READ BETWEEN THE HYPERPLANES: ON SPECTRAL PROJECTION AND SAMPLING APPROACHES TO RANDOMIZED KACZMARZ[*]

JAMES A. NGUYEN[†], OLEG PRESNYAKOV[†], AND ADITYAKRISHNAN RADHAKRISHNAN[†]

Abstract. Among recent developments centered around Randomized Kaczmarz (RK), a row-sampling iterative projection method for large-scale linear systems, several adaptions to the method have inspired faster convergence. Focusing solely on ill-conditioned and overdetermined linear systems, we highlight inter-row relationships that can be leveraged to guide directionally aware projections. In particular, we find that improved convergence rates can be made by (i) projecting onto pairwise row differences, (ii) sampling from partitioned clusters of nearly orthogonal rows, or (iii) more frequently sampling spectrally-diverse rows.

Key words. Numerical Linear Algebra, Iterative Methods, Ill-Conditioned, Overdetermined, Randomized Kaczmarz, Sampling Kaczmarz Motzkin, Clustering, Pairwise Differences, Spectral Analysis, Directionally Aware Projections, Singular Vector Analysis, Convergence Rate

AMS subject classifications. 65B99, 65F10, 65F20

1. Introduction.

The size of computational models and applications continues to grow at an exponential rate. Thus, we seek faster, more robust algorithms to handle the large-scale linear systems that depend on them. In the past decade, two prevalent examples, Randomized Kaczmarz (RK) and Sampling Kaczmarz-Motzkin (SKM), have seen drastic improvement in contributions that utilize an added element of randomization. The aforementioned methods add an element of randomization and partial greediness to promote faster convergence [2]. These methods are row-iterative projection methods for solving the aforementioned large-scale linear systems $Ax = b$, where x^* is the intended solution in the set of feasible solutions $x = \{x \in \mathbb{R}^m : Ax \leq b\}$. RK randomizes the sampling of the existing Kaczmarz projection method; if $A \in \mathbb{R}^{m\times n}$, the method updates the current iterate as follows:

$$x_{k+1} = x_k - \frac{\langle a_i, x_k\rangle - b_i}{||a_i||^2} x_k, \tag{1.1}$$

where a_i is the i^{th} row, randomly sampled from $1, \dots, m$. Traditionally, it is common to use a sampling distribution that selects rows with probability proportional to its squared Euclidean norm $||a_i||^2$; this extension allows us to guarantee, within reasonable probability, convergence of the k^{th} iterate x_k to x^* that is at least linear in expectation:

$$\mathbb{E}(||\varepsilon_k||^2) \leq \left(1 - \frac{\sigma^2_{\min}(A)}{||A||_F^2}\right)^k ||\varepsilon_0||^2, \tag{1.2}$$

where ϵ_k is the k^{th} error vector $x_k - x^*$ [5]. Similarly, the SKM algorithm takes a partially greedy approach to sampling, promoting faster iterative convergence at a higher computational cost. Given a greediness parameter $\beta \in \mathbb{N}$, this algorithm samples without replacement a set τ_k with $|\tau_k| = \beta$ from the system $A \in \mathbb{R}^{m\times n}$, then it projects onto the row $a_i \in \tau_k$ that maximizes the residual $\langle a_i, x_k\rangle - b_i$. By greedily

[*] Advised by Dr. Anna Ma (University of Calofornia, Irvine)

[†] Department of Mathematics, University of California, Irvine.

choosing a row from τ_k at each iterate k, the convergence in expectation improves drastically (See Appendix A) [3].

Note that the bound for the k^{th} expected error of either algorithm is dependent on the scaled conditioning of the matrix. Consequently, we would expect that the algorithms would have weaker theoretical guarantees for ill-conditioned linear systems. Thus, we seek to explore several methods for sampling and clustering to address RK and SKM's limitations under these conditions. For the remainder of this study, we restrict our attention to the ill-conditioned case for overdetermined consistent linear systems.

Throughout this paper, we suggest three different methods to improve the convergence rate and approximation error of RK and SKM: (i) a transformed linear feasibility problem that considers pairwise row differences in section 2, (ii) the construction of a coreset to use partitions of the system to compute its solution in section 3, and (iii) an increased sampling frequency of rows corresponding to most underrepresented singular values of the system in section 4.

2. Pairwise Comparisons for Binary Classification. In the following section, we offer an alternative solution to solving overdetermined binary classification problems using row-iterative methods. We propose a procedure to re-imagine a binary classification problem as a linear feasibility one, and we aim to leverage variations of RK to iteratively converge toward a solution. To transform the problem, we define the following Hadamard product operation to scale each row of the system by the sign of its label.

Definition 2.1 (Hadamard Product). *Let $A, B \in \mathbb{R}^{m\times n}$. Then, we take the Hadamard product to be $\odot : \mathbb{R}^{m\times n} \times \mathbb{R}^{m\times n} \to \mathbb{R}^{m\times n}$ defined by:*

$$(A \odot B)_{ij} = A_{ij}B_{ij},$$

the element-wise matrix product.

We consider a consistent system of inequalities $Ax^* \leq b$ for some matrix A and an unknown intended solution vector x^* such that

$$b_i = \begin{cases} -1, & (Ax^*)_i < 0 \\ 1, & (Ax^*)_i \geq 0 \end{cases}.$$

To setup the linear feasibility problem, we obtain the matrix $B \in \mathbb{R}^{m\times n}$, where $B_{:j} = b$ for all $j = 1, \ldots, n$. Using this matrix, along with the Hadamard product, we define:

$$A' := -B \odot A$$

and the zero vector $b' = 0$. Thus, we end up with the linear feasibility problem $A'x^* \leq b'$.

Recall that the original goal of the problem is to find a vector x such that

$$\text{sign}(\langle A_i, x\rangle) = \text{sign}(\langle A - i, x^*\rangle) = b_i.$$

Note that the following conditions are equivalent:

$$\begin{aligned} b_i \times \operatorname{sign}(\langle A_i, x\rangle) &> 0 \\ \iff (B_i A_i)x &> 0 \\ \iff ((-B \odot A)x)_i &< 0 \\ \iff (A'x)_i &< b'_i \end{aligned} \tag{2.1}$$

Thus, we can solve a binary classification problem for an overdetermined linear system by using row-iterative methods such as RK and SKM, as long as we transform it by the described procedure.

2.1. Approach.

Although it would be valid to simply apply the RK or SKM algorithm to the resulting linear feasibility problem, we suggest that additional informative comparisons can be made between rows in the system. To test this hypothesis, we consider extracting from a matrix A', the pairwise differences matrix $P \in \mathbb{R}^{\binom{m}{2}\times n}$ defined by:

$$P_{h,:} = A'_i - A'_j,$$

where $h \in \{1, \dots, \binom{m}{2}\}$ is a unique combination of indices $i \neq j$, and $i, j \in \{1, \dots, m\}$. If we row bind A' and P, we obtain a combined matrix $A' \cup P \in \mathbb{R}^{m+\binom{m}{2}\times n}$ with additional comparisons. Since the rows of A' have been scaled by the signs of Ax^*, we have an similar problem to 2.1. We can again proceed to using RK or SKM to find a vector $x \in \mathbb{R}^{m+\binom{m}{2}}$ that satisfies $(A' \cup P)x < 0$.

If we define $y = x_{1:m,:}$, we hypothesize that the k^{th} iterate $y^{(k)}$ converges to x^*. Furthermore, we hypothesize that it will converge faster than the iterate $x'^{(k)}$, obtained by applying RK or SKM to find $x' \in \mathbb{R}^m$ satisfying $A'x' < 0$.

2.2. Methods.

To analyze this relationship, we run numerical experiments on 100 random matrices, obtained by taking reversing the Singular Value Decomposition of two random orthonormal matrices U, V and a diagonal matrix S an exponentially scaled singular values to replicate an ill-conditioned system.

For the following experiments, we employ the SKM algorithm with a greediness parameter $\beta = 3$.

Algorithm 2.1 SKM for linear feasibility

```
procedure SKM($A$, $b$, $x_0$, $x^*$, $\beta$, $\lambda$, $K$)
    $k = 1$
    while $k < K$
        Choose a sample of $\beta$ rows, $\tau_k$, from $A$.
        $t_k := \operatorname*{argmax}_{i \in \tau_k} a_i^T x_{k-1} - b_i$
        $x_k = x_{k-1} - \lambda(a_{t_k}^T x_{k-1} - b_{t_k})^+ a_{t_k}$
        $k = k + 1$
    return $x_k$
end procedure
```

We apply the algorithm to the matrix $(A' \cup P)$ described in subsection 2.1, and we compare the average approximation error, Chebyshev error, and binary classification accuracy of the algorithm performed using three different sampling schemes:

1. first m rows sampled uniformly, remaining $\binom{m}{2}$ rows not sampled,
2. all $m + \binom{m}{2}$ rows weighted uniformly,
3. first m rows not sampled, remaining $\binom{m}{2}$ rows sampled uniformly.

We define the Chebyshev error to be the distance from the iterate vector to the Chebyshev center of the feasible region.

Definition 2.2 (Chebyshev Center of a Feasible Region). *The point within a feasible region that maximizes the volume of a feasible sphere centered around it.*

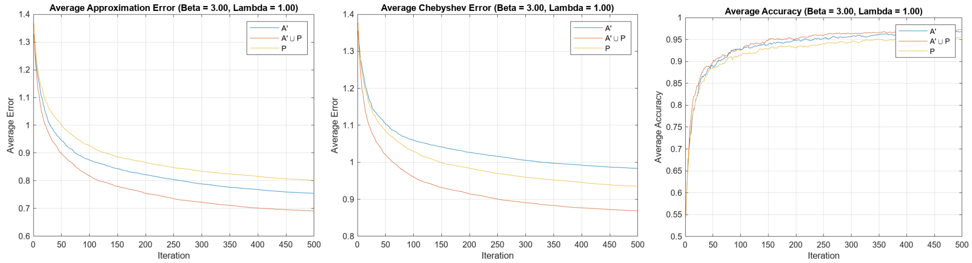

Fig. 1. *Quantitative results for $A' \in \mathbb{R}^{240\times 12}$, averaged over 100 trials: (i) Iteration vs. Average Approximation Error, (ii) Iteration vs. Average Chebyshev Error, and (iii) Iteration vs. Average Accuracy (left to right). All results obtained by applying SKM using $\beta = 3$, with no over-projection parameter ($\lambda = 1$). The above plots feature metrics associated with the sampling schemes 1, 2, and 3, referenced by $A', A' \cup P$, and P, respectively.*

2.3. Results.

The numerical experiments suggest that sampling scheme 2 retains similar success in producing an accurate binary classifier; however, it reveals that additionally sampling pairwise differences promotes a reduced average approximation error and Chebyshev error. Although it seems that sampling a pairwise difference row could be more informative, we also see that strictly sampling these rows produces a less accurate binary classifier with greater approximation error. Thus, it should not be assumed that these comparisons are strictly more informative of the true solution to the system.

3. Clustering.

Another key question we pursued was how to extract a compact coreset from an over-determined linear system. We conjecture that any system of size $m \times n$ contains a coreset of cardinality $O(cn)$, where c is a small constant. By isolating this subset, existing algorithms could solve the reduced linear problem far more rapidly. Our intuition was to partition the matrix A into clusters of rows, enabling each cluster to be processed independently and thus lowering the overall computational cost. The next section explains the underlying idea and the algorithms that realize it.

3.1. Existence of a “Good” Cluster.

To obtain a quick proof–of–concept that a small *inner-product coreset* can solve an over-determined linear system almost as accurately as the full data set, we ran the following procedure on a synthetic ill-conditioned matrix.

1. **Data generation.** For most of our tests we generated *ill-conditioned matrices*, which usually constitute the worst-case scenario for iterative algorithms. Consequently, insights gained on such matrices tend to generalize well to arbitrary random ones. To build each ill-conditioned matrix we employed an

SVD-based procedure: two random orthogonal factors were sampled, and the singular values were manually assigned so that the ratio between the largest and the smallest was on the order of 10^7.

2. **Row scoring.** For every row a_i compute the absolute inner product $s_i = \left|\langle a_i,\, x^\star\rangle\right|$. Large scores indicate rows whose directions are highly aligned with $x^\star$.
3. **Coreset extraction.** Sort the scores and keep the $k = c\,n$ rows with the *smallest* absolute inner products (we use the heuristic factor $c = 2$). The retained rows form a sub-matrix B with the matching right-hand side b_B.
4. **Least-squares solves.**
   - Full system: $x_A = \arg\min_x \|A\,x - b\|_2$.
   - Coreset system: $x_B = \arg\min_x \|B\,x - b_B\|_2$.

   Both minimizers are obtained with `numpy.linalg.lstsq`.
5. **Evaluation.** Measure the relative error $\|x_B - x_A\|_2 / \|x_A\|_2$ and compare residual norms $\|Ax_B - b\|_2$ and $\|Bx_B - b_B\|_2$.

3.2. Algorithms.

In this section we will talk about all the algorithms we've been trying to utilize in order to achieve the best clustering. At the end of the section we will provide plots with all clustering methods to compare its result on a real data.

3.2.1. ϵ-cover net Algorithm.

The first algorithm is based on an ϵ-cover net idea. For visualization, imagine each row as a vector in an n-dimensional vector space that connects the origin and the point defined by coefficients of a matrix. Then we are trying to understand how each row acts on the other vectors in our space with respect to an inner product. It is not hard to imagine that if points are close to each other, then they act similarly. As a result, we decided to use partition based on how far from each other points are. For details, refer to Algorithm 3.1.

Algorithm 3.1 Epsilon-Cover Net Algorithm

```
Assign $a_1$ to Cluster $S_1$.
for each $a_k$, $k > 1$ do
  if $\exists\ S_i$ such that $\left\langle a_k, \frac{1}{p}\sum_{j=1}^p s_j \right\rangle < \varepsilon$ then
    Append $a_k$ to $\arg\min_i S_i$
  else
    Initialize new cluster, append $a_k$.
  end if
end for
```

After clustering, we project iterates onto one random row from each cluster. Then pick the "best" cluster, S^*, where the criterion for "best" is being the most orthogonal cluster to the current iterate (which is our best approximation for x^*), i.e., we let S^* be cluster S_i where

$$i := \operatorname*{argmin}_i |\langle x_k, \frac{1}{p}\sum_{i=1}^{p} s_i\rangle|.$$

The algorithm's performance hinges on the chosen coverage radius ϵ. If ϵ is set too small relative to the spacing of the vectors, clusters become sparse and uninformative; if too large, nearly all rows collapse into a single cluster. Hence, our next objective is to identify an ϵ that yields a well-balanced matrix partition. In addition, because we track the mean projection in every cluster, we can dynamically reassign rows to different clusters once the current partition stops producing error reductions in further iterations.

3.2.2. Online cluster updating.

Another idea is essentially taken from a well-known K-means algorithm. Since we know that the Karzmarz algorithm will eventually converge to a solution, we were trying to find an optimization that will be updated during the iterations. Each fixed number of iterations we were trying to reduce the matrix A that we were working with. Particularly, for an $m \times n$ matrix:

1. Start with a cluster of the $2n$ best rows from the first active set which is the entire matrix of m rows.
2. Every $(2^k \times 100)$-th iteration $(k = 0, 1, ...)$, reduce the size of the active set by a factor of 2 $(\frac{m}{2}, \frac{m}{4}, ...)$ and stop at $4n$ rows.
3. For each active set, we recluster and choose the $2n$ best rows based on the current iterate.

The "best" row is defined as the one most orthogonal to the current iterate:

$$i \;:=\; \operatorname*{arg\,min}_{i \in [m]} \left(a_i^\top x_{k-1} \;-\; b_i \right).$$

With each iteration the rows we discard contribute less and less: their projections cease to reveal new information about the solution subspace. Although the exact pattern of elimination depends on the particular right-hand side, the sequence of discarded sets still carries useful structure. We can harness this history to pre-form clusters whenever the same matrix must be solved against multiple right-hand sides.

3.3. Results.

We include two main ways to test the quality of partition. The first one is the condition testing. We hoped that our cluster would find a subset with a better conditioning than the random sample of the same size. Intuitively, partitioning

4. Convergence along Singular Directions.

4.1. Background.

The latest avenue of our exploration emphasizes a direction-aware approach to RK algorithms. Whilst previous approaches focus on measures of orthogonality, we redirect that focus to analyzing the singular directions present in our system. Our analysis is inspired by recent developments revealing that the error vector converges along the system's smallest singular vector [4]. Steinerberger's analysis of the convergence of the error vector suggests that RK – under traditional sampling techniques – struggles to iterate in the directions least represented by the data.

We see that Figure 2 confirms Steinerberger's results from [4] and suggest the component of the error vector ε_k in the direction of the largest singular vectors converge to 0 most quickly.

4.2. Approach.

To compensate for the under-representation of small singular vectors in the trajectory of our iterate x_k, we seek to adapt our sampling distribution to select with higher probability rows that have a greater component in the direction of the least singular vector. Let a_i be the ith row of $A \in \mathbb{R}^{m \times n}$. Then, a_i is some linear combination of the n singular vectors:

$$a_i = \sum_{j=1}^{n} c_j^{(i)} \vec{v_j} = c_1^{(i)} \vec{v_1} + \cdots c_n^{(i)} \vec{v_n}, \qquad c_j^{(i)} \in \mathbb{R},\; i = 1, \ldots m. \tag{4.1}$$

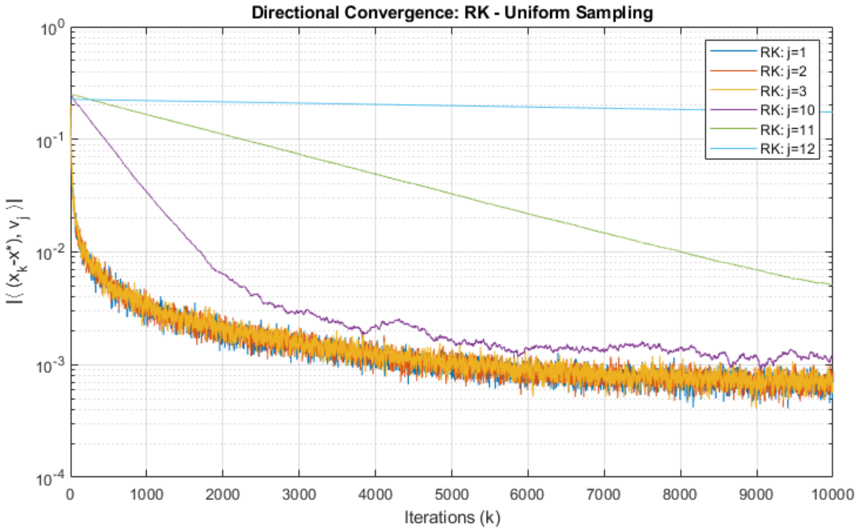

FIG. 2. *$A \in \mathbb{R}^{240\times 12}$ is constructed as described in Section 4.3. The figure demonstrates the convergence of $|\langle x_k - x^*, v_j\rangle|$ for each singular vector v_j $(j = 1,\dots,n)$ at each iteration k. It demonstrates the ordinal spectrum of convergence by decreasing singular value.*

where each $\vec{v}_j$ is the j^{th} singular vector of A. In the context of our problem, it should not be assumed that knowledge of the singular vectors be known; otherwise, the solution could be reconstructed and the problem would be trivial. However, we analyze the convergence using this knowledge to inspire a new algorithm. Considering that $A = U\Sigma V^\mathsf{T}$, we obtain each $c_n^{(i)}$ by:

$$c_n = [c_n^{(1)}, \dots, c_n^{(m)}]^\mathsf{T} = AV_{:n} \tag{4.2}$$

With these nth spectral coefficients c_n, we construct spectral sampling weights ω_i for each row of the system:

$$\omega_i := \frac{|c_n^{(i)}|}{\sum_{l=1}^{m} |c_n^{(l)}|}, \qquad i = 1,\dots,m. \tag{4.3}$$

By substituting the row norm-based sampling distribution by the one determined by the spectral weights from (4.3), we run a series of numerical experiments, expecting to improve upon the approximation error horizon encountered by the RK algorithm.

4.3. Numerical Experimentation. For all numerical experiments in this section, we refer to the following matrix construction: We produce $A \in \mathbb{R}^{240\times 12}$ by taking the QR-decompositions of the Gaussian matrices $\hat{U} \in \mathbb{R}^{240\times 240}$ and $\hat{V} \in \mathbb{R}^{12\times 12}$ to obtain orthogonal matrices $U \in \mathbb{R}^{240\times 240}$ and $V \in \mathbb{R}^{12\times 12}$. We then construct a matrix $\Sigma \in \mathbb{R}^{240\times 12}$ with entries $\Sigma_{jj} = e^{n-j+1}$, $(j = 1,\dots,n)$ and 0 everywhere else such that we obtain $A = U\Sigma V^T$. To complete the system, we let $x^* = [0, 0, \dots, 1]^\mathsf{T}$ be the length-n solution vector to $Ax^* = b$.

To quantify the progress of the iterate x_k in the direction of each singular vector, we follow the dot product of the error vector ε_k with each singular vector. We define the jth singular error at each iteration as:

$$\varepsilon_k^{(j)} := |\langle x - x^*, v_j\rangle|, \qquad j = 1, \ldots, n \tag{4.4}$$

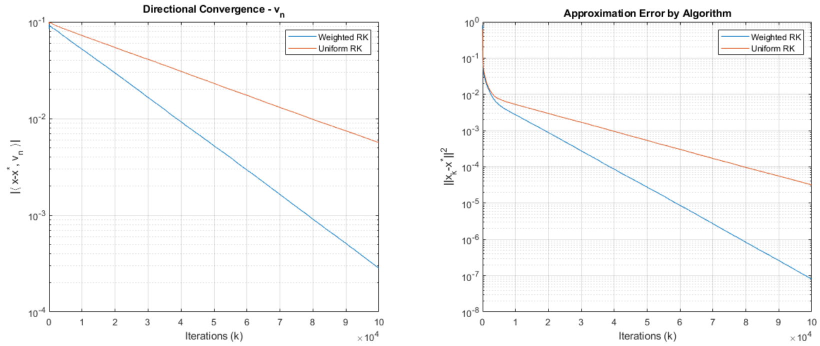

FIG. 3. *A matrix $A \in \mathbb{R}^{240\times 12}$ is constructed as described in Section 4.3. Plot data averages results over 50 different random matrices. The left plot describes the directional convergence $|\langle x_k - x^*, v_n\rangle|$ for each algorithm. The right plot demonstrates the improved convergence rate for weighted sampling.*

We notice in Figure 3 that the modified sampling method inspired by the directional weighting scheme improves the convergence in the direction of the least singular vector. It does not improve the directional convergence in any of the other singular vectors. However, this is expected as we determine our sampling weights using only the coefficient c_n. This method effectively targets the restricting factor of (1.2) by sampling each vector with a probability proportional to the magnitude of its component in the direction of v_j. Figure 3 also reveals that this weighted sampling distribution, on average, improves the overall convergence rate of the algorithm. Further plotting would reveal that x_k converges to x^* within machine error ($\epsilon_{machine} = 1 \times 10^{-15}$) in less than half the number of iterations when using weighted sampling compared to uniform.

Although this method requires knowledge of the singular value decomposition of A, the trend it unveils should be used to inspire methods that improve existing RK algorithms for ill-conditioned systems. We propose some solutions that leverage these properties in Section 5.

5. Future Work. We remind the reader that the weighted sampling method proposed in Section 4 requires the knowledge of the SVD of the matrix beforehand. Since the process of obtaining the SVD is, itself, expensive and would render the problem trivial, we suggest approximating the singular vectors v_j of A iteratively. We suggest two possible approaches: (i) iteratively adapt sampling weights as the singular vectors are approximated, or (ii) perform Average Block Kaczmarz (ABK) [1] with β many rows and adjust the projection weight of each row proportionally to the magnitude of its component $c_n^{(i)}$ relative to the other rows'.

REFERENCES

[1] K. Du, W.-T. Si, and X.-H. Sun, *Randomized extended average block kaczmarz for solving least squares*, SIAM Journal on Scientific Computing, 42 (2020), pp. A3541–A3559, https://doi.org/10.1137/20M1312629, https://doi.org/10.1137/20M1312629, https://arxiv.org/abs/https://doi.org/10.1137/20M1312629.

[2] J. Haddock and A. Ma, *Greed works: An improved analysis of sampling kaczmarz-motzkin*, 2020, https://arxiv.org/abs/1912.03544, https://arxiv.org/abs/1912.03544.

[3] J. D. Loera, J. Haddock, and D. Needell, *A sampling kaczmarz-motzkin algorithm for linear feasibility*, 2019, https://arxiv.org/abs/1605.01418, https://arxiv.org/abs/1605.01418.

[4] S. Steinerberger, *Randomized kaczmarz converges along small singular vectors*, 2021, https://arxiv.org/abs/2006.16978, https://arxiv.org/abs/2006.16978.

[5] T. Strohmer and R. Vershynin, *A randomized kaczmarz algorithm with exponential convergence*, Journal of Fourier Analysis and Applications, 15 (2009), pp. 262–278.

Appendix A. Sampling Kaczmarz Motzkin Convergence.