An Implicit Representation and Iterative Solution of Randomly Sketched Linear Systems

Vivak Patel, Mohammad Jahangoshahi & Daniel Adrian Maldonado

Abstract

Randomized linear system solvers have become popular as they have the potential to reduce floating point complexity while still achieving desirable convergence rates. One particularly promising class of methods, random sketching solvers, has achieved the best known computational complexity bounds in theory, but is blunted by two practical considerations: there is no clear way of choosing the size of the sketching matrix a\textit{priori}; and there is a nontrivial storage cost of the projected system. In this work, we make progress towards addressing these issues by implicitly generating the sketched system and solving it simultaneously through an iterative procedure. As a result, we replace the question of the size of the sketching matrix with determining appropriate stopping criteria; we also avoid the costs of explicitly representing the sketched linear system; and our implicit representation also solves the system at the same time, which controls the per-iteration computational costs. Additionally, our approach allows us to generate a connection between random sketching methods and randomized iterative solvers (e.g., randomized Kaczmarz method). As a consequence, we exploit this connection to (1) produce a new convergence theory for such randomized iterative solvers, and (2) improve the rates of convergence of randomized iterative solvers at the expense of a user-determined increases in per-iteration computational and storage costs. We demonstrate these concepts on numerical examples on forty-nine distinct linear systems.

1 Introduction

The solution of linear systems continues to be of interests across disciplines owing to their central role in a variety of algorithms from solving partial differential equations to training machine learning models. Over the past few decades, randomized linear system solvers have become popular as they have the potential to reduce floating point complexity or maintain limited memory footprints, while still achieving desirable convergence rates [e.g., Strohmer and Vershynin, 2009, Woodruff, 2014]. In particular, the noniterative class of randomized linear system solvers, based on random matrix sketching [see Woodruff, 2014], have exceptionally low computational complexities, at least in theory.

Unfortunately, the theoretical promise of these random matrix sketching solvers is blunted by their practical limitations: there is no clear way of choosing the size of the sketching matrix and there is a nontrivial storage cost of the projected system [Mahoney, 2016]. Indeed, when these random matrix sketching solvers are embedded in iterative methods, these practical issues become exacerbated. In fact, the practical challenges of random matrix sketching solvers have prevented them from being fully embraced by the numerical optimization community [e.g., Nocedal, 2018].

In this work, we begin to address these two primary practical issues of random matrix sketching, which we recall are: the challenge of choosing the size of the sketching matrix, and the challenge of storing the projected system. Our main insight is to recast the separate sketch-\textit{then}-solve core of random sketching methods into an equivalent, iterative sketch-\textit{and}-solve, in which the sketching matrix is generated incrementally without being explicitly stored and the system is incrementally solved from the implicitly derived sketched matrix.\footnote{It is worth mentioning that the random sketch solvers have been used iteratively in a different sense [e.g., see}
As a result of our approach, (1) we can implicitly grow the size of the sketching matrix until a user-determined stopping criteria is reached without having to determine the size of the sketching matrix apriori; (2) we implicitly represent the sketched system without having to explicitly store the projected system, which allows us to avoid the cost of storing the projected system; and (3) we can naturally implement random sketching solvers within distributed and parallel computing paradigms.

Thus, our approach of converting the usual sketch-then-solve procedure to a sketch-and-solve procedure begins to address the aforementioned practical challenges of random matrix sketching. Moreover, our approach provides a bridge between the newer concerns around sketching-based solvers and more classical areas of applied mathematics research such as stopping criteria. One such bridge is the placement of random sketching methods and (what we will call) base randomized iterative methods\(^2\) on a single spectrum of procedures, which has several immediate consequences.

First, the number of rows of the sketching matrix that results in the most precise solution (this number is a random quantity) is intimately related to geometric properties of the original system. As a result, this random integer encodes important information about the performance of base randomized iterative methods. To be specific, we use this information to provide an alternative rate-of-convergence result for general base randomized iterative methods that guarantees a rate-of-convergence less than one along a random subsequence of the iterates that bound the convergence of the whole sequence (Theorem 6). Our result complements the rate-of-convergence result of Gower and Richtrik [2015] (see Table 1 and Lemma 4.2), which only guarantees that the rate of convergence is less than or equal to one for these general base methods.

Second, we can generate a series of “intermediate” procedures between sketching methods and base methods that trade-off between computational resources (e.g., floating-point operations, storage) and rates of convergence. Thus, we can take a sketching method and reduce its computational footprint in exchange for a slower rate of convergence, or increase the computational footprint of base methods to improve their rate of convergence (Algorithm 2). We note that while we present this algorithm, we will leave its analysis and further improvement to future work.

Finally, by shifting our perspective from improving the sketch-then-solve procedure to improving the performance of base methods, we find that our approach is a randomized orthogonalization procedure in the row space of the coefficient matrix of the linear system. Thus, by presenting our approach from this latter perspective, we will simplify the introduction and the related theory of our approach. Now, before pursuing this further, we reiterate our main contributions.

1. First, we turn the typical sketch-then-solve noniterative random sketching solver into an iterative, sketch-and-solve method, which lays a foundation for addressing the previously enumerated practical challenges of random sketching solvers: there is no clear way of choosing the size of the sketching matrix apriori; and there is a nontrivial storage cost of the projected system.

2. Second, through our approach, we place random sketching methods and base randomized iterative methods (e.g., randomized Kaczmarz, randomized Gauss-Seidel, and the general method studied by Gower and Richtrik [2015]) on a single spectrum of methods.

3. Third, owing to this connection, we are able to generate “intermediate” methods between random sketching and base methods, which can trade-off between computational resources and rates of convergence.

4. Fourth, owing to this connection, we use the geometric implications of random sketching methods to develop an alternative rate-of-convergence result for general base methods that guarantees a convergence rate less than one, and complements the recent results of Gower and Richtrik [2015].

\[^2\]We will be more precise about what we refer to as base methods. For now, such methods are exemplified by randomized Kaczmarz [Strohmer and Vershynin, 2009] and randomized Gauss-Seidel [Leventhal and Lewis, 2010].
The remainder of this paper is organized as follows. In Section 2, we introduce our procedure heuristically as an orthogonalization procedure for base methods; we state the connection between our procedure and random sketching methods, which allows us to convert the less practical sketch-then-solve approach to our sketch-and-solve approach; and, finally, we introduce our general algorithm and variants for low-memory environments, shared memory environments, distributed memory environments, and large, sparse, structured linear systems. In Sections 3 and 4, we develop the convergence theory for the two methodological extremes—sketching and base methods—leaving the intermediate, more complex cases to future work, and discuss particular examples. In Section 5, we test our algorithms on forty-nine distinct linear systems. In Section 6, we conclude this work and preview future efforts.

2 Our Procedure

While our motivating application is to address the practicality of random sketching methods, our approach is best introduced from the perspective of base randomized iterative methods. Here, we review the basic formulation of randomized iterative methods (Subsection 2.1), which we then use to heuristically introduce our general procedure (Subsection 2.2). We then refine our procedure for the case of rank-one methods, such as Randomized Kaczmarz and Randomized Gauss-Seidel, which allows us to restate random sketching from a sketch-then-solve procedure to a sketch-and-solve procedure (Subsection 2.3). We conclude this section with comments on algorithmic refinements for parallel platforms (Subsection 2.4.1), limited memory platforms (Subsection 2.4.2), and, for structured linear systems, limited communication platforms (Subsection 2.4.3).

2.1 A Brief Overview

Let \( A \in \mathbb{R}^{n \times d} \) and \( b \in \mathbb{R}^n \) be the coefficient matrix and constant vector, respectively. Assuming consistency, our goal is to determine an \( x^* \in \mathbb{R}^d \), not necessarily unique, such that

\[
Ax^* = b. \tag{1}
\]

In a base randomized iterative approach, a sequence of iterates \( \{x_k : k \in \mathbb{N}\} \) is generated that has the form

\[
x_{k+1} = x_k + V_k (b - Ax_k), \tag{2}
\]

where \( V_k \in \mathbb{R}^{d \times n} \) are independent random variables, which we call residual projection matrices (RPM). The RPM defines the base technique which is being used. To make this formulation concrete, we give two examples of randomized iterative methods that have this formulation, and refer to (2.7) in Gower and Richtrik [2015] for many more.

**Randomized Kaczmarz.** Let \( A_i \in \mathbb{R}^d \) denote the \( i^{th} \) row of \( A \) and let \( e_i \) denote the \( i^{th} \) standard basis vector of dimension \( n \). Define the random variable \( I \) such that

\[
\mathbb{P}[I = i] = \begin{cases} \frac{\|A_i\|^2_F}{\|A\|^2_F} & i = 1, \ldots, n \\ 0 & \text{otherwise}\end{cases}.
\]

Now, given an independent copy of \( I \) at each \( k \), define the RPM, \( V_k = A_I, e'_I / \|A_I\|^2_2 \). Then, using (2),

\[
x_{k+1} = x_k + A_I, e'_I (b - Ax_k) = x_k + A_I (b_I - A'_I, x_k) / \|A_I\|^2_2,
\]

which is the Randomized Kaczmarz method of Strohmer and Vershynin [2009].
Randomized Gauss-Seidel. Let $A_j \in \mathbb{R}^n$ denote the $j^{th}$ column of $A$ and let $f_j$ denote the $j^{th}$ standard basis vector of dimension $d$. Define a random variable $J$ such that

$$
P[J = j] = \begin{cases} \frac{||A_j||_F^2}{\|A\|_F^2} & j = 1, \ldots, d \\ 0 & \text{otherwise} \end{cases}$$

Now, given an independent copy of $J$ at each $k$, define the RPM, $V_k = e_j A_j' / ||A_j||_2^2$. Then, using (2),

$$x_{k+1} = x_k + e_j A_j' (b - A x_k) / ||A_j||_2^2,$$

which is the Randomized Gauss-Seidel method of Leventhal and Lewis [2010].

2.2 A Heuristic Derivation

Here, given a strategy for defining $\{V_k : k + 1 \in \mathbb{N}\}$, we consider how to augment the randomized iterative method with prior information in order to improve convergence. For this purpose, we propose defining a sequence of matrices $\{M_k : k + 1 \in \mathbb{N}\} \subset \mathbb{R}^{d \times d}$ (discussed below) and modify (2) to be

$$x_{k+1} = x_k + M_k V_k (b - A x_k).\quad (3)$$

Of course, $M_k$ can simply be absorbed by $V_k$; however, our goal is to augment a randomized iterative method. For this reason, we will keep these two quantities separate.

The main question now is how to choose $\{M_k : k + 1 \in \mathbb{N}\}$. Our guiding principle is that $M_k$ should minimize some measure of error between $x_{k+1}$ and $x^*$. However, implementing this guiding principle requires (1) choosing an appropriate error measure and (2) handling the fact that $x^*$ is unknown. In order to convey the intuition behind our procedure, we now state the heuristics that we use to make these choices.

Choosing an Error Measure. Temporarily, suppose $x^*$ is known, and suppose we choose the $l^1$ error as our measure. Then, we must minimize the difference between the next iterate and $x^*$. While this error metric might have merit, solving it is a convex optimization problem that is as difficult to solve as the original linear system. Therefore, we will need an error measure which gives an explicit representation for $M_k$. Hence, one sensible choice is to use the Mahalanobis norm,

$$\|x_{k+1} - x^*\|_B^2,\quad (4)$$

where $B$ is a positive definite, symmetric $\mathbb{R}^{d \times d}$ matrix.

Compensating for the Unknown Solution. Now, we consider the task of compensating for the unknown $x^*$. For a fixed $x^*$ and for all $k + 1 \in \mathbb{N}$, let $S_k = (x_k - x^*)(x_k - x^*)'$. Then, $S_{k+1}$ is related to $S_k$ by

$$S_{k+1} = (I - M_k V_k A) S_k (I - M_k V_k A)',\quad (5)$$

where we have made use of (3). Using (5), we can rewrite (4) as

$$\|x_{k+1} - x^*\|_B^2 = \text{tr} [B (I - M_k V_k A) S_k (I - M_k V_k A)'].$$

To find an optimal $M_k$, we differentiate the right hand side and set the quantity equal to zero, which, explicitly is

$$M_k (V_k A S_k A' V_k') - S_k A' V_k' = 0.\quad (6)$$

Clearly, $V_k A S_k A' V_k'$ is positive semi-definite, so the solution to such a system will be the minimizer of the original objective function. However, (6) may have many possible solutions or may fail to be consistent. In
the case of nonunique solutions, we arbitrarily choose the solution with the smallest Frobenius norm. In the case of an inconsistent system, we arbitrarily choose the solution that minimizes the Frobenius norm of the residual and has the minimal Frobenius norm. In both cases, a straightforward calculation gives

\[ M_k = S_k A' V_k' (V_k AS_k A' V_k')^\dagger, \quad (7) \]

where \( \dagger \) represents the Moore-Penrose Pseudo-inverse. Using (7) with (5), we have the following recursion

\[ S_{k+1} = S_k - S_k A' V_k' (V_k AS_k A' V_k')^\dagger V_k A_k S_k. \quad (8) \]

From (7) and (8), it is clear that if \( S_0 \) were known, then the remaining unknown quantities could be determined.

**Our Procedure.** Since \( S_0 \) is unknown, we use the following heuristic procedure instead. First, we let \( S_0 = I_d \), where \( I_d \) is the \( d \)-dimensional identity matrix. Then, we recursively define \( M_k \) and \( S_k \) according to (7) and (8). To summarize, given \( \{V_k : k + 1 \in \mathbb{N}\} \), we let \( S_0 = I_d \), let \( x_0 \in \mathbb{R}^d \), and define

\[ x_{k+1} = x_k + M_k V_k (b - Ax_k), \quad (9) \]

where

\[ M_k = S_k A' V_k' (V_k AS_k A' V_k')^\dagger; \quad (10) \]

and

\[ S_{k+1} = S_k - S_k A' V_k' (V_k AS_k A' V_k')^\dagger V_k A_k S_k. \quad (11) \]

### 2.3 Rank-One Refinements and Random Sketching

By choosing \( x_0 \in \mathbb{R}^d \) and \( S_0 = I_d \), (9)–(11) describe an orthogonal projection procedure for typical randomized iterative procedures. However, because our goal is to improve the practicality of random sketching methods, we will need to focus on a particular refinement of the general procedure that occurs when \( \{V_k\} \) are rank-one matrices, that is, when there exist pairs of vectors \( \{(v_k, w_k)\} \) such that \( V_k = v_k w_k' \) for each \( k \). In this case, (10) and (11) become

\[ M_k = \begin{cases} \frac{1}{v_k' A S_k w_k} S_k A' w_k w_k' S_k A' w_k \neq 0 \\ 0 \end{cases} \quad (12) \]

and

\[ S_{k+1} = \begin{cases} S_k - \frac{1}{w_k' A S_k w_k} S_k A' w_k w_k' A S_k & S_k A' w_k \neq 0 \\ S_k \end{cases} \quad (13) \]

Moreover, if we substitute (12) into (9), we recover

\[ x_{k+1} = \begin{cases} x_k + \frac{1}{w_k' A S_k w_k} S_k A' w_k w_k' (b - Ax_k) & S_k A' w_k \neq 0 \\ x_k \end{cases} \quad (14) \]

It follows from (13) and (14) that in the case of a rank-one RPM, the left singular vector of the RPM is not important. To give some explicit examples, recall that rank-one RPM methods include the important special cases of randomized Kaczmarz and Gauss-Seidel.
Randomized Kaczmarz with Orthogonalization. Let $A_{i} \in \mathbb{R}^{d}$ denote the $i^{th}$ row of $A$ and let $e_{i}$ denote the $i^{th}$ standard basis vector of dimension $n$. Define the random variable $I$ arbitrarily taking values in $\{1, \ldots, n\}$. Now, given an independent copy of $I$ at each $k$, the randomized Kaczmarz method has rank-one RPM, $V_k = A_{I}e_{I}'/\|A_{I}\|^{2}$. Then, using (13) and (14), the randomized Kaczmarz method with orthogonalization is

$$x_{k+1} = x_k + \frac{1}{e_{I}'A_{I}A_{I}'}S_{k}A_{I}e_{I}(b - Ax_{k})$$

$$S_{k+1} = \left(I_d - \frac{1}{e_{I}'A_{I}A_{I}'}S_{k}A_{I}e_{I}A\right)S_{k},$$

when $S_{k}A_{I}e_{I} \neq 0$, or is $x_{k+1} = x_k$ and $S_{k+1} = S_{k}$ otherwise. ■

Randomized Gauss-Seidel with Orthogonalization. Let $A_{j} \in \mathbb{R}^{n}$ denote the $j^{th}$ column of $A$ and let $f_{j}$ denote the $j^{th}$ standard basis vector of dimension $d$. Define a random variable $J$ arbitrarily taking values in $\{1, \ldots, d\}$. Now, given an independent copy of $J$ at each $k$, the randomized Gauss-Seidel method has rank-one RPM, $V_k = e_{j}A_{J}e_{J}/\|A_{J}\|^{2}$. Then, using (13) and (14), the randomized Gauss-Seidel method with orthogonalization is

$$x_{k+1} = x_k + \frac{1}{A_{J}A_{J}'}S_{k}A_{J}A_{J}e_{J}(b - Ax_{k})$$

$$S_{k+1} = \left(I_d - \frac{1}{A_{J}A_{J}'}S_{k}A_{J}A_{J}e_{J}A\right)S_{k},$$

when $S_{k}A_{J}A_{J}e_{J} \neq 0$, or is $x_{k+1} = x_k$ and $S_{k+1} = S_{k}$ otherwise. ■

Again, we see from the two preceding examples that the left singular vector of the rank-one RPM does not play a role in the updates for our procedure. As we now explain, this observation is critical for converting the impractical, noniterative randomized sketch-then-solve methods into iterative randomized sketch-and-solve methods.

Recall that the fundamental sketch-then-solve procedure is to construct a specialized matrix $M^{\text{sketch}} \in \mathbb{R}^{k \times n}$, then generate and solve the smaller, projected problem $(M^{\text{sketch}}A)x = M^{\text{sketch}}b$ [see Woodruff, 2014, Ch. 1]. The special matrix $M^{\text{sketch}}$, called the sketching matrix, can be generated in a variety of ways such as making each entry an independent, identically distributed Gaussian random variable [Indyk and Motwani, 1998], or by dividing $S$ into blocks of rows and setting the columns of $M^{\text{sketch}}$ as uniformly sampled columns (with replacement) of the appropriately-dimensioned identity matrix [Cormode and Muthukrishnan, 2005].

In order to convert the usual sketch-then-solve procedure into our sketch-and-solve procedure, we simply set $\{w_{k} : k+1 \in \mathbb{N}\} \subset \mathbb{R}^{n}$ to the rows of $M^{\text{sketch}}$, which we will rigorously demonstrate in Section 3. Of course, this requires that we have a streaming procedure for generating arbitrarily many rows of $M^{\text{sketch}}$. For the Gaussian strategy [Indyk and Motwani, 1998] and the sparse Count-Sketch strategy [Cormode and Muthukrishnan, 2005], this is a trivial task. Thus, if we let $\text{RPMStrategy()}$ define a generic user-defined procedure for choosing $\{w_{k} : k+1 \in \mathbb{N}\}$, then this observation gives us Algorithm 1 for (1) converting the sketch-then-solve procedure into a sketch-and-solve procedure, and (2) adding orthogonalization to such base methods as randomized Kaczmarz and randomized Gauss-Seidel.

2.4 Algorithmic Refinements Considering the Computing Platform

Algorithm 1 implicitly assumes the traditional sequential programming paradigm. However, the performance of the algorithm can be improved by taking advantage of parallel computing architectures. Here, we will

---

3We note that the typical formulation considers linear regression rather than a linear system.
**Algorithm 1: Rank-One RPM Method**

**Data:** Initialization $x_0$, RPMStrategy() for $w_0, w_1, \ldots$, TerminationCriteria()

**Result:** Estimate $\hat{x}$

$k \leftarrow 0$

$S \leftarrow I_d$

**while** TerminationCriteria() == false **do**

\[ w_k \leftarrow \text{RPMStrategy}() \]

\[ q_k \leftarrow A'w_k \]

\[ u_k \leftarrow S_k q_k \]

**// Check if $S_k A' w_k = 0$**

**if** $u_k == 0$ **then**

\[ k \leftarrow k + 1 \]

continue to next iteration

**end**

**// Update Iterate**

\[ r_k \leftarrow b'w_k - q'_k x_k \]

\[ \gamma_k \leftarrow u_k' q_k \]

\[ x_{k+1} \leftarrow x_k + u_k (r_k/\gamma_k) \]

**// Update Projection Matrix**

\[ S_{k+1} \leftarrow (I - \frac{1}{\gamma_k} u_k q_k') S_k \]

**// Update Iteration Counter**

\[ k \leftarrow k + 1 \]

**end**

return $x_{k+1}$

consider a handful of important computing architecture abstractions and how our procedure can adapt to different configurations. In **Subsection 2.4.1**, we will consider the case of a parallel computing architecture for which the communication overhead, which is proportional to the dimension $d$, is not a limiting factor. For this subsection, the problems that we have in mind come from data and imaging sciences, where $n \gg d$ and $d$ is reasonably sized. In **Subsection 2.4.2**, we consider a similar class of problems where the communication of $O[d]$-sized vectors is acceptable and $n \gg d$, but that $d$ is so large that storing and manipulating a matrix in $\mathbb{R}^{d \times d}$ is burdensome. Finally, in **Subsection 2.4.3** we will consider problems in which computational overhead becomes a bottleneck for scalability, but that we have structured systems that will allow us to circumvent this issue. For this ultimate subsection, the problems that we have in mind here come from the solution of systems of differential equations [e.g., Dongarra and Srensen, 1986].

### 2.4.1 Asynchronous Parallelization on Shared and Distributed Memory Platforms

First, when we are using a matrix sketch for RPMStrategy(), one of the expensive components of the computation is determining $[A \ b]' w_k$. Fortunately, in our sketch-and-solve procedure, this expensive computation can be trivially asynchronously parallelized on a shared memory platform when

1. the data within the rows $[A \ b]$ are stored together, and

2. the RPMStrategy() generates $\{w_k : k+1 \in \mathbb{N}\}$ that are either independent (e.g., the Gaussian Strategy) or can be grouped into independent subsets (e.g., the Count-Sketch strategy).
When these two requirements are met, each processor can generate its own \( \{ w_k : k + 1 \in \mathbb{N} \} \) independently of the other processors, and evaluate \( \left[ A \ b \right] w_k \). It can then simply write the resulting row to an address reserved for performing the iterate and \( S_k \) matrix updates by the master processor. Importantly, this procedure does not require locking any of the rows of \( \left[ A \ b \right] \), and the reserved addresses can use fine grained locks to prevent any wasted calculations.

Similarly, in our sketch-and-solve procedure, computing \( \left[ A \ b \right] w_k \) can be trivially asynchronously parallelized on a distributed memory platform using a Fork-join model, when

1. the rows of \( \left[ A \ b \right] \) are distributed across the different storages, and
2. the \texttt{RPMStrategy()} generates \( \{ w_k : k + 1 \in \mathbb{N} \} \) such that \( w_k \) have independent groups of components (e.g., the Gaussian Strategy and the Count-Sketch strategy).

When these two requirements are met, each processor can generate its own \( \{ w_k : k + 1 \in \mathbb{N} \} \) and operate on the local rows of \( \left[ A \ b \right] \). It can then simply pass the resulting row to the master processor which performs the iterate and \( S_k \) matrix updates. For each iteration, a scattering and gathering of the data is performed but no other data exchange is required.

Table 1 summarizes the time and total computational costs of computing \( x_k \) and \( S_k \) from \( x_0 \) and \( S_0 \) in the following context: (1) the sequential platform refers to the case where there is a single processor with a sufficiently large memory to store the system, and perform the necessary operations in Algorithm 1; (2) the shared memory platform assumes that there are \( p + 1 \) processors that share a sufficiently large memory. One of the processors is dedicated to performing the iterate and matrix updates, while the remaining \( p \) processors compute \( \left[ A \ b \right] w_k \); (3) the distributed memory architecture assumes that there are \( p + 1 \) processors each with a sufficient memory capacity. The rows of \( \left[ A \ b \right] \) are split evenly or nearly evenly amongst \( p \) of the processors, and each process only manipulates its local information about \( A \) and \( b \). Finally, master processor is dedicated to performing the iterate and matrix updates.

| Platform               | Computing \( \left[ A \ b \right] w_k \) | Update Costs | Network |
|------------------------|------------------------------------------|--------------|---------|
|                        | Time | Total Effort | Iterate | Matrix |         |
| Sequential             | \( \mathcal{O}[knd] \)                   | \( \mathcal{O}[knd] \) | \( \mathcal{O}[kd^2] \) | \( \mathcal{O}[kd^3] \) | No   |
| Shared Memory          | \( \mathcal{O}[knd/p] \)                | \( \mathcal{O}[knd] \) | \( \mathcal{O}[kd^2] \) | \( \mathcal{O}[kd^3] \) | No   |
| Distributed Memory     | \( \mathcal{O}[knd/p^2] \)              | \( \mathcal{O}[knd/p] \) | \( \mathcal{O}[kd^2] \) | \( \mathcal{O}[kd^3] \) | Yes  |

Table 1: A summary of the time and total computational cost (effort) incurred by Algorithm 1 and its parallelized variants. We do not report any advantages that should be exploited when \( A \) or \( w \) are sparse. In the shared and distributed memory platforms, we assume that there are \( p \) processors dedicated to computing \( A'w \) and \( b'w \), and one processor dedicated to computing the updates. The “Network” column refers to whether communication costs over a network are incurred.

### 2.4.2 Memory-Reduced Procedure

Another notable aspect of Algorithm 1 (and its aforementioned parallel variants described above) is that it must store and manipulate the matrix \( S_k \) at each iteration, which is clearly expensive when \( d \) is large or is excessive when \( d^3 \) is comparable to \( n \) or greater than \( n \). This difficulty motivates a partial orthogonalization approach, as described in Algorithm 2. In this approach, a user-defined parameter \( m < d \) specifies the number of \( d \)-dimensional vectors needed to implicitly store an approximate representation of \( S_k \) (based
Algorithm 2: Rank-One RPM Method with Partial Orthogonalization

Data: Initialization $x_0$, RPMStrategy() for $w_0, w_1, \ldots$, TerminationCriteria(), Memory Storage Parameter $m$
Result: Estimate $\hat{x}$

$k, j \leftarrow 0, 0$
$S \leftarrow \emptyset$

while TerminationCriteria() == false do

  // Compute search direction
  Generate $w_k$
  $q_k \leftarrow A'w_k$
  Compute $u_k$ using Algorithm 3 on $q_k$ with vectors in $S$
  if $u_k == 0$ then
    $k \leftarrow k + 1$
    continue to next iteration
  end

  // Update iterate
  $r_k \leftarrow b'w_k - q'_kx_k$
  $x_{k+1} = x_k + u_kr_k/(u'_kq_k)$

  // Update Memory Storage
  $z_{k+1} \leftarrow u_k/\|u_k\|$  
  if $j == m$ then
    Remove $z_{k+1-m}$ from $S$ and append $z_{k+1}$ to $S$
  else
    Append $z_{k+1}$ to $S$
    $j \leftarrow j + 1$
  end

  // Update Iteration Counter
  $k \leftarrow k + 1$

end

return $x_{k+1}$

on Theorem 1). With this implicit representation, the cost of computing $u_k$ reduces to $O[md]^4$ which, consequently, reduces the overall cost of updating $x_k$ to $x_{k+1}$ to $O[md]$. Moreover, because $S_k$ is implicitly represented by a $m d$-dimensional vectors in $S$, there is no notable additional computational cost incurred for updating $S_k$ to $S_{k+1}$. Thus, an entire iteration incurs a computational cost $O[md]$ plus the cost of computing $[A \ b]'w$, which can be mollified under the strategies above in shared memory or distributed memory platforms.

2.4.3 Optimizing Communication Overhead. Structured Systems

In the above approaches, we take for granted that $d$ is not so large such that communicating $O[d]$ vectors is acceptable during the procedure. However, for many problems coming from the solution of differential equations [e.g., see Dongarra and Srensen, 1986], $d$ and $n$ are of the same order and are so large that communicating $O[d]$ vectors at arbitrary points during the procedure is impossible. Fortunately, linear system problems in this class are highly sparse and structured [Saad, 2003, Ch. 2]. A simple example is the case where $A$ is a square, banded system with nonzero bandwidth $\tilde{Q} + 1$ for some $\tilde{Q} \ll n = d$; that is, $A_{ij} = 0$ if $|i - j| > \tilde{Q}$ and the remaining $A_{ij}$ can take arbitrary values.

\textsuperscript{4}If $q_k$ replace $u_k$ in the calculation of $z_k$, then the cost of computing $u_k$ is $O[dm^2]$ [see Golub and Van Loan, 2012, Ch. 5.2].
Algorithm 3: Modified Gram-Schmidt

Data: Vector $q_k$, Orthonormal Set $\{ z_1, \ldots, z_{k-1} \}$

Result: Projection $q_k$ onto subspace orthogonal to $\{ z_1, \ldots, z_{k-1} \}$

$$j \leftarrow 0$$

$$t_0 \leftarrow q_k$$

while $j \leq k - 1$ do

$$j \leftarrow j + 1$$

$$t_j \leftarrow t_{j-1} - (z'_j t_{j-1}) z_j$$

end

return $t_k$

---

Figure 1: A representation of a $20 \times 20$ banded matrix with bandwidth $\tilde{Q} + 1 = 5$, whose rows are split across five compute nodes (represented by the dashed line). Note, the empty grid points represent zeros, while the filled grid points represent nonzero values.

For such sparse and structured problems, our methodology can be efficiently implemented across a distributed memory platform with $p$ processors under some additional qualifications. However, to understand these qualifications, let us first introduce some notation and concepts that define the communication pattern across the $p$ nodes.

Suppose somehow that we distribute the equations of our linear system of interest across $p$ nodes. Figure 1 shows how the coefficient matrix of a $20 \times 20$ banded system with bandwidth 5 can be distributed across five nodes. Note, in this example, the entries of the constant vector would be stored on the same processor as the corresponding rows of the coefficient matrix. Moreover, we need a way of tracking which components of $x$ are manipulated by each node: let $X_i$ be the set of indices of the components of $x$ with nonzero coefficients at node $i$ in the distributed system for $i = 1, \ldots, p$. In our example, $X_1 = \{1, \ldots, 6\}$, $X_2 = \{3, \ldots, 10\}$, $X_3 = \{7, \ldots, 14\}$, $X_4 = \{11, \ldots, 18\}$, and $X_5 = \{15, \ldots, 20\}$. Finally, for any vector $z$ and any set $E$ over the indices of $z$, let $z[E]$ be the vector whose elements are the elements of $z$ indexed by $E$.

From this example and from our discussion in Subsection 2.4.1 of distributing the RPMStrategy(), we can use the local rows of $A$ at Node 1 and a Gaussian sketch to generate a $q_1 \in \mathbb{R}^d$ such that $q_1[\{1, \ldots, 6\}]$ are arbitrarily valued and $q_1[\{7, \ldots, 20\}] = 0$. Thus, our vector $q_k$ is highly sparse and can be generated locally on the node. However, following Algorithm 1, the next step of computing $u_k$ requires computing the product between $S_k$ and $q_k$, which, in a naive implementation, would require storing a dense $d \times d$ matrix $S_k$ and computing a global matrix-vector product. Such a required computation raises several concerns, which we detail and address in the following enumeration.

1. Given that $d$ is relatively large to the computing environment, is storing a $d \times d$ matrix even feasible?

   Generally, the answer will be that storing such a matrix is infeasible. However, by exploiting the properties of $S_k$ (see Theorem 1), we will approximately and implicitly store $S_k$ as $S$, which is a collection of orthonormal vectors.
2. Even if we use $S$ in place of $S_k$, will the resulting implicit matrix-vector product and update of $S$ incur prohibitive communication costs? To answer these questions completely, we will need to specify how the implicit matrix-vector product will be computed and how $S$ will be stored. Here, we will compute the implicit matrix-vector product by using twice-iterated classical Gram-Schmidt (Algorithm 4), which was shown to be numerically stable in the seminal work of Giraud et al. [2005]. Owing to this calculation pattern, we can store $S$ in a distributed fashion across the $p$ processors, which we detail below along with the communication cost of the synchronization of $S$.

**Algorithm 4: Twice Iterated Gram-Schmidt**

| Data: | Vector $q_k$, Orthonormal Set $\{z_1, \ldots, z_{k-1}\}$ |
|-------|----------------------------------------------------------|
| Result: | Projection $q_k$ onto subspace orthogonal to $\{z_1, \ldots, z_{k-1}\}$ |

$t_0 \leftarrow q_k$

for $j = 0 : 1$ do

  // Compute projection onto Orthogonal Set
  // $\mathcal{P}$ is a set of vectors
  $\mathcal{P} \leftarrow \text{map}(l \mapsto (t_j^* z_l)z_l, l = 1 : k - 1)$

  // Compute orthogonal component
  // sum sums over the set $\mathcal{P}$
  $t_{j+1} \leftarrow t_j - \text{sum}(\mathcal{P})$

end

return $t_2$

To understand the costs associated with computing $u$ from the orthonormal vectors in $S$ and the vector $q$, we will characterize the support of $u$ (i.e., index set of its nonzero entries).

**Lemma 1.** Let $q \in \mathbb{R}^d$ and let $\mathcal{Q} = \{i : q[i] \neq 0\} \subset \{1, \ldots, d\}$. Let $\{z_1, \ldots, z_m\} \subset \mathbb{R}^d$ be a set of orthonormal vectors (hence, $m \leq d$), and let $\mathcal{Z}_j = \{i : z_j[i] \neq 0\} \subset \{1, \ldots, d\}$ for $j = 1, \ldots, m$. If $u$ denotes the result of Algorithm 4 applied to $q$ over the set $\{z_1, \ldots, z_m\}$ then

$$
\mathcal{U} := \{i : u[i] \neq 0\} \subset \left( \bigcup_{j \in \mathcal{O}} \mathcal{Z}_j \right) \cup \mathcal{Q}, \tag{15}
$$

where $\mathcal{O} = \{j : \mathcal{Q} \cap \mathcal{Z}_j \neq \emptyset\} \subset \{1, \ldots, m\}$.

**Proof.** Letting $Z$ denote the matrix whose columns are elements of the orthonormal set, we recall that classical Gram-Schmidt generates $u = (I_d - ZZ^t)q$. Thus, twice iterated Gram-Schmidt can be written as

$$(I_d - ZZ')(I_d - ZZ^t)q = (I_d - 2ZZ' + ZZ^t)q = (I_d - ZZ')q = u, \tag{16}$$

which is expected in exact arithmetic. Thus, we can consider classical Gram-Schmidt and ignore the iteration to compute the support of $u$. For any $l = 1, \ldots, d$,

$$u[l] = q[l] - \sum_{j=1}^k (q^j z_j)[l] = q[l] - \sum_{j \in \mathcal{O}} (q^j z_j)[l], \tag{17}$$

where we use the fact that if $j \notin \mathcal{O}$ then $q^j z_j = \sum_{l \in \mathcal{Q} \cap \mathcal{Z}_j} q[l] z_j[l] = \sum_{l \notin \mathcal{Q}} q[l] z_j[l] = 0$. For a contradiction, suppose $l \in \mathcal{U}$ such that

$$l \notin \left( \bigcup_{j \in \mathcal{O}} \mathcal{Z}_j \right) \cup \mathcal{Q}. \tag{18}$$

Then, $q[l] = 0$ and $z_j[l] = 0$ for $j \in \mathcal{O}$. Using the above formula for $u[l]$, $u[l] = 0 - \sum_{j \in \mathcal{Q}} (q^j z_j)[l] = 0$, which is a contradiction. \[\square\]
At iteration $k$, Lemma 1 states that the support of $u_k$ will depend on the support of $\{z_m, \ldots, z_1\}$, which, in turn, has elements whose support depend on (a subset of) $\{u_{k-1}, \ldots, u_0\}$. Moreover, if $\{u_{k-1}, \ldots, u_0\}$ has elements whose combined support cover $\{1, \ldots, d\}$, which will be necessary to solve the system, it is possible that the support of $u_k$ will be all of $\{1, \ldots, d\}$ (ignoring any trivial independence in the system). Thus, it appears that we will eventually have to store vectors in $S$ whose support is all of $\{1, \ldots, d\}$. Naively, we may think that we need a faithful copy of $\mathcal{S}$ at each node in the system, which incurs prohibitive communication costs as the support of $u_k$ tends to $\{1, \ldots, d\}$. While this is true, a careful inspection of Gram-Schmidt and the nonzero patterns of $q_k$ suggest a less naive approach, which we now detail.

We begin by supposing that on a processor $i \in \{1, \ldots, p\}$, only $z_j|\mathcal{X}_i|$ are stored on the node for every $j = 1, \ldots, k$. Immediately, we have eliminated the need for synchronizing all of $\mathcal{S}$ on each processor. Instead, we need only to synchronize those components of $z_j$ in $\mathcal{X}_i \cap \mathcal{X}_j$ for all $i \neq j$. Thus, we have that our synchronization costs will depend on the maximum overlap, $Q$, between two processors, which, formally, is

$$Q = \max_{i \neq j} |\mathcal{X}_i \cap \mathcal{X}_j|.$$  

(19)

Now, we can understand the precise nature of this synchronization by inspecting Algorithm 4. If for some $j = 1, \ldots, p$, $q_k|\mathcal{X}_j^r| = 0$, then

$$t_1[l] = \begin{cases} -\sum_{t=1}^{m} \left( \sum_{r \in \mathcal{X}_j} q_k[r]z_t[r] \right) z_t[l] & \forall l \in \mathcal{X}_j^r \\ q_k[l] - \sum_{t=1}^{m} \left( \sum_{r \in \mathcal{X}_j} q_k[r]z_t[r] \right) z_t[l] & \forall l \in \mathcal{X}_j \end{cases}$$

(20)

From (20), we see that we must communicate the values of $q[l]$ to all nodes $i \in \{1, \ldots, p\} \setminus \{j\}$ such that $\mathcal{X}_j \cap \mathcal{X}_i \neq \emptyset$, and we must communicate the $m$ inner products to all $p - 1$ nodes. The resulting number of floating point values that must be communicated (counting each replicate to a node individually) during the first iteration of Algorithm 4 is

$$\sum_{i \in \Omega_j \setminus \{j\}} |\mathcal{X}_j \cap \mathcal{X}_i| + m(p - 1),$$

(21)

where $\Omega_j = \{i : \mathcal{X}_i \cap \mathcal{X}_j \neq \emptyset\}$ for $j = 1, \ldots, p$ (see the notation in Lemma 1). For the second iteration of Algorithm 4, we must broadcast $m$ inner products that are partially computed (using some ordering that respects the non-associative property of floating point complexity) on each node to the remaining $p - 1$ nodes. Thus, the number of floating point values that must be communicated (counting each replicate to a processor individually) to ensure synchronization is

$$\sum_{i \in \Omega_j \setminus \{j\}} |\mathcal{X}_j \cap \mathcal{X}_i| + m(p - 1) + mp(p - 1),$$

(22)

which we can bound by

$$Q(F - 1) + m(p^2 - 1), \text{ where } F = \max_j |\Omega_j|. \quad (23)$$

Noting that $Q$ represents the maximum shared indices between two nodes and that $F$ represents the maximum number of nodes that overlap, the first term in the bound can be controlled by the ordering choice of the differential equations that generate the system, but a discussion of this topic is beyond the scope of this work. Algorithm 5 summarizes a simple version of the procedure described here. We can also modify this algorithm to the low memory context of Algorithm 1 by limiting the number of vectors that can be stored in $\mathcal{S}$.

3 Convergence Theory for Orthogonalization

Here, we will prove that the complete orthogonalization approach (i.e., Algorithm 1) converges to the solution under a variety of sampling RPM strategies. In Subsection 3.1, we establish a collection of core results that

\footnote{Note, if the combined supports of the elements of $\{u_{k-1}, \ldots, u_0\}$ do not cover all of $\{1, \ldots, d\}$, then some components of our iterates, $\{x_k\}$ will not be updated.}
Algorithm 5: Rank-One RPM Method for Limited Communication

**Data:** Initialization $x_0$, Distributed RPMStrategy() for $\{w_k\}$, Covering $\{j_k\} \subset \{1, \ldots, p\}$, TerminationCriteria(), memory storage parameter $m$

**Result:** Estimate $\hat{x}$

$k, j \leftarrow 0, 0$
$S \leftarrow \emptyset$

while TerminationCriteria() == false do

for Node $j_k$ do

// Compute search direction on node $j_k$
Generate $w_k$ from distributed RPMStrategy()
$q_k \leftarrow A'w_k$
$r_k \leftarrow b'w_k - q_k[X_{j_k}] x_k[X_{j_k}]$

// First Gram-Schmidt Iteration, note $q_k[X_{j_k}] = 0$
$I \leftarrow \text{map}(z \rightarrow q_k[X_{j_k}] z[X_{j_k}], z \in S)$
Communicate inner products in $I$ to remaining $p - 1$ nodes.
Communicate $q_k[X_{j_k} \cap X_i]$ for $i \neq j_k$.
if $u_k == 0$ then
    $k \leftarrow k + 1$
    continue to next iteration
end

end

for Each Node $j$ do

Compute $t_1[X_j]$ locally from (20)

// Second Gram-Schmidt Iteration
$I_j \leftarrow \text{map}(z \rightarrow t_1[X_j] z[X_j], z \in S)$
Synchronize inner products in $I_j$ to remaining $p - 1$ nodes for $j = 1, \ldots, p$.
$u_k[X_j] \leftarrow t_1[X_j] - \sum_{z \in S} (\sum_{l=1}^{p} t_1[X_l] z[X_l]) z[X_j]$

// Synchronize and Update $S$
Locally store $z_{k+1}[X_j] = u_k[X_j] / \|u_k\|_2$ in $S$.
end

for Node $j_k$ do

// Compute Step Size
Synchronize $\alpha_k \leftarrow r_k / (u_k[X_{j_k}] q_k[X_{j_k}])$
end

for Each Node $j$ do

// Update Iterate
$x_{k+1}[X_j] \leftarrow x_k[X_j] + \alpha_k u_k[X_j]$
end

// Update Iteration Counter
$k \leftarrow k + 1$
end

return $x_{k+1}$

are useful in characterizing the behavior of our procedure. A key feature of these core results is that they will rely on a stopping time $T$, which will depend on the random variables $\{w_k\}$. Therefore, in Subsection 3.2, we characterize $T$ under common probabilistic relationships between the elements of $\{w_k\}$. All statements hold with probability one unless stated otherwise.
3.1 Core Results

We establish two key results. First, we establish that our procedure is an orthogonalization procedure: that is, the matrices \( \{S_k\} \) project the current search direction onto a subspace that is orthogonal to previous search directions. Second, we characterize the limit point of our iterates, \( \{x_k\} \), in terms of a true solution of the linear system and the subspace generated by the rank-one RPMs, \( \{V_k\} \).

**Theorem 1.** Let \( \{w_l : l + 1 \in \mathbb{N}\} \subset \mathbb{R}^n \) be an arbitrary sequence in \( \mathbb{R}^n \), and let \( \mathcal{R}_0 = \{0\} \subset \mathbb{R}^d \) and \( \mathcal{R}_l = \text{span} \left\{ A'w_0, \ldots, A'w_{l-1} \right\} \) for \( l \in \mathbb{N} \). Now, let \( S_0 = I_d \) and \( \{S_l : l \in \mathbb{N}\} \) be defined recursively as in (13). Then, for \( l \geq 0 \), \( S_l \) is an orthogonal projection matrix onto \( \mathcal{R}_l^\perp \).

**Proof.** We will prove the result by induction. For the base case, \( l = 0 \), \( S_0 = I_d \). It follows that \( S_0 \) is an orthogonal projection onto \( \mathcal{R}_0^\perp = \mathbb{R}^d \) since \( S_0^2 = I_d^2 = I_d = S_0 \) and \( \text{range}(I_d) = \mathbb{R}^d \). Now suppose that the result holds for \( l > 0 \). If \( S_lA'w_l = 0 \) then there is nothing to show. Therefore, for the remainder of this proof, suppose \( S_lA'w_l \neq 0 \).

First, we show that \( S_{l+1} \) is a projection matrix by verifying that \( S_{l+1}^2 = S_{l+1} \) by direct calculation. Making use of the recursive definition of \( S_{l+1} \) and the induction hypothesis that \( S_l^2 = S_l \) (since \( S_l \) is a projection),

\[
S_{l+1}^2 = \left( S_l - \frac{S_lA'w_lw_l'AS_l}{w_l'AS_lA'w_l} \right) \left( S_l - \frac{S_lA'w_lw_l'AS_l}{w_l'AS_lA'w_l} \right) = \left( S_l - \frac{S_lA'w_lw_l'AS_l}{w_l'AS_lA'w_l} \right) = S_l = S_{l+1}.
\]

(24)

Second, we use the fact that a projection is orthogonal if and only if it is self-adjoint to show that \( S_{l+1} \) is an orthogonal projection. By induction, because \( S_l \) is an orthogonal projection, \( S_l' = S_l \), and so

\[
S_{l+1}' = S_l' - \frac{S_lA'w_lw_l'AS_l}{w_l'AS_lA'w_l} = S_{l+1}.
\]

(25)

Finally, let \( v \) be in the range of \( S_{l+1} \) and we can decompose \( v \) into the components \( u \) and \( y \) such that \( v = u + y \), \( 0 = u'y \) and \( y \in \mathcal{R}_{l+1} \). We will show that \( y = 0 \), which characterizes the range of \( S_{l+1} \) as being all vectors orthogonal to \( \mathcal{R}_{l+1} \). To show this note that because \( S_{l+1} \) is a projection matrix, we have that

\[
u + y = v = S_{l+1}v = S_{l+1}u + S_{l+1}y.
\]

(26)

By construction \( \mathcal{R}_l \subset \mathcal{R}_{l+1} \) and so \( u \in \mathcal{R}_l^\perp \). Using the induction hypothesis, we then have that \( S_lu = u \). Moreover, because \( u \in \mathcal{R}_{l+1}^\perp \) by induction, \( u'A'w_l = 0 \). Then, using the recursive definition of \( S_{l+1} \), we have that

\[
S_{l+1}u = S_lu - \frac{S_lA'w_lw_l'AS_lu}{w_l'AS_lA'w_l} = u - \frac{S_lA'w_lw_l'Au}{w_l'AS_lA'w_l} = u.
\]

(27)

Therefore, \( u = S_{l+1}u \) and, by (26), \( y = S_{l+1}y \). We now decompose \( y \) into \( y_1 \) and \( y_2 \) where \( y_1 \in \mathcal{R}_l \) and \( y_2 \in \mathcal{R}_l^\perp \cap \mathcal{R}_{l+1} \). By the induction hypothesis, \( \mathcal{R}_l^\perp \cap \mathcal{R}_{l+1} = \text{span} \{ S_lA'w_l \} \). Therefore, \( S_ly = y_2 \) and \( \exists \alpha \in \mathbb{R} \) such that \( y_2 = \alpha S_lA'w_l \). Finally, using the recursive formulation of \( S_{l+1} \) and \( S_ly = y = \alpha S_lA'w_l \),

\[
y = S_{l+1}y = S_ly - \frac{S_lA'w_lw_l'AS_ly}{w_l'AS_lA'w_l} = \alpha S_lA'w_l - \alpha S_lA'w_l = 0.
\]

(28)

Thus, we have shown that the range of \( S_{l+1} \) is orthogonal to \( \mathcal{R}_{l+1} \). \( \square \)
From Theorem 1, we see that our procedure is an orthogonalization procedure just like quasi-Newton methods [Nocedal and Wright, 2006, Ch. 8] and conjugated direction methods [Hestenes, 2012]. As a consequence, we have the following common and insightful characterization of the iterates of such an orthogonalization procedure.

**Corollary 1.** In addition to the setting of Theorem 1, let \( x_0 \in \mathbb{R}^d \) be arbitrary and let \( \{ x_l : l \in \mathbb{N} \} \) be defined according to (14). For any \( l \geq 0 \), \( x_{l+1} \in \text{span} \{ x_0, A'w_0, \ldots, A'w_l \} \).

**Proof.** We again proceed by induction. Because \( S_0 = I_n \), the case of \( x_1 \) follows by recursion formula, (14). Now suppose that the result holds up to some \( l > 0 \). Note, by the recursion formula

\[
x_{l+1} = x_l + \gamma S_l A'w_l,
\]

where

\[
\gamma = \begin{cases} w_l'(b-Ax_l) / w_l'AS_lA'w_l & S_lA'w_l \neq 0 \\ 0 & \text{otherwise.} \end{cases}
\]

Therefore, \( x_{l+1} \in \text{span} \{ x_l, S_l A'w_l \} \). Now, using the induction hypothesis,

\[
\text{span} \{ x_l, S_l A'w_l \} \subseteq \text{span} \{ x_0, A'w_0, \ldots, A'w_{l-1}, S_l A'w_l \}.
\]

Second, when \( S_l A'w_l = 0 \), then \( A'w_l \in \mathcal{R}_l \). Consequently,

\[
x_{l+1} \in \text{span} \{ x_0, A'w_0, \ldots, A'w_{l-1} \} = \text{span} \{ x_0, A'w_0, \ldots, A'w_l \}.
\]

Now suppose \( S_l A'w_l \neq 0 \). By Theorem 1, \( S_l \) is an orthogonal projection onto \( \text{span} \{ A'w_0, A'w_1, \ldots, A'w_{l-1} \}^\perp \). Hence, \( x_{l+1} \in \text{span} \{ x_l, S_l A'w_l \} \), which is contained in \( \text{span} \{ x_0, A'w_0, \ldots, A'w_l \} \).

**Corollary 1** demonstrates that, as is common with orthogonalization procedures, the iterates are in a subspace generated by the initial iterate and the search directions \( \{ A'w_0, \ldots, A'w_l \} \). For deterministic procedures, such a characterization is usually sufficient and the next step would be to demonstrate that the iterates are the closest points to the true solutions within the given subspace. However, for a procedure in which the subspace is randomly generated, there is substantially more nuance. In order to be conscientious, we will not go through the litany of issues, but rather skip to the appropriate definitions and characterizations.

First, we begin by defining the maximal possible subspace that can be generated by a random quantity \( A'w \). Let \( w \in \mathbb{R}^n \) be a random variable defined on a space \( \Omega \), and let

\[
\mathcal{N}(w) = \text{span} \{ z \in \mathbb{R}^d : \mathbb{P} [ z' A' w = 0 ] = 1 \} \text{ and } \mathcal{R}(w) = \mathcal{N}(w)^\perp.
\]

Moreover, we define the subspace \( \mathcal{V}(w) \) such that \( \mathcal{V}(w) \perp \mathcal{R}(w) \) and \( \mathcal{V}(w) + \mathcal{R}(w) = \text{row}(A) \) (hence, \( \mathcal{V}(w) \oplus \mathcal{R}(w) = \text{row}(A) \)). Correspondingly, let \( P_W \) denote the orthogonal projection matrix onto a subspace \( W \subset \mathbb{R}^d \). The following result characterizes \( \mathcal{R}(w) \).

**Lemma 2.** For \( \mathcal{R}(w) \) as defined in (32), \( \mathcal{R}(w) \) is the smallest subspace of \( \mathbb{R}^d \) such that \( \mathbb{P} [ A'w \in \mathcal{R}(w) ] = 1 \).

**Proof.** First, we verify that \( \mathbb{P} [ A'w \in \mathcal{R}(w) ] = 1 \). Suppose that \( \mathbb{P} [ A'w \in \mathcal{R}(w) ] < 1 \). Then,

\[
\mathbb{P} [ \exists z \perp \mathcal{R}(w) : z' A' w \neq 0 ] > 0.
\]

However, we know that for any \( z \) such that \( z \perp \mathcal{R}(w) \), \( z \in \mathcal{N}(w) \) and \( z' A' w = 0 \) with probability one, which is a contradiction. Hence, \( \mathbb{P} [ A'w \in \mathcal{R}(w) ] = 1 \).

Now suppose there is a proper subspace of \( \mathcal{R}(w) \), \( U \), such that \( \mathbb{P} [ A'w \in U ] = 1 \). Let \( U^\perp \mathcal{R}(w) \) denote the subspace orthogonal to \( U \) relative to \( \mathcal{R}(w) \). Then, \( \mathbb{P} [ z' A' w = 0 ] = 1 \) for any \( z \in U^\perp \mathcal{R}(w) \), which implies that \( U^\perp \mathcal{R}(w) \subseteq \mathcal{N}(w) \). However, since \( U^\perp \mathcal{R}(w) \subseteq \mathcal{R}(w) \perp \mathcal{N}(w) \), \( U^\perp \mathcal{R}(w) = \{ 0 \} \). Thus, \( \mathcal{R}(w) \) is the smallest subspace such that \( \mathbb{P} [ A'w \in \mathcal{R}(w) ] = 1 \).
Second, we must define when the maximal possible subspace of $A'w$ can be achieved by a sequence of random variables $\{A'w_0, \ldots, A'w_l\}$, which may or may not be related to $A'w$. Note, by not requiring a relationship between $\{A'w_0, \ldots, A'w_l\}$ and $A'w$ our next result is particularly general and applies to a variety of situations, from the case in which $\{w_l\}$ are independent copies of $w$ to the case where $\{w_l\}$ have complex dependencies. Now, let $\{w_l : l + 1 \in \mathbb{N}\} \subset \mathbb{R}^n$ be random variables defined on $\Omega$, and let $T$ be a stopping time defined by

$$T = \min\{k \geq 0 : \text{span} [A'w_0, \ldots, A'w_k] \supset \mathcal{R}(w)\}.$$  \hfill (34)

Using this notation, we have the following fundamental characterization result of the limit points of $\{x_l\}$.

**Theorem 2.** Let $w$ be a random variable, and let $\mathcal{R}(w)$, $\mathcal{N}(w)$ and $\mathcal{V}(w)$ be as defined above (see (32)). Moreover, let $w_0, w_1, \ldots \in \mathbb{R}^n$ be random variables such that $\mathbb{P}[A'w_l \in \mathcal{R}(w)] = 1$ for all $l + 1 \in \mathbb{N}$, and let $T$ be as defined in (34). Let $x_0 \in \mathbb{R}^d$ be arbitrary and $S_0 = I_d$, and let $\{x_l : l \in \mathbb{N}\}$ and $\{S_l : l \in \mathbb{N}\}$ be defined as in (13) and (14). On the event $\{T < \infty\}$,

1. For any $s \geq T + 1$, $S_{T+1} = S_s$ and $x_{T+1} = x_s$.

2. If $Ax = b$ admits a solution $x^*$, then

$$x_{T+1} = P_{\mathcal{N}(w)}x_0 + P_{\mathcal{R}(w)}x^*.$$  \hfill (35)

**Proof:** Recall that $\mathcal{R}_{k+1} = \text{span}[A'w_0, \ldots, A'w_k]$. Therefore, by the definition of $T$, $\mathcal{R}_{T+1} = \mathcal{R}(w)$ on the event that $\{T < \infty\}$. Therefore, by Theorem 1, $S_{T+1}$ is an orthogonal projection onto $\mathcal{N}(w)$ and its null space is $\mathcal{R}(w)$.

We now proceed by induction. Because $\ker(S_{T+1}) = \mathcal{R}(w)$ and $A'w_{T+1} \in \mathcal{R}(w)$ with probability one (by hypothesis), $S_{T+1}A'w_{T+1} = 0$. Therefore, by the recursion equations, (13) and (14), $S_{T+2} = S_{T+1}$ and $x_{T+2} = x_{T+1}$. Suppose now that $S_{T+1} = S_{T+1}$ and $x_{T+1} = x_{T+1}$ for $l > 1$. Again, by hypothesis, $A'w_{T+l} \in \mathcal{R}(w) = \ker(S_{T+1})$. Therefore, $S_{T+l}A'w_{T+l} = 0$. By the recursion equations, (13) and (14), $S_{T+l+1} = S_{T+l} = S_{T+1}$ and $x_{T+l+1} = x_{T+1} = x_{T+1}$.

To establish the second part of the result, we must first establish that for any $l \geq 0$,

$$x_{l+1} - x^* = S_{l+1}(x_0 - x^*).$$  \hfill (36)

We will prove this by induction. For $l = 0$,

$$x_1 - x^* = x_0 - x^* + \frac{S_0A'w_0w_0'(A^* - Ax_0)}{w_0'AS_0Aw_0}$$

$$= \left(I_d - \frac{S_0A'w_0w_0'A}{w_0'AS_0Aw_0}\right)(x_0 - x^*),$$  \hfill (37)

by the recursion equations, (14). Noting that $S_0 = I_d$ and by using (13), we conclude that $x_1 - x^* = S_1(x_0 - x^*)$. Now suppose that this relationship holds for some $l > 0$. Again, using (14),

$$x_{l+1} - x^* = x_l - x^* + \frac{S_lA'w_lw_l'(A^* - Ax_l)}{w_l'AS_lAw_l}$$

$$= \left(I_d - \frac{S_lA'w_lw_l'A}{w_l'AS_lAw_l}\right)(x_l - x^*).$$  \hfill (38)

Using the induction hypothesis, $x_l - x^* = S_l(x_0 - x^*)$ and (13),

$$x_{l+1} - x^* = \left(I_d - \frac{S_lA'w_lw_l'A}{w_l'AS_lAw_l}\right)S_l(x_0 - x^*) = S_{l+1}(x_0 - x^*).$$  \hfill (39)

6 Below we will assume that $A'w \in \mathcal{R}(w)$ with probability one. If we relax this, this will change the results in a predictable manner but will require additional notation. To avoid such notation, we will leave this more general case to future work if there is a sampling case that merits it.
With this result established and noting that \( S_{T+1} \) is a projection onto \( \mathcal{N}(w) \) (i.e., \( P_{\mathcal{N}(w)} = S_{T+1} \)), on the event \( \{ T < \infty \} \),

\[
x_{T+1} = x^* + S_{T+1}(x_0 - x^*) \\
= (P_{\mathcal{N}(w)} + P_{\mathcal{R}(w)}) x^* + P_{\mathcal{N}(w)}x_0 - P_{\mathcal{N}(w)}x^* \\
= P_{\mathcal{R}(w)}x^* + P_{\mathcal{N}(w)}x_0.
\]  

(40)

With Theorem 2 in hand, the natural subsequent question is when the limit point of the iterates is actually a solution to the original system. This question is addressed in the following corollary.

Corollary 2. Under the setting of Theorem 2, on the event \( \{ T < \infty \} \), \( Ax_{T+1} = b \) if and only if \( P_{\mathcal{V}(w)}x_0 = P_{\mathcal{V}(w)}x^* \).

Proof. Recall that \( \text{row}(A) \perp \text{ker}(A) \). Because \( \mathcal{R}(w) \subset \text{row}(A) \), \( \mathcal{N}(w) = \mathcal{V}(w) + \text{ker}(A) \). Moreover, by the definition of \( \mathcal{V}(w) \subset \text{row}(A) \), \( \mathcal{V}(w) \perp \text{ker}(A) \). Therefore, \( P_{\mathcal{N}(w)} = P_{\text{ker}(A)} + P_{\mathcal{V}(w)} \). Now, using the characterization in Theorem 2,

\[
Ax_{T+1} = AP_{\text{ker}(A)}x_0 + AP_{\mathcal{V}(w)}x_0 + AP_{\mathcal{R}(w)}x^* = AP_{\mathcal{V}(w)}x_0 + AP_{\mathcal{R}(w)}x^*.
\]

(41)

Similarly, because \( I_d = P_{\text{ker}(A)} + P_{\mathcal{V}(w)} + P_{\mathcal{R}(w)} \),

\[
b = Ax^* + AP_{\mathcal{V}(w)}x^* + AP_{\mathcal{R}(w)}x^* = AP_{\mathcal{V}(w)}x^* + AP_{\mathcal{R}(w)}x^*.
\]

(42)

Setting these two quantities equal to each other, we conclude that \( Ax_{T+1} = b \) if and only if \( AP_{\mathcal{V}(w)}x^* = AP_{\mathcal{V}(w)}x_0 \). Clearly, if \( P_{\mathcal{V}(w)}x_0 = P_{\mathcal{V}(w)}x^* \) then \( Ax_{T+1} = b \). So, what we have left to show is that \( AP_{\mathcal{V}(w)}x^* = AP_{\mathcal{V}(w)}x_0 \) implies \( P_{\mathcal{V}(w)}x_0 = P_{\mathcal{V}(w)}x^* \).

Let \( A^+ \) denote the Moore-Penrose pseudo-inverse of \( A \), and recall that \( A^+A \) is a projection onto \( \text{row}(A) \). Moreover, \( \text{range}(P_{\mathcal{V}}) \subset \text{row}(A) \). Therefore, since if \( Ax_{T+1} = b \) then \( AP_{\mathcal{V}(w)}x_0 = AP_{\mathcal{V}(w)}x^* \), if \( Ax_{T+1} = b \) then

\[
P_{\mathcal{V}(w)}x_0 = (A^+A)P_{\mathcal{V}(w)}x_0 = A^+(AP_{\mathcal{V}(w)}x_0) = A^+AP_{\mathcal{V}(w)}x^* = P_{\mathcal{V}(w)}x^*.
\]

(43)

Corollary 2 provides criteria on the initial condition and on \( \mathcal{V}(w) \) to determine when our procedure will solve the linear system. However, we would rarely have a way of choosing the initial condition a priori such that the requirement of Corollary 2 holds. Thus, the alternative is to design \( w \) and \( \{ w_l \} \) so that \( \mathcal{V}(w) = \{ 0 \} \), which would guarantee that \( Ax_{T+1} = b \) on the event \( \{ T < \infty \} \). It is worth reiterating that we have made very limited assumptions about the relationships between \( w \) and \( \{ w_l \} \) and amongst \( \{ w_l \} \). This is important because it allows us to apply the preceding results to a variety of common relationship patterns between \( w \) and \( \{ w_l \} \). In the next subsection, we explore some specific relationships and whether these relationships will result in \( \mathcal{V}(w) = \{ 0 \} \).

3.2 Common Sampling Patterns

There are two general sampling patterns that emerge: random permutation sampling and independent and identically distributed sampling. The former sampling pattern is exemplified by randomly permuting the equations of the linear system. More concretely, let \( e_1, \ldots, e_n \in \mathbb{R}^n \) be the standard basis; let \( w \) be a random variable with nonzero probability on each element of the basis; let \( \{ w_l \} \) be random variables sampled from \( \{ e_1, \ldots, e_n \} \) without replacement (until the set is exhausted, then we repopulate the set with its original elements and repeat the sampling without replacement). The following statement provides a simple characterization of this sampling scheme.
Lemma 3. Let \( \{W_1, \ldots, W_N\} \subset \mathbb{R}^n \). Let \( w \) be a random variable such that

\[
\mathbb{P}[w = W_j] > 0 \quad j = 1, \ldots, N, \quad \text{and} \quad \sum_{j=1}^{N} \mathbb{P}[w = W_j] = 1.
\]

Moreover, let \( \{w_l : l+1 \in \mathbb{N}\} \) be random variables sampled from \( \{W_1, \ldots, W_N\} \) without replacement (and once the set is exhausted, we repopulate the set with its original elements and repeat sampling without replacement). Then \( T \leq N - 1 \). Moreover, \( Ax_{T+1} = b \) for every initialization if \( \text{span}[A'W_1, \ldots, A'W_N] = \text{row}(A) \), which holds if \( \text{span}[W_1, \ldots, W_N] = \mathbb{R}^n \).

Proof. First, note that \( \mathcal{N}(w) = \{z \in \mathbb{R}^d : z'A'W_j = 0, \forall j = 1, \ldots, N\} \). Therefore,

\[
\mathcal{R}(w) = \mathcal{N}(w)^\perp = \text{span}[A'W_1, \ldots, A'W_N].
\]

In turn, because \( \{w_0, \ldots, w_{N-1}\} = \{W_1, \ldots, W_N\} \), \( T \) is at most \( N - 1 \).

By Corollary 2, \( Ax_{T+1} = b \) if and only if \( P_{\mathcal{N}(w)}x_0 = P_{\mathcal{R}(w)}x^* \) where \( x^* \) satisfies \( Ax^* = b \). Now, given that \( \mathcal{R}(w) + \mathcal{V}(w) = \text{row}(A) \) and \( \mathcal{R}(w) = \text{span}[A'W_1, \ldots, A'W_N] \), if \( \text{span}[A'W_1, \ldots, A'W_N] = \text{row}(A) \) then \( \mathcal{V}(w) = \{0\} \). Therefore, \( Ax_{T+1} = b \) for any initialization. The final claim is straightforward. \( \square \)

The second sampling scheme, independent and identically distributed sampling, is exemplified by randomly sampling equations from the system with uniform discrete probability. However, we do not need to limit ourselves to sampling from a finite population of elements. As the next result shows, we can do much more.

Theorem 3. Suppose that \( w, w_0, w_1, \ldots \) are independent, identically distributed random variables. There exists a \( p_0 \in (0, 1) \) such that

\[
\inf \{\mathbb{P}[v'A'w \neq 0] : v \in \mathcal{R}(w), \|v\|_2 = 1\} \geq p_0.
\]

Moreover, \( T < \infty \) and \( \mathbb{P}[T = k] \leq (k-r)^{-1}(1-p_0)^k r \) where \( r = \dim(\mathcal{R}(w)) \) and \( k \geq r \).

Proof. First, we show that there exists \( p_0 > 0 \) such that for any nontrivial, proper subspace \( V \subseteq \mathcal{R}(w) \), \( \mathbb{P}[A'w \notin V] \geq p_0 \), which implies (46) when we take \( V \) to be the relative orthogonal compliment to the span of a unit vector \( v \in \mathcal{R}(w) \). Suppose there is no such \( p_0 \). Then, for every \( p \in (0, 1) \), there is a nontrivial subspace \( V \subseteq \mathcal{R}(w) \) such that \( \mathbb{P}[A'w \in V] \geq 1 - p \). Let \( r \) between 0 and \( \dim(\mathcal{R}(w)) \) be the smallest integer such that

\[
\sup_{V \subseteq \mathcal{R}(w), \dim[V] = r} \mathbb{P}[A'w \in V] = 1.
\]

For \( \epsilon > 0 \), let \( V_1 \subseteq \mathcal{R}(w) \) be an \( r \)-dimensional subspace with \( \mathbb{P}[A'w \in V_1] \geq 1 - \epsilon/2 \). Note, by Lemma 2, \( \mathbb{P}[A'w \notin V_1] < 1 \). Therefore, let \( V_2 \subseteq \mathcal{R}(w) \) be an \( r \)-dimensional subspace with \( \mathbb{P}[A'w \in V_2] > \mathbb{P}[A'w \in V_1] \geq 1 - \epsilon/2 \). Given that \( V_1 \) and \( V_2 \) are distinct and the inclusion-exclusion principle,

\[
\mathbb{P}[A'w \in V_1 \cap V_2] \geq \mathbb{P}[A'w \in V_1] + \mathbb{P}[A'w \in V_2] - 1 \geq 1 - \epsilon.
\]

However, this is contradicts the minimality of \( r \) since \( \epsilon > 0 \) is arbitrary and \( \dim(V_1 \cap V_2) < r \). Thus, we conclude that such a \( p_0 \) exists.

It follow from (46) that for any \( k \),

\[
\mathbb{P}[\dim(\text{span}[A'w_0, \ldots, A'w_k]) > \dim(\text{span}[A'w_0, \ldots, A'w_{k-1}]) \geq p_0.
\]
Therefore, we can bound $P[T = k]$ by a negative binomial distribution. In particular,

$$P[T = k] \leq \binom{k - 1}{r - 1} (1 - p_0)^{k-r} \leq (k - r)^{r-1}(1 - p_0)^{k-r}.$$  

\[\square\]

In light of the two preceding results, we may be convinced that there is a gap between the convergence properties between random permutation sampling and the independent and identically distributed sampling. However, by modifying the structure of the rank-one RPM, we can find more intermediate cases. The next result demonstrates this behavior with a somewhat contrived example, and we will leave more complex cases to future work.

**Theorem 4.** Suppose $w, w_0, w_1, \ldots$ are i.i.d. random variables such that the entries of $A'w$ are independent, identically distributed subgaussian random variables with mean zero and unit variance. Then, there exists a $p \in (0, 1)$ depending only on the distribution of the entries of $A'w$ such that $P[T = k] \geq 1 - p^k$ for $k \geq d$.

**Proof.** Let $H_k$ denote a $k \times d$ ($k \geq d$) random matrix whose entries are independent and identically distributed subgaussian random variables with zero mean and unit variance. As a consequence of Theorem 1.1 of Rudelson and Vershynin [2009], there exists a $p$ that depends on the distribution of the entries such that for all $k \geq d$, $P[\sigma_{\min}(H_k) > 0] \geq 1 - p^k$. At iteration $k$, let $W_k$ denote the matrix whose rows are given by $w_0, w_1, \ldots$. Then, by hypothesis, $W_kA$ has entries that are independent, identically distributed subgaussian random with zero mean and unit variance. Therefore, there exists a $p \in (0, 1)$ depending only on the distribution of the entries in $A'w$ such that $P[T = k] = P[\sigma_{\min}(W_kA) > 0] \geq 1 - p^k$ for $k \geq d$.

\[\square\]

### 4 Convergence Theory for Base Methods

In the previous section, we proved convergence for the complete orthogonalization method (i.e., Algorithm 1) and explored some specific sampling patterns. Here, we will consider the extreme opposite of the complete orthogonalization method: the “base” randomized iterative approach (e.g., Randomized Kaczmarz). That is, we consider Algorithm 2 in the case $m = 0$, and, we note that, in exact arithmetic and perfect communication, the results below also apply to Algorithm 5 when $m = 0$. In this case, (14) supplies the simplified iteration

$$x_{k+1} = x_k + \frac{A'w_k w_k'(b - Ax_k)}{\|A'w_k\|^2}, \quad (47)$$

which, for certain choices of $w_k$, would recover the basic randomized Kaczmarz method and randomized Gauss-Seidel method. This would suggest that we follow the approach of Strohmer and Vershynin [2009] or Gower and Richtrik [2015], however, we will take a slightly more probabilistic approach that differs from these previous analyses. One particular highlight of our approach is that we guarantee that the rate of convergence is strictly less than one, which is missing from the results of Gower and Richtrik [2015]. Our main approach is an extension of Meany’s inequality (see Subsection 4.1) combined with stopping time arguments, as derived in Subsection 4.2. We then explore some common sampling patterns in Subsection 4.3.

#### 4.1 An Extension of Meany’s Inequality

Here, we will derive an extension of Meany’s Inequality [1969], which, under a different extension, has recently been used to study the convergence rate of row-action solvers including the a block-variant of the Kaczmarz method [Bai and Liu, 2013]. We begin by stating a geometric lemma derived by Meany [1969], and follow it with the extension, which closely follows Meany’s original proof with several modifications.
Lemma 4 (Meany [1969]). Let $f_1, \ldots, f_k \in \mathbb{R}^n$ with $k \leq n$. Write $f_k = f^S + f^N$ where $f^S$ belongs to the space $S$ spanned by $f_1, \ldots, f_{k-1}$ and $f^N$ is perpendicular to $S$. Let $F$ be the matrix whose columns are $f_1, \ldots, f_{k-1}$, and let $\bar{F}$ be the matrix whose columns are $f_1, \ldots, f_k$. Then,

$$\det(F^t F) = \|f^N\|_2^2 \det(\bar{F}^t \bar{F}).$$

(48)

Theorem 5. Let $v_1, \ldots, v_k$ be unit vectors in $\mathbb{R}^n$ for some $k \in \mathbb{N}$. Let $S = \text{span}[v_1, \ldots, v_k]$. Let $F$ denote all matrices $F$ where the columns of $F$ are the vectors $\{f_1, \ldots, f_r\} \subset \{v_1, \ldots, v_k\}$ that are a maximal linearly independent subset. Then

$$\sup_{y \in S, \|y\|_2 = 1} \|Qy\|_2 \leq \sqrt{1 - \max_{F \in \mathcal{F}} \det(F^t F)},$$

where

$$Q = (I - v_k v_k^t)(I - v_{k-1} v_{k-1}^t) \cdots (I - v_1 v_1^t).$$

(49)\hspace{1cm}(50)

Proof. The proof proceeds by induction. For the case $k = 1$, both sides of the inequality are zero and so the result holds. Now suppose that the result holds for $k = j - 1$. To prove the case $k = j$, we need the following additional notation.

Let $\bar{S} = \text{span}[v_1, \ldots, v_{j-1}]$; let $\{f_1, \ldots, f_r\}$ denote a maximal linearly independent subset of the unit vectors $\{v_1, \ldots, v_{j-1}\}$; let $\bar{F}$ be the matrix whose columns are $f_1, \ldots, f_r$; and let

$$\bar{Q} = (I - v_{j-1} v_{j-1}^t)(I - v_{j-2} v_{j-2}^t) \cdots (I - v_1 v_1^t).$$

(51)

For a unit vector $y \in S$, let $y^{\bar{S}}$ denote the component of $y$ in $\bar{S}$, and let $y^N$ denote the component of $y$ orthogonal to $\bar{S}$. Moreover, let $z = Qy^{\bar{S}}$. Then, by the induction hypothesis,

$$\|z\|_2 = \|Qy^{\bar{S}}\|_2 \leq \|y^{\bar{S}}\|_2 \sqrt{1 - \det(\bar{F}^t \bar{F})}.$$ 

(52)

Similarly, write $v_j = v^{\bar{S}} + v^N$ where $v^{\bar{S}} \in \bar{S}$ and $v^N$ is perpendicular to $\bar{S}$.

Case A: Suppose that $S = \bar{S}$. If all $v_1, \ldots, v_j$ are colinear then $Q = (I - v_1 v_1^t)$ and we have reduced to the case of $k = 1$ and the result holds. Now, suppose that $\dim(\bar{S}) > 1$, then choose a vector $\tilde{v}_j \in \{v_1, \ldots, v_j\}$ that is linearly independent of the rest of the vectors in the set, and denote the remaining vectors by $\tilde{v}_1, \ldots, \tilde{v}_{j-1}$. Then, by the induction hypothesis, the result holds when applied to $\{\tilde{v}_1, \ldots, \tilde{v}_{j-1}\}$, and by construction $\text{span}[\tilde{v}_1, \ldots, \tilde{v}_{j-1}] \subset \text{span}[v_1, \ldots, v_j]$. Thus, this case becomes a special case of the following one (i.e., Case B).

Case B: Suppose that $S \supseteq \bar{S}$. Then,

$$\|Qy\|_2^2 = \|(I - v_j v_j^t)(z + y^N)\|_2^2 = (z + y^N)'(I - v_j v_j^t)(z + y^N)$$

$$= \|z\|^2_2 + \|y^N\|^2_2 + 2z' y^N - z' v_j v_j^t y^N + (v_j y^N)^2$$

$$= \|z\|^2_2 + \|y^N\|^2_2 - (z' v^{\bar{S}})z - 2z' v^{\bar{S}} y^N \|y^N\|_2 - \|y^N\|_2^2 \|y^N\|_2^2,$$

(53)\hspace{1cm}(54)\hspace{1cm}(55)

where we have made use of $v^N$ and $y^N$ are colinear, implying that their inner product is equal to the product of their norms. Finally, since $-2z' v^{\bar{S}} \leq 2|z' v^{\bar{S}}|,$

$$\|Qy\|_2^2 \leq \|z\|_2^2 + \|y^N\|_2^2 - \left(\|v^{\bar{S}}\| - \|v^N\|_2 \|y^N\|_2\right)^2.$$

(56)
Case B(1): Suppose that $\|v^N\|_2 \leq \|y^S\|_2$. Then,

$$\|Qy\|_2^2 \leq \|z\|_2^2 + \|y^N\|_2^2 - \left(\|z^\prime v^S\|_2 - \|v^N\|_2 \|y^N\|_2\right)^2 \quad \text{(by (56))}$$

$$\leq \|z\|_2^2 + \|y^N\|_2^2$$

$$\leq \|y^S\|_2^2 (1 - \text{det}(F^\prime F)) + \|y^N\|_2^2 \quad \text{(by (52))}$$

$$= \|y\|_2^2 - \|y^S\|_2^2 \text{det}(F^\prime F)$$

$$\leq 1 - \|v^N\|_2^2 \text{det}(F^\prime F)$$

( $\|y\|_2 = 1$ and $\|v^N\|_2 \leq \|y^S\|_2$)

$$\leq 1 - \text{det}(F^\prime F), \quad \text{(57)}$$

where, in the last line, we use Lemma 4 and, since $S \neq \bar{S}$, $f_{r+1} = v_j$, which, in turn, implies $f^N = v^N$.

Case B(2): Suppose that $\|v^N\|_2 > \|y^S\|_2$. Since $\|v_j\|_2 = \|y\|_2 = 1$, then $\|v^S\|_2 \leq \|y^N\|_2$. Using these inequalities and (52),

$$\|y^N\|_2 \|v^N\|_2 \geq \|v^S\|_2 \|y^S\|_2 \geq \|v^N\|_2 \|z\|_2 \geq |z^\prime v^S|.$$

Therefore,

$$\|y^N\|_2 \|v^N\|_2 - |z^\prime v^S| \geq \|y^N\|_2 \|v^N\|_2 - \|z\|_2 \|v^S\|_2 \geq 0.$$  \quad \text{(59)}

Applying this relationship to (56),

$$\|Qy\|_2^2 \leq \|z\|_2^2 + \|y^N\|_2^2 - \left(\|y^N\|_2 \|v^N\|_2 - \|z\|_2 \|v^S\|_2\right)^2$$

$$= \|z\|_2^2 \|v^N\|_2^2 + \|y^N\|_2^2 \|v^S\|_2^2 + 2 \|v^S\|_2 \|z\|_2 \|y^N\|_2 \|v^N\|_2$$

$$= \left(\|z\|_2 \|v^N\|_2 + \|y^N\|_2 \|v^S\|_2\right)^2$$

$$\leq \left(\sqrt{1 - \text{det}(F^\prime F)} \|v^N\|_2 \|y^S\|_2 + \|y^N\|_2 \|v^S\|_2\right)^2 \quad \text{(by (52))}$$

$$\leq \left(\|y^S\|_2 + \|y^N\|_2\right)^2 \left(\|v^N\|_2^2 (1 - \text{det}(F^\prime F)) + \|v^S\|_2^2\right) \quad \text{(by Cauchy-Schwarz)}$$

$$= 1 - \|v^N\|_2^2 \text{det}(F^\prime F)$$

$$= 1 - \text{det}(F^\prime F),$$

where, in the last line, we use Lemma 4 and, since $S \neq \bar{S}$, $f_{r+1} = v_j$, which, in turn, implies $f^N = v^N$.

Therefore, from Cases A, B(1) and B(2), we conclude that for all unit vectors $y \in S$ and for any $F \in F$, $\|Qy\|_2 \leq \sqrt{1 - \text{det}(F^\prime F)}$. Since the choices of $y$ and $F$ are independent of each other, the result holds. \qed

### 4.2 Main Convergence Result

Recall that $w \in \mathbb{R}^n$ is a random variable and $\{w_l : l + 1 \in \mathbb{N}\}$ is a sequence of random variables taking value in $\mathbb{R}^n$ such that $A^\prime w_l \in \mathcal{R}(w)$.\footnote{Again, we can avoid this requirement and consider set inclusions below. However, this generalization will require additional, cumbersome notation and there is no practical reason for considering this case.} We will now define a sequence of stopping times $\{\tau_l : l + 1 \in \mathbb{N}\}$ where $\tau_0 = 0$,

$$\tau_1 = \min\{k \geq 0 : \text{span} \{A^\prime w_0, \ldots, A^\prime w_k\} = \mathcal{R}(w)\}, \quad \text{(60)}$$
and, if \( \tau_{l-1} < \infty \), we define
\[
\tau_l = \min \{ k > \tau_{l-1} : \text{span} \left[ A'w_{\tau_{l-1}+1}, \ldots, A'w_k \right] = \mathcal{R}(w) \},
\] (61)
ext unless \( \tau_1 = \infty \). As an aside, it is worthwhile to note the commonalities between the definition of \( \{ \tau_l \} \) and the stopping time \( T \) from (34).

Moreover, whenever the stopping times are finite, we will define the collection, \( F_l \), for \( l \in \mathbb{N} \), that contains all matrices \( F \) whose columns are maximal linearly independent subsets of
\[
\left\{ \frac{A'w_{\tau_{l-1}+1}}{\|A'w_{\tau_{l-1}+1}\|_2}, \ldots, \frac{A'w_{\tau_l}}{\|A'w_{\tau_l}\|_2} \right\}.
\] (62)
Moreover, define
\[
\gamma_l = 1 - \max_{F \in F_l} \det(F'F).
\] (63)
Note, it follows by Hadamard’s inequality that \( \gamma_l \in [0, 1) \).

**Theorem 6.** Suppose \( Ax = b \) admits a solution \( x^* \). Let \( w \) be a random variable valued in \( \mathbb{R}^n \), and let \( \mathcal{R}(w), \mathcal{N}(w) \) and \( \mathcal{V}(w) \) be defined as above (see (32)). Moreover, let \( \{ w_l : l \in \mathbb{N} \} \) be random variables such that \( \mathbb{P}[A'w_l \in \mathcal{R}(w)] = 1 \) for all \( l \in \mathbb{N} \). Let \( x_0 \in \mathbb{R}^d \) be arbitrary and let \( \{ x_k : k \in \mathbb{N} \} \) be defined as in (47). Then, for any \( l \), on the event \( \{ \tau_l < \infty \} \),
\[
\|x_{\tau_{l+1}} - x^* - P_N(w)(x_0 - x^*)\|_2^2 \leq \left( \prod_{j=1}^l \gamma_j \right) \|P_{\mathcal{R}(w)}(x_0 - x^*)\|_2^2,
\] (64)
where \( \gamma_j \) are defined in (63) and \( \gamma_j \in [0, 1) \). Therefore, for any \( k \)
\[
\|x_k - x^* - P_N(w)(x_0 - x^*)\|_2^2 \leq \left( \prod_{j=1}^L(k) \gamma_j \right) \|P_{\mathcal{R}(w)}(x_0 - x^*)\|_2^2,
\] (65)
where \( L(k) = \max\{ l : k \geq \tau_{l+1} \} \); and where we are on the event \( \{ \tau_{L(k)} < \infty \} \).

**Proof.** From the basic iteration stated in (47), we have
\[
x_{k+1} - x^* = x_k - x^* - \frac{A'w_kw_k'A}{\|A'w_k\|^2_2}(x_k - x^*) = \left( I - \frac{A'w_kw_k'A}{\|A'w_k\|^2_2} \right)(x_k - x^*).
\] (66)
Iterating on this relationship, we conclude
\[
x_{k+1} - x^* = \left( I - \frac{A'w_kw_k'A}{\|A'w_k\|^2_2} \right) \cdots \left( I - \frac{A'w_0w_0'A}{\|A'w_0\|^2_2} \right)(x_0 - x^*).
\] (67)
Moreover, by assumption \( A'w_l \in \mathcal{R}(w) \) with probability one, which implies that \( A'w_l \perp \mathcal{N}(w) \). Therefore,
\[
x_{k+1} - x^* = P_N(w)(x_0 - x^*) + \left( I - \frac{A'w_kw_k'A}{\|A'w_k\|^2_2} \right) \cdots \left( I - \frac{A'w_0w_0'A}{\|A'w_0\|^2_2} \right) P_{\mathcal{R}(w)}(x_0 - x^*),
\] (68)
and \( P_N(w)(x_k - x^*) = P_N(w)(x_0 - x^*) \).

Note, when \( k = \tau_l \), then the span of \( \{ A'w_0, \ldots, A'w_{\tau_l} \} = \mathcal{R}(w) \). Therefore, on the event \( \tau_1 < \infty \), Theorem 5 implies that
\[
\|x_{\tau_{l+1}} - x^* - P_N(w)(x_0 - x^*)\|_2^2 \leq \gamma_l \|P_{\mathcal{R}(w)}(x_0 - x^*)\|_2^2.
\] (69)
We now proceed by induction. Suppose (64) holds for some \( l \in \mathbb{N} \). Using (68), for \( k > \tau_l \),
\[
x_k - x^* - P_{N(w)}(x_0 - x^*) = \left( I - \frac{A^w w_{k}^t A}{\|A^w w_k\|_2^2} \right) \cdots \left( I - \frac{A^w w_{\tau_l+1} w_{\tau_l+1}^t A}{\|A^w w_{\tau_l+1}\|_2^2} \right) P_{\mathcal{R}(w)}(x_{\tau_l+1} - x^*). \tag{70}
\]
Now, when \( k = \tau_{l+1} + 1 \), the conditions of Theorem 5 are satisfied. Therefore,
\[
\|x_{\tau_l+1} - x^* - P_{N(w)}(x_0 - x^*)\|_2^2 \leq \gamma_{l+1} \|P_{\mathcal{R}(w)}(x_{\tau_l+1} - x^*)\|_2^2 \\
= \gamma_{l+1} \|x_{\tau_l+1} - x^* - P_{N(w)}(x_0 - x^*)\|_2^2.
\]
By applying the induction hypothesis, we conclude that (64) holds on the event \( \{\tau_{l+1} < \infty\} \).

Now, for an orthogonal projection matrix, \( I - vv', \|I - vv'\|_2 = 1 \). The bound on \( x_k - x^* - P_N(x_0 - x^*) \) follows by applying this fact and the definition of \( L(k) \).

As an analogue of Corollary 2, we have the following characterization of whether \( \lim_{k \to \infty} x_k \) solves the system \( Ax = b \).

**Corollary 3.** Under the setting of Theorem 6, on the events \( \bigcap_{l=0}^{\infty} \{\tau_{l} < \infty\} \) and \( \{l \to \infty \prod_{j=0}^{l} \gamma_j = 0\} \), \( \lim_{k \to \infty} Ax_k = b \) if and only if \( P_{\mathcal{V}(w)} x_0 = P_{\mathcal{V}(w)} x^* \).

**Proof.** By Theorem 6, and on the events \( \bigcap_{l=0}^{\infty} \{\tau_{l} < \infty\} \) and \( \{l \to \infty \prod_{j=0}^{l} \gamma_j = 0\} \),
\[
\lim_{k \to \infty} x_k = x^* + P_{N(w)}(x_0 - x^*) = x^* + P_{\ker(A)}(x_0 - x^*) + P_{\mathcal{V}(w)}(x_0 - x^*). \tag{72}
\]
Therefore, \( \lim_{k \to \infty} Ax_k = b + AP_{\mathcal{V}(w)}(x_0 - x^*) \), which implies \( \lim_{k \to \infty} Ax_k = b \) if and only if \( AP_{\mathcal{V}(w)} x_0 = AP_{\mathcal{V}(w)} x^* \). Clearly, if \( P_{\mathcal{V}(w)} x_0 = P_{\mathcal{V}(w)} x^* \), then \( AP_{\mathcal{V}(w)} x_0 = AP_{\mathcal{V}(w)} x^* \). Now, since \( \mathcal{V}(w) \subset \text{row}(A) \), if \( AP_{\mathcal{V}(w)} x_0 = AP_{\mathcal{V}(w)} x^* \), then \( P_{\mathcal{V}(w)} x_0 = P_{\mathcal{V}(w)} x^* \) follows from (43).

### 4.3 Common Sampling Patterns

We now revisit the sampling patterns that we studied in Subsection 3.2. The first result provides a proof of convergence when we sample without replacement from a finite population. We note that the result is quite general and does not depend on the nature of the sampling without replacement or the dependency of the samples whenever the finite population is exhausted. As a result, the bounds are slightly loose, which may be unsatisfying. Should particular sampling patterns become sufficiently important to warrant a more detailed analysis, we will do so in future work.

**Theorem 7.** Let \( w \) and \( \{w_l : l + 1 \in \mathbb{N}\} \) be defined as in Lemma 3. Then, under the setting of Theorem 6, \( \tau_l - \tau_{l-1} \leq 2N \) for all \( l \in \mathbb{N} \), and \( \lim_{l \to \infty} \prod_{j=1}^{l} \gamma_j = 0 \). Moreover, \( \gamma_j \) are uniformly bounded by a \( \gamma \in [0, 1) \) that depends on \( \{AW_1, \ldots, AW_N\} \). Therefore,
\[
\|x_{2N} - x^* - P_{N(w)}(x_0 - x^*)\|_2^2 \leq \gamma^l \|P_{\mathcal{R}(w)}(x_0 - x^*)\|_2^2. \tag{73}
\]

**Proof.** Note that by the definition of \( w \) in Lemma 3, \( \mathcal{R}(w) = \text{span}[AW_1, \ldots, AW_N] \). Moreover, by the definitions of \( \{w_l\} \), we are sampling from \( W_1, \ldots, W_N \) without replacement. Then, we are guaranteed that \( \{A^t w_{\tau_{l-1} + 1}, \ldots, A^t w_{\tau_l}\} \) spans \( \mathcal{R}(w) \) if \( \{W_1, \ldots, W_N\} \subset \{w_{\tau_{l-1} + 1}, \ldots, w_{\tau_l}\} \). Now, suppose that at iteration \( \tau_{l-1} \), \( W \subset \{W_1, \ldots, W_N\} \) are exhausted. Then, to ensure that \( \{W_1, \ldots, W_N\} \) is contained in \( \{w_{\tau_{l-1} + 1}, \ldots, w_{\tau_l}\} \), we need to exhaust \( W^c \) and then the entire set \( \{W_1, \ldots, W_N\} \). Since \( |W^c| \leq N \), we need at most \( 2N \) more iterations from \( \tau_{l-1} \) to achieve \( \tau_l \). Therefore, \( \tau_l - \tau_{l-1} \leq 2N \). Now, let \( \mathcal{F} \) denote all matrices whose columns are maximal linearly independent subsets of
\[
\left\{ \frac{A^t W_1}{\|A^t W_1\|_2}, \ldots, \frac{A^t W_N}{\|A^t W_N\|_2} \right\}. \tag{74}
\]
Then, $F_l \subset F$. Therefore,

$$\gamma_l = 1 - \max_{F \in F_l} \det(F^T F) \leq 1 - \min_{F \in F} \det(F^T F) \leq 1 - \min_{F \in F} \det(F^T F) = \gamma.$$  \hspace{1cm} (75)

It is clear, by Hadamard’s inequality, that $\gamma \in [0, 1)$. Hence, $\lim_{l \to \infty} \prod_{j=1}^{l} \gamma_j \leq \lim_{l \to \infty} \gamma^l = 0$. The result follows by Theorem 6.

The next result revisits the case of independent and identically distributed sampling. The result makes intuitive sense as, for such a situation, we should expect the difference in the stopping times to be independent and identically distributed, which, results in the natural conclusion that $\gamma_l$ are also independent and identically distributed.

**Theorem 8.** Let $w$ and $\{w_l : l + 1 \in \mathbb{N}\}$ be defined as in Theorem 3. Then, under the setting of Theorem 6, $\tau_l < \infty$ for all $l \in \mathbb{N}$, and $\{\gamma_l : l \in \mathbb{N}\}$ are independent and identically distributed such that $\mathbb{E}[\gamma_1] = 1 - \mathbb{E}[\max_{F \in F_1} \det(F^T F)] < 1$. Hence, for all $l \in \mathbb{N}$,

$$\mathbb{E}\left[\left\|x_{\tau_{l+1}} - x^* - P_{N\gamma}(x_0 - x^*)\right\|^2\right] \leq \mathbb{E}[\gamma_1^l] \left\|P_{R\gamma}(x_0 - x^*)\right\|^2,$$

where $\mathbb{E}[\gamma_l] \in [0, 1)$. Moreover, $\lim_{l \to \infty} \tau_l / l = \mathbb{E}[\tau_1]$.

**Proof.** Again, our main workhorse will be Theorem 4.1.3 from Durrett [2010]. By this result, conditioned on $\tau_{l-1}$, $\{A'w_{\tau_{l-1}+1}, A'w_{\tau_{l-1}+2}, \ldots\}$ are independent and identically distributed. By this property, conditioned on $\tau_{l-1}$, $\tau_l - \tau_{l-1}$ is independent of $\tau_{l-1}$ and have the same distribution for all $l \in \mathbb{N}$. We conclude then that since $\gamma_l$ is a function of $\{A'w_{\tau_{l-1}+1}, \ldots, A'w_{\tau_l}\}$, then $\gamma_l$ are independent and identically distributed. The result follows by Theorem 6.

## 5 Numerical Experiments

Here, we present a variety of numerical experiments to study the practicality of our approach in a sequential computing environment. Specifically, we test forty-nine systems with five hundred equations and five hundred unknowns. The coefficients are generated from forty-nine built-in matrices found in the MatrixDepot package for the Julia programming language [Zhang and Higham, 2016]. The solution to the equation is then generated using a standard, multi-variate normal vector. The constant vector is generated by the product of the two. Then, using the generated coefficient matrix and the generated constant vector, we solve the systems by varying the sample-generation method (i.e., the generation of $w$ and $\{w_l\}$) and the solver. The sample generation method is either produced by the count-sketch approach, the Gaussian approach, by uniformly sampling the equations of the matrix with replacement, or by uniformly sampling the equations of the matrix without replacement. The solver is either a base method, the complete method, an intermediate method with $m = 5$, or an intermediate method with $m = 10$. Finally, we initialize $x_0 = 0$.

We recorded the wall clock time to improve the initial residual norm by a factor of ten with an upper bound of three seconds (note, a single iteration of a base method requires approximately $10^{-6}$ seconds, which allows the base method on the order of $10^6$ iterations on a $500 \times 500$ system). If the temporal upper bound is reached before a ten fold improvement in the initial residual norm is observed, the wall clock times is reported as $10^{09}$. Inherently, this metric results in a disadvantage for complete orthogonalization methods as such methods pay more for marginal improvements, but generate precise solutions with fewer iterations. However, with an eye towards solving much larger systems that require using a parallel or distributed environment, this metric of time-to-ten-fold improvement is the appropriate choice as the complete method would not be appropriate in such environments owing to the high communication costs that would be incurred. For the count-sketch sampling method, the wall clock times are reported in Table 2. For the remaining sampling approaches, the wall clock times are reported in the appendix.
## Table 2: Wall clock time for ten-fold improvement of four solvers under the count-sketch sampling approach.

| System     | Base   | Partial, $m = 5$ | Partial, $m = 10$ | Complete |
|------------|--------|------------------|-------------------|----------|
| baart      | $1.000 \times 10^{99}$ | $1.355 \times 10^{-5}$ | $1.584 \times 10^{-5}$ | $0.003$  |
| cauchy     | $1.000 \times 10^{99}$ | $4.138 \times 10^{-5}$ | $5.346 \times 10^{-5}$ | $0.029$  |
| chebspec   | $1.000 \times 10^{99}$ | $0.006$            | $0.007$           | $0.208$  |
| chow       | $1.000 \times 10^{99}$ | $0.000$            | $0.000$           | $0.042$  |
| circul     | $1.000 \times 10^{99}$ | $0.035$            | $0.021$           | $0.258$  |
| clement    | $1.000 \times 10^{99}$ | $0.031$            | $0.044$           | $1.302$  |
| companion  | $1.000 \times 10^{99}$ | $1.170 \times 10^{-5}$ | $1.730 \times 10^{-5}$ | $0.002$  |
| deriv2     | $0.136$            | $2.724 \times 10^{-5}$ | $8.065 \times 10^{-5}$ | $0.008$  |
| dingdong   | $1.000 \times 10^{99}$ | $0.023$            | $0.042$           | $1.336$  |
| erdrey     | $1.000 \times 10^{99}$ | $0.037$            | $0.067$           | $1.179$  |
| fiedler    | $1.000 \times 10^{99}$ | $0.000$            | $9.097 \times 10^{-5}$ | $0.031$  |
| forsythe   | $1.000 \times 10^{99}$ | $0.021$            | $0.040$           | $1.205$  |
| foogood    | $1.000 \times 10^{99}$ | $2.125 \times 10^{-5}$ | $5.083 \times 10^{-5}$ | $0.009$  |
| frank      | $1.000 \times 10^{99}$ | $0.025$            | $0.021$           | $0.266$  |
| gilbert    | $1.000 \times 10^{99}$ | $0.046$            | $0.076$           | $1.125$  |
| golub      | $1.000 \times 10^{99}$ | $0.029$            | $0.049$           | $0.807$  |
| gravity    | $1.000 \times 10^{99}$ | $3.325 \times 10^{-5}$ | $0.000$           | $0.024$  |
| grcar      | $1.000 \times 10^{99}$ | $0.030$            | $0.054$           | $1.296$  |
| hankel     | $1.000 \times 10^{99}$ | $0.186$            | $0.106$           | $0.540$  |
| heat       | $0.017$            | $0.001$            | $0.001$           | $0.037$  |
| hilb       | $1.000 \times 10^{99}$ | $3.475 \times 10^{-5}$ | $0.011$           | $0.006$  |
| invol      | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ |
| kahan      | $1.000 \times 10^{99}$ | $0.003$            | $0.004$           | $0.105$  |
| kms        | $1.000 \times 10^{99}$ | $0.062$            | $0.116$           | $1.314$  |
| lehmer     | $1.000 \times 10^{99}$ | $7.301 \times 10^{-5}$ | $7.385 \times 10^{-5}$ | $0.008$  |
| lotkin     | $1.000 \times 10^{99}$ | $4.020 \times 10^{-5}$ | $7.991 \times 10^{-5}$ | $0.006$  |
| magic      | $1.000 \times 10^{99}$ | $2.089 \times 10^{-5}$ | $3.537 \times 10^{-5}$ | $0.004$  |
| minij      | $1.000 \times 10^{99}$ | $5.071 \times 10^{-5}$ | $5.443 \times 10^{-5}$ | $0.012$  |
| moler      | $1.000 \times 10^{99}$ | $4.053 \times 10^{-5}$ | $4.540 \times 10^{-5}$ | $0.009$  |
| oscillate  | $1.000 \times 10^{99}$ | $0.047$            | $0.074$           | $1.174$  |
| parter     | $1.000 \times 10^{99}$ | $0.024$            | $0.040$           | $1.268$  |
| pei        | $1.000 \times 10^{99}$ | $1.093 \times 10^{-5}$ | $1.686 \times 10^{-5}$ | $0.002$  |
| phillips   | $1.000 \times 10^{99}$ | $8.462 \times 10^{-5}$ | $0.000$           | $0.022$  |
| prolate    | $1.000 \times 10^{99}$ | $8.553 \times 10^{-5}$ | $0.000$           | $0.023$  |
| randcorr   | $1.000 \times 10^{99}$ | $0.042$            | $0.075$           | $1.236$  |
| rando      | $1.000 \times 10^{99}$ | $0.088$            | $0.110$           | $1.162$  |
| randsvd    | $0.007$            | $0.012$            | $0.012$           | $0.250$  |
| rohess     | $1.000 \times 10^{99}$ | $0.021$            | $0.039$           | $1.384$  |
| sampling   | $1.000 \times 10^{99}$ | $0.053$            | $0.102$           | $1.160$  |
| shaw       | $1.000 \times 10^{99}$ | $4.764 \times 10^{-5}$ | $4.025 \times 10^{-5}$ | $0.014$  |
| smallworld | $1.000 \times 10^{99}$ | $0.038$            | $0.064$           | $1.160$  |
| spikes     | $1.000 \times 10^{99}$ | $7.608 \times 10^{-5}$ | $6.528 \times 10^{-5}$ | $0.015$  |
| toepitz    | $1.000 \times 10^{99}$ | $3.409 \times 10^{-5}$ | $7.750 \times 10^{-5}$ | $0.010$  |
| triding    | $1.000 \times 10^{99}$ | $0.040$            | $0.071$           | $1.146$  |
| triw       | $1.000 \times 10^{99}$ | $0.007$            | $0.010$           | $0.170$  |
| ursell     | $1.284 \times 10^{-5}$ | $9.278 \times 10^{-6}$ | $1.442 \times 10^{-5}$ | $0.002$  |
| vand       | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ |
| wilkinson  | $1.000 \times 10^{99}$ | $0.039$            | $0.066$           | $1.262$  |
| wing       | $6.140 \times 10^{-6}$ | $9.868 \times 10^{-6}$ | $1.342 \times 10^{-5}$ | $0.002$  |
While further analysis of each system would be necessary to understand the behavior of the solvers on each system, there are several important messages within Table 2. First, the base method often fails to achieve a ten-fold improvement despite the substantial number of iterations that the base solver is allowed (again, on the order of $10^6$). Unfortunately, the base method’s poor behavior is observed even for the other sampling methods. Based on Theorem 6, this would imply that either the stopping times $\{\tau_l\}$ are large and/or the rate of convergence (determined by $\{\gamma_l\}$) are too small. Given that this behavior is observed even for the random cyclic sampling case (which, by Theorem 7, implies that the differences between the stopping times are bounded by a thousand), it is likely that the rate of convergence for such systems is close to unity.

However, we see a tremendous benefit even from a small amount of partial orthogonalization. That is, the intermediate solvers with $m = 5$ and $m = 10$ perform quite well. In particular, whenever complete orthogonalization achieves a ten-fold improvement within the allotted time, the partial orthogonalization methods also achieve the ten-fold improvement within the allotted time and often orders of magnitude faster. Thus, for cases when the base method performs poorly, a small amount of partial orthogonalization is able to usually able to remedy this poor behavior. One final observation is that the $m = 5$ method often outperforms the $m = 10$ method. This seems to be because of the memory-management and garbage collection time related to modifying the set $S$, which we did not optimize for these experiments. Thus, a more complete study would require a detailed optimization of how $S$ is handled.

6 Conclusion

To reiterate, our motivation was to address the two practical challenges of the typical sketch-then-solve approach for solving linear systems. These practical challenges are: there is no clear way of choosing the size of the sketching matrix apriori; and there is a nontrivial storage costs of the projected system. While we do not address these challenges fully, we made progress towards them by reformulating the sketch-then-solve approach to a sketch-and-solve approach in which the sketched system is implicitly constructed and solved simultaneously. The main idea of the reformulation is to construct the equations of the sketched system one-at-a-time and then use an orthogonalization scheme to solve the system as each sketched equation is observed. As a result, we addressed the concern of determining the sketching matrix’s dimensions because, under our reformulation, the sketching matrix could be grown to an arbitrary size during the calculation up to a user-defined stopping criteria, which may be based on closeness to a solution or based on a computational budget. Moreover, we addressed the cost of storing the sketched system because we do not need to explicitly form the entire sketched system under our reformulation. However, we traded this storage problem with another one—albeit less onerous—of storing the matrix $S$. Finally, we address overlooked practical challenge of solving the sketched system by using our orthogonalization scheme to solve the implicitly sketched system under our reformulation. Again, while our algorithms supply a complete methodology, for sufficiently large systems, our orthogonalization is not practical because of the storage and floating-point arithmetic costs associated with manipulating $S$.

Because of the challenges introduced by $S$, we proposed intermediate methods that implicitly stored $S$ using only a handful of vectors. The result was a collection of partial orthogonalization schemes, and, in the limit of not storing any vectors for $S$ (i.e., $S$ becomes the identity), we recovered what we called “base methods,” which included the important special cases of randomized Kaczmarz and randomized Gauss-Seidel. As a result, we were able to make a conceptual connection between random sketching method (i.e., complete orthogonalization methods under our formulation) and the usual randomized iterative methods (i.e., base methods under our formulation). Importantly, we were able to leverage this conceptual relationship between the two to connect the convergence theory of the complete orthogonalization method to the convergence theory of the base methods. The key ingredient here is that the stopping time that was defined for the complete orthogonalization method encoded information about exploration of a subspace that contained the solution of the sketched system. This stopping time was then used (in a repeated fashion) to guarantee that a certain amount of progress for the base methods is achieved. As a result, we were able to produce
a convergence theory for these base methods that was both quite general and complemented the results of Gower and Richtrik [2015].

The predominant missing component of this work is the rigorous analysis of the so-called intermediate methods that reside between the base methods and the complete methods. Such an analysis is certainly warranted owing to the impressive numerical performance of these base methods as demonstrated in our experiments. Owing, primarily, to the additional complexity of analyzing such partial orthogonalization methods and, secondarily, of space limitations, a rigorous analysis of these methods will be the focus of future work. Additionally, an efficient implementation at scale for challenging problems arising in partial differential equations with a detailed comparison to existing state-of-the-art methods will be included in future work.

References

Zhong-Zhi Bai and Xin-Guo Liu. On the meany inequality with applications to convergence analysis of several row-action iteration methods. *Numerische Mathematik*, 124(2):215–236, 2013.

Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

Jack J Dongarra and DC Srensen. Linear algebra on high-performance computers. In Deborah F. Lockhart and Darrell L. Hicks, editors, *Applications of Supercomputers*, pages 57–88. Elsevier Science Publishing Co., 1986.

Rick Durrett. *Probability: theory and examples*. Cambridge university press, 2010.

Luc Giraud, Julien Langou, Miroslav Rozložník, and Jasper van den Eshof. Rounding error analysis of the classical gram-schmidt orthogonalization process. *Numerische Mathematik*, 101(1):87–100, 2005.

Gene H Golub and Charles F Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

Robert M Gower and Peter Richtrik. Randomized iterative methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 36(4):1660–1690, 2015.

Magnus Rudolph Hestenes. *Conjugate direction methods in optimization*, volume 12. Springer Science & Business Media, 2012.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

Dennis Leventhal and Adrian S Lewis. Randomized methods for linear constraints: convergence rates and conditioning. *Mathematics of Operations Research*, 35(3):641–654, 2010.

Michael W Mahoney. Lecture notes on randomized linear algebra. *arXiv preprint arXiv:1608.04481*, 2016.

R. K. Meany. A matrix inequality. *SIAM Journal on Numerical Analysis*, 6(1):104–107, 1969.

Jorge Nocedal. Optimization Methods for Training Neural Networks. 2018. URL https://ismp2018.sciencesconf.org/data/bookFullProgram.pdf.

Jorge Nocedal and Stephen J Wright. *Numerical optimization 2nd*. Springer, 2006.

Mark Rudelson and Roman Vershynin. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 62(12):1707–1739, 2009.

Yousef Saad. *Iterative methods for sparse linear systems*, volume 82. siam, 2003.

Thomas Strohmer and Roman Vershynin. A randomized Kaczmarz algorithm with exponential convergence. *Journal of Fourier Analysis and Applications*, 15(2):262, 2009.

David P Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(12):1–157, 2014.
A Numerical Experiments, Continued

Here, we present the wall clock time to a ten-fold improvement for the remaining sampling schemes: Gaussian (Table 3), uniformly choosing an equation with replacement (Table 4), uniformly choosing an equation without replacement (Table 5).

Again, we recorded the wall clock time to improve the initial residual norm by a factor of ten with an upper bound of three seconds (note, a single iteration of a base method requires approximately $10^{-6}$ seconds, which allows the base method on the order of $10^6$ iterations on a $500 \times 500$ system). If the temporal upper bound is reached before a ten fold improvement in the initial residual norm is observed, the wall clock times is reported as $10^{99}$. Inherently, this metric results in a disadvantage for complete orthogonalization methods as such methods pay more for marginal improvements, but generate precise solutions with fewer iterations. However, with an eye towards solving much larger systems that require using a parallel or distributed environment, this metric of time-to-ten-fold improvement is the appropriate choice as the complete method would not be appropriate in such environments owing to the high communication costs that would be incurred.
| System   | Base    | Partial, $m = 5$ | Partial, $m = 10$ | Complete |
|----------|---------|------------------|-------------------|----------|
| baart    | $1.000 \times 10^{99}$ | $7.601 \times 10^{-6}$ | $1.284 \times 10^{-5}$ | 0.002    |
| cauchy   | $1.000 \times 10^{99}$ | $2.317 \times 10^{-5}$ | $3.810 \times 10^{-5}$ | 0.006    |
| chebspec | $1.000 \times 10^{99}$ | 0.001             | 0.001             | 0.103    |
| chow     | $1.000 \times 10^{99}$ | 0.011             | 0.015             | 0.223    |
| circul   | $1.000 \times 10^{99}$ | 0.005             | 0.004             | 0.174    |
| clement  | $1.000 \times 10^{99}$ | 0.030             | 0.049             | 1.208    |
| companion| $1.000 \times 10^{99}$ | $1.218 \times 10^{-5}$ | $1.681 \times 10^{-5}$ | 0.002    |
| deriv2   | 0.000   | $3.919 \times 10^{-5}$ | $6.436 \times 10^{-5}$ | 0.016    |
| dingdong | $1.000 \times 10^{99}$ | 0.028             | 0.040             | 1.227    |
| erdrey   | $1.000 \times 10^{99}$ | 0.054             | 0.082             | 1.129    |
| fiedler  | $1.000 \times 10^{99}$ | $4.624 \times 10^{-5}$ | $7.422 \times 10^{-5}$ | 0.016    |
| forsythe | $1.000 \times 10^{99}$ | 0.028             | 0.042             | 1.264    |
| foxgood  | $1.011 \times 10^{-5}$ | $2.194 \times 10^{-5}$ | $3.336 \times 10^{-5}$ | 0.005    |
| frank    | $1.000 \times 10^{99}$ | 0.005             | 0.011             | 0.157    |
| gilbert  | $1.000 \times 10^{99}$ | 0.049             | 0.082             | 1.232    |
| golub    | $1.000 \times 10^{99}$ | 0.043             | 0.063             | 0.754    |
| gravity  | $1.000 \times 10^{99}$ | $4.650 \times 10^{-5}$ | $7.896 \times 10^{-5}$ | 0.017    |
| grcar    | $1.000 \times 10^{99}$ | 0.040             | 0.066             | 1.282    |
| hankel   | $1.000 \times 10^{99}$ | 0.028             | 0.041             | 0.475    |
| heat     | 0.009   | 0.001             | 0.001             | 0.053    |
| hilb     | $1.000 \times 10^{99}$ | $4.865 \times 10^{-5}$ | $7.846 \times 10^{-5}$ | 0.015    |
| invol    | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ |
| kahan    | $1.000 \times 10^{99}$ | 0.004             | 0.001             | 0.086    |
| kms      | $1.000 \times 10^{99}$ | 0.058             | 0.090             | 1.134    |
| lehmer   | $1.000 \times 10^{99}$ | 0.000             | 0.000             | 0.028    |
| lotkin   | $1.000 \times 10^{99}$ | $1.165 \times 10^{-5}$ | $1.836 \times 10^{-5}$ | 0.002    |
| magic    | $1.000 \times 10^{99}$ | $2.377 \times 10^{-5}$ | $3.534 \times 10^{-5}$ | 0.004    |
| minij    | $1.000 \times 10^{99}$ | $2.975 \times 10^{-5}$ | $4.727 \times 10^{-5}$ | 0.008    |
| moler    | $1.000 \times 10^{99}$ | $2.064 \times 10^{-5}$ | $3.265 \times 10^{-5}$ | 0.006    |
| oscillate| $1.000 \times 10^{99}$ | 0.059             | 0.085             | 1.233    |
| parter   | $1.000 \times 10^{99}$ | 0.032             | 0.050             | 1.313    |
| pei      | $1.000 \times 10^{99}$ | 0.036             | 0.054             | 1.159    |
| phillips | $1.000 \times 10^{99}$ | $5.262 \times 10^{-5}$ | $8.703 \times 10^{-5}$ | 0.018    |
| prolate  | $1.000 \times 10^{99}$ | 0.001             | 0.000             | 0.019    |
| randcorr | $1.000 \times 10^{99}$ | 0.039             | 0.068             | 1.166    |
| rando    | $1.000 \times 10^{99}$ | 0.031             | 0.046             | 1.015    |
| randsvd  | $1.000 \times 10^{99}$ | 0.012             | 0.016             | 0.221    |
| rohess   | $1.000 \times 10^{99}$ | 0.024             | 0.041             | 1.221    |
| sampling | $1.000 \times 10^{99}$ | 0.068             | 0.096             | 1.127    |
| shaw     | $1.000 \times 10^{99}$ | $1.683 \times 10^{-5}$ | $2.702 \times 10^{-5}$ | 0.005    |
| smallworld| $1.000 \times 10^{99}$ | 0.047             | 0.087             | 1.122    |
| spikes   | $1.000 \times 10^{99}$ | $3.809 \times 10^{-5}$ | $7.089 \times 10^{-5}$ | 0.013    |
| toeplitz | $1.000 \times 10^{99}$ | $4.664 \times 10^{-5}$ | $7.507 \times 10^{-5}$ | 0.020    |
| tridiag  | $1.000 \times 10^{99}$ | 0.029             | 0.041             | 0.980    |
| triw     | $1.000 \times 10^{99}$ | 0.044             | 0.043             | 0.939    |
| ursell   | $3.616 \times 10^{-5}$ | $1.883 \times 10^{-5}$ | $2.988 \times 10^{-5}$ | 0.007    |
| vand     | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ | $1.000 \times 10^{99}$ |
| wilkinson| $1.000 \times 10^{99}$ | 0.026             | 0.040             | 1.162    |
| wing     | $3.492 \times 10^{-5}$ | $1.233 \times 10^{-5}$ | $1.750 \times 10^{-5}$ | 0.002    |

Table 3: Wall clock time for ten-fold improvement of four solvers under the Gaussian sampling approach.
Table 4: Wall clock time for ten-fold improvement of four solvers when equations are sampled uniformly with replacement.
Table 5: Wall clock time for ten-fold improvement of four solvers when equations are sampled uniformly without replacement.