

WORTH THEIR WEIGHT: RANDOMIZED AND REGULARIZED BLOCK KACZMARZ ALGORITHMS WITHOUT PREPROCESSING *

GIL GOLDSHLAGER [†], JIANG HU [†], AND LIN LIN ^{†‡}

Abstract. Due to the ever growing amounts of data leveraged for machine learning and scientific computing, it is increasingly important to develop algorithms that sample only a small portion of the data at a time. In the case of linear least-squares, the randomized block Kaczmarz method (RBK) is an appealing example of such an algorithm, but its convergence is only understood under sampling distributions that require potentially prohibitively expensive preprocessing steps. To address this limitation, we analyze RBK when the data is sampled uniformly, showing that its iterates converge in a Monte Carlo sense to a *weighted* least-squares solution. Unfortunately, for general problems the bias of the weighted least-squares solution and the variance of the iterates can become arbitrarily large. We show that these quantities can be rigorously controlled by incorporating regularization into the RBK iterations, yielding the regularized algorithm ReBlock. Numerical experiments including examples arising from natural gradient optimization demonstrate that ReBlock can outperform both RBK and minibatch stochastic gradient descent for inconsistent problems with rapidly decaying singular values.

Key words. randomized Kaczmarz, linear least-squares, inconsistent problems, uniform sampling, regularization, natural gradient descent

MSC codes. 65F10, 65F20, 68W20, 15A06

1. Introduction.

Consider the linear least-squares problem

$$(1.1) \quad \min_{x \in \mathbb{R}^n} \|Ax - b\|^2, \\ A \in \mathbb{R}^{m \times n}, \quad b \in \mathbb{R}^m.$$

The minimal-norm ordinary least-squares solution is

$$(1.2) \quad x^* = A^+ b,$$

where A^+ is the Moore-Penrose pseudoinverse of A .

In this work, we are interested in algorithms that solve (1.1) by sampling just a small number of rows at a time. Such algorithms can be useful for extremely large problems for which direct methods and Krylov subspace methods are prohibitively expensive [7]. Of particular interest are problems in which the dimension n is so large that it is only possible to access $k \ll n$ rows at a time, but not so large that it is necessary to take $k = 1$. As a canonical example, we might have $m = 10^9$, $n = 10^6$, and $k = 10^3$.

As further motivation for algorithms of this type, there are some applications in which sampling rows is the only efficient way to access the data. For example, the

*G.G. acknowledges support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Department of Energy Computational Science Graduate Fellowship under Award Number DE-SC0023112. This effort was supported by the SciAI Center, and funded by the Office of Naval Research (ONR), under Grant Number N00014-23-1-2729 (J.H., L.L.). L.L. is a Simons Investigator in Mathematics. This research used the Savio computational cluster resource provided by the Berkeley Research Computing program at the University of California, Berkeley.

[†]Department of Mathematics, UC Berkeley, Berkeley, CA 94720 USA (ggoldsh@berkeley.edu, jianghu@berkeley.edu, linlin@berkeley.edu).

[‡]Applied Mathematics and Computational Research Division, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 USA (linlin@lbl.gov).

rows may be computed on the fly, especially when solving the “semi-infinite” version of (1.1) in which the rows are indexed by continuous variables rather than discrete integers [41]. An example that captures both of these features is the problem of calculating natural gradient directions for continuous function learning problems; see Section 6.

One well-known approach for solving (1.1) using only a few rows at a time is minibatch stochastic gradient descent (mSGD). The mSGD algorithm works by sampling k rows uniformly at random and using them to calculate an unbiased estimator of the gradient of the least-squares loss function. This is equivalent to averaging the k independent gradient estimators furnished by each sampled row. For a thorough discussion of mSGD, see [24].

The technique of averaging used in mSGD, while simple and efficient, is a relatively crude way of processing a block of k rows. A more sophisticated approach is provided by the randomized block Kaczmarz method, which goes beyond averaging by making use of the pseudoinverse of the sampled block [30]. As we shall see, the use of the pseudoinverse opens up the possibility of faster convergence, but it also creates a number of complications regarding the implementation and analysis of the algorithm.

1.1. Notation. We denote by $r = b - Ax^*$ the residual vector of (1.1), by $a_i^\top \in \mathbb{R}^n$ the row of A with index i , and by $b_i \in \mathbb{R}$ the corresponding entry of b . Additionally, for an index set $S \subseteq \{1, \dots, m\}$ with $|S| = k$, let $A_S \in \mathbb{R}^{k \times n}$ represent the block of rows of A whose indices are in S , and let $b_S \in \mathbb{R}^k$ represent the corresponding entries of b . The same subscript notations will also be applied as needed to any matrices and vectors other than A and b . Additionally, denote by $\mathbf{U}(m, k)$ the uniform distribution over all size- k subsets of $\{1, \dots, m\}$.

For symmetric matrices X, Y , denote by $X \succ Y$ that the difference $X - Y$ is positive definite, and define \succeq, \prec, \preceq correspondingly. For any vector x and any positive definite matrix Y of the same size, let $\|x\|_Y = \sqrt{x^\top Y x}$. For any matrix X , denote by $\|X\|_F$ its Frobenius norm and by $\sigma_{\min}^+(X)$ its minimum nonzero singular value.

1.2. Randomized Kaczmarz. The modern version of the randomized Kaczmarz method (RK) was proposed by Thomas Strohmer and Roman Vershynin in 2009 [42]. In RK, an initial guess x_0 is updated iteratively using a two-step procedure:

1. *Sample* a row index $i_t \in \{1, \dots, m\}$ with probability proportional to $\|a_{i_t}\|^2$.
2. *Update*

$$(1.3) \quad x_{t+1} = x_t + a_{i_t} \frac{b_{i_t} - a_{i_t}^\top x_t}{\|a_{i_t}\|^2}.$$

To reduce the sampling cost, it is also possible to run RK with uniform sampling, which is equivalent to running RK on a diagonally reweighted problem [31].

The iteration (1.3) has the interpretation of projecting x_t onto the hyperplane of solutions to the individual equation $a_{i_t}^\top x = b_{i_t}$. For consistent systems, $Ax^* = b$, the RK iterates x_t converge linearly to x^* with a rate that depends on the conditioning of A . For inconsistent systems, $Ax^* \neq b$, RK converges only to within a finite horizon of the ordinary least-squares solution x^* [29]. In particular, the expected squared error $\mathbb{E} \|x_t - x^*\|^2$ converges to a finite, nonzero value that depends on the conditioning of A and the norm of the residual vector $r = b - Ax^*$. See Theorem 7 of [44] for the strongest known convergence bound of this type.

1.3. Tail Averaging. Tail averaging is a common technique for boosting the accuracy of stochastic algorithms [35, 24, 19]. Given a series of stochastic iterates x_0, \dots, x_T and a burn-in time T_b , the tail-averaged estimator is given by

$$(1.4) \quad \bar{x}_T = \frac{1}{T - T_b} \sum_{t=T_b+1}^T x_t.$$

The recent work [19] shows that applying tail averaging to the RK iterates yields exact convergence (with no finite horizon) to the ordinary least-squares solution x^* , even for inconsistent systems. Building on these results, we will make use of tail averaging to obtain exact convergence to a *weighted* least-squares solution in the block case.

1.4. Randomized Block Kaczmarz. Randomized block Kaczmarz (RBK) is an extension of RK which uses blocks of rows to accelerate the convergence and make better use of parallel and distributed computing resources [18, 30]. Like RK, each RBK iteration proceeds in two steps:

1. *Sample* a subset $S_t \subset \{1, \dots, m\}$ of the row indices from some chosen sampling distribution ρ .
2. *Update*

$$(1.5) \quad x_{t+1} = x_t + A_{S_t}^+ (b_{S_t} - A_{S_t} x_t).$$

Here $A_{S_t}^+$ is the Moore-Penrose pseudoinverse of A_{S_t} . The iteration (1.5) has the interpretation of projecting x_t onto the hyperplane of solutions to the block of equations $A_{S_t} x = b_{S_t}$. The RBK method can also be viewed as a “sketch-and-project” algorithm; see [21].

There have been many proposals for how to choose the blocks in the RBK method. One idea is to use a preprocessing step to partition the matrix A into well-conditioned blocks, then sample this fixed set of blocks uniformly [30]. It has also been suggested to preprocess the matrix with an incoherence transform, which can make it easier to generate a well-conditioned partition [30] or, relatedly, enable RBK with uniform sampling to converge rapidly for the transformed problem [17]. The work of [17] also provides an analysis of the RBK algorithm when sampling from a determinantal point process, and other proposals include greedy block Kaczmarz algorithms such as [25] which require evaluating the complete residual vector at each iteration.

Unfortunately most of these proposals apply only to consistent linear systems, and all of them require at least a preprocessing step in which the entire data matrix must be accessed. Such preprocessing can be prohibitively expensive for very large-scale problems, for which 1) it can be necessary to furnish an approximate solution without processing the entire data set even once (for instance, in the semi-infinite case), and 2) it can be impossible to manipulate more than a tiny subset of the data at a time due to storage constraints. This leads us to the central question of our work:

QUESTION 1.1. *Can RBK, or some variant thereof, be applied to solve inconsistent linear systems without preprocessing the input matrix?*

To avoid preprocessing, we consider the case that the sampling strategy is chosen *a priori* in that it is not necessarily adapted to the data matrix A or the right-hand side b . To simplify the analysis, we focus concretely on the case when the sampling is uniform. This is a natural choice since (1.1) can be viewed as a uniform mixture of

m distinct rows. We emphasize that this choice is made for simplicity and that our results are not inherently restricted to the case of uniform sampling. For example, our results can be readily generalized to both weighted and semi-infinite problems of the form

$$(1.6) \quad \min_{x \in \mathbb{R}^n} \mathbb{E}_{i \sim \mu} [(a_i^\top x - b_i)^2],$$

for which the corresponding algorithms would independently sample k indices $i_1, \dots, i_k \sim \mu$. This is especially relevant for scientific applications, in which data is often continuous and may be sampled using a physics-based probability distribution. For example, in the case of neural network wavefunctions [23], the sampling distribution is known as the Born probability density.

1.5. Contributions. In this work, we make substantial progress towards answering our central question 1.1. We first provide a new analysis of the RBK algorithm under uniform sampling which shows both the strengths and weaknesses of the algorithm. Inspired by this analysis, we propose and analyze a regularized algorithm, ReBlock, which behaves more robustly for general types of data. Concretely, we summarize our contributions as follows:

1. **We demonstrate that the RBK algorithm with uniform sampling (RBK-U) converges in a Monte Carlo sense to a weighted least-squares solution for both consistent and inconsistent linear systems.** In particular, Theorem 3.1 shows that convergence is obtained by both expectation values of individual iterates and tail averages of the sequence of iterates. The weight matrix depends on the block size k and the matrix A , but not on the vector b . Our results provide a new perspective on RBK in the inconsistent case, going beyond previous analyses that only characterized proximity to the ordinary least-squares solution.
2. **We provide a new perspective on the pitfalls of RBK with uniform sampling.** Concretely, our analysis reveals that when the problem contains many blocks A_S that are nearly singular, the bias of the weighted least-squares solution and the variance of the iterates can become arbitrarily large. See Theorems 3.1 and 3.2 for theoretical barriers and Figure 2 for numerical examples that manifest these issues.
3. **We show that RBK-U performs robustly and efficiently for Gaussian data.** Concretely, Theorem 4.1 shows that convergence is obtained to the ordinary least-squares solution when the data arises from a multivariate Gaussian distribution. Furthermore, in this case both the variance of the iterates and the convergence parameter α can be explicitly bounded. When the singular values of the covariance matrix decay rapidly, the convergence rate can be much faster than mSGD; see Corollary 4.2 and Figure 3.
4. **We propose and analyze a regularized algorithm to handle more general types of data.** To provide a more general solution, we propose to regularize the RBK iterations as follows:

$$(1.7) \quad x_{t+1} = x_t + A_{S_t}^\top (A_{S_t} A_{S_t}^\top + \lambda k I)^{-1} (b_{S_t} - A_{S_t} x_t).$$

We refer to this algorithm as the regularized block Kaczmarz method, or ReBlock. Similar to RBK-U, we show that ReBlock with uniform sampling (ReBlock-U) converges in a Monte Carlo sense to a weighted least-squares solution; see Theorem 5.1. Unlike for RBK-U, the bias of the weighted least-squares solution and the variance of the iterates can be controlled in terms of

just λ and some coarse properties of the data A, b . This makes ReBlock-U much more reliable than RBK-U in practice; see Figures 6 and 8. As an added benefit, ReBlock iterations can be significantly more efficient than RBK iterations; see Subsection 5.2 and Figure 12.

5. **We demonstrate promising results for using ReBlock to calculate natural gradient directions.** Our initial motivation for this work came from the problem of calculating natural gradient directions for deep neural networks. In Section 6, we explain how this setting naturally lends itself to the kinds of linear least-squares solvers that we study in this paper. Encouragingly, Figure 8 shows that ReBlock-U outperforms mSGD and RBK-U when calculating natural gradient directions for a simple neural function regression task.

Altogether, our results demonstrate that regularization can improve the robustness of the randomized block Kaczmarz algorithm and unlock new applications to challenging problems for which sophisticated preprocessing schemes are out of reach. Furthermore, while our focus is on the case of uniform sampling, the same proof techniques can be applied to other sampling distributions. For example, in section E we show that sampling from an appropriate determinantal point process can in theory enable tail averages of ReBlock iterates to converge rapidly to the ordinary least-squares solution for arbitrary inconsistent problems.

1.6. Related Works. Our regularized algorithm, ReBlock, is closely related to the iterated Tikhonov-Kaczmarz method [12]. ReBlock can also be viewed as a specific application of stochastic proximal point algorithms (sPPA) [3, 2, 11] for solving stochastic optimization problems with objective functions of the least-squares type. To ensure the exact convergence of sPPA, diminishing step sizes are required; see for example [33]. In contrast, our work investigates the convergence of sPPA with a large constant step size for the special case of a least-squares loss. Such an approach allows for aggressive updates throughout the algorithm, potentially improving its practical efficiency.

The ReBlock algorithm is additionally a special case of the SlimLS algorithm proposed by [9] for solving massive linear inverse problems. In fact, their convergence result for the expectation of the ReBlock iterates, found in result (a) of their Theorem 3.1, is equivalent to equation (5.3) of our Theorem 5.1. However, by identifying the limiting expectation value as a weighted least-squares solution, we provide greater conceptual clarity regarding its meaning. Additionally, our bounds regarding the variance of the ReBlock iterates, the convergence of tail averages, and the suboptimality of the weighted least-squares solution are all much stronger than the corresponding bounds of [9]. These stronger bounds serve to explain our numerical findings that ReBlock can be robust and efficient even when λ is very small and the problem contains many nearly singular blocks A_S , which is the most relevant case and is not explained by the previous analysis.

Our work is also related to the nearly concurrent paper [15], which introduces Tikhonov regularization into the RBK iterations just like ReBlock. [15] focuses on consistent systems and in this context, the regularization gives rise to optimal convergence rates in the presence of Nesterov acceleration. On the other hand, our work focuses on solving inconsistent systems, in which case the regularization is needed to ensure the stability of the algorithm. Exploring the combination of regularization and Nesterov acceleration in the inconsistent case is a promising direction for future work.

2. Overview of the Analysis. Our results are stated in terms of a unified framework that includes RBK, ReBlock, and even mSGD as special cases. Consider the following iteration:

1. *Sample* $S_t \sim \rho$.
2. *Update*

$$(2.1) \quad x_{t+1} = x_t + A_{S_t}^\top M(A_{S_t})(b_{S_t} - A_{S_t} x_t).$$

Here $M(\cdot) : \mathbb{R}^{k \times n} \rightarrow \mathbb{R}^{k \times k}$ takes in the sampled block A_{S_t} and returns a positive semidefinite “mass” matrix. RBK is recovered by setting $M(A_{S_t}) = (A_{S_t} A_{S_t}^\top)^+$ and ReBlock by setting $M(A_{S_t}) = (A_{S_t} A_{S_t}^\top + \lambda k I)^{-1}$. See Algorithm 2.1 for the full procedure with and without tail averaging.

Once the function $M(A_S)$ is chosen, let

$$(2.2) \quad W(S) = I_S^\top M(A_S) I_S, \quad P(S) = A_S^\top M(A_S) A_S,$$

where I_S represents the k rows of the $n \times n$ identity matrix whose indices are in S (recall $|S| = k$). These quantities are natural because they enable the general iteration (2.1) to be rewritten as

$$(2.3) \quad x_{t+1} = (I - P(S_t))x_t + A^\top W(S_t)b.$$

Note that $P(S_t)$ is a projection matrix in the case of RBK.

Next, let

$$(2.4) \quad \overline{W} = \mathbb{E}_{S \sim \rho} [W(S)], \quad \overline{P} = \mathbb{E}_{S \sim \rho} [P(S)].$$

Additionally, define the weighted solution $x^{(\rho)}$ and the weighted residual $r^{(\rho)}$ via

$$(2.5) \quad x^{(\rho)} = \operatorname{argmin}_{x \in \mathbb{R}^n} \|Ax - b\|_{\overline{W}}^2, \quad r^{(\rho)} = b - Ax^{(\rho)}.$$

When the solution to the weighted problem is not unique, let $x^{(\rho)}$ refer to the minimal-norm solution.

Using these definitions, we can further rewrite the iteration (2.1) as

$$(2.6) \quad x_{t+1} - x^{(\rho)} = (I - P(S_t))(x_t - x^{(\rho)}) + A^\top W(S_t)r^{(\rho)}.$$

Since the normal equations for (2.5) can be written as $A^\top \overline{W} r^{(\rho)} = 0$, we observe that the final term in (2.6) vanishes in expectation. Indeed, identifying the appropriate weighted solution $x^{(\rho)}$ to enable the generalized iteration (2.1) to be written in the form of (2.6), namely as a linear contraction of the error plus a zero-mean additive term, is the main technical innovation underlying our results. From here, the analysis of [19] can be readily generalized to show that convergence to $x^{(\rho)}$ is obtained.

3. RBK without Preprocessing . We begin by presenting our general convergence bounds for RBK-U. We emphasize that these bounds should not be viewed as a proof that algorithm is effective, but rather as a helpful framework to understand the properties and the potential problems of the algorithm. Indeed we will later demonstrate severe failure modes of RBK-U that are inspired by the analysis (Theorem 3.2, Figure 2) and we will ultimately show that regularization is needed in order to eliminate these failure modes and obtain satisfying bounds on the bias and the variance (Theorem 5.1).

Algorithm 2.1 Generalized iterative least-squares solver with optional tail averaging

Input: Data A, b , block size k , initial guess x_0
Input: Mass matrix $M(A_S)$, sampling distribution ρ
Input: Total iterations T , optional burn-in time T_b
for $t = 0$ **to** $T - 1$ **do**
 Sample $S_t \sim \rho$
 $x_{t+1} = x_t + A_{S_t}^\top M(A_{S_t})(b_{S_t} - A_{S_t}x)$
end for
if T_b is not provided **then**
 Return x_T
end if
 $\bar{x}_T = \frac{1}{T-T_b} \sum_{t=T_b+1}^T x_t$
Return \bar{x}_T

THEOREM 3.1. Consider the RBK-U algorithm, namely Algorithm 2.1 with $M(A_S) = (A_S A_S^\top)^+$ and $\rho = \mathbf{U}(m, k)$. Let $\alpha = \sigma_{\min}^+(\bar{P})$ and assume that $x_0 \in \text{range}(A^\top)$. Then the expectation of the RBK-U iterates x_T converges to $x^{(\rho)}$ as

$$(3.1) \quad \left\| \mathbb{E}[x_T] - x^{(\rho)} \right\| \leq (1 - \alpha)^T \left\| x_0 - x^{(\rho)} \right\|.$$

Furthermore, the tail averages \bar{x}_T converge to $x^{(\rho)}$ as

$$(3.2) \quad \mathbb{E} \left\| \bar{x}_T - x^{(\rho)} \right\|^2 \leq (1 - \alpha)^{T_b+1} \left\| x_0 - x^{(\rho)} \right\|^2 + \frac{2}{\alpha^2(T - T_b)} V$$

with $V = \mathbb{E}_{S \sim \rho} \|A_S^+ r_S^{(\rho)}\|^2$. Finally, when the rows of A are in general position the condition number $\kappa(\bar{W})$, the weighted residual $r^{(\rho)}$, the bias $x^{(\rho)} - x^*$, and the variance V are bounded as

$$(3.3) \quad \begin{aligned} \kappa(\bar{W}) &\leq k \cdot \max_i \|a_i\|^2 \cdot \max_S \|A_S^+\|^2, \\ \|r^{(\rho)}\| &\leq \sqrt{\kappa(\bar{W})} \cdot \|r\|, \\ \|x^{(\rho)} - x^*\| &\leq \sqrt{\kappa(\bar{W}) - 1} \cdot \|A^+\| \cdot \|r\|, \\ V &\leq \max_S \|A_S^+\|^2 \cdot \kappa(\bar{W}) \cdot \frac{k \|r\|^2}{m}. \end{aligned}$$

To our knowledge, this is the first result for inconsistent linear systems that characterizes the exact solution to which the randomized block Kaczmarz iterates converge, albeit in a Monte Carlo sense. The $O(1/T)$ convergence rate for the tail-averaged bound is optimal for row-access methods, and a reasonable default for the burn-in time is $T_b = T/2$; see [19] for a more thorough discussion of these points. The proof of Theorem 3.1, which takes advantage of an orthogonal decomposition of the error term $x_{t+1} - x^{(\rho)}$, can be found in section B.

Unfortunately, Theorem 3.1 does not imply robust convergence for general problems. Indeed, for problems containing nearly singular blocks A_S , the bounds on the condition number, the weighted residual, the bias, and the variance can be arbitrarily large. In fact, these issues are not just defects of the analysis but real problems with RBK-U. To substantiate this, we present a single-parameter family of 3×2 examples that completely breaks RBK-U:

EXAMPLE 3.2 (No-go for RBK-U). *Let $\epsilon > 0$ and*

$$(3.4) \quad A = \begin{bmatrix} 0 & 1 \\ 1 & \epsilon^2 \\ 1 & -\epsilon^2 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 1 + \epsilon \\ 1 - \epsilon \end{bmatrix}.$$

Then $\lim_{\epsilon \rightarrow 0^+} \kappa(A) = \sqrt{2}$ and $\lim_{\epsilon \rightarrow 0^+} \|r\| = 0$, indicating that the problem is well conditioned and approaches consistency as $\epsilon \rightarrow 0$. On the other hand, as $\epsilon \rightarrow 0$ the bottom 2×2 block approaches singularity and so for $k = 2$ the upper bounds of $\kappa(\bar{W})$, $\|r^{(\rho)}\|$, $\|x^{(\rho)} - x^\|$, and V from Theorem 3.1 all diverge in this limit. In fact, this divergence is necessary as it can be verified that for RBK-U with $k = 2$, it holds*

$$(3.5) \quad \lim_{\epsilon \rightarrow 0^+} \kappa(\bar{W}) = \lim_{\epsilon \rightarrow 0^+} \|r^{(\rho)}\| = \lim_{\epsilon \rightarrow 0^+} \|x^{(\rho)} - x^*\| = \lim_{\epsilon \rightarrow 0^+} V = \infty.$$

The fact that the bias *and* variance of the RBK-U iterates diverges while the underlying problem is approaching consistency represents a clear and catastrophic failure of the algorithm. We can understand this failure by noting that geometrically, this problem represents three constraints in two dimensions which together form an isosceles triangle with vertices at $(1 - \epsilon, 0)$, $(1 + \epsilon, 0)$, and $(1, 1/\epsilon)$. When running RBK-U with $k = 2$, the algorithm jumps between the three vertices of the triangle with equal probabilities, and as a result $x^{(\rho)}$ coincides with the centroid of the triangle. The problem is that as $\epsilon \rightarrow 0$ the top vertex walks off to infinity and drags the centroid with it, while the least-squares solution x^* converges to $(1, 0)$. This insight can be used to directly show the divergence of the bias, the residual, and the variance. The residual bound $\|r^{(\rho)}\| \leq \sqrt{\kappa(\bar{W})} \cdot \|r\|$ then implies the divergence of $\kappa(\bar{W})$ since $\|r^{(\rho)}\|$ diverges while $\|r\|$ goes to zero. The situation is depicted visually in Figure 1.

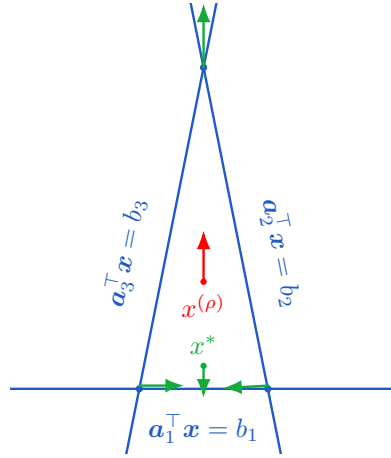


FIG. 1. Visual depiction of the 3×2 linear system (3.4) that causes RBK-U to fail catastrophically when $k = 2$. As $\epsilon \rightarrow 0$ the top vertex walks off to infinity and the recovered solution $x^{(\rho)}$ goes with it, while the true solution x^* approaches the x -axis.

It is worth noting that such problems are not limited to any particular batch size such as $k = 2$. More generally we can view the skinny isosceles triangle as a stretched

2-simplex and indeed, the same arguments can be readily generalized to show that for arbitrary k , a similarly stretched k -simplex suffices to break RBK-U with a block size of k . The problem is also not directly related to the use of uniform sampling, and similar problems could arise for any *a priori* sampling distribution.

3.1. Numerical demonstration. We now demonstrate that the issues highlighted by Theorem 3.2 can occur for more realistic problems. To do so, we set $m = 10^5$, $n = 10^2$ and construct two inconsistent problems whose columns are discretized representations of continuous functions, leading the matrices A to contain many nearly singular blocks of rows. We apply both minibatch SGD (mSGD) and RBK-U to the resulting problems, applying tail averaging to each algorithm to observe convergence beyond the variance horizon. The results in Figure 2 confirm that RBK-U performs poorly for these problems, just as our theory suggests. In fact RBK-U is not even able to reliably attain a relative error of 1.0 in either case, meaning the algorithm performs worse than simply guessing $x^* = 0$. For these and other experiments in Sections 3 to 5 we choose the burn-in time T_b to demonstrate the different phases of convergence as clearly as possible.

We note that for such realistic examples, it is difficult to pin down exactly how large the bias $x^{(\rho)} - x^*$ is. The reason is that the variance V is so large that running the algorithm to convergence would be prohibitively expensive. Nonetheless it is evident that RBK-U is not effective for these problems, even after incorporating tail averaging. The code for all of our experiments can be found at <https://github.com/ggoldsh/block-kaczmarz-without-preprocessing>, and further details on these particular experiments can be found in Section F.

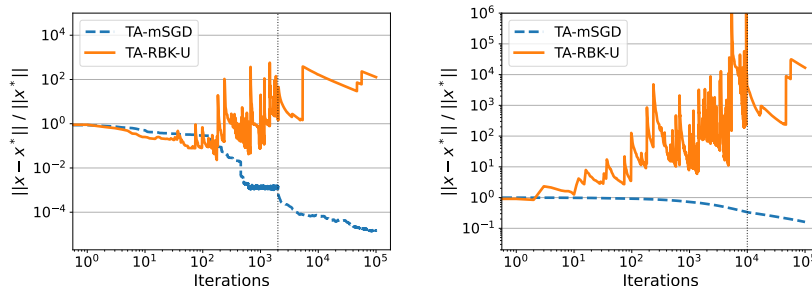


FIG. 2. Failure of RBK-U for two inconsistent problems with many nearly singular blocks, with mild singular value decay (left) and rapid singular value decay (right). The vertical dotted line indicates the burn-in time, before which results are shown for individual iterates.

3.2. Implementation Details. To stably implement the RBK iteration (1.5), we employ a QR-based least-squares solver to calculate $A_{S_t}^+(b_{S_t} - A_{S_t}x)$. The most expensive part of this procedure is the QR decomposition, which has an asymptotic cost of $O(nk^2)$. This cost is greater than the cost of an mSGD iteration, which is $O(nk)$.

4. Linear Least-squares with Gaussian Data. Given the problems with RBK-U for general data, it is natural to wonder if there are more restricted categories of benign problems for which the bias and variance of RBK-U can be controlled. Previous works such as [14, 17, 15] have characterized the performance of RBK-U for consistent systems that have been preprocessed with a randomized Hadamard transform. After preprocessing the resulting system is highly unlikely to contain nearly

singular blocks or other problematic coherence properties, leading to strong convergence guarantees for RBK-U. However, it is not clear why matrices that have not been preprocessed would resemble such preprocessed matrices. Thus, instead of directly extending these results to the inconsistent case, we instead consider inconsistent problems whose rows arise from a Gaussian distribution, which seem more likely to appear in practice. Furthermore, to simplify the analysis we consider the infinite version of the problem in which each uniformly selected row is drawn directly from the corresponding Gaussian distribution. This enables us to exploit an elegant connection between uniform block samples of Gaussian data and dense Gaussian sketches of arbitrary data. It should be possible to extend our results to provide high probability bounds for finite problems using more involved techniques such as those of [14, 17, 15], but we consider this beyond the scope of the current work.

Now, consider the statistical least-squares problem

$$(4.1) \quad x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} \mathbb{E}_{[a^\top \ b] \sim \mathcal{N}(0, Q)} [(a^\top x - b)^2].$$

where $Q \in \mathbb{R}^{(n+1) \times (n+1)}$ is a positive semidefinite covariance matrix. Furthermore, let Q_n be the top left $n \times n$ block of Q , assume for simplicity that Q_n is full rank, and let $Q_n = L_n L_n^\top$ be its Cholesky decomposition. Denote the singular values of L_n by $\sigma_1 \geq \dots \geq \sigma_n > 0$.

THEOREM 4.1. *Consider the equivalent of the RBK-U algorithm for the statistical least-squares problem (4.1), for which each sample A_{S_t}, b_{S_t} is drawn by generating k independent samples from $\mathcal{N}(0, Q)$. Then the results of Theorem 3.1 apply and it holds $x^{(\rho)} = x^*$. In addition, the convergence parameter α satisfies*

$$(4.2) \quad \alpha \geq C_{n,k} \max \left\{ \frac{k\sigma_n^2}{\|L_n\|_F^2}, \max_{2 \leq \ell < k} \frac{(\ell-1)\sigma_n^2}{\sum_{i \geq k-\ell-1} \sigma_i^2} \right\}$$

with $C_{n,k} \rightarrow 1$ as $n \rightarrow \infty$ for fixed k . Furthermore, as long as $k \geq 6$ and $\operatorname{rank}(L_n) \geq 2k$, the variance term satisfies

$$(4.3) \quad V \leq \frac{200}{\sigma_{2k}^2} \cdot \mathbb{E}_{[a^\top \ b] \sim \mathcal{N}(0, Q)} [(a^\top x^* - b)^2].$$

In summary, in the case of Gaussian data the ordinary least-squares solution is recovered, the convergence rate α improves at least linearly with the block size k , and the variance of the iterates is bounded. In addition, the following corollary, which is based on Corollary 3.4 of [16], shows that RBK converges much faster than mSGD for polynomially decaying singular values.

COROLLARY 4.2. *Consider the setting of Theorem 4.1 with fixed $k \leq n/2$, and assume the L_n factor has polynomial spectral decay $\sigma_i^2 \leq i^{-\beta} \sigma_1^2$ for all i and some $\beta > 1$. Then the convergence parameters of RBK and mSGD satisfy*

$$(4.4) \quad \alpha^{\text{RBK}} \geq C k^\beta \frac{\sigma_n^2}{\|L_n\|_F^2}, \quad \alpha^{\text{mSGD}} \leq k \frac{\sigma_n^2}{\|L_n\|_F^2}$$

for some constant $C = C(\beta) > 0$.

Similarly, the faster convergence rate of RBK over mSGD extends to exponentially decaying singular values. The proofs of Theorem 4.1 and Corollary 4.2, provided in section B, rely on a connection between RBK-U for Gaussian data and block Gaussian Kaczmarz for arbitrary data. These results generalize the techniques of [16] to the case of inconsistent linear systems, improving upon the results of [36] in terms of both the convergence rate and the variance.

4.1. Numerical Demonstration. We verify our theoretical results for the RBK-U and mSGD algorithms on two problems with Gaussian data, with results in Figure 3. We apply tail averaging to each algorithm to observe convergence beyond the variance horizon. As expected, tail-averaged RBK-U (TA-RBK-U) converges much more rapidly than tail-averaged mSGD (TA-mSGD) in the presence of fast singular value decay. We additionally explore the effect of choosing different burn-in periods for TA-RBK-U, with results in Figure 4. As expected, optimal performance is attained by setting T_b to be just slightly after the iterates reach their fixed convergence horizon. More details on these experiments can be found in Section F.

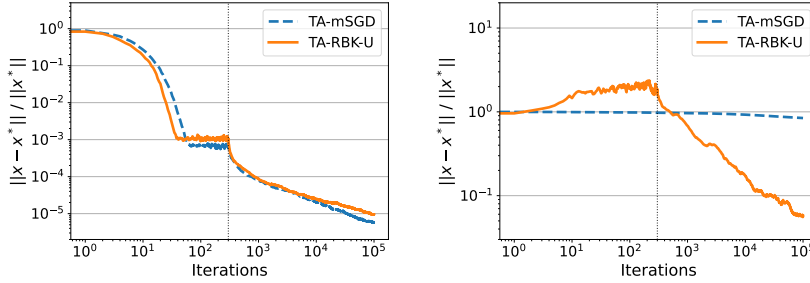


FIG. 3. Comparison of methods on two problems with Gaussian data, with no singular value decay (left) and rapid singular value decay (right). The vertical dotted line indicates the burn-in time, before which results are shown for individual iterates.

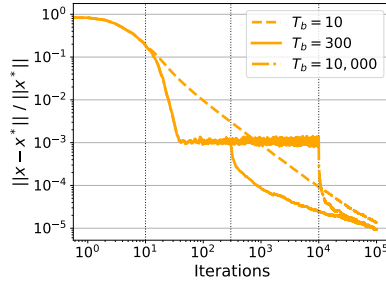


FIG. 4. Behavior of TA-RBK-U with different burn-in periods T_b . The problem is the same as in the left panel of Figure 3.

5. Robustness through Regularization. To address the shortcomings of RBK-U in the case of general data, we propose to incorporate regularization into the RBK algorithm. A natural way to do this is to replace the RBK iteration with a stochastic proximal point iteration [2], namely

$$(5.1) \quad x_{t+1} = \operatorname{argmin}_{x \in \mathbb{R}^n} \left[\|A_{S_t} x - b_{S_t}\|^2 + \lambda k \|x - x_t\|^2 \right].$$

This minimization problem leads to the closed form

$$(5.2) \quad x_{t+1} = x_t + A_{S_t}^\top (A_{S_t} A_{S_t}^\top + \lambda k I)^{-1} (b_{S_t} - A_{S_t} x_t),$$

and that the RBK iteration (1.5) is recovered in the limit $\lambda \rightarrow 0$. To incorporate a mild regularization without significantly slowing down the convergence, we propose to

use a small, constant value of λ throughout the algorithm. We suggest $\lambda = 0.001$ as a practical default value, but to obtain optimal performance the value will need to be tuned on a case-by-case basis; see Figure 10 for some numerical results for different values of λ . We refer to the resulting scheme as the regularized block Kaczmarz method, or ReBlock.

THEOREM 5.1. *Consider the ReBlock-U algorithm, namely Algorithm 2.1 with $M(A_S) = (A_S A_S^\top + \lambda k I)^{-1}$ and $\rho = \mathbf{U}(m, k)$. Let $\alpha = \sigma_{\min}^+(\bar{P})$ and assume $x_0 \in \text{range}(A^\top)$. Then the expectation of the ReBlock iterates x_T converges to $x^{(\rho)}$ as*

$$(5.3) \quad \left\| \mathbb{E}[x_T] - x^{(\rho)} \right\| \leq (1 - \alpha)^T \left\| x_0 - x^{(\rho)} \right\|.$$

Furthermore, the tail averages \bar{x}_T converge to $x^{(\rho)}$ as

$$(5.4) \quad \mathbb{E} \left\| \bar{x}_T - x^{(\rho)} \right\|^2 \leq 2(1 - \alpha)^{T_b+1} \left\| x_0 - x^{(\rho)} \right\|^2 + \frac{4}{\alpha^2(T - T_b)} V.$$

with $V = \mathbb{E}_{S \sim \rho} \|A_S^\top (A_S A_S^\top + \lambda k I)^{-1} r_S^{(\rho)}\|^2$. Finally, the condition number $\kappa(\bar{W})$, the weighted residual $r^{(\rho)}$, the bias $x^{(\rho)} - x^*$, and the variance V are bounded as

$$(5.5) \quad \begin{aligned} \kappa(\bar{W}) &\leq 1 + \frac{1}{\lambda} \cdot \max_i \|a_i\|^2, \\ \|r^{(\rho)}\| &\leq \sqrt{\kappa(\bar{W})} \cdot \|r\|, \\ \|x^{(\rho)} - x^*\| &\leq \sqrt{\kappa(\bar{W}) - 1} \cdot \|A^+\| \cdot \|r\| \leq \frac{1}{\sqrt{\lambda}} \cdot \kappa(A) \cdot \|r\|, \\ V &\leq \frac{1}{4\lambda} \cdot \kappa(\bar{W}) \cdot \frac{\|r\|^2}{m}. \end{aligned}$$

Note that the values of $x^{(\rho)}$, α , and V here differ from those in Section 2 due to the different choice of $M(A_S)$ used by ReBlock. The advantage relative to RBK-U is that ReBlock-U is able to control the condition number of the weight matrix, the weighted residual, the bias of the weighted least-squares solution, and the variance of the iterates in terms of the reciprocal of the regularization parameter λ . As a result, the algorithm converges robustly even when the problem contains many nearly singular blocks A_S .

As a concrete example of the benefits of regularization, recall the isosceles triangle problem (3.4) for which the bias and variance of RBK-U both diverge as $\epsilon \rightarrow 0$. In contrast, for ReBlock-U we have the following corollary:

COROLLARY 5.2. *Consider the isosceles triangle problem (3.4) from Theorem 3.2. For ReBlock-U with any fixed $\lambda > 0$, the bounds (5.5) imply*

$$(5.6) \quad \lim_{\epsilon \rightarrow 0^+} \|r^{(\rho)}\| = \lim_{\epsilon \rightarrow 0^+} \|x^{(\rho)} - x^*\| = \lim_{\epsilon \rightarrow 0^+} V = 0$$

since the row norms a_i remain bounded and the residual norm $\|r\|$ vanishes as $\epsilon \rightarrow 0$.

In other words, introducing even a small amount of regularization prevents the algorithm from failing catastrophically. We confirm this effect numerically by directly calculating the bias for various values of ϵ and λ , with results in Figure 5. Note that for fixed ϵ , the bias decreases monotonically with λ , as expected.

It is worth noting that these advantages could also be attained by truncating the small singular values in the RBK iteration (1.5). However, the ReBlock iteration

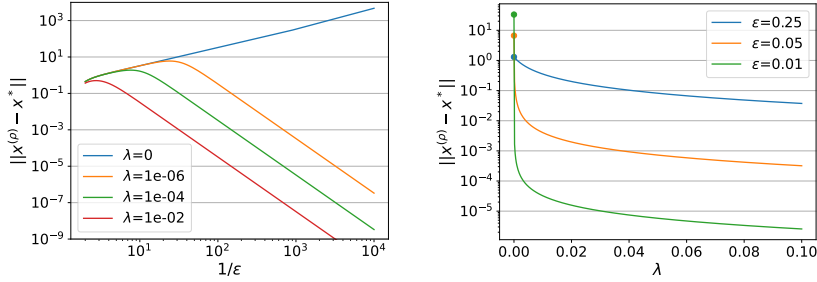


FIG. 5. Size of the bias $x^{(\rho)} - x^*$ for ReBlock-U with $k = 2$ on the isosceles triangle problem of Theorem 3.2, for various values of ϵ and λ . Left: for $\lambda = 0$ the bias grows without bound as $1/\epsilon \rightarrow \infty$, whereas for any $\lambda > 0$ the bias reaches a fixed maximum value and then decays to zero. Right: for any fixed ϵ the bias decreases monotonically with λ .

is additionally justified by the fact that it is cheaper to implement than the RBK iteration; see Subsection 5.2 and Figure 12 for further discussion of this point. The proof of Theorem 5.1 is provided in section C. Relative to the proof of Theorem 3.1, the proof of Theorem 5.1 is more complicated because the ReBlock iteration does not lead to an orthogonal decomposition of the error term $x_{t+1} - x^{(\rho)}$. Instead, the proof relies on a bias-variance decomposition inspired by [13, 24, 19], which also leads to the extra factors of 2 in the convergence bound.

We are not yet able to analyze the convergence rate parameter α of ReBlock-U, even in the case of Gaussian data, as to our knowledge there is no existing work bounding the quality of a regularized Gaussian sketch. However, a fast rate of convergence to the ordinary least-squares solution can be shown when sampling from an appropriate determinantal point process; see section E. Additionally, convergence to the ordinary least-squares solution is obtained for a broader class of *noisy* linear least-squares problems; see section D.

5.1. Numerical Demonstration. We now demonstrate that the benefits of regularization can translate to more realistic types of problems. To do so we revisit the examples from Figure 2, for which RBK-U exhibited severe instabilities. We observe in Figure 6 that ReBlock-U with $\lambda = 1e - 3$ is stable for both of these problems and converges much faster than TA-mSGD in the presence of rapid singular value decay. To understand the impact of the inconsistency on the algorithms, we re-run the case with rapid singular value decay with varying levels of inconsistency, with results in Figure 7. We find qualitatively similar results in every case. Further details on these experiments can be found in Section F.

5.2. Implementation Details. To implement the ReBlock iterations (1.7), we directly calculate the $k \times k$ matrix $A_{S_t} A_{S_t}^\top + \lambda k I$ and then use a Cholesky-based linear system solver to calculate $(A_{S_t} A_{S_t}^\top + \lambda k I)^{-1} (b_{S_t} - A_{S_t} x)$. This computation is stable as long as λ is not chosen to be too small. Using this approach, the most expensive part of the ReBlock iteration is calculating $A_{S_t} A_{S_t}^\top$, which has an asymptotic cost of $O(nk^2)$ just like RBK. Nonetheless, in practice the matrix-matrix multiplication for ReBlock can have a much smaller preconstant than the QR decomposition used for RBK. For example, in the experiments of Section 6, ReBlock iterations are over twenty-five times faster than RBK iterations, as reported in Figure 12. We emphasize that this gap is not related to the use of the Cholesky decomposition but instead arises because ReBlock avoids doing any matrix decomposition at all directly on the

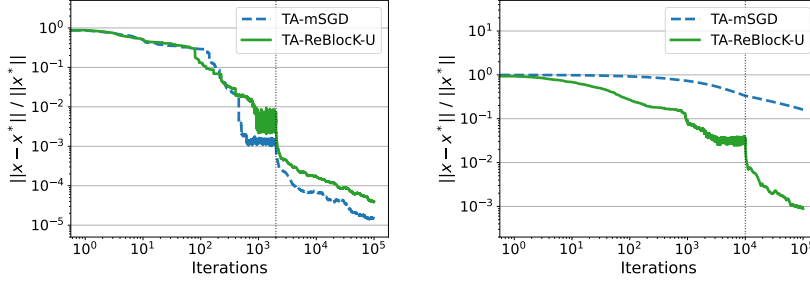


FIG. 6. Performance of ReBlock-U for the same two inconsistent problems from Figure 2. These problems both contain many nearly singular blocks, with mild singular value decay (left) and rapid singular value decay (right). The vertical dotted line indicates the burn-in time, before which results are shown for individual iterates.

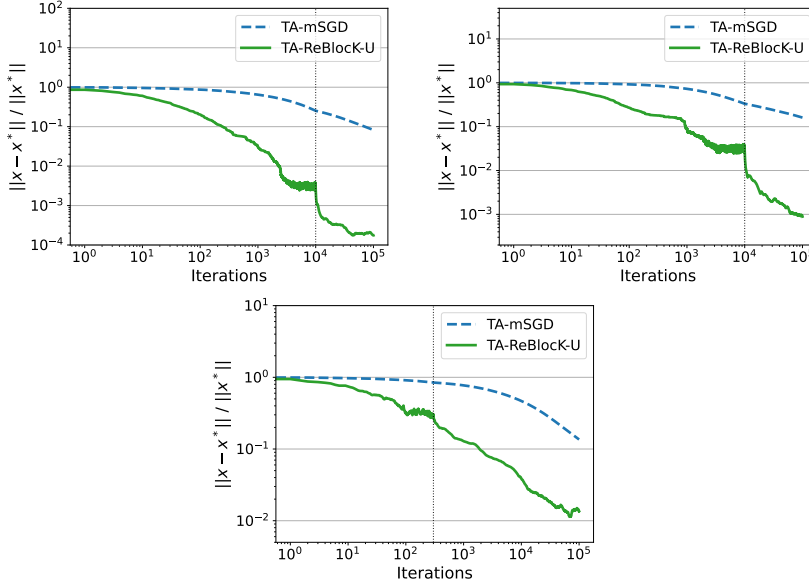


FIG. 7. Exploration of the effect of the inconsistency on the behavior of the algorithms. The variance of each entry of the noise vector is set to $1e-6$ (top left), $1e-4$ (top right), and $1e-2$ (bottom). The top right panel here is identical to the right panel of Figure 6, and is reproduced here only for comparison with the top left and bottom.

$k \times n$ matrix A_S . For the largest problems, the ReBlock iterations could be further accelerated using iterative solvers; see for example Section 4.1 of [15].

6. Natural Gradient Optimization. Our original motivation for this work comes from the problem of training deep neural networks using natural gradient descent [1], which is based on an efficient natural gradient induced by a problem-dependent Riemannian metric. Natural gradient descent has been studied extensively in the machine learning community; see [27, 37, 26]. Furthermore, there is increasing evidence that natural gradient methods can improve the accuracy when training neural networks to solve physical equations. See [28, 10] for applications to physics-informed neural networks and [34, 39] for applications to neural network wavefunctions.

To elucidate the structure of the natural gradient direction, consider the function learning problem

$$(6.1) \quad \min_{\theta} L(\theta), \quad L(\theta) := \frac{1}{2} \int_{\Omega} (f_{\theta}(s) - f(s))^2 ds,$$

where $\Omega \subset \mathbb{R}^d$ is the domain of the functions, $f : \Omega \rightarrow \mathbb{R}$ is the target function and $f_{\theta} : \Omega \rightarrow \mathbb{R}$ is a function represented by a neural network with parameters $\theta \in \mathbb{R}^n$. The standard definition of natural gradient descent for this problem is

$$(6.2) \quad \theta \leftarrow \theta - \eta G_N, \quad G_N := F^{-1} \nabla_{\theta} L(\theta),$$

where η is the step size, F is the Fisher information matrix

$$(6.3) \quad F = \int_{\Omega} \nabla_{\theta} f_{\theta}(s) \nabla_{\theta} f_{\theta}(s)^{\top} ds = J^{\top} J,$$

and the Euclidean gradient $\nabla_{\theta} L(\theta)$ takes the form

$$(6.4) \quad \nabla_{\theta} L(\theta) = \int_{\Omega} \nabla_{\theta} f_{\theta}(s) (f_{\theta}(s) - f(s)) ds = J^{\top} [f_{\theta} - f].$$

Here $J : \mathbb{R}^n \rightarrow \mathbb{R}^{\Omega}$ represents the Jacobian, which is a linear operator from the space of parameters to the space of real-valued functions on Ω . J^{\top} represents the adjoint.

Calculating the natural gradient direction G_N using (6.2) requires a linear solve against the $n \times n$ matrix F , which is very challenging in realistic settings when $n \geq 10^6$. This has motivated the development of approximate schemes such as [27]. An alternative approach is to reformulate G_N using the structure of F and $\nabla_{\theta} L(\theta)$:

$$(6.5) \quad G_N = (J^{\top} J)^{-1} J^{\top} [f_{\theta} - f]$$

$$(6.6) \quad = \operatorname{argmin}_{x \in \mathbb{R}^n} \|Jx - [f_{\theta} - f]\|^2,$$

where the norm in the final expression is the L_2 -norm in function space. This least-squares formulation has been pointed out for example by [26, 8, 20], with the work of Chen and Heyl empowering major advances in the field of neural quantum states. A major goal of the current work is to provide a more solid foundation for the development of natural gradient approximations along these lines.

The natural way to access the data when training a neural network is to sample a set of points in the domain Ω and evaluate the target function f , the network outputs f_{θ} , and the network gradients $\nabla_{\theta} f_{\theta}$ at the sampled points. This is precisely equivalent to sampling a small subset of the rows of (6.6), which motivates the consideration of row-access least-squares solvers for calculating natural gradient directions. Furthermore, both empirical evidence from scientific applications [32, 43] and theoretical evidence from the literature on neural tangent kernels [4, 38, 6] suggest that J should be expected to exhibit fast singular value decay, motivating the possibility that RBK and ReBloK could converge rapidly when solving (6.6). Note that these observations translate straightforwardly from the simple function learning problem (6.1) to the realistic problems of training physics-informed neural networks or neural network wavefunctions.

It is also worth noting that while we focus on the problem of calculating natural gradient directions, similar linear systems also arise when using neural networks to simulate the time evolution of either differential equations [5] or quantum systems [40]. When simulating time dynamics it is essential to solve each linear system to high accuracy. This provides additional motivation for developing block row access methods that can solve such linear systems accurately without preprocessing.

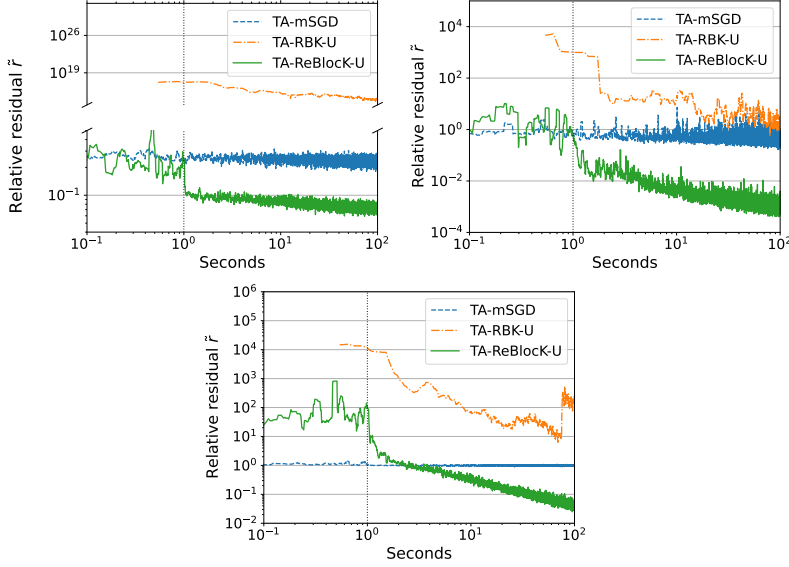


FIG. 8. Comparison of methods for calculating natural gradient directions for a small neural network. The network parameters θ are taken from three snapshots of a single training run, with one snapshot from the “pre-descent” phase before the loss begins to decrease (top left), one snapshot from the “descent” phase during which the loss decreases rapidly (top right), and one snapshot from the “post-descent” phase when the decay rate of the loss has slowed significantly (bottom). The algorithms are measured in terms of their progress towards reducing the relative residual $\tilde{r} = \|Jx - [f_\theta - f]\| / \|f_\theta - f\|$ for the least-squares problem (6.6), which measures how well the function-space update direction Jx agrees with the function-space loss gradient $f_\theta - f$. The burn-in time is set to $T_b \approx T/100$ in each case, as indicated by the vertical dotted line.

6.1. Numerical Demonstration. To test our algorithms for calculating natural gradient directions, we train a neural network with about 7.5×10^5 parameters to learn a rapidly oscillating function on the unit interval. We take three snapshots from the training process and form the least-squares problem (6.6) for each. The methods TA-mSGD, TA-RBK-U, and TA-ReBlock-U are then compared on these problems with results in Figure 8. The computations are performed on an A100 GPU to simulate a deep learning setting and in double precision to enable each algorithm’s iterations to be computed accurately, and progress is measured with wall-clock time on the horizontal axis. TA-ReBlock-U performs best in every case. Interestingly, RBK-U fails catastrophically on the first snapshot and also exhibits instabilities in the last snapshot, providing further evidence that the issues highlighted by the no-go Theorem 3.2 are not merely of theoretical interest. We also provide per iteration convergence plots in Figure 9. In the first snapshot the catastrophic failure of TA-RBK-U persists while in the second and third snapshots, we find that TA-RBK-U is more competitive on a per iteration basis, but is still outperformed by TA-ReBlock-U.

In Figure 10, we investigate the effect of the parameter λ on the performance of ReBlock. We observe that choosing $\lambda = 1$ already provides an advantage relative to mSGD, and the advantage is further improved by reducing λ down to $1e-3$ or $1e-6$. However, when choosing $\lambda = 1e-9$ the variance becomes so large as to significantly hinder the performance of the algorithm. These results suggest that the performance of ReBlock is optimized by choosing λ as small as possible without introducing significant instabilities. Next, in Figure 11 we explore the effect of the batch size k on the performance of the algorithms. We observe that ReBlock performs

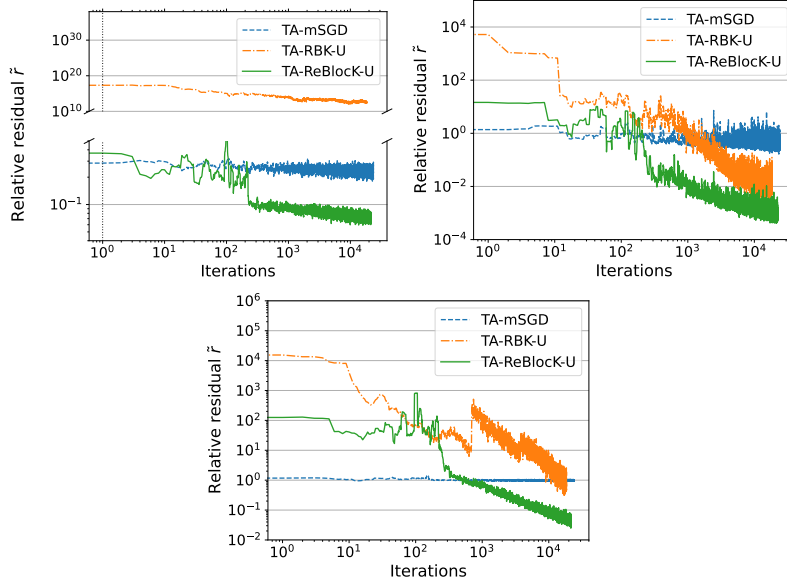


FIG. 9. *Per iteration plots for the same three snapshots from Figure 8, with TA-RBK-U run for much longer than the other methods to produce a comparable number of iterations. The algorithms are measured in terms of their progress towards reducing the relative residual $\tilde{r} = \|Jx - [f_\theta - f]\| / \|f_\theta - f\|$ for the least-squares problem (6.6), which measures how well the function-space update direction Jx agrees with the function-space loss gradient $f_\theta - f$. The burn-in period is set based on seconds rather than iterations and so is not indicated in this plot.*

best across all batch sizes. Finally, in Figure 12, we report the number of iterations per second for mSGD, RBK, and ReBlock for the experiments of this section. We find that the ReBlock iterations are over twenty-five times faster than the RBK iterations and only about 10% slower than the mSGD iterations, which provides a significant boost to TA-ReBlock-U even beyond its improved stability. Further details on these experiments can be found in Section G.

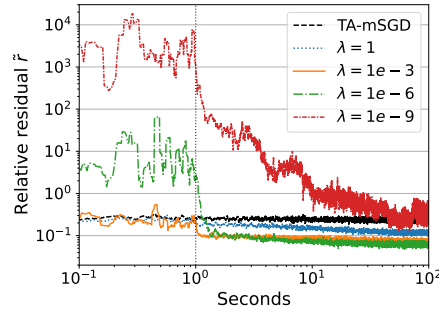


FIG. 10. *Performance of TA-ReBlock-U on the problem from the top left panel of Figure 8 with various values of λ .*

Our results, though preliminary due to their synthetic nature, suggest that ReBlock is a promising method for calculating natural gradient directions. This provides justification for the Kaczmarz-inspired SPRING algorithm [20] and suggests a fam-

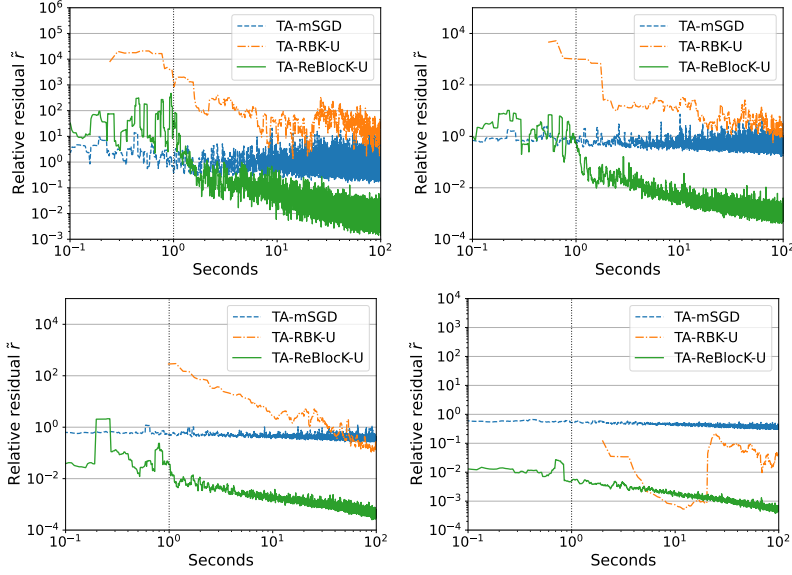


FIG. 11. Performance of algorithms for different values of the block size k . The problem is the same as in the top right panel of Figure 8 and the values of k are $k = 25$ (top left), $k = 50$ (top right), $k = 100$ (bottom left), and $k = 200$ (bottom right). The top right panel here is identical to the top right panel of Figure 6, and is reproduced here only for comparison with the other quadrants.

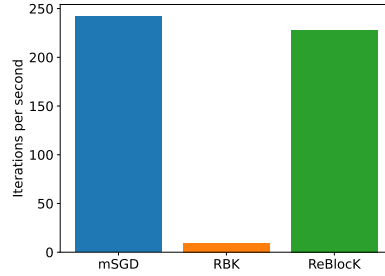


FIG. 12. Iteration speed of each algorithm for the natural gradient experiments with $k = 50$.

ily of related methods to be explored by future works. Open questions include how many ReBlock iterations to run between each update of θ , whether to incorporate averaging, and how to tune λ .

7. Conclusions. In this work, we have explored the problem of solving large-scale linear least-squares problems without preprocessing the data matrix A . Our results suggest that ReBlock is a more effective algorithm than either RBK or mSGD for inconsistent problems that exhibit rapid singular value decay. More broadly, our work highlights the value of incorporating regularization as a path towards broadening the applicability of block Kaczmarz methods, and our analysis suggests that a Monte Carlo perspective can be useful in elucidating the behavior of randomized block row-access methods in general. Finally, our work provides motivation and suggests new directions for both natural gradient optimizers and time stepping methods in scientific machine learning.

Acknowledgements. We thank Ethan Epperly, Robert Webber, Marius Zeinhofer, Ruizhe Zhang, and the anonymous reviewers for their thoughtful discussions and feedback.

Appendix A. Helpful Lemmas. In this appendix we prove five lemmas which will help us analyze both RBK and ReBLoK. Lemma A.1 shows how the expectation of the iterates evolves. Lemma A.2 shows that the iterates of the algorithms always stay within $\text{range}(A^\top)$, and Lemma A.3 provides a contraction property of applying $I - \bar{P}$ to certain types of vectors. Lemma A.4 shows how to derive bounds on the expected square error of tail-averaged iterates under relevant assumptions. Finally, Lemma A.5 bounds the weighted residual and the bias based on the condition number of \bar{W} .

LEMMA A.1 (Convergence of single-iterate expectation). *Consider the generalized iteration (2.1) with some fixed choice of a positive semidefinite mass matrix $M(A_S)$ and sampling distribution ρ , and fix two indices $r < s$. Then the expectation of x_s conditioned on x_r satisfies*

$$(A.1) \quad \mathbb{E} [x_s - x^{(\rho)} | x_r] = (I - \bar{P})^{s-r} (x_r - x^{(\rho)}).$$

Proof of Lemma A.1. Fix $t \in r, \dots, s-1$, let $P_t = P(S_t)$ and $W_t = W(S_t)$. Recall from (2.6) that the generalized iteration (2.1) can be reformulated as

$$(A.2) \quad x_{t+1} - x^{(\rho)} = (I - P_t)(x_t - x^{(\rho)}) + A^\top W_t r^{(\rho)}.$$

Recall also that the normal equations for $x^{(\rho)}$ imply that $A^\top \bar{W} r^{(\rho)} = 0$. Taking the expectation over the choice of S_t , it thus obtains

$$(A.3) \quad \mathbb{E} [x_{t+1} - x^{(\rho)} | x_t] = (I - \bar{P})(x_t - x^{(\rho)}) + A^\top \bar{W} r^{(\rho)} = (I - \bar{P})(x_t - x^{(\rho)}).$$

Using the law of total expectation and the linearity of expectation, iterating this result for $t = r, \dots, s-1$ yields the lemma. \square

LEMMA A.2. *Consider the generalized iteration (2.1) with some fixed choice of a positive semidefinite mass matrix $M(A_S)$ and sampling distribution ρ . Then $x^*, x^{(\rho)} \in \text{range}(A^\top)$ and if $x_0 \in \text{range}(A^\top)$, then $x_t \in \text{range}(A^\top)$ for all $t \geq 0$.*

Proof. Recall that x^* is the minimal-norm solution to (1.1) and $x^{(\rho)}$ is the minimal-norm solution to (2.5). Suppose for the sake of contradiction that $x^* \notin \text{range}(A^\top)$. Then let \tilde{x} be the projection of x^* onto $\text{range}(A^\top)$. It follows that $\|\tilde{x}\| < \|x^*\|$ and $Ax^* = A\tilde{x}$, making \tilde{x} a minimizer of (1.1) with a smaller norm than x^* (contradiction). The same argument holds for $x^{(\rho)}$.

To show $x_t \in \text{range}(A^\top)$ for all $t \geq 0$ we apply an inductive argument. The claim holds for $t = 0$ by assumption, so now suppose that it holds for some arbitrary $t \geq 0$. Recall

$$(A.4) \quad x_{t+1} = x_t + A_{S_t}^\top M(A_{S_t})(b_{S_t} - A_{S_t} x_t). \quad \square$$

Then $x_t \in \text{range}(A^\top)$ by assumption and $A_{S_t}^\top M(A_{S_t})(b_{S_t} - A_{S_t} x_t) \in \text{range}(A^\top)$ since $\text{range}(A_{S_t}^\top) \subseteq \text{range}(A^\top)$. It follows that $x_{t+1} \in \text{range}(A^\top)$, completing the proof.

LEMMA A.3. *Suppose that $x \in \text{range}(\bar{P})$, $\bar{P} \preceq I$, and $\alpha = \sigma_{\min}^+(\bar{P})$. Then for any $s \geq 0$,*

$$(A.5) \quad x^\top (I - \bar{P})^s x \leq (1 - \alpha)^s \|x\|^2$$

and

$$(A.6) \quad \|(I - \bar{P})^s x\| \leq (1 - \alpha)^s \|x\|.$$

Proof. First expand x in the basis of eigenvectors of the symmetric matrix \bar{P} , then note that every eigenvector that has a nonzero coefficient in the expansion has its eigenvalue in the interval $[\alpha, 1]$.

LEMMA A.4 (Convergence of tail-averaged schemes). *Consider the generalized iteration (2.1) with some fixed mass matrix $M(A_S)$ and sampling distribution ρ . Let $\alpha = \sigma_{\min}^+(\bar{P})$ and suppose that $x_0 \in \text{range}(A^\top)$, $\text{range}(\bar{P}) = \text{range}(A^\top)$, and $\bar{P} \preceq I$. Additionally, suppose the stochastic iterates x_0, x_1, \dots satisfy*

$$(A.7) \quad \mathbb{E} \|x_t - x^{(\rho)}\|^2 \leq (1 - \alpha)^t B + \tilde{V}$$

for all t and some constants B, \tilde{V} , where we use \tilde{V} to distinguish from the variance term V in Theorems 3.1 and 5.1. Then the tail averages \bar{x}_T of the iterates, with burn-in time T_b , satisfy

$$(A.8) \quad \mathbb{E} \|\bar{x}_T - x^{(\rho)}\|^2 \leq (1 - \alpha)^{T_b+1} B + \frac{2}{\alpha(T - T_b)} \tilde{V}.$$

Proof. We follow closely the Proof of Theorems 1.2 and 1.3 of [19]. Decompose the expected mean square error as

$$(A.9) \quad \mathbb{E} \|\bar{x}_T - x^{(\rho)}\|^2 = \frac{1}{(T - T_b)^2} \sum_{r,s=T_b+1}^T \mathbb{E} \left[(x_r - x^{(\rho)})^\top (x_s - x^{(\rho)}) \right].$$

For $T_b \leq r < s$ bound the covariance term using Lemma A.1 and our assumptions on \bar{P} :

$$\begin{aligned} \mathbb{E} \left[(x_r - x^{(\rho)})^\top (x_s - x^{(\rho)}) \right] &= \mathbb{E} \left[(x_r - x^{(\rho)})^\top \mathbb{E} [x_s - x^{(\rho)} | x_r] \right] \\ &= \mathbb{E} \left[(x_r - x^{(\rho)})^\top (I - \bar{P})^{s-r} (x_r - x^{(\rho)}) \right] \\ &\leq (1 - \alpha)^{s-r} \mathbb{E} \|x_r - x^{(\rho)}\|^2 \\ &\leq (1 - \alpha)^s B + (1 - \alpha)^{s-r} \tilde{V}, \end{aligned}$$

where the third line uses Lemmas A.2 and A.3 and the assumptions $\text{range } \bar{P} = \text{range}(A^\top)$ and $\bar{P} \preceq I$.

Using the coarse bound $(1 - \alpha)^s B \leq (1 - \alpha)^{T_b+1} B$ for $s \geq T_b + 1$, it follows

$$(A.10) \quad \mathbb{E} \|\bar{x}_T - x^{(\rho)}\|^2 \leq (1 - \alpha)^{T_b+1} B + \frac{\tilde{V}}{(T - T_b)^2} \sum_{r,s=T_b+1}^T (1 - \alpha)^{|s-r|}.$$

Apply another coarse bound

$$(A.11) \quad \sum_{r,s=T_b+1}^T (1 - \alpha)^{|s-r|} \leq 2 \sum_{r=T_b+1}^T \sum_{s=0}^{\infty} (1 - \alpha)^s = \frac{2(T - T_b)}{\alpha}$$

to obtain the final result:

$$(A.12) \quad \mathbb{E} \|\bar{x}_T - x^{(\rho)}\|^2 \leq (1 - \alpha)^{T_b+1} B + \frac{2}{\alpha(T - T_b)} \tilde{V}. \quad \square$$

LEMMA A.5 (Residual and bias bounds). *Assuming $\overline{W} \succ 0$, it holds*

$$(A.13) \quad \begin{aligned} \|r^{(\rho)}\|^2 &\leq \kappa(\overline{W}) \cdot \|r\|^2, \\ \|x^{(\rho)} - x^*\| &\leq \sqrt{\kappa(\overline{W}) - 1} \cdot \|A^+\| \cdot \|r\|. \end{aligned}$$

Proof. To bound the weighted residual, use operator norms and the optimality of $x^{(\rho)}$ for the weighted least-squares problem:

$$\begin{aligned} \|Ax^{(\rho)} - b\|^2 &\leq \|\overline{W}^{-1}\| \|Ax^{(\rho)} - b\|_{\overline{W}}^2 \\ &\leq \|\overline{W}^{-1}\| \|Ax^* - b\|_{\overline{W}}^2 \\ &\leq \|W^{-1}\| \|\overline{W}\| \|r\|^2 \\ &= \kappa(\overline{W}) \|r\|^2. \end{aligned}$$

Next, note that the weighted residual admits an orthogonal decomposition $r^{(\rho)} = A(x^* - x^{(\rho)}) + r$. It follows that

$$(A.14) \quad \|A(x^{(\rho)} - x^*)\|^2 = \|r^{(\rho)}\|^2 - \|r\|^2 \leq (\kappa(\overline{W}) - 1) \cdot \|r\|^2.$$

Taking square-roots and using $\|x^{(\rho)} - x^*\| \leq \|A^+\| \cdot \|A(x^{(\rho)} - x^*)\|$ yields the bias bound

$$(A.15) \quad \|x^{(\rho)} - x^*\| \leq \sqrt{\kappa(\overline{W}) - 1} \cdot \|A^+\| \cdot \|r\|. \quad \square$$

Appendix B. Proofs of RBK Convergence Theorems. In this section we provide the proofs of Theorems 3.1 and 4.1 and Corollary 4.2. We first prove the following lemma, which can be viewed as a more general version of Theorem 1.2 of [30]:

LEMMA B.1. *Consider the RBK iterates (1.5) and fix some sampling distribution ρ . Suppose that $x_0 \in \text{range}(A^\top)$ and $\text{range}(\overline{P}) = \text{range}(A^\top)$. Then for all t it holds*

$$(B.1) \quad \mathbb{E} \|x_t - x^{(\rho)}\|^2 \leq (1 - \alpha)^t \|x_0 - x^{(\rho)}\|^2 + \frac{1}{\alpha} \mathbb{E}_{S \sim \rho} \|A_S^+ r_S^{(\rho)}\|^2.$$

Proof. Let $P_s = P(S_s)$ and $W_s = W(S_s)$. Using (2.6) and the definition $M(A_S) = (A_S A_S^\top)^+$, the RBK iteration (1.5) can be reformulated as

$$(B.2) \quad x_{s+1} - x^{(\rho)} = (I - P_s)(x_s - x^{(\rho)}) + A_{S_s}^+ r_{S_s}^{(\rho)}.$$

This represents an orthogonal decomposition of $x_{s+1} - x^{(\rho)}$ since $I - P_s$ is the projector onto the null space of A_{S_s} , which is in turn orthogonal to the range of the pseudoinverse $A_{S_s}^+$.

It thus holds

$$\begin{aligned} \|x_{s+1} - x^{(\rho)}\|^2 &= \|(I - P_s)(x_s - x^{(\rho)})\|^2 + \|A_{S_s}^+ r_{S_s}^{(\rho)}\|^2 \\ &= (x_s - x^{(\rho)})^\top (I - P_s)(x_s - x^{(\rho)}) + \|A_{S_s}^+ r_{S_s}^{(\rho)}\|^2, \end{aligned}$$

where the second line uses the idempotency of the projector $I - P_s$. Note that $x_s - x^{(\rho)} \in \text{range}(A^\top)$ by Lemma A.2 and $\bar{P} \preceq I$ since $P(S)$ is always a projection matrix. Taking expectations and applying Lemma A.3 thus yields

$$\begin{aligned} \mathbb{E} \left\| x_{s+1} - x^{(\rho)} \right\|^2 &= (x_s - x^{(\rho)})^\top (I - \bar{P})(x_s - x^{(\rho)}) + \mathbb{E}_{S \sim \rho} \left\| A_S^+ r_S^{(\rho)} \right\|^2 \\ &\leq (1 - \alpha) \mathbb{E} \left\| x_s - x^{(\rho)} \right\|^2 + \mathbb{E}_{S \sim \rho} \left\| A_S^+ r_S^{(\rho)} \right\|^2. \end{aligned}$$

Iterating from $s = 0$ to $s = t - 1$ and utilizing $\sum_{s=0}^{t-1} (1 - \alpha)^s < \sum_{s=0}^{\infty} (1 - \alpha)^s = 1/\alpha$ yields the desired result. \square

Proof of Theorem 3.1. The result follows from the appropriate application of Lemmas A.1, A.4, and B.1. To apply these lemmas it is first required to verify $\bar{P} \preceq I$ and $\text{range}(\bar{P}) = \text{range}(A^\top)$. For the first result, it suffices to observe that $P(S)$ is the orthogonal projector onto the row space of A_S and thus $P(S) \preceq I$ for all S .

For the second result, first note that since $\bar{P} = A^\top \bar{W} A$ it is clear that $\text{range}(\bar{P}) \subseteq \text{range}(A^\top)$. Utilizing this containment and the fact that \bar{P} is symmetric, it suffices to show that there is no $x \in \text{range}(A^\top)$ such that $\bar{P}x = 0$. Now, consider any such $x \in \text{range}(A^\top)$. There must exist a row index i such that $a_i^\top x \neq 0$. Thus

$$(B.3) \quad x^\top \bar{P}x = \mathbb{E}_{S \sim \mathbf{U}(m,k)} [x^\top A_S^\top (A_S A_S^\top)^{-1} A_S x] \geq \frac{k}{m} \cdot \frac{(a_i^\top x)^2}{\|a_i\|^2} > 0,$$

where the intermediate step uses the facts that row i is chosen with probability k/m and that $A_S^\top (A_S A_S^\top)^{-1} A_S \succeq a_i (a_i^\top a_i)^{-1} a_i^\top$ when $i \in S$. It follows that $\bar{P}x \neq 0$ and so indeed $\text{range}(\bar{P}) = \text{range}(A^\top)$.

With the assumptions verified, Lemma A.1 can be applied using $r = 0$, $s = T$ to yield

$$(B.4) \quad \mathbb{E} [x_T] - x^{(\rho)} = (I - \bar{P})^T (x_0 - x^{(\rho)}).$$

This directly implies

$$(B.5) \quad \left\| \mathbb{E} [x_T] - x^{(\rho)} \right\| \leq (1 - \alpha)^T \left\| x_0 - x^{(\rho)} \right\|$$

using Lemmas A.2 and A.3 and the proven properties of \bar{P} .

Furthermore, Lemma B.1 holds and provides the conditions for Lemma A.4 with $B = \left\| x_0 - x^{(\rho)} \right\|^2$, $\tilde{V} = \frac{1}{\alpha} \mathbb{E}_{S \sim \rho} \left\| A_S^+ r_S^{(\rho)} \right\|^2$. The application of Lemma A.4 then directly implies the convergence result for the tail-averaged RBK-U algorithm.

Now, we proceed with upper bounds on condition number, the weighted residual, the bias, and the variance. For the condition number, first note that

$$(B.6) \quad \|A_S A_S^\top\| = \|A_S^\top A_S\| = \left\| \sum_{i \in S} a_i a_i^\top \right\| \leq \sum_{i \in S} \|a_i a_i^\top\| \leq k \cdot \max_i \|a_i\|^2.$$

Since we have assumed the rows of A are in general position, this implies $\|(A_S A_S^\top)^+\| \geq 1/(k \cdot \max_i \|a_i\|^2)$. Since $\bar{W} = \mathbb{E}_S [I_S^\top (A_S A_S^\top)^+ I_S]$ follows that

$$\frac{1}{k \cdot \max_i \|a_i\|^2} \cdot \mathbb{E} [I_S^\top I_S] \preceq \bar{W} \preceq \max_S \|A_S^+\|^2 \cdot \mathbb{E} [I_S^\top I_S].$$

When S is sampled uniformly, $\mathbb{E}_{S \sim \mathcal{U}(m,k)} [I_S^\top I_S] = \frac{k}{m} I$ and so this implies

$$\kappa(\overline{W}) \leq k \cdot \left(\max_i \|a_i\|^2 \right) \cdot \max_S \|A_S^+\|^2.$$

The bounds on the weighted residual and the bias follow directly from Lemma A.5. Finally, regarding the variance, it follows from the definition of V that

$$\begin{aligned} (B.7) \quad V &= \mathbb{E}_S \|A_S^+ r_S^{(\rho)}\|^2 \leq \max_S \|A_S^+\|^2 \cdot \mathbb{E}_S \|r_S^{(\rho)}\|^2 \\ &= \max_S \|A_S^+\|^2 \cdot \frac{k \|r^{(\rho)}\|^2}{m}. \end{aligned}$$

Applying the weighted residual bound from Lemma A.5 the desired result. \square

Proof of Theorem 4.1. In the case of Gaussian data, we consider the statistical linear regression problem

$$(B.8) \quad x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} \mathbb{E}_{[a^\top \ b] \sim \mathcal{N}(0, Q)} [(a^\top x - b)^2].$$

The RBK-U algorithm is applied by generating k independent samples per iteration from the data distribution $\mathcal{N}(0, Q)$. Denote by $Q = LL^\top$ the Cholesky decomposition of Q . Let $L = \begin{bmatrix} L_a \\ \ell_b^\top \end{bmatrix}$ so L_a represents the first n rows of L and ℓ_b^\top represents the last row of L . Due to the lower triangular nature of L it follows that $L_a = [L_n \ 0]$ where L_n is the $n \times n$ Cholesky factor of Q_n (which we previously defined to be the top left $n \times n$ block of Q).

In each iteration of RBK-U, the k data points $[a_1^\top \ b_1], \dots, [a_k^\top \ b_k]$ from $\mathcal{N}(0, Q)$ can be equivalently redistributed as $z_1^\top [L_a^\top \ \ell_b], \dots, z_k^\top [L_a^\top \ \ell_b]$ for $z_i \sim \mathcal{N}(0, I_{n+1})$. Collecting the random vectors z_1, \dots, z_k into the columns of a single matrix Z_t , the RBK-U update can be written as

$$(B.9) \quad x_{t+1} = x_t + (Z_t^\top L_a^\top)^+ (Z_t^\top \ell_b - Z_t^\top L_a^\top x_t).$$

Now, note that $\mathbb{E}[aa^\top] = L_a L_a^\top$, $\mathbb{E}[ba] = L_a \ell_b$ and $\mathbb{E}[b^2] = \ell_b^\top \ell_b$. Plugging these identities into the definition of x^* , we find

$$(B.10) \quad x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} \mathbb{E}_{[a^\top \ b] \sim \mathcal{N}(0, Q)} [(a^\top x - b)^2] = \operatorname{argmin}_{x \in \mathbb{R}^n} \|L_a^\top x - \ell_b\|^2.$$

We can thus define an ‘‘underlying’’ residual vector by $\tilde{r} = \ell_b - L_a^\top x^*$. Additionally defining $P_t = (Z_t^\top L_a^\top)^+ Z_t^\top L_a^\top$, the update (B.9) can be reformulated as

$$(B.11) \quad x_{t+1} - x^* = (I - P_t)(x_t - x^*) + (Z_t^\top L_a^\top)^+ Z_t^\top \tilde{r}.$$

Now, decompose $Z_t^\top = [Z_t^1 \ Z_t^2]$ where Z_t^1 consists of the first n columns and Z_t^2 of the last column. Note that Z_t^1, Z_t^2 are independent mean-zero Gaussian matrices and note also that since $L_a = [L_n \ 0]$ it holds $Z_t^\top L_a^\top = Z_t^1 L_n^\top$. Furthermore since L_n is full rank, $\operatorname{range}(L_a^\top) = \operatorname{span}(e_1, \dots, e_n)$ and thus $\tilde{r} \parallel e_{n+1}$ (here e_i is the i th standard basis vector in \mathbb{R}^{n+1}). It follows that $Z_t^\top \tilde{r} = Z_t^2 \tilde{r}_{n+1}$ and so

$$\mathbb{E} [(Z_t^\top L_a^\top)^+ (Z_t^\top \tilde{r})] = \mathbb{E} [(Z_t^1 L_n^\top)^+ Z_t^2 \tilde{r}_{n+1}] = \mathbb{E} [(Z_t^1 L_n^\top)^+] \mathbb{E} [Z_t^2 \tilde{r}_{n+1}] = 0$$

since $\mathbb{E}[Z_t^2] = 0$. We note that a similar independence lemma has also been established in Lemma 3.14 of [36]. Furthermore, taking the expectation on both sides of (B.11), we have

$$\mathbb{E}[x_{t+1} - x^*] = (I - \bar{P})(x_t - x^*),$$

where $\bar{P} := \mathbb{E}[P_t]$. Consequently, $\mathbb{E}[x_t]$ converges to x^* using the same logic as the proof of Theorem 3.1, which proves that $x^{(\rho)} = x^*$.

Now, consider the bound on the value of α . Note that the entries of $Z_t \in \mathbb{R}^{(n+1) \times k}$ are independently drawn from the standard Gaussian distribution. This property enables us to leverage existing results from [16] concerning the spectrum of the matrix \bar{P} . Specifically, $P_t = (Z_t L_a^\top)^\top + Z_t L_a^\top$, which has a similar formula to the matrix P considered in Equation (6) of [16]. Then, noting that $L_a = [L_n \ 0]$ has the same singular value spectrum as L_n , applying Theorem 3.1 of [16] gives the result stated in Theorem 4.1.

It remains to show the bound on the variance. Note that the variance term $\|A_{S_t}^\top r_{S_t}\|^2$ in Theorem 3.1 here takes the form

$$(B.12) \quad \|(Z_t^\top L_a^\top)^\top + Z_t^\top \tilde{r}\|^2.$$

We bound

$$(B.13) \quad \|(Z_t^\top L_a^\top)^\top + Z_t^\top \tilde{r}\|^2 = \|(Z_t^\top L_n^\top)^\top + Z_t^\top \tilde{r}_{n+1}\|^2 \leq \sigma_{\min}^{-2}(Z_t^\top L_n^\top) \|Z_t^\top \tilde{r}_{n+1}\|^2.$$

We can bound the expectation of the smallest singular value of $Z_t^\top L_n^\top$ using a similar technique to the proof of Lemma 22 in [16]. To start, let $L_n^\top = W \Sigma Y^\top$ be the SVD of L_n^\top . Then

$$(B.14) \quad \sigma_{\min}^2(Z_t^\top L_n^\top) = \sigma_{\min}(Z_t^\top L_n^\top L_n (Z_t^\top)^\top) = \sigma_{\min}(Z_t^\top W \Sigma^2 W^\top (Z_t^\top)^\top).$$

Let W_{2k} denote the first $2k$ columns of W and note that $\Sigma \succeq \text{diag}(\sigma_{2k}, \dots, \sigma_{2k}, 0, \dots, 0)$ where we recall that σ_i is the i th singular value of L_n . It thus holds

$$\sigma_{\min}^2(Z_t^\top L_n^\top) \geq \sigma_{2k}^2 \cdot \sigma_{\min}(Z_t^\top W_{2k} W_{2k}^\top (Z_t^\top)^\top) = \sigma_{\min}^2(Z_t^\top W_{2k}).$$

Since the columns of W_{2k} are orthonormal the random matrix $Z_t^\top W_{2k}$ can be redistributed as a single Gaussian random matrix G_{2k} of size $k \times (2k)$ and with standard normal entries. By Lemma 3.16 of [36], we thus have for $k \geq 6$,

$$(B.15) \quad \mathbb{E}[\sigma_{\min}^{-2}(Z_t^\top L_n^\top)] \leq \frac{20}{(\sqrt{2k} - \sqrt{k})^2 \sigma_{2k}^2} \leq \frac{200}{k \sigma_{2k}^2}.$$

Hence, by the independence between Z_t^1 and Z_t^2 we have

$$(B.16) \quad \mathbb{E} \|(Z_t^\top L_a^\top)^\top + Z_t^\top \tilde{r}\|^2 = \mathbb{E}[\sigma_{\min}^{-2}(Z_t^\top L_n^\top)] \mathbb{E} \|Z_t^2 \tilde{r}_{n+1}\|^2 \leq \frac{200}{\sigma_{2k}^2} \cdot \|\tilde{r}\|^2.$$

Noting that $\mathbb{E}_{[a^\top \ b] \sim \mathcal{N}(0, Q)} [(a^\top x^* - b)^2] = \|L_a^\top x^* - \ell_b\|^2 = \|\tilde{r}\|^2$, we prove (4.3). \square

Proof of Corollary 4.2. By Corollary 3.4 of [16], if L_n has the polynomial spectral decay of order $\beta > 1$, i.e., $\sigma_i^2 \leq i^{-\beta} \sigma_1^2$ for all i , the dependence of $C_{n,k}$ on n and k in Theorem 4.1 can be eliminated when $k \leq n/2$. Furthermore, there is a constant $C = C(\beta)$ such that for any $k \leq n/2$, the linear convergence rate satisfies

$$\alpha^{\text{RBK}} \geq C \frac{k^\beta \sigma_n^2}{\|L_n\|_F^2}.$$

Regarding the convergence rate of mSGD, first consider a finite matrix A . It is demonstrated in [31] that, with a minibatch size of 1 and importance sampling, the corresponding convergence parameter can be at most $\alpha^{\text{SGD}} \leq \kappa_{\text{dem}}^{-2}(A)$ for convergent step sizes. It is straightforward to generalize this result to the statistical problem (4.1), where the corresponding convergence bound is $\alpha^{\text{SGD}} \leq \kappa_{\text{dem}}^{-2}(L_n)$. When a larger minibatch size k is used, [24] shows that the learning rate can be increased at most linearly, corresponding to a rate of at most

$$\alpha^{\text{mSGD}} \leq k\kappa_{\text{dem}}^{-2}(L_n) = \frac{k\sigma_n^2}{\|L_n\|_F^2}.$$

□

Appendix C. Proofs of ReBlock Convergence Theorems. This appendix contains the proof of the ReBlock convergence bound Theorem 5.1. We first establish a useful lemma which bounds the error of the individual ReBlock iterates under arbitrary sampling distributions.

LEMMA C.1. *Consider the ReBlock iterates (1.7) and fix some sampling distribution ρ . Suppose that $x_0 \in \text{range}(A^\top)$ and $\text{range}(\bar{P}) = \text{range}(A^\top)$. Then for all t it holds*

$$(C.1) \quad \mathbb{E} \|x_t - x^{(\rho)}\|^2 \leq 2(1-\alpha)^t \|x_0 - x^{(\rho)}\|^2 + \frac{2}{\alpha} \mathbb{E}_{S \sim \rho} \|A_S^\top (A_S A_S^\top + \lambda k I)^{-1} r_S^{(\rho)}\|^2.$$

Proof. Since the ReBlock iteration does not admit a simple orthogonal decomposition, we instead proceed by utilizing a *bias-variance decomposition* inspired by [13, 24, 19].

Let $P_s = P(S_s)$ and $W_s = W(S_s)$. Bias and variance sequences are defined respectively by

$$(C.2) \quad d_0 = x_0 - x^{(\rho)}, \quad d_{s+1} = (I - P_s)d_s,$$

$$(C.3) \quad v_0 = 0, \quad v_{s+1} = (I - P_s)v_s + A^\top W_s r^{(\rho)}.$$

Intuitively, the bias sequence captures the error due to the initialization $x_0 \neq x^{(\rho)}$ and the variance sequence captures the rest of the error. It can be verified by mathematical induction and (2.6) that $x_s - x^{(\rho)} = d_s + v_s$ for all s . As a result, it holds

$$(C.4) \quad \mathbb{E} \|x_t - x^{(\rho)}\|^2 \leq 2\mathbb{E} \|d_t\|^2 + 2\mathbb{E} \|v_t\|^2.$$

Note also that it is a simple extension of Lemma A.2 that $d_s, v_s \in \text{range}(A^\top)$ for all s .

To analyze the bias, first calculate

$$(C.5) \quad \mathbb{E} [\|d_{s+1}^2\| | d_s] = d_s^\top \mathbb{E} [(I - P_s)^2] d_s = d_s^\top (I - 2\bar{P} + \mathbb{E}_{S \sim \rho} [P(S)^2]) d_s.$$

Observe that

$$(C.6) \quad P(S) = A_S^\top (A_S A_S^\top + \lambda k I)^{-1} A_S \preceq A_S^\top (A_S A_S^\top)^{-1} A_S \preceq I,$$

and hence $P(S)^2 \preceq P(S)$ and $\mathbb{E}_{S \sim \rho} [P(S)^2] \preceq \bar{P}$. As a result, $I - 2\bar{P} + \mathbb{E}_{S \sim \rho} [P(S)^2] \preceq I - \bar{P}$. Additionally leveraging the properties $d_s \in \text{range}(A^\top) = \text{range}(\bar{P})$ and $\bar{P} \preceq I$, Lemma A.3 implies

$$(C.7) \quad \mathbb{E} [\|d_{s+1}^2\| | d_s] \leq (1 - \alpha) \|d_s\|^2.$$

Iterating this inequality yields a simple bound on the bias term:

$$(C.8) \quad \mathbb{E} \|d_t\|^2 \leq (1 - \alpha)^t \left\| x_0 - x^{(\rho)} \right\|^2.$$

The analysis of the variance term is less straightforward. To begin, note that v_s follows the same recurrence as $x_s - x^{(\rho)}$, so we can apply a slight modification of Lemma A.1 to the variance sequence to obtain

$$(C.9) \quad \mathbb{E} [v_s] = (I - \bar{P})^s v_0 = 0.$$

Now, square the iteration for v_{s+1} conditioned on v_s :

$$\begin{aligned} \mathbb{E} \left[\|v_{s+1}\|^2 | v_s \right] &= v_s^\top \mathbb{E}_{S \sim \rho} [(I - P(S))^2] v_s + 2v_s^\top \mathbb{E}_{S \sim \rho} \left[(I - P(S)) A^\top W(S) r^{(\rho)} \right] + \mathbb{E}_{S \sim \rho} \left\| A^\top W(S) r^{(\rho)} \right\|^2 \\ &\leq (1 - \alpha) \|v_s\|^2 + 2v_s^\top \mathbb{E}_{S \sim \rho} \left[(I - P(S)) A^\top W(S) r^{(\rho)} \right] + \mathbb{E}_{S \sim \rho} \left\| A^\top W(S) r^{(\rho)} \right\|^2, \end{aligned}$$

where the second step uses our previous observations that $\mathbb{E}_{S \sim \rho} [(I - P(S))^2] \preceq I - \bar{P}$, $v_s \in \text{range}(\bar{P})$, and Lemma A.3. Take expectations over v_s as well to eliminate the cross-term, yielding

$$\mathbb{E} \|v_{s+1}\|^2 \leq (1 - \alpha) \mathbb{E} \|v_s\|^2 + \mathbb{E}_{S \sim \rho} \left\| A^\top W(S) r^{(\rho)} \right\|^2.$$

Iterating this last inequality for $s = 0, \dots, t-1$, using $\sum_{s=0}^{t-1} (1 - \alpha)^s < \sum_{s=0}^{\infty} (1 - \alpha)^s = 1/\alpha$, and recalling that $v_0 = 0$, we find

$$(C.10) \quad \mathbb{E} \|v_t\|^2 \leq \frac{1}{\alpha} \mathbb{E}_{S \sim \rho} \left\| A^\top W(S) r^{(\rho)} \right\|^2.$$

Noting that

$$(C.11) \quad A^\top W(S) r^{(\rho)} = A_S^\top (A_S A_S^\top + \lambda k I)^{-1} r_S^{(\rho)},$$

we can combine the bias bound (C.8) and the variance bound (C.10) using (C.4) to obtain the desired result. \square

Proof of Theorem 5.1. The result will follow from the appropriate application of Lemmas A.1, A.4, and C.1. To apply these lemmas it is first required to verify $\bar{P} \preceq I$ and $\text{range}(\bar{P}) = \text{range}(A^\top)$. For the first result, it suffices to observe that $P(S) \preceq P_{RBK}(S) \preceq I$ for all S .

For the second result, the logic follows almost identically to the same part of the proof of Theorem 3.1. The only difference is in how we show that $\bar{P}x \neq 0$ for any $x \in \text{range}(A^\top)$. Like before, we start by noting that there must exist a row index i such that $a_i^\top x \neq 0$. We then bound

$$(C.12) \quad x^\top \bar{P}x = \mathbb{E}_{S \sim \mathbf{U}(m, k)} \left[x^\top A_S^\top (A_S A_S^\top + \lambda k I)^{-1} A_S x \right] \geq \frac{k}{m} \cdot \frac{(a_i^\top x)^2}{\|A^\top A + \lambda k I\|_2} > 0,$$

where the intermediate step uses the facts that $\|A_S A_S^\top + \lambda k I\| = \|A_S^\top A_S + \lambda k I\| \leq \|A^\top A + \lambda k I\|$, that row i is chosen with probability k/m , and that $x^\top A_S^\top A_S x \geq x^\top a_i a_i^\top x$ when $i \in S$. It follows that $\bar{P}x \neq 0$ and so indeed $\text{range}(\bar{P}) = \text{range}(A^\top)$.

With the assumptions verified, Lemma A.1 can be applied using $r = 0$, $s = T$ to yield

$$(C.13) \quad \mathbb{E}[x_T] - x^{(\rho)} = (I - \bar{P})^T(x_0 - x^{(\rho)}).$$

This directly implies

$$(C.14) \quad \left\| \mathbb{E}[x_T] - x^{(\rho)} \right\| \leq (1 - \alpha)^T \left\| x_0 - x^{(\rho)} \right\|.$$

using Lemmas A.2 and A.3 and the proven properties of \bar{P} .

The conditions of Lemma C.1 are also satisfied, the results of which provide the conditions for Lemma A.4 with $B = \|x_0 - x^{(\rho)}\|^2$, $\tilde{V} = \frac{2}{\alpha} \mathbb{E}_{S \sim \rho} \left\| A_S^\top (A_S A_S^\top + \lambda k I)^{-1} r_S^{(\rho)} \right\|^2$. Lemma A.4 then yields the desired bound for the tail averages.

The next step is to bound the condition number $\kappa(\bar{W})$. Recall from the proof of Theorem 3.1 that $\|A_S A_S^\top\| \leq k \cdot \max_i \|a_i\|^2$, so that

$$(C.15) \quad \lambda k I \preceq A_S A_S^\top + \lambda k I \preceq (\max_i \|a_i\|^2 + \lambda) \cdot k I.$$

Now, when S is sampled uniformly, $\mathbb{E}_{S \sim \mathbf{U}(m,k)} [I_S^\top I_S] = \frac{k}{m} I$. It follows that

$$(C.16) \quad \bar{W} = \mathbb{E}_{S \sim \mathbf{U}(m,k)} [I_S^\top (A_S A_S^\top + \lambda k I)^{-1} I_S] \succeq \frac{1}{(\max_i \|a_i\|^2 + \lambda)m} I,$$

$$(C.17) \quad \bar{W} = \mathbb{E}_{S \sim \mathbf{U}(m,k)} [I_S^\top (A_S A_S^\top + \lambda k I)^{-1} I_S] \preceq \frac{1}{\lambda m} I.$$

Putting these together implies

$$(C.18) \quad \kappa(\bar{W}) \leq 1 + \frac{1}{\lambda} \cdot \max_i \|a_i\|^2.$$

The bounds on the weighted residual and the bias follow from Lemma A.5 with the bias bound taking the standard form

$$(C.19) \quad \|x^{(\rho)} - x^*\| \leq \sqrt{\kappa(\bar{W}) - 1} \cdot \|A^+\| \cdot \|r\|.$$

Since $\max_i \|a_i\|^2 \leq \|A\|^2$, the combination of (C.18) and (C.19) implies the looser bound

$$(C.20) \quad \|x^{(\rho)} - x^*\| \leq \frac{1}{\sqrt{\lambda}} \cdot \kappa(A) \cdot \|r\|.$$

Lastly, consider the variance $V = \mathbb{E} \|A_S^\top (A_S A_S^\top + \lambda k I)^{-1} r_S^{(\rho)}\|^2$. Since for $\sigma \geq 0$ one has $\sigma/(\sigma + \lambda k)^2 \leq 1/(4\lambda k)$, it holds

$$\left\| A_S^\top (A_S A_S^\top + \lambda k I)^{-1} \right\|^2 = \lambda_{\max} \left((A_S A_S^\top) (A_S A_S^\top + \lambda k I)^{-2} \right) \leq \frac{1}{4\lambda k}.$$

Thus

$$V \leq \frac{1}{4\lambda k} \cdot \mathbb{E} \|r_S^{(\rho)}\|^2 = \frac{1}{4\lambda m} \cdot \|r^{(\rho)}\|^2 \leq \frac{1}{4\lambda} \cdot \kappa(\bar{W}) \cdot \frac{\|r\|^2}{m}.$$

□

Appendix D. Noisy Linear Least-squares. In this section we consider the special case in which the inconsistency in the problem is generated by zero-mean noise. This case is more general than the case of Gaussian data, but less general than the case of arbitrary inconsistency. In particular, we consider the statistical linear regression problem

$$(D.1) \quad x^* = \operatorname{argmin}_{x \in \mathbb{R}^n} \mathbb{E}_{a \sim \mathbf{A}, z \sim \mathbf{Z}} [(a^\top x - (a^\top x^* - z))^2]$$

where \mathbf{A} is arbitrary, \mathbf{Z} satisfies $\mathbb{E}_{z \sim \mathbf{Z}} [z] = 0$, and $x^* \in \mathbb{R}^n$. This scenario might arise, for example, when making noisy measurements of x^* along the directions a .

The RBK-U and ReBlock-U algorithms are applied to (D.1) by generating k independent samples per iteration from the data distributions \mathbf{A}, \mathbf{Z} . The result is that RBK and ReBlock both converge (in a Monte Carlo sense) to the ordinary least-squares solution rather than a weighted solution. This can be viewed as an advancement over previous results on noisy linear systems, such as [29], which show only convergence to a finite horizon even for arbitrarily large noisy systems. The advance comes from treating the noise model explicitly and applying tail averaging to the algorithm.

THEOREM D.1. *Consider the RBK-U algorithm applied to the noisy linear least-squares problem (D.1). Then the results of Theorem 3.1 hold and $x^{(\rho)} = x^*$.*

THEOREM D.2. *Consider the ReBlock-U algorithm applied to the noisy linear least-squares problem (D.1). Then the results of Theorem 5.1 hold and $x^{(\rho)} = x^*$.*

Unified proof of Theorems D.1 and D.2. Recall that the weighted solution $x^{(\rho)}$ is characterized by

$$(D.2) \quad \bar{P}x^{(\rho)} = A^\top \bar{W}b = \bar{P}x^* + A^\top \bar{W}r = \bar{P}x^* + \mathbb{E} [A_S^\top M(S)r_S]$$

where $r_S = A_S x^* - b_S$. Based on the definition of the noisy equation (D.1), each entry of the residual satisfies $r_i = a_i^\top x^* - (a_i^\top x^* - z_i) = z_i$ and is thus distributed exactly according to \mathbf{Z} . In particular each entry of r_S is independent from A_S and the other entries of r_S and has a mean of zero. It follows that

$$(D.3) \quad \mathbb{E} [A_S^\top M(S)r_S] = \mathbb{E} [A_S^\top M(S)] \mathbb{E} [r_S] = 0,$$

which completes the proof that $x^{(\rho)} = x^*$. \square

Appendix E. Sampling from a Determinantal Point Process . In Section 5, we have presented the convergence of ReBlock to a weighted least-squares solution under the uniform sampling distribution $\rho = \mathbf{U}(m, k)$. However, the explicit convergence rate $1 - \alpha$ is still unknown. Motivated by recent work on Kaczmarz algorithms with determinantal point process (DPP) sampling [17], we now show that under DPP sampling, ReBlock converges to the ordinary least-squares solution and the convergence of ReBlock can have a much better dependence on the singular values of A than mSGD.

The sampling distribution that we consider is $\rho = k\text{-DPP}(AA^\top + \lambda kI)$, namely

$$(E.1) \quad \Pr[S] = \frac{\det(A_S A_S^\top + \lambda kI)}{\sum_{|S'|=k} \det(A_{S'} A_{S'}^\top + \lambda kI)}.$$

Our analysis is similar to parts of the nearly concurrent work [15], though [15] does not incorporate a fixed size k for their regularized DPP distribution.

THEOREM E.1. *Consider the ReBLoK algorithm with k -DPP sampling, namely $M(A_S) = (A_S A_S^\top + \lambda k I)^{-1}$ and $\rho = k\text{-DPP}(AA^\top + \lambda k I)$. Let $\alpha = \sigma_{\min}^+(\bar{P})$ and assume $x_0 \in \text{range}(A^\top)$. Then the expectation of the ReBLoK iterates x_T converges to x^* as*

$$(E.2) \quad \|\mathbb{E}[x_T] - x^*\| \leq (1 - \alpha)^T \|x_0 - x^*\|.$$

Furthermore, the tail averages \bar{x}_T converge to x^* as

$$(E.3) \quad \mathbb{E} \|\bar{x}_T - x^*\|^2 \leq 2(1 - \alpha)^{T_b+1} \|x_0 - x^*\|^2 + \frac{4}{\alpha^2(T - T_b)} V$$

with $V = \mathbb{E}_{S \sim \rho} \|A_S^\top (A_S A_S^\top + \lambda k I)^{-1} r_S\|^2$. The variance term is bounded by

$$(E.4) \quad V \leq \frac{1}{4\lambda} \cdot \max_i r_i^2.$$

Finally, for any $\ell < k$ the convergence parameter α satisfies

$$(E.5) \quad \alpha \geq \frac{k - \ell}{(k - \ell) + \kappa_\ell^2(A) + (m + k - 2\ell) \frac{\lambda k}{\sigma_{n_r}^2}},$$

where $\kappa_\ell^2(A) := \sum_{j>\ell}^{n_r} \sigma_j^2(A) / \sigma_{n_r}^2(A)$ and $\sigma_1(A) \geq \dots \geq \sigma_{n_r}(A) > 0 = \sigma_{n_r+1}(A) = \dots = \sigma_n(A)$ are the singular values of A .

By choosing $\ell = 1$ and a small value of λ , the above theorem indicates that when $n_r = n$ the value of α is at least on the order of $k/\kappa_{\text{dem}}^2(A)$ where $\kappa_{\text{dem}} := \|A\|_F / \sigma_{\min}(A)$ is the Demmel condition number. To understand the meaning of the bound more generally, suppose additionally that $\|a_i\| = 1$ for all i so that $\|A\|_F^2 = m$. Then, choosing $\ell = k/2$ we can rewrite (E.5) as

$$(E.6) \quad \begin{aligned} \alpha &\geq \frac{k}{k + 2\kappa_{k/2}^2(A) + 2m\lambda k / \sigma_n^2} \\ &\approx \frac{k}{2\kappa_{k/2}^2(A) + 2\lambda k \kappa_{\text{dem}}^2(A)}. \end{aligned}$$

As noted in Section 4, the convergence parameter for mSGD is bounded by

$$\alpha^{\text{SGD}} \leq k / \kappa_{\text{dem}}^2(A).$$

Hence, when the singular values of A decay rapidly and λk is significantly smaller than one, ReBLoK with DPP sampling can have a much faster convergence rate than mSGD.

Our bound for the variance of the tail-averaged estimator is crude and could likely be improved using techniques such as those of [15]. Nonetheless, it is interesting to note that the dependence of the variance on the residual vector can be worse in the case of DPP sampling than in the case of uniform sampling. This is because the DPP distribution can potentially sample the residual vector in unfavorable ways. It is thus unclear whether DPP sampling is a good strategy for arbitrary inconsistent systems, even if it can be implemented efficiently

Proof. The majority of the proof follows similarly to the proofs of Theorems 3.1 and 5.1, and we only highlight the differences, of which there are three: the fact that

$x^{(\rho)} = x^*$, the value of the parameter α , and the bound on the variance term in the tail-averaged bound.

We begin by verifying that $x^{(\rho)} = x^*$. For a vector $u \in \mathbb{R}^m$, we denote its elementary symmetric polynomial by $p_k(u) := \sum_{S \in \binom{[m]}{k}} \prod_{i \in S} u_i$, where $\binom{[m]}{k}$ is the set of all subsets of size k from the set $[m] = \{1, \dots, m\}$. Applying equation (5.3) of [17] by using $B = AA^\top + \lambda k I$ leads to

$$\overline{W} = \mathbb{E}_{S \sim \rho} [I_S^\top (I_S (AA^\top + \lambda k I) I_S^\top)^{-1} I_S] = \frac{U \text{diag}(p_{k-1}(q_{-1}), \dots, p_{k-1}(q_{-m})) U^\top}{p_k(q)}, \quad (\text{E.7})$$

where $A = U \Sigma V^\top$ denotes the full singular value decomposition of A , and $q_i = \sigma_i^2 + \lambda k$ when $i \leq n_r$ and $q_i = \lambda k$ otherwise, with $\sigma_1 \geq \dots \geq \sigma_{n_r}$ representing the sorted singular values and n_r the rank of A . Denote by $\overline{W} = U D U^\top$ with the diagonal matrix $D := \frac{\text{diag}(p_{k-1}(q_{-1}), \dots, p_{k-1}(q_{-m}))}{p_k(q)} \succ 0$. Then, by the definition $x^{(\rho)} = \arg\min_{x \in \mathbb{R}^n} \|Ax - b\|_{\overline{W}}^2$, we have

$$A^\top \overline{W} A x^{(\rho)} = A^\top \overline{W} b. \quad (\text{E.8})$$

Note that $A^\top \overline{W} A = V \Sigma^\top D \Sigma V^\top$ and $A^\top \overline{W} = V \Sigma^\top D U^\top b$. Also, because $\Sigma \in \mathbb{R}^{m \times n}$ and $D \in \mathbb{R}^{m \times m}$ are diagonal, we have $\Sigma^\top D = \hat{D} \Sigma^\top$, where $\hat{D} := D[1:n, 1:n]$ is the leading principal submatrix of D . Substituting this into (E.8) gives $V \hat{D} \Sigma^\top \Sigma V^\top x^{(\rho)} = V \hat{D} \Sigma^\top U^\top b$. Thus, we can apply V^\top on the left and use $D \succ 0$ to conclude

$$\Sigma^\top \Sigma V^\top x^{(\rho)} = \Sigma^\top U^\top b.$$

Applying V now on the left implies $A^\top A x^{(\rho)} = A^\top b$, i.e., $x^{(\rho)} \in \arg\min_{x \in \mathbb{R}^n} \|Ax - b\|^2$. Since $x^{(\rho)} \in \text{range}(A^\top)$ this implies that $x^{(\rho)} = x^*$.

Regarding α , plugging (E.7) into the definition of \overline{P} leads to

$$\overline{P} = A^\top \overline{W} A = V \Sigma^\top \frac{\text{diag}(p_{k-1}(q_{-1}), \dots, p_{k-1}(q_{-m}))}{p_k(q)} \Sigma V^\top. \quad (\text{E.9})$$

We now follow closely the logic of the proof of Lemma 4.1 of [17], making appropriate modifications to handle the fact that in ReBLoK we invert $AA^\top + \lambda k I$ rather than just AA^\top . Note also that through (E.9) we have explicitly verified that $\text{range}(\overline{P}) = \text{range}(A^\top)$, which is the needed condition for the contraction properties to hold in our convergence bound.

First, for any $\ell < k$ we can construct an approximation matrix

$$B_\ell = U \text{diag}(q_1, \dots, q_m) U^\top + \frac{1}{k - \ell} \left(\sum_{j > \ell}^m q_j \right) I. \quad (\text{E.10})$$

Note that we have used a mildly simpler and looser approximation than [17] by replacing $\frac{k - \ell - 1}{k - \ell}$ with 1 in the first term.

The same arguments as in [17] then imply that

$$\alpha = \sigma_{\min}^+(\overline{P}) \geq \sigma_{\min}^+(A^\top B_\ell^{-1} A). \quad (\text{E.11})$$

Furthermore we have

$$\begin{aligned}
\sigma_{\min}^+(A^\top B_\ell^{-1} A) &= \sigma_{\min}^+(V \Sigma^\top U^\top B_\ell^{-1} U \Sigma V^\top) \\
&= \sigma_{\min} \left(V \text{diag} \left(\frac{\sigma_1^2}{q_1 + \frac{1}{k-\ell} \sum_{j>\ell}^m q_j}, \dots, \frac{\sigma_{n_r}^2}{q_{n_r} + \frac{1}{k-\ell} \sum_{j>\ell}^m q_j} \right) V^\top \right) \\
&= \frac{\sigma_{n_r}^2}{q_{n_r} + \frac{1}{k-\ell} \sum_{j>\ell}^m q_j} \\
&= \frac{\sigma_{n_r}^2}{\sigma_{n_r}^2 + \frac{1}{k-\ell} \sum_{j>\ell}^{n_r} \sigma_j^2 + \frac{m+k-2\ell}{k-\ell} \lambda k} \\
&= \frac{k-\ell}{(k-\ell) + \kappa_\ell^2(A) + (m+k-2\ell) \frac{\lambda k}{\sigma_{n_r}^2}}.
\end{aligned}$$

This completes the bound for α .

Finally, we consider the bound for the variance term in the tail-averaged bound. Following the proof of Theorem 5.1 we can obtain an initial bound of

$$(E.12) \quad V \leq \frac{1}{4\lambda k} \cdot \mathbb{E}_{S \sim \rho} \|r_S\|^2.$$

Unfortunately, in the case of DPP sampling, we have no guarantee on how S samples the residual vector $r^{(\rho)}$. Thus, the best we can do is uniformly bound $\|r_S\|^2 \leq k \cdot \max_i r_i^2$. This yields the bound in the theorem. \square

In this supplement we provide more details on the numerical examples throughout the paper. The code to run the experiments can be found at <https://github.com/ggoldsh/block-kaczmarz-without-preprocessing>.

Appendix F. Experiments with random matrices. In this section we provide the details of the synthetic experiments from Sections 3.1, 4.1, and 5.1. All of these experiments have $m = 10^5$, $n = 10^2$, and $k = 30$. The experiments differ only in how the matrix A is generated. Thus, we first describe the matrix construction for each case separately, then describe the rest of the set-up in a unified manner.

Beginning with the experiments of Subsection 4.1, for the left panel of Figure 3 we generate the matrix A by setting each entry independently as $A_{ij} \sim \mathcal{N}(0, 1)$. For the right panel, we construct $A = GU$ where $G \in \mathbb{R}^{m \times n}$ has independent standard normal entries and U has $\sigma_i = 1/i^2$ and random orthonormal singular vectors. The condition number of the matrix A is $\kappa(A) \approx 1$ in the left panel and $\kappa(A) \approx 10^4$ in the right panel.

Regarding Subsections 3.1 and 5.1, the examples these sections are constructed as discretizations of underlying continuous problems, which is both a realistic scenario and serves to ensure that A contains many nearly singular blocks. In both cases, we first generate an $n \times n$ matrix C . For the left panel we set $C = I$, whereas for the right panel we set C with singular values $\sigma_i = 1/i$ and random orthonormal singular vectors. Once C is constructed, we define a set of n functions f_1, \dots, f_n by

$$(F.1) \quad f_j(s) = \sum_{\ell=1}^n C_{j\ell} T_\ell(s),$$

where T_ℓ is the ℓ^{th} Chebyshev polynomial of the first kind. The functions f correspond to the columns of A .

Next, we construct a vector v of m coordinates uniformly spaced across the interval $[-1, 1]$, namely

$$(F.2) \quad v_i = \left(-1 + 2 \frac{i-1}{m-1} \right).$$

Finally, the matrix A is designed as

$$(F.3) \quad A_{ij} = f_j(v_i),$$

so that column j of A is a discretized representation of the function f_j . The condition number of the resulting matrices A are $\kappa(A) \approx 11$ in the left panel and $\kappa(A) \approx 450$ in the right panel.

For all experiments in Subsections 3.1, 4.1, and 5.1 the vectors b are constructed by setting

$$(F.4) \quad b_i = a_i^\top y + z_i$$

where $y \sim \mathcal{N}(0, I_n)$ and $z_i \sim \mathcal{N}(0, \sigma)$. The noise level is set to $\sigma = 1e-2$ except for in Figure 7 where the noise level is explicitly varied. The initial guess is $x_0 = 0$ and the number of iterations is $T = 10^5$, which is equivalent to thirty passes over the data. The step size for minibatch SGD is constant within each run and has been tuned independently for each example, specifically to be as large as possible without introducing any signs of instability, up to a factor of 2. The value of λ for ReBlock has been set to $\lambda = 0.001$ and is not optimized on a per-example basis. The reported quantity is the relative solution error $\|x - x^*\| / \|x^*\|$. These experiments with random matrices are carried out in double precision on a 1.1 GHz Quad-Core Intel Core i5 CPU.

Appendix G. Natural Gradient Experiments. In this section we describe in detail the numerical experiments of Section 6. The training problem for the neural network is a simple function regression task on the unit interval. The target function is chosen to be periodic to avoid any consideration of boundary effects, and the neural network is correspondingly designed to be periodic by construction. Specifically, the target function is constructed as

$$(G.1) \quad f(s) = q(\sin(2\pi s)),$$

with the polynomial q defined as

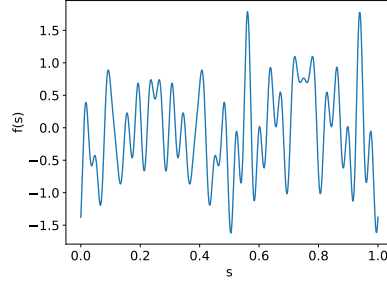
$$(G.2) \quad q(s) = \frac{1}{\sqrt{d}} \sum_{\ell=1}^d c_\ell T_\ell(s)$$

for $d = 30$ and each c_ℓ chosen randomly from a standard normal distribution. The resulting function is pictured in Figure 13.

The neural network model is a simple periodic ResNet [22] with 5 layers. For input $s \in \mathbb{R}$, the network outputs $f(s) \in \mathbb{R}$ are given by

$$(G.3) \quad y_1 = \tanh(W_1 \sin(2\pi s) + b_1),$$

$$(G.4) \quad y_i = y_{i-1} + \tanh(W_i y_{i-1} + b_i); \quad i = 2, 3, 4,$$

FIG. 13. *Target function for neural network training.*

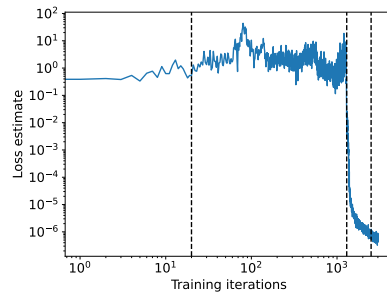
$$(G.5) \quad f(s) = W_5 y_4 + b_5.$$

The intermediate layers have dimensions $y_i \in \mathbb{R}^{500}$ and the weight matrices W_i and bias vectors b_i have the appropriate dimensions to match. The weights are initialized using a Lecun normal initialization and the biases are all initialized to zero. The parameters are collected into a single vector $\theta \in \mathbb{R}^{753,001}$ for convenience, leading to the neural network function $f_\theta(s)$.

The loss function is defined as in (6.1) and the network is trained using subsampled natural gradient descent, as described for example in [37], with a batch size of $N_b = 500$, a Tikhonov regularization of $\lambda = 0.01$, and a step size of $\eta = 0.5$. This corresponds to the parameter update

$$(G.6) \quad \theta_{t+1} = \theta_t - \eta J_S^\top (J_S J_S^\top + N_b \lambda I)^{-1} [f_\theta - f]_S,$$

where S represents the set of N_b sample points. The setting of $N_b = 500$ is intended to represent something close to a full batch training regime, which is only practical since our function regression problem lives in a very compact, one-dimensional domain. The resulting training curve is presented in Figure 14.

FIG. 14. *Training curve for the neural network function regression example. The three vertical lines indicate the training snapshots that are used to generate the least-squares problems studied in Figure 8.*

From this training run, three snapshots are taken and used to generate the least-squares problems for Figures 8 to 12, following (6.6). The first snapshot is from the “pre-descent” phase before the loss begins to decrease, the second is from the “descent” phase during which the loss decreases rapidly, and the third is from the

“post-descent” phase when the decay rate of the loss has slowed substantially. The snapshots are indicated by the vertical dotted lines in Figure 14 and form the basis of the experiments of Figures 8 to 12. The batch size of $k = 50$ used in Figure 8 is meant to represent the realistic scenario when each iteration uses too few samples to thoroughly represent the target function. Furthermore, the continuous problems are treated directly by uniformly sampling k points from the domain $[0, 1]$ at each iteration and calculating the network outputs and gradients at these points.

The initial guess is $x_0 = 0$ in every case and the step size for minibatch SGD is constant within each run and has been tuned independently for each example, specifically to be as large as possible without introducing any signs of instability, up to a factor of 2. The value of λ for ReBlock has been set to $\lambda = 0.001$ unless otherwise specified, and is not optimized on a per-example basis. The reported quantity is the relative residual $\tilde{r} = \|Jx - [f_\theta - f]\| / \|f_\theta - f\|$, which measures how well the function-space update direction Jx agrees with the function-space loss gradient $f_\theta - f$. This quantity is estimated at each iteration using the available sample points.

Disclaimer. This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

REFERENCES

- [1] S.-I. AMARI, *Natural gradient works efficiently in learning*, Neural computation, 10 (1998), pp. 251–276.
- [2] H. ASI AND J. C. DUCHI, *Stochastic (approximate) proximal point methods: Convergence, optimality, and adaptivity*, SIAM Journal on Optimization, 29 (2019), pp. 2257–2290.
- [3] D. P. BERTSEKAS, *Incremental proximal methods for large scale convex optimization*, Mathematical programming, 129 (2011), pp. 163–195.
- [4] A. BIETTI AND J. MAIRAL, *On the inductive bias of neural tangent kernels*, Advances in Neural Information Processing Systems, 32 (2019).
- [5] J. BRUNA, B. PEHERSTORFER, AND E. VANDEN-ELINDEN, *Neural galerkin schemes with active learning for high-dimensional evolution equations*, Journal of Computational Physics, 496 (2024), p. 112588.
- [6] Y. CAO, Z. FANG, Y. WU, D.-X. ZHOU, AND Q. GU, *Towards understanding the spectral bias of deep learning*, arXiv preprint arXiv:1912.01198, (2019).
- [7] Y. CENSOR, *Row-action methods for huge and sparse systems and their applications*, SIAM review, 23 (1981), pp. 444–466.
- [8] A. CHEN AND M. HEYL, *Empowering deep neural quantum states through efficient optimization*, Nature Physics, 20 (2024), pp. 1476–1481.
- [9] J. CHUNG, M. CHUNG, J. T. SLAGEL, AND L. TENORIO, *Sampled limited memory methods for massive linear inverse problems*, Inverse Problems, 36 (2020), p. 054001.
- [10] F. DANGEL, J. MÜLLER, AND M. ZEINHOFFER, *Kronecker-factored approximate curvature for physics-informed neural networks*, Advances in Neural Information Processing Systems, 37 (2024), pp. 34582–34636.
- [11] D. DAVIS AND D. DRUSVYATSKIY, *Stochastic model-based minimization of weakly convex functions*, SIAM Journal on Optimization, 29 (2019), pp. 207–239.
- [12] A. DE CEZARO, J. BAUMEISTER, AND A. LEITAO, *Modified iterated Tikhonov methods for solving*

- systems of nonlinear ill-posed equations*, Inverse Problems and Imaging, 5 (2011), pp. 1–17.
- [13] A. DÉFOSSEZ AND F. BACH, *Averaged least-mean-squares: Bias-variance trade-offs and optimal sampling distributions*, in Artificial Intelligence and Statistics, PMLR, 2015, pp. 205–213.
 - [14] M. DEREZIŃSKI, D. LEJEUNE, D. NEEDELL, AND E. REBROVA, *Fine-grained analysis and faster algorithms for iteratively solving linear systems*, Journal of Machine Learning Research, 26 (2025), pp. 1–49.
 - [15] M. DEREZIŃSKI, D. NEEDELL, E. REBROVA, AND J. YANG, *Randomized Kaczmarz methods with beyond-Krylov convergence*, SIAM Journal on Matrix Analysis and Applications, 46 (2025), pp. 2558–2588.
 - [16] M. DEREZIŃSKI AND E. REBROVA, *Sharp analysis of sketch-and-project methods via a connection to randomized singular value decomposition*, SIAM Journal on Mathematics of Data Science, 6 (2024), pp. 127–153.
 - [17] M. DEREZIŃSKI AND J. YANG, *Solving dense linear systems faster than via preconditioning*, in Proceedings of the 56th Annual ACM Symposium on Theory of Computing, 2024, pp. 1118–1129.
 - [18] T. ELFVING, *Block-iterative methods for consistent and inconsistent linear equations*, Numerische Mathematik, 35 (1980), pp. 1–12.
 - [19] E. N. EPPERLY, G. GOLDSHLAGER, AND R. J. WEBBER, *Randomized Kaczmarz with tail averaging*, Applied and Computational Harmonic Analysis, (2025), p. 101812.
 - [20] G. GOLDSHLAGER, N. ABRAHAMSEN, AND L. LIN, *A Kaczmarz-inspired approach to accelerate the optimization of neural network wavefunctions*, Journal of Computational Physics, 516 (2024), p. 113351.
 - [21] R. M. GOWER AND P. RICHTÁRIK, *Randomized iterative methods for linear systems*, SIAM Journal on Matrix Analysis and Applications, 36 (2015), pp. 1660–1690.
 - [22] K. HE, X. ZHANG, S. REN, AND J. SUN, *Deep residual learning for image recognition*, in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.
 - [23] J. HERMANN, J. SPENCER, K. CHOO, A. MEZZACAPO, W. M. C. FOULKES, D. PFAU, G. CARLEO, AND F. NOÉ, *Ab initio quantum chemistry with neural-network wavefunctions*, Nature Reviews Chemistry, 7 (2023), pp. 692–709.
 - [24] P. JAIN, S. M. KAKADE, R. KIDAMBI, P. NETRAPALLI, AND A. SIDFORD, *Parallelizing stochastic gradient descent for least squares regression: mini-batching, averaging, and model misspecification*, Journal of machine learning research, 18 (2018), pp. 1–42.
 - [25] Y. LIU AND C.-Q. GU, *On greedy randomized block Kaczmarz method for consistent linear systems*, Linear Algebra and Its Applications, 616 (2021), pp. 178–200.
 - [26] J. MARTENS, *New insights and perspectives on the natural gradient method*, Journal of Machine Learning Research, 21 (2020), pp. 1–76.
 - [27] J. MARTENS AND R. GROSSE, *Optimizing neural networks with Kronecker-Factored Approximate Curvature*, in International conference on machine learning, PMLR, 2015, pp. 2408–2417.
 - [28] J. MÜLLER AND M. ZEINHOFER, *Achieving high accuracy with PINNs via energy natural gradient descent*, in International Conference on Machine Learning, PMLR, 2023, pp. 25471–25485.
 - [29] D. NEEDELL, *Randomized Kaczmarz solver for noisy linear systems*, BIT Numerical Mathematics, 50 (2010), pp. 395–403.
 - [30] D. NEEDELL AND J. A. TROPP, *Paved with good intentions: analysis of a randomized block Kaczmarz method*, Linear Algebra and its Applications, 441 (2014), pp. 199–221.
 - [31] D. NEEDELL, R. WARD, AND N. SREBRO, *Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm*, Advances in neural information processing systems, 27 (2014).
 - [32] C.-Y. PARK AND M. J. KASTORYANO, *Geometry of learning neural quantum states*, Physical Review Research, 2 (2020), p. 023232.
 - [33] A. PĂTRAȘCU, *New nonasymptotic convergence rates of stochastic proximal point algorithm for stochastic convex optimization*, Optimization, 70 (2021), pp. 1891–1919.
 - [34] D. PFAU, J. S. SPENCER, A. G. MATTHEWS, AND W. M. C. FOULKES, *Ab initio solution of the many-electron Schrödinger equation with deep neural networks*, Physical review research, 2 (2020), p. 033429.
 - [35] A. RAKHLIN, O. SHAMIR, AND K. SRIDHARAN, *Making gradient descent optimal for strongly convex stochastic optimization*, arXiv preprint arXiv:1109.5647, (2011).
 - [36] E. REBROVA AND D. NEEDELL, *On block Gaussian sketching for the Kaczmarz method*, Numerical Algorithms, 86 (2021), pp. 443–473.
 - [37] Y. REN AND D. GOLDFARB, *Efficient subsampled Gauss-Newton and natural gradient methods*

- for training neural networks*, arXiv preprint arXiv:1906.02353, (2019).
- [38] B. RONEN, D. JACOBS, Y. KASTEN, AND S. KRITCHMAN, *The convergence rate of neural networks for learned functions of different frequencies*, Advances in Neural Information Processing Systems, 32 (2019).
- [39] Z. SCHÄTZLE, P. B. SZABÓ, M. MEZERA, J. HERMANN, AND F. NOÉ, *Deepqmc: An open-source software suite for variational optimization of deep-learning molecular wave functions*, The Journal of Chemical Physics, 159 (2023).
- [40] M. SCHMITT AND M. HEYL, *Quantum many-body dynamics in two dimensions with artificial neural networks*, Physical Review Letters, 125 (2020), p. 100503.
- [41] P. F. SHUSTIN AND H. AVRON, *Semi-infinite linear regression and its applications*, SIAM Journal on Matrix Analysis and Applications, 43 (2022), pp. 479–511.
- [42] T. STROHMER AND R. VERSHYNIN, *A randomized Kaczmarz algorithm with exponential convergence*, Journal of Fourier Analysis and Applications, 15 (2009), pp. 262–278.
- [43] S. WANG, X. YU, AND P. PERDIKARIS, *When and why PINNs fail to train: A neural tangent kernel perspective*, Journal of Computational Physics, 449 (2022), p. 110768.
- [44] A. ZOUZIAS AND N. M. FRERIS, *Randomized extended Kaczmarz for solving least squares*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 773–793.