Adaptive s-step GMRES with randomized and truncated low-synchronization orthogonalization

Robert Ernstbrunner
University of Vienna
Faculty of Computer Science
Vienna, Austria
robert.ernstbrunner@univie.ac.at

Wilfried N. Gansterer
University of Vienna
Faculty of Computer Science
Vienna, Austria
wilfried.gansterer@univie.ac.at

***Abstract*—Iterative solvers for large, sparse linear systems are widely used on HPC machines. When solving very large problems, communication poses a significant bottleneck, which has prompted the development of communication-avoiding iterative s-step methods. At the same time, randomization has had a profound impact on numerical linear algebra, leading to orders-of-magnitude performance improvements for many existing algorithms. In this work, we focus on the application of ideas from randomized numerical linear algebra to communication-avoiding s-step GMRES methods. We propose a novel randomized s-step GMRES algorithm called RTBGS-GMRES that improves performance in the construction of the basis for the solution subspace for some matrices, while minimizing the number of global synchronizations in parallel computing environments. We compare our novel algorithm with the state-of-the-art randomized and deterministic s-step GMRES methods in terms of numerical stability, convergence, performance, and scalability. Numerical experiments on a large cluster show that with suitable parameter settings the parallel randomized GMRES methods in general outperform the parallel deterministic s-step method BCGSI2-GMRES. Our novel RTBGS-GMRES outperforms the other methods and achieves speedups of about 2× and about 4× over BCGSI2-GMRES for two different basis types.**

***Index Terms*—s-step GMRES methods, randomized sketching, parallel computing**

I. Introduction

Solving large, sparse linear systems is a central component of many scientific simulations running on HPC machines. Iterative Krylov subspace methods like GMRES are commonly used to solve such systems effectively. At scale, communication – the movement of data between different levels of the memory hierarchy and between HPC nodes – becomes a significant bottleneck of large-scale HPC applications. Communication-avoiding s-step GMRES methods [24] improve efficiency by advancing s steps (instead of one) per iteration, thus reducing communication overhead by a factor of s compared to traditional algorithms. Although this reduction lowers the cost per iteration, it introduces potential sources of numerical instability that negatively impact convergence [24]. One potential source of instability arises from the generation of s new Krylov basis vectors. A straightforward approach is to generate these vectors in the monomial basis, which can become ill-conditioned quickly, as the vectors converge toward the largest eigenvector of the input matrix. Several basis types have been proposed to address this issue [1, 31, 34]. If other basis types are ineffective, a further strategy involves varying the step size [25, 34]. Another potential source of instability arises from the block orthogonalization of the Krylov basis. Some block orthogonalization methods are efficient, using fast BLAS-3 operations and reducing global synchronizations by a factor of $\mathcal{O}(s)$ in distributed systems. However, these methods exhibit worse stability properties and impose stronger assumptions on the vectors to be orthogonalized compared to non-blocked methods [13]. Block orthogonalization should provide sufficient stability for s-step GMRES methods while reducing the number of global synchronizations. The current state-of-the-art s-step GMRES method in [34] that achieves good performance and stability in practice requires four global synchronizations per s steps for basis orthogonalization. [15] introduces s-step GMRES methods with provably stable block orthogonalization requiring only one to two global synchronizations every s steps.

Recently, randomized numerical linear algebra (RNLA) has emerged as a very exciting area in scientific computing [29]. RNLA is known for its versatility in enhancing the performance of various algorithms. Randomization techniques based on subspace embeddings [22] have been applied to GMRES methods, offering the potential to improve performance, stability, and scalability. Currently, two primary randomization techniques are applicable to GMRES. The first strategy, introduced in [6], integrates randomness in the orthogonalization of the Krylov basis. In particular, the process orthogonalizes low-dimensional sketches of high-dimensional basis vectors and uses the resulting orthogonalization coefficients to approximately orthogonalize the high-dimensional vectors, resulting in a well-conditioned basis. In [5], this technique has been extended to randomized block orthogonalization with a brief explanation of how this strategy is applicable to randomized s-step GMRES. Moreover, randomized orthogonalization can be modified to require a single global synchronization while still providing excellent stability [5, 6]. The second strategy, introduced in [30], solves a sketched least-squares problem instead of relying on the Arnoldi relation to solve the original least-squares problem of GMRES. This approach removes the orthogonality constraint on the Krylov basis while imposing only a mild constraint on its condition number, allowing greater flexibility in constructing the solution space. Consequently, the authors in [30] propose fast truncated orthogonalization for basis construction, which has a computational cost that grows linearly with the number of iterations, in contrast to the quadratic cost of full orthogonalization.

A. *Contributions of this paper*

We investigate state-of-the-art s-step GMRES methods and their combination with novel randomization strategies. Our contributions are as follows:

- We extend the work in [5] by conducting parallel experiments with their randomized s-step GMRES algorithm that requires one global synchronization per s steps.
- We introduce a novel randomized s-step GMRES method called Randomized Truncated Block Gram-Schmidt GMRES (RTBGS-GMRES) combining randomization strategies from [5] and [30] into a single algorithm. Our approach further reduces the orthogonalization cost for some matrices, at the expense of one global synchronization every s steps.
- We apply ideas from [34] originally developed for deterministic s-step GMRES to detect and correct numerical instabilities in the randomized methods. Besides the monomial basis, we also employ the scaled Newton basis [34] for further stability improvements.
- We evaluate (existing as well as our novel) randomized s-step GMRES methods for various suitable subspace embeddings and compare them to the deterministic s-step GMRES method from [34] in terms of runtime, stability, and scalability. The deterministic method uses an efficient block classical Gram-Schmidt process with inner reorthogonalization (BCGSI2) and is particularly effective for large step sizes [34], achieving speedups of up to 15$\times$ over GMRES(m) in our numerical experiments.

The rest of this paper is structured as follows. We introduce the notation used throughout this work in the following. We review related work in Section II. Section III summarizes the state-of-the-art in randomized s-step GMRES methods. Section IV presents our novel randomized s-step GMRES algorithm. Section V summarizes numerical experiments. We conclude in Section VI.

B. *Notation*

We use Matlab-like notation. Consider a matrix $\mathbf{X} \in \mathbb{R}^{n\times m}$, $1 \leq i, j \leq n$, and $1 \leq k, l \leq m$. Then $\mathbf{X}_{i:j,k:l}$ consists of the elements that are both in rows i to j and in columns k to l of $\mathbf{X}$. The columns k to l are selected using the simplified expression $\mathbf{X}_{k:l}$. If $j < i$ or $l < k$, the range is empty and the selection of the respective subset of $\mathbf{X}$ has dimension zero. The k^{th} column of $\mathbf{X}$ is represented by the column vector $\mathbf{x}_k$, while the scalar $x_{i,k}$ denotes the element in the i^{th} row and k^{th} column of $\mathbf{X}$. $\|\mathbf{X}\|$ and $\|\mathbf{X}\|_{\mathrm{F}}$ denote the 2-norm and the Frobenius norm of $\mathbf{X}$ respectively. $\kappa(\mathbf{X})$ represents the 2-norm condition number of $\mathbf{X}$. $\sigma_{\min}(\mathbf{X})$ and $\sigma_{\max}(\mathbf{X})$ indicate the smallest and largest singular value of $\mathbf{X}$, respectively.

II. Related Work

The seminal work of Hoemmen et al. [24] presents s-step GMRES methods with efficient communication-avoiding kernels. Imberti et al. [25] introduce s-step GMRES methods with variable step sizes, proposing to increase the step size according to the Fibonacci sequence for improved stability. Xu et al. [34] propose mechanisms to account for instabilities that can arise from advancing s steps rather than just one. One such mechanism adaptively reduces the step size to maintain stability in the s-step GMRES algorithm. Yamazaki et al. [38] present low-synchronization schemes for s-step GMRES methods, although without formal proofs of their stability properties. In [36], parallel s-step GMRES methods with GPU acceleration are investigated. The authors in [37] compare the performance of pipelined and s-step GMRES methods, with their combined use demonstrating the best performance in numerical experiments. An s-step GMRES algorithm with a "two-stage" orthogonalization process is proposed in [35]. This approach reduces communication cost in parallel execution compared to traditional block orthogonalization, but it lacks a rigorous stability analysis [14]. The authors in [16] provide some insight into the backward stability of s-step GMRES methods. The work in [15] presents s-step GMRES with stable block orthogonalization schemes that require at most two global synchronizations every s steps.

Nakatsukasa et al. [30] suggest using randomization to approximately solve the high-dimensional least-squares problem encountered in GMRES methods, which allows for replacing expensive orthogonalization with more cost-effective strategies. Balabanov et al. [6] introduce a randomized orthogonalization process that enhances the stability and scalability of GMRES methods. The work of [6] is extended in [5], introducing a randomized block orthogonalization process suitable for s-step GMRES algorithms. The authors in [12] apply the randomization strategy from [30] to GMRES methods with deflated restarting. Güttel et al. [21] present a randomized orthogonalization method similar to that in [6], but their approach incorporates selective orthogonalization to improve performance. The authors further introduce concepts related to our novel randomized s-step method, though applied only to non-blocked methods, without addressing parallelization. Jang et al. [26] investigate randomized orthogonalization with deterministic reorthogonalization, with application to GMRES. Flexible GMRES (FGMRES) [33] is a generalization of GMRES that allows the preconditioner to change in every iteration. A randomized FGMRES method was introduced in [27].

Although potential advantages of randomized GMRES algorithms over deterministic approaches have been analyzed theoretically, the existing literature does not contain thorough experimental comparisons between randomized and deterministic approaches. Moreover, the effectiveness of different subspace embeddings for randomized methods in practice is unclear, and the performance of randomized GMRES when restarted multiple times has not been thoroughly examined.

III. Randomized s-step GMRES

This section outlines the randomized block Gram-Schmidt (RBGS) GMRES method from [5], highlighting the matrix powers kernel (MPK) and the RBGS process as key components. We discuss suitable subspace embeddings for the RBGS process and examine the s-step RBGS-Arnoldi method, which builds on the MPK and the RBGS process, and forms the core of the RBGS-GMRES algorithm.

A. Matrix Powers Kernel (MPK)

The MPK [24] generates s new vectors for the Krylov basis along with a corresponding change-of-basis matrix. It requires a sparse matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, a start vector $\mathbf{v}_1 \in \mathbb{R}^n$, and a sequence of polynomials $\{p_j(\mathbf{A})\}_{j=0}^{s}$, defined by the recurrence relation

$$p_{i+1}(\mathbf{A}) = (\mathbf{A} - \beta_i \mathbf{I})p_i(\mathbf{A})/\gamma_i, \tag{1}$$

for $i = 1, \dots, s$ and $\gamma_i \neq 0$, where $p_0(\mathbf{A}) = \mathbf{I}$ and each $p_j(\mathbf{A})$ has degree j. The new Krylov basis vectors are given by

$$\mathbf{V}_{2:s+1} = [\mathbf{v}_2, \dots, \mathbf{v}_{s+1}] = [p_1(\mathbf{A})\mathbf{v}_1, p_2(\mathbf{A})\mathbf{v}_1, \dots, p_s(\mathbf{A})\mathbf{v}_1],$$

and the corresponding change-of-basis matrix $\mathbf{B} \in \mathbb{R}^{(s+1)\times s}$ satisfies

$$\mathbf{A}\mathbf{V}_{1:s} = \mathbf{V}_{1:s+1}\mathbf{B}.$$

The monomial basis, one of the most fundamental basis types, corresponds to setting $\gamma_i = 1$ and $\beta_i = 0$ in (1) for all i, resulting in

$$\mathbf{V}_{\text{monomial}} = [\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \mathbf{A}^2\mathbf{v}_1, \dots, \mathbf{A}^s\mathbf{v}_1].$$

The monomial basis can become ill-conditioned quickly, which limits the step size to very small s. For example, $s = 5$ is a common choice in practice [24, 35].

A straightforward implementation of the MPK involves s successive matrix-vector operations. Hoemmen et al. [24] introduce a communication-avoiding MPK (CA-MPK) that optimizes these operations by modifying the matrix-vector distribution and communication patterns. Their approach reduces the number of messages sent by a factor of $\mathcal{O}(s)$ at the cost of some redundant computation. However, CA-MPK faces two main challenges: 1) limited use of preconditioners, which are crucial for iterative solvers and often require communication that may disrupt the method's efficiency, and 2) difficulties in achieving significant performance improvements, as the algorithm's sparse matrix-vector operations are often memory-bound rather than compute-bound.

B. Randomized subspace embeddings

Randomized sketching maps a high-dimensional problem to a low-dimensional subspace, maintaining its inherent structure. Solving the problem embedded in the low-dimensional subspace yields a fast approximate solution to the original problem with a certain probability. Subspace embeddings approximate high-dimensional basis vectors and inner products of high-dimensional basis vectors. The following definition specifies the inner product approximation [6].

Definition III.1. *Let the columns of* $\mathbf{V}$ *be a basis for an* m*-dimensional subspace* $V \subset \mathbb{R}^n$ *and let* $\epsilon \in (0, 1)$*. The sketch matrix* $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$*, with sketch dimension* $d \ll n$*, is said to be an* ϵ*-subspace embedding for* V*, if*

$$\forall \mathbf{x}, \mathbf{y} \in \mathbf{V} : |\mathbf{x}^{\mathrm{T}}\mathbf{y} - (\mathbf{\Theta}\mathbf{x})^{\mathrm{T}}\mathbf{\Theta}\mathbf{y}| \leq \epsilon\|\mathbf{x}\|\|\mathbf{y}\|. \tag{2}$$

An *oblivious* subspace embedding (OSE) sketches unknown m-dimensional subspaces in $\mathbb{R}^n$.

Definition III.2. $\mathbf{\Theta}$ *is said to be an* (ϵ, δ, m) *OSE, if*

$$\mathbb{P}(\mathbf{\Theta} \textit{ is an } \epsilon\textit{-embedding for } V) \geq 1 - \delta. \tag{3}$$

That is, without having prior knowledge of the subspace V, an OSE *simultaneously* approximates the norms of all vectors in V with some probability $1 - \delta$ [18].

Corollary III.3. *If* $\mathbf{\Theta}$ *is an* ϵ*-embedding for* $\mathbf{V}$*, the singular values of* $\mathbf{V}$ *are bounded by*

$$\begin{aligned} \sigma_{\min}(\mathbf{V}) &\geq (1+\epsilon)^{-\xi}\sigma_{\min}(\mathbf{\Theta}\mathbf{V}), \\ \sigma_{\max}(\mathbf{V}) &\leq (1-\epsilon)^{-\xi}\sigma_{\max}(\mathbf{\Theta}\mathbf{V}), \end{aligned} \tag{4}$$

with $\xi = 1/2$ [6]. From (4) follows the condition number bound

$$\left(\frac{1-\epsilon}{1+\epsilon}\right)^{\xi}\kappa_2(\mathbf{\Theta}\mathbf{V}) \leq \kappa_2(\mathbf{V}) \leq \left(\frac{1+\epsilon}{1-\epsilon}\right)^{\xi}\kappa_2(\mathbf{\Theta}\mathbf{V}). \tag{5}$$

OSEs are constructed using probability distributions over $\mathbf{\Theta}$, such as Gaussian or Rademacher distributions [7]. More advanced methods construct embeddings with structure, allowing for faster application times, such as the Subsampled Randomized Hadamard Transform (SRHT) or CountSketch matrix (see [11] and [17] respectively). When randomized GMRES methods are executed on large-scale distributed systems, highly parallelizable sketch matrices should be employed. Furthermore, a large sketch dimension d increases the overall cost of randomized GMRES methods, including storage, arithmetic operations, and communication. These costs can be mitigated by selecting sketches that require relatively small d. Taking everything into account, sketch matrices should be fast to apply, highly parallelizable, and should require only a small sketch dimension d to fulfill (3) with high probability.

We discuss properties of suitable subspace embeddings that require at most one global reduction operation in distributed systems.

1) Gaussian embeddings: A rescaled Gaussian matrix has independent Gaussian entries with zero mean and variance d^{-1} [7]. It is applied to $\mathbf{V}$ in $\mathcal{O}(nmd)$ time. The embedding dimension must satisfy $d \geq \mathcal{O}\left(\epsilon^{-2}(m + \log\frac{1}{\delta})\right)$ [7].

2) Block SRHT (BSRHT): SRHTs are applied to $\mathbf{V}$ in $\mathcal{O}(nm\log d)$ time but lack effective parallelization due to high communication overhead. The BSRHT from [4] combines high parallelizability with the fast local application time of SRHT matrices, although with a slightly worse bound on the sketch dimension $d \geq \mathcal{O}\left(\epsilon^{-2}(m + \log\frac{n}{\delta})\log\frac{m}{\delta}\right)$ [4].

3) CountSketch: A CountSketch matrix is a sparse embedding, with one i.i.d. Rademacher entry per column, placed uniformly at random, and zeros otherwise [17]. CountSketch matrices are applied to $\mathbf{V}$ in $\mathcal{O}(mn)$ time. Their embedding dimension bound is $d \geq \mathcal{O}\left(\epsilon^{-2} m^2 \delta^{-1}\right)$ [17, 28], indicating higher sensitivity to lower failure probabilities δ and larger m, as d grows with δ^{-1} and m^2.

In short, Gaussian embeddings offer the strongest theoretical guarantees but the weakest performance, CountSketches have the best performance but the weakest guarantees, and BSRHTs are in between.

C. Randomized Block Gram-Schmidt (RBGS) process

The RBGS process from [5] is a blocked version of the Randomized Gram-Schmidt process from [6], which constructs an approximately orthonormal and well-conditioned basis from a set of high-dimensional basis vectors $\mathbf{V} \in \mathbb{R}^{n \times m}$. This is achieved through orthogonalization of the compressed matrix $\mathbf{\Theta V}$ rather than $\mathbf{V}$ itself. The resulting R factor is applied to $\mathbf{V}$, yielding a matrix $\mathbf{Q} \in \mathbb{R}^{n \times m}$ that we say is $\mathbf{\Theta}$-orthonormal, meaning that its sketch $\mathbf{S} = \mathbf{\Theta Q}$ is orthonormal. Algorithm 1 outlines the RBGS algorithm.

Algorithm 1 RBGS process

```
Input: $\mathbf{V} \in \mathbb{R}^{n \times m}$, step size $s$, $\mathbf{\Theta} \in \mathbb{R}^{d \times n}$, $d \ll n$, $m = sp$
Output: $\mathbf{\Theta}$-orthonormal $\mathbf{Q} \in \mathbb{R}^{n \times m}$, upper triangular $\mathbf{R} \in \mathbb{R}^{m \times m}$
1: for $j = 1 : p$ do
2:     $i = s(j-1)$
3:     $b = i+1 : i+s$
4:     $\mathbf{P}_b = \mathbf{\Theta V}_b$
5:     $\mathbf{R}_{1:i,b} = \operatorname{argmin}_{\mathbf{Y}} \|\mathbf{S}_{1:i}\mathbf{Y} - \mathbf{P}_b\|_{\mathrm{F}}$
6:     $\mathbf{Q}'_b = \mathbf{V}_b - \mathbf{Q}_{1:i}\mathbf{R}_{1:i,b}$
7:     Apply randomized QR to $\mathbf{Q}'_b$ to obtain $\mathbf{Q}_b\mathbf{R}_{b,b}$
8:     $\mathbf{S}_b = \mathbf{\Theta Q}_b$
9: end for
```

1) Stability guarantees: The authors in [5] show that, under the condition $\epsilon \leq 1/2$ and additional assumptions, the singular values of $\mathbf{Q}$ satisfy the bounds

$$\begin{aligned} &\sigma_{\min}(\mathbf{Q}) \geq (1+\epsilon)^{-1/2}(1 - \Delta_m - 0.1\tilde{u}), \\ &\sigma_{\max}(\mathbf{Q}) \leq (1-\epsilon)^{-1/2}(1 + \Delta_m + 0.1\tilde{u}), \text{ and} \\ &\quad \Delta_m \leq 20\tilde{u}m^2\kappa(\mathbf{V}), \end{aligned} \tag{6}$$

where $\Delta_m = \|\mathbf{I} - \mathbf{S}^{\mathrm{T}}\mathbf{S}\|_{\mathrm{F}}$, $\tilde{u} = F(m,n)u$, $F(m,n)$ is a low-degree polynomial and u denotes the unit roundoff. We highlight key aspects of two randomized QR strategies discussed in [5] for orthonormalizing $\mathbf{Q}'_b$ with respect to $\mathbf{\Theta}$ in line 7 of Algorithm 1.

a) Randomized QR with explicit sketch: The explicit strategy performs a randomized Cholesky QR factorization [3], which involves an unconditionally stable QR factorization of the sketch $\mathbf{\Theta Q}'_b$ with subsequent use of the R factor to compute $\mathbf{Q}_b = \mathbf{Q}'_b\mathbf{R}_{b,b}^{-1}$ by forward substitution. Randomized Cholesky QR comes with great stability guarantees, requiring only the mild condition $\kappa(\mathbf{Q}'_b) < \tilde{u}^{-1}$ [3].

b) Randomized QR with implicit sketch: The implicit strategy avoids the sketching step, instead computing $\mathbf{S}'_b = \mathbf{\Theta Q}'_b$ as $\mathbf{S}'_b = \mathbf{P}_b - \mathbf{S}_{1:i}\mathbf{R}_{1:i,b}$ from the previously sketched vectors $\mathbf{P}_b$ and $\mathbf{S}_{1:i}$. The authors in [5] note that using the implicit approach in the RBGS process is equivalent to using the explicit strategy with two perturbations. These perturbations increase $\kappa(\mathbf{\Theta Q}_b)$ by at most a factor of $\mathcal{O}(\tilde{u}\kappa(\mathbf{V})F(m))$, where $F(m)$ is a small polynomial. Using (5), it follows that the implicit strategy does not affect the stability guarantees of RBGS up to small constants [5].

The stability of RBGS further depends on the quality of the solution to the least-squares problem computed in line 5 of Algorithm 1. If the sketch dimension d is sufficiently small, using an unconditionally stable QR factorization method, such as Householder QR, is recommended. If d is too large, direct solvers become prohibitively expensive. In such cases, a few rounds of Richardson iterations can be applied to the normal equations $(\mathbf{S}_{1:i}^{\mathrm{T}}\mathbf{S}_{1:i})\mathbf{R}_{1:i,b} = \mathbf{S}_{1:i}^{\mathrm{T}}\mathbf{P}_b$, as long as $\mathbf{S}_{1:i}$ is well-conditioned (see [5] for details).

2) Performance analysis: The RBGS algorithm requires roughly half the overall computational cost of block classical Gram-Schmidt methods used in deterministic s-step methods, with half the number of passes over high-dimensional objects [5]. Using the explicit randomized QR strategy in Algorithm 1 requires three global synchronizations every s steps, while the variant with the implicit strategy requires only two. An additional global synchronization can be avoided by postponing the sketching step in line 8 of Algorithm 1 to the next iteration, where it is combined with the sketching step in line 4 of Algorithm 1. When using this modification in combination with the implicit randomized QR strategy, Algorithm 1 requires a single global synchronization in total.

D. s-step RBGS-Arnoldi

The Arnoldi method approximates the eigenvalues and eigenvectors of large, sparse matrices. It generates an orthonormal basis $\mathbf{Q} \in \mathbb{R}^{n \times m}$ for the Krylov subspace associated with a given matrix $\mathbf{A}$ and a start vector $\mathbf{v}_1$ and produces an upper Hessenberg matrix $\mathbf{H}$ that satisfies the Arnoldi relation

$$\mathbf{AQ}_{1:m-1} = \mathbf{QH}. \tag{7}$$

The s-step RBGS-Arnoldi method generates a $\mathbf{\Theta}$-orthonormal Krylov basis in blocks. It employs the MPK to produce a small block of the Krylov basis, followed by RBGS orthogonalization. Algorithm 2 illustrates the s-step RBGS-Arnoldi method using the implicit randomized QR strategy (cf. line 7 of Algorithm 1). The algorithm preserves the stability guarantees and computational advantages of the RBGS process [5].

1) Constructing the upper Hessenberg matrix: The computation of the upper Hessenberg matrix in deterministic s-step Arnoldi methods is rather intricate (see, e.g., [24]). Randomization facilitates this process. The columns of $\mathbf{H}$ in line 17 of Algorithm 2 are obtained by solving least-squares problems that require the sketch $\mathbf{\Theta AQ}_{\bar{b}}$, where $\bar{b}$ is defined as in line 16 of Algorithm 2. We discuss two strategies for computing this sketch, either explicitly or implicitly.

Algorithm 2 s-step RBGS-Arnoldi

```
Input: $\mathbf{A} \in \mathbb{R}^{n\times n}$, start vector $\mathbf{v}_1 \in \mathbb{R}^n$, step size $s$, $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, $d \ll n$, $m = sp+1$
Output: $\mathbf{\Theta}$-orthonormal $\mathbf{Q} \in \mathbb{R}^{n\times m}$, upper Hessenberg $\mathbf{H} \in \mathbb{R}^{m\times m-1}$
 1: $\mathbf{p}_1 = \mathbf{\Theta}\mathbf{v}_1$
 2: $r_{1,1} = \|\mathbf{p}_1\|$
 3: $\mathbf{q}_1 = \mathbf{v}_1/r_{1,1}$
 4: $\mathbf{s}_1 = \mathbf{\Theta}\mathbf{q}_1$
 5: for $j = 1 : p$ do
 6:     $i = s(j-1)+1$
 7:     $b = i+1 : i+s$
 8:     $[\mathbf{V}_b, \mathbf{B}] = \mathtt{mpk}(\mathbf{A}, \mathbf{q}_i, s)$
 9:     $\mathbf{P}_b = \mathbf{\Theta}\mathbf{V}_b$
10:     $\mathbf{R}_{1:i,b} = \text{argmin}_{\mathbf{Y}}\|\mathbf{S}_{1:i}\mathbf{Y} - \mathbf{P}_b\|_{\text{F}}$
11:     $\mathbf{Q}'_b = \mathbf{V}_b - \mathbf{Q}_{1:i}\mathbf{R}_{1:i,b}$
12:     $\mathbf{S}'_b = \mathbf{P}_b - \mathbf{S}_{1:i}\mathbf{R}_{1:i,b}$          ▷ Implicit sketch $\mathbf{\Theta}\mathbf{Q}'_b$
13:     $[\sim, \mathbf{R}_{b,b}] = \mathtt{qr}(\mathbf{S}'_b)$
14:     $\mathbf{Q}_b = \mathbf{Q}'_b\mathbf{R}_{b,b}^{-1}$ via triangular solve
15:     $\mathbf{S}_b = \mathbf{\Theta}\mathbf{Q}_b$
16:     $\bar{b} = i : i+s-1$
17:     $\mathbf{H}_{1:i+s,\bar{b}} = \text{argmin}_{\mathbf{Y}}\|\mathbf{S}_{1:i+s}\mathbf{Y} - \mathbf{\Theta}\mathbf{A}\mathbf{Q}_{\bar{b}}\|_{\text{F}}$
18: end for
```

a) Explicit sketch of $\mathbf{AQ}_{\bar{b}}$*:* The explicit strategy is an expensive, but reliable method for accurately sketching $\mathbf{AQ}_{\bar{b}}$. For efficiency, the computation of $\mathbf{AQ}_{\bar{b}}$ can be deferred to the next iteration, where it is integrated into the MPK in line 8 of Algorithm 2. The result is subsequently sketched along with $\mathbf{V}_b$ in line 9 of Algorithm 2 [5]. Despite these optimizations, we found that the explicit approach is rather impractical, as it requires additional operations on high-dimensional objects, potentially doubling the cost of both the MPK and the sketching step involving $\mathbf{V}_b$.

b) Implicit sketch of $\mathbf{AQ}_{\bar{b}}$*:* The implicit approach computes $\mathbf{AQ}_{\bar{b}}$ recursively without additional communication or operations on high-dimensional objects. Using the change-of-basis relation $\mathbf{AV}_{\bar{b}} = \mathbf{V}_{i:i+s}\mathbf{B}$ we first compute

$$[\mathbf{\Theta A q}_i, \mathbf{\Theta A V}_{\hat{b}}] = [\mathbf{s}_i, \mathbf{P}_b]\mathbf{B},$$

where $\hat{b} = i+1 : i+s-1$, $\mathbf{s}_i = \mathbf{\Theta q}_i$, $\mathbf{P}_b = \mathbf{\Theta V}_b$, and $\mathbf{\Theta A q}_i$ is the first column of $\mathbf{\Theta AQ}_{\bar{b}}$. Using $\mathbf{\Theta AV}_{\hat{b}}$ in the relation

$$\mathbf{\Theta AQ}_{\hat{b}} = \left(\mathbf{\Theta AV}_{\hat{b}} - \mathbf{\Theta AQ}_{1:i}\mathbf{R}_{1:i,\hat{b}}\right)\mathbf{R}_{\hat{b},\hat{b}}^{-1},$$

the rest of the $s-1$ columns $\mathbf{\Theta AQ}_{\hat{b}}$ in $\mathbf{\Theta AQ}_{\bar{b}}$ are obtained. The cost of updating the Hessenberg matrix in line 17 of Algorithm 2 using the implicit approach is $\mathcal{O}\left(di^2\right)$, or $\mathcal{O}\left(dis\right)$ reusing the QR factorizations of the respective columns of $\mathbf{S}$ from previous iterations. For deterministic s-step GMRES methods, the Hessenberg update takes $\mathcal{O}\left(i^3\right)$ time, or $\mathcal{O}\left(is^2\right)$ time using the optimized procedure described in [24]. In either case, the computational cost for the randomized approach is higher than that of the deterministic methods, since $d > i$.

E. RBGS-GMRES

RBGS-GMRES is directly derived from the s-step RBGS-Arnoldi method. When solving the linear system $\mathbf{Ax} = \mathbf{b}$ with RBGS-GMRES, assume for simplicity that the initial guess is zero. Using Algorithm 2 with start vector $\mathbf{b}$, we obtain $r_{1,1} = \|\mathbf{\Theta b}\|$, a $\mathbf{\Theta}$-orthonormal basis $\mathbf{Q}$, and an upper Hessenberg matrix $\mathbf{H}$. The extra steps in RBGS-GMRES involve solving the small least-squares problem $\mathbf{y} = \text{argmin}_{\mathbf{z}}\|r_{1,1}\mathbf{e}_1 - \mathbf{Hz}\|$ and computing an approximate solution $\mathbf{Q}_{1:m-1}\mathbf{y} \approx \mathbf{x}$. RBGS-GMRES minimizes the *sketched* residual norm. This becomes evident by expressing the least-squares problem as follows:

$$\text{argmin}_{\mathbf{z}}\|r_{1,1}\mathbf{e}_1 - \mathbf{Hz}\| = \text{argmin}_{\mathbf{z}}\|\mathbf{\Theta Q}(r_{1,1}\mathbf{e}_1 - \mathbf{Hz})\| = \text{argmin}_{\mathbf{z}}\|\mathbf{\Theta}(\mathbf{b} - \mathbf{QHz})\| = \text{argmin}_{\mathbf{z}}\|\mathbf{\Theta}(\mathbf{b} - \mathbf{AQ}_{1:m-1}\mathbf{z})\|.$$

If $\mathbf{\Theta}$ is an ϵ-embedding for $\mathbf{Q}$, and if (6) holds, then the sketched residual norm of RBGS-GMRES differs from the residual norm by a factor of $\sqrt{(1+\epsilon)/(1-\epsilon)}$ [5]. More specifically, let $\mathbf{x}_\star$ be the approximate solution of standard GMRES and let $\tilde{\mathbf{x}}$ be the approximate solution of RBGS-GMRES. Using (2), the residual norm with respect to $\tilde{\mathbf{x}}$ is bounded by

$$\|\mathbf{b} - \mathbf{Ax}_\star\| \le \|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\| \le \left(\frac{1+\epsilon}{1-\epsilon}\right)^{\xi}\|\mathbf{b} - \mathbf{Ax}_\star\|, \quad (8)$$

with $\xi = 1/2$.

F. Improving numerical stability

The MPK and block orthogonalization are potential sources of numerical instability in s-step GMRES methods [34]. We summarize techniques discussed in [34] that address instabilities in their deterministic s-step GMRES algorithm (referred to as BCGSI2-GMRES here), which are also applicable to the randomized methods.

1) A different MPK basis: To improve numerical stability in the MPK, bases other than the monomial basis can be employed, with the Newton basis being a popular choice. The Newton basis uses Ritz values, which approximate the eigenvalues of $\mathbf{A}$, as shifts to differ from the monomial basis as much as possible [24]. The Ritz values are computed by performing s iterations of the Arnoldi method or standard GMRES. If many Ritz values are required, monomial-based s-step RBGS-Arnoldi or s-step GMRES may potentially speed up this process. The Newton basis corresponds to (1) with $\beta_i = \theta_i$ and $\gamma_i = 1$ for all i, resulting in

$$\mathbf{V}_{\text{Newton}} = [\mathbf{v}_1, (\mathbf{A} - \theta_1\mathbf{I})\mathbf{v}_1, \ldots, \prod_{i=1}^{s}(\mathbf{A} - \theta_i\mathbf{I})\mathbf{v}_1],$$

where $\theta_1, \theta_2, \ldots, \theta_s$ are the Ritz values of $\mathbf{A}$. To improve the conditioning of the Newton basis, the Ritz values are arranged according to the Leja ordering [1]. The modified Leja ordering [1] avoids complex arithmetic when the Ritz values of a real matrix are complex [24].

The scaled Newton basis [34] further improves numerical stability and is obtained with $\beta_i = \theta_i$ and $\gamma_i = |\bar{\theta} - \theta_i|$ for all

i in (1), where $\bar{\theta}$ is the mean of the computed Ritz values. As more Ritz values are computed, $\bar{\theta}$ will be closer to the mean eigenvalue spectrum of $\mathbf{A}$. This approach aims to keep the basis vector length around $\sim \mathcal{O}(1)$, which potentially results in significantly larger admissible step sizes for which the s-step algorithm is stable [34].

2) Adaptive step size: Smaller step sizes improve the numerical stability of block orthogonalization in s-step methods. The incremental condition estimator (ICE) [9, 10] can be used to detect numerical instabilities during the QR factorization of an $n \times s$ matrix $\mathbf{V} = \mathbf{QR}$. Once the k^{th} column of $\mathbf{R}$ has been computed, the ICE estimates $\kappa(\mathbf{R}_{1:k,1:k}) = \kappa(\mathbf{V}_{1:k})$ in $\mathcal{O}(k)$ time. This estimate is usually at most ten times the actual value [34]. For a detailed algorithm description of the ICE, see [10]. If $\kappa(\mathbf{R}_{1:k,1:k})$ (or the estimate) exceeds a user-given threshold Ω, the factorization process stops, resulting in a *partial* QR factorization $\mathbf{V}_{1:k-1} = \mathbf{Q}_{1:k-1}\ \mathbf{R}_{1:k-1,1:k-1}$. The partial QR factorization discards the remaining vectors in $\mathbf{V}$ and, as part of the block orthogonalization process, reduces the step size in adaptive s-step methods to $k-1$ (see [34] for details). The s-step RBGS-Arnoldi and RBGS-GMRES algorithms can be made adaptive by incorporating the ICE into the QR factorization of $\mathbf{S}'_b$ in line 13 of Algorithm 2. In practice, $\Omega = \mathcal{O}\left(10^{-1}u^{-1/2}\right)$ works well for BCGSI2-GMRES [34]. In general, we recommend this value for the randomized methods as well, and $\Omega = \mathcal{O}\left(10^{-2}u^{-1/2}\right)$ for challenging cases.

3) Step size estimation: The authors in [34] introduce a step size estimator for the scaled Newton basis to estimate the *optimal* step size, which is not easily determined *a priori*. The optimal step size s_{opt} is defined as the largest admissible step size that prevents the computation of discarded basis vectors, given the condition number bound Ω of the ICE. Specifically, $s_{\text{opt}} = \text{argmax}_j\{\kappa(\mathbf{V}_{1:j}) < \Omega\}$, where $\mathbf{V} \in \mathbb{R}^{n\times s}$ is a scaled Newton basis. The estimator efficiently monitors vector growth in $\mathbf{V}$ based on the Ritz values, without requiring additional communication. It detects an ill-conditioned basis $\mathbf{V}$ due to poor vector scaling, but it does not account for nearly linearly dependent vectors. Consequently, the estimate may be significantly larger than s_{opt} in practice.

IV. RTBGS-GMRES

We present a novel GMRES algorithm that integrates two randomization strategies from [5] and [30]. Specifically, the method employs a randomized truncated block Gram-Schmidt (RTBGS) process for fast Krylov basis generation and solves implicitly sketched least-squares problems, all at the cost of a single global synchronization every s steps.

A. Randomized Truncated Block Gram-Schmidt (RTBGS)

The RTBGS process is similar to RBGS (Algorithm 1), but performs partial orthogonalization by projecting new basis vectors against the t most recently orthonormalized vectors. In particular, RTBGS replaces lines 5 and 6 of the RBGS process in Algorithm 1 with the lines

$$\begin{aligned} 5: &\quad \mathbf{R}_{b_t,b} = \text{argmin}_{\mathbf{Y}}\|\mathbf{S}_{b_t}\mathbf{Y} - \mathbf{P}_b\|_{\text{F}} \\ 6: &\quad \mathbf{Q}'_b = \mathbf{V}_b - \mathbf{Q}_{b_t}\mathbf{R}_{b_t,b}, \end{aligned}$$

where $b_t = \max(1, i-t+1) : i$. The orthogonalization cost is reduced from quadratic $\mathcal{O}\left(ni^2\right)$ time to linear $\mathcal{O}(nit)$ time at iteration i. The resulting basis $\mathbf{Q}$ may be poorly conditioned due to truncated orthogonalization, but its conditioning can be notably better than that of the input matrix $\mathbf{V}$ [30].

B. RTBGS-GMRES algorithm

Standard GMRES uses the Arnoldi relation in (7) to efficiently solve the least-squares problem

$$\text{argmin}_{\mathbf{z}} \|\mathbf{b} - \mathbf{A}\mathbf{Q}_{1:m-1}\mathbf{z}\|, \tag{9}$$

The strategy proposed in [30] solves the sketched least-squares problem

$$\text{argmin}_{\mathbf{z}} \|\mathbf{\Theta}(\mathbf{b} - \mathbf{A}\mathbf{Q}_{1:m-1}\mathbf{z})\| \tag{10}$$

directly, without relying on the Arnoldi relation. As a result, there are no orthogonality constraints on the Krylov basis $\mathbf{Q}$, and (10) imposes the mild condition $\kappa(\mathbf{\Theta}\mathbf{A}\mathbf{Q}_{1:m-1}) \lesssim u^{-1}$. With this approach, the RTBGS-GMRES algorithm efficiently constructs a Krylov basis using the RTBGS process and reliably solves the corresponding sketched least-squares problem, even when the problem is poorly conditioned. Furthermore, RTBGS-GMRES allows for implicit sketching of the least-squares problem (see the discussion in Section III-D1). With additional optimizations similar to those discussed in Section III-C2, the algorithm requires a single global synchronization every s steps. RTBGS-GMRES is presented in Algorithm 3. Components motivated by [30] (i.e., solving sketched least-squares problems and truncated orthogonalization) are highlighted in red, and ideas related to low-synchronization from [5] (i.e., postponed and implicit sketching) are marked in yellow.

1) Stability considerations: The following definition applies to high-dimensional least-squares problems embedded in low-dimensional subspaces [30].

Definition IV.1. *Let the columns of $\mathbf{V}$ be a basis for an m-dimensional subspace $V \subset \mathbb{R}^n$ and let $\epsilon \in (0,1)$. The sketch matrix $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, with sketch dimension $d \ll n$, is said to be an ϵ-subspace embedding for V, if*

$$\forall \mathbf{x} \in \mathbb{R}^m : (1-\epsilon)\|\mathbf{V}\mathbf{x}\| \le \|\mathbf{\Theta}\mathbf{V}\mathbf{x}\| \le (1+\epsilon)\|\mathbf{V}\mathbf{x}\|. \tag{11}$$

The OSE property in (3) also applies to (11). Furthermore, we have that (4) and (5) hold with $\xi = 1$. According to (5), if $\mathbf{\Theta}$ is a subspace embedding for $\mathbf{A}\mathbf{Q}_{1:m-1}$, then $\kappa(\mathbf{A}\mathbf{Q}_{1:m-1}) \lesssim u^{-1}$, as long as $\kappa(\mathbf{\Theta}\mathbf{A}\mathbf{Q}_{1:m-1}) \lesssim u^{-1}$. $\kappa(\mathbf{\Theta}\mathbf{A}\mathbf{Q}_{1:m-1})$ can be inexpensively monitored using the ICE with condition number bound $\Omega_{\mathbf{C}} = \mathcal{O}\left(u^{-1}\right)$ during the QR factorization in line 24 of Algorithm 3, which should be unconditionally stable (e.g., Householder QR). Once $\kappa(\mathbf{\Theta}\mathbf{A}\mathbf{Q}_{1:i}) > \Omega_{\mathbf{C}}$, we either restart the algorithm or "whiten" the basis (refer to [30] for details).

Algorithm 3 RTBGS-GMRES

```
Input: $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{x}^{(0)} \in \mathbb{R}^n$, step
    size $s$, restart length $m$, embedding $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, $d \ll n$,
    truncation parameter $t$
Output: approximate solution $\tilde{\mathbf{x}} \in \mathbb{R}^n$ to $\mathbf{Ax} = \mathbf{b}$
 1: for $k = 0, 1, \ldots,$ until convergence do
 2:     $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{Ax}^{(k)}$
 3:     $\mathbf{p}_1 = \mathbf{\Theta}\mathbf{r}^{(k)}$
 4:     $r_{1,1} = \|\mathbf{p}_1\|$
 5:     $\mathbf{q}_1 = \mathbf{r}^{(k)}/r_{1,1}$
 6:     $b_p = 1$                                          ▷ Previous block indices
 7:     $i = 1$
 8:     while $i \leq m$ do
 9:         if $i + s > m + 1$ then $s = m + 1 - i$
10:         $b_c = i + 1 : i + s$                          ▷ Current block indices
11:         $[\mathbf{V}_{b_c}, \mathbf{B}] = \texttt{mpk}(\mathbf{A}, \mathbf{q}_i, s)$
12:         $[\mathbf{S}_{b_p}, \mathbf{P}_{b_c}] = \mathbf{\Theta}[\mathbf{Q}_{b_p}, \mathbf{V}_{b_c}]$                   ▷ Global sync.
13:         $b_t = \max(1,\, i - t + 1) : i$
14:         $\mathbf{R}_{b_t, b_c} = \mathrm{argmin}_{\mathbf{Y}} \|\mathbf{S}_{b_t}\mathbf{Y} - \mathbf{P}_{b_c}\|_{\mathrm{F}}$
15:         $\mathbf{Q}'_{b_c} = \mathbf{V}_{b_c} - \mathbf{Q}_{b_t}\mathbf{R}_{b_t, b_c}$
16:         $\mathbf{S}'_{b_c} = \mathbf{P}_{b_c} - \mathbf{S}_{b_t}\mathbf{R}_{b_t, b_c}$     ▷ Implicit sketch $\mathbf{\Theta Q}'_{b_c}$
17:         $[\sim, \mathbf{R}_{b_c, b_c}] = \texttt{qr}(\mathbf{S}'_{b_c})$                   ▷ Thin QR
18:         $\mathbf{Q}_{b_c} = \mathbf{Q}'_{b_c}\mathbf{R}^{-1}_{b_c, b_c}$ via triangular solve
19:         $\hat{b}_c = i + 1 : i + s - 1$                  ⎫
20:         $[\mathbf{c}_i, \hat{\mathbf{C}}_{\hat{b}_c}] = [\mathbf{s}_i, \mathbf{P}_{b_c}]\mathbf{B}_{1:s+1,1:s}$        ⎬ $\mathbf{\Theta A Q}_{\bar{b}_c}$,
21:         $\mathbf{C}'_{\hat{b}_c} = \hat{\mathbf{C}}_{\hat{b}_c} - \mathbf{C}_{b_t}\mathbf{R}_{b_t, \hat{b}_c}$           ⎬ with
22:         $\mathbf{C}_{\hat{b}_c} = \mathbf{C}'_{\hat{b}_c}\mathbf{R}^{-1}_{\hat{b}_c, \hat{b}_c}$ via triangular solve  ⎭ $\bar{b}_c = i : i + s - 1$
23:         $i = i + s$
24:         $[\hat{\mathbf{Q}}_{1:i-1}, \hat{\mathbf{R}}_{1:i-1,1:i-1}] = \texttt{qr}(\mathbf{C}_{1:i-1})$    ▷ Thin QR
25:         $\hat{r} = \|(\mathbf{I} - \hat{\mathbf{Q}}_{1:i-1}\hat{\mathbf{Q}}^{\mathrm{T}}_{1:i-1})\mathbf{p}_1\|$ ▷ Residual estimate
26:         $b_p = b_c$
27:         Use $\hat{r}$ to check convergence
28:     end while
29:     $\mathbf{y}^{(k)} = \hat{\mathbf{R}}^{-1}_{1:i-1,1:i-1}(\hat{\mathbf{Q}}^{\mathrm{T}}_{1:i-1}\mathbf{p}_1)$ via triangular solve
30:     $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{Q}_{1:i-1}\mathbf{y}^{(k)}$           ▷ Current solution $\tilde{\mathbf{x}}$
31: end for
```

2) Performance analysis: An iteration with the single-step method *sGMRES* from [30] requires at least two global synchronizations for orthogonalization and one for sketching the least-squares problem in (10). A potential s-step version of the method requires at least as many global synchronizations every s steps. In contrast, RTBGS-GMRES requires only one global synchronization. Moreover, truncated orthogonalization effectively reduces the projection cost, making RTBGS-GMRES particularly suitable for small step sizes, as the projection step would otherwise be a dominant cost in this case.

3) Convergence guarantees: Convergence guarantees for RTBGS-GMRES are derived similarly to those in [30] for sGMRES. Let $\tilde{\mathbf{x}}$ be the approximate solution of RTBGS-GMRES. Using (11), the residual norm with respect to $\tilde{\mathbf{x}}$ is bounded as in (8) with $\xi = 1$. Analogously to [30], the residual estimate in line 25 of Algorithm 3 is checked for convergence.

V. Numerical Experiments

This section presents parallel numerical experiments for BCGSI2-GMRES, RBGS-GMRES, and RTBGS-GMRES using both the monomial basis and the scaled Newton basis.

A. Experimental setup

The algorithms were implemented using the Trilinos C++ library [23] (version 14.0), a collection of open-source software packages for solving complex scientific and engineering problems through advanced parallel algorithms. Our code is based on the Belos package [8], a framework for iterative solvers of large, sparse linear systems. The experiments were conducted on the Vienna Scientific Cluster (VSC-5), a system with 564 nodes, each equipped with two 64-core CPUs. We utilized up to 128 processes per node, assigning one process per core as the mode of parallelization.

1) Problem and parameter setting: Our test cases include application matrices from Table I and 3D Laplace problems. The right-hand side of a linear system was generated following the procedure described in Appendix B.2 of [24]. The initial start vector was the zero vector.

Convergence was monitored using the implicit residual norm relative to $\|\mathbf{r}^{(0)}\|$ for BCGSI2-GMRES, or an estimate relative to $\|\mathbf{\Theta r}^{(0)}\|$ for the randomized methods. After convergence was detected, the solvers continued until the computed solution $\tilde{\mathbf{x}}$ satisfied $\|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\| < \|\mathbf{b}\|\texttt{cTol}$, where `cTol` is the convergence tolerance. In Table I, `cTol` is relatively large for the matrices `af_3` and `ML_r`. Among all test cases, `af_3` has the longest runtime by far (see Tables II and III). Increasing `cTol` for this matrix allowed the collection of experimental data in reasonable time. For `ML_r`, all methods, including GMRES(m), exhibit residual stagnation around $2.08 \cdot 10^{-7}$ when using preconditioners from the Ifpack2 Trilinos package [32]. Identifying a better preconditioner for this matrix is beyond the scope of this work.

Restart lengths $m \leq 60$ for GMRES(m) are commonly found in the literature (see, e.g., [2, 19]). In our numerical experiments with s-step methods, we observed that significantly larger restart lengths (e.g., $200 \leq m \leq 400$) led to improved runtime performance due to faster convergence.

In all s-step methods, the ICE (see Section III-F2) estimated the condition number of the basis which was generated by the MPK through s conventional sparse matrix-vector multiplications. If the estimate exceeded the bound Ω, the step size was reduced accordingly. Unless stated otherwise, the default parameter was $\Omega = 10^{-1}u^{-1/2}$, with u in double precision.

Like RTBGS-GMRES, the RBGS-GMRES implementation requires one global synchronization every s steps. For sketch dimensions $d \leq 1500$, block least-squares problems involving the matrix $\mathbf{S}$ (cf. lines 10 and 17 of Algorithm 2) were solved using Householder QR, with reflectors from previous iterations recycled for efficiency. For $d = 5000$, the corresponding normal equations were solved using five rounds of Richardson iterations. The initial truncation parameter of RTBGS-GMRES was set to $t = 8$ in all experiments. In line 24 of Algorithm 3, the QR factorizations from previous iterations were reused, and the condition number of $\hat{\mathbf{R}}_{1:i-1,1:i-1}$ was estimated using the ICE. If this estimate exceeded $\Omega_{\mathbf{C}} = 10^{14}$, the method was restarted with the current solution as initial guess and a new truncation parameter $t_{new} = \max(2t, m)$.

2) Ifpack2 preconditioners: We briefly describe the parallel preconditioners from the Ifpack2 Trilinos package listed in Table I. All preconditioners were used as right preconditioners.

a) COLEQUI: Column equilibration, where each column of $\mathbf{A}$ is divided by its column one-norm.

b) ILU(k): Incomplete LU Factorization performed on the diagonal (local) blocks of the block-row distributed matrix $\mathbf{A}$, with $k \geq 0$.

c) ILUT(k): Incomplete LU Factorization with Thresholding performed on the diagonal (local) blocks of the block-row distributed matrix $\mathbf{A}$. The number of additional elements per row in the L and U factors is computed as $\frac{(k-1)\text{nnz}(\mathbf{A})}{2n}$, with $k \geq 1$.

TABLE I
TEST MATRICES FROM THE SUITESPARSE MATRIX COLLECTION AND MATRIX MARKET: n DENOTES THE DIMENSION AND `NNZ` THE NUMBER OF NONZEROS. `NP` DENOTES THE NUMBER OF PROCESSES, `PREC` THE PRECONDITIONER, `cTol` THE CONVERGENCE TOLERANCE, AND m THE RESTART LENGTH USED IN THE EXPERIMENTS WITH THE CORRESPONDING TEST MATRIX.

label	matrix	n	nnz/n	np	prec	cTol	m
e200	e20r5000	4 241	31.0	1	ILU(8)	10^{-14}	60
mat3	matrix-new_3	125 329	7.1	4	ILUT(1.)	10^{-14}	200
xen2	xenon2	157 464	24.6	4	COLEQUI	10^{-14}	300
radn	radiation	223 104	24.8	8	COLEQUI	10^{-12}	200
af_3	af_shell3	504 855	34.8	16	COLEQUI	10^{-7}	300
atml	atmosmodl	1 489 752	6.9	64	COLEQUI	10^{-14}	300
ML_r	ML_Geer	1 504 002	73.6	64	COLEQUI	10^{-6}	400

B. Experiments with the monomial basis

We present performance results for application matrices from Table I and show strong and weak scaling experiments involving 3D Laplace problems. All s-step methods were evaluated with $s = 5$, a value commonly used with the monomial basis [24, 34, 35].

1) Application matrices: Table II compares the randomized methods with BCGSI2-GMRES in terms of number of iterations, runtime, and speedup. Additionally, we compare BCGSI2-GMRES with GMRES(m) based on the classical Gram-Schmidt process with reorthogonalization, which is known for its excellent stability and low communication overhead among non-blocked orthogonalization methods [20]. For BCGSI2-GMRES, we present the mean speedup of 20 runs over a single run of GMRES(m), while for the randomized methods, the median speedup of 20 runs over BCGSI2-GMRES is reported. In each run, the randomized methods sampled a CountSketch embedding of size $d = 600$ when $m \leq 300$, and $d = 1500$ when $m = 400$.

The step size did not change for all test cases in Table II, except for `mat3`, where poor conditioning in the initial MPK-generated basis caused BCGSI2-GMRES to reduce the step size to $s = 3$. Similarly, the randomized methods reduced the step size to $s = 4$ for the same reason.

TABLE II
FOR BCGSI2-GMRES, THE NUMBER OF ITERATIONS UNTIL CONVERGENCE (#ITS), MEAN RUNTIME (IN SECONDS) OVER 20 RUNS, AND SPEEDUP S_{GMRES} OVER A SINGLE RUN OF GMRES(m) ARE SHOWN. FOR THE RANDOMIZED ALGORITHMS RBGS-GMRES AND RTBGS-GMRES, THE DIFFERENCE Δ_{ITS} TO BCGSI2-GMRES IN TERMS OF MEDIAN NUMBER OF ITERATIONS UNTIL CONVERGENCE AND THE MEDIAN SPEEDUP OVER BCGSI2-GMRES ACROSS 20 RUNS ARE REPORTED, WITH INTERQUARTILE RANGES (IQR) IN BRACKETS.

matrix	BCGSI2-GMRES			RBGS-GMRES		RTBGS-GMRES	
	#its	time [s]	S_{GMRES}	Δ_{its}	S_{BCGSI2}	Δ_{its}	S_{BCGSI2}
mat3	156	0.97	1.73	6.0 (44)	1.55(0.52)	4.0 (12)	**1.97(0.12)**
xen2	2875	17.07	3.60	0.0 (40)	2.15(0.05)	220.0(193)	**3.78(0.45)**
radn	285	2.02	2.75	2.5 (5)	1.55(0.04)	45.0 (10)	**1.75(0.03)**
af_3	2310	42.95	3.47	−7.5(418)	1.83(0.40)	117.5(540)	**2.46(0.77)**
atml	445	4.30	4.07	0.0 (0)	2.29(0.01)	0.0 (0)	**4.81(0.06)**
ML_r	745	15.95	3.17	25.0 (5)	1.43(0.02)	35.0 (5)	**1.93(0.02)**

The test matrix `e200` in Table I is much too small for efficient parallelization and is therefore not included in Table II (nor in Table III). Figure 1 illustrates the convergence histories for `e200`. The RTBGS-GMRES algorithm required several early restarts due to poor conditioning of the Krylov basis. The truncation parameter t was doubled at each restart until it eventually reached $t = m$. This resulted in slower convergence and increased computational cost, making this strategy less suitable for test matrix `e200`.

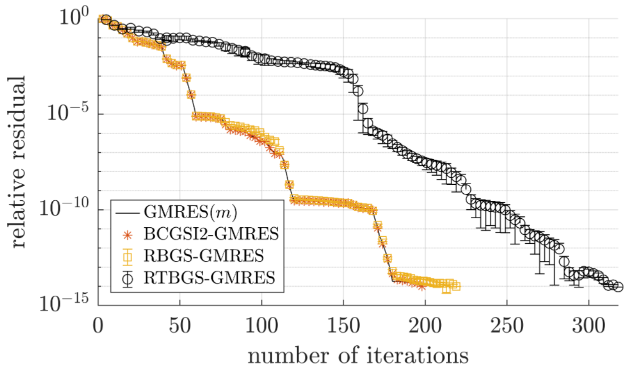

Fig. 1. Convergence histories for `e200`. The randomized methods show residual means with 95% confidence intervals based on 20 runs. BCGSI2-GMRES used the ICE with a condition number bound of $\Omega = 10^{-1}u^{-1/2}$ for Krylov basis orthogonalization, while the randomized methods used $\Omega = 10^{-2}u^{-1/2}$. With an initial step size of $s = 5$, the ICE restricted all s-step methods to three steps per iteration.

Although RTBGS-GMRES does not initiate early restarts due to ill-conditioning for the other test cases in Table I, it generally requires more iterations to converge than RBGS-GMRES and BCGSI2-GMRES. However, the strongly reduced cost per iteration results in the best overall performance, achieving a speedup of up to 4.81× (for test matrix `atml`). Note that larger IQR values in Table II indicate greater variability in the performance of the randomized methods.

2) Strong and weak scaling experiments: We present strong scaling experiments for a 3D Laplace problem of size 64M (1M = 10^6) and restart length $m = 400$. The randomized methods used a CountSketch embedding of size $d = 1500$. Figure 2 illustrates runtime performance with a breakdown of the most important kernels including the MPK, the projection and normalization steps of orthogonalization, the sketching of high-dimensional vectors performed by the randomized methods, and local computation represented by the "Other" kernel. In BCGSI2-GMRES and RBGS-GMRES, a major part of this kernel consists of the computation of the Hessenberg matrix. In RTBGS-GMRES, the "Other" kernel primarily consists of computing and factorizing the implicitly sketched Krylov basis $\mathbf{C}_{1:i-1} = \mathbf{\Theta} \mathbf{A} \mathbf{Q}_{1:i-1}$ (see lines 19–24 of Algorithm 3). The speedup of RTBGS-GMRES mainly results from the reduced projection cost achieved through truncated orthogonalization. The speedup values corresponding to Figure 2 are shown in Figure 3. For 2^{12} processes and beyond, all methods deviate from linear speedup. The deviation is stronger in the randomized methods, as local operations on sketched objects become relatively more expensive when the number of rows per process decreases and approaches the sketch dimension d. Note that with 2^{13} processes, the degree of parallelism is very high, with an average of only 7812.5 rows per processor. In contrast, the systems involving matrices from Table I are distributed across 23 277 to 39 366 rows per processor, corresponding to a parallelization degree of approximately 2^{11} processes in Figure 3.

Figure 4 presents weak scaling experiments for 3D Laplace problems of sizes 1M to 128M. Although BCGSI2-GMRES requires four times the number of global synchronizations compared to the randomized methods, all the methods indicate similar scaling behavior. A possible explanation for this could be that the number of nodes is still relatively low (at most 64 nodes, with 128 processes per node). The relatively large fraction of runtime for "Other" for RBGS-GMRES using 2^{13} processes still remains to be analyzed. The computations summarized in this category do not involve communication, nor do they depend on the matrix size or increase locally.

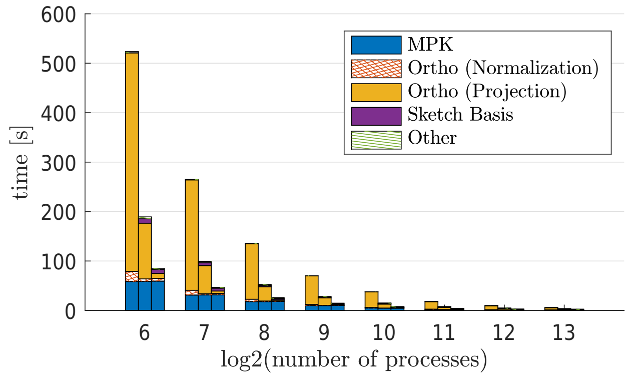

Fig. 2. Mean runtime performance over 20 runs (with negligible variance in runtime) for strong scaling experiments with a 3D Laplace matrix of size 64M and convergence tolerance 10^{-7}. Algorithms in a bar group: BCGSI2-GMRES (left), RBGS-GMRES (middle), and RTBGS-GMRES (right).

C. Experiments with the scaled Newton basis

We present performance results for application matrices comparable to those in Section V-B1, but for the scaled

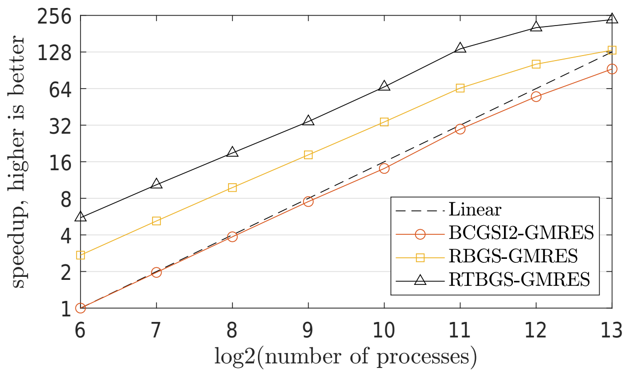

Fig. 3. Mean speedup over 20 runs for strong scaling experiments with a 3D Laplace matrix of size 64M, using BCGSI2-GMRES with 2^6 processes as the baseline for all methods.

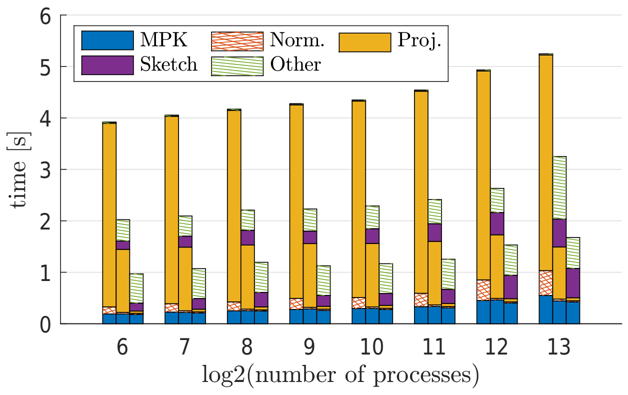

Fig. 4. Mean runtime over 20 runs for weak scaling experiments with 3D Laplace matrices ranging from 1M to 128M (15,625 rows per process). Algorithms and kernels are as in Figure 2. The s-step methods terminated when the relative residual improvement exceeded that of GMRES(m) at the 396^{th} iteration (which happens after approximately 400 iterations).

Newton basis. Additionally, we report the performance of randomized s-step GMRES methods with different subspace embeddings of varying sizes.

1) Application matrices: Table III presents results for the scaled Newton basis. The randomized methods used the same sketch type and dimensions as those in Section V-B1. For each test case, 100 Ritz values were computed as a preprocessing step using Arnoldi's method. The initial step size $\hat{s}$ in Table III was obtained using the step size estimator from [34]. We determined the optimal step size as the smallest value that, if reduced by the ICE, remained unchanged until convergence. The results show that the gap between the estimated and optimal step sizes can be significant. However, we made the interesting observation that, in some cases, the truncated orthogonalization of RTBGS-GMRES leads to optimal step sizes that are closer to the estimate, which benefits performance. In this regard, the randomized methods outperform BCGSI2-GMRES in all cases, with RTBGS-GMRES achieving a speedup of 2.35× over BCGSI2-GMRES for the matrix `xen2`.

2) Different subspace embeddings of varying sizes: Figures 5–7 illustrate the performance of randomized s-step GM-

TABLE III

$\hat{s}$ DENOTES THE ESTIMATE OF THE OPTIMAL STEP SIZE s_{OPT} FOR BCGSI2-GMRES AND RBGS-GMRES, AND OF THE OPTIMAL STEP SIZE $\bar{s}_{\text{OPT}}$ FOR RTBGS-GMRES, RESPECTIVELY. THE OTHER PARAMETERS ARE AS IN TABLE II.

matrix	$\hat{s}$ $/s_{\text{opt}}$ $/\bar{s}_{\text{opt}}$	BCGSI2-GMRES			RBGS-GMRES		RTBGS-GMRES	
		#its	time [s]	S_{GMRES}	Δ_{its}	S_{BCGSI2}	Δ_{its}	S_{BCGSI2}
`mat3`	100/ 3 / 3	156	1.23	1.37	3 (43)	1.34 (0.37)	3 (0)	**1.62 (0.01)**
`xen2`	100/ 15 / 30	2910	9.28	6.64	870 (773)	1.23 (0.22)	−30 (60)	**2.35 (0.05)**
`radn`	14 / 12 / 12	284	1.40	3.97	12 (0)	1.25 (0.02)	60 (12)	**1.25 (0.06)**
`af_3`	97 / 20 / 75	2320	21.90	6.80	−10 (420)	1.34 (0.26)	−70 (251)	**1.57 (0.30)**
`atml`	100/100/100	500	1.16	15.20	0 (0)	1.36 (0.01)	0 (0)	**1.47 (0.02)**
`ML_r`	85 / 24 / 66	762	9.68	5.23	24 (0)	1.09 (0.01)	34 (0)	**1.31 (0.01)**

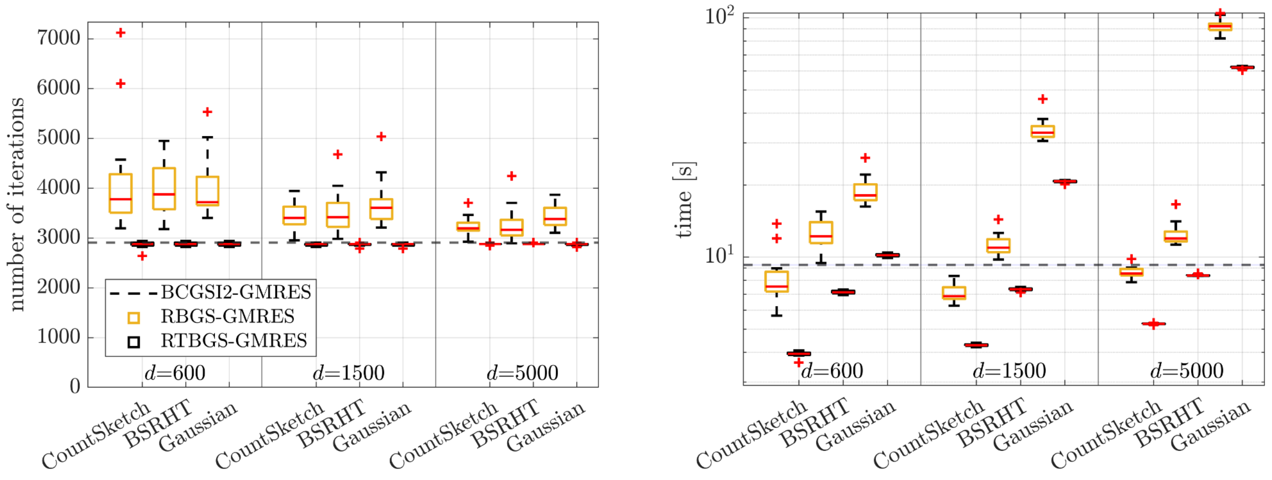

Fig. 5. Performance of randomized s-step GMRES methods for test matrix `xen2` using various embedding types and sketch dimensions d, based on 20 runs. The legend in the left plot corresponds to both plots. For BCGSI2-GMRES, the mean runtime is shown as a dashed horizontal line. The variance in runtime was negligible. Boxplots represent the randomized methods, alternating between RBGS-GMRES (yellow boxes) and RTBGS-GMRES (black boxes), starting with RBGS-GMRES. **Left:** Number of iterations until convergence with respect to `cTol`, listed in Table I. **Right:** Runtime performance (log scale).

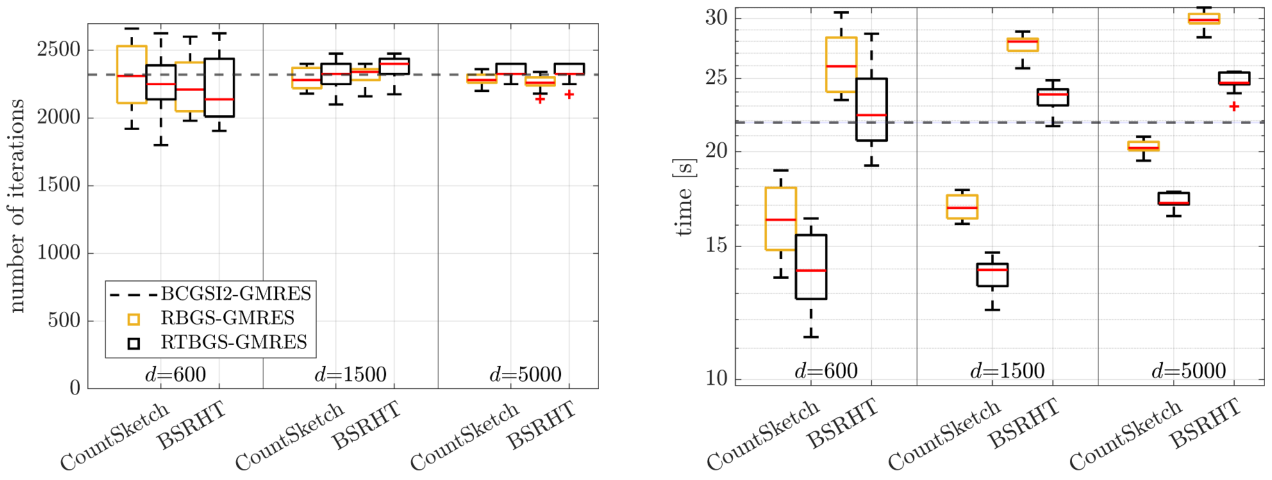

Fig. 6. Performance of randomized s-step GMRES methods for test matrix `af_3` (description as in Figure 5). The Gaussian embedding was not evaluated due to the high computational cost for $d = 5000$.

RES methods with different subspace embeddings of varying sizes. The results show no clear advantage in the number of iterations until convergence among the different subspace embeddings. The number of iterations is more influenced by the sketch dimension d (see Figures 5 and 6). Therefore, CountSketch embeddings are recommended, as their parallel application time is significantly faster compared to BSRHT and Gaussian sketches. In Figure 5, RTBGS-GMRES exhibits

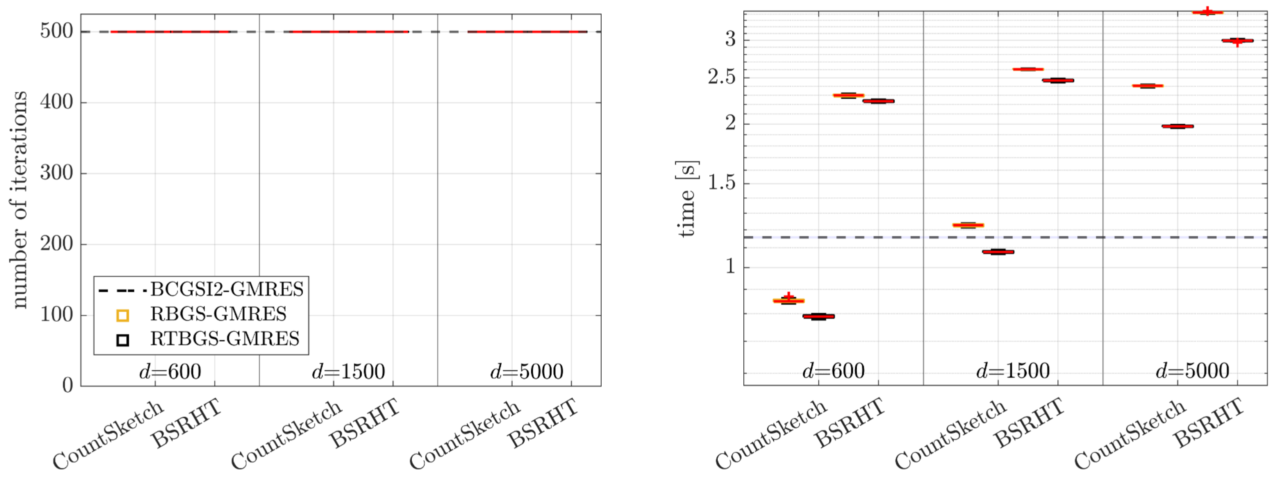

Fig. 7. Performance of randomized s-step GMRES methods for matrix `atml` (description as in Figure 5). The optimal step size is $s = 100$ for all methods, leading to convergence within the same number of iterations. The Gaussian embedding was not evaluated due to the high computational cost for $d = 5000$.

less sensitivity to the sketch dimension size in terms of number of iterations compared to RBGS-GMRES. This suggests that RBGS-GMRES could benefit from solving the sketched least-squares problem in (10) rather than relying on the Arnoldi relation. Additionally, convergence of all methods within the same step is more likely with very large step sizes (see Figure 7), as the residual improvement is only checked every s steps.

VI. Conclusion

We investigated parallel deterministic and randomized s-step GMRES methods with a focus on stability, performance, and scalability. The randomized methods, RBGS-GMRES and the novel RTBGS-GMRES algorithm proposed in this paper, require only one global synchronization to advance s steps while maintaining sufficient numerical stability. In contrast, the deterministic algorithm BCGSI2-GMRES requires four global synchronizations. For matrices of sizes up to $128 \cdot 10^6$, all methods demonstrate similar behavior in terms of weak scalability. However, with appropriately sized CountSketch embeddings, the randomized methods outperform the deterministic BCGSI2-GMRES algorithm in all test cases, and our novel RTBGS-GMRES algorithm outperforms the other methods. In particular, RTBGS-GMRES achieves speedups over BCGSI2-GMRES of up to $4.81\times$ using the monomial basis and up to $2.35\times$ using the scaled Newton basis.

Acknowledgment

The computational results presented have been achieved using the Vienna Scientific Cluster (VSC-5).

References

[1] Z. Bai, D. Hu, and L. Reichel. "A Newton basis GMRES implementation". In: *IMA Journal of Numerical Analysis* 14.4 (Oct. 1994), pp. 563–581.

[2] A. H. Baker, E. R. Jessup, and T. Manteuffel. "A technique for accelerating the convergence of restarted gmres". eng. In: *SIAM journal on matrix analysis and applications* 26.4 (2005), pp. 962–984.

[3] O. Balabanov. "Randomized Cholesky QR factorizations". Version 2. In: *arXiv* (2022).

[4] O. Balabanov, M. Beaupère, L. Grigori, and V. Lederer. "Block subsampled randomized Hadamard transform for low-rank approximation on distributed architectures". working paper or preprint. Oct. 2022.

[5] O. Balabanov and L. Grigori. "Randomized block Gram-Schmidt process for solution of linear systems and eigenvalue problems". working paper or preprint. Dec. 2021.

[6] O. Balabanov and L. Grigori. "Randomized Gram-Schmidt Process with Application to GMRES". In: *SIAM Journal on Scientific Computing* 44.3 (2022), A1450–A1474.

[7] O. Balabanov and A. Nouy. "Randomized Linear Algebra for Model Reduction. Part I: Galerkin Methods and Error Estimation". In: *Advances in Computational Mathematics* 45.5–6 (2019), pp. 2969–3019.

[8] E. Bavier, M. Hoemmen, S. Rajamanickam, and H. Thornquist. "Amesos2 and Belos: Direct and iterative solvers for large sparse linear systems". In: *Scientific Programming* 20.3 (July 2012), pp. 241–255.

[9] C. H. Bischof. "Incremental Condition Estimation". In: *SIAM journal on matrix analysis and applications* 11.2 (1990), pp. 312–322.

[10] C. H. Bischof and P. T. Tang. *Robust Incremental Condition Estimation*. Tech. rep. Argonne National Laboratory, IL (United States), 1991.

[11] C. Boutsidis and A. Gittens. "Improved Matrix Algorithms via the Subsampled Randomized Hadamard Transform". In: *SIAM Journal on Matrix Analysis and Applications* 34.3 (2013), pp. 1301–1340.

[12] L. Burke, S. Güttel, and K. M. Soodhalter. "GMRES with randomized sketching and deflated restarting". Version 2. In: *arXiv* (2023).

[13] E. Carson, K. Lund, Y. Ma, and E. Oktay. "On the loss of orthogonality in low-synchronization variants of reorthogonalized block classical Gram-Schmidt". Version 1. In: *arXiv* (2024).

[14] E. Carson, K. Lund, Y. Ma, and E. Oktay. "Reorthogonalized Pythagorean variants of block classical Gram-Schmidt". Version 3. In: *arXiv* (2024).

[15] E. Carson and Y. Ma. "A stable one-synchronization variant of reorthogonalized block classical Gram-Schmidt". Version 1. In: *arXiv* (2024).

[16] E. Carson and Y. Ma. "On the backward stability of s-step GMRES". Version 1. In: *arXiv* (2024).

[17] K. Clarkson and D. Woodruff. "Low-Rank Approximation and Regression in Input Sparsity Time". In: *Journal of the ACM* 63.6 (2017).

[18] M. B. Cohen. "Nearly tight oblivious subspace embeddings by trace inequalities". In: *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '16. Arlington, Virginia: Society for Industrial and Applied Mathematics, 2016, pp. 278–287.

[19] E. De Sturler. "Truncation Strategies for Optimal Krylov Subspace Methods". eng. In: *SIAM journal on numerical analysis* 36.3 (1999), pp. 864–889.

[20] L. Giraud, J. Langou, M. Rozložník, and J. Eshof. "Rounding error analysis of the classical Gram-Schmidt orthogonalization". In: *Numerische Mathematik* 101 (Jan. 2005), pp. 87–100.

[21] S. Güttel and I. Simunec. "A Sketch-and-Select Arnoldi Process". In: *SIAM Journal on Scientific Computing* 46.4 (2024), A2774–A2797.

[22] N. Halko, P. G. Martinsson, and J. A. Tropp. "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions". In: *SIAM Review* 53.2 (2011), pp. 217–288.

[23] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, A. Williams, and K. Stanley. "An overview of the Trilinos project". In: *ACM Transactions on Mathematical Software* 31.3 (Sept. 2005), pp. 397–423.

[24] M. Hoemmen. *Communication-Avoiding Krylov Subspace Methods*. 2010.

[25] D. Imberti and J. Erhel. "Varying the s in Your s-step GMRES". In: *Electronic Transactions on Numerical Analysis* 47 (2017), pp. 206–230.

[26] Y. Jang and L. Grigori. "Randomized orthogonalization process with reorthogonalization". working paper or preprint. Sept. 2024.

[27] Y. Jang, L. Grigori, E. Martin, and C. Content. "Randomized Flexible GMRES with Singular Vectors Based Deflated Restarting". In: *SSRN* (2022).

[28] Y. Li, H. Lin, S. Liu, A. Vakilian, and D. Woodruff. "Learning the Positions in CountSketch". In: *The Eleventh International Conference on Learning Representations*. 2023.

[29] R. Murray, J. Demmel, M. W. Mahoney, N. B. Erichson, M. Melnichenko, O. A. Malik, L. Grigori, P. Luszczek, M. Derezinski, M. E. Lopes, T. Liang, H. Luo, and J. Dongarra. *Randomized Numerical Linear Algebra: A Perspective on the Field With an Eye to Software*. Tech. rep. UCB/EECS-2023-19. EECS Department, University of California, Berkeley, Feb. 2023.

[30] Y. Nakatsukasa and J. A. Tropp. "Fast and Accurate Randomized Algorithms for Linear Systems and Eigenvalue Problems". In: *SIAM journal on matrix analysis and applications* 45.2 (2024), pp. 1183–1214.

[31] B. Philippe and L. Reichel. "On the generation of Krylov subspace bases". In: *Applied numerical mathematics* 62.9 (2012), pp. 1171–1186.

[32] A. Prokopenko, C. M. Siefert, J. J. Hu, M. Hoemmen, and A. Klinvex. *Ifpack2 User's Guide 1.0*. Tech. rep. SAND2016-5338. Sandia National Labs, 2016.

[33] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003.

[34] Z. Xu, J. J. Alonso, and E. Darve. "A Numerically Stable Communication-Avoiding s-Step GMRES Algorithm". In: *SIAM Journal on Matrix Analysis and Applications* 45.4 (2024), pp. 2039–2074.

[35] I. Yamazaki, A. J. Higgins, E. G. Boman, and D. B. Szyld. "Two-Stage Block Orthogonalization to Improve Performance of s-step GMRES". In: *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2024, pp. 26–37.

[36] I. Yamazaki, H. Anzt, S. Tomov, M. Hoemmen, and J. Dongarra. "Improving the Performance of CA-GMRES on Multicores with Multiple GPUs". In: *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. 2014, pp. 382–391.

[37] I. Yamazaki, M. Hoemmen, P. Luszczek, and J. Dongarra. "Improving Performance of GMRES by Reducing Communication and Pipelining Global Collectives". In: *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. 2017, pp. 1118–1127.

[38] I. Yamazaki, S. Thomas, M. Hoemmen, E. G. Boman, K. Świrydowicz, and J. J. Elliott. "Low-synchronization orthogonalization schemes for s-step and pipelined Krylov solvers in Trilinos". In: *Proceedings of the 2020 SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, 2020, pp. 118–128.