universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Scalable GMRES methods

verfasst von | submitted by

Robert Ernstbrunner BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 921
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Informatik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Wilfried Gansterer M.Sc.

Abstract

Large sparse linear systems arise in many complex and computationally intensive problems across various scientific fields. High Performance Computing (HPC) and highly parallelizable iterative methods are important tools for solving these systems efficiently. Communication – the process of moving data within the HPC system – can become a significant bottleneck when solving very large problems, which motivated the development of communication-avoiding iterative methods. At the same time, randomization has had a profound impact on numerical linear algebra in recent years. Randomization techniques work with many existing algorithms, often providing improved scalability and orders-of-magnitude speedups over deterministic approaches. With these aspects in mind, this master's thesis investigates ideas from randomized numerical linear algebra applied to communication-avoiding iterative linear system solvers. In particular, we explore state-of-the-art deterministic and randomized s-step Generalized Minimal Residual (GMRES) methods and propose a novel randomized s-step variant called RTBGS-GMRES that improves performance in the construction of the basis for the solution subspace while minimizing the number of global synchronizations in parallel computing environments. We compare our novel method 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 randomized GMRES methods generally outperform the parallel deterministic s-step method BCGSI2-GMRES. Our novel RTBGS-GMRES method outperforms the other methods and achieves speedups of about $2\times$ and about $4\times$ over BCGSI2-GMRES for two different basis types.

Zusammenfassung

Große dünn besetzte lineare Gleichungssysteme sind zentral für viele komplexe und rechenintensive Probleme in verschiedenen wissenschaftlichen Bereichen. Hochleistungsrechnen (High Performance Computing, HPC) und hochgradig parallelisierbare iterative Verfahren sind wichtige Werkzeuge, um diese Systeme effizient zu lösen. Kommunikation – der Prozess der Datenübertragung innerhalb des HPC-Systems – kann zu einem erheblichen Engpass werden, insbesondere bei der Lösung sehr großer Probleme, was die Entwicklung kommunikationsvermeidender iterativer Verfahren motiviert hat. Gleichzeitig hat die Randomisierung in den letzten Jahren einen tiefgreifenden Einfluss auf die numerische lineare Algebra gehabt. Randomisierungstechniken lassen sich in viele bestehende Algorithmen integrieren und bieten häufig eine verbesserte Skalierbarkeit und Geschwindigkeitssteigerungen im Vergleich zu deterministischen Ansätzen. Vor diesem Hintergrund untersucht diese Masterarbeit Ideen aus der randomisierten numerischen linearen Algebra, die auf kommunikationsvermeidende iterative lineare Systemlöser angewendet werden. Insbesondere betrachten wir den Stand der Technik bei deterministischen und randomisierten *s*-step Generalized Minimal Residual (GMRES) Methoden und schlagen eine neue randomisierte *s*-step Variante namens RTBGS-GMRES vor, die die Performance beim Aufbau der Basis für den Lösungsunterraum verbessert und dabei die Anzahl der globalen Synchronisierungen in parallelen Rechenumgebungen minimiert. Wir vergleichen unsere neue Methode mit modernen randomisierten und deterministischen *s*-step GMRES-Methoden hinsichtlich numerischer Stabilität, Konvergenz, Performance und Skalierbarkeit. Numerische Experimente auf einem großen Cluster zeigen, dass die randomisierten GMRES-Methoden bei geeigneten Parameterwerten die parallele deterministische *s*-step Methode BCGSI2-GMRES in der Regel übertreffen. Unsere neue RTBGS-GMRES Methode übertrifft die anderen Methoden und erzielt Geschwindigkeitssteigerungen von etwa 2× und etwa 4× gegenüber BCGSI2-GMRES für zwei verschiedene Basistypen.

Preface

The following parts were reused in a conference submission that was accepted for publication in IPDPS'25[1] on October 11, 2024: the Abstract and Notation; Chapter 1 (Introduction and Sections 1.2–1.4); Chapter 3 (Sections 3.1 and 3.3); Chapters 4 and 5; Chapter 6 (Section 6.2); and Chapter 7.

[1] IEEE International Parallel and Distributed Processing Symposium (IPDPS), "IPDPS 2025 Conference". Available: https://www.ipdps.org/ipdps2025/. Accessed: March 30, 2025.

Contents

Notation

We define general notation for this work.

- Scalars: italic lower-case letters, e.g., x
- Column vectors: bold lower-case letters, e.g., $\mathbf{x}$
- Matrices: bold upper-case letters, e.g., $\mathbf{X}$
- u: unit roundoff or machine epsilon (the difference between 1 and the next larger floating point number)
- $\mathbf{X}^{-1}$: inverse of $\mathbf{X}$
- $\mathbf{X}^{\dagger}$: Moore-Penrose inverse of $\mathbf{X}$
- $\sigma_{min}(\mathbf{X})$: minimum singular value of $\mathbf{X}$
- $\sigma_{max}(\mathbf{X})$: maximum singular value of $\mathbf{X}$
- $\kappa(\mathbf{X}) := \frac{\sigma_{max}(\mathbf{X})}{\sigma_{min}(\mathbf{X})}$, 2-norm condition number of $\mathbf{X}$
- $\mathbf{I}$: identity matrix
- $\mathbf{e}_i$: the i^{th} column vector of the identity matrix
- Definitions for a matrix or vector ξ:
  - ξ^{T}: transpose of ξ
  - $\|\xi\|$: spectral norm of a matrix, resp. Euclidean vector norm
  - $\|\xi\|_{\mathrm{F}}$: Frobenius norm
  - $\xi \perp \xi'$: ξ is orthogonal to ξ'

Row and column subset selection of matrices We use Matlab-like notation with 1-based indexing. Row and column selection of a matrix is indicated by index range subscripts. For example, consider 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. A single column or element in $\mathbf{X}$ is represented by a column vector or scalar, respectively. For example, $\mathbf{x}_k$ denotes the k^{th} column of $\mathbf{X}$, while $x_{i,k}$ denotes the element in row i and column k of $\mathbf{X}$. Note that a sequence of matrices is indicated by a subscript index. For example, $\mathbf{X}_1$ and $\mathbf{X}_2$ are two matrices of same size. Note that therefore, the k^{th} column vector of $\mathbf{X}$ is indicated by $\mathbf{x}_k$, *not* $\mathbf{X}_k$.

Iterands A vector with a superscript in round brackets denotes the sequence of an iterand. For example, $\mathbf{x}^{(i)}$ denotes the i^{th} update of the initial vector $\mathbf{x}^{(0)}$.

Chapter 1

Introduction

Large, sparse, and potentially nonsymmetric linear systems arise in many models of real-world problems. The Generalized Minimal Residual (GMRES) method is a widely used iterative method for solving such systems, particularly on High Performance Computing (HPC) machines consisting of interconnected nodes with multiple processors suitable for parallel execution. In general, the performance of an algorithm depends on both the number of *arithmetic operations* performed and the *communication* involved. Communication refers to the process of transferring data (for example, within the memory hierarchy or between nodes in an HPC system).
With continuous advancements towards more powerful machines, communication has become a significant cost in terms of time and energy consumption [33]. The demand for better performance has driven the development of both improved hardware and more efficient algorithms. Popular algorithm modifications either *hide* communication by overlapping it with other operations or *avoid* communication by restructuring the algorithm, often at the cost of introducing additional local operations. Communication-hiding pipelined GMRES methods [36] perform a single non-blocking reduction per iteration, overlapping dot products with sparse matrix-vector multiplication to minimize communication latency. Communication-avoiding s-step GMRES methods [45] advance s steps per iteration instead of one, reducing communication overhead by a factor of s over traditional algorithms. Communication-reducing modifications introduce potential sources of numerical instability in the algorithm [36, 45]. One potential source of instability in s-step GMRES methods arises from generating 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 [4, 59, 65]. If other basis types are ineffective, a further strategy involves varying the step size [46, 65]. Another potential source of instability arises from block orthogonalization of the Krylov basis. Some block orthogonalization methods are efficient, using fast BLAS-3 operations and reducing communication 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 [18].
Recently, randomized numerical linear algebra (RNLA) has emerged as a very exciting area in scientific computing [57]. RNLA is known for its versatility in enhancing the performance of various algorithms. Randomization techniques based on subspace embeddings [42] have been applied to GMRES methods, offering the potential to improve algorithm performance, stability, and scalability. To keep the scope manageable, this work focuses on communication-avoiding s-step GMRES methods, for which randomization strategies are already established. Specifically, we investigate state-of-the-art s-step GMRES methods and their integration with novel randomization ideas.

1.1 Research goal

Our research aims to enhance the performance and scalability of GMRES methods for large sparse linear problems. We first establish the foundation of GMRES methods, review state-of-the-art developments, and then advance our goal of designing more efficient algorithms.

1.2 Related work

See [4, 23, 28, 49, 50, 56] for the development history of s-step GMRES methods. The seminal work of Hoemmen et al. [45] presents s-step GMRES methods with efficient communication-avoiding kernels. Imberti et al. [46] 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. [65] propose mechanisms to account for instabilities arising from advancing s steps rather than just one. One of these mechanisms estimates the step size at which the s-step GMRES algorithm maintains stability. Yamazaki et al. [70] present low-synchronization schemes for s-step GMRES methods, although without formal proofs of their stability properties. In [68], parallel s-step GMRES methods with GPU acceleration are investigated. The authors in [69] compare the performance of pipelined and s-step GMRES methods, with their combined use demonstrating the best performance in numerical experiments. Furthermore, the authors propose using different step sizes for individual s-step GMRES kernels to achieve the best performance for each kernel. An s-step GMRES algorithm with a "two-stage" orthogonalization process is introduced in [67]. This approach reduces communication costs in parallel execution compared to traditional block orthogonalization, but it lacks a rigorous stability analysis [19]. The work in [22] provides insight into the backward stability of s-step GMRES methods. The authors in [21] present s-step GMRES with stable block orthogonalization schemes that require at most two global synchronizations every s steps.
Nakatsukasa et al. [58] 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. [9] introduce a randomized orthogonalization process that enhances the stability and scalability of GMRES methods. The work of [9] is extended in [8], introducing a randomized block orthogonalization process suitable for s-step GMRES algorithms. The authors in [17] apply the randomization strategy from [58] to GMRES methods with deflated restarting. Güttel et al. [41] present a randomized orthogonalization method similar to that in [9], 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. [47] investigate randomized orthogonalization with deterministic reorthogonalization, with application to GMRES. Flexible GMRES (FGMRES) [62] is a generalization of GMRES that allows the preconditioner to change in every iteration. A randomized FGMRES method was introduced in [48].
Our bachelor's thesis discusses only one small aspect of this topic: the s-step GMRES method from [45] (Algorithm 23). This master's thesis has much more breadth as well as depth, more specifically:

- This work provides a comprehensive and detailed discussion of Krylov subspace methods and GMRES, offering a much deeper algorithmic understanding of basic concepts.

- The bachelor's thesis focused only on one particular s-step GMRES method, while this work explores various advanced state-of-the-art s-step methods that also detect and avoid numerical instabilities. The method studied in the bachelor's thesis uses a different orthogonalization process, but is otherwise similar to the algorithm in [65], referred to here as BCGSI2-GMRES (see Algorithm 8 in Section 3.3).

- The bachelor's thesis contains only numerical results obtained with *sequential* C++ code using the Intel MKL library [64]. In this work, we fully implemented parallel s-step GMRES methods using the Trilinos C++ library and conducted extensive parallel scaling experiments on the Vienna Scientific Cluster, a large HPC system.

1.3 Scientific gap

- The authors in [22] present stability results and limitations for s-step GMRES using various orthogonalization methods, including the block classical Gram-Schmidt process with inner reorthogonalization (BCGSI2; see Section 3.2.2 for details). Specifically, their analysis focuses on BCGSI2 with *unconditionally* stable intra-orthogonalization. The stability of BCGSI2-GMRES in earlier work [65] has not yet been thoroughly explored. BCGSI2-GMRES employs *conditionally* stable intra-orthogonalization, but incorporates techniques to mitigate numerical instabilities, such as novel basis types and adaptive step size reduction. The impact of these mechanisms on stability has not been fully analyzed.

- Global synchronizations are costly as they require communication between all participating processes. We can balance the number of global synchronizations with numerical stability in iterative s-step methods. For example, BCGSI2-GMRES needs at least four global synchronizations every s steps to retain the convergence properties of standard GMRES for difficult problems [65]. The authors in [21] present s-step methods that require no more than two synchronizations while maintaining stability comparable to s-step GMRES using BCGSI2 with unconditionally stable intra-orthogonalization. Open questions remain regarding the practical performance of these algorithms compared to BCGSI2-GMRES. Furthermore, scalability and the number of global synchronizations required to achieve similar stability in randomized s-step GMRES methods have not been comprehensively analyzed.

- The existing literature does not fully address the practical aspects of randomized parallel s-step GMRES methods. The effectiveness of various subspace embeddings for these methods in practice is unclear, and their performance when restarted multiple times has not been thoroughly examined.

1.4 Contributions

- We experimentally evaluate various deterministic and randomized block orthogonalization methods in terms of stability.

- We extend the work in [8] by conducting parallel experiments with their randomized s-step GMRES algorithm that requires one global synchronization every s steps.

- We introduce a novel randomized s-step GMRES method called Randomized Truncated Block Gram-Schmidt GMRES (RTBGS-GMRES) combining randomization strategies from [8] and [58] 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 [65] 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 [65] 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 BCGSI2-GMRES method from [65] in terms of runtime, stability, and scalability. BCGSI2-GMRES uses efficient block orthogonalization and is particularly effective for large step sizes [65], achieving speedups of up to 15× over GMRES(m) in our numerical experiments.

1.5 Synopsis

- Chapter 2 provides an overview of the fundamental concepts behind basic GMRES methods, including various orthogonalization techniques (Section 2.1), a basic understanding of Krylov subspace methods (Section 2.2), and important properties specific to GMRES (Section 2.3).
- Chapter 3 introduces s-step GMRES methods, starting with two fundamental kernels: the matrix powers kernel for basis generation (Section 3.1) and block orthogonalization methods like the block classical Gram-Schmidt process (Section 3.2). Section 3.3 presents the adaptive s-step GMRES method BCGSI2-GMRES and reviews two strategies to mitigate potential sources of numerical instability specific to s-step GMRES methods. The first strategy involves a partial Cholesky factorization, leading to potential step size reduction to addresses numerical issues caused by poor conditioning in the Krylov basis (Section 3.3.1). The second strategy consists of a step size estimator that estimates the step size for which the s-step algorithm remains free from such issues (Section 3.3.2). Finally, Section 3.3.3 discusses stability and performance of BCGSI2-GMRES.
- Chapter 4 introduces two strategies for incorporating randomization into GMRES methods using several randomized sketching techniques discussed in Section 4.1. We present stability and performance results for both strategies. The first approach uses a sketched Krylov basis to achieve approximate orthogonalization of the original basis (Section 4.2). The second approach approximates the GMRES solution by sketching the GMRES least-squares problem (Section 4.3).
- Chapter 5 presents two randomized s-step GMRES methods: randomized s-step GMRES in Section 5.1, and randomized & sketched s-step GMRES in Section 5.2. The first method processes blocks of the sketched Krylov basis to approximately orthogonalize the original basis, similar to the method discussed in Section 4.2. The second method introduces our novel approach that combines randomized block orthogonalization with sketched least-squares problems, discussed in Section 4.3. Additionally, stability-preserving mechanisms, similar to those discussed in Chapter 3, are applied to both randomized s-step GMRES methods.

- Chapter 6 presents numerical experiments for deterministic and randomized s-step GMRES methods, starting with a stability comparison of various deterministic and randomized block orthogonalization methods with fixed step size s. The results indicate that the stability of randomized orthogonalization using one global synchronization is comparable to that of deterministic orthogonalization, which require at least four global synchronizations. However, all deterministic methods fail in problematic cases, where the input matrix is rank deficient, while randomized methods that succeed are too costly for practical use. In such cases, the authors in [65] suggest to adaptively reduce the step size for improved stability. The remaining tests address the parallel performance and scalability of deterministic and randomized s-step GMRES methods. With appropriately sized and efficient subspace embeddings, the randomized methods outperform the deterministic BCGSI2-GMRES algorithm for a set of application matrices, and our novel randomized algorithm outperforms the other methods. Furthermore, for 3D Laplace problems of sizes up to $128 \cdot 10^6$, all methods demonstrate similar scalability in terms of weak scalability.

- Finally, Chapter 7 offers some concluding remarks.

Chapter 2

Background

Basis orthogonalization is an essential component of GMRES methods. This chapter starts with an overview of the orthogonalization methods relevant to GMRES in Section 2.1, followed by a discussion of fundamental concepts, including Krylov subspace methods in Section 2.2, and concepts specific to GMRES in Section 2.3.

2.1 Orthogonalization methods

Orthogonalization methods take a set of m linearly independent vectors $\mathbf{V}$, where $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m] \in \mathbb{R}^{n\times m}$ and $n \geq m$ and perform a QR factorization $\mathbf{V} = \mathbf{QR}$, where $\mathbf{Q} = [\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_m] \in \mathbb{R}^{n\times m}$ is a basis for $\mathbf{V}$ with orthonormal columns and $\mathbf{R} \in \mathbb{R}^{m\times m}$ is upper triangular. A stable method is indicated by a low orthogonalization error

$$\|\mathbf{I} - \mathbf{Q}^{\mathrm{T}}\mathbf{Q}\| \tag{2.1}$$

representing the "loss of orthogonality" in $\mathbf{Q}$. The relative factorization error

$$\|\mathbf{QR} - \mathbf{V}\|/\|\mathbf{V}\| \tag{2.2}$$

and the relative error in the R factor $\|\mathbf{R}^{\mathrm{T}}\mathbf{R} - \mathbf{V}^{\mathrm{T}}\mathbf{V}\|/\|\mathbf{V}\|$ can be small, even if $\mathbf{Q}$ is not orthonormal. There are two primary approaches to perform the QR factorization:

1. *Orthogonal triangularization* applies a series of orthogonal projectors $\mathbf{Q}_i \in \mathbb{R}^{n\times n}$, with $1 \leq i \leq m$ to the left of $\mathbf{V}$ such that $\mathbf{R}' = \mathbf{Q}_m\mathbf{Q}_{m-1}\ldots\mathbf{Q}_1\mathbf{V}$ is upper triangular, and $\mathbf{Q}' = \prod_{i=1}^{m}\mathbf{Q}_i^{\mathrm{T}}$. Set $\mathbf{R} = \mathbf{R}'_{1:m,1:m}$ and $\mathbf{Q} = \mathbf{Q}'_{1:m}$ to obtain the factors defined above. This formulation is also known as *thin QR factorization* due to the reduced dimensions of the factors.

2. *Triangular orthogonalization* applies a series of upper triangular matrices to the right of $\mathbf{V}$ such that $\mathbf{Q} = \mathbf{V}\prod_{i=1}^{m}\mathbf{R}_i$, and $\mathbf{R} = \mathbf{R}_m^{-1}\mathbf{R}_{m-1}^{-1}\ldots\mathbf{R}_1^{-1}$.

Transformations with orthogonal projectors are known for their excellent numerical stability. It can be shown that any Q factor produced by orthogonal triangularization is orthonormal up to machine precision, regardless of $\kappa(\mathbf{V})$, the condition of $\mathbf{V}$ (see, for example, [44], Section 18.3). In contrast, the stability of triangular orthogonalization with respect to the orthogonalization error depends on the method used to compute the R factor. For triangular orthogonalization, we focus on Gram-Schmidt methods and the Cholesky QR (CholQR) decomposition, which are suitable for distributed vectors in parallel computing environments (see Section 2.3.4). For orthogonal triangularization, we discuss the Householder QR (HH-QR) factorization – a procedure often applied to low-dimensional vectors when high stability is required – and the Tall and Skinny QR (TSQR) factorization, a highly stable method suited for parallel computing environments.

2.1.1 Orthonormal Gram-Schmidt process

The Gram-Schmidt process [38] obtains the i^{th} orthonormal basis vector $\mathbf{q}_i$ by computing an orthogonal projection $\mathbf{q}'_i = (\mathbf{I} - \mathbf{Q}_{1:i-1}\mathbf{Q}_{1:i-1}^{\mathrm{T}})\mathbf{v}_i$ and its normalized representation $\mathbf{q}_i = \mathbf{q}'_i/\|\mathbf{q}'_i\|$. This process formulation corresponds to Algorithm 1 using

Algorithm 1 Gram-Schmidt process

Input: Set of vectors $\mathbf{V} \in \mathbb{R}^{n\times m}$, $m \leq n$
Output: Orthonormal $\mathbf{Q} \in \mathbb{R}^{n\times m}$, upper triangular $\mathbf{R} \in \mathbb{R}^{m\times m}$
1: **for** $i = 1 : m$ **do**
2: $\quad \mathbf{q}'_i = \mathbf{\Pi}^{(i-1)}\mathbf{v}_i$, obtain projection coefficients $\mathbf{r}_{1:i-1,i}$
3: $\quad$ Set $r_{i,i} = \|\mathbf{q}'_i\|$
4: $\quad \mathbf{q}_i = \mathbf{q}'_i/r_{i,i}$
5: **end for**

the classical Gram-Schmidt (CGS) projector $\mathbf{\Pi}_{CGS}^{(i-1)} := \mathbf{I} - \mathbf{Q}_{1:i-1}\mathbf{Q}_{1:i-1}^{\mathrm{T}}$ in line 2. CGS is prone to numerical instabilities. A different order of operations and reorthogonalization provides better stability, resulting in the modified Gram-Schmidt (MGS) process and CGS with reorthogonalization (CGS2). The respective projectors are defined as:

$$\mathbf{\Pi}_{MGS}^{(i)} := (\mathbf{I} - \mathbf{q}_i\mathbf{q}_i^{\mathrm{T}})(\mathbf{I} - \mathbf{q}_{i-1}\mathbf{q}_{i-1}^{\mathrm{T}})\dots(\mathbf{I} - \mathbf{q}_1\mathbf{q}_1^{\mathrm{T}}),$$
$$\mathbf{\Pi}_{CGS2}^{(i)} := \mathbf{\Pi}_{CGS}^{(i)}\mathbf{\Pi}_{CGS}^{(i)}.$$

The computation of the projection coefficients in line 2 of Algorithm 1 varies, depending on the projector used. For example, CGS and MGS first compute $r_{1,1} = \|\mathbf{v}_1\|$. For $i > 1$, CGS then proceeds with

$$\mathbf{r}_{1:i-1,i} = \mathbf{Q}_{1:i-1}^{\mathrm{T}}\mathbf{v}_i,$$

while MGS evaluates

$$\mathbf{r}_{1:j,i} = \mathbf{Q}_{1:j}^{\mathrm{T}}(\mathbf{v}_i - \mathbf{Q}_{1:j-1}\mathbf{r}_{1:j-1,i}), \text{ for } j = 1,\dots,i-1.$$

2.1.2 Cholesky QR factorization

The Cholesky QR (CholQR) factorization [66] uses the fact that $\mathbf{V}^{\mathrm{T}}\mathbf{V} = (\mathbf{QR})^{\mathrm{T}}\mathbf{QR} = \mathbf{R}^{\mathrm{T}}\mathbf{R}$. The method computes a Cholesky factorization of the Gram matrix $\mathbf{G} = \mathbf{V}^{\mathrm{T}}\mathbf{V}$ to obtain an R factor, used to compute $\mathbf{Q} = \mathbf{V}\mathbf{R}^{-1}$ via forward substitution. If $\mathbf{V}$ is distributed among processes, the computation of $\mathbf{G}$ requires one global reduction operation, where the reduction operation consists of the addition of small dense matrices [35]. Subsequently, forward substitution is performed in parallel without communication. CholQR is one of the fastest methods for orthogonalizing a set of vectors $\mathbf{V}$, but it assumes $\kappa(\mathbf{V}) \leq \mathcal{O}\left(u^{-1/2}\right)$ for numerical stability and provides only $\mathcal{O}(u)\,\kappa^2(\mathbf{V})$ loss of orthogonality, where u is the unit roundoff. If the algorithm is run twice (CholQR2), a loss of orthogonality of $\mathcal{O}(u)$ is guaranteed under the same assumptions. Note that, compared to Gram-Schmidt methods, the CholQR factorization is superior in terms of communication cost but expects all vectors in $\mathbf{V}$ to be readily available.

2.1.3 Householder QR factorization

To orthogonalize the first i columns of $\mathbf{V}$, with $1 \leq i < m$, the Householder QR (HH-QR) factorization employs Householder matrices $\mathbf{P}_i = \mathbf{I} - 2\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}$. These matrices are constructed using Householder reflectors $\mathbf{u}_i$ of unit length, chosen such that $\prod_{j=1}^{i} \mathbf{P}_{i+1-j}\mathbf{V}_{1:i}$ is upper triangular. To orthogonalize the next s columns $\mathbf{V}_{i+1:i+s}$, with $i + s \leq m$, the factorization performs the update $\mathbf{V}' = \prod_{j=1}^{i} \mathbf{P}_{i+1-j}\mathbf{V}_{i+1:i+s}$. Subsequently, $\tilde{\mathbf{V}} = \prod_{j=1}^{s} \mathbf{P}_{i+1+s-j}\mathbf{V}'$ is computed by determining s new reflectors such that $\tilde{\mathbf{V}}_{i+1:n,1:s}$ is upper triangular. Note that, as in Gram-Schmidt methods, columns $\mathbf{V}_{i+1:n}$ are not required to process the first i columns of $\mathbf{V}$ and can therefore be generated *a posteriori*. The first $i + s$ columns of $\prod_{j=1}^{i+s} \mathbf{P}_j$ form an orthonormal basis for $\mathbf{V}_{1:i+s}$ which, in many cases, is not computed explicitly unless required.
For details on HH-QR and how to determine reflectors, see [38].

2.1.4 Tall and Skinny QR factorization

The Tall and Skinny QR (TSQR) factorization employs a divide-and-conquer strategy to orthonormalize $\mathbf{V}$, working through a reduction tree. In parallel, this approach is similar to CholQR, as both require a global reduction where all vectors must be readily available for processing. However, TSQR is approximately twice as costly in terms of floating-point operations. Additionally, the reduction operation involves an unconditionally stable QR factorization, unlike CholQR, which uses addition in its reduction process. For details on the TSQR factorization algorithm, see [30].

Properties concerning algorithm stability and suitability for parallel computations of various orthogonalization methods are shown in Tables 2.1 and 2.2.

TABLE 2.1: Bounds on the "loss of orthogonality" in the orthonormal basis $\mathbf{Q}$, resulting from different methods for orthogonalizing $\mathbf{V}$, some of which require prior assumptions made on $\kappa(\mathbf{V})$ with respect to the unit roundoff u.

Algorithm	$\|\mathbf{I} - \mathbf{Q}^T\mathbf{Q}\|$ bound	Assumption on $\kappa(\mathbf{V})$	Reference
CGS	$\mathcal{O}(u)\,\kappa^{n-1}(\mathbf{V})$	$\mathcal{O}(u)\,\kappa(\mathbf{V}) < 1$	[52]
MGS	$\mathcal{O}(u)\,\kappa(\mathbf{V})$	$\mathcal{O}(u)\,\kappa(\mathbf{V}) < 1$	[15]
CGS2	$\mathcal{O}(u)$	$\mathcal{O}(u)\,\kappa(\mathbf{V}) < 1$	[37]
HH-QR	$\mathcal{O}(u)$	none	[38]
TSQR	$\mathcal{O}(u)$	none	[30, 38]
CholQR	$\mathcal{O}(u)\,\kappa^2(\mathbf{V})$	$\mathcal{O}(u)\,\kappa^2(\mathbf{V}) < 1$	[66]
CholQR2	$\mathcal{O}(u)$	$\mathcal{O}(u)\,\kappa^2(\mathbf{V}) < 1$	[66]

TABLE 2.2: Communication-avoiding and communication-expensive orthogonalization methods. TSQR and CholQR are methods that minimize communication overhead. Unlike communication-expensive methods, they require all vectors to be readily available for processing.

	Orthogonal triangularization	Triangular orthogonalization
Communication expensive	HH-QR	Gram-Schmidt
Communication avoiding	TSQR	CholQR

2.2 Krylov subspace methods

Krylov subspace methods are a useful framework for solving large sparse linear systems and for finding eigenvalues and eigenvectors. The solution of a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{A} \in \mathbb{R}^{n\times n}$ is approximated by constructing a sequence of nested subspaces of the solution space, known as Krylov subspaces. Krylov subspace methods find the best approximation of $\mathbf{x}$ within these subspaces. We provide insight into why the (approximate) solution is embedded in a Krylov subspace involving $\mathbf{A}$ and $\mathbf{b}$. The i^{th} Krylov subspace, with respect to $\mathbf{A}$ and $\mathbf{x}$ is given by

$$\mathcal{K}_i(\mathbf{A}, \mathbf{x}) := \text{span}\{\mathbf{x}, \mathbf{A}\mathbf{x}, \ldots, \mathbf{A}^{i-1}\mathbf{x}\}.$$

Clearly, the solution lies in this space. From this, we will show that $\mathbf{x} \in \mathcal{K}_m(\mathbf{A}, \mathbf{b})$, for a regular matrix $\mathbf{A}$ with m distinct eigenvalues. Krylov methods find the roots (eigenvalues) of the minimal polynomial of $\mathbf{A}$ within m iterations (see, e.g., [55] Section 2.4). Consequently, any Krylov subspace formed by $\mathbf{A}$ and $\mathbf{x}$ is limited to a dimension of at most m. We have that $\mathcal{K}_t(\mathbf{A}, \mathbf{x}) = \mathcal{K}_m(\mathbf{A}, \mathbf{x}), \forall t > m$, i.e., $\mathcal{K}_m(\mathbf{A}, \mathbf{x})$ is *invariant* under multiplication by $\mathbf{A}$. Substituting $\mathbf{x}$ with $\mathbf{A}^{-1}\mathbf{b}$ gives

$$\text{span}\{\mathbf{x}, \mathbf{A}\mathbf{x}, \ldots, \mathbf{A}^{m-1}\mathbf{x}\} = \text{span}\{\mathbf{A}^{-1}\mathbf{b}, \mathbf{b}, \ldots, \mathbf{A}^{m-2}\mathbf{b}\}.$$

Subsequent application of $\mathbf{A}$ does not change the span of this space due to the invariance property. Thus, we have

$$\text{span}\{\mathbf{A}^{-1}\mathbf{b}, \mathbf{b}, \ldots, \mathbf{A}^{m-2}\mathbf{b}\} = \text{span}\{\mathbf{b}, \mathbf{A}\mathbf{b}, \ldots, \mathbf{A}^{m-1}\mathbf{b}\} = \mathcal{K}_m(\mathbf{A}, \mathbf{b}),$$

which implies that $\mathbf{x} \in \mathcal{K}_m(\mathbf{A}, \mathbf{b})$. Let $\mathbf{x}$ be written as a linear combination of a given basis for $\mathcal{K}_m(\mathbf{A}, \mathbf{b})$. Since

$$[\mathbf{b}, \mathbf{A}\mathbf{b}, \ldots, \mathbf{A}^{m-1}\mathbf{b}] \tag{2.3}$$

is a basis for $\mathcal{K}_m(\mathbf{A}, \mathbf{b})$, there exists a $\mathbf{y} = [y_1, y_2, \ldots, y_m]^{\mathrm{T}}$, such that

$$\mathbf{x} = [\mathbf{b}, \mathbf{A}\mathbf{b}, \ldots, \mathbf{A}^{m-1}\mathbf{b}]\mathbf{y} = (y_1\mathbf{I} + y_2\mathbf{A} + \cdots + y_m\mathbf{A}^{m-1})\mathbf{b}. \tag{2.4}$$

In general, (2.4) can be expressed as $\mathbf{x} = q_{m-1}(\mathbf{A})\mathbf{b} = \mathbf{A}^{-1}\mathbf{b}$, where $q_{m-1}(\mathbf{A})$ is a certain matrix polynomial of degree $m-1$. The basis in (2.3), commonly referred to as the *monomial* basis, exhibits poor conditioning, leading to issues in numerical computations. Therefore, linear system solvers like GMRES deliberately choose $q_{m-1}(\mathbf{A})$ to yield a well-conditioned basis for $\mathcal{K}_m(\mathbf{A}, \mathbf{b})$. Consequently, after i iterations, with $i \leq m$ the approximation $\tilde{\mathbf{x}}$ of $\mathbf{x}$ is given by $\tilde{\mathbf{x}} = q_{i-1}(\mathbf{A})\mathbf{b} \approx \mathbf{A}^{-1}\mathbf{b}$. Including an initial guess $\mathbf{x}^{(0)}$ extends this formulation to:

$$\tilde{\mathbf{x}} = \mathbf{x}^{(0)} + q_{i-1}(\mathbf{A})\mathbf{r}^{(0)} \approx \mathbf{A}^{-1}\mathbf{b}, \tag{2.5}$$

with $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ [61]. The closer $\mathbf{x}^{(0)}$ is to $\mathbf{x}$, the closer $\mathbf{r}^{(0)}$ will be to the zero vector. A quick derivation for (2.5) shows that if $i = m$, then

$$\begin{aligned}\mathbf{x} &= \mathbf{x}^{(0)} + q_{m-1}(\mathbf{A})\mathbf{r}^{(0)} \\ &= \mathbf{x}^{(0)} - q_{m-1}(\mathbf{A})\mathbf{A}\mathbf{x}^{(0)} + q_{m-1}(\mathbf{A})\mathbf{b} \\ &= \mathbf{A}^{-1}\mathbf{b}.\end{aligned}$$

Lastly, (2.5) implies the residual relation for the i^{th} residual update

$$\mathbf{r}^{(i)} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}} = \mathbf{r}^{(0)} - \mathbf{A}q_{i-1}(\mathbf{A})\mathbf{r}^{(0)}. \tag{2.6}$$

2.3 Introduction to GMRES

The Generalized Minimal Residual Method (GMRES) is a Krylov subspace method for solving large sparse (possibly nonsymmetric) linear systems, first introduced by Saad et al. [61]. GMRES monotonically decreases the norm of the residual by iteratively updating the initial guess $\mathbf{x}^{(0)}$ with a correction term $q_{i-1}(\mathbf{A})\mathbf{r}^{(0)}$. More specifically, GMRES formulates $q_{i-1}(\mathbf{A})\mathbf{r}^{(0)}$ in (2.5) as $\mathbf{Q}_{1:i}\mathbf{y}^{(i-1)}$, where $\mathbf{Q} \in \mathbb{R}^{n\times(m+1)}$, and $\mathbf{y}^{(i-1)}$ is a vector of length i, such that the approximate solution is computed as $\tilde{\mathbf{x}} = \mathbf{x}^{(0)} + \mathbf{Q}_{1:i}\mathbf{y}^{(i-1)}$. $\mathbf{Q}_{1:i}$ is an orthonormal basis for the i^{th} Krylov subspace

$$\mathcal{K}_i := \text{span}\{\mathbf{r}^{(0)}, \mathbf{A}\mathbf{r}^{(0)}, \ldots, \mathbf{A}^{i-1}\mathbf{r}^{(0)}\}.$$

$\mathbf{y}^{(i-1)}$ is the solution to the least-squares problem

$$\mathbf{y}^{(i-1)} = \underset{\mathbf{y}}{\text{argmin}} \, \|\mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} + \mathbf{Q}_{1:i}\mathbf{y})\|, \tag{2.7}$$

yielding a linear combination of $\mathbf{Q}_{1:i}$ such that the norm of the i^{th} residual in (2.6) is minimized, with $\tilde{\mathbf{x}}$ as the closest approximate solution in $\mathcal{K}_i$ to $\mathbf{x}$. Solving the least-squares problem is computationally expensive. However, reformulating the problem in (2.7) enables efficient computation.

2.3.1 An easier least-squares problem

In constructing an orthonormal basis for $\mathcal{K}_i$, GMRES ensures that $\mathbf{Q}_{1:i}$ is part of the Arnoldi relation

$$\mathbf{A}\mathbf{Q}_{1:i} = \mathbf{Q}_{1:i+1}\mathbf{H}, \tag{2.8}$$

where $\mathbf{H}$ is a $i+1$ by i upper Hessenberg coefficient matrix. Arnoldi's method [2] (see Algorithm 2) builds (2.8) using an orthogonalization process. To emphasize this orthogonalization process, (2.8) is extended and reinterpreted as a QR factorization

$$[\mathbf{r}^{(0)}, \mathbf{A}\mathbf{Q}_{1:i}] = \mathbf{Q}_{1:i+1}\mathbf{R},$$

where the R factor has entries $r_{1,1} = \|\mathbf{r}^{(0)}\|$ and $\mathbf{R}_{1:i+1,2:i+1} = \mathbf{H}$. Using (2.8) and the fact that $\mathbf{q}_1 = \mathbf{r}^{(0)}/\|\mathbf{r}^{(0)}\|$, (2.7) is expressed as a $(i+1)\times i$ dimensional least-squares problem:

$$\underset{\mathbf{y}}{\text{argmin}} \, \|\mathbf{r}^{(0)} - \mathbf{A}\mathbf{Q}_{1:i}\mathbf{y}\| = \underset{\mathbf{y}}{\text{argmin}} \, \|\mathbf{r}^{(0)} - \mathbf{Q}_{1:i+1}\mathbf{H}\mathbf{y}\| =$$
$$\underset{\mathbf{y}}{\text{argmin}} \, \|\mathbf{Q}_{1:i+1}^{\mathrm{T}}\mathbf{r}^{(0)} - \mathbf{H}\mathbf{y}\| = \underset{\mathbf{y}}{\text{argmin}} \, \|\|\mathbf{r}^{(0)}\|\mathbf{e}_1 - \mathbf{H}\mathbf{y}\|.$$

Solving the smaller least-squares problem requires only i Givens rotations applied to $\mathbf{H}$ and $\|\mathbf{r}^{(0)}\|\mathbf{e}_1$. Let $\mathbf{z}^{(i)}$ be the vector of length $i+1$ after i Givens rotations applied to $\|\mathbf{r}^{(0)}\|\mathbf{e}_1$. The i^{th} implicit residual norm is revealed in the absolute value of the $i+1^{st}$ component of $\mathbf{z}^{(i)}$. In exact arithmetic, residuals do not need to be updated explicitly, as the quality of the approximate solution can be efficiently checked using the implicit residual norm. In finite precision, the explicit residual should be checked from time to time, as it may differ from the implicit residual due to rounding errors.

Algorithm 2 Arnoldi's method

Input: $\mathbf{A} \in \mathbb{R}^{n \times n}$, normalized start vector $\mathbf{q}_1 \in \mathbb{R}^n$
Output: Orthonormal basis $\mathbf{Q} \in \mathbb{R}^{n \times (m+1)}$, $\mathbf{H} \in \mathbb{R}^{(m+1) \times m}$, such that $\mathbf{AQ}_{1:m} = \mathbf{QH}$

1: **for** $i = 1 : m$ **do**
2: $\quad \mathbf{v}_i = \mathbf{A}\mathbf{q}_i$
3: $\quad \mathbf{q}'_i = \mathbf{\Pi}^{(i-1)}_{MGS}\mathbf{v}_i$, obtain projection coefficients $\mathbf{h}_{1:i,i}$
4: $\quad h_{i+1,i} = ||\mathbf{q}'_i||$
5: $\quad$ **if** $h_{i+1,i} = 0$ **then** stop ▷ The minimal polynomial w.r.t. $\mathbf{q}_1$ is of degree i.
6: $\quad \mathbf{q}_{i+1} = \mathbf{q}'_i / h_{i+1,i}$
7: **end for**

2.3.2 Monotonically decreasing residual norms

GMRES performs an oblique projection of the residual, meaning that in iteration i, an approximation $\mathbf{x}^{(i)}$ is found via a linear combination of the columns of $\mathbf{Q}_{1:i} \in \mathcal{K}_i$, such that $\mathbf{r}^{(i)} \in \mathcal{K}_{i+1}$ is orthogonal to the subspace $\mathcal{L}_i := \mathbf{A}\mathcal{K}_i$ (and thus oblique to $\mathcal{K}_i$)[1]. Figures 2.1 and 2.2 illustrate the projection of $\mathbf{r}^{(0)}$ onto $\mathcal{L}_i$, for $i = 1, 2$, and demonstrate how residual norms decrease due to properties of orthogonal triangles. Thus, new residual norms will be at most as large as the previous residual norms.

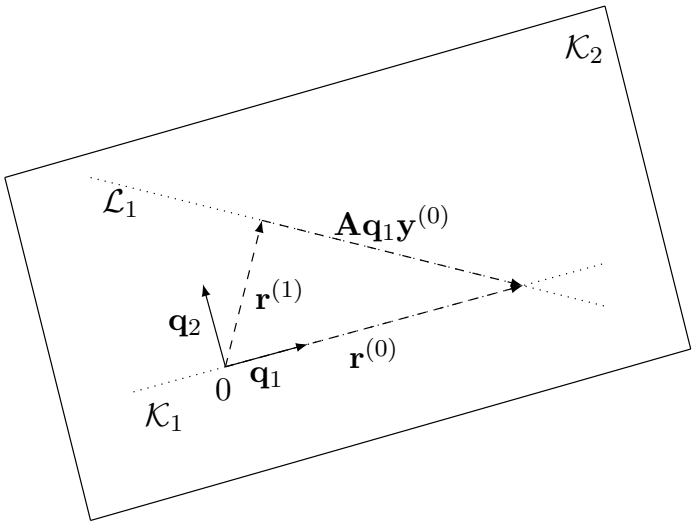

FIGURE 2.1: Illustration of GMRES minimizing the residual norm after one iteration in the $\mathcal{K}_2$ plane, spanned by one-dimensional subspaces $\mathcal{K}_1$ and $\mathcal{L}_1$. The process involves projecting $\mathbf{r}^{(0)}$ onto $\mathcal{L}_1$ and determining $\mathbf{y}^{(0)}$ such that $\mathbf{r}^{(1)} = \mathbf{r}^{(0)} - \mathbf{Aq}_1\mathbf{y}^{(0)}$ is orthogonal to $\mathcal{L}_1$. The orthonormal basis $\mathbf{Q}_{1:2} = [\mathbf{q}_1, \mathbf{q}_2] \in \mathcal{K}_2$ can be used to compute $\mathbf{r}^{(2)} \perp \mathcal{L}_2$ (see Fig. 2.2).

[1] There is *orthogonal* and *oblique* projection. In orthogonal projection, the solution subspace $\mathcal{K}_i$ is equivalent to the subspace $\mathcal{L}_i$ associated with the orthogonality constraint on $\mathbf{r}^{(i)}$.

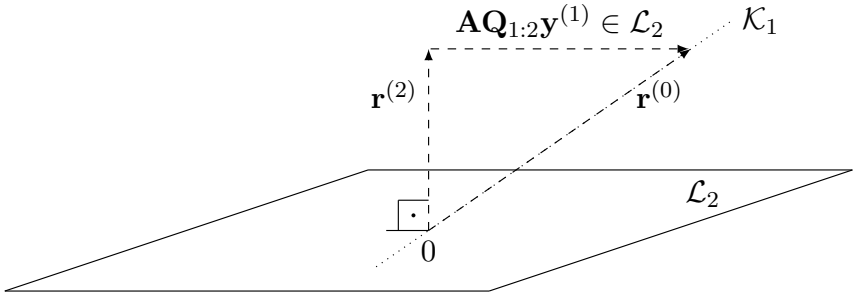

FIGURE 2.2: Illustration of GMRES minimizing the residual norm after two iterations in three-dimensional subspace $\mathcal{K}_3$, spanned by $\mathcal{L}_2 := \mathbf{A}\mathcal{K}_2$ and $\mathcal{K}_1$. $\mathbf{r}^{(0)}$ is projected onto $\mathcal{L}_2$, by finding a linear combination $\mathbf{Q}_{1:2}\mathbf{y}^{(1)} \in \mathcal{K}_2$ such that $\mathbf{r}^{(2)} = \mathbf{r}^{(0)} - \mathbf{A}\mathbf{Q}_{1:2}\mathbf{y}^{(1)} \perp \mathcal{L}_2$.

2.3.3 Convergence, preconditioning, and restarting

If $\mathbf{S} := (\mathbf{A} + \mathbf{A}^{\mathrm{T}})/2$ is positive definite, then we have

$$\|\mathbf{r}^{(i)}\| \leq \left(1 - \frac{\lambda_{\min}^2(\mathbf{S})}{\lambda_{\max}(\mathbf{A}^{\mathrm{T}}\mathbf{A})}\right)^{i/2} \|\mathbf{r}^{(0)}\| \text{ [31].}$$

For general $n \times n$ matrices $\mathbf{A}$, it can be shown that the residual norm can remain unchanged for $n-1$ iterations before suddenly decreasing to zero on the n^{th} iteration [39]. However, GMRES tends to work well in practice, especially when preconditioners are used to accelerate convergence.

Ideally, a preconditioner $\mathbf{M} \in \mathbb{R}^{n\times n}$ improves the condition number of $\mathbf{A}$ so that $\kappa(\mathbf{M}^{-1}\mathbf{A}) \approx 1$, and $\mathbf{M}^{-1}$ is inexpensively applied. Typically, there is a trade-off between the effectiveness of the preconditioner and its computation and application cost. Left preconditioning applies $\mathbf{M}^{-1}$ to the left of the system, such that $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$. Right preconditioning modifies the system to $\mathbf{A}\mathbf{M}^{-1}(\mathbf{M}\mathbf{x}) = \mathbf{b}$. Although both techniques yield similar performance in practice [62], left preconditioning has a potential drawback. In iteration i, left-preconditioned GMRES minimizes the preconditioned residual norm $\|\mathbf{M}^{-1}\mathbf{r}^{(i)}\|$. If $\mathbf{M}^{-1}$ is poorly conditioned, $\|\mathbf{M}^{-1}\mathbf{r}^{(i)}\|$ can deviate significantly from the unpreconditioned residual norm $\|\mathbf{r}^{(i)}\|$ [62]. In contrast, the unpreconditioned residual is readily available in right-preconditioned GMRES, making it the preferred choice in practice.

The memory requirements of GMRES increase linearly with the number of iterations, while the computational cost grows quadratically. Consequently, GMRES is frequently restarted after a certain number of iterations, using the current approximate solution as the next initial guess. Restarting may lead to the loss of valuable information in the basis, which can potentially cause stagnation in convergence[2]. Selecting an appropriate restart length involves considering the convergence rate, computational cost, and memory constraints [45].

Algorithm 3 shows the restarted GMRES method, GMRES(m), with right preconditioning, commonly applied in practice. To ensure backward stability, the projector in line 6 of Algorithm 3 must at least possess the stability properties of MGS. Note that indices of vector updates correspond to the number of restarts rather than the number of iterations. For example, the algorithm performs m iterations between $\mathbf{r}^{(k)}$ and $\mathbf{r}^{(k+1)}$, where k denotes the number of restarts.

[2] Although this behavior is frequently observed in practice, it is not generally true [32].

Algorithm 3 GMRES(m) with right preconditioning

Input: $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$, preconditioner $\mathbf{M}$, initial guess $\mathbf{x}^{(0)}$
Output: Approximate solution to $\mathbf{Ax} = \mathbf{b}$

1: **for** $k = 0, 1, \ldots,$ until convergence **do**
2: $\quad \mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$
3: $\quad \mathbf{q}_1 = \mathbf{r}^{(k)} / \|\mathbf{r}^{(k)}\|$
4: $\quad$ **for** $i = 1 : m$ **do**
5: $\quad\quad \mathbf{v}_i = \mathbf{A}\mathbf{M}^{-1}\mathbf{q}_i$
6: $\quad\quad \mathbf{q}_i' = \mathbf{\Pi}^{(i-1)}\mathbf{v}_i$, obtain projection coefficients $\mathbf{h}_{1:i,i}$
7: $\quad\quad h_{i+1,i} = ||\mathbf{q}_i'||$
8: $\quad\quad$ **if** converged **then break**
9: $\quad\quad \mathbf{q}_{i+1} = \mathbf{q}_i' / h_{i+1,i}$
10: $\quad$ **end for**
11: $\quad \mathbf{y}^{(k)} = \operatorname{argmin}_{\mathbf{y}} \|\|\mathbf{r}^{(k)}\|\mathbf{e}_1 - \mathbf{H}_{1:i+1,1:i}\mathbf{y}\|$
12: $\quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{M}^{-1}\mathbf{Q}_{1:i}\mathbf{y}^{(k)}$
13: **end for**

2.3.4 Parallel GMRES

Here, we discuss aspects of parallel GMRES algorithms. The sparse system matrix $\mathbf{A}$ is typically partitioned among processes using a block-row distribution. This distribution assigns blocks of matrix rows to different processes, with each process owning a block of contiguous rows. High-dimensional vectors are partitioned into segments of roughly equal length along their large dimension. For example, on a parallel system with p processes, $\mathbf{X} \in \mathbb{R}^{n \times m}$, with $m \ll n$ is partitioned such that

$$\mathbf{X} = [\mathbf{X}^{\{1\}} | \mathbf{X}^{\{2\}} | \ldots | \mathbf{X}^{\{p\}}],$$

where $\mathbf{X}^{\{j\}} \in \mathbb{R}^{n_j \times m}$ belongs to process j, and $n = \sum_{j=1}^{p} n_j$.
We discuss two communication patterns relevant for GMRES: *point-to-point* communication and *global synchronization*. Out of the two, point-to-point communication is a relatively inexpensive operation that scales well, as each process communicates with only a few other processes. This type of communication occurs, for example, in lines 2, 5, and 12 of Algorithm 3, assuming that multiplying $\mathbf{A}$ and $\mathbf{M}^{-1}$ with a vector requires some communication. A global synchronization is a parallel reduction operation involving all p processes that requires $\mathcal{O}(\log p)$ time. Compared to point-to-point communication, this operation poses a significant bottleneck at scale. In GMRES, reduction operations are primarily required by the underlying orthogonalization method. For example, the projector in line 6 of Algorithm 3 requires $i - 1$ global synchronizations for MGS and one global synchronization for CGS. The normalization step in line 7 requires an additional global synchronization.
Although CGS2 requires twice the work of MGS, it is a better choice in parallel computing due to its superior scalability, as it requires only two global synchronizations for two projection steps instead of $i-1$ for one projection step. Moreover, CGS2 offers better stability than MGS (cf. Table 2.1).
Typically, the reduce operations in the orthogonalization step are implemented as *allreduce* operations. This ensures that the Hessenberg matrix, the result of the operation, is available on all processes so that the least-squares problem in line 11 of Algorithm 3 can be solved in parallel without further communication.

Chapter 3

s-step GMRES methods

Unlike standard (single-step) GMRES methods, s-step GMRES methods advance s steps per iteration, potentially reducing communication by a factor of $\mathcal{O}(s)$ [45]. This chapter explores s-step GMRES methods, focusing on two of their key components: the matrix powers kernel in Section 3.1, and block orthogonalization in Section 3.2. Section 3.3 investigates adaptive s-step methods designed to address potential sources of instability that arise from advancing s steps instead of just one.

3.1 Matrix Powers Kernel (MPK)

Most of Section 3.1 is from [45]. The *matrix powers kernel* (MPK) generates s new vectors for the Krylov basis along with a corresponding change-of-basis matrix. It requires a sparse matrix $\mathbf{A}$, a start vector $\mathbf{v}_1$, and a sequence of polynomials $\{p_j(\mathbf{A})\}_{j=0}^{s}$, defined by the recurrence relation

$$p_{i+1}(\mathbf{A}) = (\alpha_i \mathbf{A} - \beta_i \mathbf{I}) p_i(\mathbf{A})/\gamma_i, \tag{3.1}$$

for $i = 1, \ldots, s$, $\alpha_i \neq 0$, 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

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

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 design of $\mathbf{B}$ depends on how the parameters in (3.1) are chosen. For details on constructing $\mathbf{B}$ with different basis types, see [45].
In the following, we will discuss the monomial basis, the Newton basis [45], and the scaled Newton basis introduced in [65]. For information on the Chebyshev basis, another popular basis type, refer to [45].

3.1.1 Monomial basis

The monomial basis corresponds to setting $\alpha_i = \gamma_i = 1$ and $\beta_i = 0$ in (3.1) for all i, resulting in

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

The monomial basis can become ill-conditioned quickly, as its vectors converge toward the largest eigenvector of the input matrix, often restricting the step size s to very small values. For example, $s = 5$ is commonly used in practice [45, 67].

3.1.2 Newton basis

The Newton basis incorporates the *Ritz values*, which approximate the eigenvalues of $\mathbf{A}$ [45]. Constructing an $n \times (s+1)$ Newton basis requires at least s Ritz values. After s iterations of the Arnoldi or GMRES method, the Ritz values are obtained from the resulting $(s+1) \times s$ upper Hessenberg matrix $\mathbf{H}$, as the eigenvalues of $\mathbf{H}_{1:s,1:s}$. The Newton basis corresponds to (3.1) with $\alpha_i = \gamma_i = 1$ and $\beta_i = \theta_i$ for all i, resulting in

$$\mathbf{V}_{\text{Newton}} = [\mathbf{v}_1, (\mathbf{A} - \theta_1\mathbf{I})\mathbf{v}_1, (\mathbf{A} - \theta_2\mathbf{I})(\mathbf{A} - \theta_1\mathbf{I})\mathbf{v}_1, \ldots, \prod_{i=1}^{s}(\mathbf{A} - \theta_i\mathbf{I})\mathbf{v}_1],$$

where $\theta_1, \theta_2, \ldots, \theta_s$ are the Ritz values of $\mathbf{A}$. Sorting the Ritz values according to the *Leja ordering* improves the condition of the Newton basis. The Leja ordering determines the first Ritz value by maximum absolute value. Subsequent Ritz values are chosen to maximize the distance from the previously sorted values. The *modified Leja ordering* [4], outlined in Algorithm 4, avoids complex arithmetic during Newton basis computation (see [45] for details).

Algorithm 4 Modified Leja ordering

Input: Ritz value set $\Lambda = \{\lambda_1, \ldots, \lambda_s\}$ arranged such that complex conjugate pairs appear consecutively (value with positive imaginary part first)
Output: Ordered Ritz values $\theta_1, \ldots, \theta_s$

1: $\theta_1 = \underset{\lambda \in \Lambda \wedge \Im(\lambda) \geq 0}{\operatorname{argmax}} |\lambda|$
2: **for** $i = 1 : s-1$ **do**
3: $\quad \Lambda = \Lambda \setminus \{\theta_i\}$ ▷ Remove θ_i from the Ritz value set.
4: $\quad$ **if** $\Im(\theta_i) > 0$ **then**
5: $\quad\quad \theta_{i+1} = \bar{\theta}_i$ ▷ Include the conjugate of θ_i.
6: $\quad$ **else**
7: $\quad\quad \theta_{i+1} = \underset{\lambda \in \Lambda \wedge \Im(\lambda) \geq 0}{\operatorname{argmax}} \prod_{j=1}^{i} |\lambda - \theta_j|$ ▷ Determine the next Ritz value.
8: $\quad$ **end if**
9: **end for**

3.1.3 Scaled Newton basis

The scaled Newton basis [65] is obtained with $\alpha_i = 1$, $\beta_i = \theta_i$, and $\gamma_i = |\bar{\theta} - \theta_i|$ for all i in (3.1), where $\bar{\theta}$ represents the mean of the computed Ritz values. As more Ritz values are computed, $\bar{\theta}$ approaches the mean eigenvalue spectrum of $\mathbf{A}$. This strategy, suggested in [65], aims to maintain the basis vector length at approximately $\mathcal{O}(1)$. However, the authors do not address how to manage zero or nearly zero values for γ_i, which may arise for Ritz values close to $\bar{\theta}$.

3.1.4 Communication-avoiding MPK

A straightforward implementation of the MPK invokes s successive matrix-vector operations using the recurrence (3.1). Hoemmen et al. [45] present sequential and parallel communication-avoiding MPK (CA-MPK) algorithms. These methods reduce the number of messages sent by a factor of $\mathcal{O}(s)$ through modifications to the matrix-vector distribution and communication patterns, at the cost of some redundant computation. This approach has two main drawbacks.

1) Limitations of the preconditioner choice Preconditioners are essential for iterative linear system solvers. For example, solving a left preconditioned system $\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$ using an s-step method changes the Krylov basis vectors generated by the MPK to

$$[p_1(\mathbf{M}^{-1}\mathbf{A})\mathbf{v}_1, p_2(\mathbf{M}^{-1}\mathbf{A})\mathbf{v}_1, \ldots, p_s(\mathbf{M}^{-1}\mathbf{A})\mathbf{v}_1],$$

which implies the need for preconditioner application between each matrix-vector operation involving $\mathbf{A}$ [45]. Preconditioners that require communication must be adapted to preserve the communication-avoiding property of the preconditioned CA-MPK (e.g., see [40] for a communication-avoiding ILU(0) preconditioner). However, adapting many sophisticated and widely used preconditioners, such as multigrid preconditioners, remains an active area of research.

2) Challenging performance improvements Improving algorithm performance is a primary objective, but it is important to assess whether the effort justifies the achievable gains. The MPK mainly consists of sparse matrix-vector operations. In parallel, these operations require point-to-point communication between processors, which, at scale, do not pose severe communication bottlenecks compared to global synchronizations. Even with a reduced number of messages in point-to-point communication, a naïve implementation of the CA-MPK may perform poorly because operations involving sparse matrices are bound by memory rather than computation. Hoemmen et al. [45] use optimizations to improve performance over the straightforward MPK approach, including branch elimination in loops to maximize instruction pipeline throughput, special hardware instructions for parallel computations of adjacent floating-point values (SIMD), register tiling, and the use of shorter integers for indices in cache blocks. Some of these optimizations are quite sophisticated[1], and recent advances in [1] have significantly improved the performance of straightforward MPK methods using matrix caching techniques.

3.1.5 Equilibration

Equilibrating a matrix $\mathbf{A}$ scales its spectral radius to approximately unit length and alters its relative eigenvalue distribution. This prevents significant changes in vector length during Krylov basis generation in the MPK, increasing stability and enabling larger step sizes for s-step GMRES methods. Equilibration is a form of preconditioning, which scales the linear system using two diagonal matrices, $\mathbf{D}_r$ and $\mathbf{D}_c$, such that

$$\mathbf{D}_r\mathbf{A}\mathbf{D}_c(\mathbf{D}_c^{-1}\mathbf{x}) = \mathbf{D}_r\mathbf{b}.$$

Row equilibration sets $\mathbf{D}_c = \mathbf{I}$, corresponding to left preconditioning, where GMRES minimizes the preconditioned residual norm $\tilde{r} = \|\mathbf{D}_r\mathbf{r}\|$, for some residual $\mathbf{r}$. If $\mathbf{D}_r$ is ill-conditioned, row scaling should be avoided, as $\tilde{r}$ may differ significantly from the unpreconditioned residual norm (see Section 2.3.3). Column equilibration sets $\mathbf{D}_r = \mathbf{I}$, corresponding to right preconditioning, where GMRES minimizes the unpreconditioned residual norm.

[1] We implemented a CA-MPK variant (PA1, Algorithm 5 in [45]) incorporating several optimizations. However, the performance did not surpass that of the straightforward MPK approach for general sparse matrices.

3.2 Block Gram-Schmidt process

Block orthogonalization methods outperform unblocked methods by reducing communication overhead in both parallel and sequential operations. They orthogonalize a set (a block) of vectors instead of a single vector and replace BLAS-2 with cache-friendly BLAS-3 operations. Block variants of orthogonal triangularization, such as the block HH-QR factorization (BHH-QR) (see [38], Section 5.2.3), offer unconditional stability. However, these methods are considered too expensive in terms of computation and communication when compared to other algorithms like block Gram-Schmidt (BGS). Moreover, orthogonalization methods cannot resolve issues with rank-deficient or ill-conditioned Krylov bases generated by the MPK, as linearly dependent or nearly dependent vectors do not have sufficient information about the solution space [65]. These issues must be addressed separately from the orthogonalization method.
In practice, some BGS methods appear to provide sufficient stability for s-step GMRES algorithms [65]. The current state-of-the-art primarily focuses on block classical Gram-Schmidt (BCGS) methods, which are highly parallelizable, unlike block modified Gram-Schmidt variants [45]. Specifically, BCGS with inner reorthogonalization (BCGSI2) [11] has gained attention in recent years. A low-synchronization variant (BCGSI2-LS) was presented in [20]. Carson et al. showed in [18] that BCGSI2-LS is less stable than BCGSI2. For a stability comparison of these block orthogonalization methods, see Table 3.1.

TABLE 3.1: Bounds on the "loss of orthogonality" in the orthonormal basis $\mathbf{Q}$ for an input matrix $\mathbf{V} \in \mathbb{R}^{n\times sp} = [\mathbf{V}_1, \ldots, \mathbf{V}_p]$, for various block orthogonalization schemes, assuming *unconditionally stable* intra-orthogonalization of the column blocks $\mathbf{V}_i \in \mathbb{R}^{n\times s}$, $1 \le i \le p$.

Algorithm	$\|\mathbf{I} - \mathbf{Q}^T\mathbf{Q}\|$ bound	Assumption on $\kappa(\mathbf{V})$	Reference
BCGS	$\mathcal{O}(u)\,\kappa^{m-1}(\mathbf{V})$	$\mathcal{O}(u)\,\kappa(\mathbf{V}) < 1$	[18]
BCGSI2	$\mathcal{O}(u)$	$\mathcal{O}(u)\,\kappa(\mathbf{V}) < 1$	[11]
BCGSI2-LS	$\mathcal{O}(u)\,\kappa^2(\mathbf{V})$	$\mathcal{O}(u)\,\kappa^3(\mathbf{V}) < 1$	[18]
BHH-QR	$\mathcal{O}(u)$	none	[44]

3.2.1 Block Classical Gram-Schmidt (BCGS)

BCGS employs the skeleton-muscle scheme, a term coined by Hoemmen et al. in [45]. This scheme ensures orthogonality through both *inter*-orthogonalization (against a previously orthogonalized set of vectors, the *skeleton*) and *intra*-orthogonalization (within a set of vectors, the *muscle*). Different combinations of skeletons and muscles allow for balancing the trade-off between communication reduction and orthogonalization quality. The methods listed in Table 2.1 can serve as muscles, although only TSQR and CholQR are communication-avoiding methods, and both CGS and CholQR entail very restrictive stability conditions. Algorithm 5 outlines the BCGS skeleton, with the projector $\mathbf{\Pi}^{(i)}_{BCGS} := (\mathbf{I} - \mathbf{Q}_{1:i}\mathbf{Q}^{\mathrm{T}}_{1:i})$ introduced in line 4.

3.2.2 BCGS with inner reorthogonalization (BCGSI2)

BCGSI2 [11] (see Algorithm 6) performs BCGS with inner reorthogonalization, applying both inter-orthogonalization and intra-orthogonalization twice in sequence. Two additional operations improve stability by combining the two orthogonalization steps with negligible computational cost (see lines 8 and 9 of Algorithm 6).

Algorithm 5 Block Classical Gram-Schmidt process

Input: $\mathbf{V} \in \mathbb{R}^{n\times m}$, step size s, $m = sp$
Output: Orthonormal $\mathbf{Q} \in \mathbb{R}^{n\times m}$, upper triangular $\mathbf{R} \in \mathbb{R}^{m\times m}$

for $j = 1 : p$ **do**
  $i = s(j-1)$
  $b = i+1 : i+s$
  $\mathbf{Q}'_b = \mathbf{\Pi}^{(i)}_{BCGS}\mathbf{V}_b$, obtain projection coefficients $\mathbf{R}_{1:i,b}$
  Compute thin QR factorization $\mathbf{Q}'_b = \mathbf{Q}_b\mathbf{R}_{b,b}$
end for

Algorithm 6 BCGSI2

Input: $\mathbf{V} \in \mathbb{R}^{n\times m}$, step size s, $m = sp$
Output: Orthonormal $\mathbf{Q} \in \mathbb{R}^{n\times m}$, upper triangular $\mathbf{R} \in \mathbb{R}^{m\times m}$

for $j = 1 : p$ **do**
  $i = s(j-1)$
  $b = i+1 : i+s$
  $\mathbf{Q}'_b = \mathbf{\Pi}^{(i)}_{BCGS}\mathbf{V}_b$, obtain projection coefficients $\mathbf{R}^{(1)}_{1:i,b}$ ▷ Inter-ortho I
  Compute thin QR factorization $\mathbf{Q}'_b = \mathbf{Q}^{(1)}_b\mathbf{R}^{(1)}_{b,b}$ ▷ Intra-ortho I
  $\mathbf{Q}^{(2)}_b = \mathbf{\Pi}^{(i)}_{BCGS}\mathbf{Q}^{(1)}_b$, obtain projection coefficients $\mathbf{R}^{(2)}_{1:i,b}$ ▷ Inter-ortho II
  Compute thin QR factorization $\mathbf{Q}^{(2)}_b = \mathbf{Q}_b\mathbf{R}^{(2)}_{b,b}$ ▷ Intra-ortho II
  $\mathbf{R}_{1:i,b} = \mathbf{R}^{(1)}_{1:i,b} + \mathbf{R}^{(2)}_{1:i,b}\mathbf{R}^{(1)}_{b,b}$ ▷ Combine both steps
  $\mathbf{R}_{b,b} = \mathbf{R}^{(2)}_{b,b}\mathbf{R}^{(1)}_{b,b}$
end for

3.3 Adaptive s-step GMRES

Section 3.3 discusses the adaptive s-step GMRES method introduced in [65], which we denote as BCGSI2-GMRES. The algorithm uses a block Arnoldi method derived from BCGSI2, similar to how GMRES uses an Arnoldi method derived from MGS (see Section 2.3.1). BCGSI2-GMRES identifies numerical instabilities in the Krylov basis vectors generated by the MPK. The algorithm adaptively reduces the step size to maintain the convergence behavior of single-step GMRES methods. This involves discarding previously generated basis vectors. Before we discuss the BCGSI2-GMRES algorithm in more detail, two methods are introduced: the incremental condition estimator, outlined in Section 3.3.1, which determines the number of vectors to discard, and the step size estimator, described in Section 3.3.2, which aims to prevent the generation of ill-conditioned basis vectors.

3.3.1 Incremental condition estimator

The incremental condition estimator (ICE) [13, 14] efficiently monitors the condition number of an upper triangular matrix as it is generated column by column. For instance, this happens during a thin QR factorization of an $n \times m$ matrix $\mathbf{V} = \mathbf{QR}$. Once the factorization process has computed the k^{th} column of $\mathbf{R}$, the ICE estimates $\kappa(\mathbf{R}_{1:k,1:k}) = \kappa(\mathbf{V}_{1:k})$ in $\mathcal{O}(k)$ time. In practice, this estimate is typically close to $\kappa(\mathbf{V}_{1:k})$, but it can be up to ten times smaller than the actual value [65]. The total cost of the ICE is $\mathcal{O}(m^2)$ which is relatively cheap compared to the singular value decomposition (SVD), which would require $\mathcal{O}(m^4)$ total time. For a detailed

algorithm description of the ICE, see [14]. Algorithm 7 integrates the ICE into the partial Cholesky factorization, which is used as part of the CholQR factorization described in Section 2.1.2. The ICE function in line 6 of Algorithm 7 takes as input an upper triangular matrix $\mathbf{R}_{1:j,1:j}$ and a condition number tolerance Ω and returns a scalar $j' \leq j$ such that $\mathbf{R}_{1:j',1:j'} \leq \Omega$. The partial Cholesky factorization terminates when the ICE returns a value smaller than the current size of $\mathbf{R}_{1:j,1:j}$ (see lines 6–10 of Algorithm 7).

Algorithm 7 Partial Cholesky factorization

Input: Gram matrix $\mathbf{G} \in \mathbb{R}^{m \times m}$, condition number tolerance Ω
Output: j, upper triangular $\mathbf{R} \in \mathbb{R}^{j \times j}$ with $\mathbf{G}_{1:j,1:j} = \mathbf{R}^{\mathrm{T}}\mathbf{R}$

1: **for** $j = 1 : m$ **do**
2: **for** $i = 1 : j - 1$ **do**
3: $r_{i,j} = (g_{i,j} - \mathbf{r}_{1:j-1,i}\mathbf{r}_{1:j-1,j})/r_{i,i}$
4: **end for**
5: $r_{j,j} = \sqrt{g_{j,j} - \|\mathbf{r}_{1:j-1,j}\|^2}$
6: $j' = \mathrm{ICE}(\mathbf{R}_{1:j,1:j}, \Omega)$
7: **if** $j' \neq j$ **then**
8: $j = j'$
9: **break**
10: **end if**
11: **end for**

3.3.2 Step size estimator

The step size estimator from [65] works with a scaled Newton basis $\tilde{\mathbf{V}} \in \mathbb{R}^{n \times (s+1)}$ (see Section 3.1.3). Its task is to estimate the optimal step size, denoted by

$$s_{\mathrm{opt}} = \underset{j}{\mathrm{argmax}}\{\kappa(\tilde{\mathbf{V}}_{1:j}) < \Omega\},$$

where Ω is equal to (or comparable in magnitude to) the condition number tolerance used in the ICE.
For simplicity, assume that $\mathbf{A}$ is diagonalizable, with the eigendecomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1}$, where $\mathbf{U}$ contains the eigenvectors of $\mathbf{A}$ and $\mathbf{\Lambda}$ is the corresponding (diagonal) eigenvalue matrix. Then $\tilde{\mathbf{V}}$ can be expressed as

$$\tilde{\mathbf{V}} = \mathbf{U}\mathbf{C}, \tag{3.2}$$

where $\mathbf{C} \in \mathbb{R}^{n \times (s+1)}$ describes the relation between the scaled Newton basis and the eigenspace of $\mathbf{A}$. Given the base case $\tilde{\mathbf{v}}_1 = \mathbf{U}\mathbf{c}_1$, $\mathbf{C}$ can be expressed as a recurrence relation $c_{i,j+1} = c_{i,j}\dfrac{\lambda_i - \theta_j}{|\bar{\theta} - \theta_j|}$, or equivalently:

$$c_{i,j} = c_{i,1}\prod_{k=1}^{j-1}\frac{\lambda_i - \theta_k}{|\bar{\theta} - \theta_k|}, \qquad 1 \leq i \leq n,\ 1 \leq j \leq s+1. \tag{3.3}$$

In (3.2), it holds that $\|\tilde{\mathbf{v}}_j\| = \|\mathbf{c}_j\|$, $1 \leq j \leq s+1$. Thus, vector growth in $\tilde{\mathbf{V}}$ is monitored by approximating the product term in (3.3) using a small $s \times s$ matrix $\mathbf{E}$,

whose elements are defined as follows:

$$e_{i,j} = \begin{cases} \prod_{k=1}^{j-1} \frac{|\theta_i - \theta_k|}{|\bar{\theta} - \theta_k|} & j < i \\ \left(\prod_{k=1}^{j-1} \frac{|\theta_i - \theta_k|}{|\bar{\theta} - \theta_k|} \right) \mathcal{O}(u) & j = i \\ \left(\prod_{k=1}^{i-1} \frac{|\theta_i - \theta_k|}{|\bar{\theta} - \theta_k|} \right) \mathcal{O}(u) \left(\prod_{k=i+1}^{j-1} \frac{|\theta_i - \theta_k|}{|\bar{\theta} - \theta_k|} \right) & i < j \leq s. \end{cases}$$

See [65] for details on deriving $\mathbf{E}$ from $\mathbf{C}$. The norm of the j^{th} column of $\mathbf{E}$ approximates the ratio $\|\mathbf{c}_j\|/\|\mathbf{c}_1\|$. Thus, the initial step size is estimated as $s_{\text{est}} = \text{argmax}_j\{\|\mathbf{e}_j\| < \Omega\}$. This estimation strategy detects an ill-conditioned basis $\tilde{\mathbf{V}}$ caused by poor vector scaling, but it does not account for nearly linearly dependent vectors. Consequently, s_{est} may be significantly larger than s_{opt}.

3.3.3 BCGSI2-GMRES

Algorithm 8 presents the BCGSI2-GMRES method. It is based on an adaptive version of BCGSI2 (Algorithm 6), using CholQR (see Section 2.1.2) with a partial Cholesky factorization (see Algorithm 7) for intra-orthogonalization. The first $s+1$ steps, outlined in lines 5–11 of Algorithm 8, process the initial vector together with s newly generated basis vectors. Therefore, the partial Cholesky factorization in line 6 operates on an $(s+1) \times (s+1)$ Gram matrix and returns a step size that accounts for both the first basis vector and the s additional vectors. To process the intended number of vectors in subsequent steps, step sizes $s > 1$ must be reduced by 1, as indicated in line 14 of Algorithm 8. Refer to [45] for details on the reconstruction of the Hessenberg matrix $\mathbf{H}$ in lines 11 and 29 of Algorithm 8.

Stability analysis

The stability guarantees in Table 3.1 do not apply to the BCGSI2 process with conditionally stable CholQR for intra-orthogonalization. Carson et al. show in [18] that if the first s vectors of $\mathbf{V}$ are orthogonalized to machine precision, BCGSI2 with CholQR achieves $\mathcal{O}(\epsilon)$ loss of orthogonality, assuming $\mathcal{O}(u)\,\kappa^2(\mathbf{V}) < 1/2$. The authors in [65] improve the stability of CholQR-based BCGSI2 in Algorithm 8 by dynamically reducing the step size using a partial Cholesky factorization with condition number tolerance $\Omega = \mathcal{O}\left(10^{-1}u^{-1/2}\right)$. This produces small enough step sizes so that the necessary stability conditions for the Cholesky factorization are satisfied (see Table 2.1). To orthogonalize the first s vectors to machine precision, CholQR2 is employed in lines 6–10 of Algorithm 8. The authors in [65] further argue that for very ill-conditioned input matrices, the step size may reduce to $s = 1$. In this case, the BCGSI2 method with CholQR reduces to the CGS2 process, which offers similar guarantees as BCGSI2 with unconditionally stable intra-orthogonalization (see Table 2.1). Since BCGSI2 assumes $s \geq 1$ in previous iterations, it is still an open question whether the adaptive BCGSI2 with CholQR is as stable as CGS2.

Performance analysis

In parallel computing, Algorithm 8 requires four global synchronizations every s steps (lines 19–20 and 23–24 of Algorithm 8), thus large step sizes are beneficial for reducing communication overhead. For some matrices, a scaled Newton basis allows for very large step sizes (see [65] for examples with $s = 500$). The Newton basis requires Ritz values, which are obtained by single-step methods such as Arnoldi or GMRES. Compared to s-step methods, single-step methods perform poorly. Therefore, if fewer

Ritz values are computed, a more efficient s-step method can be employed sooner. On the other hand, increasing the number of Ritz values improves the eigenvalue approximation quality. If approximation quality is important, it may be necessary to compute a Ritz value set larger than the step size used. Furthermore, Algorithm 8 only tests for convergence after every s steps, making larger step sizes costly if convergence occurs earlier. When a linear system supports large step sizes, one must balance the cost of computing Ritz values, the reduction of communication cost, and the expense of detecting convergence later than necessary.

Algorithm 8 BCGSI2-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, condition number tolerance Ω

Output: Approximate solution to $\mathbf{Ax} = \mathbf{b}$

1: **for** $k = 0, 1, \ldots,$ until convergence **do**
2: $\quad \mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$
3: $\quad \mathbf{v}_1 = \mathbf{r}^{(k)}$
4: $\quad \mathbf{q}_1 = \mathbf{v}_1 / \|\mathbf{r}^{(k)}\|$
5: $\quad [\mathbf{V}_{2:s+1},\, \mathbf{B}] = \texttt{MPK}(\mathbf{A}, \mathbf{q}_1, s)$
6: $\quad [s,\, \mathbf{Z}] = \texttt{PartialCholesky}\ ([\mathbf{q}_1, \mathbf{V}_{2:s+1}]^{\mathrm{T}}[\mathbf{q}_1, \mathbf{V}_{2:s+1}], \Omega)$
7: $\quad \mathbf{Q}'_{1:s} = [\mathbf{q}_1, \mathbf{V}_{2:s}]\mathbf{Z}^{-1}$ via triangular solve
8: $\quad [s,\, \mathbf{Z}'] = \texttt{PartialCholesky}\ (\mathbf{Q}'^{\mathrm{T}}_{1:s}\mathbf{Q}'_{1:s}, \Omega)$
9: $\quad \mathbf{Q}_{1:s} = \mathbf{Q}'_{1:s}\mathbf{Z}'^{-1}$ via triangular solve
10: $\quad \mathbf{R}_{1:s,1:s} = \mathbf{Z}'\mathbf{Z}_{1:s,1:s}$
11: $\quad$ Reconstruct $\mathbf{H}_{1:s,1:s-1}$ ▷ Use $\mathbf{R}_{1:s,1:s}$ and $\mathbf{B}$
12: $\quad i = s$
13: $\quad$ **if** converged **then** go to line 33
14: $\quad s = \max(1, s-1)$
15: $\quad$ **while** $i \leq m$ **do**
16: $\quad\quad$ **if** $i + s \geq m$ **then** $s = m + 1 - i$
17: $\quad\quad b = i + 1 : i + s$
18: $\quad\quad [\mathbf{V}_b, \mathbf{B}] = \texttt{MPK}(\mathbf{A}, \mathbf{q}_i, s)$
19: $\quad\quad \mathbf{V}'_b = \mathbf{\Pi}^{(i)}_{BCGS}\mathbf{V}_b$, obtain projection coefficients $\mathbf{R}^{(1)}_{1:i,b}$ ▷ Global sync I
20: $\quad\quad [s,\, \mathbf{Z}] = \texttt{PartialCholesky}\ (\mathbf{V}'^{\mathrm{T}}_b\mathbf{V}'_b, \Omega)$ ▷ Global sync II
21: $\quad\quad b = i + 1 : i + s$
22: $\quad\quad \mathbf{Q}'_b = \mathbf{V}'_b\mathbf{Z}^{-1}$ via triangular solve
23: $\quad\quad \tilde{\mathbf{Q}}_b = \mathbf{\Pi}^{(i)}_{BCGS}\mathbf{Q}'_b$, obtain projection coefficients $\mathbf{R}^{(2)}_{1:i,b}$ ▷ Global sync III
24: $\quad\quad [s,\, \mathbf{Z}'] = \texttt{PartialCholesky}\ (\tilde{\mathbf{Q}}^{\mathrm{T}}_b\tilde{\mathbf{Q}}_b, \Omega)$ ▷ Global sync IV
25: $\quad\quad b = i + 1 : i + s$
26: $\quad\quad \mathbf{Q}_b = \tilde{\mathbf{Q}}_b\mathbf{Z}'^{-1}$ via triangular solve
27: $\quad\quad \mathbf{R}_{1:i,b} = \mathbf{R}^{(1)}_{1:i,b} + \mathbf{R}^{(2)}_{1:i,b}\mathbf{Z}_{1:s,1:s}$
28: $\quad\quad \mathbf{R}_{b,b} = \mathbf{Z}'\mathbf{Z}_{1:s,1:s}$
29: $\quad\quad$ Reconstruct $\mathbf{H}_{1:i+s,b}$
30: $\quad\quad i = i + s$
31: $\quad\quad$ **if** converged **then break**
32: $\quad$ **end while**
33: $\quad \mathbf{y}^{(k)} = \operatorname{argmin}_{\mathbf{y}} \|\|\mathbf{r}^{(k)}\|\mathbf{e}_1 - \mathbf{H}_{1:i,1:i-1}\mathbf{y}\|$
34: $\quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{Q}_{1:i-1}\mathbf{y}^{(k)}$
35: **end for**

Chapter 4

Randomization in GMRES methods

This chapter discusses randomized GMRES methods. Starting with the basics of randomized sketching techniques in Section 4.1, we investigate two approaches to utilize randomization in the GMRES algorithm: *randomized* GMRES in Section 4.2, and *sketched* GMRES in Section 4.3. These methods form the basis of two randomized s-step GMRES algorithms introduced in Chapter 5, one of which is our novel algorithm.

4.1 Randomized sketching

Randomized sketching maps a problem in a high-dimensional subspace onto a low-dimensional subspace while 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. A subspace embedding is used to approximate the inner products of high-dimensional vectors [9].

Definition 4.1.1. *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*

$$\forall \mathbf{x}, \mathbf{y} \in \mathbf{V} : |\mathbf{x}^{\mathrm{T}}\mathbf{y} - (\mathbf{\Theta}\mathbf{x})^{\mathrm{T}}\mathbf{\Theta}\mathbf{y}| \leq \epsilon \|\mathbf{x}\|\|\mathbf{y}\|. \tag{4.1}$$

We give the following definition for *oblivious* subspace embeddings (OSE), used to sketch unknown m-dimensional subspaces in $\mathbb{R}^n$.

Definition 4.1.2. $\mathbf{\Theta}$ *is said to be an* (ϵ, δ, m) *OSE, if*

$$\mathbb{P}(\mathbf{\Theta} \text{ is an } \epsilon\text{-embedding for } V) \geq 1 - \delta. \tag{4.2}$$

That is, without having prior knowledge of the subspace V, an OSE *simultaneously* approximates the norms of all vectors in V with some probability $1 - \delta$ [25]. OSEs are constructed using probability distributions over $\mathbf{\Theta}$. Unstructured sketch matrices are based on Gaussian or Rademacher distributions. More sophisticated methods construct embeddings from structured matrices with faster application times. Examples of such sketches include the hierarchical Subsampled Randomized Hadamard Transform (SRHT) and the sparse CountSketch matrix.
Randomized GMRES algorithms can be used in large-scale parallel environments. Since they rely on randomized sketching techniques, the sketch matrices employed should be highly parallelizable. Many sketch matrices can be generated *on-the-fly* using a seeded random number generator (RNG). However, generating them once in advance often yields better performance [7]. Thus, an effective sketch matrix should be generated once and still be memory-efficient. Furthermore, a large sketch dimension d increases the overall cost of randomized GMRES methods, including storage,

arithmetic operations, and communication. These costs are mitigated by selecting sketches that require relatively small d. Taking everything into account, sketch matrices should be highly parallelizable, memory-efficient, fast to apply, and require only a small sketch dimension d to fulfill (4.2) with high probability.

4.1.1 Gaussian and Rademacher embeddings

Gaussian and Rademacher embeddings are unstructured matrices with (potentially) high storage costs and expensive application times. They benefit from high parallelizability and a relatively small theoretical lower bound on the sketch dimension d. The elements of a rescaled Gaussian matrix are independent Gaussian random variables with zero mean and variance d^{-1} [10]. The entries in a rescaled Rademacher matrix are discrete variables $\pm d^{-1/2}$ drawn with equal probability. Gaussian and Rademacher matrices are applied to $\mathbf{V}$ in $\mathcal{O}(nmd)$ time. In a parallel environment, this requires a single reduce operation, where the reduction operation consists of matrix addition. For both distributions, the theoretical lower bound for the embedding dimension that satisfies (4.2) is $d \geq \mathcal{O}\left(\epsilon^{-2}(m + \log \frac{1}{\delta})\right)$. More specifically, with $0 < \epsilon < 0.572$ and $0 < \delta < 1$, the rescaled Gaussian and rescaled Rademacher distributions over $\mathbb{R}^{d \times n}$ are (ϵ, δ, m) OSEs with

$$d \geq 7.87\epsilon^{-2}(6.9m + \log \tfrac{1}{\delta}) \tag{4.3}$$

(see [10], Proposition 3.7).

4.1.2 Subsampled Randomized Hadamard Transform (SRHT)

Subsampled Randomized Hadamard Transform (SRHT) matrices have the form

$$\mathbf{\Theta} = \sqrt{\frac{n}{d}}\mathbf{RHD},$$

where:

- $\mathbf{D}$ is an $n \times n$ diagonal matrix with ± 1 entries uniformly chosen at random.
- $\mathbf{H}$ is an $n \times n$ Walsh-Hadamard matrix, scaled by $n^{-1/2}$ for orthonormality. This matrix is implicitly applied in $\mathcal{O}(n \log(d))$ time using a series of butterfly operations, similar to the Fourier transformation [9].
- $\mathbf{R} \in \mathbb{R}^{d \times n}$ selects d rows of $\mathbf{H}$ uniformly at random.

By definition, Walsh-Hadamard matrices exist only for values of n that are powers of two. For arbitrary n, the partial SRHT (P-SRHT) is defined as the first n columns of an SRHT matrix of size 2^k, with $k \in \mathbb{N}$ and $n \leq 2^k < 2n$ [9]. If $\epsilon, \delta \in (0, 1)$, the P-SRHT distribution is an (ϵ, δ, m) OSE with

$$d \geq 2(\epsilon^2 - \epsilon^3/3)^{-1}\left(\sqrt{m} + \sqrt{8\log \tfrac{6n}{\delta}}\right)^2 \log \tfrac{3m}{\delta} \text{ [10].} \tag{4.4}$$

This bound is slightly worse than the bound in (4.3) for Gaussian and Rademacher distributions. Additionally, (4.4) depends logarithmically on n, while (4.3) is independent of n. However, sketching $\mathbf{V}$ with a (P-)SRHT matrix takes only $\mathcal{O}(nm \log d)$ time due to the beneficial properties of $\mathbf{H}$, presenting a significant advantage over unstructured matrices [9]. In general, (P-)SRHT matrices cannot be effectively parallelized due to excessive communication overhead.

4.1.3 Block SRHT

The block SRHT in [7] combines high parallelizability with the fast application time of (P-)SRHT matrices, although with a slightly worse bound on d. If $\epsilon, \delta \in (0,1)$, the block SRHT matrix $\mathbf{\Theta} \in \mathbb{R}^{d \times n}$ is an (ϵ, δ, m) OSE with

$$d \geq 3.7\epsilon^{-2}\left(\sqrt{m} + 4\sqrt{\log\tfrac{n}{\delta} + 6.3}\right)^2 \log\tfrac{5m}{\delta} \text{ [7].}$$

$\mathbf{\Theta}$ is distributed among p processes such that $\mathbf{\Theta} = [\mathbf{\Theta}^{\{1\}}, \mathbf{\Theta}^{\{2\}}, \ldots, \mathbf{\Theta}^{\{p\}}]$, where $\mathbf{\Theta}^{\{i\}} \in \mathbb{R}^{d \times n_i}$ belongs to process i, and $n = \sum_{i=1}^{p} n_i$. $\mathbf{\Theta}^{\{i\}}$ is defined as

$$\mathbf{\Theta}^{\{i\}} = \sqrt{\frac{n_i}{d}}\tilde{\mathbf{D}}^{\{i\}}\mathbf{R}\mathbf{H}\mathbf{D}^{\{i\}}, \text{ where:}$$

- $\tilde{\mathbf{D}}^{\{i\}} \in \mathbb{R}^{d \times d}$ and $\mathbf{D}^{\{i\}} \in \mathbb{R}^{n_i \times n_i}$ are diagonal matrices with independent and identically distributed (i.i.d.) Rademacher entries ± 1, generated independently among processes.
- $\mathbf{R} \in \mathbb{R}^{d \times n_i}$ is a unique sampling matrix that selects d rows *with* replacement, if $n_i < m$ and *without* replacement otherwise.
- $\mathbf{H} \in \mathbb{R}^{n_i \times n_i}$ is a unique Walsh-Hadamard matrix, rescaled by $n_i^{-1/2}$.

In their theoretical analysis, the authors in [7] assume a uniform dimension n_i for every process. In practice, n_i can differ between processes. In this case, $\mathbf{R}$ and $\mathbf{H}$ are considered unique if they were generated with unique RNG seeds.

4.1.4 CountSketch

A CountSketch matrix $\mathbf{\Theta} \in \mathbb{R}^{d \times n}$ is a sparse embedding, with one i.i.d. Rademacher entry per column, placed uniformly at random, and zeros otherwise [24]. CountSketch matrices are applied to $\mathbf{V}$ in $\mathcal{O}(mn)$ time and are highly parallelizable. The embedding dimension of a CountSketch matrix is of size $d = \mathcal{O}\left(\epsilon^{-2}m^2\delta^{-1}\right)$ [24, 54], resulting in higher sensitivity to lower failure probabilities δ and larger m, as d grows with δ^{-1} instead of $\log\frac{1}{\delta}$ and with m^2 instead of m like the other OSEs.

4.1.5 CountGauss

The CountGauss embedding in [51] combines two subspace embeddings. The high-dimensional subspace is reduced to a subspace of dimension $\tilde{d}$ using a fast (but less reliable) subspace embedding, such as the CountSketch matrix. Subsequently, a more expensive, yet highly reliable subspace embedding, such as the Gaussian sketch matrix, is used to embed the $\tilde{d}$-dimensional subspace into a d-dimensional subspace, with $\tilde{d} = \mathcal{O}\left(m^2 d\right)$. Using a (P-)SRHT embedding instead of a Gaussian sketch further reduces the application time [63].

4.1.6 Side-by-side comparison

Table 4.1 summarizes the theoretical bounds on d, the application time, and the memory required for each embedding type. Figure 4.1 illustrates the bounds on d with respect to different subspace dimensions m and varying failure probabilities δ. In summary, Gaussian embeddings offer the strongest theoretical guarantees but the weakest performance, CountSketches have the best performance but the weakest guarantees, and BSRHTs and CountGauss sketches are in between.

TABLE 4.1: Comparison of randomized subspace embeddings with respect to application time, memory usage, and required sketch dimension to fulfill the OSE property (4.2).

Subspace embedding	sketch dimension d	application time (local)	memory per processor
Gauss and Rademacher	$\mathcal{O}\left(\epsilon^{-2}(m+\log\frac{1}{\delta})\right)$	$\mathcal{O}\left(n_p m d\right)$	$\mathcal{O}\left(n_p d\right)$
P-SRHT*	$\mathcal{O}\left(\epsilon^{-2}(m+\log\frac{n}{\delta})\log\frac{m}{\delta}\right)$	$\mathcal{O}\left(nm\log d\right)$	$\mathcal{O}\left(n+d\right)$
Block SRHT		$\mathcal{O}\left(n_p m\log n_p\right)$	$\mathcal{O}\left(n_p+d\right)$
CountSketch	$\mathcal{O}\left(\epsilon^{-2}m^2\delta^{-1}\right)$	$\mathcal{O}\left(n_p m\right)$	$\mathcal{O}\left(n_p\right)$
CountGauss (with SRHT)	$\mathcal{O}\left(\epsilon^{-2}(m+\log\frac{1}{\delta})\right)$**	$\mathcal{O}\left(m(n_p+\tilde{d}\log d)\right)$	$\mathcal{O}\left(n_p+\tilde{d}+d\right)$

*not parallelizable **with slightly worse success probability than $1-\delta$

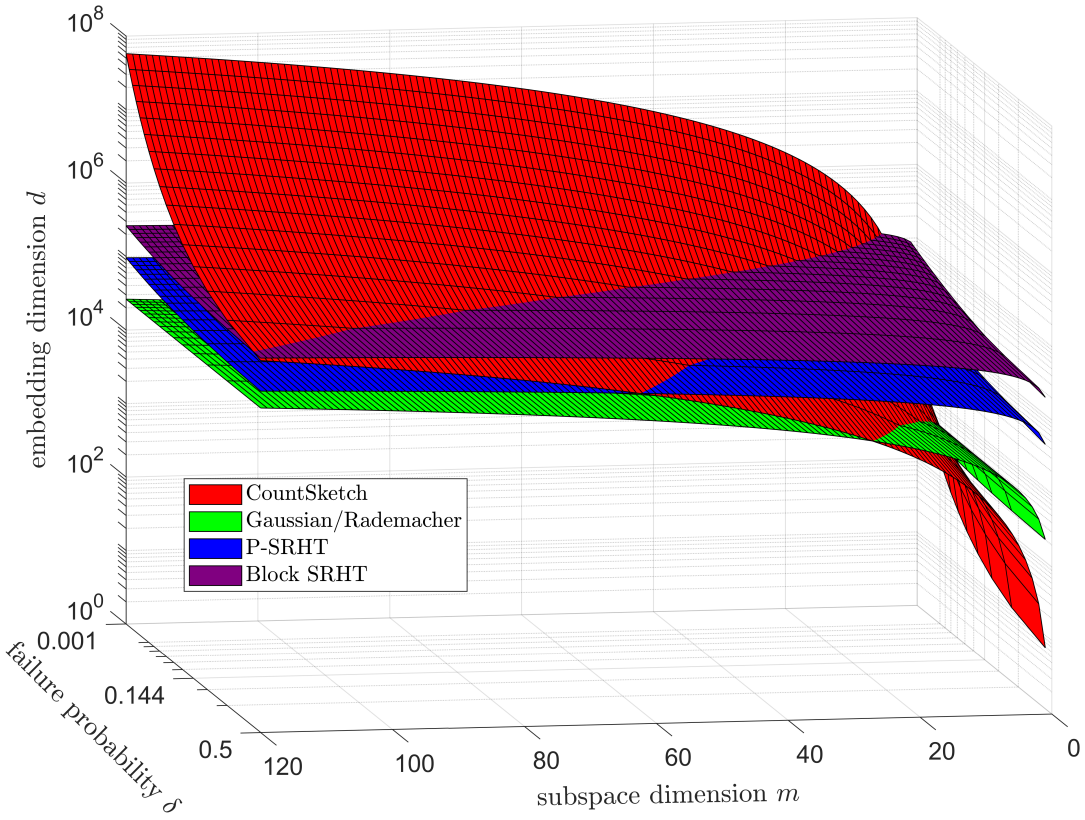

FIGURE 4.1: Illustration of embedding dimension lower bounds required by Gaussian/Rademacher, SRHT-based, and CountSketch subspace embeddings for variable failure probabilities δ and subspace dimensions m, with fixed distortion $\epsilon = 1/2$ (lower d is better). We set $n = 10^{12}$ for SRHT-based sketches, which scale logarithmically with n. CountGauss is a combination of two embeddings and is therefore not included.

4.2 Randomized GMRES

Randomized GMRES [9] is based on a randomized Gram-Schmidt (RGS) process, which orthogonalizes low-dimensional sketches of high-dimensional basis vectors and uses the resulting orthogonalization coefficients to make the high-dimensional vectors approximately orthonormal, resulting in a well-conditioned basis.

4.2.1 Randomized Gram-Schmidt (RGS) process

If $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$ is an OSE for a matrix $\mathbf{V} \in \mathbb{R}^{n\times m}$ as defined in (4.2), then the singular values of $\mathbf{V}$ are bounded by

$$\begin{aligned} \sigma_{\min}(\mathbf{V}) &\geq (1+\epsilon)^{-1/2}\sigma_{\min}(\mathbf{\Theta V}), \\ \sigma_{\max}(\mathbf{V}) &\leq (1-\epsilon)^{-1/2}\sigma_{\max}(\mathbf{\Theta V}). \end{aligned} \tag{4.5}$$

[9] contains a proof. Equation (4.5) suggests that it may be sufficient to orthogonalize the compressed matrix $\mathbf{\Theta V}$ rather than $\mathbf{V}$ to build a well-conditioned basis for $\mathbf{V}$. The resulting R factor is applied to $\mathbf{V}$ to obtain a $\mathbf{\Theta}$-orthonormal matrix $\mathbf{Q} \in \mathbb{R}^{n\times m}$. These are the essential steps of the randomized Gram-Schmidt (RGS) process presented in Algorithm 9. The method applies the projector

$$\mathbf{\Pi}_{RGS}^{(i)} := (\mathbf{I} - \mathbf{Q}_{1:i}(\mathbf{\Theta Q}_{1:i})^{\dagger}\mathbf{\Theta})$$

to the columns of $\mathbf{V}$ in lines 2–4 of Algorithm 9 as part of the QR factorization $\mathbf{P} = \mathbf{SR}$, where $\mathbf{P}$ is the sketched input matrix, $\mathbf{S}$ is the orthonormal sketched basis, and $\mathbf{R}$ is used to approximately orthonormalize the output matrix $\mathbf{Q}$. We define the loss of orthogonality in $\mathbf{S}$ by $\Delta_m = \|\mathbf{I}-\mathbf{S}^{\mathrm{T}}\mathbf{S}\|_{\mathrm{F}}$. The factorization error of the sketched input matrix is given by $\tilde{\Delta}_m = \|\mathbf{P} - \mathbf{SR}\|_{\mathrm{F}}/\|\mathbf{P}\|_{\mathrm{F}}$. Δ_m and $\tilde{\Delta}_m$ are used to confirm the stability of the algorithm (line 10 of Algorithm 9) as will be shown in the following stability analysis.

Algorithm 9 Randomized Gram-Schmidt process

Input: $\mathbf{V} \in \mathbb{R}^{n\times m}$, $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, $m \leq d \ll n$
Output: $\mathbf{\Theta}$-orthonormal $\mathbf{Q} \in \mathbb{R}^{n\times m}$, upper triangular $\mathbf{R} \in \mathbb{R}^{m\times m}$

1: **for** $i = 1 : m$ **do**
2: $\quad \mathbf{p}_i = \mathbf{\Theta v}_i$
3: $\quad \mathbf{r}_{1:i-1,i} = \operatorname{argmin}_{\mathbf{y}} \|\mathbf{S}_{1:i-1}\mathbf{y} - \mathbf{p}_i\|$ ▷ Apply $(\mathbf{\Theta Q}_{1:i-1})^{\dagger}\mathbf{p}_i$
4: $\quad \mathbf{q}'_i = \mathbf{v}_i - \mathbf{Q}_{1:i-1}\mathbf{r}_{1:i-1,i}$
5: $\quad \mathbf{s}'_i = \mathbf{\Theta q}'_i$
6: $\quad r_{i,i} = \|\mathbf{s}'_i\|$
7: $\quad \mathbf{s}_i = \mathbf{s}'_i/r_{i,i}$
8: $\quad \mathbf{q}_i = \mathbf{q}'_i/r_{i,i}$
9: **end for**
10: Optional: check $\Delta_m = \|\mathbf{I} - \mathbf{S}^{\mathrm{T}}\mathbf{S}\|_{\mathrm{F}}$ and $\tilde{\Delta}_m = \|\mathbf{P} - \mathbf{SR}\|_{\mathrm{F}}/\|\mathbf{P}\|_{\mathrm{F}}$

Stability analysis

We list requirements for two key theorems in [9]. These theorems establish the stability of the RGS algorithm independent of the large dimension n. Mixed precision results in [9] are simplified to single precision results using a fixed unit roundoff u.

- Oblivious subspace embeddings defined by (4.2) possess a bounded norm with high probability. If $\boldsymbol{\Theta}$ is an $(\epsilon, \delta/n, 1)$ oblivious subspace embedding, then with probability at least $1-\delta$, it holds that

$$\|\boldsymbol{\Theta}\|_{\mathrm{F}} \leq \sqrt{(1+\epsilon)n}. \tag{4.6}$$

Proof. The fact that $\boldsymbol{\Theta}$ is an ϵ-embedding for every canonical basis vector follows directly from (4.2) and Boole's inequality [3]. This, in turn, indicates that the two-norms of the columns of $\boldsymbol{\Theta}$ are upper bounded by $\sqrt{1+\epsilon}$, with (4.6) as a direct consequence. □

- Randomized sketching does not amplify rounding errors by a factor related to n. Thus, randomized sketching enables stable dimensionality reduction of high-dimensional subspaces [9].

- The following results use the well-known notation introduced by Higham et al. in [44]. Given a matrix or vector ξ, $|\xi|$ has the same elements as ξ in absolute value, $\widehat{\xi}$ is the computed value of ξ in finite precision arithmetic, and inequalities hold component-wise. We define the rounding error vectors

$$\begin{aligned}\Delta\mathbf{q}_i' &:= \widehat{\mathbf{q}}_i' - (\widehat{\mathbf{v}}_i - \widehat{\mathbf{Q}}_{1:i-1}\widehat{\mathbf{r}}_{1:i-1,i}),\\ \Delta\mathbf{q}_i &:= \widehat{\mathbf{q}}_i - \widehat{\mathbf{q}}_i'/\hat{r}_{i,i}\end{aligned}$$

for lines 4 and 8 of Algorithm 9. We can bound these vectors component-wise using the standard worst-case rounding error analysis in [44]:

$$|\Delta\mathbf{q}_i'| \leq 1.02\tilde{u}(|\widehat{\mathbf{v}}_i| + i|\widehat{\mathbf{Q}}_{1:i-1}||\widehat{\mathbf{r}}_{1:i-1,i}|), \tag{4.7}$$
$$|\Delta\mathbf{q}_i| \leq u|\widehat{\mathbf{q}}_i'/\hat{r}_{i,i}|, \tag{4.8}$$

where $\tilde{u} = F(m,n)u$, and $F(m,n)$ is a low-degree polynomial. Furthermore, with $\epsilon \leq 1/2$ and $1 \leq i \leq m$, we assume that the inequalities

$$\|\boldsymbol{\Theta}\Delta\mathbf{q}_i'\| \leq 1.02\tilde{u}D\||\widehat{\mathbf{v}}_i| + i|\widehat{\mathbf{Q}}_{1:i-1}||\widehat{\mathbf{r}}_{1:i-1,i}|\| \tag{4.9}$$

and

$$\|\boldsymbol{\Theta}\Delta\mathbf{q}_i\| \leq uD\|\widehat{\mathbf{q}}_i'/\hat{r}_{i,i}\| \tag{4.10}$$

hold with $D = \sqrt{1+\epsilon}$ under the stochastic rounding error model in [26], or with $D = \sqrt{(1+\epsilon)n}$ under the standard worst-case rounding error model.

Main result I [9, Theorem 3.2] Assume that inequalities (4.6)–(4.10) hold, as well as $100m^{1/2}n^{3/2}u \leq 0.01m^{-1}$, and $n \geq 100$. If $\boldsymbol{\Theta}$ is an ϵ-embedding for $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{V}}$ from Algorithm 9, with $\epsilon \leq 1/2$, and if Δ_m, $\tilde{\Delta}_m \leq 0.1$ in line 10 of Algorithm 9, then the following inequalities hold

$$(1+\epsilon)^{-1/2}(1-\Delta_m-0.1\tilde{u}) \leq \sigma_{\min}(\widehat{\mathbf{Q}}) \leq \sigma_{\max}(\widehat{\mathbf{Q}}) \leq (1-\epsilon)^{-1/2}(1+\Delta_m+0.1\tilde{u}), \tag{4.11}$$

$$\|\widehat{\mathbf{V}} - \widehat{\mathbf{Q}}\widehat{\mathbf{R}}\|_{\mathrm{F}} \leq 3.7\tilde{u}m^{3/2}\|\widehat{\mathbf{V}}\|_{\mathrm{F}}. \tag{4.12}$$

The supplementary material in [9] contains a proof. If the sketch dimension d is large enough, $\epsilon \leq 1/2$ will be satisfied. The key factor determining the stability of Algorithm 9 depends on whether Δ_m and $\tilde{\Delta}_m$ are sufficiently small. This is the case if the least-squares problem in line 3 of Algorithm 9 is solved with a sufficiently accurate

least-squares solver (for example, HH-QR). The second main result establishes bounds on Δ_m and $\tilde{\Delta}_m$.

Main result II [9, Theorem 3.3] Assume that an unconditionally stable QR factorization algorithm is used to solve the least-squares problem in line 3 of Algorithm 9. If inequalities (4.6)–(4.10) hold and if $\boldsymbol{\Theta}$ is an ϵ-embedding for $\widehat{\mathbf{Q}}_{1:m-1}$ and $\widehat{\mathbf{V}}$, with $\epsilon \leq 1/2$, and if $\tilde{u} \leq 10^{-3}\kappa(\widehat{\mathbf{V}})^{-1}m^{-2}$, then Δ_m and $\tilde{\Delta}_m$ are bounded by

$$\Delta_m \leq 20\tilde{u}m^2\kappa(\widehat{\mathbf{V}}), \qquad \tilde{\Delta}_m \leq 6\tilde{u}m^{3/2}. \tag{4.13}$$

See the supplementary material in [9] for a proof.

***A priori* and *a posteriori* analysis** Estimating $\kappa(\widehat{\mathbf{V}})$ in (4.13) is challenging in practice, making it difficult to obtain stability guarantees *a priori*. However, by computing Δ_m and $\tilde{\Delta}_m$ and using Theorem 3.2 from [9], it is possible to efficiently perform stability certification *a posteriori*. Note that the computation of Δ_m and $\tilde{\Delta}_m$ neither involves operations on high-dimensional objects nor requires communication during parallel execution.
Furthermore, it may be important to validate that $\boldsymbol{\Theta}$ is an ϵ-embedding for $\widehat{\mathbf{Q}}$ and $\widehat{\mathbf{V}}$. Assume that $\boldsymbol{\Theta}$ and $\widehat{\mathbf{V}}$ are independent of each other and that $\boldsymbol{\Theta}$ is an (ϵ, δ, m) OSE. Then $\boldsymbol{\Theta}$ serves as an ϵ-embedding for $\widehat{\mathbf{V}}$. By making another assumption that $\widehat{\mathbf{Q}}$ is generated depending on $\widehat{\mathbf{V}}$ and $\boldsymbol{\Theta}$, we have that $\boldsymbol{\Theta}$ is an ϵ'-embedding for $\widehat{\mathbf{Q}}$ with

$$\epsilon' = 2\epsilon + 180\tilde{u}m^2\kappa(\widehat{\mathbf{V}}). \tag{4.14}$$

The supplementary material of [9] contains a proof. (4.14) imposes a mild condition on the ϵ-embedding quality of $\boldsymbol{\Theta}$ with respect to $\widehat{\mathbf{Q}}$. If $\widehat{\mathbf{V}}$ depends on $\boldsymbol{\Theta}$ (as is the case in Algorithm 10 in Section 4.2.2), the *a priori* analysis can be excessively pessimistic. In this scenario, it may be important to validate the ϵ-embedding property of $\boldsymbol{\Theta}$ *a posteriori*. See [9] for details on an *a posteriori* certification process, which involves applying another OSE (in addition to $\boldsymbol{\Theta}$) to either $\widehat{\mathbf{V}}$ or $\widehat{\mathbf{Q}}$.

Performance analysis

Similar to [9], our performance analysis compares the RGS process to the CGS method. Considering high-dimensional operations, the RGS process requires only one pass over $\mathbf{Q}_{1:i-1}$ during the projection step (line 4), whereas the CGS method requires two. As a result, the projection step in line 4 of Algorithm 9 has time complexity $\mathcal{O}(2ni)$, which is half the cost of the CGS projection. Sketching the vectors $\mathbf{v}_i$ and $\mathbf{q}_i$ in lines 2 and 5 costs $\mathcal{O}(n \log d)$ time each, assuming that $\boldsymbol{\Theta}$ is an SRHT embedding. Although these steps are costly for small values of i per iteration, their impact diminishes as i increases.
Operations on low-dimensional objects are inexpensive when the sketch dimension d is small, but the cost increases significantly as d grows larger. Line 3 of Algorithm 9 involves solving the low-dimensional system $\mathbf{S}_{1:i-1}\mathbf{y} = \mathbf{p}_i$ for $\mathbf{y}$. Due to numerical stability reasons, high-quality direct solvers like HH-QR (see Section 2.1) should be employed [9]. If the sketch dimension d is very large, the authors in [9] recommend using faster but less stable methods. Orthonormal $\mathbf{S}_{1:i-1}$ implies $\mathbf{S}_{1:i-1}^{\dagger} = \mathbf{S}_{1:i-1}^{\mathrm{T}}$ and thus line 3 of Algorithm 9 could be simplified to

$$\mathbf{r}_{1:i-1,i} = \mathbf{S}_{1:i-1}^{\mathrm{T}}\mathbf{p}_i.$$

Moreover, since $\mathbf{S}$ is nearly orthonormal, the normal equations

$$\mathbf{S}_{1:i-1}^{\mathrm{T}}\mathbf{S}_{1:i-1}\mathbf{y} = \mathbf{S}_{1:i-1}^{\mathrm{T}}\mathbf{p}_i$$

can be solved efficiently using methods like Conjugate Gradients [62], GMRES, or by applying several reorthogonalization steps (i.e., Richardson iterations) [9].
Both RGS and CGS require two global synchronizations per iteration, but they can be modified to require only one. For the single-synchronization version of CGS, see [53]. To transform RGS into a single-synchronization algorithm, the computation of sketches in lines 2 and 5 of Algorithm 9 must be combined into a single step. This is achieved through delayed normalization in lines 6–8 of Algorithm 9. The global synchronization in iteration i, with $1 \leq i \leq m-1$, involves the simultaneous sketching of two vectors $\mathbf{q}_i'$, and $\mathbf{v}_{i+1}' = \mathbf{A}\mathbf{q}_i'$. Subsequently, the normalization lag consists of the following steps: $r_{i,i} = \|\mathbf{s}_i'\|$, $\mathbf{s}_i = \mathbf{s}_i'/r_{i,i}$, $\mathbf{q}_i = \mathbf{q}_i'/r_{i,i}$, $\mathbf{p}_{i+1} = \mathbf{p}_{i+1}'/r_{i,i}$, and $\mathbf{v}_{i+1} = \mathbf{v}_{i+1}'/r_{i,i}$.
In terms of communication, the number of global synchronizations is comparable to CGS.

4.2.2 RGS-GMRES

The RGS-GMRES method builds on a randomized Arnoldi algorithm, for which the RGS stability results from Section 4.2.1 also apply [9]. Algorithm 10 presents the randomized Arnoldi algorithm using the single-synchronization strategy discussed in the performance analysis in Section 4.2.1. RGS-GMRES performs additional steps, including solving a small least-squares problem and updating the approximate solution. After i iterations, RGS-GMRES computes an $n \times i$ $\boldsymbol{\Theta}$-orthonormal basis matrix $\mathbf{Q}$, and an $i \times (i-1)$ upper Hessenberg matrix $\mathbf{H}$. The method then evaluates $\mathbf{y}^{(k)} = \operatorname{argmin}_{\mathbf{y}} \|\|\boldsymbol{\Theta}\mathbf{r}^{(k)}\|\mathbf{e}_1 - \mathbf{H}\mathbf{y}\|$ and subsequently updates the new approximation as $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{Q}_{1:i-1}\mathbf{y}^{(k)}$. Using the *sketched* Arnoldi relation

$$\boldsymbol{\Theta}\mathbf{A}\mathbf{Q}_{1:i-1} = \boldsymbol{\Theta}\mathbf{Q}\mathbf{H}, \tag{4.15}$$

the least-squares problem can be expressed as follows:

$$\begin{aligned}\operatorname*{argmin}_{\mathbf{y}} \|\|\boldsymbol{\Theta}\mathbf{r}^{(k)}\|\mathbf{e}_1 - \mathbf{H}\mathbf{y}\| &= \operatorname*{argmin}_{\mathbf{y}} \|\boldsymbol{\Theta}\mathbf{Q}(\|\boldsymbol{\Theta}\mathbf{r}^{(k)}\|\mathbf{e}_1 - \mathbf{H}\mathbf{y})\| = \\ \operatorname*{argmin}_{\mathbf{y}} \|\boldsymbol{\Theta}(\mathbf{r}^{(k)} - \mathbf{Q}\mathbf{H}\mathbf{y})\| &= \operatorname*{argmin}_{\mathbf{y}} \|\boldsymbol{\Theta}(\mathbf{r}^{(k)} - \mathbf{A}\mathbf{Q}_{1:i-1}\mathbf{y})\|.\end{aligned}$$

This shows that RGS-GMRES minimizes the *sketched* residual norm. If $\boldsymbol{\Theta}$ is an ϵ-embedding for $\mathbf{Q}$, and if Δ_i and $\tilde{\Delta}_i$ are sufficiently small such that (4.11) holds, we have that the sketched residual norm of RGS-GMRES differs from the residual norm by a factor of $\sqrt{(1+\epsilon)/(1-\epsilon)}$ [9]. More specifically, let $\mathbf{x}_\star$ be the approximate solution of GMRES in iteration i, and let $\tilde{\mathbf{x}}$ be the approximate solution of RGS-GMRES in iteration i. Using (4.1), we have that the residual norm with respect to $\tilde{\mathbf{x}}$ is bounded by

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}_\star\| \leq \|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\| \leq \sqrt{\frac{1+\epsilon}{1-\epsilon}}\|\mathbf{b} - \mathbf{A}\mathbf{x}_\star\|. \tag{4.16}$$

This bound applies to non-restarted GMRES. Establishing error bounds for restarted RGS-GMRES is challenging due to the randomly distorted start vector at each restart, which can affect convergence.

Algorithm 10 Single-sync RGS-Arnoldi

Input: $\mathbf{A} \in \mathbb{R}^{n\times n}$, start vector $\mathbf{v}_1 \in \mathbb{R}^n$, $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, $d \ll n$
Output: $\mathbf{\Theta}$-orthonormal $\mathbf{Q} \in \mathbb{R}^{n\times m}$, upper triangular $\mathbf{R} \in \mathbb{R}^{m\times m}$

1: $\mathbf{p}_1 = \mathbf{\Theta}\mathbf{v}_1$
2: **for** $i = 1 : m-1$ **do**
3: $\quad \mathbf{r}_{1:i-1,i} = \operatorname{argmin}_{\mathbf{y}} \|\mathbf{S}_{1:i-1}\mathbf{y} - \mathbf{p}_i\|$ ▷ Apply $(\mathbf{\Theta}\mathbf{Q}_{1:i-1})^\dagger \mathbf{p}_i$
4: $\quad \mathbf{q}'_i = \mathbf{v}_i - \mathbf{Q}_{1:i-1}\mathbf{r}_{1:i-1,i}$
5: $\quad \mathbf{v}'_{i+1} = \mathbf{A}\mathbf{q}'_i$
6: $\quad [\mathbf{s}'_i, \mathbf{p}'_{i+1}] = \mathbf{\Theta}[\mathbf{q}'_i, \mathbf{v}'_{i+1}]$ ▷ Global sync
7: $\quad r_{i,i} = \|\mathbf{s}'_i\|$
8: $\quad \mathbf{s}_i = \mathbf{s}'_i / r_{i,i}$
9: $\quad \mathbf{q}_i = \mathbf{q}'_i / r_{i,i}$
10: $\quad \mathbf{p}_{i+1} = \mathbf{p}'_{i+1} / r_{i,i}$
11: $\quad \mathbf{v}_{i+1} = \mathbf{v}'_{i+1} / r_{i,i}$
12: **end for**
13: $\mathbf{r}_{1:m-1,m} = \operatorname{argmin}_{\mathbf{y}} \|\mathbf{S}_{1:m-1}\mathbf{y} - \mathbf{p}_m\|$
14: $\mathbf{q}'_m = \mathbf{v}_m - \mathbf{Q}_{1:m-1}\mathbf{r}_{1:m-1,m}$
15: $\mathbf{s}'_m = \mathbf{\Theta}\mathbf{q}'_m$
16: $r_{m,m} = \|\mathbf{s}'_m\|$
17: $\mathbf{s}_m = \mathbf{s}'_m / r_{m,m}$
18: $\mathbf{q}_m = \mathbf{q}'_m / r_{m,m}$
19: Optional: check $\Delta_m = \|\mathbf{I} - \mathbf{S}^{\mathrm{T}}\mathbf{S}\|_{\mathrm{F}}$ and $\tilde{\Delta}_m = \|\mathbf{P} - \mathbf{S}\mathbf{R}\|_{\mathrm{F}} / \|\mathbf{P}\|_{\mathrm{F}}$

4.3 Sketched GMRES

After i iterations, the sketched GMRES (sGMRES) method approximates the solution to the least-squares problem in (2.7) by solving the *sketched* least-squares problem

$$\operatorname*{argmin}_{\mathbf{y}} \|\mathbf{\Theta}(\mathbf{r}^{(0)} - \mathbf{A}\mathbf{Q}_{1:i}\mathbf{y})\|. \tag{4.17}$$

This differs from the RGS-GMRES method, which solves a sketched least-squares problem based on sketched inner products using (4.1). Thus, sGMRES requires a subspace embedding definition that is suitable for sketched least-squares problems.

Definition 4.3.1. *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*

$$\forall \mathbf{x} \in \mathbb{R}^m : (1-\epsilon)\|\mathbf{V}\mathbf{x}\| \le \|\mathbf{\Theta}\mathbf{V}\mathbf{x}\| \le (1+\epsilon)\|\mathbf{V}\mathbf{x}\| \text{ [58]}. \tag{4.18}$$

The OSE property (4.2) also holds for (4.18). Similar to (4.5), the singular values of $\mathbf{V}$ are bounded by

$$\begin{aligned} \sigma_{\min}(\mathbf{V}) &\ge (1+\epsilon)^{-1}\sigma_{\min}(\mathbf{\Theta}\mathbf{V}), \\ \sigma_{\max}(\mathbf{V}) &\le (1-\epsilon)^{-1}\sigma_{\max}(\mathbf{\Theta}\mathbf{V}). \end{aligned} \tag{4.19}$$

A restarted version of sGMRES, denoted sGMRES(m), is outlined in Algorithm 11. Note that the matrix $\mathbf{R}^{-1}_{1:i,1;i}$ in line 18 of Algorithm 11 is uniquely defined, since $(\mathbf{R}^{-1})_{1:i,1:i} = (\mathbf{R}_{1:i,1:i})^{-1}$ for triangular $\mathbf{R}$. Unlike RGS-GMRES, sGMRES does not require the Arnoldi relation. Therefore, there are no orthogonality constraints on the Krylov basis $\mathbf{Q}_{1:i}$, except that $\kappa(\mathbf{A}\mathbf{Q}_{1:i}) \lesssim u^{-1}$ must be satisfied. This provides a lot of

flexibility in selecting the projector in line 8 of Algorithm 11. Line 13 of Algorithm 11 monitors $\kappa(\mathbf{\Theta A Q}_{1:i}) \leq \Omega_{\mathbf{C}}$, instead of $\kappa(\mathbf{AQ}_{1:i}) \leq \Omega_{\mathbf{C}}$, where $\Omega_{\mathbf{C}} = \mathcal{O}\left(u^{-1}\right)$.
Since there is no Hessenberg matrix from which the implicit residual norm can be obtained (see Section 2.3), a residual norm estimator is provided to evaluate the sketched residual norm (line 15 of Algorithm 11) [58].

Algorithm 11 sGMRES(m)

Input: $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$, initial guess $\mathbf{x}^{(0)}$, truncation parameter t, $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, $d \ll n$, condition number tolerance $\Omega_{\mathbf{C}}$
Output: Approximate solution to $\mathbf{Ax} = \mathbf{b}$

1: Draw $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$
2: **for** $k = 0, 1, \ldots,$ until convergence **do**
3: $\quad \mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$
4: $\quad \mathbf{q}_1 = \mathbf{r}^{(k)}/\|\mathbf{r}^{(k)}\|$
5: $\quad \mathbf{v}_1 = \mathbf{A}\mathbf{q}_1$
6: $\quad [\tilde{\mathbf{r}}^{(k)}, \mathbf{c}_1] = \mathbf{\Theta}[\mathbf{r}^{(k)}, \mathbf{v}_1]$
7: $\quad$ **for** $i = 2 : m$ **do**
8: $\quad\quad \mathbf{q}'_{i-1} = \mathbf{\Pi}^{(i-1)}\mathbf{v}_{i-1}$
9: $\quad\quad \mathbf{q}_i = \mathbf{q}'_{i-1}/\|\mathbf{q}'_{i-1}\|$
10: $\quad\quad \mathbf{v}_i = \mathbf{A}\mathbf{q}_i$
11: $\quad\quad \mathbf{c}_i = \mathbf{\Theta}\mathbf{v}_i$
12: $\quad\quad$ Update thin QR factorization $\mathbf{C}_{1:i} = \hat{\mathbf{Q}}_{1:i}\mathbf{R}_{1:i,1:i}$
13: $\quad\quad j = \texttt{ICE}(\mathbf{R}_{1:i,1:i}, \Omega_{\mathbf{C}})$
14: $\quad\quad$ **if** $j \neq i$ **then** $i = j$, **break** ▷ Restart
15: $\quad\quad \hat{r} = \|(\mathbf{I} - \hat{\mathbf{Q}}_{1:j}\hat{\mathbf{Q}}_{1:j}^{\mathrm{T}})\tilde{\mathbf{r}}^{(k)}\|$ ▷ Residual estimate
16: $\quad\quad$ **if** converged **then break**
17: $\quad$ **end for**
18: $\quad \mathbf{y}^{(k)} = \mathbf{R}_{1:i,1:i}^{-1}(\hat{\mathbf{Q}}_{1:i}^{\mathrm{T}}\tilde{\mathbf{r}}^{(k)})$ via triangular solve ▷ $\operatorname{argmin}_{\mathbf{y}} \|\mathbf{\Theta}(\mathbf{r}^{(k)} - \mathbf{AQ}_{1:i}\mathbf{y})\|$
19: $\quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{Q}_{1:i}\mathbf{y}^{(k)}$
20: **end for**

4.3.1 Stability analysis

From (4.19) follows the condition number bound

$$\frac{1-\epsilon}{1+\epsilon}\kappa(\mathbf{\Theta V}) \leq \kappa(\mathbf{V}) \leq \frac{1+\epsilon}{1-\epsilon}\kappa(\mathbf{\Theta V}) \text{ [58].} \tag{4.20}$$

According to (4.20), $\kappa(\mathbf{\Theta AQ}_{1:i}) \lesssim u^{-1}$ implies $\kappa(\mathbf{AQ}_{1:i}) \lesssim u^{-1}$. Assuming that this implication holds, a nearly optimal solution to the least-squares problem (4.17) is computed using unconditionally stable QR factorizations such as HH-QR (see [44, Theorem 19.3]). The ICE (see Section 3.3.1) inexpensively monitors $\kappa(\mathbf{\Theta AQ}_{1:i})$ in line 13 of Algorithm 11.
The bounds on the sketched residual norm are derived analogously to (4.16). Let $\mathbf{x}_\star$ be the solution of GMRES in iteration i. Using (4.18) we have that the residual norm with respect to the solution $\tilde{\mathbf{x}}$ computed by sGMRES is bounded by

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}_\star\| \leq \|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\| \leq \frac{1+\epsilon}{1-\epsilon}\|\mathbf{b} - \mathbf{A}\mathbf{x}_\star\| \text{ [58].}$$

4.3.2 Performance analysis

Since there are no orthogonality constraints on the Krylov basis $\mathbf{Q}_{1:i}$, the authors in [58] use the truncated classical Gram-Schmidt (TCGS) process. TCGS performs fast partial orthogonalization rather than costly full orthogonalization. Consequently, $\mathbf{Q}_{1:i}$ is not orthonormal and likely poorly conditioned. TCGS orthogonalizes new vectors with respect to the t most recently orthogonalized vectors, where t is typically a small constant (for example, $t = 2$). The projector for TCGS in Algorithm 1 is given by

$$\mathbf{\Pi}_{TCGS}^{(i)} := (\mathbf{I} - \mathbf{Q}_{w:i}\mathbf{Q}_{w:i}^{\mathrm{T}}),$$

where $w = \max(1, i-t+1)$. TCGS reduces the orthogonalization cost from quadratic $\mathcal{O}\left(ni^2\right)$ time to linear $\mathcal{O}\left(nit\right)$ time in the number of iterations i. Furthermore, the basis is likely to be better conditioned than the monomial basis [58]. TCGS is not a conventional QR factorization, since the Q factor is not orthonormal. The R factor is a diagonal matrix, with t nonzero superdiagonals.

If $\kappa(\mathbf{\Theta}\mathbf{A}\mathbf{Q}_{1:i}) > \Omega_{\mathbf{C}}$, "basis whitening" can be used to lower the condition number of the Q factor (see [58] for details). A less expensive alternative would be to restart the algorithm early (see line 14 of Algorithm 11), although this may slow down convergence.

As with RGS-GMRES, sketching the vectors $\mathbf{v}_i$ in line 11 of Algorithm 11 is a dominant cost with $\mathcal{O}\left(n \log d\right)$ time, using an SRHT embedding.

Lastly, Algorithm 11 allows for minor performance optimizations. Assuming that a HH-QR factorization or any other suitably stable routine is used, the QR factorization in line 12 of Algorithm 11 should be updated as described in Section 2.1.3. Furthermore, the factorization computes the R factor in its entirety before it is evaluated by the ICE in line 13 of Algorithm 11. Embedding the ICE into the factorization process allows for efficient partial factorization, similar to the partial CholQR factorization described in Algorithm 7.

Chapter 5

Randomization in s-step GMRES methods

This chapter presents two randomized s-step GMRES methods that build on the single-step methods discussed in Chapter 4. The first method in Section 5.1 is a *randomized* s-step GMRES algorithm that employs the randomized block orthogonalization process from [8]. The second method, introduced in Section 5.2, is our novel *randomized & sketched* s-step GMRES approach, which combines randomized block orthogonalization with sketched least-squares problems. Both algorithms are formulated as adaptive s-step methods, similar to the BCGSI2-GMRES algorithm discussed in Section 3.3.

5.1 Randomized s-step GMRES

This section introduces the randomized block Gram-Schmidt (RBGS) process, the s-step RBGS-Arnoldi method, and the RBGS-GMRES method, with each method building upon the previous one.

5.1.1 Randomized Block Gram-Schmidt (RBGS)

The RBGS process [8] is a blocked version of the RGS process described in Section 4.2.1. Similar to RGS, RBGS transforms a set of high-dimensional basis vectors into a $\mathbf{\Theta}$-orthonormal and well-conditioned set of vectors. Algorithm 12 outlines the RBGS algorithm. We discuss three strategies from [8] for intra-orthogonalization of $\mathbf{Q}'_b$ with respect to $\mathbf{\Theta}$ in line 7 of Algorithm 12.

Algorithm 12 RBGS

Input: $\mathbf{V} \in \mathbb{R}^{n\times m}$, step size s with $m = sp$, $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, $d \ll n$
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: $\quad i = s(j-1)$
3: $\quad b = i+1 : i+s$
4: $\quad \mathbf{P}_b = \mathbf{\Theta}\mathbf{V}_b$
5: $\quad \mathbf{R}_{1:i,b} = \operatorname{argmin}_{\mathbf{Y}} \|\mathbf{S}_{1:i}\mathbf{Y} - \mathbf{P}_b\|_{\mathrm{F}}$
6: $\quad \mathbf{Q}'_b = \mathbf{V}_b - \mathbf{Q}_{1:i}\mathbf{R}_{1:i,b}$
7: $\quad$ Apply randomized QR factorization strategy to $\mathbf{Q}'_b$ to obtain $\mathbf{Q}_b\mathbf{R}_{b,b}$
8: $\quad \mathbf{S}_b = \mathbf{\Theta}\mathbf{Q}_b$
9: **end for**
10: Optional: check $\Delta_m = \|\mathbf{I} - \mathbf{S}^{\mathrm{T}}\mathbf{S}\|_{\mathrm{F}}$ and $\tilde{\Delta}_m = \|\mathbf{P} - \mathbf{S}\mathbf{R}\|_{\mathrm{F}}/\|\mathbf{P}\|_{\mathrm{F}}$

1) Explicit intra-orthogonalization The explicit strategy performs a randomized CholQR factorization [6] as described in Algorithm 13. The steps of randomized CholQR involve a QR factorization of the sketch $\mathbf{\Theta Q}'_b$ with subsequent use of the R factor to compute $\mathbf{Q}_b = \mathbf{Q}'_b\mathbf{R}_{b,b}^{-1}$ via forward substitution.

Algorithm 13 Step 7 of Algorithm 12: Randomized CholQR

Input: $\mathbf{Q}'_b \in \mathbb{R}^{n\times s}$, $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$
Output: $\mathbf{\Theta}$-orthonormal $\mathbf{Q}_b \in \mathbb{R}^{n\times s}$, upper triangular $\mathbf{R}_{b,b} \in \mathbb{R}^{s\times s}$
1: $\mathbf{S}'_b = \mathbf{\Theta Q}'_b$
2: Compute $\mathbf{R}_{b,b}$ as the R factor of the thin QR factorization of $\mathbf{S}'_b$
3: $\mathbf{Q}_b = \mathbf{Q}'_b\mathbf{R}_{b,b}^{-1}$ via triangular solve

2) Implicit intra-orthogonalization The implicit strategy, presented in Algorithm 14, avoids the sketching step. Instead, the sketch of $\mathbf{Q}'_b$ is implicitly computed from the previously sketched vectors $\mathbf{P}_b$ and $\mathbf{S}_{1:i}$ in line 1 of Algorithm 14.

Algorithm 14 Step 7 of Algorithm 12: Randomized CholQR with implicit sketch

Input: $\mathbf{Q}'_b \in \mathbb{R}^{n\times s}$, sketched basis vectors $\mathbf{P}_b \in \mathbb{R}^{d\times s}$ and $\mathbf{S}_{1:i} \in \mathbb{R}^{d\times i}$, projection coefficients $\mathbf{R}_{1:i,b} \in \mathbb{R}^{i\times s}$
Output: $\mathbf{\Theta}$-orthonormal $\mathbf{Q}_b \in \mathbb{R}^{n\times s}$, upper triangular $\mathbf{R}_{b,b} \in \mathbb{R}^{s\times s}$
1: $\mathbf{S}'_b = \mathbf{P}_b - \mathbf{S}_{1:i}\mathbf{R}_{1:i,b}$
2: Compute $\mathbf{R}_{b,b}$ as the R factor of the thin QR factorization of $\mathbf{S}'_b$
3: $\mathbf{Q}_b = \mathbf{Q}'_b\mathbf{R}_{b,b}^{-1}$ via triangular solve

3) Intra-orthogonalization with preprocessing step To further improve stability, Algorithm 15 uses a QR factorization as a preprocessing step before computing the randomized CholQR factorization.

Algorithm 15 Step 7 of Algorithm 12: Randomized CholQR with preprocessing step

Input: $\mathbf{Q}'_b \in \mathbb{R}^{n\times s}$, $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$
Output: $\mathbf{\Theta}$-orthonormal $\mathbf{Q}_b \in \mathbb{R}^{n\times s}$, upper triangular $\mathbf{R}_{b,b} \in \mathbb{R}^{s\times s}$
1: Compute thin QR factorization $\mathbf{Q}'_b = \hat{\mathbf{Q}}\hat{\mathbf{R}}$ ▷ Preprocessing step
2: $\mathbf{S}'_b = \mathbf{\Theta}\hat{\mathbf{Q}}$
3: Compute $\mathbf{R}'$ as the R factor of the thin QR factorization of $\mathbf{S}'_b$
4: $\mathbf{Q}_b = \hat{\mathbf{Q}}'\mathbf{R}'^{-1}$ via triangular solve
5: $\mathbf{R}_{b,b} = \mathbf{R}'\hat{\mathbf{R}}$

Stability analysis

We highlight important stability results from [8], which resemble those of RGS discussed in Section 4.2.1. Specifically, equations (4.11)–(4.13) apply to RBGS under similar assumptions, and stability can be verified by checking whether the following bounds are satisfied:

$$\Delta_m = \|\mathbf{I} - \mathbf{S}^{\mathrm{T}}\mathbf{S}\|_{\mathrm{F}} \leq 0.1,$$
$$\tilde{\Delta}_m = \|\mathbf{P} - \mathbf{SR}\|_{\mathrm{F}}/\|\mathbf{P}\|_{\mathrm{F}} \leq 0.1 \text{ [8]}.$$

Moreover, the *a priori* and *a posteriori* analysis of RBGS is essentially the same as that in Section 4.2.1.

Stability of intra-orthogonalization The intra-orthogonalization strategy in line 7 of Algorithm 12 must satisfy the following condition:

$$1 - 0.1\tilde{u}\kappa(\widehat{\mathbf{Q}}'_b) \leq \sigma_{\min}(\boldsymbol{\Theta}\widehat{\mathbf{Q}}_b) \leq \sigma_{\max}(\boldsymbol{\Theta}\widehat{\mathbf{Q}}_b) \leq 1 + 0.1\tilde{u}\kappa(\widehat{\mathbf{Q}}'_b), \tag{5.1}$$

where $\tilde{u} = F(m,n)u$, and $F(m,n)$ is a low-degree polynomial [8]. The randomized CholQR factorization described in Algorithm 13 offers stability guarantees similar to those of the RGS process, provided that an unconditionally stable QR factorization is used in line 2 of Algorithm 13. In practice, randomized CholQR is less numerically stable compared to RGS [9]. Nonetheless, randomized CholQR assumes $\kappa(\mathbf{V}) < \tilde{u}^{-1}$, which is a much weaker condition than the one imposed on the CholQR factorization (see Table 2.1) [6]. The intra-orthogonalization with a preprocessing step in Algorithm 15 yields equal or even better stability than Algorithm 13, provided that an unconditionally stable QR factorization is used in line 1 of Algorithm 15. Algorithms 13 and 15 achieve the bound in (5.1), whereas the implicit strategy in Algorithm 14 does not directly satisfy (5.1) [8]. The authors in [8] note that using the implicit approach in the RBGS process is equivalent to using the explicit strategy with a perturbation that changes $\widehat{\mathbf{Q}}'_b$ to a different matrix, which is implicitly sketched, followed by a second perturbation that transforms it back to $\widehat{\mathbf{Q}}'_b$ when the R factor is applied. These perturbations increase $\kappa(\boldsymbol{\Theta}\widehat{\mathbf{Q}}_b)$ by at most a factor of $\mathcal{O}\left(\tilde{u}\kappa(\mathbf{V})F(m)\right)$, where $F(m)$ is a small polynomial. Consequently, the implicit strategy does not affect the stability guarantees of RBGS up to small constants [8].

Stability of inter-orthogonalization Similar to RGS, the stability of RBGS depends on the quality of the solution to the least-squares problem computed in line 5 of Algorithm 12. If the sketch dimension d is sufficiently small, using an unconditionally stable QR factorization method, such as HH-QR, is recommended. However, when d is too large, direct solvers become prohibitively expensive. In such cases, l rounds of Richardson iterations can be applied to the normal equations $(\mathbf{S}_{1:i}^{\mathrm{T}}\mathbf{S}_{1:i})\mathbf{R}_{1:i,b} = \mathbf{S}_{1:i}^{\mathrm{T}}\mathbf{P}_b$, provided that $\mathbf{S}_{1:i}$ is well-conditioned. Algorithm 16 presents a variant of Richardson iterations that corresponds to orthogonalizing $\mathbf{P}_b$ against $\mathbf{S}_{1:i}$ using multiple rounds of CGS reorthogonalization [8]. Additionally, iterative solvers, such as Conjugate Gradients or GMRES, are efficient alternatives to direct solvers as well. Refer to the supplementary material in [8] for details.

Algorithm 16 Richardson iterations (CGS reorthogonalization), applicable in step 5 of Algorithm 12

Input: Orthonormal $\mathbf{S}_{1:i} \in \mathbb{R}^{d\times i}$, sketched basis vectors $\mathbf{P}_b \in \mathbb{R}^{d\times s}$, number of iterations l
Output: Projection coefficients $\mathbf{R}_{1:i,b} \in \mathbb{R}^{i\times s}$
$\mathbf{R}_{1:i,b} = \mathbf{0}_{1:i,b}$ ▷ Initialize a zero matrix of size $i \times s$.
for $i = 1 : l$ **do**
  $\mathbf{R}_{1:i,b} \leftarrow \mathbf{R}_{1:i,b} + \mathbf{S}_{1:i}^{\mathrm{T}}(\mathbf{P}_b - \mathbf{S}_{1:i}\mathbf{R}_{1:i,b})$
end for

Performance analysis

The RBGS algorithm with SRHT subspace embedding (see Section 4.1) requires roughly half the overall computational cost of the BCGS process (see Algorithm 5), with half the number of passes over high-dimensional objects [8].
The preprocessing step in line 1 of the intra-orthogonalization strategy in Algorithm 15 should involve an unconditionally stable QR factorization that is suitable for parallel execution. The TSQR algorithm discussed in Section 2.1.4 meets these requirements. Including the sketching step in line 2 of Algorithm 15 and the sketching steps in lines 4 and 8 of Algorithm 12, this results in a total of four global synchronizations in parallel execution. The explicit strategy in Algorithm 13 reduces the number of global synchronizations to three, while the implicit strategy in Algorithm 14 reduces it further to two. Independent of the intra-orthogonalization strategy used, an additional global synchronization is avoided by postponing the sketching step in line 8 of Algorithm 12 to the next iteration, where it is combined with the sketching step in line 4 of Algorithm 12. When using this modification in combination with the implicit strategy, Algorithm 12 requires only one global synchronization per iteration.

5.1.2 *s*-step RBGS-Arnoldi

The *s*-step RBGS-Arnoldi method presented in Algorithm 17 inherits all the stability guarantees and computational benefits of the RBGS process [8], provided that the respective subspace embedding properties hold. Like the deterministic *s*-step Arnoldi method presented in [45], the *s*-step RBGS-Arnoldi method must reconstruct the upper Hessenberg matrix. Although the reconstruction process is simpler than that described in [45], it is also more expensive.

Reconstructing the upper Hessenberg matrix

The upper Hessenberg matrix $\mathbf{H}$ in line 15 of Algorithm 17 is obtained by solving a least-squares problem that requires the sketch $\mathbf{\Theta A Q}_{\bar{b}}$, where $\bar{b}$ is defined as in line 14 of Algorithm 17. This sketch is computed either explicitly or implicitly.

Explicitly sketching $\mathbf{AQ}_{\bar{b}}$ The explicit approach 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 17 and subsequently sketched along with $\mathbf{V}_b$ in line 9 of Algorithm 17 [8]. Even with these optimizations, we observed that the explicit approach is very costly in practice, as it requires additional operations on high-dimensional objects, potentially doubling both the cost of the MPK and the cost of the sketching step involving $\mathbf{V}_b$.

Implicitly sketching $\mathbf{AQ}_{\bar{b}}$ The implicit strategy 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

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

where $\hat{b} = i+1 : i+s-1$, $\mathbf{s}_i = \mathbf{\Theta q}_i$, $\mathbf{P}_b = \mathbf{\Theta V}_b$, and $\mathbf{\Theta Aq}_i$ 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 $\mathbf{\Theta A Q}_{\hat{b}}$ in $\mathbf{\Theta A Q}_{\bar{b}}$ are obtained. The cost of updating the Hessenberg matrix in line 15 of Algorithm 17 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 in lines 11 and 29 of Algorithm 8 is performed in $\mathcal{O}\left(i^3\right)$ time, or in $\mathcal{O}\left(is^2\right)$ time using the optimized procedure described in [45]. Thus, the reconstruction cost for the randomized approach is higher compared to the deterministic method, since $i < d$.

Algorithm 17 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: $\mathbf{p}_1 = \mathbf{\Theta v}_1$
2: $r_{1,1} = \|\mathbf{p}_1\|$
3: $\mathbf{q}_1 = \mathbf{v}_1 / r_{1,1}$
4: $\mathbf{s}_1 = \mathbf{\Theta q}_1$
5: **for** $j = 1 : p$ **do**
6: $\quad i = s(j-1)+1$
7: $\quad b = i+1 : i+s$
8: $\quad [\mathbf{V}_b, \mathbf{B}] = \texttt{MPK}(\mathbf{A}, \mathbf{q}_i, s)$
9: $\quad \mathbf{P}_b = \mathbf{\Theta V}_b$
10: $\quad \mathbf{R}_{1:i,b} = \operatorname{argmin}_{\mathbf{Y}} \|\mathbf{S}_{1:i}\mathbf{Y} - \mathbf{P}_b\|_{\mathrm{F}}$
11: $\quad \mathbf{Q}'_b = \mathbf{V}_b - \mathbf{Q}_{1:i}\mathbf{R}_{1:i,b}$
12: $\quad$ Apply randomized QR factorization strategy to $\mathbf{Q}'_b$ to obtain $\mathbf{Q}_b\mathbf{R}_{b,b}$
13: $\quad \mathbf{S}_b = \mathbf{\Theta Q}_b$
14: $\quad \bar{b} = i : i+s-1$
15: $\quad \mathbf{H}_{1:i+s,\bar{b}} = \operatorname{argmin}_{\mathbf{Y}} \|\mathbf{S}_{1:i+s}\mathbf{Y} - \mathbf{\Theta A Q}_{\bar{b}}\|_{\mathrm{F}}$
16: **end for**
17: Optional: check $\Delta_m = \|\mathbf{I} - \mathbf{S}^{\mathrm{T}}\mathbf{S}\|_{\mathrm{F}}$ and $\tilde{\Delta}_m = \|\mathbf{P} - \mathbf{SR}\|_{\mathrm{F}} / \|\mathbf{P}\|_{\mathrm{F}}$

5.1.3 RBGS-GMRES

RBGS-GMRES [8] is based on the s-step RBGS-Arnoldi process. Algorithm 18 illustrates RBGS-GMRES with a single global synchronization every s steps (line 13 of Algorithm 18) and implicit reconstruction of the Hessenberg matrix $\mathbf{H}$ (lines 21–27 of Algorithm 18). Like the BCGSI2-GMRES method (see Algorithm 8), Algorithm 18 adjusts its step size adaptively, using the ICE to detect and discard low-quality basis vectors (line 18 of Algorithm 18). The algorithm supports three optimizations:

1. The QR factorization in line 17 of Algorithm 18 computes the R factor in its entirety before it is evaluated by the ICE in line 18. Assuming that a HH-QR factorization or any other suitably stable routine is used, integrating the ICE into the factorization process allows for efficient partial factorization, similar to the partial CholQR factorization outlined in Algorithm 7.

2. The diagonal blocks of $\mathbf{H}$ in line 27 of Algorithm 18 are computed from implicitly sketched Krylov basis vectors $\hat{\mathbf{S}}_{b_c}$ that are generated in line 17 of Algorithm 18. To improve the numerical quality of $\mathbf{H}$, these blocks can be recomputed in subsequent iterations using the explicitly sketched basis vectors in line 13 of Algorithm 18.

3. Sketching the last s columns of $\mathbf{Q}$ in line 34 of Algorithm 18 is only necessary for stability verification in line 35 of Algorithm 18 and can be skipped

otherwise. Stability verification may be important when using less stable inter-orthogonalization methods, such as Richardson iterations in line 14 of Algorithm 18, instead of direct solvers. It can also help to identify sources of instability when the method does not behave as expected.

In numerical experiments, we found that a condition number tolerance of $\Omega_{\mathbf{P}} = \mathcal{O}\left(10^{-1}u^{-1/2}\right)$, similar to that used in BCGSI2-GMRES, or $\Omega_{\mathbf{P}} = \mathcal{O}\left(10^{-2}u^{-1/2}\right)$ for challenging cases works well in practice. The bound $\Omega_{\mathbf{P}}$ further ensures that the inverse of the diagonal block of the R factor is safely applied via forward substitution during intra-orthogonalization and during reconstruction of the upper Hessenberg matrix (see lines 20 and 25 of Algorithm 18 respectively). Lastly, it can be deduced that the residual bound (4.16) for RGS-GMRES also applies to RBGS-GMRES.

5.2 Randomized & sketched s-step GMRES

Section 5.2 introduces our novel s-step GMRES method, which combines a randomized truncated block Gram-Schmidt (RTBGS) process with sketched least-squares problems.

5.2.1 Randomized Truncated Block Gram-Schmidt (RTBGS)

The randomized and truncated block Gram-Schmidt (RTBGS) process is similar to RBGS (Algorithm 12) but performs partial orthogonalization by projecting new basis vectors against the t most recently orthogonalized vectors. Specifically, RTBGS replaces lines 5 and 6 of the RBGS process in Algorithm 12 with the lines

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

where $w = \max(1, i - t + 1)$. The optional stability check in line 10 of Algorithm 12 is not suitable for RTBGS, since Δ_m is likely not sufficiently small. Note that in this case, equations (4.11)–(4.13) cannot be derived for RTBGS because the necessary assumptions are not met.

5.2.2 RTBGS-GMRES

Similar to sGMRES(m) (see Algorithm 11), our novel RTBGS-GMRES method solves the sketched least-squares problem (4.17) and therefore does not require the sketched Arnoldi relation (4.15), unlike RBGS-GMRES (see line 32 of Algorithm 18). Solving (4.17) requires the mild condition $\kappa(\mathbf{AQ}_{1:i}) \lesssim u^{-1}$. As a result, RTBGS-GMRES can efficiently generate a possibly poorly conditioned Krylov basis using randomized truncated orthogonalization. Adaptive RTBGS-GMRES with a single global synchronization and an implicitly sketched basis $\mathbf{AQ}_{1:i}$ is presented in Algorithm 19.
In [41], a similar (unblocked) method, called the *sketch-and-select* Arnoldi process, performs randomized truncated Gram-Schmidt orthogonalization. Instead of projecting against the t most recently orthogonalized vectors, this process analyzes the sketched basis to select a specific set of t previously orthonormalized vectors that are likely to minimize the growth of the basis condition number. We leave the possible integration of this technique into RTBGS for future work.

Algorithm 18 Adaptive single-sync RBGS-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 $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, $d \ll n$, condition number tolerance $\Omega_{\mathbf{P}}$
Output: Approximate solution to $\mathbf{Ax} = \mathbf{b}$

1: **for** $k = 0, 1, \ldots,$ until convergence **do**
2: $\quad \mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$
3: $\quad \mathbf{v}_1 = \mathbf{r}^{(k)}$
4: $\quad \mathbf{p}_1 = \mathbf{\Theta}\mathbf{v}_1$
5: $\quad r_{1,1} = \|\mathbf{p}_1\|$
6: $\quad \mathbf{q}_1 = \mathbf{v}_1 / r_{1,1}$
7: $\quad b_p = 1$ ▷ Previous block indices
8: $\quad i = 1$
9: $\quad$ **while** $i \leq m$ **do**
10: $\quad\quad$ **if** $i + s > m + 1$ **then** $s = m + 1 - i$
11: $\quad\quad b_c = i + 1 : i + s$ ▷ Current block indices
12: $\quad\quad [\mathbf{V}_{b_c}, \mathbf{B}] = \texttt{MPK}(\mathbf{A}, \mathbf{q}_i, s)$
13: $\quad\quad [\mathbf{S}_{b_p}, \mathbf{P}_{b_c}] = \mathbf{\Theta}[\mathbf{Q}_{b_p}, \mathbf{V}_{b_c}]$ ▷ Global sync
14: $\quad\quad \mathbf{R}_{1:i,b_c} = \operatorname{argmin}_{\mathbf{Y}} \|\mathbf{S}_{1:i}\mathbf{Y} - \mathbf{P}_{b_c}\|_{\mathrm{F}}$
15: $\quad\quad \mathbf{Q}'_{b_c} = \mathbf{V}_{b_c} - \mathbf{Q}_{1:i}\mathbf{R}_{1:i,b_c}$
16: $\quad\quad \mathbf{S}'_{b_c} = \mathbf{P}_{b_c} - \mathbf{S}_{1:i}\mathbf{R}_{1:i,b_c}$
17: $\quad\quad$ Compute thin QR factorization $\mathbf{S}'_{b_c} = \hat{\mathbf{S}}_{b_c}\mathbf{R}_{b_c,b_c}$
18: $\quad\quad s = \texttt{ICE}(\mathbf{R}_{b_c,b_c}, \Omega_{\mathbf{P}})$
19: $\quad\quad b_c = i + 1 : i + s$
20: $\quad\quad \mathbf{Q}_{b_c} = \mathbf{Q}'_{b_c}\mathbf{R}^{-1}_{b_c,b_c}$ via triangular solve
21: $\quad\quad \hat{b}_c = i + 1 : i + s - 1$
22: $\quad\quad \bar{b}_c = i : i + s - 1$
23: $\quad\quad [\mathbf{c}_i, \hat{\mathbf{C}}_{\hat{b}_c}] = [\mathbf{s}_i, \mathbf{P}_{b_c}]\mathbf{B}_{1:s+1,1:s}$
24: $\quad\quad \mathbf{C}'_{\hat{b}_c} = \hat{\mathbf{C}}_{\hat{b}_c} - \mathbf{C}_{1:i}\mathbf{R}_{1:i,\hat{b}_c}$
25: $\quad\quad \mathbf{C}_{\hat{b}_c} = \mathbf{C}'_{\hat{b}_c}\mathbf{R}^{-1}_{\hat{b}_c,\hat{b}_c}$ via triangular solve
26: $\quad\quad \mathbf{H}_{1:i,\bar{b}_c} = \operatorname{argmin}_{\mathbf{Y}} \|\mathbf{S}_{1:i}\mathbf{Y} - \mathbf{C}_{\bar{b}_c}\|_{\mathrm{F}}$
27: $\quad\quad \mathbf{H}_{b_c,\bar{b}_c} = \operatorname{argmin}_{\mathbf{Y}} \|\hat{\mathbf{S}}_{b_c}\mathbf{Y} - \mathbf{C}_{\bar{b}_c}\|_{\mathrm{F}}$ } Update $\mathbf{H}$
28: $\quad\quad b_p = b_c$
29: $\quad\quad i = i + s$
30: $\quad\quad$ **if** converged **then break**
31: $\quad$ **end while**
32: $\quad \mathbf{y}^{(k)} = \operatorname{argmin}_{\mathbf{y}} \|\|\mathbf{r}^{(k)}\|\mathbf{e}_1 - \mathbf{H}_{1:i,1:i-1}\mathbf{y}\|$
33: $\quad \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{Q}_{1:i}\mathbf{y}^{(k)}$
34: $\quad$ Optional: $\mathbf{S}_{b_c} = \mathbf{\Theta}\mathbf{Q}_{b_c}$
35: $\quad$ Optional: check $\Delta_m = \|\mathbf{I} - \mathbf{S}^{\mathrm{T}}\mathbf{S}\|_{\mathrm{F}}$ and $\tilde{\Delta}_m = \|\mathbf{P} - \mathbf{SR}\|_{\mathrm{F}}/\|\mathbf{P}\|_{\mathrm{F}}$
36: **end for**

Stability analysis

Using (4.20), if $\mathbf{\Theta}$ is a subspace embedding for $\mathbf{AQ}_{1:i}$, then $\kappa(\mathbf{AQ}_{1:i}) \lesssim u^{-1}$, provided that $\kappa(\mathbf{\Theta AQ}_{1:i}) \lesssim u^{-1}$. The ICE inexpensively monitors $\kappa(\mathbf{\Theta AQ}_{1:i})$ with a condition number tolerance $\Omega_{\mathbf{C}} = \mathcal{O}\left(u^{-1}\right)$ in line 29 of Algorithm 19. Once $\kappa(\mathbf{\Theta AQ}_{1:i}) > \Omega_{\mathbf{C}}$, we choose to restart Algorithm 19 in line 30, although other strategies, such as basis whitening, might be used (see the performance analysis in Section 4.3). To account for numerical instability in the basis vectors generated by the MPK, RTBGS-GMRES

can also apply the ICE to the factorization of the implicitly sketched basis vectors in line 20 of Algorithm 19. In general, we recommend using the same condition number tolerance $\Omega_{\mathbf{P}}$ as with RBGS-GMRES.
Post-scaling basis vectors generated by the MPK in line 14 of Algorithm 19 potentially improves the condition of the R factor in line 19, leading to larger admissible step sizes from the ICE in line 20. Let $\mathbf{F} \in \mathbb{R}^{(s+1)\times(s+1)}$ be a diagonal matrix whose diagonal elements are the reciprocal column norms of $[\mathbf{s}_i, \mathbf{P}_{b_c}]$. Then post-scaling involves multiplying $\mathbf{V}_{b_c}$ and $\mathbf{P}_{b_c}$ by $\mathbf{F}_{2:s,2:s}$ and using

$$\mathbf{A}[\mathbf{q}_i, \mathbf{V}_{\hat{b}_c}]\mathbf{F}_{1:s,1:s} = [\mathbf{q}_i, \mathbf{V}_{b_c}]\mathbf{FB}$$

to compute a new change-of-basis matrix $\mathbf{FBF}^{-1}_{1:s,1:s}$.

Performance analysis

We compare the performance of our RTBGS-GMRES algorithm with a potential s-step sGMRES(m) variant. The s-step sGMRES(m) method would require at least three global synchronizations every s steps, assuming that the algorithm performs truncated inter-orthogonalization, intra-orthogonalization, and randomized sketching, similar to the single-step method (lines 8, 9, and 11 of Algorithm 11 respectively). In contrast, RTBGS-GMRES requires only one global synchronization every s steps. Specifically, high-dimensional vectors are sketched together in line 13 of Algorithm 19, and $\mathbf{AQ}_{\bar{b}}$ is implicitly sketched, as described in Section 5.1.2, which does not require any communication. Furthermore, truncated orthogonalization significantly reduces the projection cost. Therefore, RTBGS-GMRES is particularly suitable for small step sizes, as inter-orthogonalization would otherwise be a dominant cost in this case.
Note that, just like the scaled Newton basis described in Section 3.1.3, post-scaling the basis in line 14 of Algorithm 19 does not require any communication.
Algorithm 19 further allows for some minor performance optimizations. The QR factorization in line 28 of Algorithm 19 should be updated as described in Section 2.1.3. Furthermore, the ICE in line 29 of Algorithm 19 can be integrated into the QR factorization in line 28, where it processes the new columns of $\hat{\mathbf{R}}_{1:i-1,1:i-1}$ as they arrive.

Algorithm 19 Adaptive single-sync 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 $\mathbf{\Theta} \in \mathbb{R}^{d\times n}$, $d \ll n$, condition number tolerances $\Omega_{\mathbf{C}}$ and $\Omega_{\mathbf{P}}$, truncation parameter t

Output: Approximate solution to $\mathbf{Ax} = \mathbf{b}$

for $k = 0, 1, \ldots,$ until convergence **do**
  $\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$
  $\mathbf{v}_1 = \mathbf{r}^{(k)}$
  $\mathbf{p}_1 = \mathbf{\Theta}\mathbf{v}_1$
  $r_{1,1} = \|\mathbf{p}_1\|$
  $\mathbf{q}_1 = \mathbf{v}_1 / r_{1,1}$
  $b_p = 1$ ▷ Previous block indices
  $i = 1$
  while $i \leq m$ **do**
    if $i + s > m + 1$ **then** $s = m + 1 - i$
    $b_c = i + 1 : i + s$ ▷ Current block indices
    $[\mathbf{V}_{b_c}, \mathbf{B}] = \texttt{MPK}(\mathbf{A}, \mathbf{q}_i, s)$
    $[\mathbf{S}_{b_p}, \mathbf{P}_{b_c}] = \mathbf{\Theta}[\mathbf{Q}_{b_p}, \mathbf{V}_{b_c}]$ ▷ Global sync
    Post-scale $\mathbf{P}_{b_c}$ and $\mathbf{V}_{b_c}$, fix change-of-basis matrix $\mathbf{B}$ ▷ Optional
    $b_t = \max(1, i - t + 1) : i$ ▷ Compute truncation indices
    $\mathbf{R}_{b_t,b_c} = \operatorname{argmin}_{\mathbf{Y}} \|\mathbf{S}_{b_t}\mathbf{Y} - \mathbf{P}_{b_c}\|_{\mathrm{F}}$
    $\mathbf{Q}'_{b_c} = \mathbf{V}_{b_c} - \mathbf{Q}_{b_t}\mathbf{R}_{b_t,b_c}$
    $\mathbf{S}'_{b_c} = \mathbf{P}_{b_c} - \mathbf{S}_{b_t}\mathbf{R}_{b_t,b_c}$
    Compute $\mathbf{R}_{b_c,b_c}$ as the R factor of the thin QR factorization of $\mathbf{S}'_{b_c}$
    $s = \texttt{ICE}(\mathbf{R}_{b_c,b_c}, \Omega_{\mathbf{P}})$
    $b_c = i + 1 : i + s$
    $\mathbf{Q}_{b_c} = \mathbf{Q}'_{b_c}\mathbf{R}^{-1}_{b_c,b_c}$ via triangular solve
    $\hat{b}_c = i + 1 : i + s - 1$
    $[\mathbf{c}_i, \hat{\mathbf{C}}_{\hat{b}_c}] = [\mathbf{s}_i, \mathbf{P}_{b_c}]\mathbf{B}_{1:s+1,1:s}$
    $\mathbf{C}'_{\hat{b}_c} = \hat{\mathbf{C}}_{\hat{b}_c} - \mathbf{C}_{b_t}\mathbf{R}_{b_t,\hat{b}_c}$
    $\mathbf{C}_{\hat{b}_c} = \mathbf{C}'_{\hat{b}_c}\mathbf{R}^{-1}_{\hat{b}_c,\hat{b}_c}$ via triangular solve
    } $\mathbf{C}_{\bar{b}_c} = \mathbf{\Theta}\mathbf{A}\mathbf{Q}_{\bar{b}_c}$, with $\bar{b}_c = i : i + s - 1$
    $i = i + s$
    Update thin QR factorization $\mathbf{C}_{1:i-1} = \hat{\mathbf{Q}}_{1:i-1}\hat{\mathbf{R}}_{1:i-1,1:i-1}$
    $\hat{i} = \texttt{ICE}(\hat{\mathbf{R}}_{1:i-1,1:i-1}, \Omega_{\mathbf{C}})$
    if $\hat{i} < i - 1$ **then** $i = \hat{i} + 1$, **break** ▷ Restart
    $\hat{r} = \|(\mathbf{I} - \hat{\mathbf{Q}}_{1:i-1}\hat{\mathbf{Q}}^{\mathrm{T}}_{1:i-1})\mathbf{p}_1\|$ ▷ Residual estimate
    if converged **then break**
    $b_p = b_c$
  end while
  $\mathbf{y}^{(k)} = \hat{\mathbf{R}}^{-1}_{1:i-1,1:i-1}(\hat{\mathbf{Q}}^{\mathrm{T}}_{1:i-1}\mathbf{p}_1)$ via triangular solve
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{Q}_{1:i-1}\mathbf{y}^{(k)}$
end for

Chapter 6

Numerical experiments

This chapter presents numerical experiments for both deterministic and randomized s-step GMRES methods. Section 6.1 focuses on the stability of the underlying orthogonalization methods, while Section 6.2 discusses the implementation details of GMRES and provides results on performance and parallel scalability.

6.1 Stability of orthogonalization methods

Section 6.1 experimentally compares the stability of the BCGSI2 process (Algorithm 6) with other BCGS-based methods and variants of the RBGS process (Algorithm 12). The experiments were conducted in Matlab using the framework in [20] for testing the numerical stability of block orthogonalization methods. We consider three types of basis matrices that are relevant for s-step methods. The column block matrix $\mathbf{V} = [\mathbf{V}_1, \ldots, \mathbf{V}_p] \in \mathbb{R}^{n \times ps}$, where $n = 10000$, $p = 50$ (number of blocks), and $s = 10$ (number of columns in a block) is generated in one of the following ways:

- `monomial`: applies a diagonal matrix $\mathbf{A}$ with uniformly distributed eigenvalues in $(0.1, 10)$ to p random and normalized vectors $\mathbf{w}_i$, for $i = 1, \ldots, p$ such that

$$\mathbf{V}_i = \left[\mathbf{w}_i, \mathbf{A}\mathbf{w}_i, \ldots, \mathbf{A}^{s-1}\mathbf{w}_i\right]$$

- `s-step`: like `monomial`, but $\mathbf{w}_1$ is random with unit length, and

$$\begin{aligned} \mathbf{w}_i' &= \mathbf{A}^{s-1}\mathbf{w}_{i-1}, \\ \mathbf{w}_i &= \mathbf{w}_i'/\|\mathbf{w}_i'\|, \quad \text{for } i = 2, \ldots, p. \end{aligned}$$

  This matrix corresponds to a monomial basis, obtained from an s-step GMRES method.

- `newton`: like `s-step` but this matrix corresponds to a Newton basis, obtained from an s-step GMRES method.

Table 6.1 lists the properties of the generated basis types, indicating that the `s-step` basis is numerically rank-deficient, which makes it particularly challenging.

TABLE 6.1: Minimum and maximum singular values and condition number of different basis types.

basis type	$\sigma_{\max}(\mathbf{V})$	$\sigma_{\min}(\mathbf{V})$	$\kappa(\mathbf{V})$
`monomial`	2.32e+08	3.06e-04	7.60e+11
`s-step`	2.09e+01	2.42e-17	8.66e+17
`newton`	3.13e+00	2.61e-03	1.20e+03

6.1.1 Stability of BCGSI2 vs. other BCGS methods

Figures 6.1–6.3 illustrate the "loss of orthogonality" defined in (2.1) and the relative residual in (2.2) for the following methods: BCGS (see Algorithm 5), BCGSI2[1], BCGSI2 with three modifications that each eliminate a global synchronization (i.e., BCGSI2-SS$_{1-3}$; see Appendix A for details), the low-synchronization variant BCGSI2-LS from [20], and two BCGS methods utilizing the Pythagorean inner product (PIP), as described in [19]. The y-axis in Figures 6.1–6.3 lists various intra-orthogonalization methods. Refer to [20] for a detailed description of these methods. BCGSI2-SS$_{1-3}$, BCGSI2-LS, and both PIP-based BCGS methods, as well as the intra-orthogonalization methods CholQR, CholQR2, and ShCholQR3 perform Cholesky factorizations. A `NaN` value indicates a Cholesky factorization failure [20].

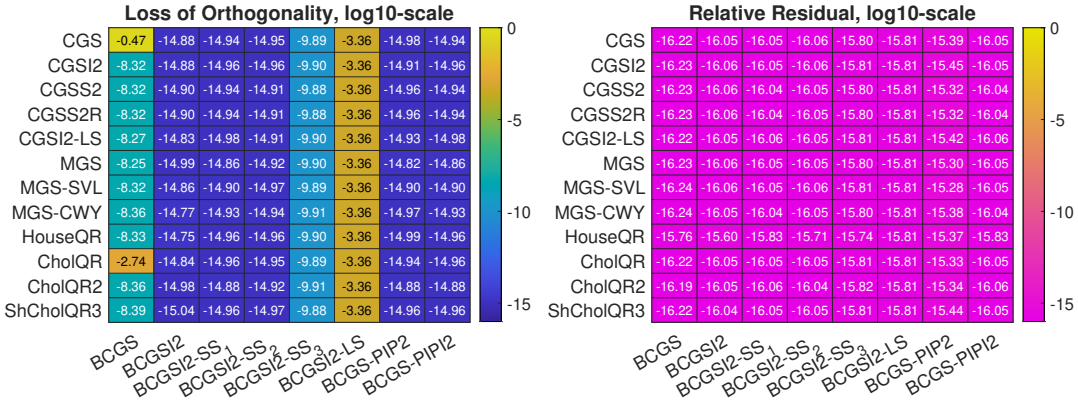

FIGURE 6.1: Measurements for the `monomial` basis

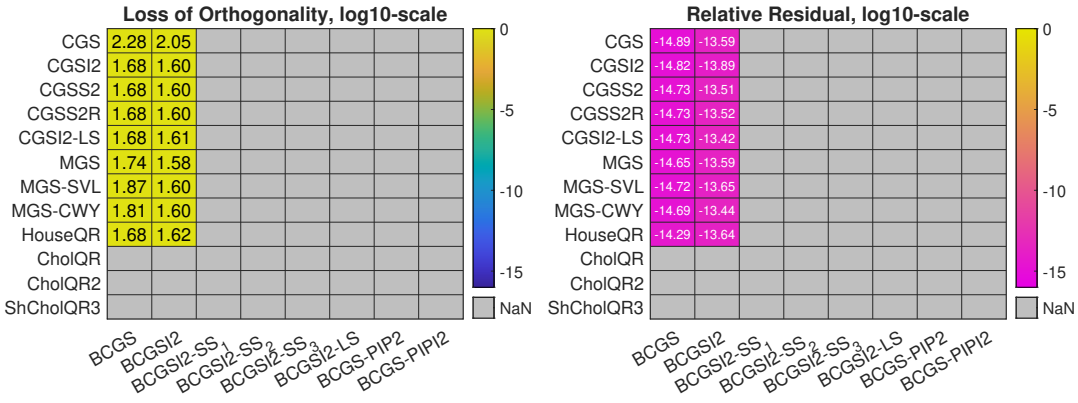

FIGURE 6.2: Measurements for the `s-step` basis

Although none of the BCGS methods achieve high-quality orthonormalization of the `s-step` basis in Figure 6.2, BCGSI2 shows excellent stability for the `monomial` basis in Figure 6.1, with competitive results from the PIP variants and BCGSI2-SS$_{1-2}$. Note that for challenging cases such as the `s-step` basis, using smaller step sizes would improve numerical stability – a strategy employed by adaptive s-step GMRES methods.

[1] Carson et al. [20] refer to this method as BCGSI+1.

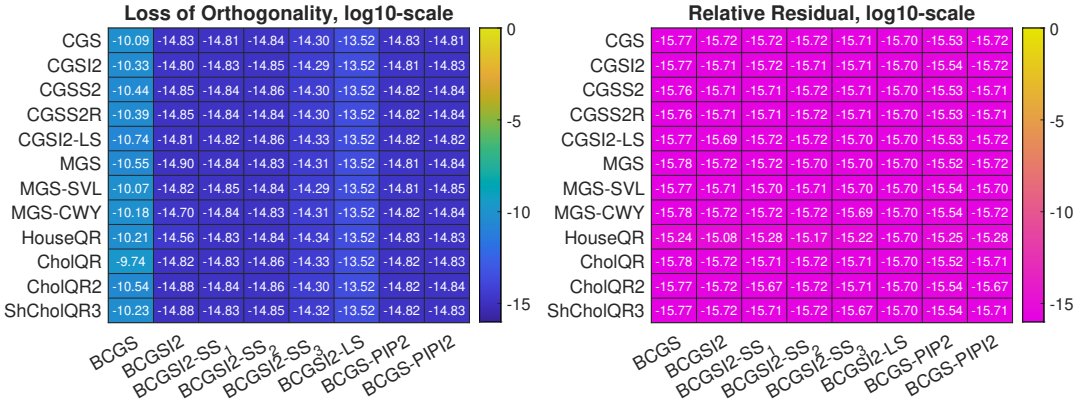

FIGURE 6.3: Measurements for the `newton` basis

6.1.2 Stability of BCGSI2 vs. RBGS methods

We compare the stability of BCGSI2 and RBGS-based methods by analyzing the condition number of the orthonormalized basis. The methods in Figures 6.4–6.5 are as follows: BCGSI2, RBGS (Algorithm 12 using the explicit strategy of Algorithm 13), RBGS-OS (Algorithm 12 using the implicit strategy of Algorithm 14), RBGS2 (RBGS, run twice), RBGS-STAB (Algorithm 12 using the explicit strategy with a preprocessing step described in Algorithm 15), RBGS2-STAB-OS (performs a round of RBGS-STAB, and a second round of RBGS-OS). Both figures show results for the `s-step` basis. Each figure features randomized methods with a different sketch dimension and strategy for inter-orthogonalizing the sketched basis vectors. Note that the randomized methods use randomized intra-orthogonalization. The intra-orthogonalization methods indicated on the y-axis of Figures 6.4–6.5 are only relevant for BCGSI2 and the deterministic preprocessing step of Algorithm 15, used by the RBGS-STAB methods. The variability in the results for the other randomized methods is caused by using a different subspace embedding in each run. The embedding type is a rescaled Gaussian matrix in all cases. Similar to Figures 6.1–6.3, a `NaN` value represents a failed Cholesky factorization.

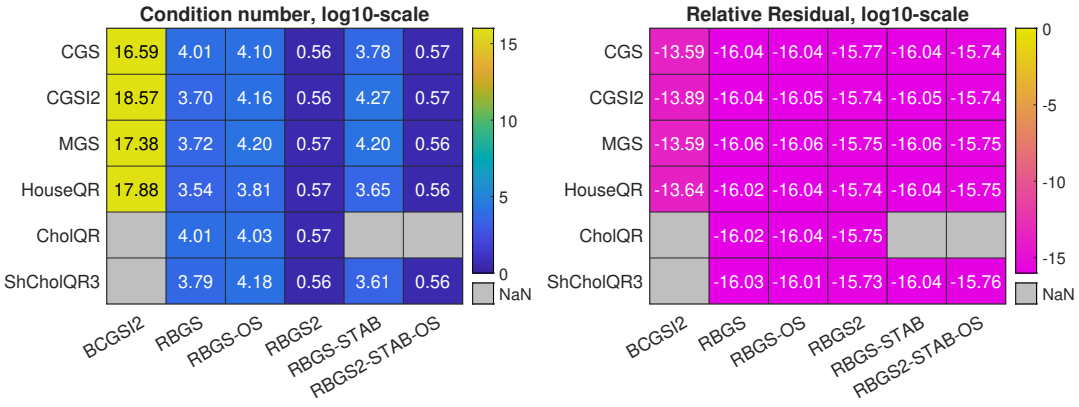

FIGURE 6.4: `s-step` with sketch dimension $d = 1500$ and HH-QR solver in RBGS.

Figure 6.4 shows that RBGS-OS achieves a significantly better condition number compared to BCGSI2, as long as an unconditionally stable QR factorization (HH-QR in this case) is used during inter-orthogonalization in line 5 of Algorithm 12. Figure 6.5 shows that using Richardson iterations instead reduces the stability of the randomized methods without reorthogonalization. For the other basis types (`monomial` and

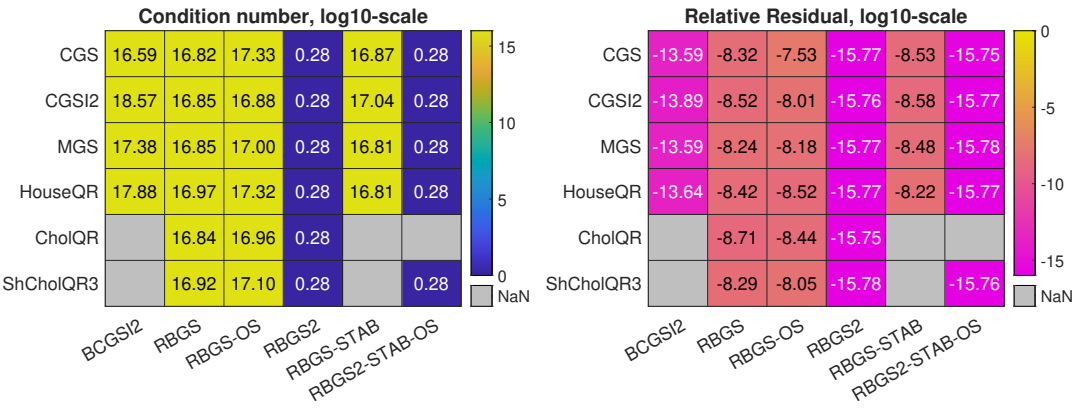

FIGURE 6.5: `s-step` with sketch dimension $d = 5000$ and 5 Richardson iterations in RBGS.

`newton`) all methods exhibit excellent stability (see Appendix B for these results).

6.2 Parallel numerical experiments

Section 6.2 shows parallel numerical experiments for BCGSI2-GMRES (Algorithm 8), RBGS-GMRES (Algorithm 18), and RTBGS-GMRES (Algorithm 19) using both the monomial basis and the scaled Newton basis.
The experiments were conducted on the Vienna Scientific Cluster 5 (VSC5), 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. The test matrices presented in Table 6.2 are taken from the SuiteSparse Matrix Collection [27], except for test matrix `e200`, which is taken from the Matrix Market [16].

TABLE 6.2: 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	sym	`np`	`prec`	`cTol`	m
`e200`	`e20r5000`	4 241	31.0	no	1	`ILU(8)`	10^{-14}	60
`mat3`	`matrix-new_3`	125 329	7.1	no	4	`ILUT(1.)`	10^{-14}	200
`xen2`	`xenon2`	157 464	24.6	no	4	`COLEQUI`	10^{-14}	300
`radn`	`radiation`	223 104	24.8	no	8	`COLEQUI`	10^{-12}	200
`af_3`	`af_shell3`	504 855	34.8	yes	16	`COLEQUI`	10^{-7}	300
`atml`	`atmosmodl`	1 489 752	6.9	no	64	`COLEQUI`	10^{-14}	300
`ML_r`	`ML_Geer`	1 504 002	73.6	no	64	`COLEQUI`	10^{-6}	400

6.2.1 Implementation notes

The algorithms were implemented using the Trilinos 14.0 C++ library [43], 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 [12], a framework for iterative solvers of large, sparse linear systems. From Trilinos, we used the reference implementation of GMRES, the preconditioners, and the matrix equilibration algorithm. We implemented the s-step methods, the ICE, the step

size estimator, the modified Leja ordering, and the four randomized subspace embeddings: CountSketch, CountGauss, BSRHT, and Gaussian embedding. Additional implementation details are listed below.

- **Convergence metric:** 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.

- **Start vector and exact solution:** The initial start vector is a zero vector. Similar to [45], the right-hand-side $\mathbf{b}$ of the linear system $\mathbf{Ax} = \mathbf{b}$ is defined such that the i^{th} element of the solution vector $\mathbf{x}$ is given by

$$x_i = z_i + \sin(2\pi i/n),$$

  where z_i is randomly chosen from a uniform distribution over the interval $[-1, 1]$.

- **Restart length:** Restart lengths $m \leq 60$ for GMRES(m) are commonly encountered in the literature. See, for example, [5, 29]. In our numerical experiments with s-step methods, we observed that restart lengths $200 \leq m \leq 400$ significantly improve runtime performance due to faster convergence.

- **ICE condition number tolerance:** In all s-step methods, the ICE (see Section 3.3.1) was used to estimate 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.

- **Randomized subspace embeddings:** The BSRHT implementation uses the Subsampled Randomized Discrete Hartley Transform (SRDHT) from the FFTW library [34], as it exhibits similar behavior and performance to the SRHT in practice [71]. In the CountGauss implementation, we replaced the Gaussian embedding with the SRDHT to improve performance.

- **Solving block least-squares problems in RBGS-GMRES:** For sketch dimensions $d \leq 1500$, block least-squares problems involving the matrix $\mathbf{S}$ (see lines 14 and 26 of Algorithm 18) 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.

- **RTBGS-GMRES truncation and restart strategy:** The RTBGS-GMRES truncation parameter in Algorithm 19 was set to $t = 8$ in all experiments. In line 28 of Algorithm 19, 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 $t_{new} = \max(2t, m)$.

- **Trilinos s-step GMRES:** We are aware of two undocumented deterministic s-step GMRES algorithms available in the Trilinos source code, which we did not use in our experiments. Furthermore, by evaluating the modified Leja ordering integrated in one of these methods, we found that the algorithm does not order

the Ritz values as specified in Algorithm 4. Thus, we implemented our own modified Leja ordering method, with two additional adaptions, as suggested in [45]. The first adaption scales the Ritz values to prevent over- and underflow in the computation of the product term in line 7 of Algorithm 4. The second modification addresses the issue of repeatedly occurring Ritz values (see [45] for details).

Ifpack2 preconditioners

We briefly describe the parallel preconditioners, `COLEQUI`, `ILU(k)`, and `ILUT(k)` from the Ifpack2 Trilinos package [60] listed in Table 6.2. All preconditioners were used as right preconditioners.

- **COLEQUI:** Column equilibration, where each column of $\mathbf{A}$ is divided by its column one-norm.
- **ILU(k):** Incomplete LU Factorization performed on the diagonal (local) blocks of the block-row distributed matrix $\mathbf{A}$, with $k \geq 0$.
- **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 $k \geq 1$.

6.2.2 Experiments with the monomial basis

We present performance results for application matrices from Table 6.2 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 [45, 65, 67].

Application matrices

Table 6.3 compares the randomized s-step methods with BCGSI2-GMRES in terms of number of iterations, runtime, and speedup. For BCGSI2-GMRES, we present the mean speedup of 20 runs over a single run of GMRES(m) (Algorithm 3), while for the randomized methods, the median speedup over BCGSI2-GMRES is reported. GMRES(m) uses CGS2 (Algorithm 1 with CGS2 projector) orthogonalization, which is known for its excellent stability and low communication overhead among non-blocked orthogonalization methods [37]. 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 6.3, 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.
The test matrix `e200` in Table 6.2 is much too small for efficient parallelization and is therefore not included in Table 6.3 (nor in Table 6.4). Figure 6.6 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`.

TABLE 6.3: 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)**

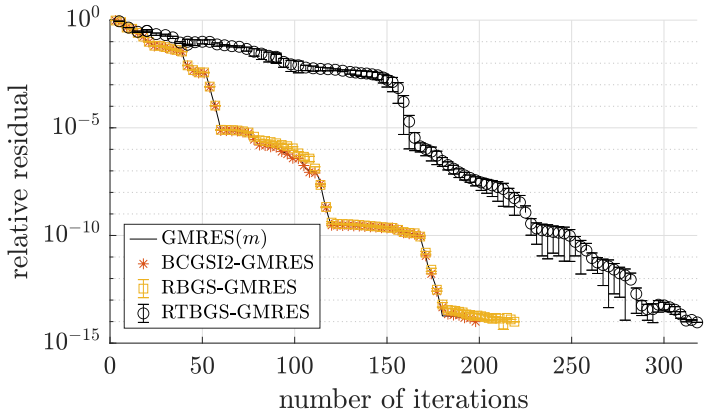

FIGURE 6.6: Convergence histories for `e200`. The randomized methods show residual means with 95% confidence intervals based on 20 runs. BCGSI2-GMRES used the ICE with a condition number bound of $\Omega = 10^{-1}u^{-1/2}$ for Krylov basis orthogonalization, while the randomized methods used $\Omega = 10^{-2}u^{-1/2}$. 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 restarts due to ill-conditioning for the other test cases in Table 6.2, 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 6.3 indicate greater variability in the performance of the randomized methods.

Strong and weak scaling experiments

We present strong scaling experiments for a 3D Laplace problem of size $n = 64$M (1M $= 10^6$) and restart length $m = 400$. The randomized methods used a CountSketch embedding of size $d = 1500$. Figure 6.7 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 (see lines 11 and 29 of Algorithm 8 and lines 21–27 of Algorithm 18, respectively). 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 23–28 of Algorithm 19). The speedup of RTBGS-GMRES mainly results from the reduced projection cost achieved through truncated orthogonalization. The speedup values corresponding to Figure 6.7 are shown in Figure 6.8. 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 6.2 are distributed across 23 277 to 39 366 rows per processor, corresponding to a parallelization degree of approximately 2^{11} processes in Figure 6.8.
Figure 6.9 presents weak scaling experiments for Laplace 3D 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.

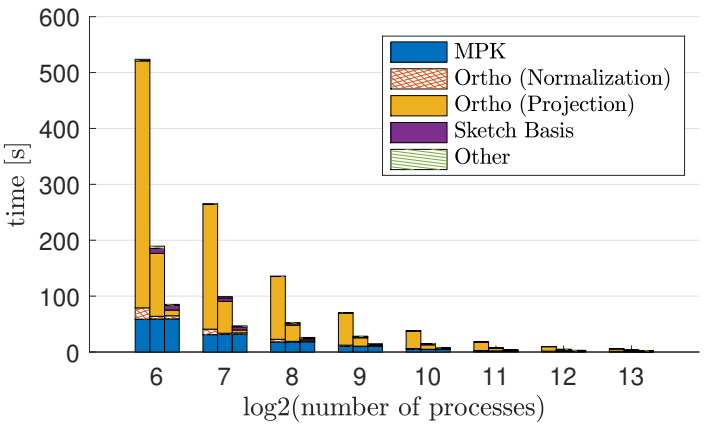

FIGURE 6.7: Mean runtime performance over 20 runs (with negligible variance in runtime) for strong scaling experiments with a 3D Laplace matrix of size 64M and convergence tolerance 10^{-7}. Algorithms in a bar group: BCGSI2-GMRES (left), RBGS-GMRES (middle), and RTBGS-GMRES (right).

6.2.3 Experiments with the scaled Newton basis

We present performance results for application matrices comparable to those in Section 6.2.2, but for the scaled Newton basis. Additionally, we report the performance of randomized s-step GMRES methods with different subspace embeddings of varying sizes. As defined in Section 3.3.2, the *optimal* step size is the largest admissible step size that avoids the computation of unused basis vectors.

Application matrices

Table 6.4 presents results for the scaled Newton basis. The randomized methods used the same sketch type and dimensions as those in Section 6.2.2. For each test case, 100 Ritz values were computed as a preprocessing step using Arnoldi's method. The

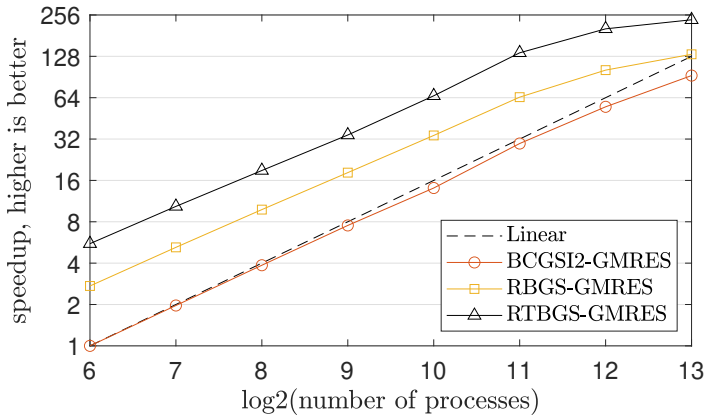

FIGURE 6.8: Mean speedup over 20 runs for strong scaling experiments with a 3D Laplace matrix of size 64M, using BCGSI2-GMRES with 2^6 processes as the baseline for all methods.

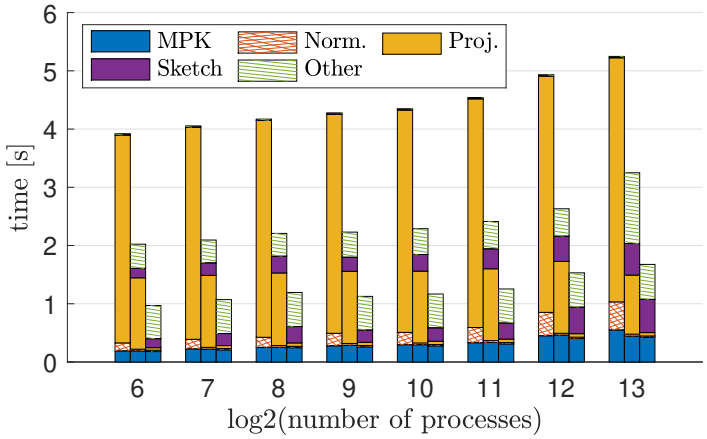

FIGURE 6.9: Mean runtime over 20 runs for weak scaling experiments with 3D Laplace matrices ranging from 1M to 128M (15625 rows per process). Algorithms and kernels are as in Figure 6.7. The s-step methods terminated when the relative residual improvement exceeded that of GMRES(m) at the 396^{th} iteration (which happens after approximately 400 iterations).

TABLE 6.4: $\hat{s}$ 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 6.3.

mat.	$\hat{s}$ / s_{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)**

initial step size $\hat{s}$ in Table 6.4 was obtained using the step size estimator described in Section 3.3.2. 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`.

Different subspace embeddings of varying sizes

Figures 6.10–6.12 illustrate the performance of randomized s-step GMRES methods with different subspace embeddings of varying sizes. The initial step sizes were estimated similarly to the strategy in Section 6.2.3. 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 6.10 and 6.11). Therefore, CountSketch embeddings are recommended, as their parallel application time is significantly faster compared to BSRHT and Gaussian/Rademacher sketches. In Figure 6.10, RTBGS-GMRES exhibits 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 (4.17) 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 6.12), as the residual improvement is only checked every s steps.

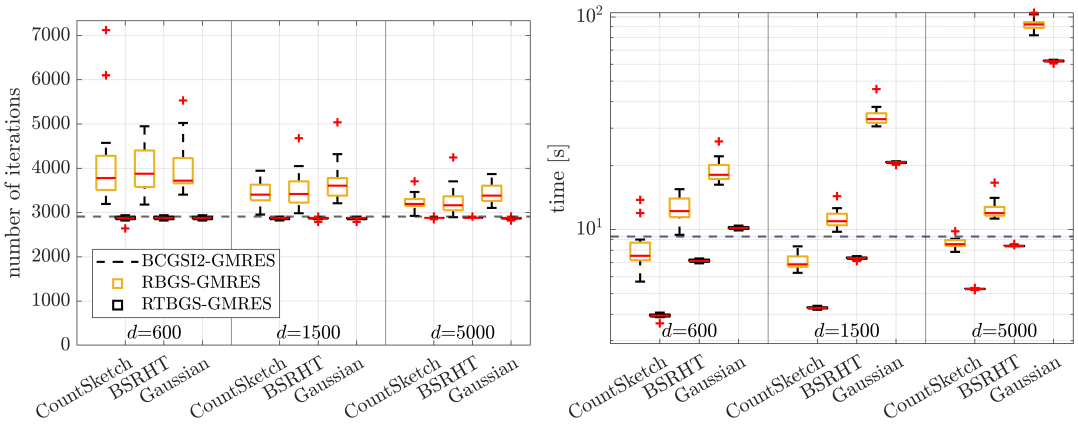

FIGURE 6.10: Performance of randomized s-step GMRES methods for test matrix `xen2` using various embedding types and sketch dimensions d, based on 20 runs. The legend in the left plot corresponds to both plots. For BCGSI2-GMRES, the mean runtime is shown as a dashed horizontal line. The variance in runtime 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 6.2. **Right:** Runtime performance (log scale).

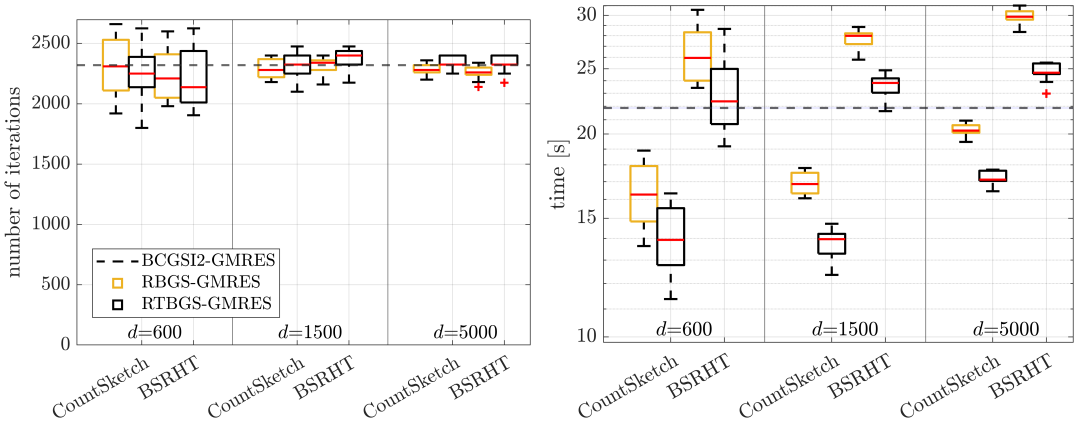

FIGURE 6.11: Performance of randomized s-step GMRES methods for matrix `af_3` (description as in Figure 6.10). The Gaussian embedding was not evaluated due to the high computational cost for $d = 5000$.

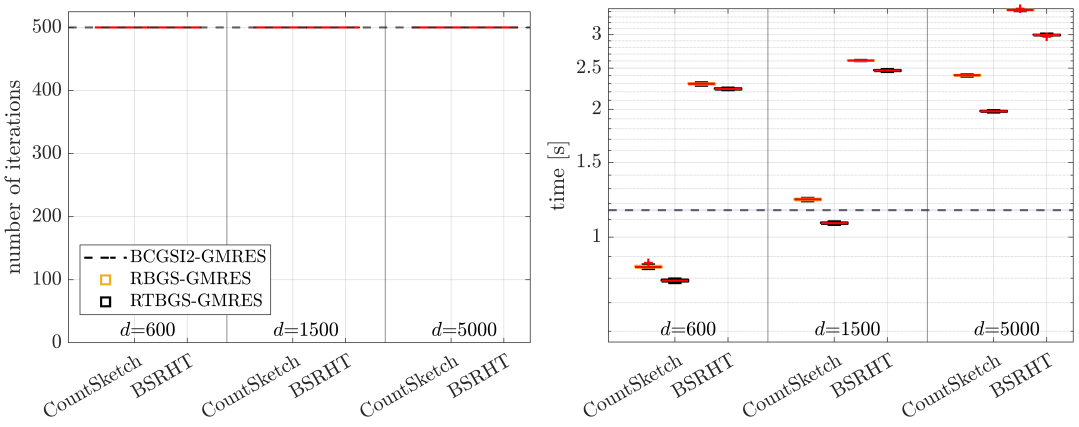

FIGURE 6.12: Performance of randomized s-step GMRES methods for matrix `atml` (description as in Figure 6.10). The optimal step size is $s = 100$ for all methods, leading to convergence within the same number of iterations. The Gaussian embedding was not evaluated due to the high computational cost for $d = 5000$.

Chapter 7

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 our novel RTBGS-GMRES algorithm, require only one global synchronization to advance s steps while maintaining sufficient numerical stability. In contrast, the deterministic method 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× using the monomial basis and up to 2.35× using the scaled Newton basis.

Appendix A: Save-Sync BCGSI2 (BCGSI2-SS)

Algorithms 20–22 present modifications to BCGSI2 (Algorithm 6), each eliminating one of its four global synchronizations. Carson et al. recently demonstrated in [18] that removing a global synchronization significantly affects stability, requiring $\mathcal{O}(u)\,\kappa^3(\mathbf{V}) \leq 1/2$ to achieve $\mathcal{O}(u)\,\kappa^2(\mathbf{V})$ loss of orthogonality, compared to the initial assumption $\mathcal{O}(u)\,\kappa^2(\mathbf{V}) \leq 1/2$ to achieve $\mathcal{O}(u)$ loss of orthogonality.

Algorithm 20 BCGSI2-SS$_1$ (replaces lines 4–5 of Algorithm 6)

// intra-ortho I with precomputed Gram matrix $\mathbf{G}$

Input: $\mathbf{V}_b$, $\mathbf{Q}_{1:i}$

Output: $\mathbf{R}^{(1)}_{1:i,b}$, $\mathbf{R}^{(1)}_{b,b}$, $\mathbf{Q}^{(1)}_b$

1: $[\mathbf{R}^{(1)}_{1:i,b}, \mathbf{G}'] = [\mathbf{Q}_{1:i}, \mathbf{V}_b]^{\mathrm{T}}\mathbf{V}_b$
2: $\mathbf{Q}'_b = \mathbf{V}_b - \mathbf{Q}_{1:i}\mathbf{R}^{(1)}_{1:i,b}$
3: $\mathbf{G} = \mathbf{G}' - \mathbf{R}^{(1)\,\mathrm{T}}_{1:i,b}\mathbf{R}^{(1)}_{1:i,b}$
4: $\mathbf{R}^{(1)}_{b,b} = \texttt{Cholesky}(\mathbf{G})$
5: $\mathbf{Q}^{(1)}_b = \mathbf{Q}'_b\mathbf{R}^{(1)\,-1}_{b,b}$ via triangular solve

Algorithm 21 BCGSI2-SS$_2$ (replaces lines 6–7 of Algorithm 6)

// intra-ortho II with precomputed Gram matrix $\mathbf{G}$

Input: $\mathbf{Q}^{(1)}_b$, $\mathbf{Q}_{1:i}$

Output: $\mathbf{R}^{(2)}_{1:i,b}$, $\mathbf{Q}^{(2)}_b$, $\mathbf{R}^{(2)}_{b,b}$, $\mathbf{Q}_b$

1: $[\mathbf{R}^{(2)}_{1:i,b}, \mathbf{G}'] = [\mathbf{Q}_{1:i}, \mathbf{Q}^{(1)}_b]^{\mathrm{T}}\mathbf{Q}^{(1)}_b$
2: $\mathbf{Q}^{(2)}_b = \mathbf{Q}^{(1)}_b - \mathbf{Q}_{1:i}\mathbf{R}^{(2)}_{1:i,b}$
3: $\mathbf{G} = \mathbf{G}' - \mathbf{R}^{(2)\,\mathrm{T}}_{1:i,b}\mathbf{R}^{(2)}_{1:i,b}$
4: $\mathbf{R}^{(2)}_{b,b} = \texttt{Cholesky}(\mathbf{G})$
5: $\mathbf{Q}_b = \mathbf{Q}^{(2)}_b\mathbf{R}^{(2)\,-1}_{b,b}$ via triangular solve

Algorithm 22 BCGSI2-SS$_3$ (replaces lines 5–6 of Algorithm 6)

// delayed intra-ortho II

Input: $\mathbf{Q}'_b$, $\mathbf{Q}_{1:i}$

Output: $\mathbf{R}^{(1)}_{b,b}$, $\mathbf{Q}^{(1)}_b$, $\mathbf{R}^{(2)}_{1:i,b}$, $\mathbf{Q}^{(2)}_b$

1: $[\mathbf{R}'_{1:i,b}, \mathbf{G}] = [\mathbf{Q}_{1:i}, \mathbf{Q}'_b]^{\mathrm{T}}\mathbf{Q}'_b$
2: $\mathbf{R}^{(1)}_{b,b} = \texttt{Cholesky}(\mathbf{G})$
3: $\mathbf{Q}^{(1)}_b = \mathbf{Q}'_b\mathbf{R}^{(1)\,-1}_{b,b}$ via triangular solve
4: $\mathbf{R}^{(2)}_{1:i,b} = \mathbf{R}'_{1:i,b}\mathbf{R}^{(1)\,-1}_{b,b}$ via triangular solve
5: $\mathbf{Q}^{(2)}_b = \mathbf{Q}^{(1)}_b - \mathbf{Q}_{1:i}\mathbf{R}^{(2)}_{1:i,b}$

Appendix B: Stability of RBGS methods

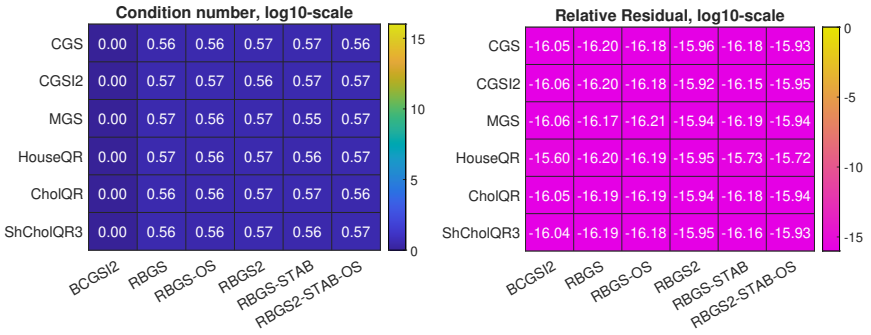

FIGURE B.1: `monomial` with $d = 1500$ and HH-QR solver.

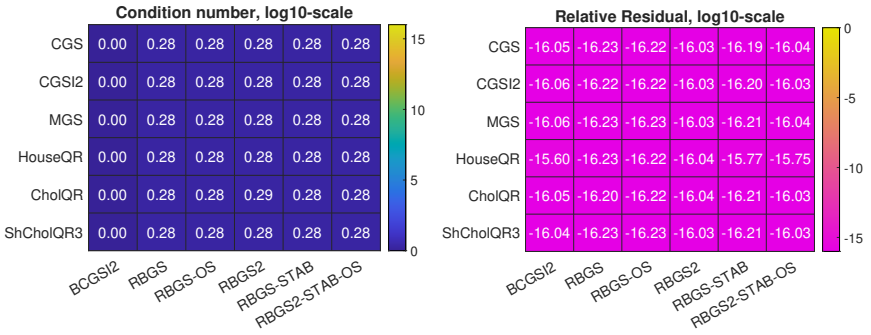

FIGURE B.2: `monomial` with $d = 5000$ and 5 Richardson iterations.

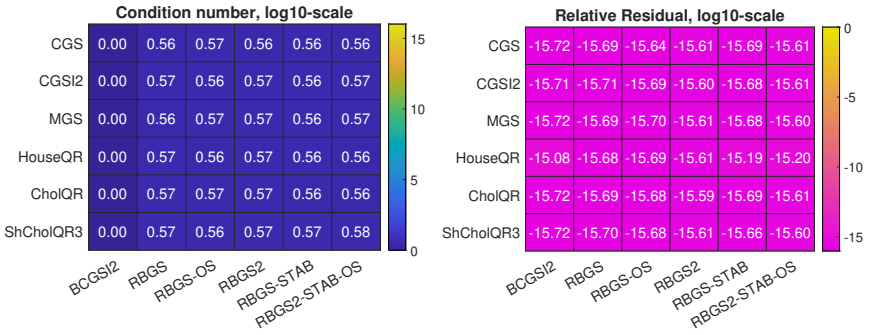

FIGURE B.3: `newton` with $d = 1500$ and HH-QR solver.

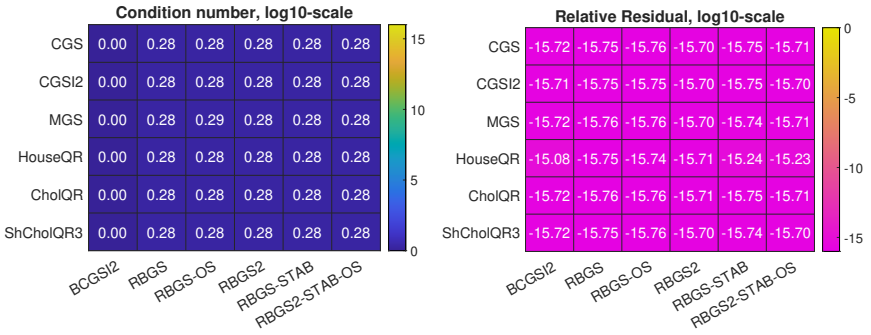

FIGURE B.4: `newton` with $d = 5000$ and 5 Richardson iterations.

Bibliography

[1] C. Alappat, J. Thies, G. Hager, H. Fehske, and G. Wellein. "Algebraic temporal blocking for sparse iterative solvers on multi-core CPUs". In: *The International Journal of High Performance Computing Applications* 0.0 (2024), pp. 1–21.

[2] W. E. Arnoldi. "The principle of minimized iteration in the solution of the matrix eigenvalue problem". In: *Quarterly of Applied Mathematics* 9 (1951), pp. 17–29.

[3] F. B. Alt. "Bonferroni Inequalities and Intervals". In: *Encyclopedia of Statistical Sciences*. Hoboken, NJ, USA: John Wiley & Sons, Ltd, 2006.

[4] 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.

[5] 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.

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

[7] 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.

[8] O. Balabanov and L. Grigori. "Randomized Block Gram–Schmidt Process for the Solution of Linear Systems and Eigenvalue Problems". In: *SIAM Journal on Scientific Computing* 47.1 (2025), A553–A585.

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

[10] 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.

[11] J. L. Barlow and A. Smoktunowicz. "Reorthogonalized block classical Gram-Schmidt". In: *Numerische Mathematik* 123.3 (Mar. 2013), pp. 395–423.

[12] 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.

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

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

[15] Å. Björck. "Solving linear least squares problems by Gram-Schmidt orthogonalization". In: *BIT Numerical Mathematics* 7 (1967), pp. 1–21.

[16] R. F. Boisvert, R. Pozo, K. Remington, R. F. Barrett, and J. Dongarra. "Matrix Market: a web resource for test matrix collections". In: *Quality of Numerical Software: Assessment and enhancement*. Ed. by R. F. Boisvert. Boston, MA: Springer US, 1997, pp. 125–137.

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

[18] 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).

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

[20] E. Carson, K. Lund, M. Rozložník, and S. Thomas. "Block Gram-Schmidt algorithms and their stability properties". In: *Linear Algebra and its Applications* 638 (2022), pp. 150–195.

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

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

[23] A. Chronopoulos and S. Kim. *S-Step Orthomin and GMRES implemented on parallel computers*. Technical Report. University of Minnesota, 1990.

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

[25] 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.

[26] M. P. Connolly, N. J. Higham, and T. Mary. "Stochastic Rounding and Its Probabilistic Backward Error Analysis". In: *SIAM Journal on Scientific Computing* 43.1 (2021), A566–A585.

[27] T. Davis and Y. Hu. "The University of Florida Sparse Matrix Collection". In: *ACM Transactions on Mathematical Software* 38.1 (2011), pp. 1–25.

[28] E. de Sturler and H. van der Vorst. "Reducing the effect of global communication in GMRES(m) and CG on parallel distributed memory computers". In: *Applied Numerical Mathematics* 18.4 (1995), pp. 441–459.

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

[30] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou. "Communication-optimal Parallel and Sequential QR and LU Factorizations". In: *SIAM Journal on Scientific Computing* 34.1 (2012), A206–A239.

[31] S. C. Eisenstat, H. C. Elman, and M. H. Schultz. "Variational Iterative Methods for Nonsymmetric Systems of Linear Equations". In: *SIAM Journal on Numerical Analysis* 20.2 (1983), pp. 345–357.

[32] M. Embree. "The Tortoise and the Hare Restart GMRES". In: *SIAM Review* 45.2 (2003), pp. 259–266.

[33] S. Fiore, M. Bakhouya, and W. W. Smari. "On the road to exascale: Advances in High Performance Computing and Simulations—An overview and editorial". In: *Future Generation Computer Systems* 82 (2018), pp. 450–458.

[34] M. Frigo and S. G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation", pp. 216–231.

[35] T. Fukaya, Y. Nakatsukasa, Y. Yanagisawa, and Y. Yamamoto. "CholeskyQR2: A Simple and Communication-Avoiding Algorithm for Computing a Tall-Skinny QR Factorization on a Large-Scale Parallel System". In: *2014 5th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems*. IEEE Press, 2014, pp. 31–38.

[36] P. Ghysels, T. J. Ashby, K. Meerbergen, and W. Vanroose. "Hiding Global Communication Latency in the GMRES Algorithm on Massively Parallel Machines". In: *SIAM Journal on Scientific Computing* 35.1 (2013), pp. C48–C71.

[37] 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.

[38] G. H. Golub and C. F. van Loan. *Matrix Computations*. Fourth Ed. The Johns Hopkins University Press, 2013.

[39] A. Greenbaum, V. Pták, and Z. Strakoš. "Any Nonincreasing Convergence Curve is Possible for GMRES". In: *SIAM Journal on Matrix Analysis and Applications* 17.3 (1996), pp. 465–469.

[40] L. Grigori and S. Moufawad. "Communication Avoiding ILU0 Preconditioner". In: *SIAM Journal on Scientific Computing* 37.2 (2015), pp. C217–C246.

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

[42] 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.

[43] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, et al. "An overview of the Trilinos project". In: *ACM Transactions on Mathematical Software* 31.3 (Sept. 2005), pp. 397–423.

[44] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. 2nd. SIAM, 2002.

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

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

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

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

[49] W. D. Joubert and G. F. Carey. "Parallelizable restarted iterative methods for nonsymmetric linear systems. Part I: Theory". In: *International Journal of Computer Mathematics* 44.1–4 (1992), pp. 243–267.

[50] W. D. Joubert and G. F. Carey. "Parallelizable restarted iterative methods for nonsymmetric linear systems. Part II: Parallel implementation". In: *International Journal of Computer Mathematics* 44.1–4 (1992), pp. 269–290.

[51] M. Kapralov, V. Potluru, and D. Woodruff. "How to Fake Multiply by a Gaussian Matrix". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. PMLR, June 2016, pp. 2101–2110.

[52] A. Kiełbasiński. "Analiza numeryczna algorytmu ortogonalizacji Grama-Schmidta". In: *Seria III: Matematyka Stosowana II* 2.2 (1974), pp. 15–35.

[53] S. K. Kim and A. T. Chrortopoulos. "An Efficient Parallel Algorithm for Extreme Eigenvalues of Sparse Nonsymmetric Matrices". In: *International Journal of High Performance Computing Applications* 6.1 (Apr. 1992), pp. 98–111.

[54] 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.

[55] G. A. Meurant. *Krylov methods for nonsymmetric linear systems : from theory to computations*. Vol. 57. Springer series in computational mathematics. Cham, Switzerland: Springer, 2020.

[56] M. Mohiyuddin, M. Hoemmen, J. Demmel, and K. Yelick. "Minimizing communication in sparse matrix solvers". In: *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. SC '09. Portland, Oregon: Association for Computing Machinery, 2009.

[57] R. Murray, J. Demmel, M. W. Mahoney, N. B. Erichson, M. Melnichenko, O. A. Malik, L. Grigori, P. Luszczek, M. Derezinski, M. E. Lopes, et al. *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.

[58] 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.

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

[60] 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.

[61] Y. Saad and M. H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (July 1986), pp. 856–869.

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

[63] A. Sobczyk and E. Gallopoulos. "Estimating Leverage Scores via Rank Revealing Methods and Randomization". In: *SIAM Journal on Matrix Analysis and Applications* 42.3 (2021), pp. 1199–1228.

[64] E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu, and Y. Wang. "Intel Math Kernel Library". In: *High-Performance Computing on the Intel Xeon Phi: How to Fully Exploit MIC Architectures*. Springer International Publishing, 2014, pp. 167–188.

[65] 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.

[66] Y. Yamamoto, Y. Nakatsukasa, Y. Yanagisawa, and T. Fukaya. "Roundoff error analysis of the CholeskyQR2 algorithm". In: *Electronic Transactions on Numerical Analysis* 44 (2015), pp. 306–326.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] J. Yang, X. Meng, and M. Mahoney. "Implementing Randomized Matrix Algorithms in Parallel and Distributed Environments". In: *Proceedings of the IEEE* 104.1 (2016), pp. 58–92.