition of determinants.

$$\det(\mathbf{N}_\tau) = (1-\rho)\mathbf{N}_{\tau-1} \tag{5.10}$$

$$\det(\mathbf{N}_2) = \rho^2 - \rho \tag{5.11}$$

Therefore,

$$\begin{aligned} B(\tau) = (\mathbf{M}_\tau^{-1})_{12} &= -\frac{\det(\mathbf{N}_\tau)}{\det(\mathbf{M}_\tau)} \overset{(5.10)}{=} -\frac{(1-\rho)^{(\tau-2)}\rho}{(1-\rho)^{(\tau-1)}((\tau-1)\rho+1)} \\ &= -\frac{\rho}{(1-\rho)((\tau-1)\rho+1)}. \end{aligned} \tag{5.12}$$

The matrix $\mathbb{E}[\mathbf{M}_{\hat{S}}^{-1}]$ can be manipulated to the from of ρ-matrix by dividing by the factor $\frac{\tau A(\tau)}{n}$. Thus, we can define a new ρ_N which serves as parameter of the new ρ-matrix.

This parameter allows us to compute the eigenvalues of the matrix, and thus, also maximal and minimal eigenvalues. From the σ_1 definition, σ_1 is the smallest eigenvalues of a product of two ρ-matrices one with ρ as parameter and one with ρ_N as the parameter. The product of two ρ-matrices is a ρ-matrix again, and as consequence of this, the smallest eigenvalue of the product is just a product of the two smallest eigenvalues ones.

$$\sigma_1 = (1-\rho)(1-\rho_N)\frac{\tau A(\tau)}{n} = (1-\rho)\left(1 - \frac{(\tau-1)B(\tau)}{(n-1)A(\tau)}\right)\frac{\tau A(\tau)}{n}. \tag{5.13}$$

And similarly θ, and its definition imply that θ is product of the two biggest eigenvalues

$$\begin{aligned} \theta &= (n\rho - \rho + 1)(n\rho_N - \rho_N + 1)\frac{\tau A(\tau)}{n} \\ &= (n\rho - \rho + 1)\left(n\frac{(\tau-1)B(\tau)}{(n-1)A(\tau)} - \frac{(\tau-1)B(\tau)}{(n-1)A(\tau)} + 1\right)\frac{\tau A(\tau)}{n}. \end{aligned} \tag{5.14}$$

This completes the proof. □

A good way to use ρ-matrix as an approximation for kernel matrices ($\mathbf{K}_{ij} = k(x^i - x^j)$) defined on a regular grid for purposes of determining θ, is to use the quantity $h := \min_{i\neq j} \left\|x^i - x^j\right\|$ (we call this quantity *spacing*), where $\{x^i\}_{i=1}^n$ defines a grid in $\mathbb{R}^d$. This scenario is by no means artificial, and occurs frequently in function interpolation [26]. An approximation for ρ can be gained via $\rho \approx \exp(-h^2/2)$. A justification for such approximation is provided in Figure 5.2, where the size of the matrix $\mathbf{K}$ is plotted against the approximated $\hat{\theta}$ and the true value θ for τ-nice sampling. The kernel matrix is created using the squared exponential kernel (5.4). It does not hold that in general that $\mathbf{K} \preceq \mathbf{M}$ (in Löwner ordering), where $\mathbf{M}$ is a ρ-matrix, but the approximation seems to be justified at the θ scale empirically.

5.3. α-tridiagonal matrix analysis

There is only a limited number of possible special matrix structures that are simple enough to be parametrized by one parameter, and at the same time function as a useful practical

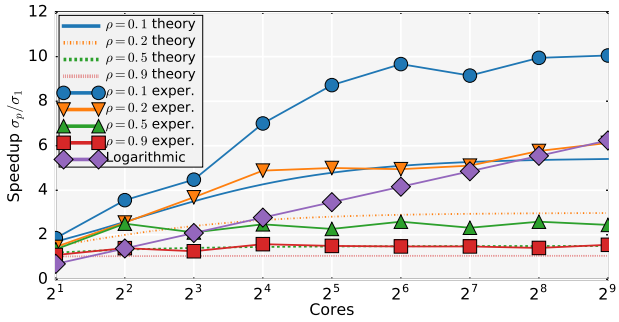

Fig. 5.1. Theoretical speedup for problem of minimizing quadratic function $f(x) = \frac{1}{2}x^\top \mathbf{M}x$ with $\mathbf{M}$ being ρ-matrix. The size of the problem is $n = 1024$, and parallel $(2, c)$-nice sampling. We observe the smaller the ρ, the greater the speedup. Also, we see that our estimate of speedup is rather conservative for small values of ρ. Intuitively, the problem should be more difficult as ρ increases since ρ is related to the condition number of $\mathbf{M}$, namely $\text{cond}(\mathbf{M}) = \frac{n\rho-\rho-1}{1-\rho}$, which is an increasing function of ρ.

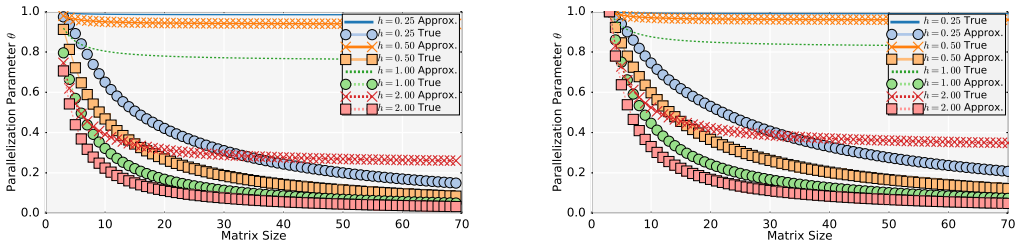

Fig. 5.2. Comparison of true θ and estimated $\hat{\theta}$ for kernel matrix defined via squared exponential kernel, and ρ matrix defined by the *spacing* (LEFT: $\tau = 2$, RIGHT: $\tau = 3$). We notice that in all instances the approximated value is higher than the true value, thus we can use it to bound the parallel speedup and the parameter b.

model. We choose to model tridiagonal Toeplitz matrices as the next model class. As the optimization method depends only on the ratio between eigenvalues we scale the matrix such that the diagonal entries are equal to 1. Matrices with a special structure such as banded, pentadiagonal or even tridiagonal matrices occur frequently in finite difference or finite element schemes, and are not only toy examples.

Definition 5.2 (α-tridiagonal matrix) *Let $\alpha \in [0, \frac{1}{2})$, then $\mathbf{T}(\alpha) \in \mathbb{R}^{n\times n}$*

$$\mathbf{T}(\alpha) = \begin{pmatrix} 1 & \alpha & 0 & 0 & \dots & 0 \\ \alpha & 1 & \alpha & 0 & \dots & 0 \\ 0 & \alpha & 1 & \alpha & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \alpha & 1 \end{pmatrix} \tag{5.15}$$

is a α-tridiagonal positive-definite matrix.

Similarly as in the previous section we minimize a quadratic function $\frac{1}{2}x^\top \mathbf{T}(\alpha)x$ and look at the theoretical speedup that we are able to achieve in Figure 5.3 (RIGHT). Due to the sparse nature of the matrix, we choose τ-list matrix sampling with $\tau = 2$. The simple structure allows for explicit calculation of $\mathbb{E}[((\mathbf{T}(\alpha))_{\hat{S}})^{-1}]$, which results in pentadiagonal matrix of particular form

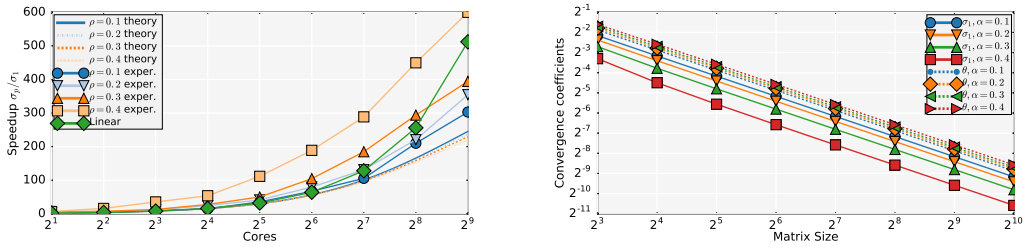

Fig. 5.3. Properties of problems with α-tridiagonal matrices. LEFT: Theoretical and practical speedup. RIGHT: Values of σ_1 and θ.

for which eigenvalues can be efficiently computed algorithmically even for large matrix sizes, or estimated using the following proposition. The dependence, which confirm the Proposition 5.2, is visualized in Figure 5.3 (LEFT).

Proposition 5.2. *Let* $\mathbf{T}(\alpha)$ *be a* $n \times n$ α*-tridiagonal* ($\alpha \in [0, \frac{1}{2}]$) *matrix. Then for* 2*-list parallel or serial sampling the parameter* θ *is bounded from above by,*

$$\theta \leq \frac{2}{(1-\alpha)n}. \tag{5.16}$$

Proof. First we observe that the matrix $\mathbf{T}_t = \mathbb{E}[(\mathbf{T}(\alpha)_{\hat{S}})^{-1}]$ has a special tridiagonal structure with different elements only in two entries. By multiplying together $\mathbf{T}(\alpha)\mathbf{T}_t$ we obtain matrix whose eigenvalue spectrum is the same as the one of $\mathbf{T}(\alpha)^{1/2}\mathbf{T}_t\mathbf{T}(\alpha)^{1/2}$, due to cyclic property of λ_{max}. The rest of the proof applies Gershgorin circle theorem [5], which bounds eigenvalues. The resulting matrix $\mathbf{T}(\alpha)\mathbf{T}_t$ is constant on the diagonal and the inequalities arising from Gershgorin circle theorem are nested, thus, we are left with only one condition in (5.16). □

The Proposition 5.2 reveals that already in the worst case $\alpha = \frac{1}{2}$, we can show parallel speedup for matrices of size $n \geq 5$ even with the crude bound presented. Should, we want to solve large tridiagonal system as $\min_x \frac{1}{2}\|\mathbf{T}(\alpha)x - b\|^2$, we can use Proposition 4.1 with explicit formula for eigenvalues of tridiagonal matrix from [10] to bound the eigenvalues for any n,

$$\lambda_{max}(\mathbf{T}(\alpha)) \leq 1 + 2\alpha \text{ and } \lambda_{min}((\mathbf{T}(\alpha))) \geq 1 - 2\alpha.$$

Thus, the bound for θ becomes $\theta \leq \frac{\tau}{n}\frac{(1+2\alpha)^2}{(1-2\alpha)^2}$, which for sufficiently big n leads to theoretical parallel speedup again.

The utilization of PSN for tridiagonal matrices for small problems is unlikely as there exists a very efficient $O(n)$ algorithm known in literature as Thomas algorithm [25]. However similar approaches, as presented here, can be applied to general banded matrices if they have special structure, and the bound on eigenvalues can be provided by analytical means or via Gershgorin circle theorem.

6. Numerical performance

All experiments were run on Lenovo T450S Intel Core i7 (Broadwell) with 2.6 GHz cores, and the C++ code was compiled with GCC compiler version 5 and directives *-O3 -mfma*.

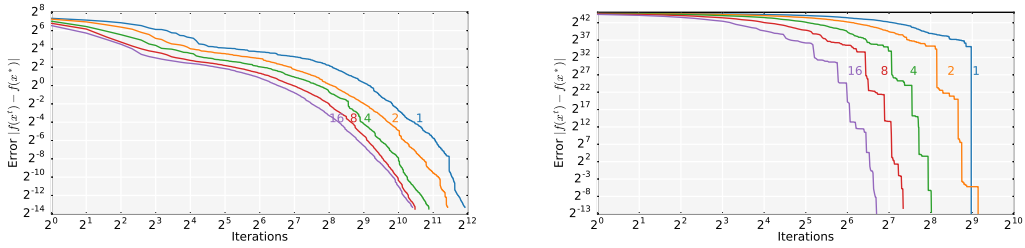

Fig. 6.1. Performance of PSN on *houses* dataset where $n = 1024$ and $m = 64$. LEFT: A ridge regression model with parallel $(2, c)$-nice sampling. The parameter c in the graph represents the number of computational units used. We used $\theta = 0.2$. RIGHT: Gaussian process fitting (corresponds to ridge regression $n = m = 1024$). We used $\theta = 0.01$. We run parallel $(10, c)$-nice sampling due to larger matrix $\mathbf{M}$.

6.1. Artificial data examples

We performed two experiments with artificial data to demonstrate the parallelization speedup of the algorithm in practice. We first compare the serial and the parallel method on minimization of $\frac{1}{2}\|\mathbf{X}x - y\|^2$ with respect to x, where $\mathbf{X}$ is fully dense, and show superior convergence properties in terms of iteration. We report empirical speedup of the methods in Figure 6.3 (LEFT). The θ value for this experiment was handpicked to be 0.7.

Secondly, we compare our parallel method with PCDM. While the PSN was run with specific parallel (τ, c)-nice sampling all PCDM runs were run with τc-nice sampling to ensure a fair comparison. The PSN algorithm was run with a handpicked value of θ equal to 0.7 in the experiments below. The main comparison of quadratic function minimization as in the previous experiment is presented in Figure 6.2 (LEFT). All artificial data (entries) for the dense matrix $\mathbf{X}$ and y were generated by sampling standard normal distribution.

6.2. Machine Learning examples

We conduct three experiments on real datasets to further explore the performance of PSN and contrast it to the serial algorithm. We fit logistic regression, linear ridge regression and Gaussian process regression on real datasets. Firstly, we report moderate improvement on *mushroom* dataset in Figure 6.2 (RIGHT), where logistic regression has been fit.

Secondly, we work with *houses* dataset [7], where the goal is to predict the house price from various information. The dataset is of size $n = 1460$ and $m = 64$ after preprocessing. We apply two machine learning algorithms on the dataset. Firstly, we show optimization error on standard ridge regression (regularization parameter $\lambda = \frac{1}{n}$) and optimize it with PSN in Figure 6.1 (LEFT). We run the algorithm with $\theta = 0.2$, which was hand picked.

Additionally, we try a more advanced model, namely Gaussian process, which boils down to ridge regression with special covariance structure defined via squared exponential kernel as in (5.4). For further details we refer the reader to [15]. The regularization parameter was used as in the previous ridge regression. We note that due to special structure of the problem, the θ used could be much smaller; in our case $\theta = 0.01$. We report the variation with number of computational units in Figure 6.1 (RIGHT).

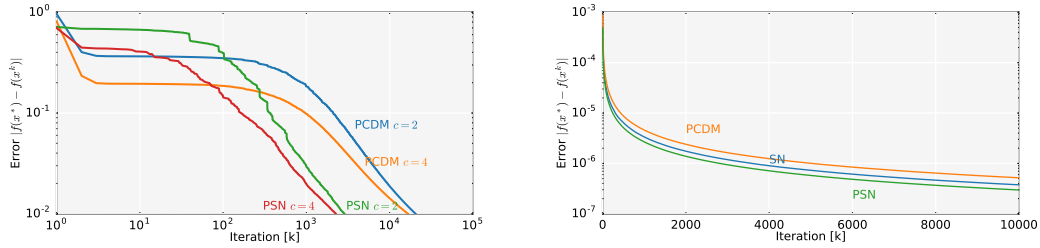

Fig. 6.2. Performance of PSN versus PCDM on linear and logistic regression problems. LEFT: A linear regression model with an artificial dataset such that $n = m = 10^3$, and $(3, c)$-nice sampling. The parameter c in the graph represents the number of computational units. RIGHT: A logistic regression model based on *mushroom* dataset with $n = 8124$, $m = 22$ and $(2, c)$-nice sampling. We run PSN with 1 (SN) and 4 cores, and PCDM with τc-nice sampling with the constant $\tau c = 8$. We scaled the eigenvalues of the problem before optimization.

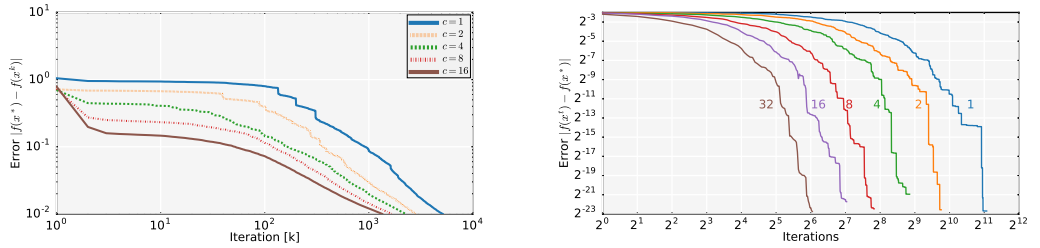

Fig. 6.3. Comparison of a serial and a parallel stochastic Newton method. We run the parallel method with a different number of computational units. LEFT: In all cases, we fit a linear regression with $n = m = 10^3$ examples and variables. We utilize the $(3, c)$-nice sampling. RIGHT: The number of iterations required to solve the linear system ($n = 10^3$) from the implicit finite difference scheme to the accuracy 10^{-8}. The labels in the graph represents the number of computational units.

6.3. Finite Differences example

In addition, we perform a numerical experiment that arises in numerical analysis when solving the heat equation with finite differences using a high-order scheme. Namely, we simulate $\partial_t u(x,t) = \partial_{xx} u(x,t)$ on $x \in [-L, L]$ from $t \in [0, T]$. The initial condition is $u(x, 0) = \cos(\frac{\pi x}{2L})$. We use the following finite difference scheme to discretize the equations

$$\partial_t u(x,t) = \frac{u(x, t + \Delta t/2) - u(x, t - \Delta t/2)}{\Delta t}$$
$$\partial_{xx} u(x,t) = \frac{-\frac{1}{12}u(x+2h,t) + \frac{4}{3}u(x+h,t) - \frac{5}{2}u(x,t) + \frac{4}{3}u(x-2h,t) - \frac{1}{12}u(x-h,t)}{h}.$$

Rearranging, the equations using the implicit finite difference scheme we arrive at linear system with a penta-diagonal design matrix $\mathbf{M}$ to be solved at each time step. In order to estimate the θ, we use Proposition 4.1 and estimate the bound on condition number using Gershgorin circle theorem which gives us $\text{cond}(\mathbf{M}) \leq 1.68$. Consequently, for 5-list parallel sampling we pick $\theta = \frac{8.4}{n}$. The optimization process is investigated in Figure 6.3 (RIGHT), and confirms that the convergence is super-linear in the later stages of the optimization (cca. $t > 2^4$), and that with the increasing number of cores, the speedup achieved is nearly linear.

7. Empirical Risk Minimization in Parallel Settings

A specific application of this optimization method is optimizing the error function of statistical estimation called Empirical Risk Minimization (ERM). This was the main application area of the algorithm in [13] and [24] among many others. We reproduce for the sake of readers convenience the modifications to ERM formulations needed such that PSN can be directly applied.

Algorithm 7.1 PSN: Parallel Stochastic Netwon for ERM

Parameters: Parallel Sampling $\hat{S}$, aggregation parameter b.
Initialization: Pick $\alpha_0 \in \mathbb{R}^n$ and $\bar{\alpha}_0 = \frac{1}{\lambda n}\mathbf{A}$
  for $k = 0, 1, 2, \ldots$ **do**
    Primal update: $w^k = \nabla g^*(\bar{\alpha}_k)$
    Generate a random set of sets S_k distributed according to $\hat{S}$
    for $j = 0, \ldots, c$ **do**
      $h^{k_j} \leftarrow \arg\min_{h \in \mathbb{R}^n} \langle I_{S_j^k} \mathbf{A}^\top w^k, h\rangle + \frac{1}{2}\langle h, \mathbf{X}_{S_j^k} h\rangle + \sum_{i \in S_j^k} \nabla\psi_i(\alpha_i^k) h_i$
    end for
    Dual update: $\alpha^{k+1} \leftarrow \alpha^k + \frac{1}{b}\sum_{j=1}^c h^{k_j}$
    Average update: $\bar{\alpha}^{k+1} \leftarrow \alpha^k + \frac{1}{n\lambda b}\sum_{j=1}^c \mathbf{A}^\top h^{k_j}$
  end for

Empirical risk minimization (ERM) problems can be cast in the form

$$\min_{w \in \mathbb{R}^d} \left[P(w) := \frac{1}{n}\sum_{i=1}^n \phi_i(a_i^\top w) + \lambda g(w) \right]. \tag{7.1}$$

We assume g is $1-$strongly convex with respect to l_2 norm and $\phi_i : \mathbb{R} \to \mathbb{R}$ are γ_i-strongly convex and smooth. The vector a_i represents a feature vector of data point i.

Using Fenchel duality theory, we are able to derive a dual optimization problem to the one in (7.1). Fenchel conjugate function of g is denoted g^* and defined via $g^*(s) := \sup_{w \in \mathbb{R}^d} \langle w, s\rangle - g(w)$ in this work. Given this definition, we are able to formulate the dual problem where the solution is equivalent to the primal problem given strong duality holds.

$$\max_{\alpha \in \mathbb{R}^n} \left[D(\alpha) := \frac{1}{n}\sum_{i=1}^n -\phi_i^*(-\alpha_i) - \lambda g^*\left(\frac{1}{n\lambda}\mathbf{A}\alpha\right) \right]. \tag{7.2}$$

Also, we want to mention that under strong duality conditions the relation between primal and dual variables can be simply expressed as

$$w = w(\alpha) = \nabla f\left(\frac{1}{\lambda n}\mathbf{A}\alpha\right), \tag{7.3}$$

and consequently $w(\alpha^*) = w^*$ where the star denotes the optimal solution to the optimization problems.

Redefining the functions in expression (7.2) by choosing $f(\alpha) = \lambda g^*(\frac{1}{\lambda n}\mathbf{A}\alpha)$ and $\psi_i(\alpha_i) = \frac{1}{n}\phi_i^*(-\alpha_i)$, and exchanging maximization for minimization, we yield the following problem

$$\min_{\alpha \in \mathbb{R}^n} \left[F(\alpha) := f(\alpha) + \sum_{i=1}^n \psi_i(\alpha_i) \right], \tag{7.4}$$

where f satisfies the Assumption 2.1 and ψ_i is strongly convex and smooth with the constant $\frac{1}{\gamma_i n}$. Thus, we see that the assumptions needed to apply Algorithm 7.1 are satisfied and we can define a specialized Algorithm 7.1 for this particular problem.

Due to duality theory, g being 1-strongly convex implies that g^* has 1-Lipschitz continuous gradient [1], and thus $\nabla^2 g^*(x) \preceq \mathbf{I}_{d\times d}$. Consequently, this implies that for $f(\alpha)$ defined via g^* we have

$$\nabla^2 f(\alpha) \preceq \frac{1}{\lambda n^2}\mathbf{A}^\top \mathbf{A}.$$

Therefore, in this case, summing the two matrices to obtain matrix for Assumption 2.1, we get

$$\mathbf{X} := \frac{1}{\lambda n^2}\mathbf{A}^\top \mathbf{A} + \frac{\mathbf{D}(\gamma)^{-1}}{n}.$$

As we have just applied the previous Algorithm 7.1 to solve a specific problem in (7.2), the convergence rates are the same, where we just need to replace $\mathbf{M}$ with $\mathbf{X}$ in the Theorem 3.2. We do not present a bound on the duality gap due to technical difficulties, but we conjecture that the bound presented in [13] most likely holds.

8. Conclusion

We have developed and analyzed a parallel variant of the stochastic Newton (SN) introduced in [13]. We establish that converge guarantees are improved in certain regimes. The algorithm performs better than its coordinate counterpart known as parallel coordinate descent method (PCDM), both in theory and in practice. We highlighted cases when the parallel version enjoys theoretical speedup over the serial method. We include examples from statistical estimation as well as numerical analysis.

A. Proof of Lemma 4.3

We assume that S is a sampling resulting in the selection of a random sub-matrix of size k. The proof of the ordering (4.4) is equivalent to showing the result for $k \times k$ matrix principal minors $(\mathbf{M}_{SS})^{-1}$ from (1.2).

In the following analysis, we suppose (without loss of generality) that the sub-matrix that is sampled with S is located in upper-left corner of $\mathbf{M}$. Let us denote this sub-matrix by $\mathbf{A}$. Moreover, let $\mathbf{M}$ and $\mathbf{M}^{-1}$ have the following block structure:

$$\mathbf{M} = \begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{C} \end{pmatrix}, \qquad \mathbf{M}^{-1} = \begin{pmatrix} \mathbf{X} & \mathbf{Y} \\ \mathbf{Y}^\top & \mathbf{Z} \end{pmatrix}.$$

By [28, Theorem 2.4], we know that $\mathbf{X}$ is related to the Shur complement of $\mathbf{A}$. We denote the Shur complement of $\mathbf{A}$ as $\tilde{\mathbf{A}}$. Hence, $\mathbf{X} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\tilde{\mathbf{A}}^{-1}\mathbf{B}^\top\mathbf{A}^{-1} = \mathbf{A}^{-1} + \mathbf{K}$. Then as $\mathbf{K}$ is positive definite by Lemma 4.2 we have

$$(\mathbf{M}^{-1})_{SS} = \mathbf{X} = \mathbf{A}^{-1} + \mathbf{K} \succeq \mathbf{A}^{-1} = (\mathbf{M}_{SS})^{-1}. \tag{A.1}$$

Should S be a sampling that does not sample a principal sub-matrix in the upper left corner then we can consider the matrix $\mathbf{Z} = \mathbf{Q}^\top\mathbf{M}\mathbf{Q}$ where $\mathbf{Q}$ is a matrix that permutes the elements

such that the desired sub-matrix is in the upper left position. Then we can define a sampling S' s.t. $\mathbf{I}_{:S} = \mathbf{Q}\mathbf{I}_{:S'}$, which yields

$$\begin{aligned}(\mathbf{M}_{SS})^{-1} &\overset{(1.2)}{=} (\mathbf{I}_{:S}^\top \mathbf{M}\mathbf{I}_{:S})^{-1} = (\mathbf{I}_{:S'}^\top \mathbf{Q}^\top \mathbf{M}\mathbf{Q}\mathbf{I}_{:S'})^{-1} \\ &= (\mathbf{Z}_{S'S'})^{-1} \overset{(A.1)}{\succeq} (\mathbf{Z}^{-1})_{S'S'} = (\mathbf{M}^{-1})_{SS}.\end{aligned}$$

B. Proof of Theorem 3.2

It will be convenient to introduce the following notation:

$$\mathbf{X} := \mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]^{1/2}\mathbf{G}\mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]^{1/2} \tag{B.1}$$

$$g^k := \nabla f(x^k) \tag{B.2}$$

$$z^k := (\mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}])^{1/2}\nabla f(x^k) \tag{B.3}$$

$$\mathbf{Y} := \mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]^{1/2}\mathbf{M}\mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]^{1/2}. \tag{B.4}$$

Recalling (2.1), we get

$$f(x^{k+1}) = f(x^k + h^k) \le f(x^k) + \langle g^k, h^k\rangle + \frac{1}{2}\langle h^k, \mathbf{M}h^k\rangle,$$

and thus

$$\begin{aligned}f(x^{k+1}) - f(x^k) &\le \langle g^k, h^k\rangle + \frac{1}{2}\langle h^k, \mathbf{M}h^k\rangle \\ &\overset{(3.5)}{=} -\frac{1}{b}\langle g^k, \sum_{i=1}^c (\mathbf{M}_{S_i^k})^{-1}g^k\rangle + \frac{1}{2b^2}\langle \mathbf{M}\sum_{j=1}^c (\mathbf{M}_{S_j^k})^{-1}g^k, \sum_{i=1}^c (\mathbf{M}_{S_i^k})^{-1}g^k\rangle \\ &\overset{(1.3)}{=} -\frac{1}{b}\langle g^k, \sum_{i=1}^c (\mathbf{M}_{S_i^k})^{-1}g^k\rangle + \frac{1}{2b^2}\langle \mathbf{M}\sum_{j=1}^c \mathbf{I}_{S_j^k}(\mathbf{M}_{S_j^k})^{-1}g^k, \sum_{i=1}^c \mathbf{I}_{S_i^k}(\mathbf{M}_{S_i^k})^{-1}g^k\rangle \\ &= -\frac{1}{b}\sum_{i=1}^c \langle g^k, (\mathbf{M}_{S_i^k})^{-1}g^k\rangle + \frac{1}{2b^2}\sum_{j=1}^c\sum_{i=1}^c \langle \mathbf{M}\mathbf{I}_{S_j^k}(\mathbf{M}_{S_j^k})^{-1}g^k, \mathbf{I}_{S_i^k}(\mathbf{M}_{S_i^k})^{-1}g^k\rangle.\end{aligned}$$

We split the last term

$$\begin{aligned}&f(x^{k+1}) - f(x^k) \\ &\le -\frac{1}{b}\sum_{i=1}^c \langle g^k, (\mathbf{M}_{S_{k_i}})^{-1}g^k\rangle + \frac{1}{2b^2}\sum_{i=1}^c \langle \mathbf{M}\mathbf{I}_{S_{k_i}}(\mathbf{M}_{S_{k_i}})^{-1}g^k, \mathbf{I}_{S_{k_i}}(\mathbf{M}_{S_{k_i}})^{-1}g^k\rangle \\ &\quad + \frac{1}{2b^2}\sum_{j\ne i}^c \langle \mathbf{M}\mathbf{I}_{S_j^k}(\mathbf{M}_{S_j^k})^{-1}g^k, \mathbf{I}_{S_i^k}(\mathbf{M}_{S_i^k})^{-1}g^k\rangle \\ &\overset{(1.4),(1.3)}{=} -\frac{1}{b}\sum_{i=1}^c \langle g^k, (\mathbf{M}_{S_i^k})^{-1}g^k\rangle + \frac{1}{2b^2}\sum_{i=1}^c \langle g^k, (\mathbf{M}_{S_i^k})^{-1}g^k\rangle \\ &\quad + \frac{1}{2b^2}\sum_{j\ne i}^c \langle \mathbf{M}(\mathbf{M}_{S_j^k})^{-1}g^k, (\mathbf{M}_{S_i^k})^{-1}g^k\rangle.\end{aligned}$$

By taking expectations, we can write

$$\begin{aligned}
&\mathbb{E}[f(x^{k+1}) - f(x^k)] \\
&\overset{(4.1)}{\leq} -\left(\frac{1}{b} - \frac{1}{2b^2}\right)\sum_{i=1}^{c}\langle g^k, \mathbb{E}[(\mathbf{M}_S)^{-1}]g^k\rangle + \frac{1}{2b^2}\sum_{j\neq i}^{c}\langle \mathbf{M}\mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]g^k, \mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]g^k\rangle \\
&\overset{(B.3)}{=} -\left(\frac{c}{b} - \frac{c}{2b^2}\right)\langle z^k, z^k\rangle + \frac{c^2-c}{2b^2}\langle \mathbf{M}\mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]^{1/2}z^k, \mathbb{E}[(\mathbf{M}_{\hat{S}})^{-1}]^{1/2}z^k\rangle \\
&\overset{(B.4)}{=} -\left(\frac{c}{b} - \frac{c}{2b^2}\right)\langle z^k, z^k\rangle + \frac{c^2-c}{2b^2}\langle z^k, \mathbf{Y}z^k\rangle \\
&\overset{(B.1),(3.7)}{\leq} -\left(\frac{c}{b} - \frac{c}{2b^2}\right)\langle z^k, z^k\rangle + \frac{\lambda(c^2-c)}{2b^2}\langle z^k, \mathbf{X}z^k\rangle \\
&\overset{(3.8)}{\leq} -\left(\frac{c}{b} - \frac{c}{2b^2}\right)\langle z^k, z^k\rangle + \frac{\lambda\theta(c^2-c)}{2b^2}\langle z^k, z^k\rangle\,. \qquad \text{(B.5)}
\end{aligned}$$

Finally,

$$\mathbb{E}[f(x^{k+1}) - f(x^k)] \overset{(B.5),(3.6)}{=} -\frac{c}{2b}\langle z^k, z^k\rangle \overset{(B.3),(3.4)}{\leq} -\frac{c}{2b}\sigma_1\langle g^k, \mathbf{G}^{-1}g^k\rangle \overset{(2.3),(3.10)}{\leq} -\sigma_p(f(x^k) - f(x^*)).$$

Acknowledgments. The first author would like to thank EPRSC Vacational scholarship of the University of Edinburgh for supporting this work. The second author would like acknowledge support through EPSRC Early Career Fellowship in Mathematical Sciences.

References

[1] Celestine Dünner, Simone Forte, Martin Takáč, and Martin Jaggi. Primal-dual rates and certificates. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 783–792, 2016.

[2] Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled Newton methods. In *Advances in Neural Information Processing Systems*, pp. 3034–3042, 2015.

[3] Olivier Fercoq and Peter Richtárik. Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, 25(4):1997–2023, 2015.

[4] Kimon Fountoulakis and Rachael Tappenden. A flexible coordinate descent method for big data applications. *arXiv preprint arXiv:1507.03713*, 2015.

[5] Semyon Gerschgorin. Über die Abgrenzung der Eigenwerte einer Matrix. *Izv. Akad. Nauk. USSR Otd. Fiz.-Matt. Nauk (in German)*, 6:749–754, 1931.

[6] Robert M Gower, Donald Goldfarb, and Peter Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1869–1878, 2016.

[7] Kaggle. House prices: Advanced regression techniques. Online, July 2017.

[8] Dong C Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

[9] Jakub Mareček, Peter Richtárik, and Martin Takáč. Distributed block coordinate descent for minimizing partially separable functions. *Numerical Analysis and Optimization, Springer Proceedings in Math. and Statistics*, 134:261–288, 2015.

[10] Silvia Noschese, Lionello Pasquini, and Lothar Reichel. Tridiagonal Toeplitz matrices: properties and novel applications. *Numerical Linear Algebra with Applications*, 20(2):302–326, 2013.

[11] Mert Pilanci and Martin J Wainwright. Newton Sketch: A linear-time optimization algorithm with linear-quadratic convergence. *arXiv preprint arXiv:1505.02250*, 2015.

[12] Zheng Qu and Peter Richtárik. Coordinate descent with arbitrary sampling II: Expected separable overapproximation. *Optimization Methods and Software*, 31(5):858–884, 2016.

[13] Zheng Qu, Peter Richtárik, Martin Takáč, and Olivier Fercoq. SDNA: Stochastic dual Newton ascent for empirical risk minimization. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1823–1832, 2016.

[14] Zheng Qu, Peter Richtárik, and Tong Zhang. Randomized dual coordinate ascent with arbitrary sampling. In *Advances in Neural Information Processing Systems*, pp. 865–873, 2015.

[15] Carl E. Rasmussen and Christopher K. I. Williams. Gaussian processes for machine learning. *The MIT Press, Cambridge*, 10, 2006.

[16] Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 693–701, 2011.

[17] Peter Richtárik and Martin Takáč. Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function. *Mathematical Programming*, 144(2):1–38, 2014.

[18] Peter Richtárik and Martin Takáč. On optimal probabilities in stochastic coordinate descent methods. *Optimization Letters*, 10(6):1233–1243, 2015.

[19] Peter Richtárik and Martin Takáč. Distributed coordinate descent method for learning with big data. *Journal of Machine Learning Research*, 17(75):1–25, 2016.

[20] Peter Richtárik and Martin Takáč. Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1):433–484, 2016.

[21] Peter Richtárik and Martin Takáč. Stochastic reformulations of linear systems: Algorithms and convergence theory. *arXiv preprint arXiv:1706.01108*, 2017.

[22] Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled Newton methods I: Globally convergent algorithms. *arXiv preprint arXiv:1601.04737*, 2016.

[23] Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled Newton methods II: Local convergence rates. *arXiv preprint arXiv:1601.04738*, 2016.

[24] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research*, 14(1):567–599, 2013.

[25] Llewellyn Thomas. Elliptic problems in linear differential equations over a network. *Watson scientific computing laboratory report*, 1949.

[26] Holger Wendland. *Scattered data approximation*, volume 17. Cambridge university press, 2004.

[27] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, April 2009.

[28] Fuzhen Zhang. *Matrix Theory: Basic Results and Techniques*. Springer Science & Business Media, 2011.

[29] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pp. 2595–2603, 2010.