Syracuse University

SURFACE

Dissertations - ALL SURFACE

August 2017

Stable Sparse Orthogonal Factorization of Ill-Conditioned Banded Matrices for Parallel Computing

Qian Huang
Syracuse University

Follow this and additional works at: https://surface.syr.edu/etd

 Part of the Physical Sciences and Mathematics Commons

Recommended Citation

Huang, Qian, "Stable Sparse Orthogonal Factorization of Ill-Conditioned Banded Matrices for Parallel Computing" (2017). *Dissertations - ALL*. 772.
https://surface.syr.edu/etd/772

This Dissertation is brought to you for free and open access by the SURFACE at SURFACE. It has been accepted for inclusion in Dissertations - ALL by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.

ABSTRACT

Sequential and parallel algorithms based on the LU factorization or the QR factorization have been intensely studied and widely used in the problems of computation with large-scale ill-conditioned banded matrices. Great concerns on existing methods include ill-conditioning, sparsity of factor matrices, computational complexity, and scalability. In this dissertation, we study a sparse orthogonal factorization of a banded matrix motivated by parallel computing. Specifically, we develop a process to factorize a banded matrix as a product of a sparse orthogonal matrix and a sparse matrix which can be transformed to an upper triangular matrix by column permutations. We prove that the proposed process requires low complexity, and it is numerically stable, maintaining similar stability results as the modified Gram-Schmidt process. On this basis, we develop a parallel algorithm for the factorization in a distributed computing environment. Through an analysis of its performance, we show that the communication costs reach the theoretical least upper bounds, while its parallel complexity or speedup approaches the optimal bound. For an ill-conditioned banded system, we construct a sequential solver that breaks it down into small-scale underdetermined systems, which are solved by the proposed factorization with high accuracy. We also implement a parallel solver with strategies to treat the memory issue appearing in extra large-scale linear systems of size over one billion. Numerical experiments confirm the theoretical results derived in this thesis, and demonstrate the superior accuracy and scalability of the proposed solvers for ill-conditioned linear systems, comparing to the most commonly used direct solvers.

STABLE SPARSE ORTHOGONAL FACTORIZATION OF ILL-CONDITIONED BANDED MATRICES FOR PARALLEL COMPUTING

by

Qian Huang

B.S., Sun Yat-sen University, 2006

M.S., Sun Yat-sen University, 2008

Dissertation

Submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Mathematics.

Syracuse University

August 2017

Copyright © Qian Huang 2017

All Rights Reserved

Acknowledgements

I would like to express my special gratitude for advice from Professor Yuesheng Xu who gave direction to this dissertation. I would like to thank Professor Uday Banerjee and Professor Lixin Shen for their comments and suggestions. I am also indebted to my many student colleagues for providing help through years of graduate study. I am especially grateful to Liang Zhao, Feishe Chen, Xiaoxia Liu, and Xuefei Ma. Last but not least, I am sincerely thankful for the help and support from my parents and my wife who encouraged me so much while I was writing this dissertation. All remaining errors are my own.

Contents

Chapter 1

Introduction

Large-scale ill-conditioned banded matrices are commonly seen in classical problems of numerical differentiation and integration in applied mathematics, as well as modern applications in data mining and data analysis. Main issues for computing with such matrices include stability, ill-conditioning, sparsity, complexity, and scalability. Especially in big data processing, the efficiency of data retrieval and exchange is at least as important as the efficiency of computation.

Existing direct methods either have poor treatment of ill-conditioning (such as the LU factorization) or generate dense factor matrices and resulting in high computational complexity (such as the rank revealing QR factorization). Existing iterative methods such as Krylov subspace methods have the issue of scalability, due to the dependency of their communication costs on the number of iterations. Moreover, neither existing direct nor iterative methods are capable of computing with extra large-scale matrices whose sizes are over one billion since they would suffer from the issue of insufficient memory storage.

In this dissertation, we propose both sequential and parallel methods for a stable sparse orthogonal factorization of banded matrices. The proposed methods resolve the issues mentioned above, especially, the parallel method is able to treat extra large-scale matrices, and thus can be applied in big data processing.

We organize this chapter into five sections. In section 1.1, we present the motivation of this dissertation with various applications and techniques. In section 1.2, we discuss the issues of existing direct or iterative methods. We introduce our new approach for resolving the issues in section 1.3, and summarize our contribution in section 1.4. We finally describe the organization of the remaining chapters of this dissertation.

1.1 Motivation

Large-scale banded matrices arise from a variety of application areas in science and engineering. They include numerical differentiation and integration in applied mathematics, such as the spline interpolation and smoothing [48], eigenvalue problems with inverse iterations [43], multigrid methods [18], the solutions of ordinary differential equations by finite difference or finite element methods [65], and the solutions of partial differential equations by alternating direction implicit methods [5]. In many cases, the derived banded matrices are ill-conditioned. For example, the nonparametric smoothing of curves and surfaces defined by scattered data, image reconstruction with compactly supported radial basis functions [22], the solutions of backward heat equations and the Cauchy problem for parabolic equations, as well as the modeling of non-smooth solutions by generalized finite element methods [40] would lead to ill-

conditioned banded matrices.

In recent years, large-scale banded matrices appear frequently in the applications of data mining and data analysis. For instance, in the study of image segmentation [99], paper citations [17], social and biological networks [8, 42, 98], raw data are often represented by a huge connected graph, yet the essential data of interest are a subgraph including groups of nodes in the graph that are strongly-connected among themselves but are weakly-connected to the rest of the graph. Retrieved by clustering methods such as k-means and the Markov clustering, the essential subgraph has a sparse structure with nodes clustered by groups. Indeed, the subgraph can be further represented by a large-scale banded matrix which is usually ill-conditioned. Problems of this kind include, in social network analysis, extracting latent space to capture user latent features [59, 60], collaborative filtering recommender systems [50, 91, 95, 96], identifying closely-interacting communities and circles [71, 90], as well as in bioinformatics, detecting frequently occurring patterns in protein structures [67] and predicting protein structures from their molecular sequences [94], and so on. In many cases, the representing matrix of a network may have (column/row) size over one billion (see, for example, [110, 118]). We shall call such a matrix an *extra large-scale* matrix.

Among the applications in social network analysis, recommender system is one of the hottest topics. It attempts to suggest items (such as movies, books, news, Web pages, etc.) that are likely to interest users. Due to the great potential commercial value, recommender systems have attracted a lot of attention in the past decade. For instance, the Netflix Prize was an open competition for the best method to predict user ratings for films, based on previous ratings without the users or the films being

identified except by numbers assigned for the contest [45, 109]. The database (including training set and qualifying set) released for the contest contains over 100 million movie ratings made by over 0.48 million users [11]. Recommender systems have also been successfully deployed in industry, such as product recommendation at Amazon [73], music recommendation at iTunes, movie recommendation at Netflix, etc. Indeed, they have brought large profits to commerce. For example, a report in [78] said that in 2013, 35 percent of Amazon's, and 75 percent of Netflix's sales resulted from personalized recommendations.

Related recommendation techniques have been widely studied in research communities of information retrieval [60, 117], machine learning [9, 93, 102], and data mining [10, 77]. Collaborative filtering (CF) is one of the most important techniques among them [50, 95, 96, 104]. CF is a method of making automatic predictions (filtering) about the interests of a particular user by collecting preferences or taste information from similar users (collaborating). The underlying assumption of CF is that users with similar behaviors would have similar preferences. Traditional CF-based models suffer from the sparsity of the user-item rating matrix and imbalance of rating data. One typical problem is the cold start problem that it is unreliable to make recommendations for new or infrequent users due to insufficient rating data to capture their preferences [70, 97]. Another weakness of traditional CF-based models is that they ignore social interactions or connections among users. But in our real life, social recommendation is an everyday occurrence, as we always turn to colleagues or friends for recommendations. A research work conducted in [101] revealed that people who chat with each other using MSN instant messenger are more likely to share interests

(since their Web search records on the Live Search Engine are the same or topically similar). To overcome these weaknesses, many recent models have been proposed to explore additional information, such as item's content information [9, 115] and user's social network [46, 64, 119].

The key of the recent CF-based models is to compute with large-scale ill-conditioned banded matrices efficiently. For example, to incorporate social network information among users, an important step is to extract user latent features from the social network matrix, known as social matrix factorization [1, 63, 68, 77, 81]. It resorts to the low-rank matrix approximation via the truncated singular value decomposition (SVD). To make use of the sparsity of the social network matrix, iterative methods such as Arnoldi or Lanczos iteration may be applied for computing a few eigenpairs (of a related symmetric matrix), in order to obtain the truncated SVD [60, 85, 107]. Other methods include building more sophisticated models such as probability matrix factorization and collaborative topic regression. Then iterative methods such as gradient descent are employed for accomplishment [76, 89]. In either way, since computations within each iteration mainly are matrix-vector multiplications and additions, where matrices are sparse, these iterative methods are quite efficient in terms of low computational complexity. However, in modern computing systems, efficiency not only rely on short computing time, but also count on fast data access and retrieval. In this particular application, the main challenge is that social network matrix is of huge size and may be stored by parts in different locations.

We are living in a big data era. Experiments, observations, and numerical simulations in many areas of science nowadays generate terabytes of data and, in some

cases, are on the verge of generating many petabytes. This rapid growth has a great impact on scientific computing, shifting it from "compute-centric science" to "data-centric science". While traditional methods have largely focused on developing speedy algorithms within the confines of a local computing environment, the growth of big data requires new paradigms addressing how data are distributed, accessed, retrieved, exchanged, and computed [28, 66, 83, 88]. Social media such as Facebook, YouTube, LinkedIn, and Twitter have exploded beyond everyone's wildest imagination. Today some of these companies have over millions of users and the number keeps increasing every year. These users comprise giant social networks, through which they generate massive amounts of data such as texts, images, and videos. Undoubtedly, for data of such scale, there is no way to store and simultanenously process by a single or a small group of processors. Indeed, the most commonly used computing architecture nowadays is the parallel and distributed computing system, in which, large-scale data are distributed across locations, hundreds of thousands of processors are clustered as a supercomputer, interconnected by express networks, and processors access partial data and process concurrently, exchanging data among themselves when necessary [4, 31, 106].

1.2 Issues with the Existing Methods

In numerical linear algebra, direct methods such as the LU factorization, the QR factorization, and SVD play important roles for computing with ill-conditioned banded matrices. In the aspect of sparsity, the factors are banded in the LU factorization of

banded matrices. However, in general, the LU factorization has poor performance for ill-conditioned matrices due to the numerical rank deficiency. Well-known methods for treating ill-conditioning include the rank revealing QR factorization [26], the rank revealing SVD and recently, the rank revealing decomposition for computing accurate SVD [25, 35, 54]. Nevertheless, these methods are computationally expensive to implement. Moreover, even for banded matrices, they would generate dense matrices, resulting in high computational complexity. In software development, Linear Algebra Package (LAPACK) [3] is a state-of-the-art software library for numerical linear algebra. It includes the routines such as the LU and the QR factorizations.

Computing with large-scale matrices that arise from modern applications brings us new challenges on data accessing and processing. As mentioned at the end of the last section, it is impossible to process a large-scale matrix by a single processor, and thus any sequential method cannot satisfy the need for such computing. We count on parallel methods designed for the parallel and distributed computing systems. Parallel performance depends not only on concurrent computation, but also on communication (interactions among processors such as messages and data transmission between processors) [69, 75]. Especially for (extra) large-scale distributed data, a critical topic is to reduce communication costs (total number of communication messages and total number of communication words of data transmitted) [84]. For a parallel algorithm, scalability is the key measurement of its parallel performance. It is indicated by speedup, the ratio of the running time of its corresponding sequential algorithm and the running time of the parallel algorithm, optimum of which could be obtained by maximizing the concurrency of computations, as well as minimizing the communication

costs.

Parallel direct methods for computing with ill-conditioned banded matrices include elimination-based methods and parallel versions of the LU and the QR factorizations. The early parallel algorithms based on elimination include the cyclic reduction algorithm [55], the recursive doubling algorithm [103], and the block partitioned elimination algorithms [80, 114]. Dongarra and others proposed a divide-and-conquer algorithm and implemented in Scalable LAPACK (ScaLAPACK), a state-of-the-art library of high-performance linear algebra routines for parallel distributed memory machines [16, 29, 38]. Essentially this algorithm is a parallel version of the LU factorization. Improvement on it includes the single-width separator approach [37], double-width separator approach [116], and load-balanced scheme [44]. On this basis, Polizzi and Sameh proposed the SPIKE algorithm [87], a hybrid method combining the parallel LU factorization and iterative methods. For general sparse matrices, SuperLU [72] is a popular software package including parallel algorithms based on the LU factorization. Parallel versions of the QR factorization include variants of block Gram-Schmidt process [62], block-based Householder triangularization by the WY representation [12]. One main issue with these commonly used algorithms is that their performance is dominated by communication. To reduce the communication costs, Demmel and others proposed a family of algorithms using communication-avoiding techniques, including the Tall Skinny QR factorization (TSQR) and communication-avoiding QR (CAQR) as in [34]. A rank revealing version of TSQR was proposed in [56] and implemented in the Trilinos software package [52]. More studies on TSQR or CAQR can be seen in [2, 30]. Other parallel rank revealing methods include the multifrontal Householder

QR factorization [32], which is implemented in the package SuiteSparseQR of the MATLAB library.

We briefly review parallel iterative methods for computing a few eigenpairs of large-scale sparse matrices since they are widely used in applications in data analysis. They include variants of parallel Krylov subspace methods such as Arnoldi (for unsymmetric problems) and Lanczos (for symmetric problems) methods. In either method, restarted strategies are required since the storage and computational complexity grows as the iteration proceeds [51]. Implicitly restarted strategy via implicitly shifted QR iterations is more often adopted comparing to the simple explicit restarted strategy since the former is more effective in terms of convergence [21, 82]. Blocked versions of Arnoldi/Lanczos algorithms are studied in [49, 92] for better parallelism. Methods for reducing communication costs include communication-avoiding techniques and others, see [23, 24, 57, 100], for example.

Parallel direct methods such as the rank revealing QR factorization have advantages in scalability, since in their implementation, processors may access a given matrix only once with communication time of the order of the logarithm of the number of processors [7]. However, computations in these methods may impair the sparsity of derived matrices, resulting in high parallel complexity. In contrast, parallel Krylov subspace methods have low complexity for heavily using sparse matrix-vector operations. Nevertheless, it is pointed out in [100] that they are not as scalable as expected for communication, since processors have to access the given matrix, and communicate with each other for each iteration or for every certain number of iterations [7].

Notice that in all the literature mentioned above, we have not seen an effective

method to treat extra large-scale matrices. Due to the limited memory of processors, it is impossible to store matrices to be processed constantly in memory. An effective treatment for this memory issue is to design external memory algorithms so that each processor can load the given matrix by parts into memory, process, and release the intermediate results [33, 113]. However, existing direct or iterative methods are not likely to incorporate such treatment since they have to store intermediate results constantly in memory. For example, to solve a linear system, methods using the QR factorization have to store the factor matrices until the factorization process is complete, before applied to compute the solution of the system. Likewise, Krylov subspace methods must store the Krylov subspace basis vectors constantly in memory [23].

1.3 A New Approach

In this dissertation, we develop a stable sparse orthogonal factorization which we shall call the block QS factorization for banded matrices. It factorizes a banded matrix as a product of a sparse orthogonal matrix Q and another sparse matrix S, with SE being upper triangular for a permutation matrix E. We design a modified block Gram-Schmidt process for the factorization motivated by parallel computing. It orthogonalizes the banded matrix by column blocks, using an intrinsic property of block-wise quasi-orthogonality. Theoretically, we prove that the proposed process fulfils the block QS factorization. We also perform a complexity analysis, and prove the stability of the process. These theoretical results are competitive with traditional methods for matrix

orthogonal factorization such as the Householder triangularization and the modified Gram-Schmidt process.

We then develop a parallel algorithm for the factorization in the Bulk Synchronous Parallel model for distributed memory computing. In the parallel implementation, we make use of the concurrent computation in the modified block Gram-Schmidt process, and apply a binary reduction tree structure for communication. We analyze the parallel performance of the proposed algorithm, in which we have shown that its numbers of communication messages and words both reach their theoretical optimal upper bounds. We have also proved that the parallel complexity approaches its optimal upper bound, up to a factor of a logarithmic quantity, while the speedup is close to its optimal lower bound, except for a divisor of the logarithmic number of processors.

For an ill-conditioned banded system, we construct a solver that breaks it down into small-scale underdetermined systems, to which minimum 2-norm solutions can be obtained by the block QS factorization with high accuracy. The solver has a *memory-less* feature that it can compute the final solution without storing the matrices generated over the entire process. For the memory-less feature, the proposed sequential solver requires low memory storage, and is scalable to large-scale linear systems. With this feature, we apply a *partial load* strategy in the implementation of the parallel solver, so as to deal with the memory issue in solving extra large-scale systems. We finally present numerical experiments to illuminate the theoretical results, to show the outstanding accuracy of the proposed solvers among the most commonly used direct solvers, and to demonstrate the excellent scalability of the parallel solver, together with its capability of solving extra large-scale systems.

1.4 Contribution of this Dissertation

We summarize our contribution as follows.

1. We propose a modified block Gram-Schmidt process for the block QS factorization, which is a new fundamental direct method for banded matrices. The process is stable, generating sparse factor matrices, scalable for parallel computing, and able to treat ill-conditioned matrices.

2. We develop a parallel algorithm for the block QS factorization in a distributed memory system. The algorithm has nice parallel performance, with communication costs, parallel complexity, and speedup reaching or approaching their optimal bounds.

3. The proposed parallel algorithm is capable of computing with extra large-scale matrices, whose sizes are over one billion, and thus can be applied as a building block for computations in big data processing. Unique to traditional methods, it does not form factor matrices explicitly, and may release memory for intermediate matrices once applied to related computations.

1.5 Organization of this Dissertation

We organize the remaining of this dissertation into four chapters. In the second chapter, we present a stable process for the block QS factorization, then we study the sparsity of the factor matrices, the complexity, and the stability of the proposed process. A recursive algorithm will also be presented to simplify the implementation of the

process. In the third chapter, we propose the parallel algorithm for the factorization, and conduct an analysis on its parallel performance. The fourth chapter is devoted to the design of both sequential and parallel solvers for ill-conditioned systems. We also study the round-off error analysis of the sequential solver, and the strategies in the parallel solver for extra large-scale linear systems. In the fifth chapter, we present the numerical experiments with comparisons among the factorization processes, and among the direct solvers.

Chapter 2

A Stable Process for Sparse Orthogonal Factorization

In this chapter, we develop a stable orthogonalization process to factorize a banded matrix as a product of a sparse orthogonal matrix and another sparse matrix which can be transformed to an upper triangular matrix by column permutations. This will provide, for example, a convenient way to find the (pseudo-)inverse of a banded matrix. Notice that an orthogonalization process may be performed on the column vectors or the row vectors of a matrix. In this thesis, we consider the orthogonalization of the *column* vectors of a matrix, and call it the *matrix orthogonalization*. As our goal is to orthogonalize the column vectors of the matrix, we require that the second sparse matrix can be transformed to an upper triangular matrix by column permutations only.

Motivated by parallel computing, the proposed process is based on operations of column blocks of a banded matrix associated with a partition. We have found that a

banded matrix partitioned by its column blocks has a block-wise quasi-orthogonality between its non-adjacent column blocks. Through the study on the block Gram-Schmidt process (BGS) [62], we present a modified version of BGS, with the modification that we employ column block permutations during the orthogonalization process so as to utilize the block-wise quasi-orthogonality. As a result, the proposed process has low complexity and produces sparse factor matrices.

To prove the sparsity of the factor matrices, we introduce a new method to describe the sparse structure (the number and the distribution of nonzero entries) of the matrices to be orthogonalized during the proposed process. This method could be applied in the analysis of the sparse structure of a matrix whose nonzero entries are clustered in its blocks, such as banded matrices and block diagonal matrices. It can be also used in a round-off error analysis according to the IEEE standard for floating point arithmetic.

We verify that the proposed process is stable. The main idea is to prove that the process achieves numerical stability similar to the modified Gram-Schmidt process (MGS) [43]. We show that for a banded matrix with a certain partition, the proposed process produces the same orthogonal matrix as MGS applied to the matrix multiplying from the right by a permutation matrix. Moreover, for the error between the banded matrix and the product of the computed factor matrices, the proposed process obtains a smaller upper bound comparing to the one derived in [13].

We organize this chapter in six sections. In section 2.1, we describe the block QS factorization and propose the orthogonalization process. We verify in section 2.2 that the proposed process fulfils the block QS factorization. In sections 2.3 and 2.4

we present, respectively, the sparsity of the factor matrices and the complexity of the proposed process. We prove the stability of the process in section 2.5 and present a recursive algorithm that implements the process in section 2.6.

2.1 The Block QS Factorization

In this section, we derive a modified BGS for sparse orthogonal factorization of banded matrices. Specifically, we call the sparse orthogonal factorization the block QS factorization, as described below.

Let $\mathbb{N}$ denote the set of all natural numbers. For $s \in \mathbb{N}$ we let $\mathbb{N}_s := \{1, 2, \ldots, s\}$. Let $A := [a_{ij} : i \in \mathbb{N}_n, j \in \mathbb{N}_m]$ be a banded matrix of full rank with bandwidth $k/2$, where $n \geq m$, k is even, and $k \ll m$, that is, $a_{ij} = 0$, for $i \in \mathbb{N}_n$, $j \in \mathbb{N}_m$ with $|i - j| > k/2$. We wish to obtain a factorization

$$A = QS, \tag{2.1}$$

where $Q \in \mathbb{R}^{n \times m}$ is a sparse orthogonal matrix, and $S \in \mathbb{R}^{m \times m}$ is a sparse matrix with SE being upper triangular for a permutation matrix E. To develop an efficient method to accomplish this, we shall be mindful about the following features:

1. For the sparsity requirement, the number of the nonzero entries of Q or S must be $O(m \log m)$ or less.

2. The factorization process should be stable. Especially, it can handle an ill-conditioned matrix A.

3. The factorization process should have a low complexity. Specifically, the number

of floating point operations must be $O(m \log m)$ or less.

4. The factorization process is favorable to parallel computing, and is scalable to large-scale matrices. Especially, it can treat the cases when $m \geq 10^9$.

It is well-known that block-based algorithms in linear algebra are desirable for modern high performance computers, especially in a parallel environment where data are naturally distributed and computed by blocks for less communication costs. For example, BGS [62] partitions a matrix by blocks of consecutive columns (column blocks), to orthogonalize each column block within it by MGS, and the column blocks with respect to each other. In this section, we shall construct the factorization (2.1) by a process of BGS with column block permutations (BGSP). The introduction of permutations is motivated by an orthogonality property between the non-adjacent column blocks of a banded matrix partitioned by its column blocks. To be more precise, we define the notation of a matrix partitioned by column blocks.

Definition 2.1.1. *Let $A \in \mathbb{R}^{n\times m}$ be a matrix, $\gamma \in \mathbb{N}$ with $\gamma \leq m$. Let $\Pi := [m_i : i \in \mathbb{N}_\gamma]$ be a vector with $m_i \in \mathbb{N}$ and $\sum_{i\in\mathbb{N}_\gamma} m_i = m$. Matrix A is said to be partitioned by Π if for each $i \in \mathbb{N}_\gamma$, A_i is the $n \times m_i$ submatrix of A such that*

$$A := [A_i : i \in \mathbb{N}_\gamma]. \tag{2.2}$$

We call Π the partition vector of A and use $A(\Pi)$ to indicate the matrix A partitioned by the specific partition vector Π.

Through out this thesis, we shall assume that all matrices are associated with a specific partition and use the notation $A(\Pi)$ to indicate the matrix A associated with the partition Π. However, when the association is clear from the context, we shall

write the matrix as A for simplicity. Also, the terminology *column blocks* used in this thesis refers to column blocks of a matrix associated with a specific column block partition. Henceforth, we say matrix Q is column orthonormal if $Q^TQ = I$.

We review below the orthogonalization process of BGS [62]. Given a matrix A with partition (2.2), BGS proceeds the orthogonalization of A in γ steps. For $j \in \mathbb{N}_\gamma \backslash \{1\}$, suppose that the column orthonormal matrix $[Q_i : i \in \mathbb{N}_{j-1}]$ was obtained from the first $(j-1)$ column blocks of A. We then project A_j onto the orthogonal complement of the range space of Q_i for $i = 1, 2, \ldots, j-1$ by

$$\bar{A}_j := \prod_{i \in \mathbb{N}_{j-1}} (I - Q_{j-i} Q_{j-i}^T) A_j, \tag{2.3}$$

and generate the column orthonormal matrix Q_j by MGS from $\bar{A}_j$. This step results in the updated column orthonormal matrix $[Q_i : i \in \mathbb{N}_j]$. We continue this process until $j = \gamma$.

A number of variants of BGS were proposed to improve its numerical performance. For example, [62] proposed a variant B2GS of BGS which replaces MGS by MGS with reorthogonalization so that it has stability results similar to MGS. We notice that BGS or its variants fails to produce a sparse column orthonormal matrix, even when it is applied to a banded matrix. This can be seen from the calculation of Q_j. In fact, the entries of $Q := [Q_j : j \in \mathbb{N}_\gamma]$ above its diagonal are nonzero in general, and the computation in (2.3) is the main source of these nonzero entries.

The goal of this section is to propose a modified BGS so that when it is applied to a banded matrix A it generates a factorization of A with a sparse column orthonormal matrix Q and a sparse matrix S, with SE being upper triangular for a permutation matrix E. The development is motivated by the *block-wise quasi-orthogonal* property

of banded matrices, which we define below [86].

Definition 2.1.2. *Let* Π *be a partition vector. Matrix* $A(\Pi)$ *is said to be block-wise quasi-orthogonal if each of the column blocks of* $A(\Pi)$ *is orthogonal to its non-adjacent column blocks. Matrix* $A(\Pi)$ *is said to be block-wise orthogonal if the column blocks of* $A(\Pi)$ *are orthogonal to each other.*

The block-wise quasi-orthogonality is commonly seen in banded matrices. Indeed, if $A \in \mathbb{R}^{n\times m}$ is a banded matrix with bandwidth $k/2$, and $\Pi := [m_i : i \in \mathbb{N}_\gamma]$ with $m_i \geq k$, then $A(\Pi)$ is block-wise quasi-orthogonal.

Our main idea in developing the modified BGS is as follows: For a banded matrix, we partition it by column blocks so that it is block-wise quasi-orthogonal. With this partition, we may construct a submatrix A_1 of A with its all column blocks that inherit from Π orthogonal to each other. Using a sequence of permutations, we can construct a new matrix $\tilde{A}$ (consisting of two submatrices) which has the submatrix A_1 as its first submatrix and the remaining column blocks of A as its second submatrix A_2. We orthogonalize the columns of A_1 and project A_2 onto the orthogonal complement of the range space of A_1 in the range space of A, which results in a matrix denoted by $\bar{A}_2$. We then apply the same procedure described above to the matrix $\bar{A}_2$ and repeat the process until the factorization is completed. Since this procedure involves column block permutations of A and other matrices, we shall call it the modified BGS with column block permutations, with abbreviation BGSP. We remark that, if we employ MGS for the orthogonalization of A_1 and the computation of $\bar{A}_2$, then under a special partition of A, BGSP produces the same column orthonormal matrix as MGS applied to AE, for a permutation matrix E. We shall study this in section 2.5.

We now describe precisely the BGSP procedure for sparse orthogonal factorization of a block-wise quasi-orthogonal matrix A associated with a given partition. We shall specify the three keystones: construction of two submatrices A_1, A_2 of A, orthogonalization of A_1 whose column blocks are orthogonal to each other, and the projection of A_2 onto the orthogonal complement of the range space of A_1 in the range space of A. For simplicity, we denote by $\mathrm{orthn}(X)$ a orthonormalization process that generates a column orthonormal matrix Y from an input matrix X of full rank such that they have the same range space. We also denote by $\mathrm{proj}(X, Y)$ a process of projection that generates a matrix $Z \in \mathbb{R}^{n\times s}$ from input matrices $X \in \mathbb{R}^{n\times s}$ and $Y \in \mathbb{R}^{n\times t}$ such that $Z^T Y = 0$ and matrices $[Z, Y]$ and $[X, Y]$ have the same range space.

We construct a submatrix of a block-wise quasi-orthogonal matrix $A(\Pi) := [A_i : i \in \mathbb{N}_\gamma]$. For a subvector $\Omega := [s_i : i \in \mathbb{N}_\omega]$ of $\Gamma := [i : i \in \mathbb{N}_\gamma]$ with $1 \le \omega < \gamma$, we define a subvector Π_Ω of the partition vector $\Pi := [m_i : i \in \mathbb{N}_\gamma]$ by

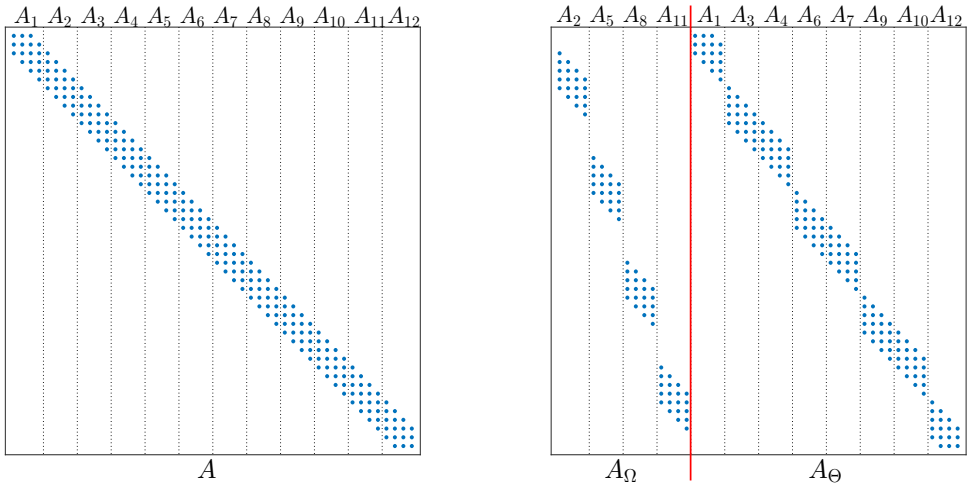
$$\Pi_\Omega := [m_{s_i} : i \in \mathbb{N}_\omega]. \tag{2.4}$$

Here, the component of the vector Ω indicates the location of the component of Π_Ω in the vector Π. Specifically, the i-th component of Π_Ω is the s_i-th component of Π. Next, for the partitioned matrix $A(\Pi)$, we define the submatrix A_Ω of $A(\Pi)$ by

$$A_\Omega := [A_{s_i} : i \in \mathbb{N}_\omega]. \tag{2.5}$$

We now construct two submatrices A_1 and A_2 of a block-wise quasi-orthogonal matrix $A(\Pi) := [A_i : i \in \mathbb{N}_\gamma]$ with $\gamma/3 \in \mathbb{N}$. Let $\Omega := [3i - 1 : i \in \mathbb{N}_{\gamma/3}]$ and $\Theta := [3i - 2, 3i : i \in \mathbb{N}_{\gamma/3}]$. We present below properties of the two submatrices $A_1 := A_\Omega$, $A_2 := A_\Theta$. An example of the construction is illustrated in Figure 2.1.

***Remark* 1.** For each $i \in \mathbb{N}$, let $s_i := 3i-1$, $t_{2i-1} := 3i-2$, $t_{2i} := 3i$. If A is block-wise quasi-orthogonal, then A_Ω is block-wise orthogonal, and for $i \in \mathbb{N}_{\gamma/3}$, $j \in \mathbb{N}_{2\gamma/3}$ with $i \neq \lceil j/2 \rceil$, $A_{s_i}^T A_{t_j} = 0$.



Construction of A_Ω, A_Θ from A

FIGURE 2.1: *Construction of A_Ω and A_Θ from a pentadiagonal matrix A.*

We next discuss a realization of orthn(A) for a block-wise orthogonal matrix A. This may be done by orthonormalizing each column block of A using stable methods such as the QR factorization via MGS, Householder, or Givens. As an example, we describe orthonormalization using MGS only. For matrices $X \in \mathbb{R}^{n\times m}$, $Y \in \mathbb{R}^{s\times t}$, we use the notation $X \oplus Y$ to denote a block diagonal matrix in $\mathbb{R}^{(n+s)\times(m+t)}$ with the first and second diagonal blocks being X and Y, respectively. Suppose that A with partition (2.2) is of full rank. For each $i \in \mathbb{N}_\gamma$, we employ MGS to construct the column orthonormal Q_i and the upper triangular R_i from A_i. We define

$$Q := [Q_i : i \in \mathbb{N}_\gamma], \quad R := \bigoplus_{i\in\mathbb{N}_\gamma} R_i. \tag{2.6}$$

Then, Q and R form an orthogonal factorization of A.

Below, we describe a realization of proj(A, Q) for partitioned matrices A and Q.

We begin with a realization of $\mathrm{proj}(A_0, Q_0)$ where A_0, Q_0 are column blocks of A and Q, respectively, with Q_0 being column orthonormal. We fulfil this by using MGS. Indeed, for matrices $A_0 \in \mathbb{R}^{n\times s}$, $Q_0 := [q_i : i \in \mathbb{N}_t] \in \mathbb{R}^{n\times t}$, we construct matrices $\bar{A}_0 \in \mathbb{R}^{n\times s}$ by

$$\bar{A}_0 := \left[\prod_{i\in\mathbb{N}_t} \left(I - q_{t+1-i}\, q_{t+1-i}^T\right)\right] A_0 \tag{2.7}$$

and

$$C_0 := [c_i^T : i \in \mathbb{N}_t]^T,$$

where

$$c_i := q_i^T \left[\prod_{r\in\mathbb{N}_{i-1}} \left(I - q_{i-r}\, q_{i-r}^T\right)\right] A_0, \quad \text{for} \quad i \in \mathbb{N}_t. \tag{2.8}$$

Now we generalize to a realization of $\mathrm{proj}(A, Q)$ for $A := [A_j : j \in \mathbb{N}_{2\gamma}]$, $Q := [Q_i : i \in \mathbb{N}_\gamma]$, with Q being column orthonormal and $Q_i^T A_j = 0$ for $i \in \mathbb{N}_\gamma$, $j \in \mathbb{N}_{2\gamma}$ with $i \neq \lceil j/2 \rceil$. In this case, for each $j \in \mathbb{N}_{2\gamma}$, realizing $\mathrm{proj}(A_j, Q)$ is reduced to computing $\mathrm{proj}(A_j, Q_r)$, for $r := \lceil j/2 \rceil$. We construct the matrices $\bar{A}_j$, C_j by $\mathrm{proj}(A_j, Q_r)$. We then define the matrices $\bar{A}$ and C by

$$\bar{A} := [\bar{A}_j : j \in \mathbb{N}_{2\gamma}], \quad C := \bigoplus_{j\in\mathbb{N}_\gamma} [C_{2j-1}, C_{2j}]. \tag{2.9}$$

Then the matrix $\bar{A}$ satisfies that $\bar{A}^T Q = 0$ and matrices $[\bar{A}, Q]$ and $[A, Q]$ have the same range space. Notice that the matrix C is for constructing the factor matrix S in (2.1).

We next assemble all the components described above for the main orthogonalization process. Given an input matrix A partitioned by Λ of length γ, we shall proceed its orthogonalization in finitely many steps. Let $A^1(\Lambda^1) := A(\Lambda)$. Suppose in the

ℓth step, it remains to orthogonalize $A^\ell(\Lambda^\ell)$. We construct two submatrices A^ℓ_Ω and A^ℓ_Θ of A^ℓ, generate a column orthonormal matrix Q^ℓ by $\mathrm{orthn}(A^\ell_\Omega)$, and then apply $\mathrm{proj}(A^\ell_\Theta, Q^\ell)$, producing a matrix $A^{\ell+1}$ to be processed in the next step. Note that if we defined $\Pi^\ell := \Lambda^\ell$ and used the above approach to obtain A^ℓ_Ω and A^ℓ_Θ, then by observation, the number of column blocks of $A^{\ell+1}$ would be two thirds of that of A^ℓ, resulting in totally $\log_{3/2}(\gamma)$ steps. Notice that the number of steps is crucial in a parallel environment, since it determines the number of times required for communication. To achieve $\log_2(\gamma)$ steps that is optimal, we shall define below Π^ℓ so that its length is three fourths of the length of Λ^ℓ, and associate A^ℓ with it.

Let $\gamma := 2^L$ for some $L \in \mathbb{N}$. For each $\ell \in \mathbb{N}_L$, let

$$\gamma_\ell := 2^{L-\ell+1}. \tag{2.10}$$

For $\ell \in \mathbb{N}_{L-1}$, given matrix A^ℓ partitioned by $\Lambda^\ell := [n_{\ell,i} : i \in \mathbb{N}_{\gamma_\ell}]$, we construct submatrices A^ℓ_Ω, A^ℓ_Θ of A^ℓ, and a column block permutation matrix E^ℓ such that

$$A^\ell = [A^\ell_\Omega, A^\ell_\Theta]E^\ell. \tag{2.11}$$

According to the above discussion, we first associate A^ℓ with a new partition vector Π^ℓ. We let

$$\Pi^\ell := [n_{\ell,4i-3},\ n_{\ell,4i-2} + n_{\ell,4i-1},\ n_{\ell,4i} : i \in \mathbb{N}_{\gamma_\ell/4}].$$

Next we define Ω^ℓ, Θ^ℓ, A^ℓ_Ω, and A^ℓ_Θ from $A^\ell(\Pi^\ell)$ using the above approach. That is, for each $i \in \mathbb{N}$, we let $s_i := 3i-1$, $t_{2i-1} := 3i-2$, $t_{2i} := 3i$. Let $\Omega^\ell := [s_i : i \in \mathbb{N}_{\gamma_\ell/4}]$, $\Theta^\ell := [t_i : i \in \mathbb{N}_{\gamma_\ell/2}]$, and let Π^ℓ_Ω, Π^ℓ_Θ be defined by (2.4), with the understanding that the subscripts Ω, Θ depend on ℓ. Then we define by (2.5) the submatrices A^ℓ_Ω and A^ℓ_Θ of $A^\ell(\Pi^\ell)$. An example of the construction is illustrated in Figure 2.2. Lastly, let

$m_\ell := \sum_{i\in\mathbb{N}_{\gamma_\ell}} n_{\ell,i}$, we construct the matrix $E^\ell \in \mathbb{R}^{m_\ell\times m_\ell}$ by

$$E_1^\ell := \bigoplus_{i\in\mathbb{N}_{\gamma_\ell/4}} \left(\left[0_{n_{\ell,4i-2}\times n_{\ell,4i-3}},\ I_{n_{\ell,4i-2}}\right] \oplus \left[I_{n_{\ell,4i-1}},\ 0_{n_{\ell,4i-1}\times n_{\ell,4i}}\right]\right),$$

$$E_2^\ell := \bigoplus_{i\in\mathbb{N}_{\gamma_\ell/4}} \left(\left[I_{n_{\ell,4i-3}},\ 0_{n_{\ell,4i-3}\times n_{\ell,4i-2}}\right] \oplus \left[0_{n_{\ell,4i}\times n_{\ell,4i-1}},\ I_{n_{\ell,4i}}\right]\right),$$

$$E^\ell := \left[(E_1^\ell)^T, (E_2^\ell)^T\right]^T.$$

It can be verified that E^ℓ is a permutation matrix, which fulfils the equation (2.11).

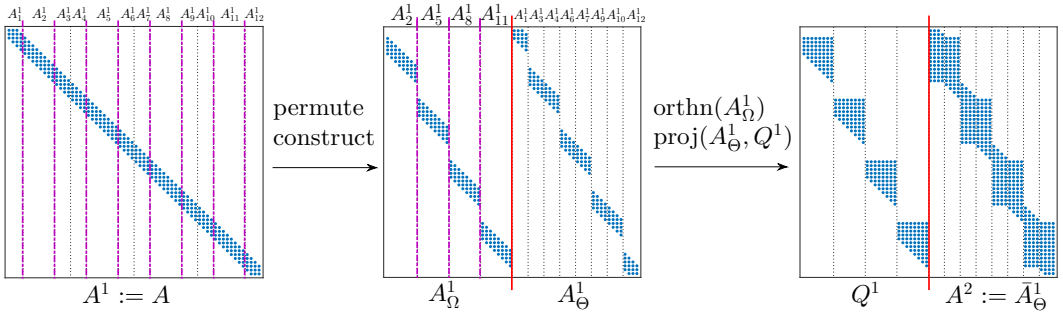


FIGURE 2.2: *The first step of BGSP applied to a pentadiagonal matrix A.*

Finally, we propose the process BGSP to construct the block QS factorization for a matrix. Throughout the rest of thesis, unless specified otherwise, the matrices A^ℓ, A^ℓ_Ω, Q^ℓ, and A^ℓ_Θ in BGSP below are partitioned by Λ^ℓ, Π^ℓ_Ω, Π^ℓ_Ω, and Π^ℓ_Θ, respectively. Suppose that $A \in \mathbb{R}^{n\times m}$ is a matrix of full rank, partitioned by $\Lambda = [n_i : i \in \mathbb{N}_\gamma]$ with $\gamma = 2^L$ for some $L \in \mathbb{N}$.

Let $A^1 := A$, $\Lambda^1 := \Lambda$. For each $\ell \in \mathbb{N}_{L-1}$,

construct matrices Q^ℓ, R^ℓ by orthn(A^ℓ_Ω),

construct matrices $A^{\ell+1}$, C^ℓ by proj(A^ℓ_Θ, Q^ℓ),

let $\Lambda^{\ell+1} := \Pi^\ell_\Theta$.

Construct matrices Q^L, R^L by MGS from A^L.

To construct the factor matrices Q and S, we define

$$Q := [Q^\ell : \ell \in \mathbb{N}_L]. \tag{2.12}$$

If $L = 1$, let $S := R^1$. If $L > 1$, we define $F^\ell \in \mathbb{R}^{m\times m}$ by

$$F^\ell := \begin{cases} E^1, & \ell = 1, \\ \left[\prod_{j\in\mathbb{N}_{\ell-1}} (I \oplus E^{\ell-j+1})\right] E^1, & \ell \in \mathbb{N}_{L-1}\backslash\{1\}, \end{cases} \tag{2.13}$$

and let

$$S^\ell := \begin{cases} [R^1, C^1]F^1, & \ell = 1, \\ \left[0,\ R^\ell,\ C^\ell\right] F^\ell, & \ell \in \mathbb{N}_{L-1}\backslash\{1\}, \\ \left[0,\ R^L\right] F^{L-1}, & \ell = L. \end{cases} \tag{2.14}$$

Then we define

$$S := \left[(S^\ell)^T : \ell \in \mathbb{N}_L\right]^T. \tag{2.15}$$

We construct the permutation matrix E by

$$E := \begin{cases} I_{m\times m}, & L = 1, \\ (F^{L-1})^T, & L > 1. \end{cases} \tag{2.16}$$

Notice that E is a permutation matrix. We shall verify in the next section that the matrices Q, S, E constructed by the above process from A satisfy that QS is the block QS factorization of A, and specifically, SE is upper triangular. As an example, we demonstrate BGSP applied to a pentadiagonal matrix A in Figures 2.2 and 2.3.

To close this section, we comment that the proposed process is favorable to parallel computing. In parallel implementation, orthn$(\cdot)$ and proj$(\cdot,\cdot)$ can be executed completely in parallelism. To see this, given $\ell \in \mathbb{N}_{L-1}$, for $i \in \mathbb{N}_{3\gamma_\ell/4}$, we let A_i^ℓ be the

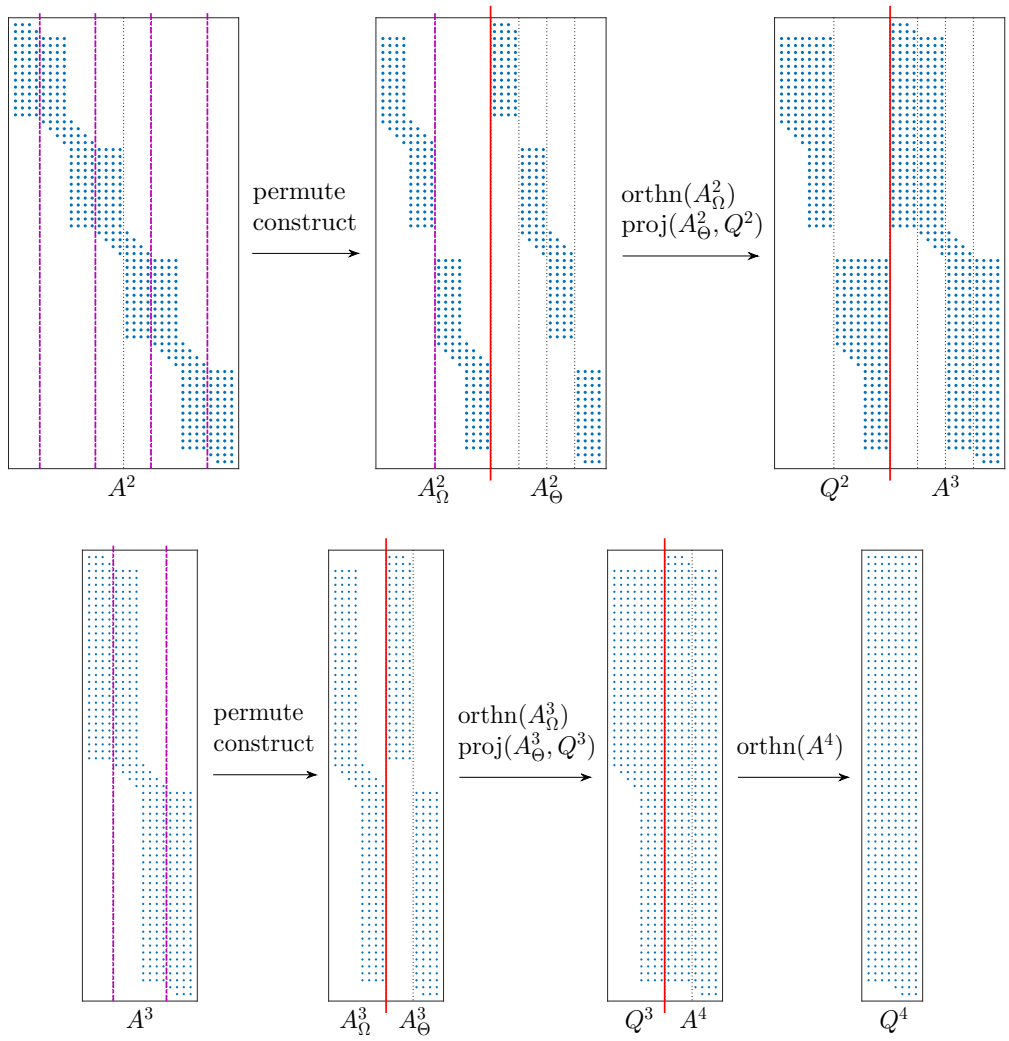


FIGURE 2.3: *BGSP applied to a pentadiagonal matrix A, from the second step.*

submatrix of A^ℓ such that

$$A^\ell(\Pi^\ell) = [A^\ell_i : i \in \mathbb{N}_{3\gamma_\ell/4}]. \tag{2.17}$$

Then as long as a processor holds the submatrix $M := [A^\ell_{3i-2}, A^\ell_{3i-1}, A^\ell_{3i}]$ of A^ℓ, it can independently construct the column orthonormal matrix Q^ℓ_i by MGS from A^ℓ_{3i-1}, and apply $\mathrm{proj}([A^\ell_{3i-2}, A^\ell_{3i}], Q^\ell_i)$. Furthermore, by tracing the calculations, we can figure out the column blocks of the input matrix $A(\Lambda)$ of BGSP that generates M. Hence, we can avoid communication once we put these column blocks together and assign as

a unit to a processor. In this way, we expect that the parallel implementation will have low communication costs, yielding good scalability. We shall study the parallel implementation in chapter 3.

2.2 The Orthonormal Factorization

In this section, we show that the matrices Q and S constructed by BGSP from a block-wise quasi-orthogonal matrix $A(\Lambda)$ of full rank form the block QS factorization of A. Specifically, we verify that Q is column orthonormal, SE is upper triangular, and $A = QS$. Note that although the matrices Q, S and E may depend on the partition vector Λ of A, we shall use the notation without indicating the dependence for notational simplicity.

We first show the column orthonormality of Q. This will be done in two steps. First, we verify the orthogonalization process for each ℓ: Given a block-wise quasi-orthogonal matrix A^ℓ, the matrix Q^ℓ constructed by $\mathrm{orthn}(A^\ell_\Omega)$ is column orthogonal, and the matrix $A^{\ell+1}$ constructed by $\mathrm{proj}(A^\ell_\Theta, Q^\ell)$ has the range space that is the orthogonal complement of the range space of Q^ℓ in the range space of A^ℓ. Second, we show that each A^ℓ inherits the block-wise quasi-orthogonality from A, and then prove the result by induction.

We present the results for $\mathrm{orthn}(A^\ell_\Omega)$. To this end, we establish a technical lemma below. Henceforth, for a matrix $A \in \mathbb{R}^{n\times m}$, we define the range space of A by $\mathscr{R}(A) := \{Av : v \in \mathbb{R}^m\}$.

Lemma 2.2.1. *Let $A = [A_i : i \in \mathbb{N}_\gamma]$ be a matrix of full rank, and let Q, R be*

the matrices constructed by orthn(A). *If A is block-wise orthogonal, then $A = QR$, $Q^TQ = I$, $\mathscr{R}(Q) = \mathscr{R}(A)$, and R is upper triangular.*

Proof. This result follows directly from the definition of the block-wise orthogonality and MGS. For each $i \in \mathbb{N}_\gamma$, since A is of full rank, so is A_i. Since Q_i, R_i are constructed by MGS from A_i, we have that $A_i = Q_iR_i$, $Q_i^TQ_i = I$, and R_i is invertible upper triangular. Formula (2.6) leads to

$$A = [A_i : i \in \mathbb{N}_\gamma] = [Q_iR_i : i \in \mathbb{N}_\gamma] = QR.$$

By the definition of R and the fact that R_i is invertible upper triangular, we observe that R is invertible upper triangular. It follows from the equation $A = QR$ and the definition of $\mathscr{R}(\cdot)$ that $\mathscr{R}(Q) = \mathscr{R}(A)$.

It remains to prove that $Q^TQ = I$. Since $Q_i^TQ_i = I$, it suffices to show for $i, j \in \mathbb{N}_\gamma$ with $i \neq j$ that $Q_i^TQ_j = 0$. By the block-wise orthogonality of A and the fact that R_i, R_j are invertible, we obtain that

$$Q_i^TQ_j = R_i^{-T}(A_i^TA_j)R_j^{-1} = 0.$$

Hence, we have that $Q^TQ = I$. □

To apply Lemma 2.2.1, we need the following remark to ensure the block-wise quasi-orthogonality of $A^\ell(\Pi^\ell)$.

***Remark* 2.** For $\Lambda = [n_i : i \in \mathbb{N}_\gamma]$, let $\Lambda_i := [n_1, \ldots, n_{i-1}, n_i + n_{i+1}, n_{i+2}, \ldots, n_\gamma]$, for $i \in \mathbb{N}_{\gamma-1}$. If $A(\Lambda)$ is block-wise quasi-orthogonal, then for each $i \in \mathbb{N}_{\gamma-1}$, $A(\Lambda_i)$ is block-wise quasi-orthogonal.

The proposition below confirms the fulfilment of orthn(A_Ω^ℓ).

Proposition 2.2.2. *Let $\ell \in \mathbb{N}_{L-1}$ be given. If A^ℓ is a block-wise quasi-orthogonal matrix of full rank, and matrices Q^ℓ, R^ℓ are constructed by* $\mathrm{orthn}(A^\ell_\Omega)$*, then $A^\ell_\Omega = Q^\ell R^\ell$, $(Q^\ell)^T Q^\ell = I$, $\mathscr{R}(Q^\ell) = \mathscr{R}(A^\ell_\Omega)$, and R^ℓ is upper triangular.*

Proof. Since A^ℓ is of full rank and A^ℓ_Ω is a submatrix of A^ℓ, we conclude that A^ℓ_Ω is of full rank. Since $A^\ell(\Lambda^\ell)$ is block-wise quasi-orthogonal, so is $A^\ell(\Pi^\ell)$ by Remark 2. It follows from the definition of Ω^ℓ and Remark 1 that A^ℓ_Ω is block-wise orthogonal. Hence by Lemma 2.2.1, we obtain the desired results. □

We next show the results for $\mathrm{proj}(A^\ell_\Theta, Q^\ell)$. To this end, we establish two technical lemmas for the process $\mathrm{proj}(\cdot,\cdot)$. The first lemma presents a result regarding $\mathrm{proj}(A_0, Q_0)$.

Lemma 2.2.3. *Let matrices A_0, $Q_0 := [q_i : i \in \mathbb{N}_t]$ be given, and let $\bar{A}_0$ be the matrix constructed by* $\mathrm{proj}(A_0, Q_0)$*. If $Q_0^T Q_0 = I$, then*

$$\bar{A}_0 = (I - Q_0 Q_0^T) A_0. \tag{2.18}$$

Proof. Using the hypothesis that $Q_0^T Q_0 = I$, it can be verified that

$$\prod_{i \in \mathbb{N}_t} \left(I - q_{t+1-i}\, q_{t+1-i}^T\right) = I - Q_0 Q_0^T.$$

By the construction (2.7) of $\bar{A}_0$, we obtain the desired formula (2.18). □

We present the lemma below regarding the matrix $\bar{A}$ constructed by $\mathrm{proj}(A, Q)$.

Lemma 2.2.4. *Let matrices $A = [A_j : j \in \mathbb{N}_{2\gamma}] \in \mathbb{R}^{n\times s}$, $Q = [Q_i : i \in \mathbb{N}_\gamma]$ be given, and let $\bar{A}$ be the matrix constructed by* $\mathrm{proj}(A, Q)$*. If $Q^T Q = I$, and for $i \in \mathbb{N}_\gamma$, $j \in \mathbb{N}_{2\gamma}$ with $i \neq \lceil j/2 \rceil$, $Q_i^T A_j = 0$, then $\bar{A} \in \mathbb{R}^{n\times s}$, $Q^T \bar{A} = 0$, and $\mathscr{R}([\bar{A}, Q]) = \mathscr{R}([A, Q])$.*

Proof. We apply Lemma 2.2.3 to each column block of $\bar{A}$. By the construction (2.9) of $\bar{A}$, we obtain that $\bar{A} \in \mathbb{R}^{n\times s}$. To prove the remaining two results, we first establish the equation

$$\bar{A} = (I - QQ^T)A \tag{2.19}$$

using Lemma 2.2.3. Given $j \in \mathbb{N}_{2\gamma}$, let $r := \lceil j/2 \rceil$. Since $Q^TQ = I$, we have that $Q_r^TQ_r = I$. Since $\bar{A}_j$ is constructed by $\mathrm{proj}(A_j, Q_r)$, by Lemma 2.2.3, we obtain that $\bar{A}_j = (I - Q_rQ_r^T)A_j$. Since for $i \in \mathbb{N}_\gamma$ with $i \neq r$, $Q_i^TA_j = 0$, we have that

$$\bar{A}_j = (I - Q_rQ_r^T)A_j + \sum_{i\in\mathbb{N}_\gamma, i\neq r} Q_i(Q_i^TA_j) = (I - QQ^T)A_j.$$

Equation (2.19) follows from the formula above and (2.9).

Since $Q^TQ = I$, by direct computation using (2.19), we have that $Q^T\bar{A} = 0$. Formula (2.19) yields that

$$\bar{A} = A - Q(Q^TA) \quad \text{and} \quad A = \bar{A} + Q(Q^TA).$$

Hence, we obtain from the definition of $\mathscr{R}(\cdot)$ that $\mathscr{R}(\bar{A}) \subset \mathscr{R}([A, Q])$ and $\mathscr{R}(A) \subset \mathscr{R}([\bar{A}, Q])$. Therefore, $\mathscr{R}([\bar{A}, Q]) = \mathscr{R}([A, Q])$. □

We then verify in the following proposition the fulfilment of $\mathrm{proj}(A_\Theta^\ell, Q^\ell)$ for $\ell \in \mathbb{N}_{L-1}$. Notice that by formula (2.17), we have that $A_\Omega^\ell = [A_{s_i}^\ell : i \in \mathbb{N}_{\gamma_\ell/4}]$ and $A_\Theta^\ell = [A_{t_j}^\ell : j \in \mathbb{N}_{\gamma_\ell/2}]$. For $i \in \mathbb{N}_{\gamma_\ell/4}$, let Q_i^ℓ denote the submatrix of Q^ℓ such that

$$Q^\ell(\Pi_\Omega^\ell) = [Q_i^\ell : i \in \mathbb{N}_{\gamma_\ell/4}]. \tag{2.20}$$

Proposition 2.2.5. *Let $\ell \in \mathbb{N}_{L-1}$ be given. If A^ℓ is a block-wise quasi-orthogonal matrix of full rank, then $(Q^\ell)^TA^{\ell+1} = 0$ and $\mathscr{R}(A^\ell) = \mathscr{R}(Q^\ell) \oplus \mathscr{R}(A^{\ell+1})$.*

Proof. We shall prove this result by employing Lemma 2.2.4, whose assumptions will

be verified by using Proposition 2.2.2 and Remark 1.

Since A^ℓ is a block-wise quasi-orthogonal matrix of full rank, by Proposition 2.2.2, we have that $(Q^\ell)^T Q^\ell = I$. To apply Lemma 2.2.4, by the construction of $A^{\ell+1}$, we shall show for $r \in \mathbb{N}_{\gamma_\ell/4}$, $j \in \mathbb{N}_{\gamma_\ell/2}$ with $r \neq \lceil j/2 \rceil$ that

$$(Q_r^\ell)^T A_{t_j}^\ell = 0. \tag{2.21}$$

Notice that $A^\ell(\Pi^\ell)$ is block-wise quasi-orthogonal. By Remark 1, we have that $(A_{s_r}^\ell)^T A_{t_j}^\ell = 0$. It follows from the construction of Q^ℓ and formula (2.20) that for each $i \in \mathbb{N}_{\gamma_\ell/4}$,

$$Q_i^\ell = A_{s_i}^\ell (R_i^\ell)^{-1}, \tag{2.22}$$

for an invertible upper triangular matrix R_i^ℓ. Formula (2.22) and $(A_{s_r}^\ell)^T A_{t_j}^\ell = 0$ lead to (2.21). Since $(Q^\ell)^T Q^\ell = I$, by Lemma 2.2.4, we have that $(Q^\ell)^T A^{\ell+1} = 0$, and $\mathscr{R}([A^{\ell+1}, Q^\ell]) = \mathscr{R}([A_\Theta^\ell, Q^\ell])$. Thus,

$$\mathscr{R}(A^\ell) = \mathscr{R}([A_\Theta^\ell, A_\Omega^\ell]) = \mathscr{R}([A_\Theta^\ell, Q^\ell]) = \mathscr{R}([A^{\ell+1}, Q^\ell]) = \mathscr{R}(Q^\ell) \oplus \mathscr{R}(A^{\ell+1}), \tag{2.23}$$

proving the desired result. □

The next proposition confirms the inheritance of the block-wise quasi-orthogonality of A^ℓ from A. To this end, given $\ell \in \mathbb{N}_L$, for $i \in \mathbb{N}_{\gamma_\ell}$, let G_i^ℓ denote the submatrix of A^ℓ such that

$$A^\ell(\Lambda^\ell) = [G_i^\ell : i \in \mathbb{N}_{\gamma_\ell}]. \tag{2.24}$$

Proposition 2.2.6. *If A is a block-wise quasi-orthogonal matrix of full rank, then for each $\ell \in \mathbb{N}_L$, A^ℓ is a block-wise quasi-orthogonal matrix of full rank.*

Proof. The proof is done by induction on $\ell \in \mathbb{N}_L$ using previous propositions. Indeed,

when $\ell = 1$, $A^1 = A$ is a block-wise quasi-orthogonal matrix of full rank. Assuming for $\ell \in \mathbb{N}_{L-1}$ that A^ℓ is a block-wise quasi-orthogonal matrix of full rank, we shall show that $A^{\ell+1}$ is a block-wise quasi-orthogonal matrix of full rank.

We first prove that $A^{\ell+1}$ is of full rank. Since $A^{\ell+1}$ is constructed by $\text{proj}(A^\ell_\Theta, Q^\ell)$, by Lemma 2.2.4, the column size of $A^{\ell+1}$ is the same as that of A^ℓ_Θ. It follows from formula (2.23) that the rank of $A^{\ell+1}$ is the difference between the ranks of A^ℓ and Q^ℓ. Since A^ℓ is of full rank, the rank of $A^{\ell+1}$ equals the difference between the column sizes of A^ℓ and A^ℓ_Ω, which equals the column size of A^ℓ_Θ. Hence, $A^{\ell+1}$ is of full rank.

It remains to show that $A^{\ell+1}(\Lambda^{\ell+1})$ is block-wise quasi-orthogonal. By (2.24), it suffices to show that $(G^{\ell+1}_\mu)^T G^{\ell+1}_\nu = 0$, for $\mu, \nu \in \mathbb{N}_{\gamma_\ell/2}$ with $\nu - \mu > 1$. To this end, we express $G^{\ell+1}_\mu$ and $G^{\ell+1}_\nu$ in column blocks of $A^\ell(\Pi^\ell)$, and then use the block-wise quasi-orthogonality of $A^\ell(\Pi^\ell)$ to conclude the result.

We express $G^{\ell+1}_j$ in terms of $A^\ell_{t_j}$. For $j \in \mathbb{N}_{\gamma_\ell/2}$, let $r := \lceil j/2 \rceil$. By the construction of $A^{\ell+1}$, we have that $G^{\ell+1}_j$ is constructed by $\text{proj}(A^\ell_{t_j}, Q^\ell_r)$. Since $(Q^\ell)^T Q^\ell = I$ by Proposition 2.2.2, we have that $(Q^\ell_r)^T Q^\ell_r = I$. By Lemma 2.2.3, we obtain that

$$G^{\ell+1}_j = A^\ell_{t_j} - Q^\ell_r (Q^\ell_r)^T A^\ell_{t_j}. \tag{2.25}$$

By formula (2.22), we have that $Q^\ell_r = A^\ell_{s_r}(R^\ell_r)^{-1}$. Let $M^\ell_j := (R^\ell_r)^{-1}(Q^\ell_r)^T A^\ell_{t_j}$. Substituting the equations above into (2.25) gives that

$$G^{\ell+1}_j = A^\ell_{t_j} - A^\ell_{s_r} M^\ell_j. \tag{2.26}$$

Since $r = \lceil j/2 \rceil$, by the definition of t_j, we have that if $j = 2r - 1$, then $t_j = 3r - 2$,

and if $j = 2r$, then $t_j = 3r$. With $s_r = 3r - 1$, employing formula (2.26), we have that

$$G_j^{\ell+1} = \begin{cases} A_{3r-2}^{\ell} - A_{3r-1}^{\ell} M_j^{\ell}, & j = 1, 3, \ldots, \gamma_\ell/2 - 1, \\ A_{3r}^{\ell} - A_{3r-1}^{\ell} M_j^{\ell}, & j = 2, 4, \ldots, \gamma_\ell/2. \end{cases} \tag{2.27}$$

Now we show that $(G_\mu^{\ell+1})^T G_\nu^{\ell+1} = 0$. For $j \in \{\mu, \nu\}$, we let $r := \lceil j/2 \rceil$, and let $\mathbb{K}_j$ be the set $\{3r-2, 3r-1\}$ if j is odd, and the set $\{3r-1, 3r\}$ if j is even. For each $r_1 \in \mathbb{K}_\mu$, $r_2 \in \mathbb{K}_\nu$, since $\nu - \mu > 1$, it can be verified that $r_2 - r_1 > 1$. By the block-wise quasi-orthogonality of $A^\ell(\Pi^\ell)$, we have that $(A_{r_1}^\ell)^T A_{r_2}^\ell = 0$. It follows from formula (2.27) and direct computation that $(G_\mu^{\ell+1})^T G_\nu^{\ell+1} = 0$. This ensures that $A^{\ell+1}$ is block-wise quasi-orthogonal.

By the induction principle, we obtain the desired result. □

In the following theorem, we verify that Q is column orthonormal and has the same range space as A.

Theorem 2.2.7. *If $A(\Lambda)$ is a block-wise quasi-orthogonal matrix of full rank, where Λ is of length $\gamma := 2^L$ for some $L \in \mathbb{N}$, then the matrix Q constructed by BGSP from A satisfies that $Q^T Q = I$ and $\mathscr{R}(Q) = \mathscr{R}(A)$.*

Proof. To show that $Q^T Q = I$, by (2.12), we shall prove that each Q^ℓ is column orthonormal and Q^ℓ's are orthogonal to each other. This will be done by applying Propositions 2.2.2 and 2.2.5 for each ℓ. The result that $\mathscr{R}(Q) = \mathscr{R}(A)$ will follow from an induction using Proposition 2.2.5. Indeed, since A is of full rank, if $L = 1$, then according to BGSP, we have that Q^1 is constructed by MGS from A, and $Q = Q^1$. Thus, we obtain the desired results in this case. It remains to consider $L > 1$.

We first show that Q is column orthonormal. With formula (2.12), it suffices to

show that for $\ell \in \mathbb{N}_L$,

$$(Q^\ell)^T Q^\ell = I, \tag{2.28}$$

and for $j, \ell \in \mathbb{N}_L$ with $j < \ell$,

$$(Q^j)^T Q^\ell = 0. \tag{2.29}$$

Since A is a block-wise quasi-orthogonal matrix of full rank, by Proposition 2.2.6, we have for $\ell \in \mathbb{N}_L$ that A^ℓ is a block-wise quasi-orthogonal matrix of full rank. Notice that Q^L is constructed by MGS from A^L, and hence $(Q^L)^T Q^L = I$. This together with Proposition 2.2.2, we obtain formula (2.28).

We next prove (2.29). Since Q^ℓ is constructed from A^ℓ_Ω, which is a submatrix of A^ℓ, it suffices to show for $j \in \mathbb{N}_{L-1}$, $\ell \in \mathbb{N}_L$ with $j < \ell$ that

$$(Q^j)^T A^\ell = 0. \tag{2.30}$$

This will be done by induction on ℓ. Indeed, when $\ell = j + 1$, by Proposition 2.2.5, we have that $(Q^j)^T A^{j+1} = 0$. Assume that (2.30) is true for ℓ with $j < \ell \leq L - 1$. Again by Proposition 2.2.5, we have that $\mathscr{R}(A^{\ell+1}) \subset \mathscr{R}(A^\ell)$, and thus by the induction hypothesis, we have that $(Q^j)^T A^{\ell+1} = 0$. By the induction principle, we obtain formula (2.30). Proposition 2.2.5 and the fact that $\mathscr{R}(Q^L) = \mathscr{R}(A^L)$ yield for each $\ell \in \mathbb{N}_L$ that $\mathscr{R}(Q^\ell) \subset \mathscr{R}(A^\ell)$. Together with formula (2.30), we obtain (2.29). Therefore, $Q^T Q = I$.

We now prove that $\mathscr{R}(Q) = \mathscr{R}(A)$. It suffices to show that

$$\mathscr{R}(A) = \bigoplus_{\ell \in \mathbb{N}_L} \mathscr{R}(Q^\ell). \tag{2.31}$$

This will be done by proving the following formula by induction on $\ell \in \mathbb{N}_{L-1}$,

$$\mathscr{R}(A) = \left(\bigoplus_{j \in \mathbb{N}_\ell} \mathscr{R}(Q^j) \right) \oplus \mathscr{R}(A^{\ell+1}). \tag{2.32}$$

Indeed, when $\ell = 1$, since $A = A^1$, by Proposition 2.2.5, we obtain the result. Assume that (2.32) is true for $\ell \in \mathbb{N}_{L-2}$. Since $(\ell + 1) \in \mathbb{N}_{L-1}$, it follows from Proposition 2.2.5 that $\mathscr{R}(A^{\ell+1}) = \mathscr{R}(Q^{\ell+1}) \oplus \mathscr{R}(A^{\ell+2})$. Together with the induction hypothesis, we have that (2.32) is true for $(\ell + 1)$. Hence, by the induction principle, we obtain formula (2.32). Substituting $\mathscr{R}(Q^L) = \mathscr{R}(A^L)$ into formula (2.32) for $\ell = L - 1$ yields (2.31). This ensures that $\mathscr{R}(Q) = \mathscr{R}(A)$. □

Next, we verify in the following theorem that SE is upper triangular.

Theorem 2.2.8. *Let $A(\Lambda) \in \mathbb{R}^{n \times m}$ be a block-wise quasi-orthogonal matrix of full rank, where $n \geq m$ and Λ is of length $\gamma := 2^L$ for some $L \in \mathbb{N}$. If S, E are the matrices constructed by BGSP from A. then SE is upper triangular.*

Proof. The proof follows from the construction of S, E, and the upper triangularity of each R^ℓ. If $L = 1$, then $E = I$. By BGSP, we have that $S = R^1$ and R^1 is the upper triangular matrix constructed by MGS. Thus SE is upper triangular.

We next prove the upper triangularity of SE for $L > 1$. Using the construction of S, E by BGSP, we directly compute below the product $S^\ell (F^{L-1})^T$. By the fact that $E^1 (E^1)^T = I$, we have that

$$S^1 (F^{L-1})^T = [R^1, C^1] \prod_{j \in \mathbb{N}_{L-2}} \left(I \oplus (E^{j+1})^T \right) = [R^1, C^1 H^1],$$

for a square matrix H^1. For $\ell \in \mathbb{N}_{L-1}$, likewise, since $E^\ell (E^\ell)^T = I$, we obtain that

$$S^\ell (F^{L-1})^T = \left[0_{m_\ell \times (m - m_\ell)}, R^\ell, C^\ell H^\ell \right],$$

for a square matrix H^ℓ. Notice that $S^{L-1}(F^{L-1})^T = [0, R^L]$, and for each $\ell \in \mathbb{N}_L$, R^ℓ is upper triangular. It follows from observation with the above three equations, and formulas (2.15), (2.16) that SE is upper triangular. □

Finally, we verify the equation $A = QS$. We will first prove the equation

$$A^\ell_\Theta = A^{\ell+1} + Q^\ell C^\ell. \tag{2.33}$$

Together with $A^\ell_\Omega = Q^\ell R^\ell$ by Proposition 2.2.2, we obtain an expression of A^ℓ in terms of Q^ℓ, R^ℓ, C^ℓ, and $A^{\ell+1}$. Then we will conclude that $A = QS$ by induction and the construction of Q and S. To show (2.33), we present the following technical lemma.

Lemma 2.2.9. *Let matrices $A := [A_j : j \in \mathbb{N}_{2\gamma}]$ and $Q := [Q_i : i \in \mathbb{N}_\gamma]$ be given, and let $\bar{A}$, C be the matrices constructed by* $\mathrm{proj}(A, Q)$. *If $Q^TQ = I$, then $A = \bar{A} + QC$.*

Proof. The proof follows from Lemma 2.2.3. By equation (2.9), it suffices to show for $j \in \mathbb{N}_{2\gamma}$, $r := \lceil j/2 \rceil$ that

$$A_j = \bar{A}_j + Q_r C_j. \tag{2.34}$$

Since $Q^TQ = I$, $Q_r^TQ_r = I$. It follows from direct computation with (2.8) and the construction of C_j that $C_j = Q_r^T A_j$. Again, since $Q_r^TQ_r = I$, by Lemma 2.2.3, we obtain formula (2.34). Therefore, $A = \bar{A} + QC$. □

The following lemma expresses A^ℓ in terms of Q^ℓ, R^ℓ, C^ℓ, and $A^{\ell+1}$.

Lemma 2.2.10. *If A is a block-wise quasi-orthogonal matrix of full rank, then for each $\ell \in \mathbb{N}_{L-1}$,*

$$A^\ell = Q^\ell[R^\ell, C^\ell]E^\ell + [0, A^{\ell+1}]E^\ell.$$

Proof. Since A is a block-wise quasi-orthogonal matrix of full rank, by Proposition 2.2.6, we have for each $\ell \in \mathbb{N}_{L-1}$ that A^ℓ is a block-wise quasi-orthogonal matrix of full rank. By Proposition 2.2.2, we have that $A^\ell_\Omega = Q^\ell R^\ell$ and $(Q^\ell)^T Q^\ell = I$. Since $A^{\ell+1}$, C^ℓ are constructed by $\mathrm{proj}(A^\ell_\Theta, Q^\ell)$, by Lemma 2.2.9, we obtain formula (2.33). Together with formula (2.11), we obtain the desired result. □

We next prove that $A = QS$.

Theorem 2.2.11. *If $A(\Lambda) \in \mathbb{R}^{n\times m}$ is a block-wise quasi-orthogonal matrix of full rank, where $n \geq m$ and Λ is of length $\gamma := 2^L$ for some $L \in \mathbb{N}$, then the matrices Q, S constructed by BGSP from A satisfy that $Q \in \mathbb{R}^{n\times m}$, $S \in \mathbb{R}^{m\times m}$, and $A = QS$.*

Proof. We first verify the matrix sizes of Q and S. By Theorem 2.2.7, we have that $Q^T Q = I$ and $\mathscr{R}(Q) = \mathscr{R}(A)$. Thus, Q is of full rank. Since $A \in \mathbb{R}^{n\times m}$ is of full rank, and $n \geq m$, so $Q \in \mathbb{R}^{n\times m}$. Hence it suffices to show that $A = QS$, since by verifying the matrix sizes, we have that $S \in \mathbb{R}^{m\times m}$. For the proof of $A = QS$, if $L = 1$, then according to BGSP, we obtain that $A = Q^1 R^1 = QS$.

Now we prove that $A = QS$ for $L > 1$. By the definitions (2.12) and (2.15), it suffices to show that

$$A = \sum_{\ell\in\mathbb{N}_L} Q^\ell S^\ell. \tag{2.35}$$

This will be done by proving the following formula by induction on $\ell \in \mathbb{N}_{L-1}$,

$$A = \sum_{j\in\mathbb{N}_\ell} Q^j S^j + [0, A^{\ell+1}]F^\ell. \tag{2.36}$$

Indeed, when $\ell = 1$, since A is a block-wise quasi-orthogonal matrix of full rank, by

Lemma 2.2.10, formulas (2.13) and (2.14), we have that

$$A^1 = Q^1[R^1, C^1]E^1 + [0, A^2]E^1 = Q^1 S^1 + [0, A^2]F^1.$$

Assuming that (2.36) is true for $\ell \in \mathbb{N}_{L-2}$, we shall show that

$$A = \sum_{j \in \mathbb{N}_{\ell+1}} Q^j S^j + [0, A^{\ell+2}]F^{\ell+1}. \tag{2.37}$$

Since $(\ell + 1) \in \mathbb{N}_{L-1}$, again by Lemma 2.2.10, we have that

$$A^{\ell+1} = Q^{\ell+1}[R^{\ell+1}, C^{\ell+1}]E^{\ell+1} + [0, A^{\ell+2}]E^{\ell+1}. \tag{2.38}$$

Equation (2.37) follows from the induction hypothesis, formulas (2.38), (2.13) and (2.14). Hence, by the induction principle, we obtain formula (2.36). Since Q^L, R^L are constructed by MGS from A^L, we have that $A^L = Q^L R^L$. This together with (2.36) for $\ell = L - 1$, and equation (2.14) yields formula (2.35). Thus, we have that $A = QS$. □

2.3 Sparsity of the Factor Matrices

We establish in this section the sparsity of the matrices Q and S constructed by BGSP from a banded matrix A of full rank. Specifically, we prove that if $A(\Lambda) \in \mathbb{R}^{n \times m}$ has bandwidth $k/2$, $n \geq m$, $k \ll m$, and $\Lambda = [k, k, \ldots, k]$, then the numbers of nonzero entries of Q and S are $O(m \log m)$ and $O(m)$, respectively. Note that the numbers of nonzero entries of Q and S depend on the sparse structure of A, and its partition vector Λ. For simplicity, we consider only the case mentioned above. We denote by $\#(A)$ the number of nonzero entries of A.

The goal of this section is to count the numbers of nonzero entries of the factor matrices constructed by BGSP from a matrix A that satisfies the following hypothesis.

Hypothesis (A). Matrix $A \in \mathbb{R}^{n\times m}$ is of full rank and is banded with bandwidth $k/2$, $n \geq m$, $m = 2^L k$ for $L \in \mathbb{N}$, $\gamma := 2^L$, and A is partitioned by the vector $\Lambda := [k, k, \ldots, k]$ of length γ.

We first count $\#(Q)$. By the construction of Q, we may trace the sparse structure of A^ℓ. By formulas (2.12) and (2.20), $\#(Q)$ is the sum of $\#(Q_i^\ell)$ over i, ℓ. Since Q_i^ℓ is the column orthonormal matrix constructed by MGS from $A_{s_i}^\ell$, $\#(Q_i^\ell)$ depends on the sparse structure of $A_{s_i}^\ell$. By examining the calculations, it can be seen that the nonzero entries of $A_{s_i}^\ell$ are clustered in a submatrix of it. For this reason, we introduce two functions $\underline{\eta}(\cdot)$ and $\bar{\eta}(\cdot)$ in the following to describe properties of the submatrix. Given a matrix $A := [a_{ij} : i \in \mathbb{N}_n, j \in \mathbb{N}_m]$, $A \neq 0$, we let $\mathbb{K} := \{(i,j) : i \in \mathbb{N}_n, j \in \mathbb{N}_m, \text{ and } a_{ij} \neq 0\}$, and define

$$\underline{\eta}(A) := \min\{i : (i,j) \in \mathbb{K}\}, \qquad \bar{\eta}(A) := \max\{i : (i,j) \in \mathbb{K}\}.$$

If $A = 0$, then $\mathbb{K}$ is empty. In this case, we let $\underline{\eta}(A) := n + 1$ and $\bar{\eta}(A) := 0$. For a vector $v \in \mathbb{R}^n$, we also define $\underline{\eta}(v)$ and $\bar{\eta}(v)$ by viewing v as a $n \times 1$ matrix. With these definitions, we have that

$$\#(A) \leq \max\{\bar{\eta}(A) - \underline{\eta}(A) + 1,\ 0\} \cdot m. \tag{2.39}$$

As a preparation, we investigate below properties of $\underline{\eta}(\cdot)$ and $\bar{\eta}(\cdot)$.

Lemma 2.3.1. *Let matrices $A := [a_{ij} : i \in \mathbb{N}_n, j \in \mathbb{N}_m]$, $B := [b_{ij} : i \in \mathbb{N}_n, j \in \mathbb{N}_m]$, $C := [c_{ij} : i \in \mathbb{N}_n, j \in \mathbb{N}_s]$, $D := [d_{ij} : i \in \mathbb{N}_n, j \in \mathbb{N}_t]$, and $M := [m_{ij} : i \in \mathbb{N}_{s+t}, j \in \mathbb{N}_m]$ be given. Let $\eta_1 := \min\{\underline{\eta}(B), \underline{\eta}(C), \underline{\eta}(D)\}$, and $\eta_2 := \max\{\bar{\eta}(B), \bar{\eta}(C), \bar{\eta}(D)\}$. If $A = B + [C, D]M$, then $\underline{\eta}(A) \geq \eta_1$ and $\bar{\eta}(A) \leq \eta_2$.*

Proof. Given $i \in \mathbb{N}_n, j \in \mathbb{N}_m$, by direct computation, we have that

$$a_{ij} = b_{ij} + \sum_{p \in \mathbb{N}_s} c_{ip} \cdot m_{pj} + \sum_{r \in \mathbb{N}_t} d_{ir} \cdot m_{s+r,j}. \tag{2.40}$$

If $i < \eta_1$, by the definition of $\underline{\eta}(\cdot)$, we have that $b_{ij} = 0$, $c_{ip} = 0$ for each $p \in \mathbb{N}_s$, and $d_{ir} = 0$ for each $r \in \mathbb{N}_t$. Hence $a_{ij} = 0$. Therefore, we obtain that $\underline{\eta}(A) \geq \eta_1$. Likewise, if $i > \eta_2$, formula (2.40) and the definition of $\bar{\eta}(\cdot)$ yield that $a_{ij} = 0$. Thus, $\bar{\eta}(A) \leq \eta_2$. □

The sparse structure of A^ℓ can be described by the following lemma. Notice that given $\ell \in \mathbb{N}_{L-1}$, we have for each $i \in \mathbb{N}_{\gamma_\ell/4}$ that

$$A^\ell_{s_i} = [G^\ell_{4i-2}, G^\ell_{4i-1}], \quad A^\ell_{t_{2i-1}} = G^\ell_{4i-3}, \quad A^\ell_{t_{2i}} = G^\ell_{4i}. \tag{2.41}$$

Lemma 2.3.2. *Suppose that matrix A satisfies Hypothesis (A). If BGSP is applied to A, then for each $\ell \in \mathbb{N}_{L-1}$,*

$$\begin{aligned} \underline{\eta}\left(G^\ell_i\right) &\geq \begin{cases} 2^{\ell-1}(i-1)k - k/2 + 1, & i = 1, 3, \ldots, \gamma_\ell - 1, \\ 2^{\ell-1}(i-2)k + k/2 + 1, & i = 2, 4, \ldots, \gamma_\ell, \end{cases} \\ \bar{\eta}\left(G^\ell_i\right) &\leq \begin{cases} 2^{\ell-1}(i+1)k - k/2, & i = 1, 3, \ldots, \gamma_\ell - 1, \\ 2^{\ell-1}ik + k/2, & i = 2, 4, \ldots, \gamma_\ell. \end{cases} \end{aligned} \tag{2.42}$$

Proof. We shall prove the results by induction on $\ell \in \mathbb{N}_{L-1}$. Indeed, when $\ell = 1$, we can verify (2.42) by the banded structure of A. Assume that (2.42) is true for $\ell \in \mathbb{N}_{L-2}$.

We consider the case $(\ell + 1)$. To this end, we first express $G^{\ell+1}_i$ in column blocks of $A^\ell(\Lambda^\ell)$, then apply Lemma 2.3.1 and the induction hypothesis to yield the desired results. Let $i \in \mathbb{N}_{\gamma_{\ell+1}}$ be given, and $r := \lceil i/2 \rceil$. Notice that A is a block-wise quasi-orthogonal matrix of full rank, so is A^ℓ by Proposition 2.2.6. Substituting (2.41) and

$r = \lceil i/2 \rceil$ into formula (2.27) yields that

$$G_i^{\ell+1} = \begin{cases} G_{2i-1}^{\ell} - [G_{2i}^{\ell}, G_{2i+1}^{\ell}]\ M_i^{\ell}, & i = 1, 3, \ldots, \gamma_{\ell+1} - 1, \\ G_{2i}^{\ell} - [G_{2i-2}^{\ell}, G_{2i-1}^{\ell}]\ M_i^{\ell}, & i = 2, 4, \ldots, \gamma_{\ell+1}. \end{cases} \tag{2.43}$$

Now we discuss case by case. If i is odd, then by Lemma 2.3.1 and the induction hypothesis, we obtain that

$$\begin{aligned} \underline{\eta}(G_i^{\ell+1}) &\geq \min_{\mu=2i-1,2i,2i+1} \underline{\eta}(G_\mu^{\ell}) \geq 2^{\ell}(i-1)k - k/2 + 1, \\ \bar{\eta}(G_i^{\ell+1}) &\leq \max_{\mu=2i-1,2i,2i+1} \bar{\eta}(G_\mu^{\ell}) \leq 2^{\ell}(i+1)k - k/2. \end{aligned}$$

Hence, (2.42) is true for $(\ell+1)$ and for odd i. If i is even, likewise, we have that

$$\begin{aligned} \underline{\eta}(G_i^{\ell+1}) &\geq \min_{\mu=2i-2,2i-1,2i} \underline{\eta}(G_\mu^{\ell}) \geq 2^{\ell}(i-2)k + k/2 + 1, \\ \bar{\eta}(G_i^{\ell+1}) &\leq \max_{\mu=2i-2,2i-1,2i} \bar{\eta}(G_\mu^{\ell}) \leq 2^{\ell}ik + k/2. \end{aligned}$$

Thus, (2.42) is true for $(\ell+1)$ and for even i. By the induction principle, we obtain formula (2.42). □

Now we present a theorem below for $\#(Q)$.

Theorem 2.3.3. *Suppose that matrix A satisfies Hypothesis (A). If Q is the matrix constructed by BGSP from A, then the number of nonzero entries of Q is at most $2km\log_2(m/k)$.*

Proof. We prove the result by counting $\#(Q)$ using previous lemmas. Indeed by formula (2.12), $\#(Q)$ is the sum of $\#(Q^{\ell})$ for $\ell \in \mathbb{N}_L$. By formula (2.20), $\#(Q^{\ell})$ is the sum of $\#(Q_i^{\ell})$ for $i \in \mathbb{N}_{\gamma_\ell/4}$. Let $\ell \in \mathbb{N}_{L-1}$ be given. Notice that A^{ℓ} is a block-wise quasi-orthogonal matrix of full rank by Hypothesis (A) and Proposition 2.2.6.

Employing (2.22) and (2.41), we obtain that

$$Q_i^\ell = A_{s_i}^\ell (R_i^\ell)^{-1} = [G_{4i-2}^\ell, G_{4i-1}^\ell]\ (R_i^\ell)^{-1}. \tag{2.44}$$

It follows from Lemmas 2.3.1 and 2.3.2 that

$$\begin{array}{rcccl} \underline{\eta}(Q_i^\ell) & \geq & \min\limits_{\mu=4i-2,4i-1} \underline{\eta}(G_\mu^\ell) & \geq & 2^{\ell+1}(i-1)k + k/2 + 1, \\ \bar{\eta}(Q_i^\ell) & \leq & \max\limits_{\mu=4i-2,4i-1} \bar{\eta}(G_\mu^\ell) & \leq & 2^{\ell+1} ik - k/2. \end{array} \tag{2.45}$$

By the definition of Λ, we can verify that $Q_i^\ell \in \mathbb{R}^{n\times 2k}$ and $Q^L \in \mathbb{R}^{n\times 2k}$. Hence by (2.39), we have that

$$\#(Q) = \sum_{\ell\in\mathbb{N}_{L-1}} \sum_{i\in\mathbb{N}_{\gamma_\ell/4}} \#(Q_i^\ell) + \#(Q^L) \leq 2kmL = 2km\log_2(m/k),$$

proving the desired result. □

We present below an upper bound for $\#(S)$.

Theorem 2.3.4. *Suppose that matrix A satisfies Hypothesis (A). If S is the matrix constructed by BGSP from A, then the number of nonzero entries of S is at most $\frac{13}{4}km$.*

Proof. The proof is done by counting $\#(S)$ by the construction of S. Notice that F^ℓ is a permutation matrix. Thus, $\#(S)$ is the sum of $\#(R^\ell)$ for $\ell \in \mathbb{N}_L$ and $\#(C^\ell)$ for $\ell \in \mathbb{N}_{L-1}$.

We count $\#(R^\ell)$ as follows. Given $\ell \in \mathbb{N}_{L-1}$, for $i \in \mathbb{N}_{\gamma_\ell/4}$, we let R_i^ℓ be the upper triangular matrix constructed by MGS from $A_{s_i}^\ell$. It follows from the construction of R^ℓ that $\#(R^\ell)$ is the sum of $\#(R_i^\ell)$. Since $A_{s_i}^\ell \in \mathbb{R}^{n\times 2k}$, we have that $R_i^\ell \in \mathbb{R}^{2k\times 2k}$. Notice that $R^L \in \mathbb{R}^{2k\times 2k}$ is the upper triangular matrix constructed by MGS from $A^L \in \mathbb{R}^{n\times 2k}$.

Now we count $\#(C^\ell)$. Given $\ell \in \mathbb{N}_{L-1}$, for each $j \in \mathbb{N}_{\gamma_\ell/2}$, let $r := \lceil j/2 \rceil$, and let C_j^ℓ be the matrix constructed by $\mathrm{proj}(A_{t_j}^\ell, Q_r^\ell)$. It follows from the construction of C^ℓ that $\#(C^\ell)$ is the sum of $\#(C_j^\ell)$. Notice that $Q_r^\ell \in \mathbb{R}^{n\times 2k}$. Together with $A_{t_j}^\ell \in \mathbb{R}^{n\times k}$, we have that $C_j^\ell \in \mathbb{R}^{2k\times k}$.

From the above results, we obtain that $\#(S)$ is bounded above by

$$\left(1 + \sum_{\ell\in\mathbb{N}_{L-1}} (\gamma_\ell/4)\right) \cdot \tfrac{1}{2}\left[(2k)^2 + 2k\right] + \sum_{\ell\in\mathbb{N}_{L-1}} (\gamma_\ell/2)\cdot(2k^2) \le \tfrac{13}{4}km,$$

proving the desired result. □

2.4 Complexity of the Proposed Process

In this section, we study the complexity of BGSP. The complexity of a process is measured by the number of floating point operations (flops) in the process. We shall prove in this section that for an input matrix satisfying Hypothesis (A), if $k \ll m$, then the complexity of BGSP is $O(m \log m)$ flops.

To present the main result, we first clarify the flops in BGSP. Note that the operation of partitioning a matrix by column blocks may be accomplished by accessing the matrix with the corresponding column indices, while the operation of permuting the columns of a matrix may be accomplished by reordering the column indices of the matrix, and hence both operations are not flops. With this note, we conclude that the complexity in BGSP is the sum of the number of flops in $\mathrm{orthn}(\cdot)$ and $\mathrm{proj}(\cdot,\cdot)$ for all the steps.

We consider the number of flops in $\mathrm{proj}(\cdot,\cdot)$. For simplicity, given a matrix or a

vector $\mathcal{X}$, we define

$$\eta(\mathcal{X}) := \max\{\bar{\eta}(\mathcal{X}) - \underline{\eta}(\mathcal{X}) + 1,\ 0\}.$$

Lemma 2.4.1. *Let $A_0 := [a_j : j \in \mathbb{N}_s] \in \mathbb{R}^{n\times s}$ and $Q_0 := [q_i : i \in \mathbb{N}_t] \in \mathbb{R}^{n\times t}$ be matrices. If $\eta(Q_0) \le p$, then the complexity of* $\mathrm{proj}(A_0, Q_0)$ *is at most $(4pst)$ flops.*

Proof. By formulas (2.7) and (2.8), $\mathrm{proj}(A_0, Q_0)$ is to compute for $j \in \mathbb{N}_s$, $i \in \mathbb{N}_t$ that

$$a_j^{(i+1)} := a_j^{(i)} - q_i(q_i^T a_j^{(i)}), \tag{2.46}$$

where $a_j^{(1)} := a_j$ and $\bar{A}_0 := [a_j^{(t+1)} : j \in \mathbb{N}_s]$. By the definitions of $\underline{\eta}(\cdot)$ and $\bar{\eta}(\cdot)$, we have that $\underline{\eta}(q_i) \ge \underline{\eta}(Q_0)$ and $\bar{\eta}(q_i) \le \bar{\eta}(Q_0)$, respectively. Hence $\eta(q_i) \le \eta(Q_0) \le p$. It follows from direct computation that the number of flops in (2.46) is bounded above by $4p$. Thus, the complexity of $\mathrm{proj}(A_0, Q_0)$ is at most $(4pst)$ flops. □

We derive in the following theorem an upper bound for the number of flops in BGSP. It is known from [108] that if we apply MGS to a (dense) matrix $X \in \mathbb{R}^{p\times r}$, then the complexity is $2pr^2$ flops.

Theorem 2.4.2. *Suppose that matrix A satisfies Hypothesis (A). If BGSP is applied to A, then the complexity is at most $24k^2 m \log_2(m/k)$ flops.*

Proof. The proof follows from counting the number of flops in $\mathrm{orthn}(\cdot)$ by the complexity statement for MGS as above, and that in $\mathrm{proj}(\cdot,\cdot)$ by the previous lemma.

We first count the number of flops in $\mathrm{orthn}(\cdot)$. Let $\ell \in \mathbb{N}_{L-1}$, $i \in \mathbb{N}_{\gamma_\ell/4}$ be given. According to BGSP, MGS is applied to $A_{s_i}^\ell$. Formulas (2.41) and (2.42) yield that

$$\underline{\eta}(A_{s_i}^\ell) \ge 2^{\ell+1}(i-1)k + k/2 + 1 \quad \text{and} \quad \bar{\eta}(A_{s_i}^\ell) \le 2^{\ell+1} ik - k/2.$$

Since $A^\ell_{s_i} \in \mathbb{R}^{n\times 2k}$, it follows from the complexity statement for MGS that the number of flops is bounded above by

$$2 \cdot \eta(A^\ell_{s_i}) \cdot (2k)^2 \le 2 \cdot (2^{\ell+1}k) \cdot (2k)^2 = 2^{\ell+4}k^3.$$

Notice that $A^L \in \mathbb{R}^{n\times 2k}$, and $\eta(A^L) \le m$ by (2.42). Thus, we obtain that the total number of flops of orthn$(\cdot)$ for all steps is at most

$$2m \cdot (2k)^2 + \sum_{\ell\in\mathbb{N}_{L-1}} (\gamma_\ell/4) \cdot (2^{\ell+4}k^3) = 8k^2mL. \tag{2.47}$$

We then count the number of flops in proj$(\cdot,\cdot)$. Given $\ell \in \mathbb{N}_{L-1}$, $j \in \mathbb{N}_{\gamma_\ell/2}$, let $r := \lceil j/2 \rceil$. By formula (2.45), we have that $\eta(Q^\ell_r) \le 2^{\ell+1}k$. Since $A^\ell_{t_j} \in \mathbb{R}^{n\times k}$, $Q^\ell_r \in \mathbb{R}^{n\times 2k}$, it follows from Lemma 2.4.1 that the number of flops in proj$(A^\ell_{t_j}, Q^\ell_r)$ is bounded above by

$$4(2^{\ell+1}k)(k)(2k) = 2^{\ell+4}k^3.$$

Hence, an upper bound for the total number of flops in proj$(\cdot,\cdot)$ for all steps is

$$\sum_{\ell\in\mathbb{N}_{L-1}} (\gamma_\ell/2) \cdot (2^{\ell+4}k^3) = 16k^2m(L-1).$$

This together with (2.47) and $L = \log_2(m/k)$ yields the desired result. □

2.5 Stability of the Proposed Process

We study in this section the stability of BGSP by a round-off error analysis. The goal of this section is to prove that BGSP achieves numerical stability similar to MGS. To this end, we show that if matrices Q, S, E are constructed by BGSP from A, and if matrices $\bar{Q}$, $\bar{R}$ are constructed by MGS from AE, then

$$Q = \bar{Q},\ SE = \bar{R}. \tag{2.48}$$

Moreover, if matrices $\hat{Q}$, $\hat{S}$, $\tilde{Q}$, and $\tilde{R}$ are respectively the computed matrices of Q, S, $\bar{Q}$ and $\bar{R}$, then

$$\hat{Q} = \tilde{Q},\ \hat{S}E = \tilde{R}. \tag{2.49}$$

Thus we can derive the stability results of BGSP using those of MGS from [13, 14].

To prove the above two formulas, we write processes of BGSP applied to A and MGS applied to AE in operations on column blocks, from which we can see that the two processes are the same except that MGS has three more inner-loops for each step.

We first write BGSP applied to the partitioned matrix A as follows.

Let $A^1 := A$. For each $\ell \in \mathbb{N}_{L-1}$, do the following.

For $i \in \mathbb{N}_{\gamma_\ell/4}$, construct matrices Q_i^ℓ, R_i^ℓ by MGS from $A_{s_i}^\ell$.

For $j \in \mathbb{N}_{\gamma_\ell/2}$, let $r := \lceil j/2 \rceil$, and construct matrices $G_j^{\ell+1}$, C_j^ℓ by $\mathrm{proj}(A_{t_j}^\ell, Q_r^\ell)$.

Let $A^{\ell+1} := [G_j^{\ell+1} : j \in \mathbb{N}_{\gamma_\ell/2}]$.

Construct matrices Q^L, R^L by MGS from A^L.

Next we write the process MGS originally presented in [43] applied to AE, with E being constructed by BGSP from A. Let $\bar{A}^1 := A$. For each $\ell \in \mathbb{N}_{L-1}$, we associate $\bar{A}^\ell$ with the partition vector Π^ℓ defined above. For $i \in \mathbb{N}_{3\gamma_\ell/4}$, we let $\bar{A}_i^\ell$ denote the column block of $\bar{A}^\ell$ such that $\bar{A}^\ell = [\bar{A}_i^\ell : i \in \mathbb{N}_{3\gamma_\ell/4}]$. According to the construction of E and the fact that $\bar{A}^\ell (E^\ell)^T = [\bar{A}_\Omega^\ell, \bar{A}_\Theta^\ell]$, we can write the process as follows.

Let $\bar{A}^1 := A$. For each $\ell \in \mathbb{N}_{L-1}$, do the following.

For each $p \in \mathbb{N}_{3\gamma_\ell/4}$, let $\bar{A}_p^{\ell,1} := \bar{A}_p^\ell$.

For $i \in \mathbb{N}_{\gamma_\ell/4}$,

for $\alpha \in \mathbb{N}_{i-1}$, construct matrices $\bar{A}_{s_i}^{\ell,\alpha+1}, X_{\alpha i}^{\ell}$ by $\mathrm{proj}(\bar{A}_{s_i}^{\ell,\alpha}, \bar{Q}_{\alpha}^{\ell})$;

construct matrices $\bar{Q}_i^{\ell}$, $\bar{R}_i^{\ell}$ by MGS from $\bar{A}_{s_i}^{\ell,i}$.

For $j \in \mathbb{N}_{\gamma_\ell/2}$, let $r := \lceil j/2 \rceil$;

for $\mu \in \mathbb{N}_{r-1}$, construct matrices $\bar{A}_{t_j}^{\ell,\mu+1}, Y_{\mu j}^{\ell}$ by $\mathrm{proj}(\bar{A}_{t_j}^{\ell,\mu}, \bar{Q}_{\mu}^{\ell})$;

construct matrices $\bar{A}_{t_j}^{\ell,r+1}$, $\bar{C}_j^{\ell}$ by $\mathrm{proj}(\bar{A}_{t_j}^{\ell,r}, \bar{Q}_r^{\ell})$;

for $\nu = r+1, r+2, \ldots, \gamma_\ell/4$, construct matrices $\bar{A}_{t_j}^{\ell,\nu+1}, Y_{\nu j}^{\ell}$ by $\mathrm{proj}(\bar{A}_{t_j}^{\ell,\nu}, \bar{Q}_{\nu}^{\ell})$.

Let $\bar{G}_j^{\ell+1} := \bar{A}_{t_j}^{\ell,\gamma_\ell/4+1}$ for $j \in \mathbb{N}_{\gamma_\ell/2}$, and let $\bar{A}^{\ell+1} := [\bar{G}_j^{\ell+1} : j \in \mathbb{N}_{\gamma_\ell/2}]$.

Construct matrices $\bar{Q}^L$, $\bar{R}^L$ by MGS from $\bar{A}^L$.

It can be verified that the block-based process written as above is the same as the traditional MGS process [43] applied to the column vectors of AE. To form the QR factorization, the column orthonormal matrix $\bar{Q}$ is constructed by

$$\bar{Q} := \left[[\bar{Q}_i^{\ell} : i \in \mathbb{N}_{\gamma_\ell}, \ell \in \mathbb{N}_{L-1}], \bar{Q}^L \right],$$

while the upper triangular matrix $\bar{R}$ is formed by the blocks $\bar{R}_i^{\ell}$, $\bar{C}_j^{\ell}$, $X_{\alpha i}^{\ell}$, $Y_{\mu j}^{\ell}$, $Y_{\nu j}^{\ell}$, and $\bar{R}^L$ such that $AE = \bar{Q}\bar{R}$. Though we shall obtain (2.48) and (2.49) from the proofs below, we emphasize here that the extra three inner-loops in MGS applied to AE heavily increase the complexity, and communication costs in a parallel environment.

We now prove (2.48) in the following proposition.

Proposition 2.5.1. *Let $A(\Lambda) \in \mathbb{R}^{n\times m}$ be a block-wise quasi-orthogonal matrix of full rank, where $n \geq m$ and Λ is of length $\gamma := 2^L$ for some $L \in \mathbb{N}$. If matrices Q, S, E are constructed by BGSP from A, and if matrices $\bar{Q}$, $\bar{R}$ are constructed by MGS from AE, then $Q = \bar{Q}$, $SE = \bar{R}$.*

Proof. The proof is done by showing that $A^\ell = \bar{A}^\ell$ by induction on $\ell \in \mathbb{N}_L$. It follows from the induction hypothesis, the block-wise orthogonality of A^ℓ_Ω, and formula (2.21) that for $\alpha < i$, $\mu < r$, $\nu > r$,

$$\bar{A}^{\ell,\alpha+1}_{s_i} = \bar{A}^{\ell,\alpha}_{s_i}, \quad \bar{A}^{\ell,\mu+1}_{t_j} = \bar{A}^{\ell,\mu}_{t_j}, \quad \text{and} \quad \bar{A}^{\ell,\nu+1}_{t_j} = \bar{A}^{\ell,\nu}_{t_j},$$

which implies the result for $(\ell+1)$. Then the desired results follow from the construction of Q, $\bar{Q}$, S, and $\bar{R}$. As we will see analog statements in the proof of (2.49), except the use of the block-wise orthogonality of A^ℓ_Ω and (2.21), we skip details of the proof here. □

We next prove (2.49). Throughout this section, unless specified otherwise, we use the hat notation to indicate the computed quantities with round-off errors, and we assume the following hypothesis, which is true according to the IEEE standard for floating point arithmetic (*IEEE 754* [61]).

Hypothesis (F). If $fl(\cdot)$ denotes a floating point arithmetic, then

$$\hat{0} = 0, \quad fl(\hat{0} \pm \hat{\mathcal{X}}) = \hat{\mathcal{X}}, \quad \text{and} \quad fl(\hat{0} \cdot \hat{\mathcal{X}}) = fl(\hat{\mathcal{X}} \cdot \hat{0}) = 0,$$

where 0 or $\mathcal{X}$ could be a number, a vector, or a matrix such that the operation on 0 and $\mathcal{X}$ is compatible.

The idea for the proof of (2.49) is to show that the three extra inner-loops in MGS described above are *unnecessary* when MGS is applied to AE. Since they all compute the form of $\text{proj}(U, P)$, by Hypothesis (F), it suffices to show that the floating point inner product of the computed matrices $\hat{U}$ and $\hat{P}$ results in 0. This can be done by proving that $\underline{\eta}(\hat{U}) > \bar{\eta}(\hat{P})$ or $\bar{\eta}(\hat{U}) < \underline{\eta}(\hat{P})$, as described in the lemma below.

Lemma 2.5.2. *Let matrices* $U := [u_j : j \in \mathbb{N}_s] \in \mathbb{R}^{n\times s}$ *and* $P := [p_i : i \in \mathbb{N}_t] \in \mathbb{R}^{n\times t}$ *be given, and let* $V := [v_j : j \in \mathbb{N}_s]$, $D := [d_{ij} : i \in \mathbb{N}_t, j \in \mathbb{N}_s]$ *be the matrices constructed by* $\mathrm{proj}(U, P)$. *Assuming Hypothesis (F), if* $\underline{\eta}(\hat{U}) > \bar{\eta}(\hat{P})$ *or* $\bar{\eta}(\hat{U}) < \underline{\eta}(\hat{P})$, *then* $\hat{V} = \hat{U}$ *and* $\hat{D} = 0$.

Proof. We shall prove the results for the case that $\underline{\eta}(\hat{U}) > \bar{\eta}(\hat{P})$, and the proof for $\bar{\eta}(\hat{U}) < \underline{\eta}(\hat{P})$ is similar. Suppose that $\underline{\eta}(\hat{U}) > \bar{\eta}(\hat{P})$ and $j \in \mathbb{N}_s$ is given. By the definitions of $\underline{\eta}(\cdot)$ and $\bar{\eta}(\cdot)$, we have for each $i \in \mathbb{N}_t$ that $\underline{\eta}(\hat{u}_j) > \bar{\eta}(\hat{p}_i)$. Expanding (2.7) and (2.8) by column vectors in computed form yields the following process for computing $\hat{v}_j$.

$$
\begin{aligned}
&\text{Let} \quad \hat{u}_j^1 := \hat{u}_j.\\
&\text{For} \quad i = 1, 2, \ldots, t,\\
&\qquad \hat{d}_{ij} := fl(\hat{p}_i^T \hat{u}_j^i),\\
&\qquad \hat{u}_j^{i+1} := fl[\hat{u}_j^i - fl(\hat{d}_{ij}\hat{p}_i)].\\
&\text{Let} \quad \hat{v}_j := \hat{u}_j^{t+1}.
\end{aligned}
$$

It follows from $\underline{\eta}(\hat{u}_j) > \bar{\eta}(\hat{p}_1)$ and Hypothesis (F) that $\hat{d}_{1j} = 0$. Again by Hypothesis (F), we have that $\hat{u}_j^2 = \hat{u}_j^1$, and so $\underline{\eta}(\hat{u}_j^2) > \bar{\eta}(\hat{p}_r)$ for each $r \in \mathbb{N}_t$. Repeating the process, we have for $i \in \mathbb{N}_t$ that $\hat{u}_j^i = \hat{u}_j^1$ and $\hat{d}_{ij} = 0$. Therefore, we obtain the desired results. □

To prove (2.49), we need the following two lemmas to obtain the sparse structure of the computed matrices in BGSP. The first lemma concerns the properties of the functions $\underline{\eta}(\cdot)$ and $\bar{\eta}(\cdot)$.

Lemma 2.5.3. *Let matrices $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{n\times s}$, $D \in \mathbb{R}^{n\times t}$, $M \in \mathbb{R}^{(s+t)\times m}$ be given. Assuming Hypothesis (F), if matrix A is computed by $A := B + [C, D]M$, then $\underline{\eta}(\hat{A}) \geq \min\{\underline{\eta}(\hat{B}), \underline{\eta}(\hat{C}), \underline{\eta}(\hat{D})\}$ and $\bar{\eta}(\hat{A}) \leq \max\{\bar{\eta}(\hat{B}), \bar{\eta}(\hat{C}), \bar{\eta}(\hat{D})\}$.*

Proof. The result follows from the proof of Lemma 2.3.1, replacing any quantity appearing in the proof by its computed quantity, and using Hypothesis (F). □

The lemma below ensures that the computed matrix $\hat{A}^\ell$ has the same sparse structure as A^ℓ.

Lemma 2.5.4. *Assuming Hypothesis (F) and that matrix A satisfies Hypothesis (A), if BGSP is applied to A, then for each $\ell \in \mathbb{N}_{L-1}$,*

$$\underline{\eta}\big(\hat{G}_i^\ell\big) \geq \begin{cases} 2^{\ell-1}(i-1)k - k/2 + 1, & i = 1, 3, \ldots, \gamma_\ell - 1, \\ 2^{\ell-1}(i-2)k + k/2 + 1, & i = 2, 4, \ldots, \gamma_\ell, \end{cases}$$

$$\bar{\eta}\big(\hat{G}_i^\ell\big) \leq \begin{cases} 2^{\ell-1}(i+1)k - k/2, & i = 1, 3, \ldots, \gamma_\ell - 1, \\ 2^{\ell-1}ik + k/2, & i = 2, 4, \ldots, \gamma_\ell. \end{cases}$$

Proof. The result follows from the proof of Lemma 2.3.2, replacing any matrix appearing in the proof by its computed matrix, together with Hypothesis (F) and Lemma 2.5.3. □

Now we prove (2.49).

Proposition 2.5.5. *Suppose that matrix A satisfies Hypothesis (A). Let Q, S, E be the matrices constructed by BGSP from A, and $\bar{Q}$, $\bar{R}$ be the matrices constructed by MGS from AE. Assuming Hypothesis (F), if matrices $\hat{Q}$, $\hat{S}$, $\tilde{Q}$, and $\tilde{R}$ are respectively the computed matrices of Q, S, $\bar{Q}$ and $\bar{R}$, then $\hat{Q} = \tilde{Q}$, $\hat{S}E = \tilde{R}$.*

Proof. In this proof, we shall use the hat notation and tilde notation respectively to indicate the computed matrices in BGSP and MGS. We shall prove the results by showing the following equation by induction on $\ell \in \mathbb{N}_L$,

$$\tilde{A}^{\ell} = \hat{A}^{\ell}. \tag{2.50}$$

Indeed, when $\ell = 1$, we have $\tilde{A}^{1} = A = \hat{A}^{1}$. Assuming that (2.50) is true for $\ell \in \mathbb{N}_{L-1}$, we have for each p that

$$\tilde{A}_p^{\ell,1} = \hat{A}_p^{\ell}. \tag{2.51}$$

We shall prove (2.50) for $(\ell+1)$. By comparing the two processes above, with Lemma 2.5.2, it suffices to show that

$$\underline{\eta}(\tilde{A}_{s_i}^{\ell,\alpha}) > \bar{\eta}(\tilde{Q}_{\alpha}^{\ell}), \quad \text{for } \alpha < i, \tag{2.52}$$

$$\bar{\eta}(\tilde{A}_{t_j}^{\ell,\mu}) < \underline{\eta}(\tilde{Q}_{\mu}^{\ell}), \quad \text{for } \mu < r, \tag{2.53}$$

$$\underline{\eta}(\tilde{A}_{t_j}^{\ell,\nu}) > \bar{\eta}(\tilde{Q}_{\nu}^{\ell}), \quad \text{for } \nu > r. \tag{2.54}$$

We first prove (2.52). To this end, we show the following equation by induction on $i \in \mathbb{N}_{\gamma_\ell/4}$,

$$\tilde{Q}_i^{\ell} = \hat{Q}_i^{\ell}. \tag{2.55}$$

Indeed, when $i = 1$, since $\tilde{A}_{s_1}^{\ell,1} = \hat{A}_{s_1}^{\ell}$, we have that $\tilde{Q}_1^{\ell} = \hat{Q}_1^{\ell}$. Assuming that (2.55) is true for all $i < \gamma_\ell/4$, we shall show the result for $(i+1)$. By formulas (2.44), (2.41), Lemmas 2.5.3 and 2.5.4, we have for $\alpha' < i'$ that

$$\underline{\eta}(\hat{A}_{s_{i'}}^{\ell}) > \bar{\eta}(\hat{Q}_{\alpha'}^{\ell}). \tag{2.56}$$

Equation (2.51) and the induction hypothesis lead to $\underline{\eta}(\tilde{A}_{s_{i+1}}^{\ell,1}) > \bar{\eta}(\tilde{Q}_1^{\ell})$. By Lemma 2.5.2, we have that $\tilde{A}_{s_{i+1}}^{\ell,2} = \hat{A}_{s_{i+1}}^{\ell}$. Repeating using (2.56) and Lemma 2.5.2 for $\alpha' =$

$2, 3, \ldots, i$ yields that $\tilde{A}^{\ell,i+1}_{s_{i+1}} = \hat{A}^{\ell}_{s_{i+1}}$. Thus, we obtain the desired result for $(i+1)$. By the induction principle, we have formula (2.55). Now for a fixed i, we have that $\tilde{A}^{\ell,1}_{s_i} = \hat{A}^{\ell}_{s_i}$ by (2.51). Repeating using (2.56), (2.55), and Lemma 2.5.2, we obtain formula (2.52).

We next prove (2.53). By formulas (2.44), (2.41), Lemmas 2.5.3 and 2.5.4, we have for $\mu' < \lceil j'/2 \rceil$ that

$$\bar{\eta}(\hat{A}^{\ell}_{t_{j'}}) < \underline{\eta}(\hat{Q}^{\ell}_{\mu'}). \tag{2.57}$$

Now for a fixed j with $r = \lceil j/2 \rceil$, we have that $\tilde{A}^{\ell,1}_{t_j} = \hat{A}^{\ell}_{t_j}$ by (2.51). Repeating using (2.57), (2.55), and Lemma 2.5.2, we obtain formula (2.53). Therefore, $\tilde{A}^{\ell,r}_{t_j} = \hat{A}^{\ell}_{t_j}$, and hence $\tilde{A}^{\ell,r+1}_{t_j} = \hat{G}^{\ell+1}_{j}$.

The proof of (2.54) follows similarly.

Formula (2.54) and Lemma 2.5.2 yield that $\tilde{A}^{\ell,\gamma_\ell+1}_{t_j} = \tilde{A}^{\ell,r+1}_{t_j}$. Hence, we obtain (2.50) for $(\ell+1)$. By the induction principle, we obtain formula (2.50).

Equation (2.50) implies that formulas (2.52), (2.53), and (2.54) are true for $\ell \in \mathbb{N}_{L-1}$. It follows from Lemma 2.5.2 that $\tilde{X}^{\ell}_{\alpha i} = 0$, $\tilde{Y}^{\ell}_{\mu j} = 0$, and $\tilde{Y}^{\ell}_{\nu j} = 0$. Therefore, $\hat{Q} = \tilde{Q}$, $\hat{S}E = \tilde{R}$. □

By the previous proposition, we can obtain the stability results of BGSP using those of MGS applied to AE. The detailed round-off error analysis of MGS for general dense matrices is given in [13, 14]. We shall show the stability of BGSP using the main results in [13], with the fact that A is banded. In the following, we assume that the floating point arithmetic in double precision is used, and denote by $\mathbf{u}$ the machine

unit error. For each $j \in \mathbb{N}_m$, we define

$$\bar{\ell}_j := \max_{1 \le \ell \le L} \big\{ \ell : 2^\ell \text{ divides } \lceil j/k \rceil \text{ or } (\lceil j/k \rceil - 1) \big\}. \tag{2.58}$$

With the notation, at the $\bar{\ell}_j$-th step of BGSP, the jth column of A is orthonormalized by orthn$(\cdot)$. Also, we denote by $\|\cdot\|_F$ the Frobenius norm, that is, for a matrix $X := [x_{ij} : i \in \mathbb{N}_n, j \in \mathbb{N}_m]$, $\|X\|_F := (\sum_{i \in \mathbb{N}_n} \sum_{j \in \mathbb{N}_m} x_{ij}^2)^{1/2}$.

To derive an error bound for the factorization, we include the following lemma proved in [13].

Lemma 2.5.6. *Let $A \in \mathbb{R}^{n \times m}$ be of full rank, $n \ge m$. Let $\tilde{Q}$ and $\tilde{R}$ be respectively the computed matrices of Q and R constructed by MGS from A. For $i \in \mathbb{N}_m$, let a_i and Δa_i be the ith column vectors of matrices A and $(A - \tilde{Q}\tilde{R})$, respectively. If*

$$2.12(n+1)\mathbf{u} < 0.01, \tag{2.59}$$

then

$$\|\Delta a_i\|_2 \ \le \ \tfrac{3}{2}(i-1)\mathbf{u}\|a_i\|_2, \tag{2.60}$$

$$\|A - \tilde{Q}\tilde{R}\|_F \ \le \ \tfrac{3}{2}(m-1)\mathbf{u}\|A\|_F. \tag{2.61}$$

We present below the error in the block QS factorization via BGSP.

Theorem 2.5.7. *Suppose that matrix A satisfies Hypothesis (A). Assuming Hypothesis (F) and (2.59), if $\hat{Q}$ and $\hat{S}$ are respectively the computed matrices of Q and S constructed by BGSP from A, then*

$$\|A - \hat{Q}\hat{S}\|_F \le 3k \log_2 (m/k)\mathbf{u}\|A\|_F. \tag{2.62}$$

Proof. The proof follows from Proposition 2.5.5 and Lemma 2.5.6. For $j \in \mathbb{N}_m$, let a_j and Δa_j be the jth column vectors of matrices A and $(A - \hat{Q}\hat{S})$, respectively. Let $\tilde{Q}$

and $\tilde{R}$ be the computed factor matrices constructed by MGS from AE. By Proposition 2.5.5, we have that $\hat{Q} = \tilde{Q}$, $\hat{S}E = \tilde{R}$, and hence

$$(A - \hat{Q}\hat{S})E = AE - \tilde{Q}\tilde{R}.$$

Let p_j be the number of column vectors of AE prior to a_j. Employing (2.60) in Lemma 2.5.6 with $i = p_j + 1$, we obtain that,

$$\|\Delta a_j\|_2 \leq \tfrac{3}{2} p_j \mathbf{u} \|a_j\|_2.$$

With the fact that the unnecessary inner-loops of MGS applied to AE are not executed in BGSP, it follows from direct counting that $p_j \leq 2\bar{\ell}_j k$. Hence,

$$\|\Delta a_j\|_2 \leq 3\bar{\ell}_j k \mathbf{u} \|a_j\|_2. \tag{2.63}$$

Since $\bar{\ell}_j \leq \log_2(m/k)$, we obtain formula (2.62). □

We remark that if $k \ll m$, then the error bound in (2.62) is much smaller that in (2.61). Moreover, if the column vectors of A have the same 2-norm, then we can achieve an improvement over (2.62), as in the theorem below. Given a matrix $A := [a_j : j \in \mathbb{N}_m]$, we can construct a diagonal matrix D whose diagonal entries are $\|a_j\|_2^{-1}$ for $j \in \mathbb{N}_m$, so that each column vector of AD has norm 1. This column normalization method is commonly used for precondition.

Theorem 2.5.8. *Suppose that matrix A satisfies Hypothesis (A), and that the column vectors of A have the same 2-norm. Assuming Hypothesis (F) and* (2.59)*, if $\hat{Q}$ and $\hat{S}$ are respectively the computed matrices of Q and S constructed by BGSP from A, then*

$$\|A - \hat{Q}\hat{S}\|_F \leq 3\sqrt{6} k \mathbf{u} \|A\|_F.$$

Proof. Let $\alpha := \|a_j\|_2$, for some column vector a_j of A. Employing (2.63), we obtain

that

$$\|\Delta a_j\|_2 \leq 3\bar{\ell}_j k\mathbf{u}\alpha.$$

By direct counting according to the definition (2.58) of $\bar{\ell}_j$, we have for each $\ell \in \mathbb{N}_{L-1}$ that there are $(m/2^\ell)$ many column vectors a_j of A with $\bar{\ell}_j = \ell$. Together with the equation $m = 2^L k$, we have that

$$\|A - \hat{Q}\hat{S}\|_F \leq \left(2L^2 k + \sum_{\ell \in \mathbb{N}_{L-1}} \ell^2 m/2^\ell\right)^{1/2} \cdot\ 3k\mathbf{u}\alpha \leq 3\sqrt{6}k\mathbf{u}\|A\|_F,$$

in which the coefficient of $\mathbf{u}\|A\|_F$ is independent of m. □

We present below an upper bound for $\|\hat{Q}^T\hat{Q} - I\|_2$, the loss of orthogonality of $\hat{Q}$. We denote by $\kappa(A) := \|A\|_2\ \|A^+\|_2$ the condition number of A. It is proved in [13] that with (2.59), if $\tilde{Q}$ is the computed matrix of Q constructed by MGS from $A \in \mathbb{R}^{n\times m}$ of full rank, and $3.42m(m+1)\mathbf{u}\kappa(A) < 1$, then there exists a constant $c := c(m)$ such that

$$\|\tilde{Q}^T\tilde{Q} - I\|_2 \leq c\mathbf{u}\kappa(A). \tag{2.64}$$

Theorem 2.5.9. *Suppose that matrix A satisfies Hypothesis (A). Let $\hat{Q}$ be the computed matrix of the column orthonormal matrix Q constructed by BGSP from A. Assuming Hypothesis (F) and* (2.59)*, if $3.42m(m+1)\mathbf{u}\kappa(A) < 1$, then there exists a constant $c := c(m)$ such that*

$$\|\hat{Q}^T\hat{Q} - I\|_2 \leq c\mathbf{u}\kappa(A).$$

Proof. The proof follows from Proposition 2.5.5 and formula (2.64). □

2.6 A Recursive Algorithm

We develop in this section a recursive algorithm called RBGSP for the block QS factorization that implements BGSP. The purpose of introducing recursion is to describe the orthogonalization process in a more simple way than BGSP, as RBGSP generates the block QS factorization of matrices directly by an intuitive idea.

To begin, we review the concept of recursive algorithms. Recall that an algorithm for solving a given problem is recursive if it obeys the following three laws of recursion [39]: First, the algorithm has a *base case* of the problem, in which a simple, non-recursive solution is generated. Second, for a case that is not a base case, the algorithm reduces it to one or more cases of the problem that are closer to the base case, using *calls* of the algorithm itself. Third, the algorithm eventually reduces the problem to the base case only. As an alternative to iteration, recursion enables us to specify a natural, simple solution to a problem that would otherwise be difficult to solve. Indeed, BGSP is an iterative process, that may be not easy to understand.

We summarize our intuitive idea for constructing the block QS factorization described in section 2.1 as follows: The problem to be solved is to obtain the block QS factorization of A with a partition vector Λ of length γ. The case when $\gamma \leq 2$ is a base case, in which we directly generate the solution by MGS. For $\gamma > 2$, we construct two submatrices of A, compute partial results by orthn$(\cdot)$ and proj$(\cdot, \cdot)$, and then reduce the problem to find the block QS factorization of a matrix partitioned by a subvector of Λ (having smaller length than Λ). We repeat the process until the case of the problem becomes a base one.

We now propose the recursive algorithm RBGSP. For a matrix $A(\Lambda)$ with $\Lambda = [n_i :$

$i \in \mathbb{N}_\gamma]$, and $\gamma = 2^L$ for some $L \in \mathbb{N}\backslash\{1\}$, we define

$$\Pi := [n_{4i-3},\ n_{4i-2} + n_{4i-1},\ n_{4i} : i \in \mathbb{N}_{\gamma/4}], \tag{2.65}$$

$$\Omega := [s_i : i \in \mathbb{N}_{\gamma/4}], \quad \text{and} \quad \Theta := [t_j : j \in \mathbb{N}_{\gamma/2}]. \tag{2.66}$$

Then we let Π_Ω, Π_Θ be defined by (2.4) and let A_Ω, A_Θ be defined by (2.5). Also, we define a permutation matrix E by

$$E := \begin{bmatrix} \bigoplus_{i\in\mathbb{N}_{\gamma/4}} \left(\left[0_{n_{4i-2}\times n_{4i-3}},\ I_{n_{4i-2}} \right] \oplus \left[I_{n_{4i-1}},\ 0_{n_{4i-1}\times n_{4i}} \right] \right) \\ \bigoplus_{i\in\mathbb{N}_{\gamma/4}} \left(\left[I_{n_{4i-3}},\ 0_{n_{4i-3}\times n_{4i-2}} \right] \oplus \left[0_{n_{4i}\times n_{4i-1}},\ I_{n_{4i}} \right] \right) \end{bmatrix},$$

so that $A = [A_\Omega, A_\Theta]E$. The algorithm is presented below.

Algorithm 2.1 The recursive algorithm for the block QS factorization (RBGSP)

Input: A matrix $A(\Lambda)$ of full rank with $\Lambda = [n_i : i \in \mathbb{N}_\gamma]$ and $\gamma = 2^L$ for some $L \in \mathbb{N}$.
Output: Three matrices Q_{rec}, S_{rec}, E_{rec}.

1: **if** $\gamma \leq 2$ **then**
2: generate matrices Q, R by MGS from A,
3: **return** $Q_{rec} := Q$, $S_{rec} := R$, $E_{rec} := I$.
4: **else**
5: generate matrices Q, R by orthn(A_Ω),
6: generate matrices $\bar{A}_\Theta$, C by proj(A_Θ, Q),
7: generate matrices $\bar{Q}$, $\bar{S}$, $\bar{E}$ by RBGSP from $\bar{A}_\Theta(\Pi_\Theta)$,
8: **return** the following

$$Q_{rec} := [Q, \bar{Q}], \tag{2.67}$$

$$S_{rec} := \begin{bmatrix} R & C \\ 0 & \bar{S} \end{bmatrix} E, \tag{2.68}$$

$$E_{rec} := E^T(I \oplus \bar{E}). \tag{2.69}$$

As RBGSP is recursive for the *calls* of itself on line 7, we verify that RBGSP must stop in finitely many steps. Suppose that in the first call of RBGSP, $\gamma = 2^L$ for some $L \in \mathbb{N}$. If $\gamma = 2$, then RBGSP returns on line 3. If $\gamma > 2$, then on line 7, the input matrix is partitioned by Π_Θ, which has length $\gamma/2$ by the definitions (2.65), (2.66), and (2.4). Thus, we conclude by induction that for each $\ell \in \mathbb{N}_L$, γ_ℓ defined by (2.10) is the length of the partition vector of the input matrix in the ℓth call of RBGSP. Hence,

if and only if $\ell = L$, $\gamma_\ell \leq 2$ and RBGSP returns. In conclusion, RBGSP will stop in L calls.

We shall prove that RBGSP implements BGSP, that is, the matrices Q_{rec}, S_{rec}, E_{rec} generated from RBGSP are respectively the same as Q, S, E constructed by BGSP. To this end, we present the following remark, which can be verified by induction.

***Remark* 3.** For $X = A$, Q, R, C, or E, let X^ℓ, $\hat{X}^\ell$ be respectively the matrix and its computed matrix generated by the ℓth call of RBGSP. Then X^ℓ is the same as the one constructed by BGSP, and $\hat{X}^\ell$ is the same as the computed matrix of X^ℓ constructed by BGSP.

We next clarify the execution order of RBGSP. Notice that RBGSP differs from BGSP in the formations of the factor matrices. Unlike a sequential order of execution, the output matrices Q_{rec}, S_{rec}, E_{rec} are formed by the returned matrices of the calls from back to front, that is, in the descending order of ℓ running from L to 1. This is due to the executive mechanism of recursive algorithms. A diagram of the process of execution for RBGSP is illustrated in Figure 2.4 below, in which we denote by Q^ℓ_{rec}, S^ℓ_{rec}, E^ℓ_{rec} the returned matrices of the ℓth call of RBGSP.

We now present in the theorem below that RBGSP implements BGSP.

Theorem 2.6.1. *Let A be a matrix of full rank with the partition vector Λ of length $\gamma := 2^L$ for some $L \in \mathbb{N}$. If matrices Q_{rec}, S_{rec}, E_{rec} are generated by RBGSP from A, matrices Q, S, E are constructed by BGSP from A, and matrices $\hat{Q}_{rec}$, $\hat{S}_{rec}$, $\hat{Q}$, $\hat{S}$*

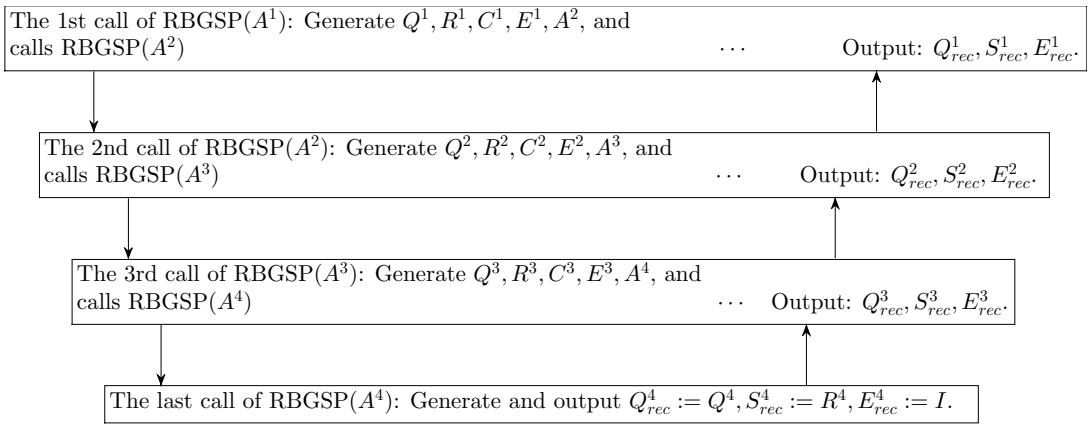


FIGURE 2.4: *Diagram of the process of execution for RBGSP with $L = 4$. Time moves along the arrows.*

are the computed matrices of Q_{rec}, S_{rec}, Q, S, respectively, then

$$Q_{rec} = Q, \quad S_{rec} = S, \quad E_{rec} = E, \tag{2.70}$$

$$\hat{Q}_{rec} = \hat{Q}, \quad \hat{S}_{rec} = \hat{S}. \tag{2.71}$$

Proof. We first prove (2.70). If $L = 1$, then according to RBGSP and BGSP, we have that $Q_{rec} = Q^1 = Q$, $S_{rec} = R^1 = S$, and $E_{rec} = I = E$. We consider $L > 1$ below.

We show that $Q_{rec} = Q$. By formula (2.12) and Remark 3, we shall prove that

$$Q_{rec} = [Q^\ell : \ell \in \mathbb{N}_L]. \tag{2.72}$$

This can be done by showing that

$$Q_{rec} = [Q^1, Q^2, \ldots, Q^\ell, Q^{\ell+1}_{rec}]. \tag{2.73}$$

by induction on $\ell \in \mathbb{N}_{L-1}$. Indeed, noticing that $\bar{Q}^j = Q^{j+1}_{rec}$, by (2.67) from RBGSP, we obtain for each $j \in \mathbb{N}_{L-1}$ that

$$Q^j_{rec} = [Q^j, Q^{j+1}_{rec}]. \tag{2.74}$$

With $Q_{rec} = Q^1_{rec}$, we obtain (2.73) for $\ell = 1$. Assume that (2.73) is true for $\ell \in \mathbb{N}_{L-2}$. Since $(\ell + 1) \in \mathbb{N}_{L-1}$, employing the induction hypothesis and formula (2.74) with

$j = \ell + 1$, we obtain (2.73) for $(\ell + 1)$. Hence by the induction principle, we obtain formula (2.73). Notice that $Q_{rec}^L = Q^L$ according to RBGSP. This together with formula (2.73) for $\ell = L - 1$ yields formula (2.72). Therefore, $Q_{rec} = Q$.

We next prove that $S_{rec} = S$. By formula (2.15), we shall prove that

$$S_{rec} = \left[(S^\ell)^T : \ell \in \mathbb{N}_L\right]^T. \tag{2.75}$$

This can be done by showing that

$$S_{rec} = \left[(S^1)^T, (S^2)^T, \ldots, (S^\ell)^T, ([0,\ S^{\ell+1}]F^\ell)^T\right]^T. \tag{2.76}$$

by induction on $\ell \in \mathbb{N}_{L-1}$. Indeed, noticing that $\bar{S}^j = S_{rec}^{j+1}$, by (2.68) from RBGSP, we obtain for each $j \in \mathbb{N}_{L-1}$ that

$$S_{rec}^j = \begin{bmatrix} R^j & C^j \\ 0 & S_{rec}^{j+1} \end{bmatrix} E^j. \tag{2.77}$$

By the definitions (2.13), (2.14) of F^1, S^1, we obtain that

$$S_{rec} = S_{rec}^1 = \begin{bmatrix} R^1 & C^1 \\ 0 & S_{rec}^2 \end{bmatrix} E^1 = \begin{bmatrix} S^1 \\ [0,\ S_{rec}^2]F^1 \end{bmatrix}.$$

Assume that (2.76) is true for $\ell \in \mathbb{N}_{L-2}$. Since $(\ell+1) \in \mathbb{N}_{L-1}$, employing the induction hypothesis and formula (2.77) with $j = \ell+1$, and the definitions (2.13), (2.14) of $F^{\ell+1}$, $S^{\ell+1}$, we obtain that

$$S_{rec} = \begin{bmatrix} [(S^t)^T : t \in \mathbb{N}_\ell]^T \\ [0,\ S_{rec}^{\ell+1}]F^\ell \end{bmatrix} = \begin{bmatrix} [(S^t)^T : t \in \mathbb{N}_\ell]^T \\ \begin{bmatrix} 0 & R^{\ell+1} & C^{\ell+1} \\ 0 & 0 & S_{rec}^{\ell+2} \end{bmatrix} (I \oplus E^{\ell+1})F^\ell \end{bmatrix} = \begin{bmatrix} [(S^t)^T : t \in \mathbb{N}_\ell]^T \\ S^{\ell+1} \\ [0,\ S_{rec}^{\ell+2}]F^{\ell+1} \end{bmatrix},$$

proving that (2.76) is true for $(\ell + 1)$. Hence by the induction principle, we obtain formula (2.76). Notice that $S_{rec}^L = R^L$ according to RBGSP, and $S^L = [0,\ R^L]F^{L-1}$.

Together with formula (2.76) for $\ell = L-1$, we obtain formula (2.75). Therefore, $S_{rec} = S$.

Since $\bar{E}^j = E_{rec}^{j+1}$, by (2.69), we obtain for each $j \in \mathbb{N}_{L-1}$ that

$$E_{rec}^j = (E^j)^T (I \oplus E_{rec}^{j+1}).$$

The proof of $E_{rec} = E$ is done by expanding the above formula for j, and expanding (2.13) for $\ell \in \mathbb{N}_{L-1}$, together with equations $E_{rec} = E_{rec}^1$, $E_{rec}^L = I$, and (2.16).

It remains to show (2.71). Equation $\hat{Q}_{rec} = \hat{Q}$ directly follows from formulas (2.72), (2.12), and Remark 3. Recall that the operations of permuting the columns of a matrix are not floating point operations, and thus do not produce any round-off errors. With the fact that E^ℓ's are permutation matrices, equation $\hat{S}_{rec} = \hat{S}$ can be derived from formulas (2.75), (2.15), (2.14), and Remark 3. □

We finally remark that comparing to BGSP, it is more convenient for RBGSP to obtain the block QS factorization of A^ℓ. Indeed, it is can be verified by induction on $\ell = L, L-1, \ldots, 1$ that $Q_{rec}^\ell S_{rec}^\ell$ is the block QS factorization of A^ℓ, and specifically, $S_{rec}^\ell E_{rec}^\ell$ is upper triangular. Thus, we obtain the block QS factorization of A^ℓ by the returned matrices in the ℓth call of RBGSP.

Chapter 3

A Parallel Algorithm for the Block QS Factorization

In this chapter, we develop a parallel algorithm for the block QS factorization in the Bulk Synchronous Parallel model for distributed memory computing.

We study keystones in the model for implementing the proposed parallel algorithm. Especially, we make use of the concurrent computation in BGSP by dividing the computations in BGSP into parts, each of which can be executed by an individual processor. A concern for the model is the time-consuming barrier synchronization between the communication phrase and the concurrent computation phrase. For this concern, we directly view the process for the block QS factorization as a special case of one collective communication operation, the all-reduction operation. Then we apply a butterfly all-reduction tree structure in order to communicate pairwise between processors, avoiding any barrier synchronization.

We conduct a detailed analysis for the parallel performance of the proposed al-

gorithm. We have observed that when applied to large-scale ill-conditioned banded matrices, the proposed algorithm generates even more sparse intermediate matrices than the sequential algorithm. Benefiting from this observation, we show that the numbers of communication messages and words both reach their theoretical least upper bounds. Also, the parallel complexity is asymptotically bounded above by its theoretical least upper bound multiplying by a logarithmic quantity that depends on the matrix size, the bandwidth, and the number of processors. Moreover, the speedup is asymptotically bounded below by its theoretical greatest lower bound dividing by the logarithm of the number of processors, indicating that the proposed algorithm has approximately ideal scalability.

We organize this chapter in four sections. In section 3.1, we introduce the Bulk Synchronous Parallel model in which we develop the parallel algorithm. We also discuss related issues and our ideas on these issues. In section 3.2, we describe the implementation of the parallel algorithm for the keystones in the model. We propose the parallel algorithm in section 3.3, and verify that it fulfils the block QS factorization, as well as inherits the nice features from BGSP. Section 3.4 is devoted to the analysis of the proposed parallel algorithm.

3.1 Introduction

The goal in this chapter is to develop a parallel algorithm for the block QS factorization by parallelizing the sequential process BGSP. In this section, we discuss the issues in parallel implementation, and present our ideas to address these issues. Recall that in massive data processing described in chapter 1, data are typically distributed in different locations. Thus, we shall design the parallel algorithm using the Bulk Synchronous Parallel (BSP) model, which is the most common model for distributed memory computing. To start with, we review the model as follows.

We briefly review the BSP model [111]. A parallel algorithm in the BSP model proceeds in a series of global supersteps. Each superstep consists of three phrases: concurrent computation, communication, and barrier synchronization. During the concurrent computation, every participating processor may perform local computations, that is, each processor can only access the data stored in its local memory, and compute independently from the other processors. In the communication phase, the processors exchange data between themselves. During barrier synchronization, each processor reaches the barrier and waits until all the others have reached the same barrier. To optimize parallel performance, one wishes to maximize the concurrency of the computation, as well as minimize the cost for communication and barrier synchronization.

The first issue is the concurrency of the computation in each superstep. To maximize the concurrency, the ideal way is to evenly divided all the computations in BGSP and assign to participating processors. Since each processor can only access the data stored in its local memory, we shall partition the input matrix by column blocks and

evenly distribute to the processors. We will study further in the next section for the concurrent computation.

The second issue is that the number of times for communication may depend on the number L of steps of BGSP. This can be seen from each call of $\mathrm{proj}(\cdot,\cdot)$ in BGSP for $\ell \in \mathbb{N}_{L-1}$. In $\mathrm{proj}(\cdot,\cdot)$, given $i \in \mathbb{N}_\gamma$, if matrices A_{2i} and Q_i are stored in two different processors, then the two processors have to exchange their matrices to each other in order to execute $\mathrm{proj}(A_{2i}, Q_i)$. To avoid the data exchange, we trace back to the column blocks of the input matrix from which A_{2i} and Q_i are derived. We then distribute these column blocks as a unit to the processors. In this way, we could assign the computations in the first few steps of BGSP to the first superstep of the parallel algorithm, in which communication is free. As a result, the number of times for communication does not depend on L, and it is the logarithm of the number of processors, which is optimal [7].

The third issue is that barrier synchronization between supersteps is very time-consuming since it is a global coordination involving all the processors. To overcome this issue, binary tree structure is studied to adopt pairwise communication between processors instead of global synchronization among all the processors (see, for example, [35, 57]). In order to apply the structure, we view the entire process for the block QS factorization as a reduction operation, with the specialty that the intermediate results must be stored in the processors that produce them. Notice that a reduction is one type of collective operations that combine multiple results into one overall result, with some associative function. Also, a reduction is usually fulfilled by the binary tree structure [47].

3.2 Implementation in the BSP model

In this section, we describe the implementation of our parallel algorithm for the keystones in the BSP model: concurrent computation, data distribution, supersteps, and communication.

We first discuss the concurrent computation in the parallel implementation. We consider the computations in BGSP that could be divided into parts to be executed concurrently by processors. We observe that the computations in procedures orthn$(\cdot)$ and proj$(\cdot,\cdot)$ introduced in chapter 2 are naturally concurrent. For orthn$(\cdot)$, given $i \in \mathbb{N}_\gamma$, if a processor holds the matrix A_i, it can apply MGS independently, without any interactions with the other processors. Ideally, if there are γ participating processors, then orthn$(\cdot)$ can be executed completely concurrently. Likewise, for proj$(\cdot,\cdot)$, given $i \in \mathbb{N}_\gamma$, $j \in \mathbb{N}_{2\gamma}$, with $i = \lceil j/2 \rceil$, if a processor holds both A_j and Q_i, then it can execute proj(A_j, Q_i) independently.

Furthermore, we can divide some computations in BGSP into parts. Using the same notation as BGSP, we provide the following sub-algorithm (BGSP$_1$) that reveals the feature of the concurrent computation. In fact, BGSP$_1$ is the same as BGSP, except the former does not proceed the Lth step. In this chapter, we assume that the input matrix A satisfies the following hypothesis.

Hypothesis ($\mathbf{A}_1$). Matrix $A(\Lambda) \in \mathbb{R}^{n\times m}$ is of full rank, where $n \geq m$ and Λ is of length $\gamma := 2^L$ for some $L \in \mathbb{N}\backslash\{1\}$.

The computations in the sub-algorithm BGSP$_1$ can be divided into parts to be executed concurrently. To see this, we let $P \in \mathbb{N}$ satisfy the hypothesis below.

Algorithm 3.1 A sub-algorithm for BGSP (BGSP_1)

Input: A matrix A that satisfies Hypothesis (A_1).
Output: Four matrices $\bar{Q}$, $\bar{S}$, $\bar{A}$, F.

1: **for** $\ell \in \mathbb{N}_{L-1}$ **do**
2: generate matrices Q^ℓ, R^ℓ by $\mathrm{orthn}(A^\ell_\Omega)$,
3: generate matrices $A^{\ell+1}$, C^ℓ by $\mathrm{proj}(A^\ell_\Theta, Q^\ell)$,
4: let $\Lambda^{\ell+1} := \Pi^\ell_\Theta$.
5: Let r be the column size of A^L, and F be the submatrix of F^{L-1} consisting of the last r rows of F^{L-1}.
6: **return** $\bar{Q} := [Q^\ell : \ell \in \mathbb{N}_{L-1}]$, $\bar{S} := \left[(S^\ell)^T : \ell \in \mathbb{N}_{L-1}\right]^T$, $\bar{A} := A^L$, F.

Hypothesis (P). Number $P \in \mathbb{N}$ with $\gamma \geq 4P$ and $P = 2^{J-1}$ for some $J \in \mathbb{N}$.

Then we partition A into P submatrices of A, each of which consists consecutive γ/P column blocks that inherit from Λ. We then apply BGSP_1 to each of the submatrices, execute concurrently, and construct the resulting matrices afterwards. That is, for each $p \in \mathbb{N}_P$, we let

$$\Gamma_p := [(p-1)\gamma/P + i : i \in \mathbb{N}_{\gamma/P}], \tag{3.1}$$

and let A_{Γ_p} be defined by (2.5). We also let $\bar{Q}_p$, $\bar{S}_p$, $\bar{A}_p$ be the output matrices generated by BGSP_1 from A_{Γ_p}, and define

$$\bar{Q} := [\bar{Q}_p : p \in \mathbb{N}_P], \quad \bar{S} := \bigoplus_{p \in \mathbb{N}_P} \bar{S}_p, \quad \bar{A} := [\bar{A}_p : p \in \mathbb{N}_P]. \tag{3.2}$$

Moreover, let

$$K := L - J,$$

and for $\ell \in \mathbb{N}_K$, let Q^ℓ, S^ℓ, $A^{\ell+1}$ be the matrices generated by BGSP_1 from A. Then we define

$$\tilde{Q} := [Q^\ell : \ell \in \mathbb{N}_K], \quad \tilde{S} := [(S^\ell)^T : \ell \in \mathbb{N}_K]^T, \quad \tilde{A} := A^{K+1}. \tag{3.3}$$

We present the lemma below for the concurrent computation.

Lemma 3.2.1. *Suppose that A, P satisfy Hypotheses (A_1), (P), respectively. If matrices $\bar{Q}$, $\bar{S}$, $\bar{A}$ are defined by* (3.2)*, and matrices $\tilde{Q}$, $\tilde{S}$, $\tilde{A}$ are defined by* (3.3)*, then there exists a permutation matrix H such that $\bar{Q} = \tilde{Q}H$, $\bar{S} = H^T\tilde{S}$, and $\bar{A} = \tilde{A}$.*

Proof. The proof follows from the definitions (3.2), (3.3), and an induction on $\ell \in \mathbb{N}_K$. For each $p \in \mathbb{N}_P$, we let Q_p^ℓ, S_p^ℓ, A_p^ℓ the matrices generated by BGSP_1 from A_{Γ_p}. Notice that A_{Γ_p} has γ/P column blocks, where $\gamma/P = 2^{K+1}$. Thus, according to BGSP_1, we have that

$$\bar{Q}_p = [Q_p^\ell : \ell \in \mathbb{N}_K], \quad \bar{S}_p = [(S_p^\ell)^T : \ell \in \mathbb{N}_K]^T, \quad \bar{A}_p = A_p^{K+1}.$$

Substituting this into formula (3.2) yields that

$$\bar{Q} = [Q_p^\ell : \ell \in \mathbb{N}_K, p \in \mathbb{N}_P], \quad \bar{S} = \bigoplus_{p\in\mathbb{N}_P} [(S_p^\ell)^T : \ell \in \mathbb{N}_K]^T, \quad \bar{A} = [A_p^{K+1} : p \in \mathbb{N}_P].$$

By the definition (3.3), it remains to show for $\ell \in \mathbb{N}_K$ that

$$Q^\ell = [Q_p^\ell : p \in \mathbb{N}_P], \tag{3.4}$$

$$S^\ell = \bigoplus_{p\in\mathbb{N}_P} S_p^\ell, \tag{3.5}$$

$$A^{\ell+1} = [A_p^{\ell+1} : p \in \mathbb{N}_P]. \tag{3.6}$$

This can be done by induction on $\ell \in \mathbb{N}_K$. Indeed, for $\ell = 1$, since

$$A^1 = A = [A_{\Gamma_p} : p \in \mathbb{N}_P] = [A_p^1 : p \in \mathbb{N}_P],$$

by the construction of Q^1 and Q_p^1 in orthn$(\cdot)$, we obtain formula (3.4). Likewise, the construction of S^1 and S_p^1 leads to formula (3.5). It follows from the construction of A^2 and A_p^2 in proj$(\cdot,\cdot)$ and formula (3.4) that equation (3.6) is true for $\ell = 1$. Assume that (3.4)-(3.6) are true for $\ell \in \mathbb{N}_{K-1}$. By the induction hypothesis, we have that $A^{\ell+1} = [A_p^{\ell+1} : p \in \mathbb{N}_P]$. Notice that $(\ell+1) \in \mathbb{N}_K$. Again, by the construction of $Q^{\ell+1}$,

$Q_p^{\ell+1}$, $S^{\ell+1}$, $S_p^{\ell+1}$, $A^{\ell+2}$, $A_p^{\ell+2}$, we obtain (3.4)-(3.6) for $(\ell+1)$. Hence, the induction principle leads to formulas (3.4)-(3.6). Therefore, we obtain the desired results. □

Notice that we cannot divide the computations in BGSP_1 into P parts for $\ell \geq K+1$. This is because A^ℓ has γ_ℓ column blocks, but $\gamma_\ell \leq 2^J$, and so $\gamma_\ell/P \leq 2$, violating the requirement of the input matrix of BGSP_1. However, we may divide into $P/2^{\ell-K}$ parts, in order to compute concurrently.

Now we design the data distribution with the purpose of avoiding possible communications. As discussed above, there are no interactions among the processors for orthn$(\cdot)$ as long as the input matrix A is distributed by column blocks. For proj$(\cdot,\cdot)$, communication is required when the matrices A_{2i-1}, A_{2i}, and Q_i are stored in different processors for some $i \in \mathbb{N}_\gamma$. Tracing back the calculations, we observe that the three matrices are computed from the submatrix $A_{\Gamma_i'}$ of A, where $\Gamma_i' := [4i-3, 4i-2, 4i-1, 4i]$. Thus, to avoid communication, we shall assign $A_{\Gamma_i'}$ as a unit to a processor. For $\ell \in \mathbb{N}_{L-1}$, we shall call such a submatrix $A^\ell_{\Gamma_i'}$ having the four consecutive column blocks of A^ℓ a *processing unit*. We conclude the data distribution as follows. Henceforth, we denote by P the number of processors or the number of cores in Message Passing Interface (MPI) environment, and $\mathbb{N}_P$ the set of the processors. If P satisfies Hypothesis (P), then for each $p \in \mathbb{N}_P$, we let Γ_p be defined by (3.1), and distribute the submatrix A_{Γ_p} of A to the processor p.

We next determine the supersteps according to the data distribution. Since each processor p holds γ/P consecutive column blocks of A, by the discussion above, it could independently run BGSP_1 with input A_{Γ_p}, and hold the output matrix $\bar{A}_p$ to be processed. Notice that $\bar{A}_p$ has only two column blocks. To form a processing unit,

the processor p has to receive the other two column blocks from a specific processor in order to continue. This is exactly the first time where communication is inevitable, and we mark the first superstep until here. In the following supersteps except the last, each one proceeds the communication phrase and computation phrase. During the communication phrase, each processor sends its local matrix to, and simultaneously receives a matrix from another specific processor. After then, each processor holds a processing unit, and processors run BGSP_1 concurrently in the computation phrase. In the last superstep, we shall compute the remaining results and output the factor matrices. These will be done without any communication among processors.

For the communication method, we will adopt a butterfly all-reduction tree structure. Reductions and all-reductions are collective communication operations that combine multiple results using some associative function into one overall result [27]. In the parallel case, a reduction leaves the overall result on exactly one processor, while an all-reduction leaves a copy of the overall result on all processors (see, for example, [47]). The process for the block QS factorization can be viewed as an (all-)reduction, with the specialty that the intermediate results (the local results of Q and S) must be stored in the processors that produce them. For communication, tree-based structure is an effective one for (all-)reductions since it adopts pairwise communication to avoid the barrier synchronization. We will use the butterfly all-reduction tree structure for its low communication costs and fault-tolerance. An example of butterfly all-reduction tree is demonstrated in Figure 3.1.

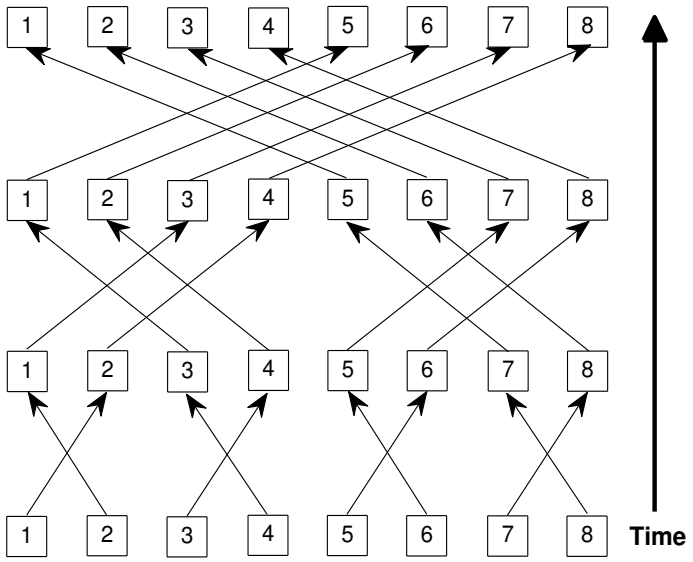


FIGURE 3.1: *Diagram of a butterfly all-reduction on a binary tree of eight processors. Each arrow represents data transmission from one processor to another. Time moves upwards.*

3.3 The Parallel Algorithm

In this section, we propose a parallel algorithm designed in the BSP model, then we verify that the algorithm indeed yields a block QS factorization for a block-wise quasi-orthogonal matrix of full rank. Moreover, we show that the parallel algorithm inherits the important features that BGSP owns such as the sparsity of factor matrices and stability, if the input matrix is banded.

With the discussion of the keystones of the BSP model in section 3.2, we present below the parallel algorithm called ParBGSP. Notice that this algorithm is executed in each processor $p \in \mathbb{N}_P$. There are $(J+1)$ many supersteps. Line 5 is the only one for communication, while the others are for concurrent computation. Equation (3.7) is to ensure the pairwise communication along the butterfly all-reduction tree.

We could construct the factor matrices Q and S as in (2.1) by the output of ParBGSP in all the processors. We present the algorithm called FormQS below. Notice

Algorithm 3.2 The parallel algorithm for the block QS factorization (ParBGSP)

Require: The set $\mathbb{N}_P$ of the P processors with $P = 2^{J-1}$ for some $J \in \mathbb{N}$, a butterfly all-reduction tree with height $\log_2 P$, my processor $p \in \mathbb{N}_P$.

Input: A matrix A of full rank partitioned by Λ of length $\gamma := 2^L$ for some $L \in \mathbb{N}$, and $L > J$.

Output: Submatrices of the factor matrices Q, S, stored locally in my processor.

1: Load the submatrix A_{Γ_p} of A into memory. Let $A_p^1 := A_{\Gamma_p}$.
2: Generate matrices Q_p^1, S_p^1, B_p^1, F_p^1 by BGSP_1 from A_p^1.
3: **for** $j \in \mathbb{N}_J \backslash \{1\}$ **do**
4: compute the processor q by

$$q := \begin{cases} p + 2^{j-2}, & \text{if } p \le \lceil p/2^{j-1} \rceil \cdot 2^{j-1} - 2^{j-2}, \\ p - 2^{j-2}, & \text{if } p > \lceil p/2^{j-1} \rceil \cdot 2^{j-1} - 2^{j-2}, \end{cases} \tag{3.7}$$

5: send B_p^{j-1} to q, and receive a matrix B_q^{j-1} from q,
6: let $r := \min\{p, q\}$, $s := \max\{p, q\}$, and let $A_p^j := \left[B_r^{j-1},\ B_s^{j-1}\right]$,
7: generate matrices Q_p^j, S_p^j, B_p^j, F_p^j by BGSP_1 from A_p^j.
8: Generate matrices Q_p^{J+1}, S_p^{J+1} by MGS from B_p^J.
9: **return** Q_p^j, S_p^j for $j \in \mathbb{N}_{J+1}$, and F_p^j for $j \in \mathbb{N}_J$.

that this algorithm will be run only if Q and S are required explicitly.

Algorithm 3.3 The construction of Q, S (FormQS)

Input: The matrices Q_p^j, S_p^j for $j \in \mathbb{N}_{J+1}$, and F_p^j for $j \in \mathbb{N}_J$ generated by ParBGSP from A, for each $p \in \mathbb{N}_P$.

Output: Two matrices Q', S'.

1: Let $\mathbb{P}^j \subset \mathbb{N}_P$ be a subset of the processors defined by

$$\mathbb{P}^j := \left\{2^{j-1}(i-1) + 1 : i \in \mathbb{N}_{2^{J-j}}\right\}, \text{ for } j \in \mathbb{N}_J, \quad \text{and} \quad \mathbb{P}^{J+1} := \{1\}. \tag{3.8}$$

2: For each $t \in \mathbb{N}_J$, let

$$\bar{F}^t := \bigoplus_{p \in \mathbb{P}^t} F_p^t. \tag{3.9}$$

3: For each $j \in \mathbb{N}_{J+1}$, let

$$\bar{Q}^j := [Q_p^j : p \in \mathbb{P}^j]. \tag{3.10}$$

$$\bar{S}^j := \bigoplus_{p \in \mathbb{P}^j} S_p^j, \tag{3.11}$$

$$\bar{V}^j := \bar{S}^j \cdot \prod_{t \in \mathbb{N}_{j-1}} \bar{F}^{j-t}. \tag{3.12}$$

4: **return** $Q' := [\bar{Q}^j : j \in \mathbb{N}_{J+1}]$, $S' := [(\bar{V}^j)^T : j \in \mathbb{N}_{J+1}]^T$.

We shall verify that ParBGSP fulfils the block QS factorization. This is done by the following theorem.

Theorem 3.3.1. *Suppose that A, P satisfy Hypotheses (A_1), (P), respectively. If matrices Q', S' are generated by ParBGSP and FormQS from A, and matrices Q, S are generated by BGSP from A, then there exists a permutation matrix H' such that*

$$Q' = QH', \quad S' = H'^T S. \tag{3.13}$$

Proof. We shall show (3.13) by Lemma 3.2.1 and an induction. With the notation in ParBGSP, we define

$$\bar{A}^j := [A_p^j : p \in \mathbb{P}^j], \text{ for } j \in \mathbb{N}_J, \quad \text{and} \quad \bar{A}^{J+1} := B_1^J. \tag{3.14}$$

According to lines 4-6 in ParBGSP and the definition (3.8), it can be verified for each $t \in \mathbb{N}_{J+1}\backslash\{1\}$ that

$$\bar{A}^t = [B_p^{t-1} : p \in \mathbb{P}^{t-1}]. \tag{3.15}$$

On the other hand, using the notation in BGSP, for $j \in \mathbb{N}_{J+1}\backslash\{1\}$, we let

$$\tilde{Q}^1 := [Q^\ell : \ell \in \mathbb{N}_K], \quad \tilde{Q}^j := Q^{K-1+j}, \tag{3.16}$$

$$\tilde{S}^1 := [(S^\ell)^T : \ell \in \mathbb{N}_K]^T, \quad \tilde{S}^j := S^{K-1+j}. \tag{3.17}$$

Then according to BGSP, we have that $Q = [\tilde{Q}^j : j \in \mathbb{N}_{J+1}]$ and $S = [(\tilde{S}^j)^T : j \in \mathbb{N}_{J+1}]^T$. By the definitions of Q' and S' in FormQS, it suffices to show for $j \in \mathbb{N}_J$ that there exists a permutation matrix H^j such that

$$\bar{Q}^j = \tilde{Q}^j H^j, \quad \bar{V}^j = (H^j)^T \tilde{S}^j. \tag{3.18}$$

We shall prove (3.18) and the following equation by induction on $j \in \mathbb{N}_J$.

$$\bar{A}^{j+1} = A^{K+j}. \tag{3.19}$$

Indeed, since $\bar{A}^1 = A$, by the definitions (3.10)-(3.12), (3.16), (3.17), and formula (3.15)

with $t = 2$, using Lemma 3.2.1, we obtain (3.18) and (3.19) for $j = 1$. Assuming that formula (3.19) holds for $j \in \mathbb{N}_{J-1}$, we shall show the case for $(j+1)$ using Lemma 3.2.1 again. Notice that A^{K+j} is of full rank having γ_{K+j} column blocks, and $\gamma_{K+j} = 2^{J-j+1}$ by its definition (2.10). Let $|\mathbb{P}^{j+1}|$ denote the cardinality of $\mathbb{P}^{j+1}$. By the definition (3.8), we have that $|\mathbb{P}^{j+1}| = 2^{J-j-1}$, and hence $\gamma_{K+j} = 4|\mathbb{P}^{j+1}|$. By the induction hypothesis, we have that $\bar{A}^{j+1} = A^{K+j}$. Thus, applying Lemma 3.2.1 to A^{K+j} and $|\mathbb{P}^{j+1}|$, together with equations (3.9)-(3.12), (3.14)-(3.17), we obtain the desired results for $(j+1)$. By the induction principle, we have (3.18) and (3.19). Employing formula (3.19) for $j = J$ yields that $\bar{A}^{J+1} = A^L$. According to ParBGSP, BGSP and the definitions (3.10)-(3.12), (3.16), (3.17), we obtain that

$$\bar{Q}^{J+1} = \tilde{Q}^{J+1}, \quad \bar{V}^{J+1} = \tilde{S}^{J+1}. \tag{3.20}$$

Let $H^{J+1} := I$, and let $H' := \bigoplus_{j \in \mathbb{N}_{J+1}} H^j$. Since each H^j is a permutation matrix, so is H'. Formulas (3.18), (3.20) and the definitions of Q', S' in FormQS yield the desired formula (3.13). □

We next confirm that ParBGSP generates sparse factor matrices for the block QS factorization of a banded matrix A that satisfies the following hypothesis.

Hypothesis (A2). Matrix $A \in \mathbb{R}^{n \times m}$ is of full rank and is banded with bandwidth $k/2$, $n \geq m$, $m = 2^L \mu k$ for $\mu \in \mathbb{N}$, $L \in \mathbb{N} \backslash \{1\}$, $\gamma := 2^L$, and A is partitioned by the vector $\Lambda := [\mu k, \mu k, \ldots, \mu k]$ of length γ.

Comparing to Hypothesis (A), here we introduce a parameter μ in the partition vector Λ of A. The introduction is to optimize local computational performance of each processor, as follows. It can be seen that almost all the computations in ParBGSP

operate on column blocks of size $n \times \mu k$. When μ is too small, those computations may be inefficient since the time spending in memory access is comparable to that in the actual computing. When μ is larger, the computational complexity will become higher. Hence we shall choose an appropriate μ to optimize the computational efficiency. The effectiveness of this treatment will be verified in our numerical experiments to be presented later.

We present the theorem below for the sparsity of the factor matrices.

Theorem 3.3.2. *Suppose that A, P satisfy Hypotheses (A2), (P), respectively. If Q', S' are the matrices generated by ParBGSP and FormQS from A, then the numbers of nonzero entries of Q' and S' are bounded above by $2\mu km \log_2(\frac{m}{\mu k})$ and $\frac{13}{4}\mu km$, respectively.*

Proof. By Theorem 3.3.1, there exists a permutation matrix H' such that formula (3.13) holds. Notice that a banded matrix with bandwidth $k/2$ also has bandwidth $\mu k/2$. The desired results follow from Theorems 2.3.3 and 2.3.4, replacing each k by μk. □

We next verify that ParBGSP is stable.

Theorem 3.3.3. *Suppose that A, P satisfy Hypotheses (A2), (P), respectively, and that the column vectors of A have the same 2-norm. Assuming Hypothesis (F) and* (2.59)*, if $\hat{Q}'$ and $\hat{S}'$ are respectively the computed matrices of Q' and S' generated by ParBGSP and FormQS from A, then*

$$\|A - \hat{Q}'\hat{S}'\|_F \leq 3\sqrt{6}\mu k\mathbf{u}\|A\|_F.$$

Proof. By Theorem 3.3.1, there exists a permutation matrix H' such that formula (3.13) holds. Thus by the construction of factor matrices in FormQS and in BGSP, we conclude that

$$\hat{Q}' = \hat{Q}H', \quad H'\hat{S}' = \hat{S}, \tag{3.21}$$

where $\hat{Q}$, $\hat{S}$ are the computed matrices of Q and S, respectively. The desired result follows from formula (3.21) and Theorem 2.5.8, replacing k by μk. □

Likewise, we can obtain an upper bound for the loss of orthogonality of $\hat{Q}'$ by formula (3.21) and Theorem 2.5.9, as follows.

Theorem 3.3.4. *Suppose that A, P satisfy Hypotheses (A2), (P), respectively. Let $\hat{Q}'$ be the computed matrix of the column orthonormal matrix Q generated by ParBGSP and FormQS from A. Assuming Hypothesis (F) and* (2.59)*, if* $3.42m(m+1)\mathbf{u}\kappa(A) < 1$*, then there exists a constant* $c := c(m)$ *such that*

$$\|(\hat{Q}')^T\hat{Q} - I\|_2 \leq c\mathbf{u}\kappa(A).$$

3.4 Analysis of Parallel Performance

In this section, we study the parallel performance of ParBGSP, including complexity, communication costs, and scalability. We shall define these terminologies under the context of the BSP model, then give a detailed analysis about the performance of ParBGSP.

We review the definitions of the parallel complexity, the number of communication messages, the number of communication words, and the speedup in the BSP model. In

a communication phase, messages are created with which data are transmitted among processors. The transmit data is measured by words. Recall that $(J+1)$ is the number of supersteps and P is the number of processors. For each $j \in \mathbb{N}_{J+1}$, $p \in \mathbb{N}_P$, we denote by c_{jp}, m_{jp}, w_{jp}, the number of flops in local computations, the number of messages and the number of words sent or received (occur simultaneously) of a processor in a superstep, respectively. Then the parallel complexity $\mathcal{C}$, the number of communication messages $\mathcal{M}$, and the number of communication words $\mathcal{W}$ are defined by

$$\mathcal{C} := \sum_{j\in\mathbb{N}_{J+1}} \max_{p\in\mathbb{N}_P}\{c_{jp}\}, \quad \mathcal{M} := \sum_{j\in\mathbb{N}_{J+1}} \max_{p\in\mathbb{N}_P}\{m_{jp}\}, \quad \mathcal{W} := \sum_{j\in\mathbb{N}_{J+1}} \max_{p\in\mathbb{N}_P}\{w_{jp}\}. \quad (3.22)$$

Also, if τ_c, τ_m, τ_w and τ_s are respectively the unit time for computing a flop, creating a message, sending or receiving a word, and barrier synchronization, then the parallel running time T_{par} of an algorithm is defined by

$$T_{par} := \mathcal{C}\tau_c + \mathcal{M}\tau_m + \mathcal{W}\tau_w + (J+1)\tau_s.$$

If T_{seq} is the running time of a sequential algorithm versus the given parallel algorithm (for example, BGSP versus ParBGSP), then the speedup is defined by

$$\text{Speedup} := T_{seq}/T_{par}. \quad (3.23)$$

Notice that the speedup is a function of P and it is the indicator of the scalability of a parallel algorithm. An algorithm has the ideal scalability if the speedup is linear in P (with coefficient 1), in which case $\mathcal{C}\tau_c = T_{seq}/P$ and the time for communication is zero.

We next analyze the parallel complexity of ParBGSP. To this end, we establish the following lemma for c_{jp}. Recall that for each $j \in \mathbb{N}_J\backslash\{1\}$, A_p^j has 4 column blocks. In

the following, for each $i \in \mathbb{N}_4$, we let G_{ip}^j denote the submatrix of A_p^j such that

$$A_p^j := [G_{ip}^j : i \in \mathbb{N}_4]. \tag{3.24}$$

We also define

$$c := 24\mu^2 k^2 m.$$

Lemma 3.4.1. *Suppose that A, P satisfy Hypotheses (A2), (P), respectively. If ParBGSP is applied to A with P processors, then*

$$c_{jp} \leq \begin{cases} cK/P, & j = 1, \\ 2^{j-1}c/P, & j \in \mathbb{N}_J \backslash \{1\}, \\ c/3, & j = J+1. \end{cases} \tag{3.25}$$

Proof. The result for $j = 1$ can be derived from Theorem 2.4.2. Let $p \in \mathbb{N}_P$ be given. According to line 2 in ParBGSP, BGSP_1 is applied to A_{Γ_p}. By Hypothesis (A2) and the definition (3.1) of Γ_p, the matrix A_{Γ_p} has banded structure that each entry a_{rs} of A_{Γ_p} satisfies that $a_{rs} = 0$, for $|(r-t)-s| > k/2$, where

$$t := (p-1)\gamma\mu k/P$$

is the number of column vectors prior to A_{Γ_p} as a submatrix of A. With the fact that $A_{\Gamma_p} \in \mathbb{R}^{n\times(m/P)}$ has 2^{K+1} column blocks, each of which has μk column vectors, it follows from Theorem 2.4.2 that

$$c_{1p} \leq 24(\mu k)^2 (m/P)K = cK/P.$$

We next prove (3.25) for $j \in \mathbb{N}_J \backslash \{1\}$. By formula (3.19) and Lemma 2.3.2, we have

that

$$\eta\left(G_{ip}^{j}\right) \leq 2^{K+j-1}\mu k, \quad \text{for } i \in \mathbb{N}_4,$$

$$\eta\left([G_{2p}^{j}, G_{3p}^{j}]\right) \leq 2^{K+j}\mu k.$$

According to line 7 in ParBGSP, with the above two formulas, it follows from the complexity statement for MGS and Lemma 2.4.1 that

$$c_{jp} \leq 2(2^{K+j}\mu k)(2\mu k)^2 + 2[4(2^{K+j}\mu k)(2\mu k)(\mu k)] = 2^{j-1}c/P.$$

For $j = J+1$, it can be verified by ParBGSP that $B_p^J \in \mathbb{R}^{n\times 2\mu k}$ and $\eta\left(B_p^J\right) \leq m$. According to MGS, $c_{_{J+1},p} \leq 2m(2\mu k)^2 = c/3$. Therefore, formula (3.25) holds. □

We present the parallel complexity of ParBGSP in the next theorem.

Theorem 3.4.2. *Suppose that A, P satisfy Hypotheses (A2), (P), respectively. If ParBGSP is applied to A with P processors, then the parallel complexity is at most $c\big(7/3 + (K-2)/P\big)$ flops.*

Proof. The proof follows directly from Lemma 3.4.1. By the definition (3.22) of $\mathcal{C}$, the parallel complexity is bounded above by

$$c\Big(K/P + \sum_{j=2}^{J} 2^{j-1}/P + 1/3\Big) = c\big(7/3 + (K-2)/P\big) \text{ flops},$$

proving the desired result. □

We shall improve the result of the parallel complexity when the input matrix A is large-scale and ill-conditioned. To start with, we discuss the computations in BGSP applied to A. In particular, we consider the matrix A^L. Recall that $A^L(\Lambda^L) = [G_1^L, G_2^L]$ by (2.24). Let $M := [G_i^1 : i \in \mathbb{N}_{\gamma-1}\backslash\{1\}]$. Then $A = [G_1^1, M, G_\gamma^1]$. Let $Z_1 := G_1^1 - G_1^L$. We can see from formulas (2.25) and (2.32) that $\mathscr{R}(Z_1) \subset \mathscr{R}(M)$. Since Z_1 is derived

from an ill-conditioned matrix A, the column vectors of Z_1 and the basis vectors of a subspace of $\mathscr{R}(M)$ are nearly linear dependent. Hence for sufficiently large size of M, we may assume that $\mathscr{R}(Z_1) \subset \mathscr{R}(M_1)$ for a submatrix M_1 of M, with $M_1 \neq M$. Notice that $M^T G_1^L = 0$ by (2.30). Hence Z_1 is indeed the projection of G_1^1 onto $\mathscr{R}(M)$, and it is also the projection of G_1^1 onto $\mathscr{R}(M_1)$. Likewise, $Z_2 := G_\gamma^1 - G_2^L$ is the projection of G_γ^1 onto $\mathscr{R}(M)$, and that onto $\mathscr{R}(M_2)$ for a submatrix M_2 of M, with $M_2 \neq M$. Recall that for matrices X, Y having the same row size, if Y is of full rank, then the projection of X onto $\mathscr{R}(Y)$ is unique and is equal to $(Y(Y^TY)^{-1}Y^T)X$ [108]. With the discussion, we present the sparse structure of A^L in the following lemma.

Lemma 3.4.3. *Suppose that A satisfy Hypothesis (A2) with $\gamma \geq 8$. Let $M_1 := [G_i^1 : i \in \mathbb{N}_{\gamma/2-1}\backslash\{1\}]$, $M_2 := [G_{i+\gamma/2}^1 : i \in \mathbb{N}_{\gamma/2-1}\backslash\{1\}]$, and $M := [M_1, G_{\gamma/2}^1, G_{\gamma/2+1}^1, M_2]$. If*

$$M(M^TM)^{-1}M^TG_1^1 = M_1(M_1^TM_1)^{-1}M_1^TG_1^1, \tag{3.26}$$

$$M(M^TM)^{-1}M^TG_\gamma^1 = M_2(M_2^TM_2)^{-1}M_2^TG_\gamma^1, \tag{3.27}$$

and $A^L = [G_1^L, G_2^L]$ is the matrix generated by BGSP from A, then $\bar{\eta}\left(G_1^L\right) < \underline{\eta}\left(G_2^L\right)$.

Proof. By formulas (2.25) and (2.41), we have for each $\ell \in \mathbb{N}_L\backslash\{1\}$ that

$$G_1^\ell = \left(\prod_{t\in\mathbb{N}_{\ell-1}} (I - Q_1^{\ell-t}(Q_1^{\ell-t})^T)\right)G_1^1, \tag{3.28}$$

$$G_{\gamma_\ell}^\ell = \left(\prod_{t\in\mathbb{N}_{\ell-1}} (I - Q_{\gamma_{\ell-t+2}}^{\ell-t}(Q_{\gamma_{\ell-t+2}}^{\ell-t})^T)\right)G_\gamma^1. \tag{3.29}$$

We first consider G_1^L. By formulas (2.21) and (2.41), we have that $(Q_r^\ell)^T G_1^\ell = 0$ for $r \in \mathbb{N}_{\gamma_\ell/4}\backslash\{1\}$. If follows from formulas (3.28), (2.20) and the column orthonormality of Q^ℓ that

$$G_1^L = \left(\prod_{t\in\mathbb{N}_{L-1}} (I - Q^{L-t}(Q^{L-t})^T)\right)G_1^1.$$

Let $W := [Q^\ell : \ell \in \mathbb{N}_{L-1}]$. Since W is column orthonormal, it can be verified by direct computation that

$$G_1^L = (I - WW^T)G_1^1.$$

Employing formula (2.32) with $\ell = L - 1$, we have that $\mathscr{R}(W) = \mathscr{R}(M)$. Since both W and M are of full rank, there exists an invertible matrix X such that $M = WX$. It follows from direct computation that $M(M^TM)^{-1}M^T = WW^T$. Hence,

$$G_1^L = (I - M(M^TM)^{-1}M^T)G_1^1. \tag{3.30}$$

Likewise, let $W_1 := [Q_i^\ell : i \in \mathbb{N}_{\gamma_\ell/8}, \ell \in \mathbb{N}_{L-2}]$, we can derive that

$$G_1^{L-1} = (I - W_1W_1^T)G_1^1 = (I - (M_1^TM_1)^{-1}M_1^T)G_1^1. \tag{3.31}$$

Formulas (3.26), (3.30), and (3.31) yields that $G_1^L = G_1^{L-1}$. By a similar proof as above with formulas (3.27) and (3.29), we can obtain that $G_2^L = G_4^{L-1}$. Replacing k by μk in Lemma 2.3.2, we have that

$$\bar{\eta}\left(G_1^L\right) = \bar{\eta}\left(G_1^{L-1}\right) \le 2^{L-1}k - k/2,$$

$$\underline{\eta}\left(G_2^L\right) = \underline{\eta}\left(G_4^{L-1}\right) \ge 2^{L-1}k + k/2 + 1.$$

Therefore, $\bar{\eta}\left(G_1^L\right) < \underline{\eta}\left(G_2^L\right)$. □

Now we analyze the parallel complexity of ParBGSP under the following hypothesis. Recall that $A = [A_{\Gamma_p} : p \in \mathbb{N}_P]$, where Γ_p is defined by (3.1).

Hypothesis (AP). Matrix A satisfies Hypothesis (A2) with $\gamma \ge 8$. Number $P \in \mathbb{N}$ with $\gamma > 4P$ and $P = 2^{J-1}$ for some $J \in \mathbb{N}$. For each $p \in \mathbb{N}_P$, let $G_{1p}, G_{2p}, G', G'' \in \mathbb{R}^{n\times\mu k}$ and $M_{1p}, M_{2p} \in \mathbb{R}^{n\times(\frac{\gamma}{2P}-2)\mu k}$ be the submatrices of A_{Γ_p} such that

$$A_{\Gamma_p} = [G_{1p}, M_{1p}, G', G'', M_{2p}, G_{2p}].$$

Let $M_p := [M_{1p}, G', G'', M_{2p}]$. Then

$$M_p(M_p^T M_p)^{-1} M_p^T G_{1p} = M_{1p}(M_{1p}^T M_{1p})^{-1} M_{1p}^T G_{1p}, \tag{3.32}$$

$$M_p(M_p^T M_p)^{-1} M_p^T G_{2p} = M_{2p}(M_{2p}^T M_{2p})^{-1} M_{2p}^T G_{2p}. \tag{3.33}$$

We comment on Hypothesis (AP). Equation (3.32) means that the projection of each column vector of the first block onto $\mathscr{R}(M_p)$ is in a subspace of $\mathscr{R}(M_p)$, and the subspace is spanned by approximately the first half of the column vectors of M_p. Notice that A_{Γ_p} has banded structure; The column vectors of the first block and any other column vector of A_{Γ_p} whose index greater than $\mu k + k$ are linear independent; If the projection of a vector onto $\mathscr{R}(M_p)$ is equal to the projection of the vector onto a subspace of $\mathscr{R}(M_{1p})$, then it must be equal to the projection of the vector onto $\mathscr{R}(M_{1p})$ (which can be proved by statements on the bases of these linear spaces). Hence, if the matrix A is ill-conditioned and banded, having sufficiently large size, then equation (3.32) will be satisfied. Equation (3.33) can be understood in a similar way. In chapter 5, we shall verify Hypothesis (AP) through parallel experiments (see, Table 5.7).

For the improvement of parallel complexity, we present the following two technical lemmas using Hypothesis (AP). The lemma below concerns the sparse structure of the matrix B_p^1 generated by BGSP_1 from A_{Γ_p}, as on line 2 of ParBGSP. As an example, Figure 3.2 below illustrates the sparse structure of B_p^1.

Lemma 3.4.4. *Suppose that A, P satisfy Hypothesis (AP). Let $p \in \mathbb{N}_P$ be given. If matrix B_p^1 is generated by $BGSP_1$ from A_{Γ_p}, and matrices G'_{1p}, G'_{2p} are the two column blocks of B_p^1 such that $B_p^1 = [G'_{1p}, G'_{2p}]$, then $\bar{\eta}\left(G'_{1p}\right) < \underline{\eta}\left(G'_{2p}\right)$.*

Proof. The proof directly follows from Hypothesis (AP) and Lemma 3.4.3. □

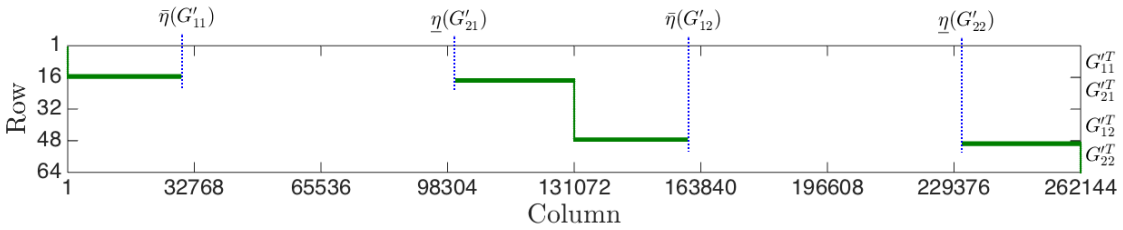


FIGURE 3.2: *The sparse structure of the matrix $[B^1_1, B^1_2]^T$ in Example 6(T1) for matrix size $n = 2^{26}$, $\mu = 8$, and $P = 512$. A dot in the graph is a nonzero entry located in the specific row and column. The blank spaces are zero entries.*

We next establish a technique lemma regarding BGSP.

LEMMA 3.4.5. *Suppose that A, P satisfy Hypothesis (AP). If BGSP is applied to A, then for $\ell = K+1, K+2, \ldots, L-1$,*

$$\begin{aligned}
\underline{\eta}\left(G_i^\ell\right) \;\geq\; & \begin{cases} 2^{\ell-1}(i-1)\mu k - \mu k/2 + 1, & i = 1,3,\ldots,\gamma_\ell - 1, \\ 2^{\ell-1} i \mu k + \mu k/2 + 1 - 2^{K+1}\mu k, & i = 2,4,\ldots,\gamma_\ell, \end{cases} \\
\bar{\eta}\left(G_i^\ell\right) \;\leq\; & \begin{cases} 2^{\ell-1}(i-1)\mu k - \mu k/2 + 2^{K+1}\mu k, & i = 1,3,\ldots,\gamma_\ell - 1, \\ 2^{\ell-1} i \mu k + \mu k/2, & i = 2,4,\ldots,\gamma_\ell. \end{cases}
\end{aligned} \tag{3.34}$$

Proof. We shall prove the results by induction on $\ell = K+1, K+2, \ldots, L-1$. Indeed, replacing k by μk in Lemma 2.3.2, we obtain formula (3.34) for $\ell = K+1$. By Hypothesis (AP), we have that A_{Γ_p} has 2^{K+1} column blocks. Hence by Lemma 3.4.4, we have for each $j \in \mathbb{N}_{\gamma_{K+2}}$ that

$$\bar{\eta}(G^{K+1}_{2j-1}) < \underline{\eta}(G^{K+1}_{2j}).$$

Let $G := [G^{K+1}_{4i-2}, G^{K+1}_{4i-1}]$. By the above inequality and the fact that $\bar{\eta}(G^{K+1}_{2j-1}) < \underline{\eta}(G^{K+1}_{2j+1})$, $\bar{\eta}(G^{K+1}_{2j-2}) < \underline{\eta}(G^{K+1}_{2j})$, we have for each $i \in \mathbb{N}_{\gamma_{K+3}}$ that

$$\bar{\eta}(G^{K+1}_{4i-3}) < \underline{\eta}(G), \quad \underline{\eta}(G^{K+1}_{4i}) > \bar{\eta}(G).$$

Thus, $G^T G_{4i-3}^{K+1} = 0$ and $G^T G_{4i}^{K+1} = 0$. It follows from (2.43) that

$$G_i^{K+2} = \begin{cases} G_{2i-1}^{K+1}, & i = 1, 3, \ldots, \gamma_{K+2} - 1, \\ G_{2i}^{K+1}, & i = 2, 4, \ldots, \gamma_{K+2}. \end{cases}$$

Employing this equation and (3.34) with $\ell = K + 1$, we obtain that formula (3.34) holds for $\ell = K + 2$.

Assume that (3.34) is true for $K + 2 \leq \ell \leq L - 2$. We consider the case $(\ell + 1)$. Let $G' := [G_{4i-2}^{\ell}, G_{4i-1}^{\ell}]$. By the induction hypothesis, we can verify that

$$\bar{\eta}(G_{4i-3}^{\ell}) < \underline{\eta}(G'), \quad \underline{\eta}(G_{4i}^{\ell}) > \bar{\eta}(G').$$

Again by (2.43) and the induction hypothesis, we obtain formula (3.34) for $(\ell + 1)$.

By the induction principle, we obtain formula (3.34). □

We present the theorem below for the parallel complexity of ParBGSP applied to a large-scale ill-conditioned banded matrix A.

Theorem 3.4.6. *Suppose that A, P satisfy Hypothesis (AP). If ParBGSP is applied to A with P processors, then the parallel complexity is at most $cP^{-1}\log_2(\frac{m}{\mu k \sqrt[3]{P}})$ flops.*

Proof. Given $p \in \mathbb{N}_P$, we derive a new upper bound for c_{jp} for $j \in \mathbb{N}_J \backslash \{1\}$. Let $G := [G_{2p}^j, G_{3p}^j]$. Employing formulas (3.19) and (3.34) in Lemma 3.4.5, we have that

$$\bar{\eta}\left(G_{1p}^j\right) < \underline{\eta}\left(G\right), \quad \underline{\eta}\left(G_{4p}^j\right) > \bar{\eta}\left(G\right), \tag{3.35}$$

$$\eta\left(G\right) \leq 2^{K+2}\mu k, \tag{3.36}$$

$$\eta\left(G_{ip}^j\right) \leq 2^{K+1}\mu k, \quad i = 1, 2, 3, 4. \tag{3.37}$$

Formula (3.35) yields that the processes $\mathrm{proj}(G_{1p}^j, G)$ and $\mathrm{proj}(G_{4p}^j, G)$ are unnecessary by Lemma 2.5.2. According to line 7 in ParBGSP, with (3.36), it follows from the

complexity statement for MGS that

$$c_{jp} \leq 2(2^{K+2}\mu k)(2\mu k)^2 \leq 2c/(3P). \tag{3.38}$$

Notice that formula (3.38) is also true for $j = J + 1$ by (3.37) and by ParBGSP.

Employing formula (3.25) for $j = 1$ and formula (3.38), we obtain that the parallel complexity is bounded above by $cP^{-1}(K + \frac{2}{3}J) \leq cP^{-1}\log_2(\frac{m}{\mu k \sqrt[3]{P}})$ flops. □

Theorem 3.4.6 is an important improvement over Theorem 3.4.2. We see from Theorem 3.4.6 that the parallel complexity is inversely proportional to the number of processors. Formula (3.37) derived in the proof is also benefit for reducing the communication costs, to be discussed below.

We next concern about the communication costs of ParBGSP. We first prove in the theorem below that the number of communication messages of ParBGSP reaches the least upper bound for collective communication operations [27].

Theorem 3.4.7. *Suppose that A, P satisfy Hypotheses (A2), (P), respectively. If ParBGSP is applied to A with P processors, then the number of communication messages is* $\log_2 P$.

Proof. Let $j \in \mathbb{N}_J \backslash \{1\}$ and $p \in \mathbb{P}^j$ be given. From line 5 of ParBGSP, the processor p sends (receives) exactly 1 message to (from) the processor q. Since both send and receive operations occur simultaneously, by the definition (3.22), the number of communication messages is $J - 1 = \log_2 P$. □

We present the theorem below for the number of communication words. In the following, we assume that one entry of a matrix equals one piece of data, having the size of one computer word.

Theorem 3.4.8. *Suppose that A, P satisfy Hypothesis (AP). If ParBGSP is applied to A with P processors, then the number of communication words is at most* $2\mu kmP^{-1}\log_2 P$.

Proof. The proof follows from counting the number of communication words for each superstep and for each processor. Let $j \in \mathbb{N}_J\backslash\{1\}$ and $p \in \mathbb{P}^j$ be given. From line 5 of ParBGSP, the processor p sends the matrix B_p^{j-1} to the processor q, and simultaneously receives the matrix B_q^{j-1} from q. Let $r := \min\{p,q\}$, $s := \max\{p,q\}$. By line 6 and formula (3.24), we obtain that $B_r^{j-1} = [G_{1p}^j, G_{2p}^j]$ and $B_s^{j-1} = [G_{3p}^j, G_{4p}^j]$. By (3.37), we have for $t = r$, s that $\#(B_t^{j-1}) \le 2^{K+2}\mu^2k^2$. It follows from the definition (3.22) that the number of communication words is at most

$$\sum_{j\in\mathbb{N}_J\backslash\{1\}} \#(B_t^{j-1}) \le 2^{K+2}\mu^2k^2(J-1) = 2\mu kmP^{-1}\log_2 P,$$

proving the desired result. □

We now analyze the scalability of ParBGSP. To this end, we estimate below the sequential complexity of BGSP applied to a large-scale ill-conditioned matrix.

Lemma 3.4.9. *Suppose that A, P satisfy Hypothesis (AP). If BGSP is applied to A, then the sequential complexity is at most* $c\log_2(\frac{m}{\mu kP})$ *flops.*

Proof. For $\ell \in \mathbb{N}_K$, it follows from the proof of Theorem 2.4.2 that the number of flops in the ℓth step is at most c. For $\ell = K+1, K+2, \ldots, L$, it can be proved from Lemma 3.4.5 and the complexity statement for MGS that the number of flops in the ℓth step is bounded above by

$$2(2^{K+2}\mu k)(2\mu k)^2 \cdot \gamma_\ell/4 = (2c/3)(2^{K-\ell}).$$

Therefore, the sequential complexity is at most $c(K+2/3) \leq c\log_2 \frac{m}{\mu k P}$ flops. □

We present in the theorem below the speedup of ParBGSP. In the following theorem, we let $c_1 := \tau_m/\tau_c, c_2 := \tau_w/\tau_c$ be the constant ratios. Notice that the speedup depends on the actual (not the upper bounds of) sequential and parallel complexity. With Lemma 3.4.9, it is reasonable to assume for some constant $c_0 \in [1, 24]$ that the number of flops in BGSP is equal to $c_0\mu^2k^2m\log_2(\frac{m}{\mu k P})$.

Theorem 3.4.10. *Suppose that A, P satisfy Hypothesis (AP). If ParBGSP is applied to A with P processors, then the speedup is at least* $c_0 P\left[24 + (8 + \frac{c_1 P}{8m} + \frac{c_2}{2})\log_2 P\right]^{-1}$.

Proof. With the assumption right above, we compute the speedup by definition. Recall that no barrier synchronization is required for ParBGSP. Thus by the definition (3.23), Theorems 3.4.6-3.4.8, we have that the speedup is at least

$$\begin{aligned}\text{Speedup} &\geq c_0(K+1)/\left[\tfrac{24}{P}\big(K+1+\tfrac{2}{3}\log_2 P\big) + \tfrac{c_1}{4m}\log_2 P + \tfrac{c_2}{P}\log_2 P\right] \\ &\geq c_0 P\left[24 + (8 + \tfrac{c_1 P}{8m} + \tfrac{c_2}{2})\log_2 P\right]^{-1},\end{aligned}$$

proving the desired result. □

We comment on the result in Theorem 3.4.10. In practical applications, c_1, c_2 are mediocre numbers, and $P \ll m$. Thus, the term $\frac{c_1 P}{8m}$ is negligible. Hence, we conclude that the speedup is at least $P/(\bar{c}\log_2 P)$ for a mediocre number $\bar{c}$.

We finally summarize the parallel performance of ParBGSP. Noticing that $\mu = 1, 2, 4$, or 8 in practical, we conclude the results in Table 3.1 below, using the optimal bounds derived from [7]. In the table, except for the last row, the optimal bound is the theoretical least upper bound, while the bound for the last row is the theoretical greatest lower bound. Note that l is said to be the theoretical least upper bound of

a quantity q if any upper bound of q derived theoretically is at least l, while g is the theoretical greatest lower bound of q if any lower bound of q derived theoretically is at most g. For functions $f, g, h : \mathbb{N}^d \to \mathbb{R}$, $d \in \mathbb{N}$, we say f is $O(g)$ if $|f|$ is asymptotically bounded above by g, while f is $\Omega(h)$ if f is asymptotically bounded below by h. That is,

$$\limsup_{\substack{\mathbf{x}=[x_1,x_2,\ldots,x_d]\in\mathbb{N}^d,\\ x_i\to\infty,\forall i}} \frac{|f(\mathbf{x})|}{g(\mathbf{x})} < \infty, \qquad \liminf_{\substack{\mathbf{x}=[x_1,x_2,\ldots,x_d]\in\mathbb{N}^d,\\ x_i\to\infty,\forall i}} \frac{f(\mathbf{x})}{h(\mathbf{x})} > 0.$$

Also, we say f is $\Theta(g)$ if f is $O(g)$ and f is $\Omega(g)$.

TABLE 3.1
Summary of the parallel performance of ParBGSP.

	ParBGSP	optimal bound
parallel complexity in flops	$O\big(\frac{k^2m}{P}\log_2(\frac{m}{k\sqrt[3]{P}})\big)$	$\Theta(\frac{k^2m}{P})$
number of communication messages	$\log_2 P$	$\log_2 P$
number of communication words	$O(\frac{km}{P}\log_2 P)$	$\frac{km}{2P}\log_2 P$
speedup	$\Omega(P/\log_2 P)$	P

Chapter 4

Application on Solving Linear Systems

In this chapter, we apply the block QS factorization to solve linear systems and present both sequential and parallel solvers for ill-conditioned banded systems.

The critical concern for developing the sequential solver is the issue of ill-conditioning. In general, we may assume that for a structured matrix, its condition number increases as its size increases. With this assumption, we propose a block row projection solver for ill-conditioned banded systems. This solver breaks down a given linear system into small-scale underdetermined systems, compute their minimum 2-norm solutions, and form the solution of the original linear system. Since the underdetermined systems are of small sizes, we expect that their minimum 2-norm solutions can be computed with high accuracy. Indeed, we have derived the condition numbers for solving these underdetermined systems, using the stability results derived from chapter 2, as well as the perturbation theories for linear sytems, underdetermined systems, and the QR

factorization. Through this study, we expect that these condition numbers should be far less than the condition number of the matrix of the given linear system. This point is confirmed by numerical experiments in chapter 5. Moreover, computations for the minimum 2-norm solutions can be done during BGSP, neither storing the generated matrices in memory over the entire process, or forming explicitly the factor matrices. We say this feature is *memory-less*.

For the parallel solver, the main problem is to treat the memory issue in solving extra large-scale linear systems. In addition to utilizing the memory-less feature, we employ a partial load strategy in the implementation of the solver. Notice that the partial load strategy is widely used in designing external memory algorithms when memory shortage is an issue [112, 113]. The concept is to sequentially load partial data that fit in memory, compute partial results, and then combine as one. We study it in detail and propose the parallel solver at the end of this chapter.

We organize this chapter in three sections. In section 4.1, we propose the sequential solver that is able to deal with the issue of ill-conditioning. We then derive an upper bound for the relative error of the solution obtained by the sequential solver in section 4.2. In section 4.3, we discuss the memory issue in solving extra large-scale linear systems, and propose a parallel solver to overcome the issue.

4.1 A Sequential Solver and the Ill-conditioning

We consider in this section solving an ill-conditioned banded system by applying the block QS factorization. For such a linear system with coefficient matrix A, we propose a block row projection solver via the block QS factorization of A^T. This idea originates from the direct row projection method via the LU factorization with permutations described in [20], but the proposed solver partitions the linear system by row blocks and uses the block QS factorization. Indeed, we break down the linear system into small-scale underdetermined systems, to which minimum 2-norm solutions can be computed by the matrices generated by BGSP applied to A^T. This BGSP solver is suitable for handling ill-conditioned matrices, because for a structured ill-conditioned matrix, its condition number usually increases superlinearly on its size.

Linear systems to be solved have ill-conditioned banded matrices. Let $A \in \mathbb{R}^{n\times n}$ be an ill-conditioned banded matrix of full rank with bandwidth $k/2$, $n = 2^L k$ for $L \in \mathbb{N}$, and let $b \in \mathbb{R}^n$ be a vector. We wish to solve the linear system

$$Ax = b. \tag{4.1}$$

We say matrix A is ill-conditioned if $\mathbf{u}\kappa(A) \geq 1$, in which case, the condition of solving (4.1) stably is violated in perturbation theories (see, for example, [43]).

To solve (4.1), we express its solution by a sum of minimum 2-norm solutions to underdetermined systems of small scale that are derived from (4.1). Observing that $x \in \mathscr{R}(A^T)$, we partition A^T as $A^T = [A_1, A_2]$ with $A_1 \in \mathbb{R}^{n\times s}$, and partition b as $b = [b_1^T, b_2^T]^T$ with $b_1 \in \mathbb{R}^s$. We express x as a sum of x_1 and x_2, where $x_1 \in \mathscr{R}(A_1)$,

and x_2 is in the orthogonal complement of $\mathscr{R}(A_1)$. Then, we write (4.1) as

$$Ax = \begin{bmatrix} A_1^T \\ A_2^T \end{bmatrix} (x_1 + x_2) = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = b.$$

Notice that $A_1^T x_2 = 0$, and hence $A_1^T x_1 = b_1$. Since $x_1 \in \mathscr{R}(A_1)$, we have that x_1 is the minimum 2-norm solution to the underdetermined system

$$x_1 = \underset{y}{\operatorname{argmin}} \{\|y\|_2 : A_1^T y = b_1\}. \tag{4.2}$$

For small s, $\kappa(A_1)$ is small in general, and hence we can compute an accurate x_1. Moreover, if matrix $\bar{A}_2$ has the same size as A_2, and its range space is the orthogonal complement of the range space of A_1, then x_2 satisfies that

$$x_2 = \underset{y}{\operatorname{argmin}} \{\|y\|_2 : \bar{A}_2^T y = \bar{b}_2\}, \tag{4.3}$$

where $\bar{b}_2$ is computed according to the construction of $\bar{A}_2$. Noticing that $\kappa(\bar{A}_2)$ might be large, to obtain x_2, we again partition (4.3) and break it down into two underdetermined systems. Using the same strategy, we repeat the breakdown process until the matrix of the last underdetermined system is not ill-conditioned.

We discuss the two keystones for the breakdown process: the solution of (4.2) and the construction of $\bar{A}_2$. If matrices Q_1, R_1 are constructed by MGS from A_1, then the solution of (4.2) is given by $x_1 = Q_1(R_1)^{-T} b_1$. However, direct computing the formula is unstable due to the loss of orthogonality of Q_1. Björck et al proposed in [15] a backward stable process for solving underdetermined systems by MGS, and we shall include it right below. For the construction of $\bar{A}_2$, we shall apply $\text{proj}(A_2, Q_1)$. Thus, by applying BGSP to A^T, we can solve the underdetermined systems using the matrices generated by the factorization process.

We review the stable process for solving underdetermined systems by MGS pro-

posed in [15]. Let $A_1 \in \mathbb{R}^{n\times s}$ be of full rank, $n \geq s$, $b_1 \in \mathbb{R}^s$. Let $Q_1 := [q_i : i \in \mathbb{N}_s]$, R_1 be the matrices constructed by MGS from A_1. The following process, called BjMGS, is to solve the underdetermined system (4.2). We first solve $R_1^T v = b_1$ for $v = [v_i : i \in \mathbb{N}_s]$ by forward substitution. Let $y^s := 0$. For $i = s, s-1, \ldots, 1$, we compute

$$y^{i-1} := y^i - (q_i^T y^i - v_i) q_i.$$

Then the solution of (4.2) is given by $x_1 := y^0$.

We present the BGSP solver (BGSPSol) for (4.1) below. Assuming that matrix A^T satisfies Hypothesis (A), with the same notation as BGSP applied to A^T, we proceed the following.

For each $i \in \mathbb{N}_\gamma$, let $b_i^1 \in \mathbb{R}^k$ be the subvector of b such that $b = [(b_i^1)^T : i \in \mathbb{N}_\gamma]^T$.

For each $\ell \in \mathbb{N}_{L-1}$, do the following.

For $i \in \mathbb{N}_{\gamma_\ell/4}$,

let $f_i^\ell := [(b_{4i-2}^\ell)^T, (b_{4i-1}^\ell)^T]^T$, $h_{2i-1}^\ell := b_{4i-3}^\ell$, and $h_{2i}^\ell := b_{4i}^\ell$,

construct matrices Q_i^ℓ, R_i^ℓ by MGS from $A_{s_i}^\ell$,

compute x_i^ℓ by BjMGS from Q_i^ℓ, R_i^ℓ, and f_i^ℓ.

For $j \in \mathbb{N}_{\gamma_\ell/2}$, let $r := \lceil j/2 \rceil$, and

construct matrices $G_j^{\ell+1}$, C_j^ℓ by $\mathrm{proj}(A_{t_j}^\ell, Q_r^\ell)$,

solve for $b_j^{\ell+1}$ the following lower triangular system by forward substitution,

$$\begin{bmatrix} R_r^\ell & C_j^\ell \\ 0 & I \end{bmatrix}^T \begin{bmatrix} v \\ b_j^{\ell+1} \end{bmatrix} = \begin{bmatrix} f_r^\ell \\ h_j^\ell \end{bmatrix}. \tag{4.4}$$

Let $A^{\ell+1} := [G_j^{\ell+1} : j \in \mathbb{N}_{\gamma_\ell/2}]$.

Construct matrices Q^L, R^L by MGS from A^L.

Compute x^L by BjMGS from Q^L, R^L, and $b^L := [(b_1^L)^T, (b_2^L)^T]^T$.

For each $\ell \in \mathbb{N}_{L-1}$, let $x^\ell := \sum_{i\in\mathbb{N}_{\gamma_\ell/4}} x_i^\ell$. Output $x := \sum_{\ell\in\mathbb{N}_L} x^\ell$.

We verify in the theorem below that the output x satisfies (4.1).

Theorem 4.1.1. *Let $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^{n\times 1}$. If A^T satisfies Hypothesis (A), then the vector x constructed by BGSPSol from A, b satisfies* (4.1).

Proof. If $L = 1$, according to BGSPSol and BjMGS, we have that $x = Q^1(R^1)^{-T}b$. Notice that $A^1 = A^T$, and Q^1, R^1 are constructed by MGS from A^1. Hence,

$$Ax = (R^1)^T(Q^1)^TQ^1(R^1)^{-T}b = b.$$

We now consider $L > 1$. By BjMGS, it can be verified for $\ell \in \mathbb{N}_{L-1}$, $i \in \mathbb{N}_{\gamma_\ell/4}$ that $x_i^\ell = Q_i^\ell(R_i^\ell)^{-1}f_i^\ell$. Let $f^\ell := [(f_i^\ell)^T : i \in \mathbb{N}_{\gamma_\ell/4}]^T$. With the notation of Q^ℓ and R^ℓ in BGSP, we have that $x^\ell = Q^\ell(R^\ell)^{-1}f^\ell$. Let $z^\ell := (R^\ell)^{-1}f^\ell$, and $z^L := (R^L)^{-1}b^L$. It follows from (2.12) and the orthogonality of Q that $Q^Tx = [(z^\ell)^T : \ell \in \mathbb{N}_L]^T$. Thus, with formula (2.15) and $A^T = QS$, to show that $Ax = b$, it suffices to prove that

$$\sum_{\ell\in\mathbb{N}_L} (S^\ell)^Tz^\ell = b. \tag{4.5}$$

This can be done by proving the following equation by induction on $\ell \in \mathbb{N}_{L-1}$.

$$\sum_{j\in\mathbb{N}_\ell} (S^j)^Tz^j = b - (F^\ell)^T \begin{bmatrix} 0 \\ b^{\ell+1} \end{bmatrix}, \tag{4.6}$$

where F^ℓ is defined by (2.13). Let $h^\ell := [(h_i^\ell)^T : i \in \mathbb{N}_{\gamma_\ell/2}]^T$. Then by formula (4.4), the construction of R^ℓ, C^ℓ, and the definition of E^ℓ, we obtain that

$$[R^\ell, C^\ell]^Tz^\ell = \begin{bmatrix} f^\ell \\ h^\ell \end{bmatrix} - \begin{bmatrix} 0 \\ b^{\ell+1} \end{bmatrix} = E^\ell b^\ell - \begin{bmatrix} 0 \\ b^{\ell+1} \end{bmatrix}. \tag{4.7}$$

Employing this equation and the definitions (2.13), (2.14) yields (4.6) for $\ell = 1$. Assume that (4.6) is true for $\ell \in \mathbb{N}_{L-2}$. Again, employing equation (4.7) for $(\ell+1)$ and the definitions (2.13), (2.14) yields that

$$(S^{\ell+1})^T z^{\ell+1} = (F^{\ell})^T \begin{bmatrix} 0 \\ b^{\ell+1} \end{bmatrix} - (F^{\ell+1})^T \begin{bmatrix} 0 \\ b^{\ell+2} \end{bmatrix}.$$

This together with the induction hypothesis yields formula (4.6) for $(\ell+1)$. Hence by the induction principle, we have (4.6). Employing (4.6) for $\ell = L-1$ and the fact that

$$(S^L)^T z^L = (F^{L-1})^T \begin{bmatrix} 0 \\ (R^L)^T \end{bmatrix} (R^L)^{-T} b^L = (F^{L-1})^T \begin{bmatrix} 0 \\ b^L \end{bmatrix},$$

we obtain formula (4.5). Therefore, formula (4.1) holds. □

4.2 Error Analysis of the Sequential Solver

In this section, we conduct an error analysis of BGSPSol. Especially, we derive an upper bound for the relative error $\|\hat{x}^\ell - x^\ell\|_2/\|x^\ell\|_2$ of the computed solution $\hat{x}^\ell$. According to the definition of x^ℓ, it suffices to bound the error of each $\hat{x}_i^\ell$.

We derive the error bound for $\|\hat{x}_i^\ell - x_i^\ell\|_2/\|x_i^\ell\|_2$. This can be done by applying the stability result of BjMGS and the perturbation theory of underdetetermined systems. To this end, we review the two lemmas below.

The following lemma shows that the computed solution of an underdetetermined system obtained by BjMGS is an exact solution to a relevant *nearby* system [15].

Lemma 4.2.1. *Let $A_1 \in \mathbb{R}^{n\times s}$ $(n \geq s)$ be of full rank. If $\hat{x}_1$ is the computed solution to (4.2) by BjMGS, then for sufficiently small $\mathbf{u}\kappa(A_1)$, there exists a constant $c := c(n,s)$*

such that

$$\hat{x}_1 = \underset{y}{\operatorname{argmin}} \{\|y\|_2 : (A_1 + \Delta A_1)^T y = b_1\}, \qquad \|\Delta A_1\|_2 \leq c\mathbf{u}\|A_1\|_2.$$

The next lemma concerns the perturbation theory of underdetetermined systems [53].

Lemma 4.2.2. *Let $A_1 \in \mathbb{R}^{n\times s}$ $(n \geq s)$ be of full rank and $0 \neq b_1 \in \mathbb{R}^s$. Suppose that $\|\Delta A_1\|_2 \leq \epsilon\|A_1\|_2$, $\|\Delta b_1\|_2 \leq \epsilon\|b_1\|_2$, and $\epsilon\kappa(A_1) < 1$. If x_1 satisfies* (4.2) *and*

$$x_\epsilon = \underset{y}{\operatorname{argmin}} \{\|y\|_2 : (A_1 + \Delta A_1)^T y = b_1 + \Delta b_1\},$$

then there exists a constant $c := c(n, s)$ such that

$$\frac{\|x_\epsilon - x_1\|_2}{\|x_1\|_2} \leq c\epsilon\kappa(A_1).$$

With the previous lemmas, we can estimate the error of the computed solution $\hat{x}_i^\ell$ obtained by BjMGS from the solution $\tilde{x}_i^\ell$ to the system

$$\tilde{x}_i^\ell = \underset{y}{\operatorname{argmin}} \{\|y\|_2 : (\hat{A}_{s_i}^\ell)^T y = \hat{f}_i^\ell\}.$$

Notice that, due to the round-off errors, $\tilde{x}_i^\ell$ is not the exact solution to

$$x_i^\ell = \underset{y}{\operatorname{argmin}} \{\|y\|_2 : (A_{s_i}^\ell)^T y = f_i^\ell\}.$$

To estimate $\|\tilde{x}_i^\ell - x_i^\ell\|_2$, we need to bound $\|\hat{A}_{s_i}^\ell - A_{s_i}^\ell\|_2$ and $\|\hat{f}_i^\ell - f_i^\ell\|_2$. For the former, we include the following lemma [13].

Lemma 4.2.3. *Assume* (2.59). *Let $Q \in \mathbb{R}^{n\times s}$ be the column orthonormal matrix constructed by MGS, $v \in \mathbb{R}^n$. If $\hat{v}'$ is the computed vector of v' constructed by* $\operatorname{proj}(v, Q)$, *then*

$$\|\hat{v}' - v'\|_2 \leq \frac{13}{4}(s-1)\mathbf{u}\|v\|_2.$$

We now present an error analysis for $\hat{x}_i^\ell$.

Proposition 4.2.4. *Let BGSPSol be applied to* (4.1)*, where* $A := [a_{ij} : i, j \in \mathbb{N}_n]$ *and* A^T *satisfies Hypothesis (A). Let* $\alpha := \max_{i,j}\{|a_{ij}|\}$*,* $\ell \in \mathbb{N}_{L-1}$*, and* $i \in \mathbb{N}_{\gamma_\ell/4}$*. Assume Hypothesis (F),* (2.59)*, and that there exists a number* $\xi^\ell \geq 1$ *such that*

$$\|\hat{b}_j^\ell - b_j^\ell\|_2 \leq \mathbf{u}\xi^\ell \|b_j^\ell\|_2, \quad \text{for } j \in \mathbb{N}_{\gamma_\ell}. \tag{4.8}$$

If $f_i^\ell \neq 0$*, and*

$$\zeta_i^\ell := \max\left\{\alpha/\|A_{s_i}^\ell\|_2,\ \xi^\ell\right\}\ \kappa(A_{s_i}^\ell),$$

then for sufficiently small $\mathbf{u}\zeta_i^\ell$*, there exists a constant* $c := c(n, \ell, k)$ *such that*

$$\frac{\|\hat{x}_i^\ell - x_i^\ell\|_2}{\|x_i^\ell\|_2} \leq c\mathbf{u}\zeta_i^\ell. \tag{4.9}$$

Proof. Since $\mathbf{u}\zeta_i^\ell$ is sufficiently small and $\zeta_i^\ell \geq \kappa(A_{s_i}^\ell)$, by Lemma 4.2.1, there exists a constant $c_1 := c_1(n, k)$ such that

$$\hat{x}_i^\ell = \operatorname*{argmin}_y \{\|y\|_2 : (\hat{A}_{s_i}^\ell + \Delta\hat{A}_{s_i}^\ell)^T y = \hat{f}_i^\ell\}, \qquad \|\Delta\hat{A}_{s_i}^\ell\|_2 \leq c_1\mathbf{u}\|\hat{A}_{s_i}^\ell\|_2. \tag{4.10}$$

Since A has bandwidth $k/2$, by the definition of α, we have for each column vector a of A^T that $\|a\|_2 \leq \sqrt{k+1}\alpha$. By Proposition 2.5.5 and Lemma 4.2.3, we have for a column vector a^ℓ of A^ℓ that

$$\|\hat{a}^\ell - a^\ell\|_2 \leq \frac{13}{4}(2\ell k - 2k - 1)\sqrt{k+1}\mathbf{u}\alpha. \tag{4.11}$$

Hence, we obtain a constant $c_2 := c_2(\ell, k)$ that

$$\|\hat{A}_{s_i}^\ell - A_{s_i}^\ell\|_2 \leq \|\hat{A}_{s_i}^\ell - A_{s_i}^\ell\|_F \leq c_2\mathbf{u}\alpha. \tag{4.12}$$

According to $\mathrm{proj}(\cdot,\cdot)$, it can be verified that

$$\|a^\ell\|_2 \leq \max\{\|a\|_2 : a \text{ is a column vector of } A^T\} \leq \sqrt{k+1}\alpha. \tag{4.13}$$

Thus, we have that

$$\|A_{s_i}^{\ell}\|_2 \leq \|A_{s_i}^{\ell}\|_F \leq \sqrt{(2k)(k+1)}\alpha. \tag{4.14}$$

Let $\Delta A_{s_i}^{\ell} := \hat{A}_{s_i}^{\ell} - A_{s_i}^{\ell} + \Delta\hat{A}_{s_i}^{\ell}$. By formulas (4.10), (4.12), and (4.14), we obtain for a constant $c_3 := c_3(n, \ell, k)$ that

$$\|\Delta A_{s_i}^{\ell}\|_2 \leq c_2\mathbf{u}\alpha + c_1\mathbf{u}\left(\|\hat{A}_{s_i}^{\ell} - A_{s_i}^{\ell}\|_2 + \|A_{s_i}^{\ell}\|_2\right) \leq c_3\mathbf{u}\alpha.$$

Notice that $\|\hat{f}_i^{\ell} - f_i^{\ell}\|_2 \leq \mathbf{u}\xi^{\ell}\|f_i^{\ell}\|_2$ by formula (4.8). Thus, we have that,

$$\hat{x}_i^{\ell} = \underset{y}{\operatorname{argmin}}\{\|y\|_2 : (A_{s_i}^{\ell} + \Delta A_{s_i}^{\ell})^T y = \hat{f}_i^{\ell}\}, \quad \|\Delta A_{s_i}^{\ell}\|_2 \leq c_3\mathbf{u}\alpha, \quad \|\hat{f}_i^{\ell} - f_i^{\ell}\|_2 \leq \mathbf{u}\xi^{\ell}\|f_i^{\ell}\|_2.$$

Since $\mathbf{u}\zeta_i^{\ell}$ is sufficiently small, we have that $c_3\mathbf{u}\zeta_i^{\ell} < 1$. By Lemma 4.2.2, we obtain formula (4.9). □

We next derive recurrence relations for ξ^{ℓ} and $\|\hat{b}_j^{\ell} - b_j^{\ell}\|_2$ in terms of ℓ. This can be done by an error analysis for the lower triangular system (4.4). To this end, we include two relevant results in the following.

The lemma below confirms that solving lower triangular systems by forward subsititution is backward stable [43].

Lemma 4.2.5. *Let $R' \in \mathbb{R}^{s\times s}$ be a upper triangular matrix of full rank. If $\hat{v}'$ is the computed solution to $R'^T v' = b'$ by forward subsititution, then there exists a constant $c := c(s)$ such that*

$$(R' + \Delta R')^T\hat{v}' = b', \qquad \|\Delta R'\|_2 \leq c\mathbf{u}\|R'\|_2.$$

The next lemma concerns the perturbation theory of linear systems [43].

Lemma 4.2.6. *Let $A' \in \mathbb{R}^{s\times s}$ and $0 \neq b' \in \mathbb{R}^s$. Suppose that $\|\Delta A'\|_2 \leq \epsilon\|A'\|_2$, $\|\Delta b'\|_2 \leq \epsilon\|b'\|_2$, and $\epsilon\kappa(A') < 1$. If x' and x'_{ϵ} respectively satisfy that $A'x' = b'$ and*

$(A' + \Delta A')x'_\epsilon = b' + \Delta b'$, *then there exists a constant* c *such that*

$$\frac{\|x'_\epsilon - x'\|_2}{\|x'\|_2} \leq c\epsilon\kappa(A').$$

With the two lemmas, if we let

$$R' := \begin{bmatrix} R_r^\ell & C_j^\ell \\ 0 & I \end{bmatrix}, \quad v' := \begin{bmatrix} v \\ b_j^{\ell+1} \end{bmatrix}, \quad b' := \begin{bmatrix} f_r^\ell \\ h_j^\ell \end{bmatrix}, \tag{4.15}$$

then we can estimate the error of the computed solution $\hat{v}'$ from the solution $\tilde{v}'$ to the system $\hat{R}'^T\tilde{v}' = \hat{b}'$. Again, we need the error bound $\|\hat{R}' - R'\|_2$ to estimate $\|\tilde{v}' - v'\|_2$. Noticing that R_r^ℓ and C_j^ℓ are submatrices of the upper triangular matrix constructed by MGS from the matrix $[A_{s_r}^\ell,\ A_{t_j}^\ell]$, we may derive the bound for $\|\hat{R}' - R'\|_2$ from the error analysis of MGS. To this end, we need the two lemmas below.

The following lemma estimates the error of the computed upper triangular matrix constructed by MGS [13].

Lemma 4.2.7. *Let* $A_0 \in \mathbb{R}^{n\times s}$ $(n \geq s)$ *be of full rank. If* $\hat{R}_0$ *is the computed matrix of the upper triangular matrix* R_0 *constructed by MGS, then there exist a matrix* ΔA_0*, a column orthonormal* $\bar{Q}_0$ *and a constant* $c := c(n, s)$ *such that*

$$A_0 + \Delta A_0 = \bar{Q}_0\hat{R}_0, \qquad \|\Delta A_0\|_2 \leq c\mathbf{u}\|A_0\|_2.$$

We include below the perturbation theory of the QR Factorization [105].

Lemma 4.2.8. *Let matrices* $A_0 \in \mathbb{R}^{n\times s}$ $(n \geq s)$ *be of full rank. Suppose that* $\|\Delta A_0\|_2 \leq \epsilon\|A_0\|_2$*, and* $\epsilon\kappa(A_0) < 1$*. If* $A_0 = Q_0R_0$ *and* $A_0 + \Delta A_0 = \bar{Q}_0(R_0 + \Delta R_0)$ *are respectively the QR factorization of* A_0 *and* $A_0 + \Delta A_0$*, then there exists a constant* $c := c(n, s)$ *such that*

$$\frac{\|\Delta R_0\|_2}{\|R_0\|_2} \leq c\epsilon\kappa(A_0).$$

We now present the proposition below for the expression of ξ^ℓ and the upper bound for $\|\hat{b}_j^\ell - b_j^\ell\|_2$.

Proposition 4.2.9. *Let BGSPSol be applied to* (4.1)*, where* $A := [a_{ij} : i, j \in \mathbb{N}_n]$ *and* A^T *satisfies Hypothesis (A). Let* $\alpha := \max_{i,j}\{|a_{ij}|\}$*, and* $\ell \in \mathbb{N}_{L-1}$*. Assume Hypothesis (F),* (2.59)*, and that there exists a number* $\xi^\ell \geq 1$ *such that* (4.8) *holds. For each* $j \in \mathbb{N}_{\gamma_\ell/2}$*, let* $r := \lceil j/2 \rceil$*,* $B_j := [A_{s_r}^\ell,\ A_{t_j}^\ell]$*, and let*

$$\xi_j^\ell := \max\left\{1 + \alpha\|B_j^+\|_2,\ \xi^\ell\right\}\ \kappa(B_j).$$

Then for sufficiently small $\mathbf{u}\xi_j^\ell$*, there exists a constant* $\tilde{c}_j := \tilde{c}_j(n, \ell, k)$ *such that*

$$\|\hat{b}_j^{\ell+1} - b_j^{\ell+1}\|_2 \leq \tilde{c}_j \mathbf{u}\xi_j^\ell(\|x_r^\ell\|_2 + \|b_j^{\ell+1}\|_2). \tag{4.16}$$

If

$$\xi^{\ell+1} := \max_j\left\{\tilde{c}_j\xi_j^\ell(\|x_r^\ell\|_2/\|b_j^{\ell+1}\|_2 + 1),\ \xi_j^\ell\right\}, \tag{4.17}$$

then

$$\|\hat{b}_j^{\ell+1} - b_j^{\ell+1}\|_2 \leq \mathbf{u}\xi^{\ell+1}\|b_j^{\ell+1}\|_2. \tag{4.18}$$

Proof. Formula (4.18) directly follows from (4.16) and the definition (4.17). To show (4.16), we consider the lower triangular system (4.4) for a fixed j. With the definition (4.15), by Lemma 4.2.5, there exists a constant $c_1 := c_1(k)$ such that

$$(\hat{R}' + \Delta\hat{R}')^T\hat{v}' = \hat{b}', \quad \|\Delta\hat{R}'\|_2 \leq c_1\mathbf{u}\|\hat{R}'\|_2. \tag{4.19}$$

Let R_j denote the upper triangular matrix constructed by MGS from B_j. Then $\|\hat{R}' - R'\|_F \leq \|\hat{R}_j - R_j\|_F$. By Lemma 4.2.7, there exist a matrix $\Delta\hat{B}_j$, a column orthonormal $\bar{Q}$, and a constant $c_2 := c_2(n, k)$ such that

$$\hat{B}_j + \Delta\hat{B}_j = \bar{Q}\hat{R}_j, \quad \|\Delta\hat{B}_j\|_2 \leq c_2\mathbf{u}\|\hat{B}_j\|_2.$$

By formula (4.11), we obtain a constant $c_3 := c_3(\ell, k)$ that

$$\|\hat{B}_j - B_j\|_2 \leq c_3 \mathbf{u}\alpha.$$

It follows from the two formulas above and (4.13) that there exists a constant $c_4 := c_4(n, \ell, k)$ such that

$$\|\hat{B}_j - B_j + \Delta\hat{B}_j\|_2 \leq c_3\mathbf{u}\alpha + c_2\mathbf{u}(\|\hat{B}_j - B_j\|_2 + \|B_j\|_2) \leq c_4\mathbf{u}\alpha.$$

Since $\mathbf{u}\xi_j^\ell$ is sufficiently small and $\xi_j^\ell \geq \alpha\|B_j^+\|_2$, we have that $c_4\mathbf{u}\alpha\|B_j^+\|_2 < 1$. By Lemma 4.2.8, we have for a constant $c_5 := c_5(n, \ell, k)$ that

$$\|\hat{R}_j - R_j\|_2 \leq c_5\mathbf{u}\alpha\|B_j^+\|_2.$$

Thus,

$$\|\hat{R}' - R'\|_2 \leq \|\hat{R}_j - R_j\|_F \leq \sqrt{3k}\|\hat{R}_j - R_j\|_2 \leq c_5\sqrt{3k}\mathbf{u}\alpha\|B_j^+\|_2.$$

Let $\Delta R' := \hat{R}' - R' + \Delta\hat{R}'$. Notice that $\|R'\|_2 \geq (1/\sqrt{3k})\|R'\|_F \geq 1/\sqrt{3}$. By the two inequalities above and (4.19), we obtain for a constant $c_6 := c_6(n, \ell, k)$ that

$$\|\Delta R'\|_2 \leq \mathbf{u}\|R'\|_2\left(2c_1 + 3c_5\alpha\sqrt{k}\|B_j^+\|_2\right) \leq c_6\mathbf{u}(1 + \alpha\|B_j^+\|_2)\|R'\|_2.$$

Together with formula (4.8), we have that

$$(R' + \Delta R')^T\hat{v}' = b' + \Delta b', \quad \|\Delta R'\|_2 \leq c_6\mathbf{u}(1 + \alpha\|B_j^+\|_2)\|R'\|_2, \quad \|\Delta b'\|_2 \leq \mathbf{u}\xi^\ell\|b'\|_2.$$

Since $c_6\mathbf{u}\xi_j^\ell < 1$ for sufficiently small $\mathbf{u}\xi_j^\ell$, by Lemma 4.2.6, there exists a constant $\tilde{c}_j := \tilde{c}_j(n, \ell, k)$ such that

$$\frac{\|\hat{v}' - v'\|_2}{\|v'\|_2} \leq \tilde{c}_j\mathbf{u}\xi_j^\ell. \tag{4.20}$$

Notice that $x_r^\ell = Q_r^\ell(R_r^\ell)^{-T}f_r^\ell$ by BjMGS, and hence $\|v'\|_2 \leq \|x_r^\ell\|_2 + \|b_j^{\ell+1}\|_2$. This together with (4.20) yields the desired formula (4.16). □

We finally present the theorem below for the error bound for the solution to (4.1) by BGSPSol. Let $\xi^1 := 1$, and for each $\ell \in \mathbb{N}_{L-1}$, let $\xi^{\ell+1}$ be defined by the recurrence relation (4.17). We also define

$$\zeta^\ell := \max_{i \in \mathbb{N}_{\gamma_\ell/4}} \zeta_i^\ell, \quad \text{and} \quad \zeta^L := \max\left\{\alpha/\|A^L\|_2,\ \xi^L\right\} \kappa(A^L).$$

The next theorem indicates that ζ^ℓ is a condition number for obtaining x^ℓ for $\ell \in \mathbb{N}_L$.

Theorem 4.2.10. *Let BGSPSol be applied to* (4.1)*, where A^T satisfies Hypothesis (A). Assume Hypothesis (F),* (2.59)*, and for each $\ell \in \mathbb{N}_{L-1}$, $i \in \mathbb{N}_{\gamma_\ell/4}$ that $f_i^\ell \neq 0$. For each $\ell \in \mathbb{N}_L$, if $\mathbf{u}\zeta^\ell$ is sufficiently small, then there exists a constant $c := c(n,k)$ such that*

$$\frac{\|\hat{x}^\ell - x^\ell\|_2}{\|x^\ell\|_2} \leq c\mathbf{u}\zeta^\ell. \tag{4.21}$$

Proof. The proof follows from the previous two propositions. Notice that for each $j \in \mathbb{N}_{\gamma_1/2}$, $\|\hat{b}_j^1 - b_j^1\|_2 \leq \xi^1\mathbf{u}\|b_j^1\|_2$ by the floating point number representation. For $\ell = 1, 2, \ldots, L-1$, since $\mathbf{u}\zeta^{\ell+1}$ is sufficiently small and $\zeta^{\ell+1} \geq \xi^{\ell+1} \geq \xi_j^\ell$ for each $j \in \mathbb{N}_{\gamma_\ell/2}$, repeatedly applying Proposition 4.2.9, we obtain formula (4.8), and that

$$\|\hat{b}^L - b^L\|_2 \leq \xi^L\mathbf{u}\|b^L\|_2. \tag{4.22}$$

Since $\mathbf{u}\zeta^\ell$ is sufficiently small and $\zeta^\ell \geq \zeta_i^\ell$ for each $i \in \mathbb{N}_{\gamma_\ell/4}$, by Proposition 4.2.4, there exists a constant $c_i^\ell := c_i^\ell(n,\ell,k)$ such that

$$\|\hat{x}_i^\ell - x_i^\ell\|_2 \leq c_i^\ell\mathbf{u}\zeta_i^\ell\|x_i^\ell\|_2. \tag{4.23}$$

Recall that $x_i^\ell \in \mathscr{R}(Q_i^\ell)$, $(Q_i^\ell)^T Q_j^\ell = 0$ for $j \neq i$, and thus $(x_i^\ell)^T x_j^\ell = 0$. Let $c := \max_{\ell,i}\{c_i^\ell\}$. With the definition of ζ^ℓ, summing up (4.23) for i yields formula (4.21).

With (4.22), we can also obtain for a constant $c^L := c^L(n, L, k)$ that

$$\|\hat{x}^L - x^L\|_2 \leq c^L \mathbf{u} \zeta^L \|x^L\|_2,$$

by a similar proof of Proposition 4.2.4, replacing ξ^ℓ, f_i^ℓ, ζ_i^ℓ, $A_{s_i}^\ell$, x_i^ℓ with ξ^L, b^L, ζ^L, A^L, and x^L respectively. Therefore, we have formula (4.21) for each $\ell \in \mathbb{N}_L$. □

We comment on the results for the solver. By formula (4.21), given $\ell \in \mathbb{N}_L$, ζ^ℓ can be regarded as the condition number for obtaining x^ℓ. Notice that ζ^ℓ is computed from the condition numbers of some matrices of column size $O(k)$. As a comparison, $\kappa(A)$ is a condition number for solving (4.1). In general, for small ℓ, ζ^ℓ is rather small and as a result, $\hat{x}^\ell$ is closed to the exact value. This could help us overcome the issue of ill-conditioning, and we shall verify it through experiments in the next chapter.

In implementation of BGSPSol, the storage for the matrices Q_i^ℓ, R_i^ℓ, C_j^ℓ, and even the corresponding subvectors can be instantly released after applied to compute x_i^ℓ. We call a solver that can compute the final solution without storing the matrices it generates over the entire process the *memory-less* solver. The memory-less feature is extremely important to deal with large-scale systems. We shall study further in the next section.

4.3 A Parallel Solver and Extra Large-scale Linear Systems

In this section, we apply ParBGSP to solve extra large-scale ill-conditioned banded systems. Inspired by BGSPSol, we break down a linear system into small-scale un-

derdetermined systems, to which minimum 2-norm solutions can be computed by the matrices generated in a superstep of ParBGSP. We also consider the memory issue in solving extra large-scale linear systems. Using the memory-less feature of BGSPSol, we treat the issue by a partial load strategy to be presented later.

We consider solving the linear system (4.1) where the matrix A is large-scale, ill-conditioned, and banded. Using a similar design as ParBGSP, we divide the computations in BGSPSol into parts to be executed concurrently. With the definition (3.1) of Γ_p, we partition (4.1) as

$$Ax = \begin{bmatrix} (A_{\Gamma_1})^T \\ (A_{\Gamma_2})^T \\ \vdots \\ (A_{\Gamma_P})^T \end{bmatrix} x = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_P \end{bmatrix} = b,$$

where $b_p \in \mathbb{R}^{n/P}$ for each $p \in \mathbb{N}_P$. Then in the first superstep of the parallel algorithm, for each p, we shall apply the following algorithm called BGSPSol_1 to A_{Γ_p} and b_p.

We next consider the memory issue when A is extra large-scale. Notice that $\#(A_{\Gamma_p}) \approx (n/P)(k+1)$. When n/P is very large, it is not possible for a processor to storage $\#(A_{\Gamma_p})$ pieces of data in memory. To overcome the memory issue, we apply a partial load strategy using the memory-less property of the solving process: Instead of loading entirely the matrix A_{Γ_p} into memory, we partition A_{Γ_p} into its submatrices, each of which consists of some consecutive column blocks of it. We then sequentially load each of these submatrices, solve the corresponding system by BGSPSol_1, sum up the solutions, and release the matrices generated. To this end, we denote by $\nu := \nu(p)$ the partial load parameter, depending on the memory volume of the processor p. We assume that ν is a power of 2, and $\gamma \geq 4\nu P$. For each $i \in \mathbb{N}_\nu$, we define the subvector

Algorithm 4.1 A sub-algorithm for BGSPSol (BGSPSol$_1$)

Input: A matrix $A \in \mathbb{R}^{n\times m}$ that satisfies Hypothesis (A2), a vector $b \in \mathbb{R}^m$.
Output: A vector $\bar{x}$, a matrix $\bar{A}$, a vector $\bar{b}$.

For each $i \in \mathbb{N}_\gamma$, let $b_i^1 \in \mathbb{R}^{\mu k}$ be the subvector of b such that $b = [(b_i^1)^T : i \in \mathbb{N}_\gamma]^T$.
for $\ell \in \mathbb{N}_{L-1}$ **do**
  for $i \in \mathbb{N}_{\gamma_\ell/4}$ **do**
    let $f_i^\ell := [(b_{4i-2}^\ell)^T, (b_{4i-1}^\ell)^T]^T$, $h_{2i-1}^\ell := b_{4i-3}^\ell$, and $h_{2i}^\ell := b_{4i}^\ell$,
    generate matrices Q_i^ℓ, R_i^ℓ by MGS from $A_{s_i}^\ell$,
    solve $(R_i^\ell)^T z_i^\ell = f_i^\ell$ for z_i^ℓ by forward substitution,
    compute x_i^ℓ by BjMGS from Q_i^ℓ, I, and z_i^ℓ,
    for $j = 2i-1, 2i$ **do**
      generate matrices $G_j^{\ell+1}$, C_j^ℓ by proj$(A_{t_j}^\ell, Q_i^\ell)$,
      solve the following lower triangular system by forward substitution,

$$\begin{bmatrix} I & C_j^\ell \\ 0 & I \end{bmatrix}^T \begin{bmatrix} v \\ b_j^{\ell+1} \end{bmatrix} = \begin{bmatrix} z_i^\ell \\ h_j^\ell \end{bmatrix},$$

      release the memory storage for $A_{t_j}^\ell$, C_j^ℓ, h_j^ℓ,
    endfor
    release the memory storage for $A_{s_i}^\ell$, Q_i^ℓ, R_i^ℓ, z_i^ℓ, f_i^ℓ,
  endfor
  let $A^{\ell+1} := [G_j^{\ell+1} : j \in \mathbb{N}_{\gamma_\ell/2}]$.
endfor
return $\bar{x} := \sum_{\ell\in\mathbb{N}_{L-1}} \sum_{i\in\mathbb{N}_{\gamma_\ell/4}} x_i^\ell$, $\bar{A} := A^L$, $\bar{b} := [(b_1^L)^T, (b_2^L)^T]^T$.

Γ_{ip} of Γ_p by

$$\Gamma_{ip} := [(p\nu - \nu + i - 1)\gamma/(\nu P) + t : t \in \mathbb{N}_{\gamma/(\nu P)}]. \tag{4.24}$$

Then $A_{\Gamma_{ip}}$ consists of $\gamma/(\nu P)$ consecutive column blocks of A_{Γ_p}. Correspondingly, given the vector $b = [b_t : t \in \mathbb{N}_n]^T$ as in (4.1), we define the subvectors b_{ip} of b by

$$b_{ip} := [b_t : t = (p\nu - \nu + i - 1)n/(\nu P) + t', \text{ for } t' \in \mathbb{N}_{n/(\nu P)}]^T. \tag{4.25}$$

With the above discussion and a similar design as ParBGSP, we present the parallel solver called ParBGSPSol for linear systems below.

To close this chapter, we comment on ParBGSPSol. It can be verified by the equations in FormQS and a similar proof of Theorem 4.1.1 that ParBGSPSol generates a vector x that satisfy (4.1). For an extra large-scale linear system, the summation of

Algorithm 4.2 The parallel BGSP solver for linear systems (ParBGSPSol)

Require: The set $\mathbb{N}_P$ of the P processors with $P = 2^{J-1}$ for some $J \in \mathbb{N}$, a butterfly all-reduction tree with height $\log_2 P$, my processor $p \in \mathbb{N}_P$, a partial load parameter ν with $\nu = 2^\imath$ for some $\imath \in \mathbb{N}$.

Input: A matrix $A \in \mathbb{R}^{n\times n}$ whose transpose satisfies Hypothesis (A2) with $L > J+\imath$, a vector $b \in \mathbb{R}^n$.

Output: A vector x.

Let $A' = A^T$. Let Γ_{ip}, b_{ip} be defined by (4.24), (4.25), respectively.
Load $A'_{\Gamma_{1p}}$ and b_{1p} into memory.
Generate $\bar{x}_{1p}$, $\bar{A}_{1p}$, $\bar{b}_{1p}$ by $\mathrm{BGSPSol}_1$ from $A'_{\Gamma_{1p}}$ and b_{1p}.
for $i \in \mathbb{N}_\nu \backslash \{1\}$ **do**
    load $A'_{\Gamma_{ip}}$ and b_{ip} into memory,
    generate x_{ip}, A'_{ip}, b'_{ip} by $\mathrm{BGSPSol}_1$ from $A'_{\Gamma_{ip}}$ and b_{ip},
    generate $\bar{x}_{ip}$, $\bar{A}_{ip}$, $\bar{b}_{ip}$ by $\mathrm{BGSPSol}_1$ from $[\bar{A}_{i-1,p}, A'_{ip}]$ and $[\bar{b}^T_{i-1,p}, b'^T_{ip}]^T$,
    release the memory storage for $\bar{A}_{i-1,p}$, A'_{ip}, $\bar{b}_{i-1,p}$, b'_{ip}.
Let $x_{1p} := 0$, $\quad x_p^1 := \sum_{i\in\mathbb{N}_\nu} (x_{ip} + \bar{x}_{ip})$, $\quad A_p^1 := \bar{A}_{\nu p}$, $\quad b_p^1 := \bar{b}_{\nu p}$.
for $j \in \mathbb{N}_J \backslash \{1\}$ **do**
    compute the processor q by (3.7),
    send x_p^{j-1}, A_p^{j-1}, b_p^{j-1} to q, and receive x_q^{j-1}, A_q^{j-1}, b_q^{j-1} from q,
    let $r := \min\{p, q\}$, $s := \max\{p, q\}$,
    generate $\bar{x}_p^j$, A_p^j, b_p^j by $\mathrm{BGSPSol}_1$ from $[A_r^{j-1}, A_s^{j-1}]$ and $[(b_r^{j-1})^T, (b_s^{j-1})^T]^T$,
    release the memory storage for A_p^{j-1}, A_q^{j-1}, b_p^{j-1}, b_q^{j-1},
    let $x_p^j := x_p^{j-1} + x_q^{j-1} + \bar{x}_p^j$.
Generate matrices Q^{J+1}, R^{J+1} by MGS from A_p^J.
Generate x_p^{J+1} by BjMGS from Q^{J+1}, R^{J+1}, and b_p^J.
return $x := \sum_{j\in\mathbb{N}_{J+1}} x_p^j$.

x can be done by the computer accumulators (accumulating x_{ip}, $\bar{x}_{ip}$, x_p^j), and we may assign some processors to save the components of x that are not being accumulated in external storage, so that the required memory storage in each processor will be less than $O(n/P)$ words. We also mention that the algorithms $\mathrm{BGSPSol}_1$ and ParBGSPSol could be easily modified in order to solve linear systems with multiple right-hand sides as $AX = B$.

Chapter 5

Applications and Numerical Experiments

In this chapter, we present numerical experiments in order to illuminate the theoretical results derived in the previous chapters, and to show that the proposed solvers maintain better accuracy and scalability in solving ill-conditioned linear systems among the most commonly used direct solvers.

We compare BGSPSol with three direct solvers on the accuracy and running speed for solving ill-conditioned linear systems derived from practical applications. We see from the comparison that BGSPSol generates more accurate solutions than the others, and it runs much faster than the solver with the second best accuracy.

In the parallel experiments, we compare ParBGSPSol with the routine for solving banded systems in ScaLAPACK on the accuracy and the scalability. We see that ParBGSPSol maintains much smaller relative errors while runs faster than the routine. Also, ParBGSPSol achieves approximately linear speedups. Moreover, we see from

another experiment that ParBGSPSol is capable of solving extra large-scale linear systems.

We organize this chapter in four sections. In section 5.1, we present an experiment to illuminate the theoretical results for BGSP. Comparisons with two other methods for matrix orthogonalization will also be presented. In section 5.2, we compare BGSPSol with three direct solvers on the results for solving linear systems derived from various applications. Parallel experiments with comparisons between ParBGSPSol and a routine in ScaLAPACK will be presented in section 5.3. Finally we briefly summarize the results in section 5.4.

5.1 Experiments for the Factorization Process

In this section, we present a numerical example to illustrate the theoretical results established in chapter 2. We compare BGSP with the QR factorization via Householder and the QR factorization via MGS. From the comparisons, we conclude that when applied to a banded matrix, BGSP generates sparse factor matrices, while MGS or Householder generates a *dense* orthogonal matrix, though Householder may obtain a sparse orthogonal matrix in its implicit vector form [43]. Also, BGSP has similar stability properties as MGS.

The experiments presented in this section are programmed and run in MATLAB [79]. In most of figures presented below, the axes of the graphs are plotted in log scale. If x, y are the variables on the horizontal and vertical axes in such a graph, respectively, then a straight line with slope c would imply that y is $O(x^c)$.

We show the following results: For a banded matrix A, if QS is the block QS factorization of A generated by BGSP, then both Q and S are sparse, Q is column orthonormal, SE is upper triangular for E defined by (2.16), and the factorization process is stable. This is done in Example 1.

***Example* 1.** We consider the tridiagonal matrix $A := [a_{ij} : i, j \in \mathbb{N}_n]$ with $a_{ii} := 2$, for $i \in \mathbb{N}_n$, and $a_{i+1,i} := -1$, $a_{i,i+1} := -1$, for $i \in \mathbb{N}_{n-1}$. The numerical results are illustrated in Figures 5.1−5.4.

Figure 5.1 shows that for BGSP, $\#(Q) < 2kn \log_2(n/k)$ and $\#(S) < \frac{13}{4}kn$, consistent with the results in Theorems 2.3.3 and 2.3.4, respectively. For MGS, $\#(Q) \approx n^2/2$, which means that Q is dense. For Householder, $\#(Q) = O(n)$ in its implicit vector

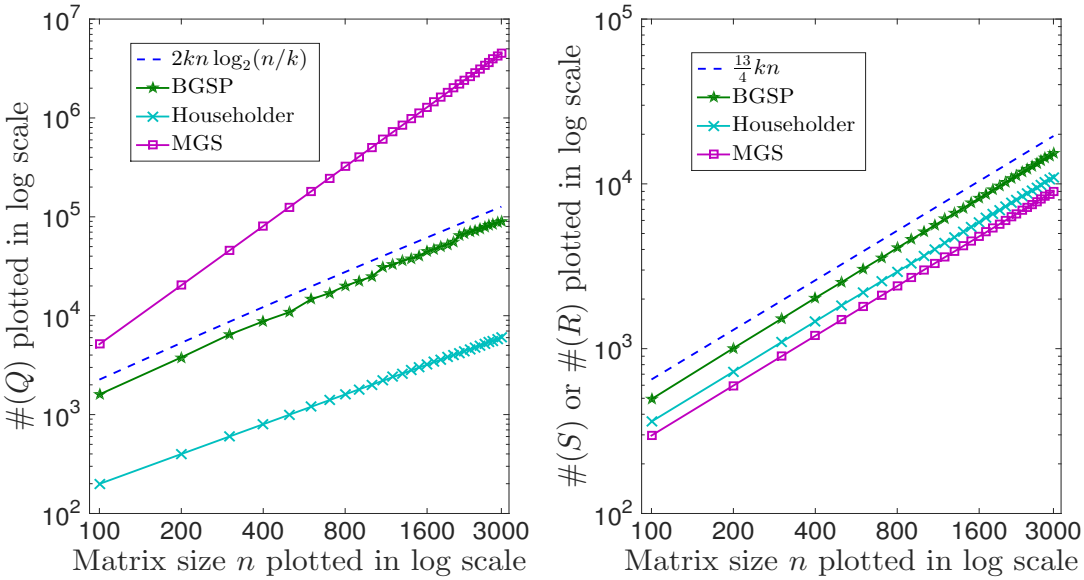


FIGURE 5.1: *Comparison of the numbers of nonzero entries of factor matrices.*

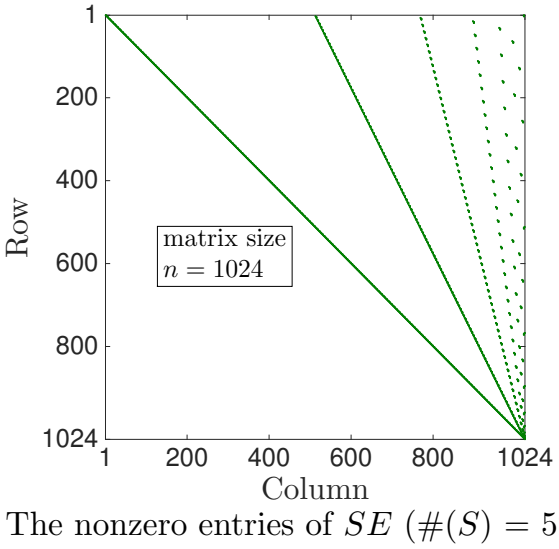


FIGURE 5.2: *The sparse structure of the matrix SE generated by BGSP. A dot in the graph is a nonzero entry of SE located in the specific row and column. The blank spaces are zero entries.*

form, however, $\#(Q) \approx n^2/2$ in its explicit form.

Figure 5.2 displays the sparse structure and the upper triangularity of SE generated by BGSP from A when $n = 1024$.

Figure 5.3 shows that for BGSP, $\|A - \hat{Q}\hat{S}\|_2 = O(\mathbf{u})$, where the machine unit error

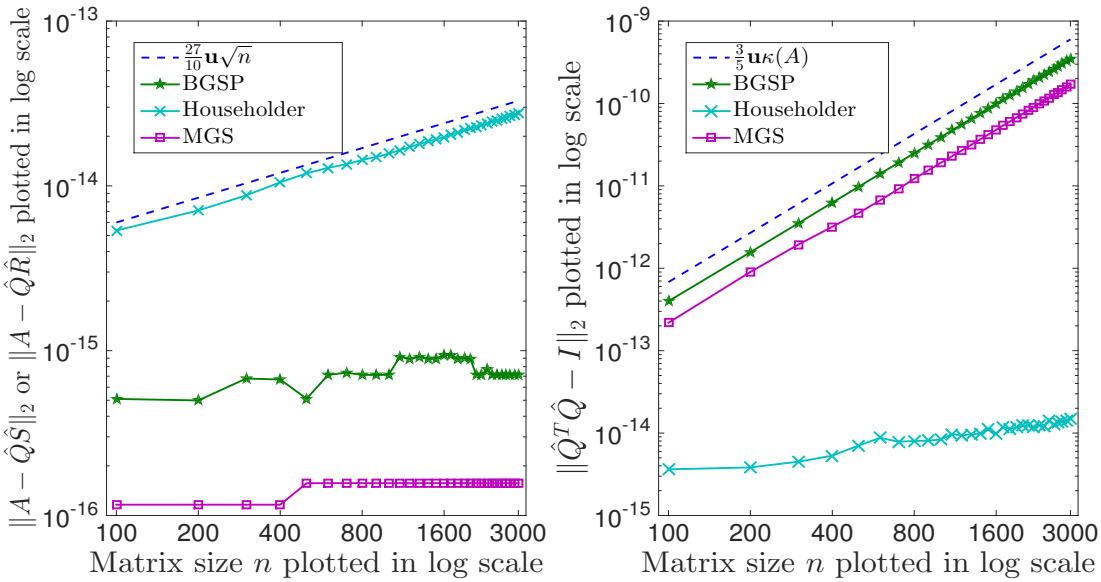


FIGURE 5.3: *Comparison of the accuracy of the three methods for matrix factorization.*

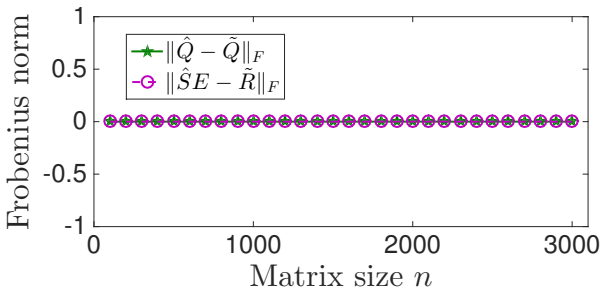


FIGURE 5.4: *BGSP applied to A versus MGS applied to AE.*

$\mathbf{u}$ is 2.22×10^{-16} in MATLAB, confirming that the block QS factorization via BGSP is stable. It also shows that $\|\hat{Q}^T\hat{Q} - I\|_2 = O(\mathbf{u}\kappa(A))$, as described in Theorem 2.5.9. MGS achieves similar results as BGSP. For Householder, $\|A - \hat{Q}\hat{R}\|_2 = O(\mathbf{u}\sqrt{n})$, and $\|\hat{Q}^T\hat{Q} - I\|_2 = O(\mathbf{u})$.

Figure 5.4 illuminates the results in Proposition 2.5.5. It shows that $\|\hat{Q} - \tilde{Q}\|_F = 0$ and $\|\hat{S}E - \tilde{R}\|_F = 0$ for all n, where matrices $\hat{Q}$, $\hat{S}$ are generated by BGSP from A, and matrices $\tilde{Q}$, $\tilde{R}$ are generated by MGS from AE.

5.2 Applications of the Sequential Solver

In this section, we compare BGSPSol for solving linear systems with three commonly used direct solvers: the solver of LU factorization with Partial Pivoting (PLUSol) [41], the Householder solver (HouseholderSol) [108], and the MGS solver (MGSSol) [15]. We conclude from the comparison that BGSPSol outperforms all other methods for solving ill-conditioned banded systems. We also use the MATLAB package to program and run the experiments in this section.

In Example 2, we compare BGSPSol with PLUSol, HouseholderSol, and MGSSol on the accuracy of solving ill-conditioned banded systems. Notices that matrices of similar structure as those in this example are used as benchmark matrices for testing LAPACK routines [3]. The accuracy is measured by the relative error $\|x^* - x\|_2/\|x^*\|_2$, where x^* is the exact solution, and x is the solution from a solver.

***Example* 2.** For the linear system (4.1), we consider two types of ill-conditioned banded matrices. Matrix $A' := [a_{ij} : i, j \in \mathbb{N}_n]$ is a banded matrix with bandwidth 3 (heptadiagonal), where $a_{ii} := 8$, for $i \in \mathbb{N}_n$, $a_{i+1,i} := -2$, $a_{i,i+1} := -4$, for $i \in \mathbb{N}_{n-1}$, and any other entry within the band is -1. Matrix $A'' := [a_{ij} : i, j \in \mathbb{N}_n]$ is a banded matrix with bandwidth 10, where $a_{ii} := 2$, for $i \in \mathbb{N}_n$, $a_{i+1,i} := 10 \ln i$, for $i \in \mathbb{N}_{n-1}$, and any other entry within the band is -1. The exact solution to (4.1) is set as $x^* := [x_i : i \in \mathbb{N}_n]^T$, with $x_i := e^{\omega_i + 6}\omega_i(1 - \omega_i)$, where $\omega_i := i/(n+1)$, for $i \in \mathbb{N}_n$. The condition numbers of the matrices for some n are listed in Table 5.1. The condition numbers ζ^ℓ for obtaining x^ℓ by BGSPSol as in (4.21) are given in Table 5.2. Notice that if x' is the computed solution to (4.1) obtained by HouseholderSol or MGSSol,

then there exists a constant $c := c(n)$ such that (see, [15, 108])

$$\frac{\|x^* - x'\|_2}{\|x^*\|_2} \leq c\mathbf{u}\kappa(A).$$

Lastly, the numerical results are presented in Figure 5.5.

TABLE 5.1
The condition numbers of the matrices in Example 2 for some n.

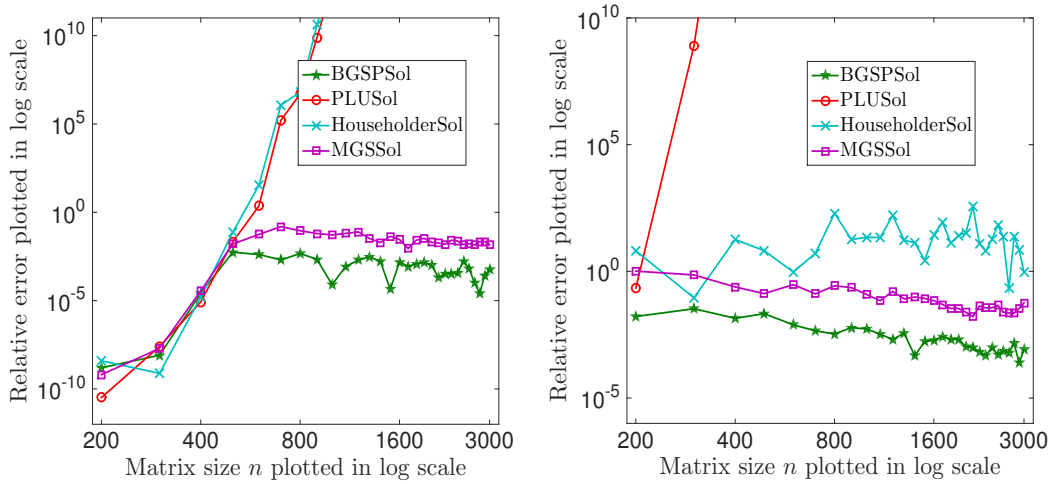
n	200	400	800	1600	3000
$\kappa(A')$	1.0431e+09	6.3773e+13	5.4973e+25	1.0030e+50	9.9438e+92
$\kappa(A'')$	1.4142e+18	2.1094e+38	1.5061e+81	1.9722e+171	∞

TABLE 5.2
The condition numbers ζ^ℓ by BGSPSol when $n = 3000$.

$\kappa(A') = 9.9438e+92$	
ℓ	ζ^ℓ
1	8.2529
2	5.4027e+05
3	8.2037e+06
4	3.6396e+11
5	3.2707e+18
6	1.4579e+25
7	2.1834e+26
8	1.4546e+29
9	1.3001e+32

$\kappa(A'') = \infty$	
ℓ	ζ^ℓ
1	2.1958
2	1.2816e+10
3	2.3683e+24
4	3.4963e+40
5	1.0198e+58
6	5.2926e+75
7	4.8769e+93
8	1.7461e+128

Figure 5.5 illustrates the superior accuracy of BGSPSol comparing to the others, and the robustness against the issue of ill-conditioning. For A', all solvers behave well when $n < 600$. The relative error of PLUSol or HouseholderSol grows roughly exponentially as n increases, and becomes unacceptable when $n \geq 600$. For $n \geq 600$, the relative error of MGSSol stays around 10^{-2}. Benefit from the breakdown process, BGSPSol maintains stably relative errors around 10^{-4} when $n \geq 600$. For A'', we observe a similar behavior for the relative errors of PLUSol, MGSSol, and BGSPSol, except that the threshold n value is 200 rather than 600. HouseholderSol obtains



Left: Relative error of solving $A'x = b$ versus matrix size n

Right: Relative error of solving $A''x = b$ versus matrix size n

FIGURE 5.5: *Comparison of the relative errors of the four solvers for Example 2.*

relative errors about $1 \sim 10^3$. This example shows that BGSPSol is robust for solving ill-conditioned systems.

In Examples 3-5 below, we present numerical results of the four solvers for the linear systems derived from three applications. Example 3 concerns a boundary value problem (BVP).

***Example* 3.** We consider the linear system (4.1), where the matrix A is tridiagonal and is derived from the discretization of the following BVP

$$\begin{cases} u''(t) + \alpha u'(t) + \beta u(t) = f(t), \quad t \in (0,1), \\ u(0) = u(1) = 0, \end{cases} \tag{5.1}$$

with α, β being constants. Using central difference method of second-order accuracy, we discretize (5.1) with n equally space points in $(0,1)$, as follows. For $i \in \mathbb{N}_{n+1} \cup \{0\}$, $d \in \mathbb{N} \cup \{0\}$, we let $h := 1/(n+1)$, $t_i := ih$, and $u_i^{(d)} = u^{(d)}(t_i)$. Then for each $i \in \mathbb{N}_n$,

$$\begin{aligned} u_i' &= \frac{1}{2h}(u_{i+1} - u_{i-1}) + O(h^2), \\ u_i'' &= \frac{1}{h^2}(u_{i+1} - 2u_i + u_{i-1}) + O(h^2). \end{aligned}$$

Hence we derive a tridiagonal system as (4.1), where $A := [a_{ij} : i, j \in \mathbb{N}_n]$ with

$$a_{ii} := -\frac{2}{h^2} + \beta, \quad a_{i+1,i} := \frac{1}{h^2} - \frac{\alpha}{2h}, \quad a_{i,i+1} := \frac{1}{h^2} + \frac{\alpha}{2h}, \quad \text{for} \quad i \in \mathbb{N}_{n-1},$$

$a_{nn} := -\frac{2}{h^2} + \beta$, and $x := [u_i : i \in \mathbb{N}_n]^T$. When $\alpha = 100$, $\beta = 100000$, the matrix A is ill-conditioned with condition numbers for some n given in Table 5.3. In this example, we let $u^*(t) := t(1-t)$, for $t \in [0, 1]$ be the exact solution of (5.1), $x^* := [u^*(t_i) : i \in \mathbb{N}_n]^T$, and $b := Ax^*$. The numerical results are illustrated in Figure 5.6.

TABLE 5.3
The condition numbers of the derived matrices in Example 3 for some n.

n	200	400	800	1600	3000
$\kappa(A)$	8.0930e+22	1.2316e+23	1.0907e+24	1.7633e+24	6.0776e+24

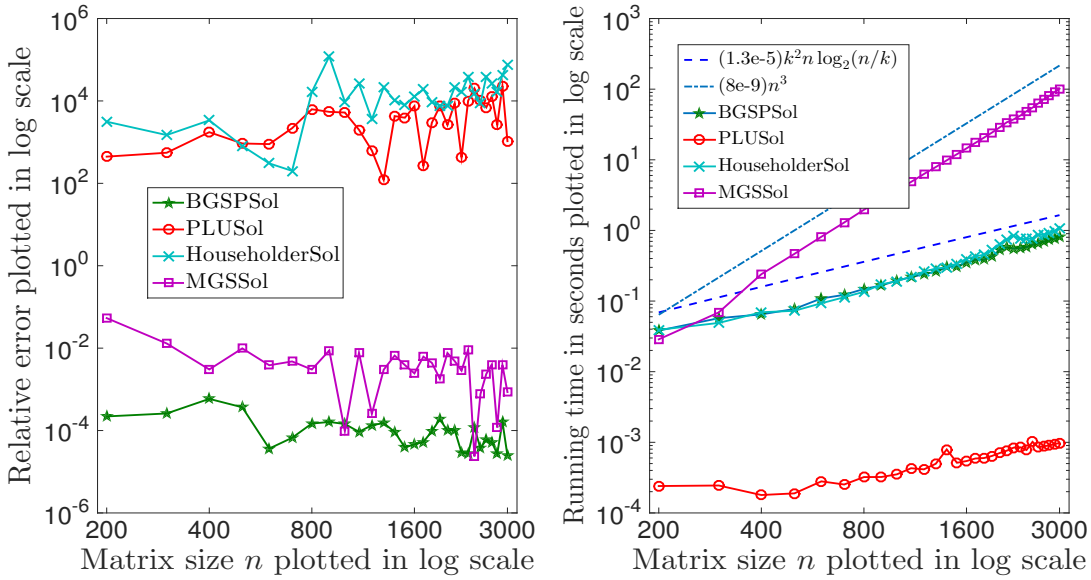


FIGURE 5.6: *Comparison of the relative errors and the running time for Example 3.*

For Example 3, Figure 5.6 shows that BGSPSol achieves the best accuracy in most cases, and runs the second fastest among the four solvers. The relative error of BGSPSol is roughly decreasing from 10^{-3} to 10^{-5} in magnitude as n increases. The relative errors of PLU and HouseholderSol are too large to be acceptable. MGS has

competitive results as BGSPSol in terms of accuracy. However, the running time of MGS is approximately proportional to n^3, while that of BGSPSol is bounded above by $O(k^2 n \log_2(n/k))$.

We present Example 4 below for solving the linear system derived from the discretization of a singularly perturbed BVP. Problems of this kind occur frequently in many applications, such as the modeling of steady and unsteady viscous flow problems with large Reynolds number, and convective heat transport problems with large Peclet number. The challenge to solve such a problem numerically is due to the occurrence of a sharp boundary-layer where the solution varies rapidly, while away from the layer the solution behaves regularly and varies slowly (see, for example, [19, 74]).

***Example* 4.** We consider the linear system (4.1), where A is a heptadiagonal matrix derived from the discretization of the following the singularly perturbed BVP with standard center difference scheme.

$$\begin{cases} \varepsilon^2 u^{(4)}(t) + t^3 u''(t) + 8u'(t) - 10e^{6-t}u(t) = f(t), \quad t \in (0,1), \\ u(0) = u'(0) = u(1) = u'(1) = 0. \end{cases} \tag{5.2}$$

For higher accuracy, we use central difference method of order $O(h^4)$ to discretize (5.2) with n equally space points in $(0,1)$, as follows. Let h, t_i and $u_i^{(d)}$ be defined as in Example 3. Then for each $i \in \mathbb{N}_{n-2}\backslash\{1,2\}$,

$$\begin{aligned} u_i' &= \frac{1}{12h}\left(-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}\right) + O(h^4), \\ u_i'' &= \frac{1}{12h^2}\left(-u_{i+2} + 16u_{i+1} - 30u_i + 16u_{i-1} - u_{i-2}\right) + O(h^4), \\ u_i^{(4)} &= \frac{1}{6h^4}\left(-u_{i+3} + 12u_{i+2} - 39u_{i+1} + 56u_i - 39u_{i-1} + 12u_{i-2} - u_{i-3}\right) + O(h^4). \end{aligned}$$

Thus the linear system (4.1) is derived, where A is a heptadiagonal matrix. When

$\varepsilon = 10^{-4}$, the condition numbers of A for some n are listed in Table 5.4. Let $u^*(t) := t^2(1-t)^2$, for $t \in [0,1]$ be the exact solution of (5.2), x, x^* and b be defined as in Example 3. We present the numerical results in Figure 5.7.

TABLE 5.4
The condition numbers of the derived matrices in Example 4 for some n.

n	400	600	800	1600	3000
$\kappa(A)$	1.6314e+21	3.3378e+35	9.6810e+38	5.6673e+43	4.5929e+45

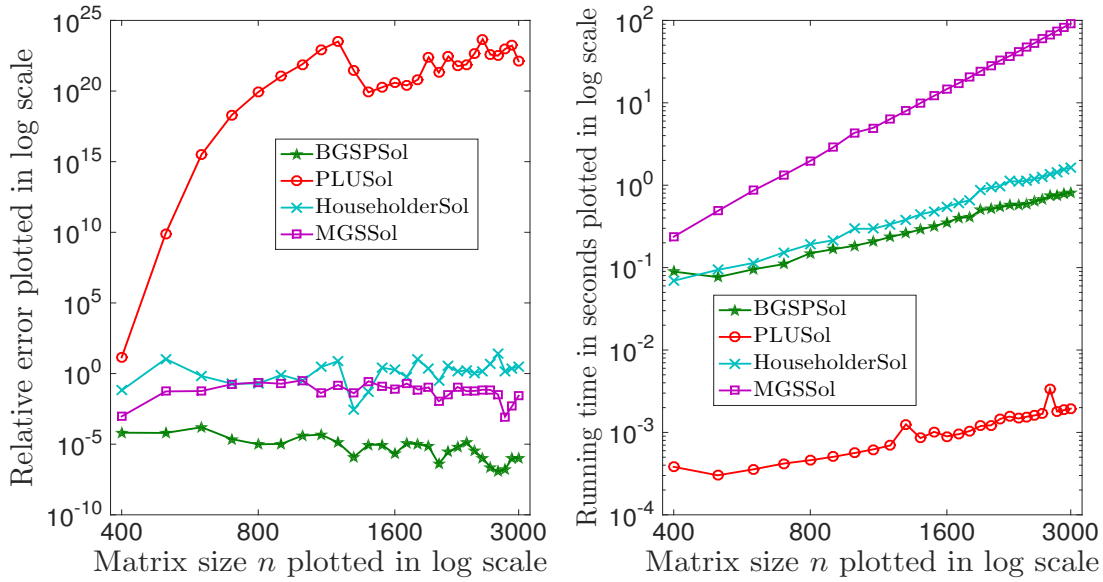


FIGURE 5.7: *Comparison of the relative errors and the running time for Example 4.*

For Example 4, Figure 5.7 illustrates that BGSPSol outperforms all the other solvers in accuracy, with gently decreasing relative errors from 10^{-5} to 10^{-7} in magnitude as n increases. The running time for each solver behaves similarly as the previous example.

In the following example, we consider the linear system derived from generalized finite element methods (GFEM). In recent years, GFEM is effectively applied to the problems of modeling non-smooth solutions such as crack-propagation and solid-fluid

interactions. A common concern is the about the stability and conditioning. Especially, for the approximation space used in GFEM, if the "angle" between the finite element space and the enrichment space is close to 0, then the derived matrix is ill-conditioned (see, for example, [6, 40]). Here, we compare the results obtained by applying the four solvers to the corresponding linear system with such a matrix.

***Example* 5.** We consider the linear system (4.1) derived from GFEM applied to the following problem,

$$\begin{cases} -u''(t) = f(t), & t \in (0,1), \\ u(0) = \alpha, \quad u'(1) = \beta. \end{cases} \tag{5.3}$$

Given n, h, t_i as above, we let $u^*(t) := e^{11t}$, for $t \in [0,1]$ be the exact solution of (5.3), and $N_i(t)$ be the standard hat-function associated with the node t_i, for each $i \in \mathbb{N}_n \cup \{0\}$. Applying GFEM based on the shape functions $\{\psi_i(x) : i \in \mathbb{N}_{2n+1}\}$ as

$$\psi_i(x) = \begin{cases} N_{(i-1)/2}(t) \cdot u^*(t), & i = 1, 3, \ldots, 2n+1, \\ N_{i/2}(t), & i = 2, 4, \ldots, 2n, \end{cases}$$

we obtain a heptadiagonal matrix $A := [a_{ij} : i, j \in \mathbb{N}_{2n+1}]$ with $a_{ij} := \int_0^1 \psi_j'(t)\psi_i'(t)\, dt$. Notice that the exact solution for (4.1) is $x^* := [1, 0, 1, 0, \ldots, 1, 0, 1]^T$. The condition numbers of A for some n are listed in Table 5.5, and the numerical results are illustrated in Figure 5.8.

TABLE 5.5
The condition numbers of the derived matrices in Example 5 for some n.

n	200	400	800	1600	3000
$\kappa(A)$	2.7042e+21	1.8153e+23	1.1896e+25	5.8345e+26	2.3661e+27

Figure 5.8 shows that BGSPSol achieves the best accuracy, with relative errors between 10^{-2} and 10^{-3} in magnitude. The other solvers fail to obtain meaningful

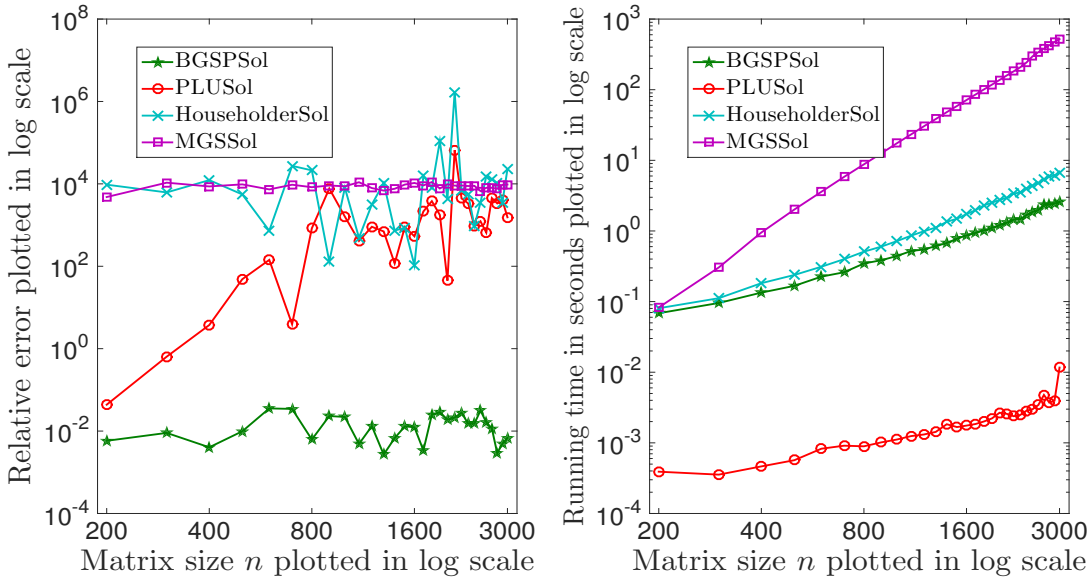


FIGURE 5.8: *Comparison of the relative errors and the running time for Example 5.*

solutions.

5.3 Parallel Experiments

In this section, we compare ParBGSPSol with the ScaLAPACK routine PxGBSV for solving linear systems in parallel computing, and present an experiment of applying ParBGSPSol to extra large-scale linear systems. The goal of this section is to demonstrate the advantages of ParBGSPSol in accuracy, scalability, and the capability of solving extra large-scale linear systems.

In the parallel environment, all the programs run on the Tianhe-2 (MilkyWay-2, or TH-2) supercomputer [36] at the National Supercomputer Center in Guangzhou, China. TH-2 is a 33.86-petaflop supercomputer based on Intel multicore (Ivy Bridge) and coprocessors (Xeon Phi), with TH-Express 2 interconnection network. There are 16,000 compute nodes in TH-2. Each compute node is composed of 2 sockets with

Intel Xeon IvyBridge E5-2692 (12 cores, 2.2 GHz) processors, and 3 Intel Xeon Phi accelerators based on the many-integrated-core (MIC) architecture. Each node has 64GB memory and runs on a custom Linux system, Kylin. Our code is compiled using Intel ICC 14.0.1 compiler and uses a custom version of MPICH on TH-2 for the MPI implementation.

In Example 6, we compare ParBGSPSol with the ScaLAPACK routine PxGBSV ([29], abbreviated by ScaLAPACKSol) on the relative error, speedup, and running time for solving ill-conditioned banded systems. Notices that matrices of similar structure as those in this example are used as benchmark matrices for testing ScaLAPACK routines [16].

***Example* 6.** We test five types of ill-conditioned systems whose matrices are banded, as described in Table 5.6. The exact solution x^* is defined as in Example 2. Hypothesis (AP) is verified by Table 5.7. The numerical results are plotted in Figures 5.9-5.11.

TABLE 5.6
The description of the five types of matrices.

test	matrix description
T1	$A := [a_{ij} : i, j \in \mathbb{N}_n]$ is a tridiagonal matrix with $a_{ii} := 2$, for $i \in \mathbb{N}_n$, and $a_{i+1,i} := -1.05$, $a_{i,i+1} := -1$, for $i \in \mathbb{N}_{n-1}$.
T2	$A := [a_{ij} : i, j \in \mathbb{N}_n]$ is a tridiagonal matrix with $a_{ii} := 2$, for $i \in \mathbb{N}_n$, and $a_{i+1,i} := i$, $a_{i,i+1} := -1$, for $i \in \mathbb{N}_{n-1}$.
T3	$A := [a_{ij} : i, j \in \mathbb{N}_n]$ is a pentadiagonal matrix with $a_{ii} := 4$, for $i \in \mathbb{N}_n$, $a_{i+1,i} := -2$, $a_{i,i+1} := -6$, for $i \in \mathbb{N}_{n-1}$, and $a_{i+2,i} = a_{i,i+2} = -1$ for $i \in \mathbb{N}_{n-2}$.
T4	$A := [a_{ij} : i, j \in \mathbb{N}_n]$ is a banded matrix with bandwidth 10. $a_{ii} := 2$, for $i \in \mathbb{N}_n$, $a_{i+1,i} := 10 \ln i$, for $i \in \mathbb{N}_{n-1}$, and any other entry within the band is -1.
T5	$A := [a_{ij} : i, j \in \mathbb{N}_n]$ is a banded matrix with bandwidth 20. $a_{ii} := 2$, for $i \in \mathbb{N}_n$, $a_{i+1,i} := i$, for $i \in \mathbb{N}_{n-1}$, and any other entry within the band is -1.

TABLE 5.7
The errors $\epsilon_{ip} := \|M_p(M_p^T M_p)^{-1} M_p^T G_{ip} - M_{ip}(M_{ip}^T M_{ip})^{-1} M_{ip}^T G_{ip}\|_F$, $i = 1, 2$, *for verifying Hypothesis (AP), where matrix size* $n = 2^{26}$, $P = 512$.

test	ϵ_{1p}, for each $p \in \mathbb{N}_P$	ϵ_{2p}, for each $p \in \mathbb{N}_P$
T1	0	0
T2	0	0
T3	0	0
T4	0	0
T5	0	0

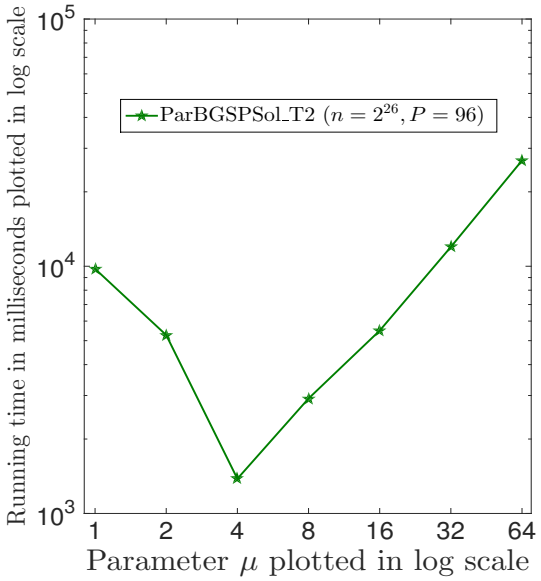


FIGURE 5.9: *The optimum of* μ *of ParBGSPSol for T2 (here the optimum is 4).*

Table 5.7 lists the errors $\epsilon_{ip} := \|M_p(M_p^T M_p)^{-1} M_p^T G_{ip} - M_{ip}(M_{ip}^T M_{ip})^{-1} M_{ip}^T G_{ip}\|_F$, for $i = 1, 2$, where M_p, G_{ip}, M_{ip} are defined in Hypothesis (AP) from the five types of matrices of size $n = 2^{26}$, and $P = 512$. It verifies from the table that $\epsilon_{ip} = 0$ for $i = 1, 2$, and for each $p \in \mathbb{N}_P$. That is, Hypothesis (AP) holds.

Figure 5.9 displays the running time of ParBGSPSol for different values of the parameter μ, which appears in the partition vector of the input matrix. In this case, the optimum of μ that minimizes the running time is 4. This verifies the effectiveness of selecting the optimal μ for computational efficiency, as described in section 3.3.

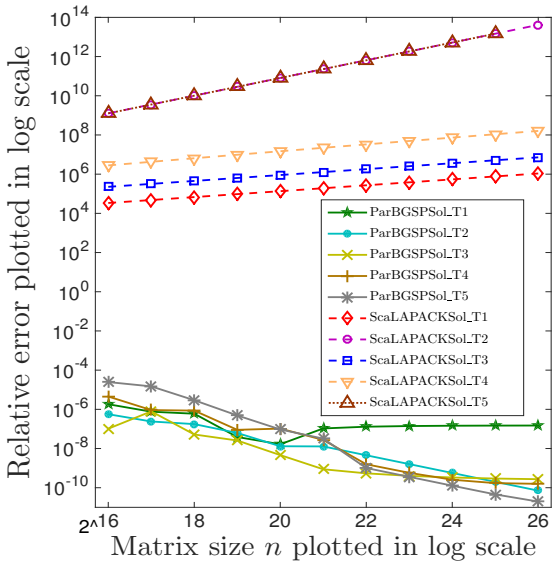


FIGURE 5.10: *Comparison of the relative errors of ParBGSPSol and ScaLAPACKSol.*

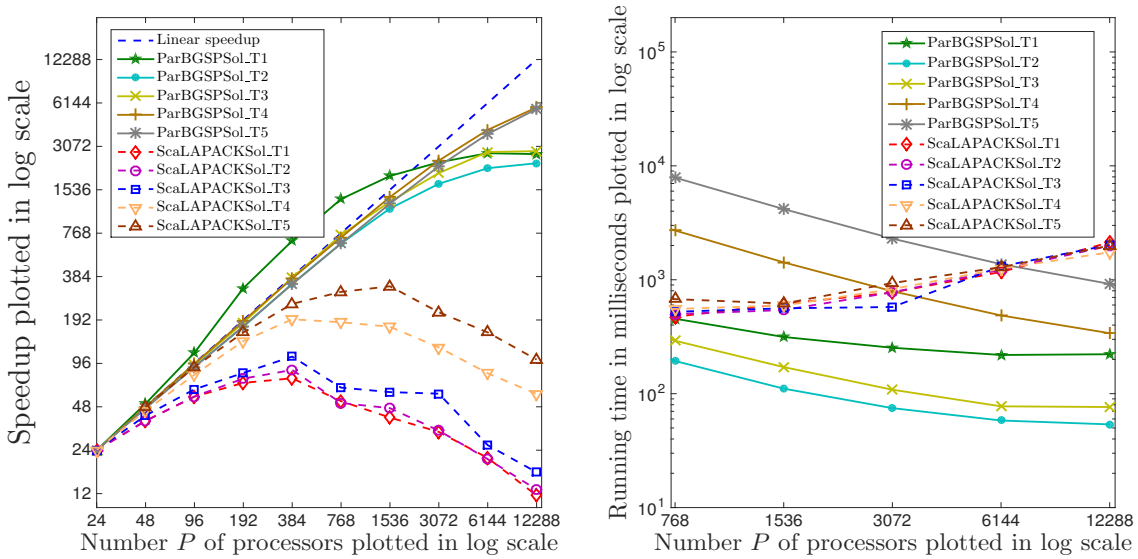


FIGURE 5.11: *Comparisons of the speedups and the running time when matrix size* $n = 2^{26}$.

Figure 5.10 shows that ParBGSPSol is superior to ScaLAPACKSol in accuracy. ParBGSPSol maintains about 10^{-5} to 10^{-11} relative errors in magnitude, while ScaLAPACKSol fails to obtain meaningful solutions due to the large relative errors.

Figure 5.11 shows that ParBGSPSol is much better than ScaLAPACKSol in scal-

ability. It shows that ParBGSPSol maintains approximately linear speedups for all tests when $P < 12288$, while ScaLAPACKSol does not even when $P > 192$. We also see that when $P \geq 6144$, ParBGSPSol runs faster than ScaLAPACKSol for all tests.

In the last example below, we demonstrate an experiment of solving extra large-scale systems by ParBGSPSol.

***Example* 7.** We consider the linear system (4.1), with matrix A of the same type as that in Example 6(T1) . The matrix size n is ranging from 8 billion to 64 billion. The exact solution x^* is defined as in Example 2. The result is presented in Figure 5.12.

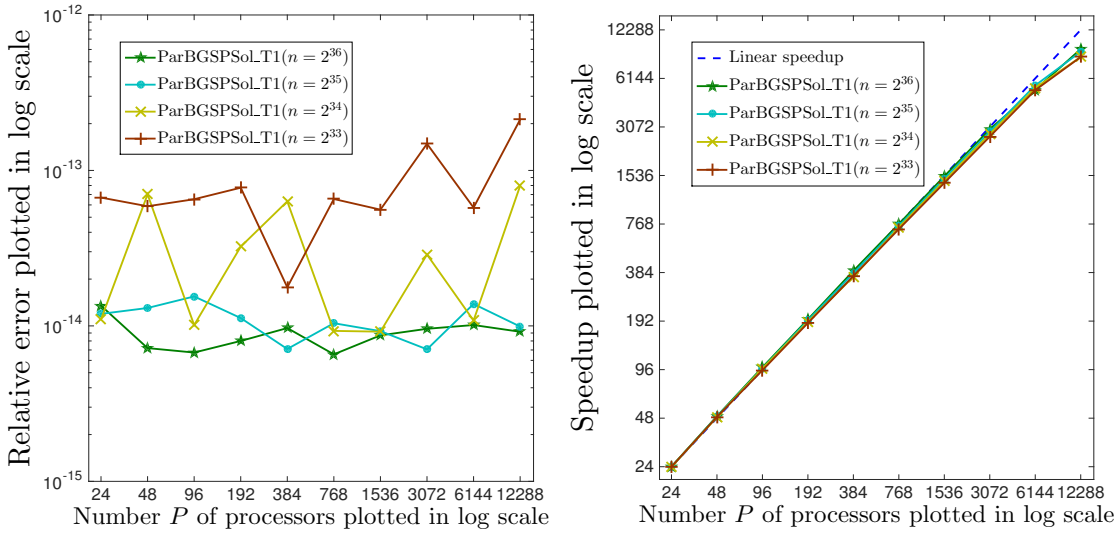


FIGURE 5.12: *The relative errors and the speedups of ParBGSPSol for Example 7.*

As seen from Figure 5.12, ParBGSPSol is capable of and efficient in solving extra large-scale systems. It maintains about 10^{-13} to 10^{-15} relative errors in magnitude, while having approximately linear speedups for $P < 12288$. Notice that ScaLAPACK-Sol fails to obtain solutions when $n/P \geq 2^{28}$ due to an error of insufficient memory.

5.4 A Brief Summary

We briefly summarize the comparison results from previous sections in Tables 5.8-5.10.

TABLE 5.8
Comparison of the three methods for matrix orthogonal factorization.

method	$\#(Q)$	$\#(S)$ or $\#(R)$	$\|A-\hat{Q}\hat{S}\|_F$ or $\|A-\hat{Q}\hat{R}\|_F$	$\|\hat{Q}^T\hat{Q}-I\|_2$
BGSP	$2km\log_2(m/k)$	$\frac{13}{4}km$	$3\sqrt{6}k\mathbf{u}\|A\|_F$	$c_m\mathbf{u}\kappa(A)$
Householder	$O(km)$ in implicit form, $O(m^2)$ in explicit form	$O(km)$	$c_{n,m}\mathbf{u}\|A\|_F$	$c_{n,m}\mathbf{u}$
MGS	$O(m^2)$	$O(km)$	$\frac{3}{2}(m-1)\mathbf{u}\|A\|_F$	$c_m\mathbf{u}\kappa(A)$

TABLE 5.9
Comparison of the four sequential solvers for linear systems.

method	condition number	relative error	running speed
BGSPSol	ζ^ℓ, with $\zeta^\ell \ll \kappa(A)$ for each ℓ	$\approx 10^{-4}$, decreasing	The second fastest
PLUSol	$\kappa(A)$	$> 10^1$, unacceptable	Fastest
HouseholderSol	$\kappa(A)$	$> 10^0$, unacceptable	Close to BGSPSol
MGSSol	$\kappa(A)$	$\approx 10^{-2}$	Slowest

TABLE 5.10
Comparison of the two parallel solvers for linear systems.

method	relative error	speedup: linear in P	running speed	ability to treat extra large-scale
ParBGSPSol	$10^{-5} \sim 10^{-15}$	Yes	Fast	Yes
ScaLAPACKSol	$10^4 \sim 10^{14}$	No	Slow	No

Bibliography

[1] D. Agarwal and B. C. Chen, *fLDA: matrix factorization through latent dirichlet allocation*, in Proceedings of the 3rd ACM International Conference on Web Search and Data Mining, ACM, 2010, pp. 91-100.

[2] M. Anderson, G. Ballard, J. Demmel, and K. Keutzer, *Communication-avoiding QR decomposition for GPUs*, in Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International, pp. 48-58.

[3] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen, *LAPACK Users' Guide*, 3rd ed., SIAM, Philadelphia, PA, 1999; also available online from http://www.netlib.org.

[4] G. R. Andrews, *Foundations of Multithreaded, Parallel, and Distributed Programming*, Addison-Wesley, Boston, MA, 2000.

[5] W. F. Ames, *Numerical Methods for Partial Differential Equations*, 3rd ed., Academic Press, Boston, MA, 2014.

[6] I. Babuška and U. Banerjee, *Stable generalized finite element method (SGFEM)*, Comput. Methods Appl. Mech. Engrg., 201 (2012), pp. 91-111.

[7] G. Ballard, E. Carson, J. Demmel, M. Hoemmen, N. Knight, and O. Schwartz, *Communication lower bounds and optimal algorithms for numerical linear algebra*, Acta Numer., 23 (2014), pp. 1-155.

[8] A. L. Barabási and Z. N. Oltvai, *Network biology: understanding the cell's functional organization*, Nature Reviews Genetics, 5.2 (2004), pp. 101-113.

[9] J. Basilico and T. Hofmann, *Unifying collaborative and content-based filtering*, in Proceedings of the 21st International Conference on Machine Learning, ACM, 2004, p. 9.

[10] R. Bell, Y. Koren, and C. Volinsky, *Modeling relationships at multiple scales to improve accuracy of large recommender systems*, in Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2007, pp. 95-104.

[11] J. Bennett, and S. Lanning, *The Netflix prize*, in Proceedings of KDD Cup and Workshop, 2007, p. 35.

[12] C. Bischof and C. Van Loan, *The WY representation for products of Householder matrices*, SIAM Journal on Scientific and Statistical Computing, 8.1 (1987), s2-s13.

[13] Å. Björck, *Solving linear least squares problems by Gram-Schmidt orthogonalization*, BIT, 7.1 (1967), pp. 1-21.

[14] Å. Björck and C. C. Paige, *Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm*, SIAM J. Matrix Anal. Appl., 13.1 (1992), pp. 176-190.

[15] Å. Björck and C. C. Paige, *Solution of augmented linear systems using orthogonal factorizations*, BIT, 34.1 (1994), pp. 1-24.

[16] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*, SIAM, Philadelphia, PA, 1997; also available online from http://www.netlib.org.

[17] L. Bolelli, S. Ertekin, and C. L. Giles, *Clustering scientific literature using sparse citation graph analysis*, in European Conference on Principles of Data Mining and Knowledge Discovery, Springer Berlin Heidelberg, 2006, pp. 30-41.

[18] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed., SIAM, Philadelphia, PA, 2000.

[19] L. Barbu and G. Morosanu, *Singularly Perturbed Boundary-Value Problems*, Springer Science & Business Media, 2007.

[20] M. Benzi and C. D. Meyer, *A direct projection method for sparse linear systems*, SIAM J. Sci. Comput., 16.5 (1995), pp. 1159-1176.

[21] D. Calvetti, L. Reichel, and D. C. Sorensen, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 2.1 (1994), pp. 1-21.

[22] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, *Reconstruction and representation of 3D objects with radial basis functions*, in Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ACM, New York, 2001, pp. 67-76.

[23] E. Carson, *Communication-avoiding Krylov Subspace Methods in Theory and Practice*, University of California, Berkeley, ProQuest Dissertations Publishing, 2015.

[24] E. Carson, N. Knight, and J. Demmel, *Avoiding communication in non-symmetric Lanczos-based Krylov subspace methods*, SIAM J. Sci. Comput., 35.5 (2013), pp. S42-S61.

[25] N. Castro-González, J. Ceballos, F. M. Dopico, and J. M. Molera, *Accurate solution of structured least squares problems via rank-revealing decompositions*, SIAM J. Matrix Anal. Appl., 34.3 (2013), pp. 1112-1128.

[26] T. F. Chan, *Rank revealing QR factorizations*, Linear Algebra Appl., 88 (1987), pp. 67-82.

[27] E. Chan, M. Heimlich, A. Purkayastha, and R. Van De Geijn, *Collective communication: theory, practice, and experience*, in Concurrency and Computation: Practice and Experience, 19, G. C. Fox and D. W. Walker, eds., John Wiley, New York, 2007, pp. 1749-1783.

[28] C. P. Chen and C. Y. Zhang, *Data-intensive applications, challenges, techniques and technologies: A survey on big data*, Inform. Sci., 275 (2014), pp. 314-347.

[29] A. Cleary and J. J. Dongarra, *Implementation in ScaLAPACK of divide-and-conquer algorithms for banded and tridiagonal linear systems*, Computer Science Dept. Technical Report CS-97-358, University of Tennessee, Knoxville, TN, 1997.

[30] P. G. Constantine and D. F. Gleich, *Tall and skinny QR factorizations in MapReduce architectures*, in Proceedings of the 2nd International Workshop on MapReduce and its Applications, ACM, 2011, pp. 43-50.

[31] G. F. Coulouris, J. Dollimore, and T. Kindberg, *Distributed Systems: Concepts and Design*, 5th ed., Pearson Education, 2011.

[32] T. A. Davis, *Algorithm 915, SuiteSparseQR: Multifrontal multithreaded rank-revealing sparse QR factorization*, ACM Trans. Math. Software, 38.1 (2011), article 8.

[33] J. Dean and S. Ghemawat, *MapReduce: Simplified data processing on large clusters*, Communications of the ACM, 51.1 (2008), pp. 107-113.

[34] J. Demmel, L. Grigori, M. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, SIAM J. Sci. Comput., 34.1 (2012), pp. A206-A239.

[35] J. Demmel, M. Gu, S. Eisenstat, I. Slapničar, K. Veselić, and Z. Drmač, *Computing the singular value decomposition with high relative accuracy*, Linear Algebra Appl., 299.1-3 (1999), pp. 21-80.

[36] J. J. Dongarra, *Trip report to Changsha and the Tianhe-2 supercomputer*, http://www.netlib.org/utk/people/JackDongarra/PAPERS/tianhe-2-dongarra-report.pdf, 2013.

[37] J. J. Dongarra and L. Johnsson, *Solving banded systems on a parallel processor*, Parallel Comput., 5.1-2 (1987), pp. 219-246.

[38] J. J. Dongarra and A. H. Sameh, *On some parallel banded system solvers*, Parallel Comput., 1.3-4 (1984), pp. 223-235.

[39] S. S. Epp, *Discrete Mathematics with Applications*, 4th ed., Cengage Learning, Boston, MA, 2011.

[40] T. P. Fries and T. Belytschko, *The extended/generalized finite element method: an overview of the method and its applications*, International Journal for Numerical Methods in Engineering, 84.3 (2010), pp. 253-304.

[41] J. R. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in MATLAB: design and implementation*, SIAM J. Matrix Anal. Appl., 13.1 (1992), pp. 333-356.

[42] M. Girvan and M. E. Newman, *Community structure in social and biological networks*, in Proceedings of the National Academy of Sciences, 99.12 (2002), pp. 7821-7826.

[43] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 4th ed., The Johns Hopkins University Press, Baltimore, MD, 2012.

[44] G. H. Golub, A. H. Sameh, and V. Sarin, *A parallel balance scheme for banded linear systems*, Numer. Linear Algebra Appl., 8.5 (2001), pp. 297-316.

[45] C. A. Gomez-Uribe and N. Hunt, *The Netflix recommender system: Algorithms, business value, and innovation*, ACM Transactions on Management Information Systems, 6.4 (2016), article 13.

[46] G. Groh and C. Ehmig, *Recommendations in taste related domains: Collaborative filtering vs. social filtering*, in Proceedings of the 2007 International ACM Conference on Supporting Group Work, ACM, 2007, pp. 127-136.

[47] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, 2nd ed., MIT Press, Cambridge, MA, 1999.

[48] C. Gu, *Smoothing Spline ANOVA Models*, 2nd ed., Springer, New York, 2013.

[49] M. R. Guarracino, F. Perla, and P. Zanetti, *A parallel block Lanczos algorithm and its implementation for the evaluation of some eigenvalues of large sparse symmetric matrices on multicomputers*, Int. J. Appl. Math. Comput. Sci., 16.2 (2006), pp. 241-249.

[50] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, *Evaluating collaborative filtering recommender systems*, ACM Transactions on Information Systems, 22.1 (2004), pp. 5-53.

[51] V. Hernández, J. E. Román, and A. Tomás, *Parallel Arnoldi eigensolvers with enhanced scalability via global communications rearrangement*, Parallel Comput., 33.7 (2007), pp. 521-540.

[52] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An overview of the Trilinos project*, ACM Trans. Math. Software, 31.3 (2005), pp. 397-423.

[53] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., SIAM, Philadelphia, PA, 2002.

[54] N. J. Higham, *QR factorization with complete pivoting and accurate computation of the SVD*, Linear Algebra Appl., 309.1-3 (2000), pp. 153-174.

[55] R. W. Hockney, *A fast direct solution of Poisson's equation using Fourier analysis*, J. ACM, 12.1 (1965), pp. 95-113.

[56] M. Hoemmen, *A communication-avoiding, hybrid-parallel, rank-revealing orthogonalization method*, in Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International, pp. 966-977.

[57] M. Hoemmen, *Communication-avoiding Krylov Subspace Methods*, University of California, Berkeley, ProQuest Dissertations Publishing, 2010.

[58] P. D. Hoff, *Multiplicative latent factor models for description and prediction of social networks*, Computational and Mathematical Organization Theory, 15.4 (2009), p. 261.

[59] P. D. Hoff, A. E. Raftery, and M. S. Handcock, *Latent space approaches to social network analysis*, J. Amer. Statist. Assoc., 97.460 (2002), pp. 1090-1098.

[60] T. Hofmann, *Probabilistic latent semantic indexing*, in Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 1999, pp. 50-57.

[61] IEEE Standards Committee, *754-2008 IEEE standard for floating-point arithmetic*, IEEE Computer Society Std, 2008.

[62] W. Jalby and B. Philippe, *Stability analysis and improvement of the block Gram-Schmidt algorithm*, SIAM Journal on Scientific and Statistical Computing, 12.5 (1991), pp. 1058-1073.

[63] M. Jamali and M. Ester, *A matrix factorization technique with trust propagation for recommendation in social networks*, in Proceedings of the 4th ACM Conference on Recommender Systems, ACM, 2010, pp. 135-142.

[64] M. Jamali and M. Ester, *Trustwalker: a random walk model for combining trust-based and item-based recommendation*, in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2009, pp. 397-406.

[65] C. Johnson, *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Courier Corporation, North Chelmsford, MA, 2012.

[66] K. Kambatla, G. Kollias, V. Kumar, and A. Grama, *Trends in big data analytics*, Journal of Parallel and Distributed Computing 74.7 (2014), pp. 2561-2573.

[67] I. Koch, T. Lengauer, and E. Wanke, *An algorithm for finding maximal common subtopologies in a set of protein structures*, J. Comput. Biol., 3.2 (1996), pp. 289-306.

[68] Y. Koren, R. Bell, and C. Volinsky, *Matrix factorization techniques for recommender systems*, in Computer, 42.8 (2009), pp. 30-37.

[69] V. Kumar, A. Grama, A. Gupta, and G. Karypis, *Introduction to Parallel Computing: Design and Analysis of Algorithms*, Benjamin-Cummings, Redwood City, CA, 1994.

[70] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong, *Addressing cold-start problem in recommendation systems*, in Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, ACM, 2008, pp. 208-211.

[71] J. Leskovec and J. J. Mcauley, *Learning to discover social circles in ego networks*, in Advances in Neural Information Processing Systems 2012, pp. 539-547.

[72] X. S. Li and J. W. Demmel, *SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems*, ACM Trans. Math. Software, 29.2 (2002), pp. 110-140.

[73] G. Linden, B. Smith, and J. York, *Amazon.com recommendations: Item-to-item collaborative filtering*, IEEE Internet computing, 7.1 (2003), pp. 76-80.

[74] S. T. Liu and Y. Xu, *Galerkin methods based on Hermite splines for singular perturbation problems*, SIAM J. Numer. Anal., 43.6 (2006), pp. 2607-2623.

[75] N. A. Lynch, *Distributed Algorithms*, Morgan Kaufmann, Burlington, MA, 1996.

[76] H. Ma, H. Yang, M. R. Lyu, and I. King, *SoRec: social recommendation using probabilistic matrix factorization*, in Proceedings of the 17th ACM Conference on Information and Knowledge Management, ACM, 2008, pp. 931-940.

[77] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King, *Recommender systems with social regularization*, in Proceedings of the 4th ACM International Conference on Web Search and Data Mining, ACM, 2011, pp. 287-296.

[78] I. MacKenzie, C. Meyer, and S. Noble, *How retailers can keep up with consumers*, McKinsey & Company, 2013.

[79] *MATLAB and Statistics Toolbox Release 2016a*, The MathWorks, Inc., Natick, MA, 2016.

[80] U. Meijer, *A parallel partition method for solving banded systems of linear equations*, Parallel Comput., 2.1 (1985), pp. 33-43.

[81] A. Mnih and R. R. Salakhutdinov, *Probabilistic matrix factorization*, in Advances in Neural Information Processing Systems, 2008, pp. 1257-1264.

[82] R. Morgan, *On restarting the Arnoldi method for large nonsymmetric eigenvalue problems*, Math. Comp., 65.215 (1996), pp. 1213-1230.

[83] National Research Council, *Frontiers in Massive Data Analysis*, National Academies Press, 2013.

[84] National Research Council, *Getting up to Speed: The Future of Supercomputing*, National Academies Press, 2005.

[85] M. E. J. Newman, *Finding community structure in networks using the eigenvectors of matrices*, Phys. Rev. E 74.3 (2006), p. 036104.

[86] W. Peng and B. N. Datta, *A sparse QS-decomposition for large sparse linear system of equations*, in Domain Decomposition Methods in Science and Engineering XIX, Y. Huang, R. Kornhuber, O. Widlund, and J. Xu, eds., Springer, New York, 2011, pp. 431-438.

[87] E. Polizzi and A. H. Sameh, *A parallel hybrid banded system solver: the Spike algorithm*, Parallel Comput., 32.2 (2006), pp. 177-194.

[88] F. Provost and T. Fawcett, *Data science and its relationship to big data and data-driven decision making*, Big Data, 1.1 (2013), pp. 51-59.

[89] S. Purushotham, Y. Liu, and C. C. J. Kuo, *Collaborative topic regression with social matrix factorization for recommendation systems*, arXiv preprint arXiv:1206.4684 (2012).

[90] U. N. Raghavan, R. Albert, and S. Kumara, *Near linear time algorithm to detect community structures in large-scale networks*, Phys. Rev. E, 76.3 (2007), p.036106.

[91] P. Resnick and H. R. Varian, *Recommender systems*, Communications of the ACM, 40.3 (1997), pp. 56-58.

[92] M. Sadkane, *Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems*, Numer. Math., 64.1 (1993), pp. 195-211.

[93] R. Salakhutdinov and A. Mnih, *Bayesian probabilistic matrix factorization using Markov chain Monte Carlo*, in Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 880-887.

[94] R. Samudrala and J. Moult, *A graph-theoretic algorithm for comparative modeling of protein structure*, Journal of Molecular Biology, 279.1 (1998), pp. 287-302.

[95] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, *Item-based collaborative filtering recommendation algorithms*, in Proceedings of the 10th International Conference on World Wide Web, ACM, 2001, pp. 285-295.

[96] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, *Collaborative filtering recommender systems*, in The Adaptive Web, Springer Berlin Heidelberg, 2007, pp. 291-324.

[97] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, *Methods and metrics for cold-start recommendations*, in Proceedings of the 25th Annual

International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2002, pp. 253-260.

[98] J. Scott, *Social Network Analysis*, Sage, Thousand Oaks, CA, 2017.

[99] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22.8 (2000), pp. 888-905.

[100] R. B. Sidje, *Alternatives for parallel Krylov subspace basis computation*, Numer. Linear Algebra Appl., 4.4 (1997), pp. 305-331.

[101] P. Singla and M. Richardson, *Yes, there is a correlation:- from social networks to personal behavior on the web*, in Proceedings of the 17th International Conference on World Wide Web, ACM, 2008, pp. 655-664.

[102] N. Srebro and T. Jaakkola, *Weighted low-rank approximations*, in Proceedings of the 20th International Conference on Machine Learning, 2003, pp. 720-727.

[103] H. S. Stone, *An efficient parallel algorithm for the solution of a tridiagonal linear system of equations*, J. ACM, 20.1 (1973), pp. 27-38.

[104] X. Su and T. M. Khoshgoftaar, *A survey of collaborative filtering techniques*, Advances in Artificial Intelligence, 2009 (2009), article 4.

[105] J. G. Sun, *Perturbation bounds for the Cholesky and QR factorizations*, BIT, 31.2 (1991), pp. 341-352.

[106] A. S. Tanenbaum and M. Van Steen, *Distributed Systems: Principles and Paradigms*, Prentice-Hall, Upper Saddle River, NJ, 2007.

[107] L. Tang and H. Liu, *Relational learning via latent social dimensions*, in Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, 2009, pp. 817-826.

[108] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[109] A. Tuzhilin, Y. Koren, J. Bennett, C. Elkan, and D. Lemire, *Large-scale recommender systems and the Netflix prize competition*, in KDD Proceedings. 2008.

[110] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow, *The anatomy of the facebook social graph*, arXiv preprint arXiv:1111.4503 (2011).

[111] L. G. Valiant, *A bridging model for parallel computation*, Communications of the ACM, 33.8 (1990), pp. 103-111.

[112] J. S. Vitter, *Algorithms and data structures for external memory*, Foundations and Trends® in Theoretical Computer Science, 2.4 (2008), pp. 305-474.

[113] J. S. Vitter, *External memory algorithms and data structures: Dealing with massive data*, ACM Computing Surveys, 33.2 (2001), pp. 209-271.

[114] H. H. Wang, *A parallel method for tridiagonal equations*, ACM Trans. Math. Software, 7.2 (1981), pp. 170-183.

[115] C. Wang and D. M. Blei, *Collaborative topic modeling for recommending scientific articles*, in Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2011, pp. 448-456.

[116] S. J. Wright, *Parallel algorithms for banded linear systems*, SIAM Journal on Scientific and Statistical Computing, 12.4 (1991), pp. 824-842.

[117] G. R. Xue, C. Lin, Q. Yang, W. Xi, H. J. Zeng, Y. Yu, and Z. Chen, *Scalable collaborative filtering using cluster-based smoothing*, in Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2005, pp. 114-121.

[118] J. Yang and J. Leskovec, *Defining and evaluating network communities based on ground-truth*, Knowledge and Information Systems, 42.1 (2015), pp. 181-213.

[119] S. H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng, and H. Zha, *Like like alike: Joint friendship and interest propagation in social networks*, in Proceedings of the 20th International Conference on World Wide Web, ACM, 2011, pp. 537-546.

BIOGRAPHICAL DATA

NAME OF AUTHOR: Qian Huang

PLACE OF BIRTH: China

DATE OF BIRTH: October 26, 1986

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

Sun Yat-sen University

DEGREES AWARDED:

B.S., Sun Yat-sen University, China, 2006

M.S., Sun Yat-sen University, China, 2008

PUBLICATION:

- Q. Huang and Y. Xu, *A Parallel Algorithm for Stable Sparse Orthogonal Factorization of Ill-Conditioned Banded Matrices*, preprint, 2017.

- Q. Huang and Y. Xu, *A Recursive Algorithm for Stable Sparse Orthogonal Factorization of Ill-Conditioned Banded Matrices*, preprint, 2017.

- Q. Huang and Y. Xu, *Stable Sparse Orthogonal Factorization of Ill-Conditioned Banded Matrices for Parallel Computing*, preprint, 2017.