



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

Approximate Linear Solvers for Scalable Statistical Algorithms

A thesis submitted to Trinity College Dublin, the University of Dublin
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Dung Tien Pham

Supervised by
Prof. Simon Wilson

Discipline of Statistics – School of Computer Science & Statistics
Trinity College Dublin, the University of Dublin.

2025

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

The copyright belongs jointly to 'Trinity College Dublin, the University of Dublin' and Dung Tien Pham.

A handwritten signature in black ink, consisting of a stylized 'D' and 'P' followed by a horizontal line.

Dung Tien Pham

March 2025

Abstract

Many statistical methods for large, multivariate datasets require solving linear systems whose dimensions are determined by either the number of observations or the size of individual data vectors. This often becomes a bottleneck when scaling these methods to accommodate large and complex datasets. In particular, the direct solution of such systems can become computationally expensive and memory-intensive, making it impractical for real-world applications involving massive data sizes. In this thesis, we explore the application of Krylov subspace methods to address this computational challenge, focusing on a source separation problem in cosmology where the data size is prohibitively large for conventional methods. Krylov subspace methods offer a way to iteratively approximate the solution of linear systems, reducing the computational complexity while maintaining accuracy. Two distinct approaches adapted from existing literature are presented: one applies the conjugate gradient method directly to the Kronecker-structured problem, and the other reformulates the problem as a Sylvester matrix equation. Both approaches are analyzed theoretically and their robust performance is demonstrated on large-scale simulated and real-world datasets.

The key contribution of this thesis is the development and implementation of approximate linear solvers that enables the scalability of statistical algorithms to very large datasets. By integrating Krylov subspace techniques with existing statistical models, we demonstrate a significant reduction in both memory requirements and computation time without compromising the accuracy of the solution. This contribution is particularly valuable for the cosmology domain, where the analysis of large-scale observational

data demands highly efficient methods for solving linear systems. Moreover, this thesis highlights the broader applicability of these approximate solvers in other fields of study, offering a scalable solution to the linear systems problem that arises in many high-dimensional statistical algorithms. The techniques developed here can be easily adapted to other statistical applications involving large datasets, such as machine learning and signal processing, where scalability is a critical factor.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Professor Simon Wilson, for his invaluable guidance and unwavering support throughout my PhD journey. His expertise and encouragement are instrumental in shaping my research direction and fostering my intellectual growth. Beyond his academic mentorship, Professor Wilson is a good friend and an inspiration, demonstrating the qualities of a true professional and gentleman.

Alongside with the mentorship from Professor Simon Wilson, I am very fortunate to receive the superb guidance and collaboration from Professor Kirk Soodhalter. His major contributions in our collaborative work have led us to multiple publications and laid the solid foundation for my research.

I would also like to thank the entire statistics community at Trinity for providing a stimulating and supportive environment. I am grateful for the opportunity to learn from my peers' passion and dedication. In particular, I would like to acknowledge Hieu Cao for his friendship and support, both academically and personally. I am also thankful to Joshua and Rishabh for their initiative in organizing the TCD Statistics Seminar Series, which has been a valuable platform for knowledge sharing and professional development. I am also indebted to the statistics faculty, led by Professors Simon Wilson and Caroline Brophy, for their generous support and commitment to fostering a thriving academic community.

Last but not least, I would like to thank my family for their unconditional love and support. My parents, who instilled in me a strong work ethic and the value of education, have been my constant source of encouragement. To my incredible wife, who has sacrificed so much for me to pursue my academic dreams, I am eternally grateful. You have given me three beautiful children and a loving home, creating the perfect environment for me to succeed. I am excited to embark on the next chapter of my life with you all.

Publications

The following journal articles and conference papers, arising from this work, are published

- Dung Pham, Kirk M. Soodhalter, and Simon Wilson. Fast solution of Sylvester-structured systems for spatial source separation of the cosmic microwave background. *Numerical Linear Algebra with Applications*, 32(1):e2597, 2025. doi: <https://doi.org/10.1002/nla.2597>.
- Dung Pham, Simon Wilson. “Compression of Autoregressive Language Models Using Kronecker Product Decomposition” *43rd Conference on Applied Statistics in Ireland* (2023)
- Dung Pham, Simon Wilson. “Exploiting Special Structures in Large Sparse Matrices for Efficient Matrix-Vector Multiplication” *13th Bayesian Inference for Stochastic Processes* (2023)

Contents

1	Introduction	1
1.1	Research Aims	2
1.2	Methodology	4
1.3	Thesis Structure	5
2	Literature Review	9
2.1	Cosmic Microwave Background	10
2.1.1	CMB Component Separation	10
2.1.2	The Linear Mixture Model	12
2.1.3	Blind Source Separation	14
2.2	Bayesian Inference	16
2.2.1	Likelihood Function	19
2.2.2	Prior Distribution	19
2.2.3	Posterior Distribution	20
2.2.4	Markov Chain Monte Carlo	21
2.3	Krylov Subspace Methods	25
2.3.1	Iterative Methods	25
2.3.2	The Conjugate Gradient Method	27
2.3.3	Lanczos-type Product Methods	30
3	Krylov Subspace Solver for Scalable CMB Source Separation	33
3.1	Summary	33

3.2	Source Separation Model for All-Sky Maps	33
3.2.1	All-Sky Map Partition	33
3.2.2	Mixing Matrix Construction	36
3.2.3	Problem Formulation	38
3.3	Solving the Linear System	42
3.3.1	Efficient matrix-vector products	45
3.3.2	Matrix-free iterative solvers	49
3.4	Solver Analysis	51
3.4.1	Theoretical Analysis	51
3.4.2	Practical Analysis	52
3.5	Conclusion	53
4	Linear Solver with Sylvester Interpretation	55
4.1	Summary	55
4.2	Sparse-Dense Sylvester Linear Equation Solvers	55
4.2.1	Sylvester Interpretation	55
4.2.2	Solving Sparse-Dense Sylvester Equation	56
4.3	Solver Analysis	58
4.3.1	Complexity Analysis	58
4.3.2	Practical Analysis	61
4.4	Conclusion	62
5	Lanczos-Sylvester Solver for Scalable Statistical Algorithms	63
5.1	Summary	63
5.2	Lanczos-Sylvester Linear Solver	63
5.2.1	Application of block Krylov approach for large-scale problems	63
5.2.2	The block Lanczos process with respect to $(\cdot, \cdot)_N$	65
5.2.3	Projection methods for Sylvester equations	68
5.3	Solver Analysis	71
5.3.1	Theoretical Analysis	71

5.3.2	Practical Analysis	74
5.3.3	Comparisons with Existing Techniques	75
5.4	Conclusion	78
6	Experimental Results	81
6.1	Summary	81
6.2	Experimental Setup	81
6.2.1	Experimental Design	81
6.2.2	Input Preparation	82
6.3	Evaluation and Comparisons	86
6.3.1	Evaluation Methodology	86
6.3.2	Evaluation Results	87
6.3.3	Code Profiling	92
7	Conclusions and Future Works	95
7.1	Conclusions	95
7.2	Future Works	96
7.2.1	CMB Source Separation with Polarization Data	96
7.2.2	Kronecker Product Exploitation in Matrix Compression	98
7.2.3	Generalized Algorithm for Gaussian Processes	101
	Appendix A: Source Code	117
	Appendix B: Code Profiling Details	118

List of Figures

2.1	Comparison of CMB results from COBE, WMAP and Planck. Image courtesy NASA/JPL-Caltech/ESA.	11
2.2	Component-separated maps at 80' resolution using SMICA methods. From left to right: Stokes I (temperature), Stokes Q and Stokes U (polarization).	16
2.3	The principle of conjugate directions method (2-D). [Yang et al., 2019]	29
3.1	HEALPix partitions. Clockwise from top left: the base partition into 12 pixels, then level 1, 2 and 3 (48, 192 and 768 pixels). Taken from https://healpix.sourceforge.io/	34
3.2	Sparseness of $\hat{Q} = Q + B^T C B$ (left) and it Cholesky decomposition (right) for a small example ($m = 4$ and $N = 256$). Non-zero elements are shaded.	44
6.1	Solution accuracy. Scatter plot of components of the true source vector $\mathcal{S}$ versus the solution vector (left) and source vector versus the error (right).	88
6.2	Running time (in seconds) by HEALPix level (left) and data size (right).	89
6.3	Total memory usage (in GB) by HEALPix level (left) and data size (right).	90

6.4	Code profiling. The functions that consume the most resources as a function of problem size. The top row is percentage of total computation time and the bottom row is percentage of maximum memory use. Table 6.5 provides more information on the functions. <i>Sylv.</i> refers to Algorithm 10.	93
7.1	Kronecker decomposition on each component of the GPT architecture .	99

List of Tables

4.1	Complexity of the sparse-dense Sylvester solver	60
6.1	Specification of the machine used in the experiments.	86
6.2	Running time comparisons between CG method and Lanczos-Sylvester method (seconds)	88
6.3	Total memory usage comparisons between CG method and Lanczos- Sylvester method (GB)	90
6.4	Comparison using Planck Data (with $m = 4$ sources), between the sparse-dense method introduced in Chapter 4 and the Lanczos-Sylvester method we proposed in 5.	91
6.5	Definition of functions listed in Figure 6.4.	92
7.1	Comparisons between KroGPT and two baseline models	100

List of Notations

n	The number of frequencies.
m	The number of sources.
N	The number of pixels in the all-sky map.
A	The mixing matrix in the CMB source separation problem.
b	The right-hand side vector of the linear system $Ax = b$.
s	The source vector at a single pixel.
$\mathcal{S}$	The map of a source.
y	The brightness of the sky maps at a single pixel.
$\mathcal{Y}$	The vector of all observations.
ϵ	The vector of independent measurement errors.
$\mathcal{E}$	The stacked vector of measurement errors.
Q	The block diagonal matrix derived from the intrinsic GMRF.
D	The matrix that encodes the neighborhood structure.
I_N	The N -dimensional identity matrix.
B	$A \otimes I_N$.
C	The precision matrix.
$\vec{T}$	The N -dimensional vector in construction of C .
$\vec{N}$	The second N -dimensional vector in construction of C .
P	The diagonal matrix that satisfies $Q = P \otimes D^2$.
μ	The mean vector solution of the linear system.
M	The reshaped vector $M = \text{reshape}(\mu, N, n)$.
Y	The reshaped vector $Y = \text{reshape}(y, N, n)$.

1 Introduction

Large-scale linear systems are fundamental to scientific computing, particularly in fields where complex data analysis is essential. As the dimension and volume of datasets grow, solving these systems efficiently becomes a significant challenge, requiring innovative algorithmic approaches. To address this, it is crucial to design algorithms that are (1) tailored to the specific characteristics of the problem at hand and (2) capable of producing approximate solutions when exact solutions become computationally infeasible. This research focuses on developing scalable linear solvers for such problems, particularly in the context of factor analysis models. A major motivating application for this work is one of the most important problems in cosmology: the observation of the Cosmic Microwave Background (CMB). Specifically, we tackle the source separation problem from all-sky maps, a critical task in studying the origins of the universe and providing further evidence for the Big Bang theory.

The CMB source separation problem presents unique challenges due to the vast amount of data involved and the need for efficient and accurate separation of mixed signals in the observed data. Traditional methods for solving these large-scale linear systems become impractical as the problem size increases, demanding new approaches that can efficiently scale with data size. In this dissertation, we propose novel matrix-free, Krylov subspace-based iterative methods that exploit the structure of the linear systems inherent to the CMB source separation problem. These methods allow us to approximate solutions to the source separation problem with high computational efficiency, significantly reducing both time and memory costs.

The key contribution of this research is the development of scalable, approximate solvers that can be applied to large-scale statistical problems, particularly in CMB data analysis. To achieve this, the dissertation presents two primary approaches: the conjugate gradient method and a Sylvester matrix-based method, both of which leverage the structure of the underlying problem to improve computational efficiency. The methods are rigorously analyzed in both theoretical and practical contexts, demonstrating their applicability not only to CMB data but also to other large-scale problems in scientific computing.

1.1 Research Aims

The primary objective of this thesis is to address the challenges posed by large-scale, sparse linear systems that arise in various scientific computing applications, with a particular focus on cosmological data analysis. The research is structured around three key aims:

1. *Provide a theoretical understanding for Krylov subspace methods in solving large-sparse linear systems:* A central goal of this thesis is to build a solid theoretical foundation for the use of Krylov subspace methods, such as the Conjugate Gradient and Lanczos processes, in the context of large-scale, sparse linear systems. These systems often emerge in complex statistical algorithms and pose significant computational challenges as data size and dimension grow. While Krylov subspace methods are well-established for solving large-sparse linear systems, this research focuses on applying these methods to systems with a unique structure, specifically those involving matrices with Kronecker products. The aim is to extend existing Krylov subspace techniques by exploiting the Kronecker structure inherent in the problem, thereby improving computational efficiency and scalability. By examining these methods in detail, we aim to provide a deeper understanding of how they exploit the inherent structure of sparse linear systems, enabling faster convergence and reduced computational cost compared to

traditional direct methods.

2. *Develop matrix-free algorithms to solve such systems, with application in the CMB source separation problem:* Another core aim is the development of matrix-free algorithms that can effectively solve large-scale linear systems without the need to explicitly store the system matrix. This is particularly important in applications like Cosmic Microwave Background (CMB) source separation, where the size of the data makes conventional methods infeasible. By leveraging Krylov subspace techniques, we propose novel algorithms that are specifically designed to take advantage of the structure of the CMB data, including Kronecker product structures and Sylvester matrix equations. These algorithms are designed to be both scalable and efficient, significantly reducing memory usage and computational time, while ensuring that approximation errors remain within acceptable bounds. The ability to solve such problems matrix-free is crucial for real-world applications involving massive datasets, such as those encountered in cosmology.
3. *Demonstrate the applicability of the proposed methods in similar problems:* Beyond their application to the CMB source separation problem, a major goal of this research is to demonstrate the versatility and broader applicability of the developed Krylov subspace-based algorithms. The techniques introduced in this thesis are tested on a variety of large-scale problems, including different formulations of CMB data analysis, such as polarization data, and other high-dimensional statistical models. Additionally, we explore the application of these methods to problems in neural network compression, where large, sparse matrices arise and efficient, scalable algorithms are crucial. By demonstrating the efficacy of these methods across multiple domains, this research seeks to establish Krylov subspace methods as a robust tool for solving large-scale statistical problems, offering scalable and adaptable solutions to other scientific and engineering challenges.

Through these aims, this thesis contributes to both the theoretical and practical aspects

of solving large-scale linear systems in scientific computing, offering novel algorithmic solutions with significant potential for broader application.

1.2 Methodology

The research methodology employed in this thesis is centered around the development, analysis, and application of Krylov subspace-based iterative methods for solving large-scale, sparse linear systems. The work is divided into several key stages, each corresponding to the development of theoretical foundations, algorithm design, and empirical validation.

The research begins by formulating the Cosmic Microwave Background (CMB) source separation problem as a large-scale linear system. This involves constructing the linear mixture model and partitioning the all-sky map data to capture the problem's structure. The first step in the methodology is to develop a theoretical understanding of Krylov subspace methods, focusing on the Conjugate Gradient method and the Lanczos process. These methods are analyzed for their efficiency in solving sparse systems, and the unique structural properties of the problem, such as Kronecker product structures and intrinsic Gaussian Markov random fields, are explored in detail.

The second stage involves the design and implementation of matrix-free iterative solvers tailored to the CMB source separation problem. Two key approaches are proposed: one based on the Conjugate Gradient method and another using a Sylvester matrix interpretation. Both methods exploit the problem's structure to avoid the explicit formation of large matrices, thereby significantly reducing memory and computational costs. The research focuses on integrating these matrix-free solvers into a practical algorithmic framework, making them suitable for large-scale applications.

The methodology further extends to adapting the Sylvester-based solver using a Lanczos-based approach. This step involves leveraging block Krylov subspaces, which allow for a block generalization of Krylov methods, making the algorithm scalable to larger datasets. The block Lanczos process is developed to solve symmetric matrices arising

from the problem, offering additional computational savings. These developments are analyzed both theoretically and through empirical testing, ensuring their applicability to the CMB source separation problem.

Extensive experimentation is conducted to validate the performance of the proposed methods. Three different implementations—(1) a baseline sparse-dense Sylvester method, (2) the Conjugate Gradient approach, and (3) the Lanczos-Sylvester method—are applied to both simulated datasets and actual Planck full-sky maps. The experimental design allows for a thorough comparison of time complexity, memory usage, and accuracy. Additionally, code profiling is used to analyze the algorithms’ performance as the problem size scales, providing insights into the efficiency of the matrix-free implementations.

The final part of the methodology explores the generalizability of the proposed Krylov subspace methods beyond CMB data. The techniques are applied to other problem domains, such as CMB polarization data, which requires similar algorithmic adaptations but with additional cosmological parameters. Furthermore, the research investigates applying Kronecker product exploitation and matrix-free operations to compress large, sparse matrices in neural networks, demonstrating the broader applicability of the methods.

1.3 Thesis Structure

We conclude our introduction with a brief summary of the the material contained in the chapters of this thesis.

- *Chapter 2 - Literature Review:* Chapter 2 begins with a comprehensive review of the fundamental aspects of large, sparse linear systems and their applications across various domains. This provides the necessary background to understand the challenges posed by such systems. The chapter then transitions to the central motivation of this thesis: the Cosmic Microwave Background (CMB) source separation problem. The significance of this problem is discussed in detail, along

with an overview of recent advancements in the field. Next, the linear mixture model, which is used to formulate the source separation problem, is introduced. Various approaches to blind source separation are reviewed, with particular emphasis on variants of Independent Component Analysis (ICA). Following this, a section is dedicated to Bayesian inference, the primary method used to model the linear systems in this research. Finally, the chapter presents an introduction to Krylov Subspace methods, with a focus on the Conjugate Gradient method and the Lanczos process, both of which form the foundation of the novel approaches developed later in the thesis.

- *Chapter 3 - Krylov Subspace Solver for Scalable CMB Source Separation:* We present a detailed formulation of the Cosmic Microwave Background (CMB) source separation problem, starting with the partitioning method for the all-sky map and the construction of the mixing matrix. The linear systems are modeled using an intrinsic Gaussian Markov random field, which captures the spatial dependencies in the data. Next, we describe the key components of our matrix-free Krylov subspace-based iterative solver, highlighting how it leverages the special structure inherent in the linear system to enhance computational efficiency. A theoretical analysis is provided to demonstrate how the algorithm exploits these structural properties to improve performance. Finally, we discuss the advantages and limitations of the method within the specific context of the CMB source separation problem.
- *Chapter 4 - Linear Solver with Sylvester Interpretation:* We introduce the development of a linear solver based on the Sylvester interpretation of the constructed linear system. By reformulating the original problem as a Sylvester matrix equation, we are able to exploit additional structures present in the data. This leads to a sparse-dense Sylvester equation, for which we review several solution techniques. We adopt the solution proposed by Benner et al. [2011] as the baseline model for solving the CMB source separation problem using the Sylvester approach. The chapter includes a detailed analysis of the solution’s computational

complexity and its application to the problem.

- *Chapter 5 - Lanczos-Sylvester Solver for Scalable Statistical Algorithms:* Chapter 5 presents the primary contribution of this thesis: a matrix-free, Lanczos-based iterative method that leverages the Sylvester formulation of the problem. This method capitalizes on the fact that equations with a Sylvester structure can be efficiently handled using block Krylov subspaces, which benefit from a block generalization of the shift-invariance property inherent to Krylov subspaces. The chapter demonstrates the application of the block Krylov approach to large-scale problems and explores block Lanczos-based methods specifically for symmetric matrices. A theoretical analysis of the proposed algorithm is provided, along with a discussion on how to enhance its computational efficiency in practical applications.
- *Chapter 6 - Experimental Results:* We present the numerical results that demonstrate the viability of both proposed approaches for the underlying application problem. We implement and compare three methods: a baseline sparse-dense Sylvester approach, the conjugate gradient method, and the Lanczos-Sylvester method. These implementations are tested on both simulated datasets of varying sizes—allowing for evaluation of actual error—and real Planck full-sky maps. The results are analyzed and compared based on time cost, memory usage, and accuracy. Additionally, code profiling is used to examine how computational requirements for different aspects of the algorithms scale with problem size.
- *Chapter 7 - Conclusions and Future Works:* Chapter 7 concludes the thesis by discussing the contributions of the proposed methods in solving the CMB source separation problem, along with their potential for generalization to other applications. We highlight several attempts to extend these methods to different research areas. The first attempt applies the algorithm to CMB polarization data, where a similar approach is required but with the inclusion of specific cosmological parameters. The second explores the use of Kronecker product

exploitation and matrix-free operations for compressing large, sparse matrices in neural networks. These applications demonstrate the versatility and scalability of the developed algorithms beyond the initial problem domain.

2 Literature Review

Large, sparse linear systems of equations appear in most applications of computational statistics; of these, partial differential equations (PDEs) are among the most popular [Cockayne et al., 2018]. Discretization of PDEs with the finite element method (FEM) or with the finite difference method (FDM) yield large, sparse linear systems which can be highly ill-conditioned. Even in the case that the original problem is nonlinear, linearization by a Newton-type method also leads to a linear problem [Gutknecht, 2007]. When working with an enormous number of data points or with a high-dimensional state space, the modeled linear system can become excessively complex and require a tailored algorithm to be solved practically.

Another significant area that often employs the solution of a linear system is the computation with Gaussian models. In particular, such system arises from modeling the symmetric positive definite covariance or precision matrix. For example, in Gaussian Markov Random Fields [Rue and Held, 2005a], the precision matrix Σ^{-1} is modeled. In many cases, the neighborhoods specifying conditional independence are small, leads to the sparseness of the matrix. An efficient solver must exploit the sparseness of either covariance matrix Σ or precision matrix Σ^{-1} to allow an efficient construction of the solution.

On the other hand, there are many cases in practice where the observed data is a mixture of mostly non-Gaussian independent sources. In such cases, the independent component analysis (ICA) has become one of the most ubiquitous method. The ICA model expresses a vector of observation $X \in R^r$ as a linear transformation of a linear

vector $S \in R^s$ with independent components:

$$X = AS \tag{2.1}$$

where $A \in R^{r \times s}$ is a deterministic mixing matrix. When the goal is to recover the unknown mixing matrix, it is called *feature extraction* [Hyvärinen and Oja, 1998]. In our work, we assume the matrix A is known and focus on recovering the original sources, also known as *source separation* problem. Traditionally, most ICA methods relies on the non-Gaussianity assumption of the source S to ensure that A is identifiable. The *FastICA* algorithms by Hyvarinen [1999] is probably the most widely known approach in this direction, which iteratively finds the one-dimensional projections that maximizes kurtosis of the data. In a more recent work, Abrahamsen and Rigollet [2018] attempt to solve the Gaussian case by imposing structure on the matrix A .

The motivating application of our work comes from solving a general case of *noisy ICA* for the Cosmic Microwave Background (CMB) source separation problem, where we allow independent additive noise $\varepsilon \sim \mathcal{N}(0, C_\sigma)$ for a diagonal non-negative matrix $C_\sigma = \text{diag}(\sigma_1^2, \dots, \sigma_r^2)$. In this chapter, we describe the CMB source separation problem from the first exploration to the most recent experimental results. After that, we review Bayesian inference, the learning approach that we use to model the linear systems arise in the context of CMB. The chapter concludes with the introduction of Krylov Subspace methods, focusing on the algorithms that later used to build the linear solvers in this research.

2.1 Cosmic Microwave Background

2.1.1 CMB Component Separation

The Cosmic Microwave Background (CMB), or relic radiation, is microwave radiation that fills all space in the observable universe. It provides a snapshot of the universe at the time of recombination. It is therefore a unique probe for cosmologists since it

carries crucial information about the dawn of the universe and the evolution to its current state [Bobin, J. et al., 2016]. Since the first discovery of the CMB radiation signal in 1965 [Dicke et al.], three notable satellite missions have been conducted, namely COBE (Cosmic Background Explorer) [Mather, 1982], WMAP (The Wilkinson Microwave Anisotropy Probe) [Bennett et al., 2003], and Planck [Aghanim et al., 2020], producing increasingly accurate pictures of the CMB and allowing researchers to study the information about the early stage of the universe.

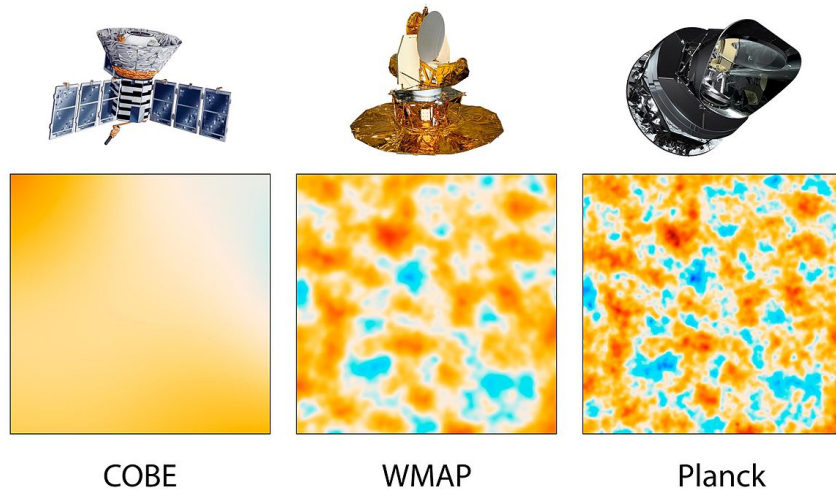


Figure 2.1: Comparison of CMB results from COBE, WMAP and Planck. Image courtesy NASA/JPL-Caltech/ESA.

The detector sensitivities of modern experiments have allowed one to measure fluctuations of the CMB temperature with such a sensitivity that contamination of the data by astrophysical foreground radiations, rather than by instrumental noise, is becoming the major source of limitation [Delabrouille and Cardoso, 2007]. In the context of CMB experiments, signals and images contain contributions from several components or *sources*. There are point sources that are relatively easy to identify and remove from the data, including stars and galaxies. The remaining sources are diffuse sources, which are meaningful for astrophysical studies, such as synchrotron radiation, galactic dust, and free-free emission. The study of the CMB radiation signal or any other diffuse source requires separating the source of interest from all other components present in the data. More generally, the objective of component separation is to estimate a set of parameters which describe the target component. This set of parameters depends

significantly on the model assumed for the components, therefore the model plays an integral role in an efficient component separation.

2.1.2 The Linear Mixture Model

The most ubiquitous model for source separation of CMB observations is the linear mixture model. In this model, all components are assumed to have an emission which can be decomposed as the product of a spatial template independent of the frequency of observations and a spectral emission law that does not depend on the pixel [Delabrouille and Cardoso, 2007]. The total emission at frequency v , at pixel p , of a source j can be written as:

$$x_j(v, p) = a(v)s_j(p)$$

where $s_j(p)$ represents the spatial template (or intensity) of source j at pixel p , and $a(v)$ is the spectral emission law at frequency v . The observation of a particular detector i is then:

$$y_i(p) = \sum_j x_j(v_i, p) + \epsilon_i(p)$$

where $\epsilon_i(p)$ is contribution of noise for the detector. For a given set of detectors, this can be formulated in a matrix-vector form as:

$$y(p) = As(p) + \epsilon(p) \tag{2.2}$$

where $y(p)$ represents the full sky map observed in different frequencies with corresponding detectors, $s(p)$ are the unobserved components (one template map –for each astrophysical source) and $\epsilon(p)$ denotes the measurement errors. The mixing matrix A is a $n \times m$ matrix of weights, with n is number of frequencies (or detectors) and m is the number of diffuse sources.

The solution for source separation consists in inverting the linear system in Equation 2.2. In the simplest case, where A is square and non singular, without any additional

information, the inversion is:

$$W = A^{-1}$$

and the source is obtained by:

$$\hat{s} = A^{-1}y = s + A^{-1}\epsilon$$

More generally, if the number of observations is significantly larger than the number of components, and the signal-to-noise level is unknown, the pseudo-inverse [Penrose, 1955] $A^\dagger$ is employed:

Definition 1. Let $\mathbb{K}$ denotes one of the fields of real or complex numbers. For $A \in \mathbb{K}^{m \times n}$, a pseudo-inverse of A is defined as a matrix $A^\dagger \in \mathbb{K}^{n \times m}$ satisfying all of the following four criteria, known as the Moore–Penrose conditions:

1. $AA^\dagger$ need not be the general identity matrix, but it maps all column vectors of A to themselves:

$$AA^\dagger A = A$$

2. $A^\dagger$ acts as a weak inverse:

$$A^\dagger AA^\dagger = A^\dagger$$

3. $AA^\dagger$ is Hermitian:

$$(AA^\dagger)^* = AA^\dagger$$

4. $A^\dagger A$ is Hermitian:

$$(A^\dagger A)^* = A^\dagger A$$

Using the pseudo-inverse defined above, we have:

$$W = [A^\dagger A]^{-1} A^\dagger$$

That leads to:

$$\hat{s} = [A^\dagger A]^{-1} A^\dagger y = s + [A^\dagger A]^{-1} A^\dagger \epsilon$$

When the noise is introduced, with information about its second order statistics, such as the noise correlation matrix R , assuming the noise for each detector is a realization of a random Gaussian field, the generalized least square (GLS) solution of the system is:

$$W = [A^\dagger R^{-1} A]^{-1} A^\dagger R^{-1}$$

and the source is:

$$\hat{s} = [A^\dagger R^{-1} A]^{-1} A^\dagger R^{-1} y = s + [A^\dagger R^{-1} A]^{-1} A^\dagger R^{-1} \epsilon$$

The solution is unbiased and yield the minimum variance error map for a deterministic signal [Delabrouille and Cardoso, 2007], therefore it is the best linear solution for the system in the case of large signal-to-noise ratio.

On the other hand, in most real-world datasets, the mixing matrix A is unknown a priori and must be learned from the data. In that case, the component separation problem can still be solved with the assumption that the components are statistically independent. This problem is often referred to as “blind source separation” and will be discussed in the sub-section 2.1.3.

2.1.3 Blind Source Separation

Blind source separation is a well-studied topic in the signal processing domain. With the ability to recover the components of a linear mixture model without prior knowledge about the mixing matrix and physical attributes of each source, the method can be applied to most multi-detector datasets, especially those with missing or imperfect information. Moreover, this allows the exploration of new sources, as one can process the data without knowing in advance which components are there. Finally, the as-

sumption of statistically independent attribute of the components is plausible in most physical systems, making the application of the blind source separation possible.

Independent Component Analysis (ICA) is a popular blind source separation method. In addition to the assumption of independent sources, ICA relies on another assumption that the target sources are non-Gaussian. Let $A^\dagger$ be the pseudo-inverse of the mixing matrix A , a single source s_i from the Equation 2.1 can be expressed as:

$$s_i = A_i^\dagger X$$

The basic concept of ICA is to compute $A_i^\dagger$ and then s_i by maximizing the non-Gaussianity of $A_i^\dagger X$ using measures such as kurtosis or negentropy. FastICA, proposed by Hyvarinen [1999] based on a negentropy approximation for a robust non-Gaussianity optimization, is the most popular algorithm for performing ICA. The authors also point out that since the sources are assumed to be uncorrelated, the optimization on all sources is trivial [Hyvärinen and Oja, 2000]. In the context of CMB source separation problem, FastICA is implemented by Maino et al. [2002] with the objective of recovering both spatial pattern and the frequency scalings of the emissions from statistically independent astrophysical processes along the line-of-sight using multi-frequency observations. The algorithm requires no prior knowledge except for the statistical independence of the components and the assumption that all, except possibly one, are non-Gaussian. These assumptions hold in the case of Gaussian CMB fluctuations and non-Gaussian foreground elements. In another approach, Delabrouille and Cardoso [2007] developed Spectral Matching ICA (SMICA) in which they fix the CMB's Gaussianity issue by trading non-Gaussianity for spectral distinction between sources. More specifically, SMICA computes S and A via maximizing a spectral matching criterion exhibited by the CMB and foreground sources [Wagner-Carena et al., 2020]. The method is used by Planck Collaboration I [Aghanim et al., 2020] to extract CMB component as well as astrophysical components, as shown in Figure 2.2.

Figure 2.2 illustrates the results of component separation using SMICA methods, dis-

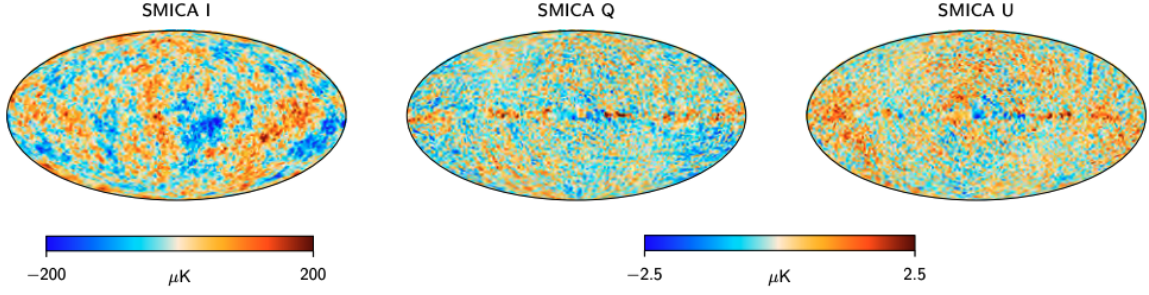


Figure 2.2: Component-separated maps at 80' resolution using SMICA methods. From left to right: Stokes I (temperature), Stokes Q and Stokes U (polarization).

playing the three Stokes parameters that characterize CMB radiation. Stokes I represents the temperature (total intensity), while Stokes Q and U together describe linear polarization patterns. These polarization components contain complementary cosmological information to temperature data arising from Thomson scattering during recombination. Although the current work focuses on temperature-only source separation (Stokes I), the methods developed in this thesis have potential for extension to polarization data (Stokes Q and U), which we identify as a promising direction for future research in Chapter 7.

2.2 Bayesian Inference

Bayesian inference is a fundamental framework in statistical analysis, offering a way to combine prior knowledge with observed data to update beliefs about unknown parameters. Unlike frequentist inference, which treats parameters as fixed and estimates them purely from data, Bayesian inference treats parameters as random variables, allowing for direct probability statements about them. This section provides an overview of Bayesian inference, comparing it to the frequentist approach, and introduces key concepts such as the likelihood function, prior distribution, and posterior distribution. It also highlights popular Bayesian models, particularly the ones relevant to this thesis.

Bayesian inference has its roots in the 18th century, with the work of Thomas Bayes (1763) on conditional probabilities and Pierre-Simon Laplace (1774), who developed

the theory of 'inverse probability'. Bayes introduced the concept of updating beliefs based on new evidence, while Laplace formalized the principle of insufficient reason, assuming uniform priors when no additional information was available. Despite its early development, Bayesian inference fell out of favor in the early twentieth century due to the rise of frequentist methods, which were popularized by statisticians such as Fisher [1922] and Neyman [1937].

Frequentist statistics dominated much of the twentieth century, interpreting probability as the long-term frequency of events and treating parameters as fixed, unknown quantities. In contrast, Bayesian inference views probability as a measure of belief or uncertainty about an event and treats parameters as random variables with associated probability distributions. The resurgence of Bayesian methods in the latter half of the twentieth century can be attributed to the work of Jeffreys [1939], Savage [1954], and the development of computational techniques such as Markov chain Monte Carlo (MCMC) in the 1990s, which made Bayesian inference feasible for complex, high-dimensional problems [Robert and Casella, 1999]. Today, Bayesian methods are widely used in many disciplines, including machine learning [Murphy, 2012], signal processing [Kay, 1998], and cosmology [Trotta, 2008].

The key philosophical distinction between Bayesian and frequentist inference lies in their treatment of parameters. In the frequentist paradigm, the parameters are fixed but unknown quantities, and the goal is to estimate them using data. Probability is interpreted as the frequency of an event occurring over repeated trials. Frequentist methods rely on tools such as confidence intervals and hypothesis tests, which provide long-term guarantees about the frequency of certain outcomes [Fisher, 1922] [Neyman, 1937]. However, these methods do not provide direct probability statements about the parameters themselves.

In contrast, Bayesian inference treats parameters as random variables and uses probability distributions to represent uncertainty about their values. The Bayesian framework naturally accommodates uncertainty in parameter estimates, providing posterior

distributions that quantify this uncertainty based on both prior beliefs and observed data. This direct probabilistic interpretation offers a more flexible and intuitive approach, especially in complex or hierarchical models where frequentist methods may struggle to incorporate all sources of variability.

Consider observations $y = \{y_1, \dots, y_n\}$ and a probability model which it is believed generated the data, having density function $f_Y(y; \psi)$, where the dependence on the parameter(s) ψ is made explicit. Without loss of generality, ψ may be a vector of parameters. Bayesian inference considers ψ to be the unknown realization of a random variable, so that one can consider the joint density of the random variables Y and Ψ :

$$f_{Y,\Psi}(y, \psi) = f_{Y|\Psi}(y|\psi)f_{\Psi}(\psi)$$

where simply $f_{Y|\Psi}(y|\psi) := f_Y(y; \psi)$.

Thus, given data y , by application of Bayes' Theorem one can directly encapsulate the uncertainty in ψ as:

$$f_{\Psi|Y}(\psi|y) = \frac{f_Y(y; \psi)f_{\Psi}(\psi)}{\int_{\Omega} f_Y(y; \psi)f_{\Psi}(d\psi)} \propto f_Y(y; \psi)f_{\Psi}(\psi) \quad (2.3)$$

where Ω is the sample space of Ψ . Proportionality follows from noticing that the denominator is independent of ψ due to the integral. Equation (3.2) is referred to as the posterior distribution of ψ and encapsulates all knowledge about the unknown parameters. [Aslett, 2012]

In practice, the integral in the denominator of Equation 2.3 — the normalizing constant — is often analytically intractable. This fact impeded widespread adoption of Bayesian inference until the advent of the computer and the development of computational techniques like Markov chain Monte Carlo (MCMC). Thus, the popularity of the Bayesian approach only really erupted in the applied statistics community as recently as the 1990s.

2.2.1 Likelihood Function

In Bayesian inference, the likelihood function plays a critical role in connecting the data to the model. For a set of observations $y = \{y_1, \dots, y_n\}$ and a model with parameters Ψ , the likelihood function $L(\psi; y) := f_Y(y; \psi)$ represents the probability of the data given the parameters. While the likelihood is central to both frequentist and Bayesian inference, its role differs between the two approaches.

In frequentist methods, the likelihood is used to derive point estimates for the parameters, such as the maximum likelihood estimate (MLE) [Fisher, 1922]. The MLE is obtained by maximizing the likelihood function with respect to the parameters, but this approach does not account for uncertainty in the estimates. Bayesian inference, on the other hand, combines the likelihood with a prior distribution to produce a posterior distribution, which fully captures the uncertainty about the parameters.

The likelihood function is also critical in the context of the *likelihood principle*, which states that all relevant information about the parameters is contained in the likelihood function. Bayesian inference adheres to this principle, as the posterior distribution depends solely on the likelihood and the prior, making the likelihood function a fundamental component of the Bayesian framework [Berger and Wolpert, 1988].

2.2.2 Prior Distribution

The prior distribution is a key element that distinguishes Bayesian inference from frequentist methods. It reflects the analyst's beliefs or knowledge about the parameters before observing the data, and can take many forms depending on the application. Formally, the prior distribution is denoted as $f_\psi(\Psi)$ and is combined with the likelihood to generate the posterior distribution. The choice of prior can be informed by expert knowledge, previous research, or chosen to be non-informative if little prior information is available [Robert and Casella, 1999].

Priors can be classified into several categories:

- **Informative priors:** These are based on prior knowledge or expert opinion, and are often used when reliable information is available before observing the data [Hagan and West, 2010]. For example, in cosmology, informative priors may reflect known constraints on certain physical parameters.
- **Non-informative or vague priors:** These priors are designed to have minimal influence on the posterior distribution, allowing the data to drive the inference. Common choices include uniform or improper priors [Jeffreys, 1939].
- **Conjugate priors:** These are chosen for mathematical convenience, as they result in posterior distributions that belong to the same family as the prior. For example, the Gaussian prior is conjugate to the Gaussian likelihood, making it easy to compute the posterior analytically [Bernardo and Smith, 2009].

In this thesis, the prior distribution plays an important role in modeling the spatial structure of the CMB data. We use a Gaussian Markov random field (GMRF) as the prior for the linear systems in the CMB source separation problem. The GMRF prior captures the spatial correlations in the CMB data, ensuring that the separated components exhibit the expected smoothness across the sky [Rue and Held, 2005b].

2.2.3 Posterior Distribution

The posterior distribution is the cornerstone of Bayesian inference, representing the updated beliefs about the parameters after incorporating both prior knowledge and the observed data. According to Bayes' theorem, the posterior distribution is proportional to the product of the likelihood and the prior:

$$f_{\Psi|Y}(\psi|y) \propto L(\psi; y)f_{\Psi}(\psi)$$

This formulation allows the analyst to update their beliefs in light of new evidence. The posterior distribution encapsulates all the information available about the parameters, providing a full probabilistic description of the uncertainty surrounding them [Gelman

et al., 2013]. The posterior can be used to compute various quantities of interest, such as point estimates (e.g., posterior mean or median), credible intervals (the Bayesian analogue of confidence intervals), and predictions for future data.

In practice, computing the posterior distribution often requires sophisticated numerical methods due to the complexity of the models and the intractability of the integrals involved. Techniques such as Markov chain Monte Carlo (MCMC) have been developed to approximate the posterior distribution in high-dimensional spaces [Robert and Casella, 1999]. In this research, we use MCMC methods to approximate the posterior distribution for the CMB source separation problem, ensuring that the computational challenges posed by the large-scale data are effectively managed.

2.2.4 Markov Chain Monte Carlo

Markov Chain Monte Carlo (MCMC) methods are a class of algorithms used to sample from probability distributions, particularly when direct sampling is infeasible due to the complexity of the model or the high dimensionality of the parameter space. MCMC plays a critical role in Bayesian inference by enabling the approximation of posterior distributions when analytical solutions are not available. In this section, we provide a theoretical foundation for MCMC, introduce key algorithms such as the Metropolis-Hastings and Gibbs sampling, and discuss the application of MCMC in Gaussian Markov Random Fields (GMRFs), which are particularly relevant to the research in this thesis.

MCMC methods are based on constructing a Markov chain whose stationary distribution is the target distribution from which we wish to sample. A Markov chain is a sequence of random variables where the probability of transitioning to the next state depends only on the current state, not on the sequence of events that preceded it. Formally, a Markov chain $\{X_t\}_{t=1}^{\infty}$ satisfies the Markov property:

$$P(X_{t+1}|X_1, X_2, \dots, X_t) = P(X_{t+1}|X_t)$$

In the context of Bayesian inference, the target distribution is the posterior distribution $P(\theta|y)$, where θ represents the parameters of the model, and y is the observed data. MCMC allows us to generate samples from $P(\theta|y)$ by constructing a Markov chain that eventually converges to this distribution. Once the chain has converged, the samples can be used to estimate posterior quantities of interest, such as means, variances, or credible intervals.

A key concept in MCMC is ergodicity, which ensures that the Markov chain will converge to the target distribution regardless of its initial state. Ergodicity requires that the chain be both irreducible (it is possible to get from any state to any other state) and aperiodic (the chain does not cycle through states in a deterministic pattern). Under these conditions, the chain will eventually “forget” its starting point and produce samples that accurately represent the target distribution [Robert and Casella, 1999].

Two of the most commonly used MCMC algorithms are the Metropolis-Hastings algorithm and Gibbs sampling, both of which are highly applicable in Bayesian inference for large-scale problems.

The Metropolis-Hastings Algorithm

The Metropolis-Hastings (MH) algorithm is a general-purpose MCMC method that generates a sequence of samples from the target distribution by proposing moves in the parameter space and accepting or rejecting them based on an acceptance probability. The algorithm works as follows:

1. **Initialization:** Start with an initial state θ_0 for the parameters.
2. **Proposal step:** Propose a new state θ^* from a proposal distribution $q(\theta^*|\theta_t)$, where θ_t is the current state.
3. **Acceptance step:** Compute the acceptance probability:

$$\alpha = \min \left(1, \frac{P(\theta^*|y)q(\theta_t|\theta^*)}{P(\theta_t|y)q(\theta^*|\theta_t)} \right)$$

4. **Update:** With probability α , accept the proposed state θ^* , setting $\theta_{t+1} = \theta^*$.
Otherwise, reject the proposal and keep the current state, $\theta_{t+1} = \theta_t$.

The MH algorithm is flexible and can be used for a wide range of target distributions, as long as the proposal distribution $q(\theta^*|\theta_t)$ is chosen appropriately. A common choice is the Gaussian proposal, which generates new states by adding a small random perturbation to the current state. One of the strengths of MH is that it can be applied to models with complex or multi-modal posterior distributions, although its efficiency depends on the choice of the proposal distribution.

Gibbs Sampling

Gibbs sampling is a special case of the Metropolis-Hastings algorithm that is particularly efficient when the target distribution has a structure that allows conditional distributions to be sampled directly. In Gibbs sampling, we update one parameter at a time, conditioning on the current values of the other parameters. The algorithm proceeds as follows:

1. **Initialization:** Start with initial values for all parameters $\theta = (\theta_1, \theta_2, \dots, \theta_d)$.
2. **Iterative updates:** For each parameter θ_i , sample from the conditional distribution $P(\theta_i|\theta_{-i}, y)$, where θ_{-i} represents all parameters except θ_i .
3. **Repeat:** Cycle through all the parameters until the chain has converged to the target distribution.

Gibbs sampling is particularly well-suited for hierarchical Bayesian models, where the full conditional distributions are often easy to derive. It is widely used in applications such as image reconstruction, latent variable models, and spatial statistics. [Casella and George, 1992]

Applications of MCMC in Gaussian Markov Random Fields

A specific application of MCMC in this thesis is its use in Gaussian Markov Random Fields (GMRFs), which are a class of probabilistic models for spatial data. A GMRF is

defined by a set of random variables $x = \{x_1, x_2, \dots, x_n\}$, each associated with a node in a graph, where the dependencies between variables are encoded in the graph structure. The joint distribution of the variables is Gaussian, but the conditional independence structure is represented by the graph, making GMRFs particularly useful for modeling spatial or temporal dependencies.

Formally, a GMRF is defined by a precision matrix Q , which is the inverse of the covariance matrix Σ of the joint Gaussian distribution. The key property of GMRFs is that the precision matrix is sparse, with non-zero entries corresponding to neighboring nodes in the graph. This sparsity allows for efficient computation of conditional distributions, which is a significant advantage when dealing with large-scale data.

The joint probability density of a GMRF can be written as:

$$P(x) \propto \exp\left(-\frac{1}{2}x^T Q x\right)$$

In Bayesian inference, GMRFs are often used as prior distributions for spatially structured data. For example, in the context of the Cosmic Microwave Background (CMB) source separation problem, GMRFs can be used to model the spatial correlations in the CMB data, allowing for the incorporation of prior knowledge about the smoothness of the underlying signal. The precision matrix encodes the spatial relationships between different points on the sky, and MCMC methods are used to sample from the posterior distribution of the signal given the observed data.

MCMC is particularly useful for inference in GMRF models because it enables efficient sampling from the posterior distribution, even when the model involves large numbers of spatially dependent variables. The sparsity of the precision matrix in GMRFs allows for computationally efficient updates in Gibbs sampling, as each variable is conditionally independent of all others given its neighbors. This property is exploited in spatial models and has been applied extensively in fields such as image analysis, geostatistics, and environmental modeling [Rue and Held, 2005b].

In this thesis, we use MCMC to sample from the posterior distribution of GMRF-based models for the CMB source separation problem. The GMRF prior captures the spatial dependencies in the CMB map, and the MCMC algorithm ensures that we can explore the posterior distribution efficiently, even for large-scale datasets. The combination of GMRFs and MCMC methods provides a powerful framework for handling the complex spatial structure of the CMB data while maintaining computational efficiency.

Markov Chain Monte Carlo methods provide a flexible and powerful framework for sampling from complex probability distributions, making them indispensable in Bayesian inference. The Metropolis-Hastings algorithm and Gibbs sampling are two key MCMC methods that have been widely used across a variety of applications. In this thesis, MCMC plays a crucial role in the inference of Gaussian Markov Random Fields, which are used to model spatial dependencies in the CMB source separation problem. The combination of MCMC and GMRFs allows for efficient exploration of the posterior distribution in large-scale, spatially structured data, making it a vital tool in modern cosmological data analysis and other applications.

2.3 Krylov Subspace Methods

2.3.1 Iterative Methods

Large, sparse linear systems are often solved by either *direct methods* or *iterative methods*. Most direct methods are variations of the Gaussian elimination, in which matrix A is first factorized as the product of easy-to-solve matrices (e.g. diagonal, permutation, triangular, orthogonal matrices) [Moreau, 2018]. For example, LU decomposition methods involve factorizing A into the product of two triangular matrices: L (lower) and U (upper). These direct solvers have been optimized for many years, either by developing strategies for permuting equations and unknowns to guarantee a stable LU decomposition and small fill-in in the triangular factors or by organizing the computation to utilize the hardware architecture [Gutknecht, 2007]. With the rapid

development of computing hardware in recent years, supercomputers nowadays can combine the computing power of thousands of nodes and process large amount of data in optimal ways. That makes the solution of increasingly large and complex problems possible, but at the cost of an increased difficulty to preserve accuracy and reach efficiency in memory and processor use [Moreau, 2018]. When the size and complexity of the problem become sufficiently large for a particular hardware architecture, the computational and storage cost of a direct method may become excessive. In such cases, *iterative methods* where no factorization is required become better approaches to solve the problem.

The main idea of iterative methods is building a sequence of X^* that converges, at a given precision, toward the solution X of the given linear system. While direct methods are often more numerically stable, iterative methods are generally faster and require less computing power and storage. Most iterative linear solvers for large-scale linear systems nowadays belong to the *preconditioned Krylov subspace solvers* class [Gutknecht, 2007]. Krylov-based algorithms iteratively compute an approximation of solution X by transforming an n -dimensional vector space into a lower m -dimensional ($m \leq n$) subspace using matrix-vector multiplications without explicitly constructing A^{-1} . Usually, the approximation starts with a rough x_0 , then incrementally advances toward the solution by adding an approximation x_m ($x = x_0 + x_m$) in the Krylov subspace $\mathcal{K}_m$:

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, \dots, A^m r_0\}, r_0 = b - Ax_0$$

where b is the right-hand side vector of the linear system $Ax = b$. Let $p(A)$ defines the minimal polynomial of the vector r_0 with respect to A , then the Krylov subspace $\mathcal{K}_m$ is of dimension m if and only if the degree of $p(A)$ (denoted by $\deg(p(A))$) is not less than m : $m \leq \deg(p(A))$ Saad [2003a]. Note that the complexity of Krylov subspace method is $O(m^2n)$ in time and $O(mn)$ in memory, hence the method is efficient only when $\deg(p(A))$ is small, meaning matrix A is sparse.

The special feature that make Krylov subspace methods powerful for solving $Ax = b$ ($A \in R^{N \times N}$) is that for any vector $y \in R^N$, we can compute Ay without explicitly representing A . It means we can build efficient procedures to calculate high-dimensional products of matrices and vectors with A as an operator. This operation, often referred to as *matrix-vector product* is the foundation of many efficient linear solvers, including our matrix-free algorithms, which will be discussed in Chapter 3.

2.3.2 The Conjugate Gradient Method

While Krylov subspace methods are efficient in terms of computational and storage cost, it is important to ensure the constructed sequence converges to an accurate approximation of the solution. The Conjugate Gradient (CG) method introduced by Hestenes and Stiefel [1952] offers rapid convergence to the solution, particularly when the system is well preconditioned.

CG is an orthogonal projection method and satisfies a minimality conditions: the error is minimal in the energy norm (or A -norm) of the error vector $d = x - x_*$, $\|d\|_A = \sqrt{d^T A d}$ where x_* denotes the solution of the linear equation $Ax = b$ (A is symmetric positive definite or Hermitian positive definite).

The connection between solving the linear system $Ax = b$ and minimizing a function comes through discretization of the continuous problem. When we discretize continuous optimization problems in physics or engineering (such as minimizing potential energy or finding equilibrium states), we often arrive at a quadratic function of the form:

$$\Psi(x) = \frac{1}{2}x^T A x - b^T x + \gamma \quad (2.4)$$

where Ψ is convex with a unique minimum when A is positive definite. The constant γ is arbitrary and does not affect the location of the minimum.

This discretization transforms our original problem of solving $Ax = b$ into the equivalent problem of finding the minimum of $\Psi(x)$. To see this equivalence, we compute

the gradient of $\Psi(x)$: $\nabla\Psi(x) = Ax - b = -r$, where r is the residual corresponding to x . Therefore, x_* as a minimizer of Ψ means that $\nabla\Psi(x_*) = 0 \Leftrightarrow Ax_* - b = 0$ or $Ax_* = b$.

If we choose $\gamma = \frac{1}{2}b^T A^{-1}b$, we have:

$$\|d\|_A^2 = \|x - x_*\|_A^2 = \|Ax - b\|_{A^{-1}}^2 = \|r\|_{A^{-1}}^2 = 2\Psi(x) \quad (2.5)$$

That shows the minimizer x_* of the quadratic function Ψ defined in Equation 2.4 is the minimizer the energy norm of the error vector and is the solution of the linear system $Ax = b$ [Gutknecht, 2007].

The goal is to find the minimizer of Ψ by descending on the surface of representing ω following the direction of steepest descent. The search direction of the descent would dictate the complexity of the solution. The idea of a conjugate direction method is to choose search directions $\{v_0, v_1, \dots, v_n\}$ that are conjugate (A -orthogonal) to each other:

$$v_i^T A v_j = 0, \forall i \neq j$$

and in each search direction, take exactly one step such that the step size w_i is orthogonal to the proposed solution x at the direction:

$$x_{i+1} = x_i + v_i \omega_i, \text{ with } r_{i+1} = r_i - A v_i \omega_i$$

By definition, the chosen w_i is locally optimal on the search line $w \mapsto x_i + v_i w$. This means:

$$w_i = \frac{\langle r_i, v_i \rangle}{\langle v_i, A v_i \rangle}$$

It is also proven that the selection yields the best $x_{n+1} \in x_0 + \text{span}\{v_0, v_1, \dots, v_n\}$ [Golub and Van Loan, 1996]. If we choose $\text{span}\{v_0, v_1, \dots, v_n\} = \mathcal{K}_{n+1}(A, r_0)$, the conjugate direction method becomes a Krylov space solver, namely **Conjugate Gradient**

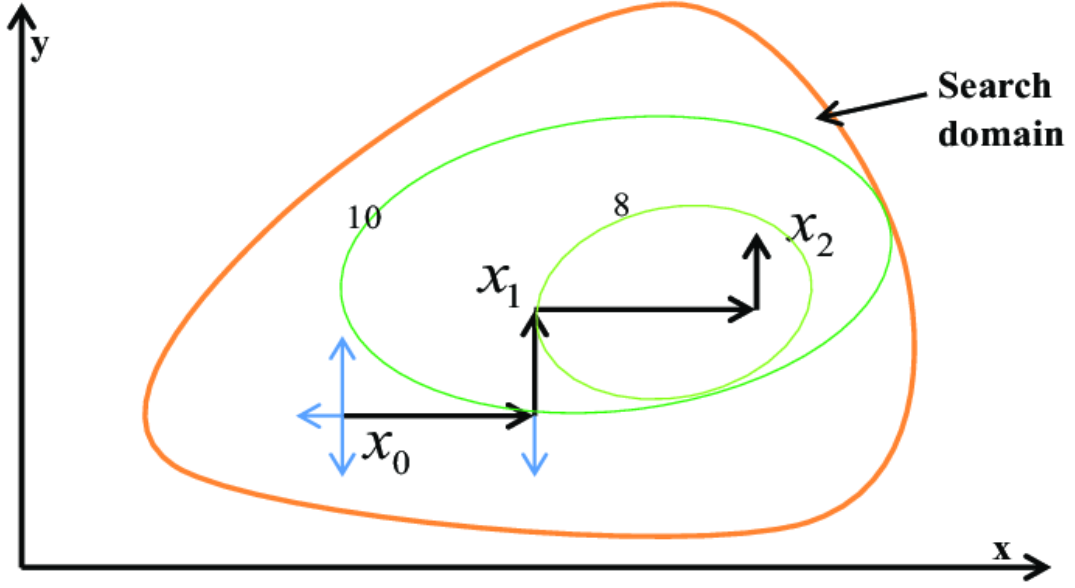


Figure 2.3: The principle of conjugate directions method (2-D). [Yang et al., 2019]

Method.

An implementation of the standard version of CG by Hestenes and Stiefel [1952] is shown in Algorithm 1.

Algorithm 1: Conjugate Gradient Method

Data: Matrix A , b , an initial approximation x_0 and termination threshold ϵ

Result: x_{i+1} as the solution of $Ax = b$

```

1  $v_0 \leftarrow b - Ax_0$ ;
2  $r_0 \leftarrow b - Ax_0$ ;
3  $\delta_0 \leftarrow \|r_0\|^2$ ;
4  $i = 0$ ;
5 while  $r_{i+1} \leq \epsilon$  do
6    $\delta'_i \leftarrow \|v_i\|_A^2$ ;
7    $w_i \leftarrow \frac{\delta_i}{\delta'_i}$ ;
8    $x_{i+1} \leftarrow x_i + v_i w_i$ ;
9    $r_{i+1} \leftarrow r_i - Aw_i w_i$ ;
10   $\delta_{i+1} \leftarrow \|r_{i+1}\|^2$ ;
11   $\psi_i \leftarrow \frac{-\delta_{i+1}}{\delta_i}$ ;
12   $v_{i+1} \leftarrow r_{i+1} - v_i \psi_i$ ;
13   $i \leftarrow i + 1$ ;
14 end
```

2.3.3 Lanczos-type Product Methods

The Conjugate Gradient Method can be generalized for non-symmetric systems. Multiplying both sides of $Ax = b$ by A^T we obtain the system of normal equation $A^T Ax = A^T b$, where the coefficient matrix is positive symmetric definite if A is full column-rank. That way, the orthogonality of the projection and the related minimality of the error is maintained. Eisenstat et al. [1983] developed generalized CG that depends on A rather than $A^T A$ called Generalized Conjugate Residual (GCR) algorithm. Saad and Schultz [1986a] later developed the Generalized Minimal Residual (GMRES) method which is mathematically equivalent to GCR but requires less storage and arithmetic operations.

In another approach, Fletcher [1976] introduced Bi-Conjugate Gradient (BiCG) method, replacing the orthogonal sequence of residuals by two mutually orthogonal sequences, at the price of no longer providing a minimization. BCG requires computing a matrix-vector product $Ap^{(k)}$ and a transpose product $A^T \tilde{p}^{(k)}$. In many applications, the transpose product may be impractical, for instance if the matrix is not formed explicitly and the regular product is only given in operation form. To resolve this issue, van der Vorst [1992] developed Bi-Conjugate Gradient Stabilized (BiCGSTAB) method that includes some local optimization and smoothing, and therefore the residual norm histories tend to be much smoother. Unlike the original BiCG method, BiCGSTAB does not require multiplication by the transpose of the system matrix. Further generalization of the method includes *BiCGSTAB(l)* for linear equations involving unsymmetric matrices with complex spectrum [Sleijpen, 1993] and GPBi-CG by Zhang [1997]. This family of methods has been called *Lanczos-type product methods*.

Lanczos-type product methods represent a powerful class of Krylov subspace solvers specifically designed to handle large, sparse, and often non-symmetric linear systems. These methods build on the foundational work of the Lanczos algorithm, originally designed for symmetric linear systems, and extend them to handle non-symmetric or indefinite systems.

The Lanczos Algorithm

The Lanczos algorithm, introduced by Lanczos [1950], is an iterative method for reducing a large, sparse, symmetric matrix A to a tridiagonal matrix T through a series of orthogonal projections onto a Krylov subspace. Given a symmetric matrix A and a starting vector v_1 , the Lanczos algorithm generates an orthonormal basis for the Krylov subspace:

$$K_m(A, v_1) = \text{span}\{v_1, Av_1, A^2v_1, \dots, A^{m-1}v_1\}$$

At each iteration, the matrix A is projected onto this subspace, yielding a tridiagonal matrix T_m that approximates A in the sense that the eigenvalues of T_m approximate the extremal eigenvalues of A . This method is particularly effective for finding a few extreme eigenvalues of large symmetric matrices.

The algorithm operates as follows:

1. **Initialization:** Choose an initial vector v_1 such that $\|v_1\| = 1$, and initialize $\beta_0 = 0$.
2. **Iteration:** For each step i , compute:

$$w = Av_i - \alpha_i v_i - \beta_{i-1} v_{i-1}$$

where $\alpha_i = v_i^T Av_i$ and $\beta_i = \|w\|$

3. **Orthonormalization:** Normalize the vector w to obtain $v_{i+1} = \frac{w}{\beta_i}$
4. **Tridiagonal Matrix Construction:** Construct the tridiagonal matrix T_m whose diagonal elements are the α_i 's and the off-diagonal elements are the β_i 's.

The Lanczos algorithm is computationally efficient because it only requires matrix-vector multiplications and does not require the full matrix to be stored in memory, making it well suited for large, sparse problems. Furthermore, the orthogonalization

procedure ensures that each new vector added to the Krylov subspace is orthogonal to the previous ones, ensuring numerical stability.

There is another class of Krylov subspace solvers, which is generalized to solve linear systems in coordinate space. The basic idea is to successively construct a basis of the Krylov space by combining the extension of the space with Gram-Schmidt orthogonalization or biorthogonalization, and update at each iteration the approximate solution of $Ax = b$ in coordinate space. Notable methods in this class include: MINRES, SYMMLQ for the symmetric Lanczos process, QMR for the non-symmetric Lanczos process, and GMRES for Arnoldi process [Gutknecht, 2007].

3 Krylov Subspace Solver for Scalable CMB Source Separation

3.1 Summary

As discussed in Chapter 2, the motivation behind the work in this thesis comes from the CMB source separation problem. We formulate the problem using the linear mixture model mentioned in 2.1.2 and develop linear solvers using iterative methods. We also provide a theoretical analysis to demonstrate the efficiency of the algorithm in terms of exploiting special structures appear in data. Finally, we discuss the advantages and disadvantages of the method in the specific problem setting.

3.2 Source Separation Model for All-Sky Maps

3.2.1 All-Sky Map Partition

The CMB source separation problem can be formulated as follows. There are n all-sky maps of microwave brightness that have been observed at frequencies $\nu_1, \dots, \nu_n$. These maps consist of a discrete grid of values over N pixels that partition the entire sky. In astronomy, the HEALPix partition into pixels of equal area is used [Górski et al., 2005].

HEALPix (Hierarchical Equal Area Iso Latitude Pixelation) is a multi-resolution partition of the sphere into elements or pixels of equal area, although not the same shape.

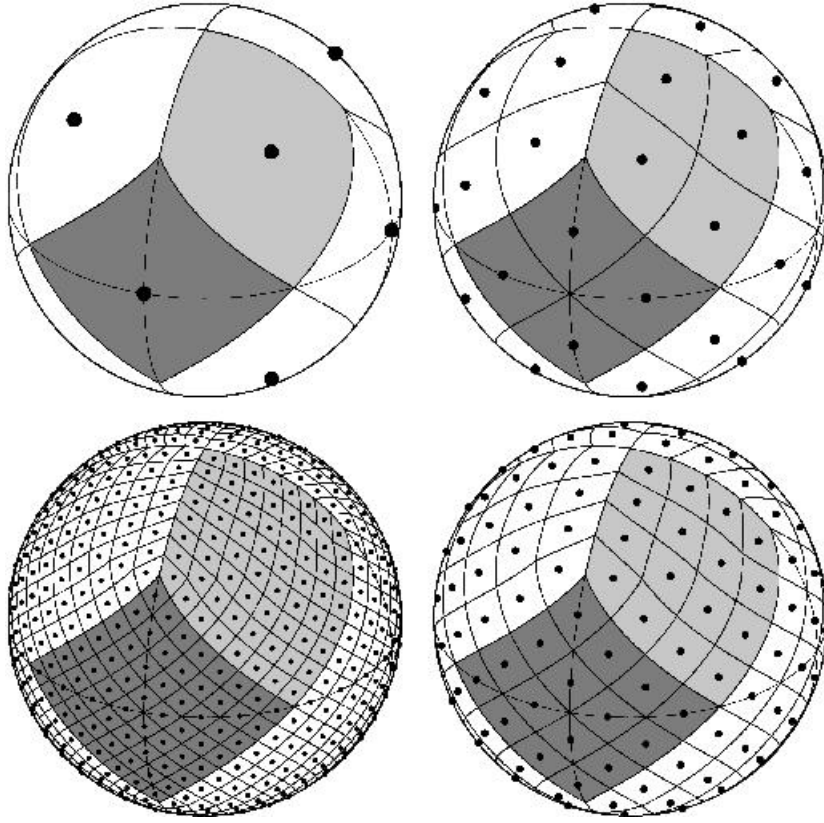


Figure 3.1: HEALPix partitions. Clockwise from top left: the base partition into 12 pixels, then level 1, 2 and 3 (48, 192 and 768 pixels). Taken from <https://healpix.sourceforge.io/>.

It was originally used to provide a discretization of all-sky observations done in connection with CMB but it is now used in many other fields. As of writing, an active website of resources is available at <https://healpix.sourceforge.io/>.

The base partition is into 12 elements. Successively more refined levels of the partition are made by dividing an element into 4 new elements (see Figure 3.1). Taking the base partition to be level 0, the HEALPix partition at level h therefore has 12×4^h elements. For WMAP data, level 9 was used while for Planck a higher resolution of level 10 was possible, hence there are $12 \times 4^{10} = 12,582,912$ pixels in each image. HEALPix software supports data input and output, visualization and features like identifying neighbouring pixel indices and the indices of pixels in discs, triangles, polygons and strips.

Most analyses with data indexed using HEALPix will do separate analyses on each of the 12 base partition elements because functions such as identifying neighbouring pixels

are only available within them. That is what is done here. So our source separation on data at HEALPix level h will consist of 12 independent analyses with $N = 4^h$.

The following analysis is repeated 12 times for each base patch and the results are stitched together:

Let $y_j \in \mathbb{R}^n$ denote the brightness of the maps at pixel j . Assuming that point sources have been dealt with, the brightness at each point in the sky is a combination of the m different diffuse sources. Letting $s_j \in \mathbb{R}^m$ denote the vector of source brightness at pixel j , the source separation model takes the form:

$$y_j = As_j + \epsilon_j; j = 1, \dots, N. \quad (3.1)$$

where A is an $n \times m$ matrix of weights (known as the mixing matrix in source separation) and ϵ_j is a vector of independent measurement errors, assumed to be independently Gaussian distributed with mean 0 and known variances $\sigma_1^2, \dots, \sigma_n^2$. In practical CMB data analysis, these variance values are derived from extensive pre-flight and in-flight calibration of the detectors, as well as from noise estimation procedures applied to the time-ordered data before map-making [Aghanim et al., 2020]. For each frequency channel, the detector noise properties are carefully characterized through dedicated calibration observations and engineering data. The assumption of known error variances is standard in CMB analysis pipelines, as it allows the separation algorithm to appropriately weight the different frequency channels according to their noise levels. While in some applications these variances might be treated as unknown parameters to be estimated, for our purpose of demonstrating the efficiency of linear solvers, we follow the common practice of treating them as known quantities, focusing our computational efforts on the source separation itself rather than on noise estimation.

The CMB is assumed to be the first component of s_j . The components of the mixing matrix A_{kl} can be interpreted as the relative contribution of the l th source at the k th frequency. In the full inference task, A is defined by a small number of parameters whose values must be learned from the data. Because that aspect of the learning task

is not relevant to this work, here the elements of A are assumed to have known values. The following section gives details about construction of matrix A .

3.2.2 Mixing Matrix Construction

The matrix A gives the coefficients of the assumed linear relationship between sources and observed brightness (Equation 3.1). It is an $n \times m$ matrix and so typically small e.g. $n = 9$ for Planck data and we might use $m = 4$ or 5 . The element A_{kl} can be interpreted as the contribution of the l th source to the k th map. It is called the mixing matrix in the context of source separation.

The construction of the mixing matrix A follows well-established parametric models in the CMB literature (Aghanim et al. [2020]; Delabrouille and Cardoso [2007]). These models incorporate prior physical knowledge about the emission mechanisms of different astrophysical components, in contrast to the blind source separation approaches like ICA described in Chapter 2 which attempt to learn the mixing matrix directly from the data without physical constraints.

The first thing to note is the well-known property of source separation solutions (and more generally factor analysis) that A and s are unique only up to an orthogonal transformation. In the CMB application, uniqueness is gained by considering the separation in units of the CMB intensity, so that the first column of A is a vector of 1's. For the other columns of A , the physics of the sources are used to give expressions for the A_{kl} .

For **synchrotron emission**, a straight power law can be used, following the standard spectral behavior established in observations [Fuskeland et al., 2014]:

$$A_{k2} = c(\nu_k) \left(\frac{\nu_k}{\nu_0} \right)^{\kappa_s},$$

for a reference frequency ν_0 at which the source maps will be expressed (for Planck data this is usually the 4th frequency at 100 GHz) and $c(\nu) = (e^\psi - 1)^2 / \psi^2 e^\psi$, where

$\psi = h\nu/k_B T_1$ with h and k_B are the Planck and Boltzmann constants and $T_1 = 18.1$ K. The parameter κ_s is known as the spectral index and is typically somewhere in the range $(-2.3, -3.0)$. Here we assume $\kappa_s = -2.65$. In the full learning task, a prior distribution on κ_s would be specified and it would be part of the learning process.

For **galactic dust**, a modified black body spectrum is used, multiplied by a power law [Aghanim et al., 2020],

$$A_{k3} = c(\nu_k) \frac{B(\nu_k)}{B(\nu_0)} \left(\frac{\nu_k}{\nu_0} \right)^{\kappa_d},$$

where $B(\nu_k) = \nu_k / [\exp(h\nu_k/k_B T_1) - 1]$. The spectral index for dust κ_d is typically in the range $(1, 2)$. Here we assume $\kappa_d = 1.5$. As with κ_s , a prior distribution on κ_s would be specified as part of the full learning process.

For **free-free emission**, a straight power law with known spectral parameter of -2.14 is used, consistent with theoretical predictions and observational constraints [Remazeilles et al., 2011]:

$$A_{k4} = c(\nu_k) \left(\frac{\nu_k}{\nu_0} \right)^{-2.14}.$$

It is worth noting that while these parametric forms are standard in the field, the specific choice of fixed spectral indices represents a simplification of the full problem. In more detailed analyses, these parameters would be learned from the data, potentially with spatial variations. However, for the purpose of demonstrating our linear solvers, this simplified model provides an appropriate test case while maintaining physical realism.

For Planck data, where the 9 maps are for microwave frequencies at 30, 44, 70, 100,

143, 217, 353, 545 and 857 GHz, this yields

$$A = \begin{pmatrix} 1.000 & 24.314 & 0.181 & 13.158 \\ 1.000 & 8.817 & 0.315 & 5.801 \\ 1.000 & 2.581 & 0.612 & 2.151 \\ 1.000 & 1.006 & 1.006 & 1.006 \\ 1.000 & 0.392 & 1.630 & 0.471 \\ 1.000 & 0.132 & 2.783 & 0.196 \\ 1.000 & 0.038 & 4.931 & 0.072 \\ 1.000 & 0.013 & 7.704 & 0.032 \\ 1.000 & 0.005 & 11.337 & 0.015 \end{pmatrix}.$$

3.2.3 Problem Formulation

Let $Y_k = (y_{1k}, \dots, y_{Nk})$ denote the k th map (in other words, the k th component of each y_j) and $\mathcal{Y} = (Y_1, \dots, Y_n) \in \mathbb{R}^{nN}$ denote the vector of all observations, stacked by map. Similarly $S_l = (s_{1l}, \dots, s_{Nl})$ is the map of the l th source and $\mathcal{S} = (S_1, \dots, S_m) \in \mathbb{R}^{mN}$. Following Equation 3.1, one has

$$\mathcal{Y} = B\mathcal{S} + \mathcal{E},$$

where $B = A \otimes I_N$ is an $nN \times mN$ matrix, I_N is the N dimensional identity matrix, and $\mathcal{E}$ is the vector of nN Gaussian distributed measurement errors with mean vector 0 and $nN \times nN$ dimensional diagonal precision matrix C , with the first N entries on the diagonal equal to $1/\sigma_1^2$, the next N equal to $1/\sigma_2^2$, and so on. Hence, given A , $\mathcal{S}$ and the σ_k^2 , $\mathcal{Y}$ is Gaussian distributed with mean $B\mathcal{S}$ and precision matrix C :

$$p(\mathcal{Y} | \mathcal{S}) = (2\pi)^{Nn/2} |C|^{1/2} \exp \left\{ -\frac{1}{2} (\mathcal{Y} - B\mathcal{S})^T C (\mathcal{Y} - B\mathcal{S}) \right\}. \quad (3.2)$$

Note that, for Planck data, $nN = 9 \times 4^{10} = 9,437,184$ and typically one uses $m = 4$ diffuse sources so that $mN = 4 \times 4^{10} = 4,194,304$.

The final element of the problem is a model for the diffuse sources. In a Bayesian learning approach, this model is a probability distribution $p(\mathcal{S})$, known as the prior distribution, that serves to regularise the separation task. In our approach, this distribution models the spatial smoothness of each diffuse source through the use of an intrinsic Gaussian Markov random field model [Rue and Held, 2005b]. This model ensures that the intensity of a source at neighbouring pixels in the sky cannot be too different, concretely that if pixel j and j' are neighbours then $y_{jk} - y_{j'k}$ is zero-mean Gaussian distributed with a variance ϕ_k whose value must be specified. The different source maps are assumed to be independent. This leads to a zero-mean multivariate Gaussian form for $p(\mathcal{S})$:

$$p(\mathcal{S}) = (2\pi)^{Nm/2} |Q|^{1/2} \exp \left\{ -\frac{1}{2} \mathcal{S}^T Q \mathcal{S} \right\}, \quad (3.3)$$

where $Q = \text{bldiag}(Q_1(\phi_1), \dots, Q_m(\phi_m))$ is an $mN \times mN$ matrix and the $Q_l(\phi_l)$ are sparse, positive semi-definite $N \times N$ matrices, each depending on a single parameter ϕ_l . The form and derivation of Q_l using intrinsic Gaussian Markov random field (IGMRF) model is described below.

Definition 2. *A set of random variables $X = (X_1, \dots, X_N)$ is a **Markov random field** if*

- *Each variable X_i has a neighbourhood set $\eta_i \subset \{1, \dots, N\}$ that satisfies the properties: $i \notin \eta_i$ and $j \in \eta_i$ iff $i \in \eta_j$.*
- *Each X_i is independent of the other variables in the set given its neighbours. In other words $p(x_i | \{x_j | j \neq i\}) = p(x_i | \{x_j | j \in \eta_i\})$.*

Intuitively, when the X_i are values of pixels in a 2-dimensional image, η_i is the indices of the neighbour pixels of pixel i . In this application using the HEALPix partition, the neighbours are the 4 pixels that share a common edge (as can be seen in Figure 3.1).

Under quite general conditions, the joint probability distribution $p(x_1, \dots, x_n)$ is uniquely

defined by the set of conditional distributions $p(x_i | \{x_j | j \in \eta_i\})$, $i = 1, \dots, N$, of each random variable given its neighbours. For the case of a 2-dimensional grid, the effect of these assumptions is that the values of nearby random variables on the grid are more correlated than those that are far away, and so realizations of the set of random variables are more spatially smooth than the case where each variable is independent.

When these conditional distributions are Gaussian, the joint distribution is a multivariate Gaussian with some mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and it is known as a *Gaussian Markov random field* (GMRF). The inverse of the covariance matrix, $Q = \boldsymbol{\Sigma}^{-1}$, known as the precision matrix, has a nice structure since an off-diagonal element is zero if and only if the corresponding random variables are not neighbours:

$$j \notin \{i\} \cup \eta_i \iff Q_{ij} = 0.$$

In most cases, including ours, since the number of neighbours is small relative to N , Q will be sparse. Often, permutation of the indices will yield Q to be a band matrix with relatively small width, permitting efficient computation of its Cholesky decomposition that is sufficient to compute the multivariate Gaussian density and also for simulation.

Finally, we get to the intrinsic Gaussian Markov Random Field. The choice of an intrinsic Gaussian Markov Random Field (IGMRF) as a prior distribution for the sources is motivated by several important considerations. First, the IGMRF provides a natural way to incorporate spatial smoothness into the model, which is physically justified for diffuse CMB and foreground components that vary smoothly across the sky [Rue and Held, 2005b]. Unlike point sources which exhibit sharp spatial transitions, diffuse components like the CMB, synchrotron radiation, and galactic dust are expected to show gradual variations between neighboring pixels.

Second, the IGMRF's computational properties make it particularly suitable for large-scale problems like CMB source separation. The precision matrix Q of the IGMRF is sparse, with non-zero elements only between neighboring pixels, which allows for

efficient computation despite the large number of pixels in modern CMB datasets. This sparsity is crucial for our approach to scalable linear solvers.

Third, the improper nature of the IGMRF, while mathematically unusual, is appropriate in this context because we are primarily interested in the spatial pattern of the sources rather than their absolute levels. The IGMRF only specifies the relationship between neighboring pixels without constraining the overall mean level, which is consistent with the CMB source separation problem where the mean level is not physically meaningful. These models look like GMRFs but their joint probability distribution takes the form of a multivariate Gaussian but with a precision matrix that is not of full rank. Hence their joint distribution is not a properly defined probability density function. This may seem to be a disqualifying property for a probability model but in fact (a) they model properties of the random variables that can be useful and (b) when used as a prior distribution, as in this application, the resulting posterior distribution is very often a properly defined probability distribution.

The vector of random variables $X = (X_1, \dots, X_N)$ is an *improper GMRF* of rank $N - k$ with parameters $(\boldsymbol{\mu}, Q)$ if its density is of the form:

$$p(x) = (2\pi)^{-0.5(N-k)} (|\widehat{Q}|)^{0.5} \exp(-0.5(x - \boldsymbol{\mu})^T Q(x - \boldsymbol{\mu})), \quad x \in \mathbb{R}^N,$$

where Q is an $N \times N$ semi-positive definite matrix of rank $N - k > 0$ and $|\widehat{Q}|$ is a generalised determinant, being the product of the $N - k$ non-zero eigenvalues of Q .

The properties of such a construction are discussed in great detail in Rue and Held [2005b]. A specific example of an IGMRF, that we use in this work, is where $\boldsymbol{\mu} = 0$ and $Q = \phi D^T D$, for $\phi > 0$ and D an $N \times N$ matrix that encodes the neighbourhood structure,

$$D_{ij} = \begin{cases} 1, & \text{if } i \in \eta_j, \\ 0, & \text{otherwise,} \end{cases} \quad (3.4)$$

giving Q a rank of $N - 1$. One can show that under this definition of Q , the distribution of X_i given its neighbours $\{X_j \mid j \in \eta_i\}$ is Gaussian with mean equal to the average of its neighbours. This makes a realisation of X somewhat smooth, in that neighbouring values in the HEALPix lattice are more likely to take similar values than those that are further apart. Another property of this construction is that there is no mean level specified — although $\boldsymbol{\mu} = 0$ this is not the ‘mean’ of X as it is not a properly defined probability distribution. The construct only gives properties of the relationship between neighbouring values in the lattice, not about their overall level. This is useful if all that one wants to model for X is this smooth spatial property, and not commit to a particular mean level a priori.

That is the approach that we have taken in modelling the sources in our separation problem. Each source is modelled as an independent IGMRF, with the l th source having parameters $\mu_l = 0$ and $Q_l = \phi_l D^T D$, with D defined as in Equation 3.4. In this thesis, we have fixed $\phi_l = 1$ for each source. Stacking these independent sources into the vector $\mathcal{S}$ yields a Gaussian form with mean 0 and a block-diagonal precision matrix $Q = \text{diag}(Q_1, \dots, Q_m)$.

3.3 Solving the Linear System

The primary task is to use the data $\mathcal{Y}$ to learn about all sources $\mathcal{S}$. A Bayesian learning approach is adopted, where the goal is to derive the so-called posterior probability distribution $p(\mathcal{S} \mid \mathcal{Y})$. For this Bayes’ Law is used:

$$p(\mathcal{S} \mid \mathcal{Y}) \propto p(\mathcal{Y} \mid \mathcal{S}) p(\mathcal{S}),$$

where Equations 3.2 and 3.3 specify $p(\mathcal{Y} \mid \mathcal{S})$ and $p(\mathcal{S})$.

Some simple properties of the Gaussian distribution show that $p(\mathcal{S} \mid \mathcal{Y})$ is another

Gaussian with precision matrix

$$\hat{Q} = B^T C B + Q \quad (3.5)$$

and mean vector $\boldsymbol{\mu}^*$ that satisfies

$$\hat{Q}\boldsymbol{\mu}^* = B^T C \mathcal{Y}. \quad (3.6)$$

Therefore, key to implementing the source separation is the solution to Equation 3.6, under the structures for Q , B and C that have been described, as this gives a point estimate of the sources. In addition, computing elements of $\hat{Q}$ (of most importance being the main diagonal that contains the precision or inverse variance of the sources at each pixel) gives information about uncertainty of and correlation between source values across the sky.

Although Q is only semi-positive definite, $\hat{Q}$ will be positive definite. Solving Equation is typically done, where possible, with a Cholesky decomposition of $\hat{Q}$. As is often the case, the sparsity of $\hat{Q}$ does not guarantee that its Cholesky decomposition will be sparse; indeed, we can demonstrate this for a small sample realization; see Figure 3.2. One could consider sparse direct methods (see, e.g., [Gilbert and Schreiber, 1992]) in this setting, but this problem has so much additional structure that we explore matrix-free approaches which take advantage of Kronecker structure. Thus, we do not consider sparse-direct methods in this work, though we do use SuperLU [Li and Demmel, 2003] as an inner solver for a method suggested in [Benner et al., 2011] against which we test our proposed solvers.

The problem being solved is of the form

$$(Q + B^T C B)\boldsymbol{\mu} = B^T C \mathcal{Y} \text{ with } (Q + B^T C B) \in \mathbb{R}^{mN \times mN} \quad (3.7)$$

where $m \in \{4, 5\}$ and $N = \mathcal{O}(10^7)$. As described in the previous section, Q is a m -block diagonal matrix with block i being of the form $Q_i(\varphi_i) = \varphi_i D^T D = \varphi_i D^2$,

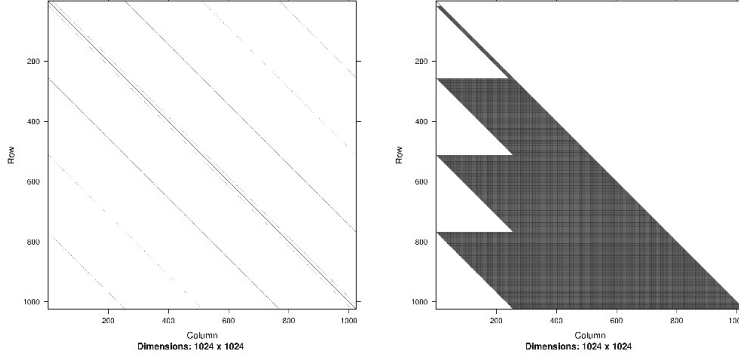


Figure 3.2: Sparseness of $\hat{Q} = Q + B^T C B$ (left) and its Cholesky decomposition (right) for a small example ($m = 4$ and $N = 256$). Non-zero elements are shaded.

for $\varphi_i > 0$. The matrix $D \in \mathbb{R}^{N \times N}$ is symmetric and its elements encode a nearest neighbor coupling of nodes from the discretization of the sky, whereby

$$D = (D_{ij}) \text{ with } D_{j_1, j_2} = \begin{cases} 1 & \text{nodes } j_1 \neq j_2 \text{ are neighbors} \\ 0 & \text{nodes } j_1 \neq j_2 \text{ are not neighbors} \\ -\sum_{\substack{\ell=1 \\ \ell \neq j}}^N D_{j\ell} & j := j_1 = j_2 \end{cases}$$

The construction of matrix D with a sum-zero constraint on each row is motivated by theoretical and practical considerations beyond mere computational convenience. This constraint ensures that D represents a discrete approximation of the Laplacian operator, which is fundamental in modeling spatial smoothness in physical fields. The Laplacian naturally enforces that the expected value of a pixel equals the average of its neighbors, a property that aligns with our physical understanding of diffuse emission processes in the CMB. Furthermore, this formulation leads to an intrinsic Gaussian Markov random field with the desirable property of being invariant to the addition of a constant to all pixels, reflecting the fact that in CMB analysis, the absolute calibration level is often less important than the spatial variations across the sky. From a statistical perspective, this construction yields a proper conditional specification of the spatial dependence while maintaining computational efficiency through the resulting sparse precision matrix structure. The sum-zero constraint thus provides a theoretically sound basis for spatial regularization that captures the essential physics of the problem while

enabling efficient computation.

The matrix $B \in \mathbb{R}^{nN \times mN}$ with $n = 9$ can be written as the Kronecker product $B = A \otimes I_N$ where $A \in \mathbb{R}^{n \times m}$. We also have that $C \in \mathbb{R}^{nN \times nN}$ is a diagonal matrix with all positive entries. The matrix C also has Kronecker structure; namely, there are N positive integer values $n_1, \dots, n_N$ and n positive values $\tau_1, \dots, \tau_n$

$$C = \text{diag}\{\tau_1, \tau_2, \dots, \tau_n\} \otimes \text{diag}\{n_1, n_2, \dots, n_N\} =: \vec{T} \otimes \vec{N}.$$

For this application, the value of vector T is provided as follow:

$$\begin{aligned} \vec{T} = \text{diag}\{629881.6, 694444.4, 783146.7, 12755102.0, 30864197.5, 30864197.5, \\ 30864197.5, 30864197.5, 30864197.5\}. \end{aligned}$$

We propose treating this problem using iterative methods built on efficient implementations of matrix-vector products (matvecs). Furthermore, most of the system matrix should never be explicitly constructed. Rather, it should only be represented as a stored procedure implementing matrix-vector product. The only matrix explicitly represented should be A .

3.3.1 Efficient matrix-vector products

The goal of building matrix-vector product operations is to calculate Gx from x (where G is a large matrix) without having to deal with the full size of G . We describe first how to perform the matrix-vector product for each part of the full system matrix and then we finish by combining them, yielding a routine for $x \mapsto (Q + B^T C B)x$. This will be necessary for using an appropriate iterative method whose core operation is this matvec routine.

As already discussed, Q is a block diagonal matrix with blocks $\varphi_i D^2$. Thus, it can be

represented as a Kronecker product, namely

$$Q = \begin{bmatrix} \varphi_1 & & & \\ & \varphi_2 & & \\ & & \ddots & \\ & & & \varphi_n \end{bmatrix} \otimes D^2 =: P \otimes D^2.$$

Further more, we can utilize the Kronecker structure presented in B and C to write:

$$B^T C B x = (A^T \otimes I_N)(T \otimes N)(A \otimes I_N)x.$$

Hence, the left hand side of Equation 3.7 becomes:

$$(Q + B^T C B)x = Qx + B^T C B x = P \otimes D^2 x + (A^T \otimes I_N)(T \otimes N)(A \otimes I_N)x$$

Observe that since D is a sparse matrix encoding the nearest-neighbor coupling for each node, the routine $v \mapsto Dv$ involves updating entry j of v (associated to node j of grid) using its four nearest neighbors and the value at the node itself. Thus a sequential matrix-free routine can be easily formulated (Algorithm 2). For Algorithm 2, since the matrix D does not change, we can pre-compute and store the rowsums $\sum_{l=1, l \neq j}^N D_{jl}, j = 1, 2, \dots, N$ to reduce the computation complexity. Furthermore, Algorithm 2 is parallelizable, as the grid can be divided up and sent to different processes, and communication between processors only needs to occur for neighboring nodes that are on different processors. The computational cost of a serial version of Algorithm 2 is $\mathcal{O}(N)$, since we do 4 floating point operations per entry of v .

Algorithm 2: Matrix-vector procedure for $v \mapsto Dv$

Data: vector $v \in \mathbb{R}^N$ **Result:** Dv

```
1 for  $j = 1, 2, \dots, N$  do
2    $v[j] \leftarrow v[j_{right}] + v[j_{left}] + v[j_{above}] + v[j_{below}] - (\sum_{l=1, l \neq j}^N D_{jl})v[j]$ 
3 end
4 return  $v$ 
```

Then we can build a simple matrix-vector routine on top of Algorithm 2 to calculate Qu from $u = w \otimes v$ with $w \in \mathbb{R}^n$:

Algorithm 3: Matrix-vector procedure for $u \mapsto Qu$

Data: vector $u = w \otimes v \in \mathbb{R}^{nN}$, with $w \in \mathbb{R}^n$ and $v \in \mathbb{R}^N$ **Result:** Qu

```
1  $w \leftarrow Pw$ 
2  $v \leftarrow Dv; v \leftarrow Dv$  (Algorithm 2)
3 return  $v$ 
```

Here we made an assumption that we have a Kronecker representation of $u = w \otimes v$, which we may not know. In that case, we can perform this matrix-vector product by reshaping u . For $u \in \mathbb{R}^{nN}$, $\text{reshape}(u, N, n) \in \mathbb{R}^{N \times n}$ is obtained by arranging every N entry of u as a column of the new matrix:

$$u = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} \text{ and } \text{reshape}(u, N, n) = [u_1 \ u_2 \ \dots \ u_n]$$

An implementation of matrix-vector product for $u \mapsto Qu$ is presented in Algorithm 4

Algorithm 4: Matrix-vector procedure for $u \mapsto Qu$ using reshape

Data: vector $u \in \mathbb{R}^{nN}$

Result: Qu

```

1  $U \leftarrow \text{reshape}(u, N, n)$ 
2  $U \leftarrow D^2UP$ 
3  $u \leftarrow \text{vectorize}(U)$ 
4 return  $u$ 

```

Now we can calculate Qx completely matrix-free from x . Likewise, for $x \mapsto B^T C B x$, if x has the known Kronecker structure $x = w_x \otimes v_x$ with $w_x \in \mathbb{R}^n$ and $v_x \in \mathbb{R}^N$, we can build a matrix-vector procedure as in Algorithm 5. If not, we can set $M = \text{reshape}(x, N, n)$, then the matrix-vector product can be efficiently executed via the block vector mapping:

$$M \mapsto (N(MA^T)T)A, \quad (3.8)$$

which enables the running of a matrix-free Krylov iteration on a Kronecker-structured system implicitly, without forming any Kronecker products (Algorithm 6).

Algorithm 5: Matrix-vector procedure for $x \mapsto B^T C B x$ when x has a Kronecker structure

Data: vector $x = w_x \otimes v_x$ with $w_x \in \mathbb{R}^n$ and $v_x \in \mathbb{R}^N$

Result: Qu

```

1  $w_x \leftarrow Aw_x$ 
2  $w_x \leftarrow Tw_x; v_x \leftarrow Nv_x$ 
3  $w_x \leftarrow A^T w_x$ 
4  $x \leftarrow w_x \otimes v_x$ 
5 return  $x$ 

```

Algorithm 6: Matrix-vector procedure for $x \mapsto B^T C B x$ using reshape

Data: vector $x \in \mathbb{R}^{nN}$

Result: Q_{u}

```

1  $X \leftarrow \text{reshape}(x, N, n)$ 
2  $X \leftarrow X A^T$ 
3  $X \leftarrow N X T$ 
4  $X \leftarrow X A$ 
5  $X \leftarrow \text{vectorize}(X)$ 
6 return  $x$ 

```

3.3.2 Matrix-free iterative solvers

The first strategy we propose entails considering the linear system 3.7 as it is presented with the coefficient matrix being built from and applied implicitly using the Kronecker product formulation. We propose to use a so-called “matrix-free” method which only requires a stored procedure to perform $x \mapsto (Q + B^T C B)x$, implemented in block form via (6.1), meaning we need no explicit representation of the coefficient matrix.

We briefly summarize the Krylov Subspace methods and our implementation for the defined problem. More details can be found in Section 2.3.

Consider solving:

$$Gx = b \text{ with } G \in \mathbb{R}^{M \times M} \text{ and } b \in \mathbb{R}^M$$

where $M \gg 0$ is large and G is sparse. Given a basis of vectors $v_1, v_2, \dots, v_j$ spanning a Krylov subspace $\mathcal{V}$, we approximate x by $x_j = \sum_{i=1}^j z_i v_i \in \mathcal{V}$ that satisfies:

$$v_i^T G x_j = v_i^T b \text{ for all } 1 \leq i \leq j$$

Let $\mathbf{H} \in \mathbb{R}^{i \times j}$ be the matrix whose $(i, k)^{\text{th}}$ element is $v_i^T G v_k$, $\mathbf{z}_j \in \mathbb{R}^j$ be the vector whose entries are the unknown coefficients z_i , and $\mathbf{s} \in \mathbb{R}^j$ be the vector whose entries

are $v_i^T b$, then solving for $\mathbf{z}_j$ is equivalent to solving the small $j \times j$ linear system:

$$\mathbf{H}\mathbf{z} = \mathbf{s}$$

Thus, we have exchanged solving a large $M \times M$ problem for solving a small $j \times j$ problem. More importantly, with the Kronecker structure of the matrix involved, we can efficiently solve the smaller problem without having to explicitly store matrix G in memory or doing any direct computation on its full size. That thanks to the Krylov subspace, whereby the construction of the basis entails one matrix-vector procedure per iteration, which means that $\mathbf{H}$ is generated automatically during basis construction. The standard Krylov subspace method can be described as follow: Given the equation:

$$G(x_0 + t) = b \tag{3.9}$$

where $G \in \mathbb{C}^{n \times n}$, $x_0 \in \mathbb{C}^{n \times n}$ and $\mathbf{t}$ is the unknown correction. Let $r_0 = b - Gx_0$ be the initial residual. In iteration j , a Krylov method selects a correction t_j from j^{th} Krylov subspace generated by G and r_0 :

$$t_j \in \mathcal{K}(G, r_0) = span\{r_0, Gr_0, G^2r_0, \dots, G^{j-1}r_0\} \tag{3.10}$$

We present the CG method for the case $G = (Q + BCB^T)$ and right-hand side $b = B^T C y$ in Algorithm 7. At iteration j , the algorithm generates the subspace $\mathcal{K}_j(G, r_0)$ and computes $t_j \in \mathcal{K}_j(G, r_0)$ such that the approximation $x_j = x_0 + t_j$ minimizes:

$$||b - Gx_j||_G = (b - Gx_j)^T G(b - Gx_j)$$

Algorithm 7: Conjugate Gradient Method

Data: y, B, C and Q defined as above and x_0 ;

$\epsilon > 0$ as convergence tolerance parameter

Result: x as solution of the linear system

```
1  $x = x_0; r = BCy - (Q + B^T C B)x; p = r_0$ 
2 while  $\|r\| > \epsilon \|r_0\|$ 
3    $r_{old} \leftarrow r$ 
4    $q \leftarrow (Q + B^T C B)p$ 
   ( $p \rightarrow Qp$  from Algorithm 3 and  $p \rightarrow B^T C Bp$  from Algorithm 5)
5    $\alpha \leftarrow \frac{r^T r}{p^T q}$ 
6    $x \leftarrow x + \alpha p$ 
7    $r \leftarrow r - \alpha q$ 
8    $\beta \leftarrow \frac{r^T r}{r_{old}^T r_{old}}$ 
9    $p \leftarrow r + \beta p$ 
10 end
11 return  $x$ 
```

3.4 Solver Analysis

3.4.1 Theoretical Analysis

As an iterative method, the conjugate gradient method monotonically (in the energy norm) improves approximations x_k to the exact solution and may reach the required tolerance after a relatively small number of iterations [Cockayne et al., 2018]. The speed of convergence is determined by the condition number $\kappa(A)$ of the system matrix A , defined as the product of the norm of A and the norm of A^{-1} : $\kappa(A) = \|A\| \|A^{-1}\|$

Theorem 1. Convergence theorem [Hackbusch, 1994]

Define a subset of polynomials as

$$\Pi_k^* := \{ p \in \Pi_k : p(0) = 1 \}$$

where Π_k is the set of polynomials of maximal degree k .

Let $(\mathbf{x}_k)_k$ be the iterative approximations of the exact solution $\mathbf{x}_*$, and define the errors as $\mathbf{e}_k := \mathbf{x}_k - \mathbf{x}_*$. The rate of convergence can be approximated as:

$$\begin{aligned} \|\mathbf{e}_k\|_{\mathbf{A}} &= \min_{p \in \Pi_k^*} \|p(\mathbf{A})\mathbf{e}_0\|_{\mathbf{A}} \\ &\leq \min_{p \in \Pi_k^*} \max_{\lambda \in \sigma(\mathbf{A})} |p(\lambda)| \|\mathbf{e}_0\|_{\mathbf{A}} \\ &\leq 2 \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^k \|\mathbf{e}_0\|_{\mathbf{A}} \\ &\leq 2 \exp \left(\frac{-2k}{\sqrt{\kappa(\mathbf{A})}} \right) \|\mathbf{e}_0\|_{\mathbf{A}}, \end{aligned}$$

where $\sigma(\mathbf{A})$ denotes the spectrum, and $\kappa(\mathbf{A})$ denotes the condition number.

Theorem 1 shows that $k = \frac{1}{2} \sqrt{\kappa(\mathbf{A})} \log(\|\mathbf{e}_0\|_{\mathbf{A}} \varepsilon^{-1})$ iterations suffices to reduce the error to 2ε for any $\varepsilon > 0$, meaning the larger $\kappa(A)$ is, the slower the convergence.

In most cases, preconditioning is necessary to ensure fast convergence of the conjugate gradient method. It replaces the original system $Ax - b = 0$ with $M^{-1}(Ax - b) = 0$ such that $\kappa(M^{-1}A) < \kappa(A)$.

3.4.2 Practical Analysis

As discussed in the previous section, the CG method works well when the coefficient matrix is well-conditioned. In the setting of our problem, we observe that the coefficient matrix arises from the Bayesian regularisation of the source separation problem. Such matrices are often more well-conditioned due to the regularisation. Another advantage

of applying CG in the CMB source separation problem is that the coefficient matrix does not need to be explicitly constructed. Furthermore, in this setting, CG is quite amenable to parallelization by distributing sky domain grid onto different processors and parallelizing the matvec procedures.

However, The CG method also has some relevant disadvantages in this setting. We still must deal with vectors of size nN . We also cannot take advantage of any structure or compressibility which may be present in Y . Most importantly, it has been shown that applying an iterative method directly to the tensor-structured problem can lead to performance hampered by the large condition number of the full-size problem [Palitta and Simoncini, 2020a]

3.5 Conclusion

This chapter introduced our Krylov-subspace method for solving the CMB source separation problem. We showed that one can exploit the Kronecker structure arises in the construction of the linear equation to build a matrix-free iterative solver. We analyzed the advantages and disadvantages of the method in both theoretical and practical aspects. Understanding both aspects allows us to develop a more robust solver using Sylvester interpretation of the linear equation. Next chapter gives details of that approach, following by experimental comparison between the two methods.

4 Linear Solver with Sylvester Interpretation

4.1 Summary

In this chapter, we introduce the development of a linear solver based on the Sylvester interpretation of the Equation 3.7. We re-formulate the it as a Sylvester matrix equation, allowing us to take advantage of additional structures that may be available in Y . The resulted sparse-dense Sylvester equation can be solved using multiple techniques, of which we briefly review. We present a solution proposed by Benner et al. [2011] as the baseline model for Sylvester approach of solving CMB source separation problem. A detailed analysis in complexity, application and experimental results of the solution is provided,

4.2 Sparse-Dense Sylvester Linear Equation Solvers

4.2.1 Sylvester Interpretation

In Section 3.4.2, we analyzed that the CG method cannot exploit any additional structure that may be present in Y . To resolve that issue, we simply reshape Equation 3.7 and interpret directly as a Sylvester matrix equation.

Let $M = \text{reshape}(\mu, N, n)$ and $Y = \text{reshape}(y, N, m)$, we have:

$$(Q + B^T C B)\mu = P \otimes D^2 \mu + (A^T \otimes I_N)(T \otimes N)(A \otimes I_N)\mu = D^2 M P + N M A^T T A$$

$$\text{and: } B^T C y = (A^T \otimes I_N)(T \otimes N)y = N Y T A$$

Hence, Equation 3.7 can be rewritten as the generalized Sylvester equation:

$$D^2 M P + N M A^T T A = N Y T A$$

As N and P are diagonal and invertible, we can multiply on the left by N^{-1} and on the right with N^{-1} , yielding:

$$N^{-1} D^2 M + M A^T T A P^{-1} = Y T A P^{-1} \quad (4.1)$$

Since $A^T T A P^{-1} \in \mathbb{R}^{m \times m}$, this falls into a category of Sylvester equations with one large, sparse coefficient matrix ($N^{-1} D^2$) and one small, dense one ($A^T T A P^{-1}$). Following [Benner et al., 2011], we denote this as *sparse-dense* category. We describe a technique from the literature wherein this structure has been previously exploited.

4.2.2 Solving Sparse-Dense Sylvester Equation

Definition 3. *A Sylvester Equation of the form:*

$$A X + X H + M = 0 \quad (4.2)$$

*with $A \in \mathbb{R}^{p \times p}$, $H \in \mathbb{R}^{r \times r}$ and $M \in \mathbb{R}^{p \times r}$ is **sparse-dense** if A is a large and sparse matrix and H is a small and dense matrix.*

This definition can also be used for the *semi generalized* Sylvester Equation

$$A X + E X H + M = 0$$

with $A \in \mathbb{R}^{p \times p}$, $E \in \mathbb{R}^{p \times p}$, $H \in \mathbb{R}^{r \times r}$ and $M \in \mathbb{R}^{p \times r}$ and the *generalized* Sylvester equation:

$$AXF + EXH + M = 0$$

with $A \in \mathbb{R}^{p \times p}$, $E \in \mathbb{R}^{p \times p}$, $F \in \mathbb{R}^{r \times r}$, $H \in \mathbb{R}^{r \times r}$ and $M \in \mathbb{R}^{p \times r}$.

Benner et al. [2011] developed an algorithm to exploit the special structure in sparse-dense Sylvester equations of the form 4.2. We describe the algorithm in details and implement it as a baseline solution to Equation 4.1.

Replacing H in 4.2 by its Schur decomposition ($H = USU^H$ with $U^H U = UU^H = I$), one has:

$$AX + XUSU^H + M = 0$$

Multiplying both side by U , one can arrange it to a different Sylvester equation:

$$AXU + XUS + MU = 0$$

or

$$A\tilde{X} + \tilde{X}S + \tilde{M} = 0 \tag{4.3}$$

with $\tilde{X} = XU$, $\tilde{M} = MU$

In Equation 4.3, the special structure is exploited to build a column-by-column solver. Observing that S is an upper triangular matrix, the first column $\tilde{X}_1$ of $\tilde{X}$ can be calculated as:

$$A\tilde{X}_1 + \tilde{X}_1 S_{11} + \tilde{M}_1 = 0 \tag{4.4}$$

$$(A + S_{11}I)\tilde{X}_1 = -\tilde{M}_1 \tag{4.5}$$

Similarly, the second column of $\tilde{X}$ can be written as:

$$A\tilde{X}_1 + S_{12}\tilde{X}_1 + S_{22}\tilde{X}_2 + \tilde{M}_1 = 0 \quad (4.6)$$

$$(A + S_{22}I)\tilde{X}_2 = -\tilde{M}_2 - S_{12}\tilde{X}_1 \quad (4.7)$$

By induction, one can build a procedure to calculate each column j of $\tilde{X}$:

$$(A + S_{jj}I)\tilde{X}_j = -\tilde{M}_j - \sum_{i=1}^{j-1} S_{ij}\tilde{X}_i \quad (4.8)$$

Based on the procedure in 4.8, a complete solution of the sparse-dense Sylvester equation is presented in Algorithm 8.

Algorithm 8: Solution of the Sparse-Dense Sylvester Equation (from [Benner et al., 2011])

Data: $A \in \mathbb{R}^{p \times p}$, $H \in \mathbb{R}^{r \times r}$ and $M \in \mathbb{R}^{p \times r}$ defining 4.2

Result: $X \in \mathbb{R}^{p \times r}$ solving 4.2

```

1 Compute Schur decomposition  $USU^T = H$ 
2  $\tilde{M} := MU$ 
3 for  $j = 1, 2, \dots, r$  do
4    $\widehat{M} := -M_j - \sum_{i=1}^{j-1} \tilde{S}_{ji}\tilde{X}_i$ 
5   Solve  $(A + S_{jj}I)\tilde{X}_j = \widehat{M}$ 
6 end
7  $X := \tilde{X}U$ 
8 return  $X$ 
```

4.3 Solver Analysis

4.3.1 Complexity Analysis

The complexity of solving the sparse-dense Sylvester equation is primarily driven by the complexity of solving the linear system at each step. As shown in Algorithm 8, the procedure involves decomposing H using Schur decomposition and solving a series of smaller linear systems for each column of the solution matrix X . Specifically, the algorithm breaks the problem into recursive steps where each column X_j is solved

iteratively based on the previous columns.

The overall complexity of the algorithm is dominated by three key components:

- **Schur Decomposition:** The Schur decomposition of H , a dense matrix of size $r \times r$, has a complexity of $\mathcal{O}(r^3)$, which is typically small relative to p since $r \ll p$.
- **Matrix-Vector Multiplications:** The complexity of the matrix-vector multiplications (e.g. MU where U comes from the Schur decomposition) is $\mathcal{O}(r^2p)$, which scales linearly with the size of the large sparse matrix A .
- **Linear System Solvers:** The complexity of solving each linear system $(A + S_{jj})\tilde{X}_j = \widehat{M}$ is denoted as $\mathcal{O}(S)$, where S depends on the method used. For example:
 - **Direct Solvers:** If a sparse direct solver like LU decomposition is used, the complexity of solving each system can be as high as $\mathcal{O}(p^3)$ in the worst case, though for many sparse matrices, especially with special structures, the practical complexity is closer to $\mathcal{O}(p^2)$ or even $\mathcal{O}(p)$ depending on the sparsity pattern.
 - **Iterative Solvers:** When iterative methods like the Conjugate Gradient (CG) or GMRES methods are used, the complexity can be significantly lower, often approaching $\mathcal{O}(p)$, especially for well-conditioned or preconditioned systems.

The overall complexity of the sparse-dense Sylvester solver is therefore:

$$\mathcal{O}(r^3) + \mathcal{O}(r^2p) + r\mathcal{O}(S)$$

where $\mathcal{O}(S)$ represents the cost of solving the sparse linear systems. For large p , this complexity is dominated by $\mathcal{O}(S)$, making the choice of the linear system solver critical to the overall efficiency.

Step	Number of calls	Complexity per call
1	1	$\mathcal{O}(r^3)$
2	1	$\mathcal{O}(r^2p)$
4	r	$\mathcal{O}(rp)$
5	r	$\mathcal{O}(S)$
7	1	$\mathcal{O}(r^2p)$

Table 4.1: Complexity of the sparse-dense Sylvester solver

The detailed estimated runtime complexity of each step in Algorithm 8 is shown in Table 4.1.

Another important aspect of theoretical complexity is the algorithm’s potential for parallelism. In practical applications, particularly when solving large-scale problems in cosmology or physics, distributed or parallel computing environments are often employed. The column-by-column nature of the Sylvester solver suggests possibilities for parallelization, but careful consideration of data dependencies is essential.

While the straightforward implementation of Algorithm 8 processes columns sequentially due to the dependencies in step 4, several parallel approaches have been proposed in the literature. Benner et al. [2003] developed parallel implementations for related Sylvester-type equations arising in control theory by focusing on the sparse linear system solves, which dominate the computational cost. Granat and Kågström [2010] further explored parallel implementations specifically for the sparse-dense case, demonstrating significant speedup on multi-core architectures.

With appropriate parallel sparse direct solvers (like *SuperLU_{MT}* as described by Li [2005]), the theoretical runtime complexity for solving the sparse linear systems could approach $\mathcal{O}(p/c)$ where c is the number of processors, though communication overhead affects practical scalability. These parallelization strategies are particularly relevant for our CMB application, where the size of the datasets necessitates efficient computational approaches.

4.3.2 Practical Analysis

In practical scenarios, several methods have been proposed and implemented to solve sparse-dense Sylvester equations efficiently, each leveraging different aspects of the matrix structure to reduce complexity and runtime.

Direct solvers like UMFPACK [Davis, 2004] and SuperLU [Li and Demmel, 2003] are widely used to solve large sparse linear systems. These solvers are based on LU decomposition and are capable of handling sparse matrices efficiently, exploiting the sparsity pattern to reduce the fill-in (additional non-zero entries that arise during factorization). In the context of solving the sparse-dense Sylvester equation, these solvers are particularly effective when A is sparse and has a favorable sparsity pattern (e.g., banded, diagonal-dominant). However, for very large systems, the $\mathcal{O}(p^3)$ worst-case complexity can still be prohibitive, and memory constraints may become an issue.

On the other hand, in many large-scale applications, iterative solvers such as the Conjugate Gradient (CG) method and Generalized Minimal Residual (GMRES) method are preferred due to their lower memory requirements and potential for faster convergence. When A is sparse and well-conditioned, or when effective preconditioning is available, iterative methods can achieve near-linear complexity. The use of Krylov subspace methods like GMRES [Saad and Schultz, 1986b] has become common in large-scale applications due to their robustness and ability to handle non-symmetric systems. For sparse-dense Sylvester equations, preconditioning plays a crucial role in reducing the number of iterations required to achieve convergence.

Several preconditioners have been proposed for Krylov subspace methods in the context of Sylvester equations. For instance:

- **Incomplete LU (ILU) Preconditioning:** ILU preconditioners approximate the LU factorization of the matrix A without computing the full decomposition. This approximation can significantly speed up convergence in iterative methods, particularly when A is ill-conditioned [Saad, 2003b].

- **Multigrid Methods:** Multigrid techniques have also been employed as preconditioners, particularly for problems arising in physics and engineering, where the system matrix A exhibits a hierarchical structure. These methods leverage different levels of resolution to accelerate convergence, making them ideal for large-scale problems [Briggs et al., 2000]

4.4 Conclusion

In Chapter 4, we reformulate the linear system into a sparse-dense Sylvester linear equation and present an existing technique in the literature to solve the equation. We review both the theoretical and practical aspects of the algorithm. The choice of solver, whether direct or iterative, significantly impacts the computational complexity, with iterative methods offering more scalability for large systems. In this thesis, this solution is implemented as a baseline method for Sylvester-representation approaches. Next, we will propose a novel algorithm to solve the formulated Sylvester equation using block Krylov subspace methods.

5 Lanczos-Sylvester Solver for Scalable Statistical Algorithms

5.1 Summary

In this chapter, we present a matrix-free, Lanczos-based iterative method that leverages the Sylvester formulation of the problem. This method capitalizes on the fact that equations with a Sylvester structure can be efficiently handled using block Krylov subspaces, which benefit from a block generalization of the shift-invariance property inherent to Krylov subspaces. The chapter demonstrates the application of the block Krylov approach to large-scale problems and explores block Lanczos-based methods specifically for symmetric matrices. A theoretical analysis of the proposed algorithm is provided, along with a discussion on how to enhance its computational efficiency in practical applications.

5.2 Lanczos-Sylvester Linear Solver

5.2.1 Application of block Krylov approach for large-scale problems

Large-scale Sylvester-structured problems of sparse-dense type are amenable to solution by generating a *block* Krylov subspace associated to the large coefficient matrix,

i.e., $N^{-1}D^2$. This space at iteration j has the form

$$\begin{aligned}\mathbb{K}_j(N^{-1}D^2, YTA P^{-1}) &:= \mathcal{K}_j(N^{-1}D^2, YTA e_1) \\ &\quad + \mathcal{K}_j(N^{-1}D^2, YTA e_2) + \cdots \\ &\quad + \mathcal{K}_j(N^{-1}D^2, YTA e_m),\end{aligned}$$

where e_i is the i th Cartesian basis vector. The Sylvester equation is projected onto this space, and a smaller Sylvester problem is solved at each iteration. The matrix $N^{-1}D^2$ is no longer symmetric positive definite in the standard inner product; however, it is if we consider a different inner product.

Any symmetric positive-definite matrix can be used to define a new inner product. The matrix N is a diagonal matrix with only positive entries on its diagonal; therefore, it is a positive-definite matrix and can be used to induce the inner product $(x, y)_N = y^T N x$. A matrix G is symmetric with respect to $(\cdot, \cdot)_N$ if for all $x, y \in \mathbb{R}^n$, $(Gx, y)_N = (x, Gy)_N$ i.e. $y^T N G x = y^T G^T N x$.

What is important here is that we exploit the property proven in the following Lemma.

Lemma 1 The coefficient matrix $N^{-1}D^2$ is symmetric with respect to $(\cdot, \cdot)_N$.

Proof: This is straightforward to show by direct computation. For all $x, y \in \mathbb{R}^n$, we have

$$(N^{-1}D^2 x, y)_N = y^T N N^{-1} D^2 x = y^T D^2 x = y^T D^2 N^{-1} N x = (x, N^{-1}D^2 y)_N.$$

Thus the symmetry with respect to $(\cdot, \cdot)_N$ is proven.

Iterative methods designed for symmetric matrices can be converted to methods for matrices symmetric with respect to any inner product simply by using that inner product in the algorithm. We can implement a Conjugate Gradient-like iteration which uses a fixed amount of memory, is amenable to parallelization, and can be implemented to have good data movement characteristics (For problems of this size,

we must concern ourselves with how much data an algorithm moves, as this becomes a dominating cost). Thus it behooves us to elaborate on block Lanczos-based methods for symmetric matrices.

5.2.2 The block Lanczos process with respect to $(\cdot, \cdot)_N$

We mentioned in 2.3.2 that CG is a Krylov subspace method. We elaborate here on some aspects of Krylov subspaces for symmetric coefficient matrices to derive an iterative method for Sylvester equations whose larger coefficient matrix (here $N^{-1}D^2$) is symmetric with respect to $(\cdot, \cdot)_N$.

Let us denote $\hat{Y} = Y T A P^{-1}$ and $\hat{L} = A^T T A P^{-1}$. The symmetric Lanczos process is an iterative procedure to generate an N -orthonormal basis for $\mathbb{K}_j(N^{-1}D^2, \hat{Y})$ block-by-block. What has been shown is that if the matrix is symmetric with respect to the inner product used for orthogonalization, at step i , the $(i + 1)$ st block of vectors generated at that step are already orthogonal with respect to all but blocks i and $i - 1$, meaning the orthogonalization procedure requires we only store the two most recently generated blocks. At step i , we have generated the orthonormal basis $\{V_1, V_2, \dots, V_i\}$ with $V_\ell \in \mathbb{R}^{n \times m}$, $V_\ell^T N V_k = 0$ for $\ell \neq k$, and $V_\ell^T N V_\ell = I_m$, the $m \times m$ identity matrix. To obtain the next basis vector, we compute $W_{i+1} = (N^{-1}D^2)V_i$. Normally, we would then need to orthogonalize these vectors against all previous, but since $N^{-1}D^2$ is symmetric with respect to $(\cdot, \cdot)_N$, it has been proven that $V_\ell^T N W_{i+1} = 0$ for $\ell < i - 1$. The orthogonalization coefficients H_i, B_i are stored in the matrix $\underline{T}_j \in \mathbb{R}^{(j+1)m, jm}$ whose first jm rows are the symmetric, block tri-diagonal matrix $T_j \in \mathbb{R}^{jm \times jm}$. These

have the structure

$$\underline{T}_j = \begin{bmatrix} H_1 & B_2 & & & & \\ B_2 & H_2 & B_3 & & & \\ & B_3 & \ddots & \ddots & & \\ & & \ddots & H_{j-1} & B_j & \\ & & & B_j & H_j & \\ & & & & & B_{j+1} \end{bmatrix} \quad \text{and} \quad T_j = \begin{bmatrix} H_1 & B_2 & & & & \\ B_2 & H_2 & B_3 & & & \\ & B_3 & \ddots & \ddots & & \\ & & \ddots & H_{j-1} & B_j & \\ & & & B_j & H_j & \end{bmatrix},$$

where each block represents an $m \times m$ matrix. The diagonal and super-diagonal blocks represent coefficients that are related to orthogonalization and the subdiagonal blocks represent blocks related to normalization. The symmetry tells us that the normalization block B_{i+1} , generated when normalizing V_{i+1} , can be reused as an orthogonalization coefficient at step $i + 1$, where it can be shown that in exact arithmetic

$$V_{i+1}B_{i+1} = N^{-1}D^2V_i - V_iH_i - V_{i-1}B_i. \quad (5.1)$$

We denote $\mathcal{V}_j = \begin{bmatrix} V_1, V_2, \dots, V_j \end{bmatrix} \in \mathbb{R}^{N \times jm}$ to be the matrix whose columns are the block Lanczos vectors. In terms of the matrix T_j , we can use 5.1 to derive the Lanczos relation, relating $N^{-1}D^2$, $\mathcal{V}_j$, and $\underline{T}_j$,

$$N^{-1}D^2\mathcal{V}_j = \mathcal{V}_{j+1}\underline{T}_j. \quad (5.2)$$

The construction of $\underline{T}_j$ is shown as a part of Algorithm 9; see, e.g., [Saad, 2003b] for further details.

The symmetric Lanczos relation (5.2) follows directly from the setup of this process. Furthermore, since the columns of $\mathcal{V}_j$ form an N orthonormal basis for $\mathbb{K}_j(N^{-1}D^2, \hat{Y})$,

Algorithm 9: Symmetric Lanczos process for iteratively generating a basis for $\mathbb{K}_{j+1}(N^{-1}D^2, \hat{Y})$

Data: : starting block of vectors $\hat{Y} \in \mathbb{R}^{N \times m}$

Result: : $(\cdot, \cdot)_N$ -orthonormal basis $\{V_1, V_2, \dots, V_{j+1}\}$

1 Orthogonalize starting block in $(\cdot, \cdot)_N$ producing QR-factorization:

$$\hat{Y} = \underbrace{V_1}_{\in \mathbb{R}^{N \times m}} \underbrace{B_0}_{\in \mathbb{R}^{m \times m}} \text{ with } V_1^T N V_1 = I_m$$

2 **for** $i = 1, 2, \dots, j$ **do**

3 $W \leftarrow N^{-1}D^2V_i$

4 **if** $i > 1$ **then**

5 $W \leftarrow W - V_{i-1}B_i$

6 **end**

7 $H_i = V_i^T N W$

8 $W \leftarrow W - V_i H_i$

9 Orthogonalize W in $(\cdot, \cdot)_N$ producing QR-factorization: $W = V_{i+1}B_{i+1}$

10 **end**

we have that $\mathcal{V}_j^T N \mathcal{V}_{j+1} = \begin{bmatrix} I_{jm} & 0 \end{bmatrix}$, and one can then show that,

$$\begin{aligned} N^{-1}D^2\mathcal{V}_j &= \mathcal{V}_{j+1}\underline{T}_j \\ \mathcal{V}_j^T N N^{-1}D^2\mathcal{V}_j &= \mathcal{V}_j^T N \mathcal{V}_{j+1}\underline{T}_j \\ \mathcal{V}_j^T D^2\mathcal{V}_j &= \begin{bmatrix} I_{jm} & 0 \end{bmatrix} \underline{T}_j \\ \mathcal{V}_j^T D^2\mathcal{V}_j &= T_j, \end{aligned} \tag{5.3}$$

which means that T_j is symmetric.

Building the basis in blocks with the computational kernel being a matrix-times-matrix product and also block orthogonalization produces an algorithm with computationally attractive properties in terms of computational intensity (i.e., the amount of calculations done per unit of data moved into and out of cache) [Parks et al., 2016], and this is a topic of current study in the context of GPU implementation Boman et al. [2023]. Unlike the classical CG approach, though, it is non-trivial to develop a short-recurrence approach to treat symmetric Sylvester equations. However, an effective, efficient approach has been described in the literature.

5.2.3 Projection methods for Sylvester equations

Simoncini and Palitta developed a solver for large-scale Sylvester problems where both matrices are symmetric [Palitta and Simoncini, 2018, Section 5.2]. We adapt this work to develop a practical method for treating 4.1. Similar to methods for solving linear systems, we can discretize the Sylvester equations by taking approximations from a subspace and solving a constrained set of equations to determine the specific approximation. In this case, the unknown from 4.1 is $M \in \mathbb{R}^{N \times m}$. At step j , we make an approximation of the form

$$M \approx M_j = \mathcal{V}_j Z_j = \sum_{i=1}^j V_i G_i \text{ where } G_i \in \mathbb{R}^{m \times m} \text{ and } Z_j = \begin{bmatrix} G_1 \\ G_2 \\ \vdots \\ G_j \end{bmatrix} \in \mathbb{R}^{jm \times m}. \quad (5.4)$$

This approximation represents a specific trade-off between computational tractability and solution accuracy for large-scale problems. While primarily motivated by computational considerations, our choice of standard block Krylov subspaces is justified for the CMB source separation context. The block structure naturally accommodates the multiple right-hand sides in our problem without requiring separate solutions for each source component. Additionally, the symmetry of $N^{-1}D^2$ with respect to the inner product $(\cdot, \cdot)_N$ allows us to employ the efficient block Lanczos algorithm, which maintains numerical stability while requiring minimal storage compared to alternative projection techniques.

Substituting into 4.1 yields

$$N^{-1}D^2\mathcal{V}_j Z_j + \mathcal{V}_j Z_j \hat{L} \approx \hat{Y}. \quad (5.5)$$

We determine Z_j to be the solution satisfying the equation

$$\mathcal{V}_j^T N(N^{-1}D^2\mathcal{V}_jZ_j + \mathcal{V}_jZ_j\widehat{L}) = \mathcal{V}_j^T N\widehat{Y}. \quad (5.6)$$

Let $E_1 \in \mathbb{R}^{jm \times m}$ be the matrix with the identity matrix in the first $m \times m$ block and zeros elsewhere. Observing that $\widehat{Y} = V_1B_0 = \mathcal{V}_jE_1B_0$ and using 5.3, we can simplify 5.6 as

$$T_jZ_j + Z_j\widehat{L} = E_1B_0. \quad (5.7)$$

The task has now become to solve a small, fairly dense set of Sylvester equations. We can use standard methods to treat this problem.

We approach 5.7 by decomposing the matrices T_j and $\widehat{L}$. The matrix T_j is symmetric, block-tridiagonal. Thus, we can efficiently compute the eigendecomposition $T_j = Q_j\Phi_jQ_j^T$ where $Q_j^TQ_j = I_j$ and $\Phi_j = \text{diag}\{\phi_1, \phi_2, \dots, \phi_j\}$ is the diagonal matrix of eigenvalues of T_j . The matrix $\widehat{L}$ is not symmetric with respect to the standard Euclidean inner product. However, we prove it is symmetric with respect to another inner product.

Lemma 2 The matrix $\widehat{L} = A^T T A P^{-1}$ is symmetric with respect to the inner product induced by P^{-1} .

Proof. The matrix P is a diagonal matrix with positive entries on the diagonal. Thus it and its inverse are symmetric positive-definite, and P^{-1} induces the inner product $(\cdot, \cdot)_{P^{-1}}$, with $(x, y)_{P^{-1}} = y^T P^{-1} x$. We can show by direct calculation that $\widehat{L}$ is symmetric with respect to $(\cdot, \cdot)_{P^{-1}}$,

$$(A^T T A P^{-1} x, y)_{P^{-1}} = y^T P^{-1} A^T T A P^{-1} x = (x, A^T T A P^{-1} y)_{P^{-1}}.$$

Since $\widehat{L}$ is symmetric with respect to $(\cdot, \cdot)_{P^{-1}}$, it is diagonalizable and its eigenvectors

form an orthonormal basis with respect to $(\cdot, \cdot)_{P^{-1}}$. Thus, we can write

$$\widehat{L} = URU^{-1} = URU^T P^{-1}$$

with $R = \text{diag}\{\rho_1, \rho_2, \dots, \rho_m\}$ being a diagonal matrix of eigenvalues, and $U^T P^{-1} U = I_m$. This decomposition can be computed once at the beginning of execution. Inserting this into 5.7, we get:

$$\begin{aligned} Q_j \Phi_j Q_j^T Z_j + Z_j URU^{-1} &= E_1 B_0 \\ \iff \Phi_j Q_j^T Z_j U + Q_j^T Z_j UR &= Q_j^T E_1 B_0 U \\ \iff \Phi_j \widehat{Z}_j + \widehat{Z}_j R &= Q_j^T E_1 B_0 U, \end{aligned} \tag{5.8}$$

where $\widehat{Z}_j = Q_j^T Z_j U \iff Z_j = Q_j \widehat{Z}_j U^{-1} = Q_j \widehat{Z}_j U^T P^{-1}$. Vectorizing 5.8, we get

$$(R \otimes I + I \otimes \Phi_j) \text{vec}(\widehat{Z}_j) = \text{vec}(Q_j^T E_1 B_0 U).$$

Observe that $(R \otimes I + I \otimes \Phi_j)$ is a diagonal matrix with diagonal entries of the form $\phi_i + \rho_k$. It follows that just as in [Palitta and Simoncini, 2018, Section 5.2], one can directly compute the solution

$$Z_j = Q_j F_j U^T P^{-1} \text{ where } F_j := (F_{ik}) = \frac{e_i^T Q_k^T E_1 B_0 U e_k}{\phi_i + \rho_k} \in \mathbb{R}^{j \times m}. \tag{5.9}$$

Remark 1 Although T_j contains T_{j-1} as its upper-left $(j-1)m \times (j-1)m$ submatrix, the eigendecomposition changes completely and must be recomputed at every iteration.

To make this approach computationally efficient, we must forego computing the solution approximations during the iteration. Rather, we run this iteration and use a cheap residual norm approximation to monitor convergence. In [Palitta and Simoncini,

2018, Section 5.2], it is shown how to calculate an expression for the residual norm without forming the solution explicitly. This done via accumulation, and in [Palitta and Simoncini, 2018, Algorithm 5], we see that, in principle, one needs to calculate the eigendecomposition of T_i at each iteration i . However, the authors discuss in [Palitta and Simoncini, 2018, Section 3.2] how one can calculate only the needed entries of the eigenvectors to calculate the residual norm. We note that, in our code, we do not take advantage of the trick and simply compute the full eigendecomposition. This is what we show in our description of the procedure, Algorithm 10. Once the residual norm estimate reaches tolerance, the iteration can be stopped, and we construct the approximate solution. We construct Z_j and rerun the Lanczos process to get the vectors again for constructing the solution, one block Lanczos vector at a time. This is because, from 5.4, we can construct the approximation M_j block-for-block meaning when we rerun the Lanczos iteration, we can use the blocks as they are constructed and then discard them, retaining the memory efficiency of Algorithm 10.

Remark 2 In Algorithm 10 we do not perform any low-rank compression of the block Lanczos vectors. In our experiments, we determined that the block Lanczos vectors and the residuals generated when running Algorithm 10 remained numerically full rank (i.e., rank 4). Thus, we do not pursue a strategy of compression in our Sylvester solver.

5.3 Solver Analysis

5.3.1 Theoretical Analysis

One of the key strengths of the Lanczos-Sylvester algorithm lies in its ability to exploit the **block structure** of Krylov subspaces. In large-scale Sylvester-structured problems, the sparse matrix $N^{-1}D^2$ represents the large coefficient matrix, and its structure allows for efficient projection onto block Krylov subspaces. This projection significantly reduces the dimensionality of the problem at each iteration, transforming

Algorithm 10: Symmetric Sparse Sylvester Solver

Data: Starting block of vectors $\hat{Y} \in \mathbb{R}^{N \times m}$; coefficient matrix $N^{-1}D^2 \in \mathbb{R}^{N \times N}$; a small coefficient matrix $\hat{L} \in \mathbb{R}^{m \times m}$; $\varepsilon > 0$ a convergence tolerance

Result: An approximation $M \in \mathbb{R}^{N \times m}$ to the solution of the Sylvester equation

```
1 Compute eigendecomposition  $\hat{L} = URU^T$ 
2 Orthogonalize starting block in  $(\cdot, \cdot)_N$  producing QR-factorization:  $\hat{Y} = VB_0$ 
3  $V_{old} = 0$ ;  $i = 0$ 
4 while  $\sqrt{\gamma} > \varepsilon B_{0F}$  do
5      $i \leftarrow i + 1$  /* Lanczos step */
6      $W \leftarrow N^{-1}D^2V$ 
7     if  $i > 1$  then
8          $W \leftarrow W - V_{old}B_i$ 
9     end
10     $H_i = V^T N W$ 
11     $W \leftarrow W - V H_i$ 
12     $V_{old} \leftarrow V$ 
13    Orthogonalize  $W$  in  $(\cdot, \cdot)_N$  producing QR-factorization:  $W = V B_{i+1}$ 
14     $\gamma \leftarrow 0$ 
15    Compute eigendecomposition  $T_i = Q_i \Phi_i Q_i^T$ 
16     $S \leftarrow (U^T B_0^T)(E_1^T Q_i)$ 
17     $J \leftarrow (Q_i E_m) B_{i+1}^T$ 
18    for  $j = 1, 2, \dots, m$  do
19         $K \leftarrow \rho_j I + \Phi_i$ 
20         $\gamma \leftarrow \gamma + (e_j^T S K^{-1} J)^2$ 
21    end
22 end
    /* constructing approximation coefficients entry-wise ***** */
23  $F = 0 \in \mathbb{R}^{i \times m}$ 
24 for  $k = 1, 2, \dots, i$  do
25     for  $\ell = 1, 2, \dots, m$  do
26          $F_{k\ell} = \frac{e_k^T Q_\ell^T E_1 B_0 U e_\ell}{\phi_k + \rho_\ell}$ ; see 5.9
27     end
28 end
29  $Z = Q_i F U^T$ 
    /* rerun Lanczos to assemble solution ***** */
30  $M = 0$ 
31  $V \leftarrow \hat{Y} B_0^{-1}$ 
32  $V_{old} = 0$ 
33 for  $j = 1, 2, \dots, i$  do
34      $M \leftarrow M + V G_j$ , according to (5.4)
    /* Lanczos step using already-computed coefficients ***** */
35      $W \leftarrow N^{-1}D^2V$ 
36     if  $j > 1$  then
37          $W \leftarrow W - V_{old}B_j$ 
38     end
39      $W \leftarrow W - V H_j$ 
40      $V_{old} \leftarrow V$ 
41      $V \leftarrow W B_{j+1}^{-1}$ 
42 end
43 Return  $M$ 
```

it into a smaller Sylvester problem that can be solved more easily.

The core of the theoretical advantage stems from the fact that the matrix $N^{-1}D^2$ is symmetric with respect to a specific inner product induced by the diagonal matrix N . This symmetry allows the algorithm to employ the **block Lanczos process**, generating an orthonormal basis for the Krylov subspace that grows block by block. Each block only needs to be orthogonalized with respect to the two most recent blocks, reducing the computational burden associated with full orthogonalization, a feature that ensures efficient memory usage. Moreover, the algorithm avoids forming large, dense matrices by working within the Krylov subspace, where smaller matrices like the tridiagonal block matrix T_j are constructed iteratively. The reduced dimensionality of this space accelerates convergence while maintaining numerical stability.

Another notable advantage of the Lanczos-Sylvester solver is its **matrix-free implementation**, meaning that the algorithm does not explicitly form or store the full matrices involved in the Sylvester equation. Instead, it relies on matrix-vector and matrix-matrix products to iteratively generate the required Krylov subspace vectors. This feature is particularly beneficial when working with large-scale problems, as it drastically reduces memory usage and computational overhead. The matrix-free design also improves **data movement efficiency**, a critical factor for large-scale problems where the cost of data movement can outweigh the computational costs. By limiting the number of vectors that need to be stored and reused, the algorithm minimizes the number of cache misses and optimizes the use of available memory.

The algorithm's robustness is rooted in its use of a **generalized inner product** and **block-wise orthogonalization**, which ensure stable convergence even in the presence of ill-conditioned matrices. Unlike traditional Krylov subspace methods that may struggle with non-symmetric systems, the Lanczos-Sylvester solver adapts to these challenges by incorporating a custom inner product. This flexibility allows the algorithm to handle a wide range of problems, including those where the matrix $N^{-1}D^2$ is non-symmetric with respect to the standard Euclidean inner product but symmetric

with respect to the inner product defined by N .

The Lanczos process used in this algorithm generates a tridiagonal block matrix T_j , which is symmetric and can be diagonalized efficiently. This symmetry guarantees that the generated Krylov subspaces provide an accurate approximation to the solution of the Sylvester equation. Furthermore, the algorithm’s use of block recurrence ensures that it can be implemented with a fixed memory footprint, a crucial feature for scalability in large-scale problems.

5.3.2 Practical Analysis

From a practical perspective, the Lanczos-Sylvester algorithm provides significant improvements over traditional solvers, making it particularly suitable for large-scale problems such as the CMB source separation task. The Lanczos-Sylvester algorithm is particularly well-suited to this task because of the following properties:

- **Block Krylov Subspace Reduction:** The use of block Krylov subspaces effectively reduces the dimensionality of the problem at each iteration. In CMB analysis, the large matrix $N^{-1}D^2$ corresponds to sparse data structures that describe the mixing of signals across different regions of the sky. By projecting the problem onto a lower-dimensional Krylov subspace, the algorithm efficiently handles these high-dimensional matrices without explicitly forming them.
- **Matrix-Free Computation:** In the context of CMB data, storing the full matrix $N^{-1}D^2$ is impractical due to its size. The matrix-free implementation of the Lanczos-Sylvester solver avoids this issue by performing only matrix-vector products. This reduces both memory and computational costs, allowing the algorithm to scale efficiently to the large datasets typically encountered in cosmology.
- **Scalability:** The algorithm’s ability to maintain a fixed memory footprint, regardless of the number of iterations, ensures that it can be applied to extremely large datasets. In CMB analysis, where millions of data points are processed simultaneously, this scalability is crucial for producing results in a reasonable

time frame.

Beyond CMB source separation, the Lanczos-Sylvester solver is highly versatile and can be applied to various statistical algorithms that involve large-scale Sylvester equations. Its use of block Krylov subspaces and block-wise orthogonalization makes it particularly effective for high-dimensional problems in fields such as signal processing, image reconstruction, and machine learning. The solver's ability to handle non-symmetric systems through generalized inner products broadens its applicability to challenges where traditional methods often fail, such as fluid dynamics or financial modeling. Additionally, its matrix-free implementation and efficient data movement make it ideal for parallel and distributed computing environments, ensuring scalability across a wide range of large-scale computational problems in fields like engineering simulations and computational biology. These features collectively enhance the solver's robustness and efficiency, making it a powerful tool for addressing diverse statistical and computational challenges.

5.3.3 Comparisons with Existing Techniques

The Lanczos-Sylvester solver offers several significant improvements over the more traditional approaches to solving sparse-dense Sylvester equations, such as the method presented in Chapter 4. Below, we elaborate on the key advantages of the Lanczos-Sylvester algorithm in terms of memory efficiency, convergence speed, and scalability, providing a detailed comparison with existing techniques.

Improved Memory Efficiency

One of the major advantages of the Lanczos-Sylvester solver is its superior memory efficiency, particularly when compared to traditional sparse-dense Sylvester solvers. The method discussed in Chapter 4, which employs direct or iterative techniques without leveraging the block Krylov subspace structure, requires storing large matrices and intermediate results, leading to significant memory overhead. For example, the direct solver must store the entire sparse matrix A , as well as dense matrices H and M , and

repeatedly operate on them.

In contrast, the Lanczos-Sylvester solver operates in a **matrix-free** manner, meaning it never explicitly forms the full matrices involved in the Sylvester equation. Instead, it computes matrix-vector and matrix-matrix products iteratively within a reduced-dimensional Krylov subspace, which drastically reduces the need for memory storage. Only the orthonormal basis vectors generated by the block Lanczos process are stored, and since the process requires only a fixed number of these vectors (due to **block-wise orthogonalization**), the memory requirements remain constant as the algorithm progresses. This is particularly important in applications like CMB source separation, where the matrices can be extremely large, and memory constraints are a major concern. By reducing memory usage, the Lanczos-Sylvester solver can handle larger problems on machines with limited memory resources.

Faster Convergence

Another significant improvement of the Lanczos-Sylvester solver is its ability to achieve faster convergence compared to existing techniques. In traditional solvers, convergence rates can be slower due to the lack of exploitation of any underlying matrix structure. For example, direct solvers or basic iterative solvers like Conjugate Gradient must perform full matrix-vector operations at each iteration, and their convergence rates can suffer, especially for ill-conditioned or large systems.

The Lanczos-Sylvester algorithm, by contrast, benefits from the block Krylov subspace projection and the Lanczos process, which takes advantage of the inherent symmetry in the problem (with respect to a custom inner product). This projection onto a smaller Krylov subspace significantly reduces the dimensionality of the problem at each iteration, accelerating convergence. Additionally, the block Lanczos process efficiently handles the block structure of the problem, allowing for multiple vectors to be updated simultaneously. This reduces the number of iterations needed to achieve a given accuracy, particularly in large-scale systems like those encountered in CMB source separation.

Moreover, the use of block orthogonalization ensures that the algorithm remains numerically stable even when dealing with ill-conditioned systems, a scenario where traditional solvers may struggle. This leads to faster and more reliable convergence, making the Lanczos-Sylvester solver more robust in practice.

Better Scalability

Scalability is another area where the Lanczos-Sylvester solver significantly outperforms existing techniques. The method presented in Chapter 4, although effective for smaller systems, struggles to scale efficiently to extremely large datasets due to its reliance on full matrix operations and the need to store large, dense matrices. Direct solvers like LU decomposition, for example, have cubic complexity in the size of the matrix, making them impractical for very large systems.

The Lanczos-Sylvester solver, on the other hand, is designed to scale efficiently with problem size. By projecting the problem onto a lower-dimensional Krylov subspace and working within this reduced space, the algorithm avoids the computational and memory bottlenecks that plague traditional solvers. Its matrix-free design allows it to handle extremely large datasets, such as those found in CMB analysis, without requiring excessive memory or computational resources. The block Lanczos process further enhances scalability by allowing the solver to work with blocks of vectors rather than individual vectors, thus reducing the overall number of iterations required for convergence.

Additionally, the Lanczos-Sylvester solver is well-suited for parallel and distributed computing environments. Unlike traditional solvers that may require complex data dependencies or synchronization between processors, the block Krylov subspace method is inherently parallelizable. The matrix-vector products that form the core of each iteration can be distributed across multiple processors, allowing the algorithm to take full advantage of modern multi-core or distributed architectures. This scalability is particularly important in cosmological data analysis, where the size of the datasets continues to grow, and computational efficiency is paramount.

Exploitation of Matrix Structure

The Lanczos-Sylvester solver also stands out in its ability to exploit the structure of the matrix in the Sylvester equation. In Chapter 4, the solver treats the sparse-dense nature of the equation in a fairly straightforward manner, without fully leveraging the potential benefits of the block structure or any symmetry present in the system. While such approaches may work for moderate-sized systems, they do not fully take advantage of the opportunities for optimization presented by the problem’s structure.

In contrast, the Lanczos-Sylvester solver explicitly exploits the block structure and symmetry with respect to a custom inner product to improve both memory efficiency and computational speed. The use of block Krylov subspaces allows the algorithm to process multiple vectors simultaneously, and the generalized inner product ensures that the matrix operations remain well-conditioned. This ability to handle non-standard inner products and block structures sets the Lanczos-Sylvester solver apart from more traditional methods, making it more robust and efficient for large-scale statistical algorithms.

5.4 Conclusion

In this chapter, we introduced the Lanczos-Sylvester solver, a matrix-free, block Krylov subspace-based iterative method designed to efficiently solve large-scale Sylvester equations. By leveraging the block structure and symmetry of the matrices involved, the solver achieves significant improvements in memory efficiency, computational speed, and scalability compared to traditional methods. Its ability to exploit the custom inner product induced by the matrix structure makes it particularly robust for solving high-dimensional problems like the Cosmic Microwave Background (CMB) source separation.

Theoretical analysis demonstrated the solver’s advantages in terms of convergence and computational complexity, while practical analysis highlighted its suitability for large-scale applications in both CMB data analysis and other statistical algorithms. The

block Krylov approach enables faster convergence with reduced memory usage, and its matrix-free nature ensures scalability across parallel and distributed computing environments.

In the next chapter, Chapter 6: Experimental Results, we will present a detailed evaluation of the Lanczos-Sylvester solver in comparison with the Conjugate Gradient method and the baseline sparse-dense Sylvester approach through a series of numerical experiments. These experiments will validate the theoretical claims made in this chapter and compare the solver’s performance to existing methods, focusing on aspects such as accuracy, time complexity, memory usage, and overall efficiency in large-scale applications.

6 Experimental Results

6.1 Summary

We have proposed using two approaches described in the literature for solving this source separation problem by exploiting the Kronecker structure of the linear system. In the first method, we employ conjugate gradients applied directly to the Kronecker form of the system, and the structure is exploited to obtain a matrix-free iteration. In the latter method, we similarly exploit the Sylvester formulation of the problem using a Lanczos-based iterative method, and we are again able to do this in a matrix-free way.

In this chapter, we demonstrate with our numerical results that both approaches are viable for the underlying application problem. For comparison purposes, we implement sparse-dense Sylvester solver introduced in Chapter 4 as a baseline method.

6.2 Experimental Setup

6.2.1 Experimental Design

Three implementations (conjugate gradient method and two Sylvester approaches) are applied to **both** *simulated data* of different sizes (allowing for evaluation of the actual error) as well as *actual Planck full sky maps*. Results are compared in terms of time cost, memory usage, and accuracy. Code profiling explores how these requirements for different aspects of the algorithm change with problem size.

The experiment is designed as follows. The 3 solutions were run with simulated data having different values of N_{side} . Since the source S is specified in each simulation, this allows us to compare the results of the algorithms with the true solution of the simulated problem. The problem size is specified as the HEALPix level from 1 to 10. Recalling that the solution dimension at level h is $m \times N_{side} = m \times 12 \times 4^h$, hence increasing the level by 1 increases the dimension of the solution by 4.

For the simulated and Planck data, $m = 4$ sources and $n = 9$ images were assumed, as discussed in Chapter 3. Data were simulated by specifying values for the source vector $\mathcal{S}$ (for simplicity, these are generated as independent samples from a normal distribution), and the matrix A and vector T were defined as in Section 3.2.2. That allows the data vector $\mathcal{Y}$ to be randomly generated as in Equation 3.1.

6.2.2 Input Preparation

For the implementation of the three algorithms, several input parameters are required:

- Number of sources $m = 4$, number of images (or frequencies) $n = 9$ as defined in the previous section.
- Mixing matrix A and precision matrix T as defined in Chapter 3
- The adjacency matrix D .
- Input vector y aggregated from the Planck dataset.

The following sections described the preparation of the mentioned input parameters.

Adjacency Matrix D

The adjacency matrix D encodes a nearest neighbor coupling of nodes from the discretization of the sky.

From Section 3.3, D is defined as:

$$D = (D_{ij}) \text{ with } D_{j_1, j_2} = \begin{cases} 1 & \text{nodes } j_1 \neq j_2 \text{ are neighbors} \\ 0 & \text{nodes } j_1 \neq j_2 \text{ are not neighbors} \\ -\sum_{\substack{\ell=1 \\ \ell \neq j}}^N D_{j\ell} & j := j_1 = j_2 \end{cases} \quad (6.1)$$

When the level N_{side} is 1, D is an adjacency matrix of the following matrix:

$$L_1 = \begin{bmatrix} 0 & 3 & 6 & 9 \\ 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \end{bmatrix} \quad (6.2)$$

The values in L_1 represents the indexes of matrix D . Hence, we have the following adjacency matrix D when $N_{side} = 1$

$$D_1 = \begin{bmatrix} -2 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -3 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -3 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -3 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -3 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2 \end{bmatrix}$$

In our experiment, the size of adjacency matrix D is $N_{side} = 12 \times 4^h, h = 1 \dots 10$. Therefore, it would not be explicitly built in the algorithm. Instead, we implemented

a procedure to retrieve any row from D easily. The procedure is described in Algorithm 11.

Algorithm 11: RowView of matrix D

Data: $rowIdx$ as the row index; Nf_{side} of fits file

Result: $neighbors$ as a index list of all the neighbor nodes

```

1  $row = 3 \times Nf_{side}$ 
2  $col = 4 \times Nf_{side}$ 
3  $neighbors \leftarrow \{rowIdx - row, rowIdx + row, rowIdx + 1, rowIdx - 1\};$ 
4  $/* rowIdx$  is in the first row  $*/$ 
5 if  $rowIdx \% row == 0$  then
6   |  $neighbors.remove(rowIdx - 1)$ 
7 end
8  $/* rowIdx$  is in the last row  $*/$ 
9 if  $rowIdx == row - 1$  then
10  |  $neighbors.remove(rowIdx + 1)$ 
11 end
12  $/* rowIdx$  is in the first column  $*/$ 
13 if  $rowIdx < row$  then
14  |  $neighbors.remove(rowIdx - row)$ 
15 end
16  $/* rowIdx$  is in the last column  $*/$ 
17 if  $rowIdx > row \times col - 1 - row$  then
18  |  $neighbors.remove(rowIdx + row)$ 
19 end
20 return  $neighbors$ 

```

Input vector y

For the experiments conducted in this research, we use the Planck All-Sky Maps obtained from the Planck Legacy Archive [Aghanim et al., 2020], which includes observations from the full mission. The data is stored in the form of HEALPix maps, which allow for efficient and uniform representation of the spherical sky.

The all-sky maps in the archive are provided for each of the nine frequency channels used by the Planck satellite:

- 30 GHz, 44 GHz, 70 GHz (Low Frequency Instrument - LFI)
- 100 GHz, 143 GHz, 217 GHz, 353 GHz, 545 GHz, 857 GHz (High Frequency Instrument - HFI)

The data is stored in FITS (Flexible Image Transport System) format and can be accessed as full-mission maps or broken down into survey maps (Survey 1 to Survey 9 for LFI, Survey 1 to Survey 5 for HFI). Additionally, each frequency channel is accompanied by polarization maps, representing the Stokes Q and U parameters, which describe the linear polarization of the CMB radiation.

The maps represent the observed sky in units of thermodynamic temperature for the frequencies 30 GHz to 353 GHz (expressed in K_{CMB}), and in MJy/sr for the higher frequencies, 545 GHz and 857 GHz. Conversion factors between these units are provided in the explanatory supplement accompanying the dataset.

Algorithm 12: Building the input vector y

Data: Nf_{side} of fits file, 9 Planck full sky maps downloaded from the Planck Legacy Archive

Result: y

```

1  $y = 0$ 
2 for  $f = 1 \dots 9$  do
3    $curmap = frequency\_maps[f]$ 
4   for  $patch = 1 \dots 12$  do
5      $temp\_vector \leftarrow extract(curmap, patch)$ 
6      $y \leftarrow append(y, temp\_vector)$ 
7   end
8 end

```

Algorithm 9 shows the process of building the vector y . In the fourth step, we extract each base from a Planck full sky map using *cfitsio* library [Pence, 1999]. As described above, Planck All Sky Maps are in HEALPix format with minimum $Nf_{side} = 1024$, and each map contains 12 different patches, and we can extract a vector of length $12 \times 4^{10} = 12,582,912$ from each map at the highest level $h = 10$. Then, these 9 vectors were concatenated together, resulting in a vector of length $9 \times 12,582,912 = 113,246,208$.

Environmental Setup

Code was written in Python 3.10 and deployed to a high performance computing cluster with specification described in Table 6.1.

Property	Value
Architecture	64
Number of Nodes	100
Ram per node	24GB
RAM	2.4TB
Clock Speed	2.66GHz
Total number of cores	1200
Theoretical Peak Performance	12.76TF
Operating System	Scientific Linux 6.x
Allocated Memory	50GB
Allocated Nodes	2
Allocated Cores per Node	12

Table 6.1: Specification of the machine used in the experiments.

6.3 Evaluation and Comparisons

6.3.1 Evaluation Methodology

Accuracy Evaluation on Simulated Data

The accuracy of the solvers is evaluated using simulated data, where the true source vector $\mathcal{S}$ is known. The simulated datasets are generated with varying levels of complexity (specified by HEALPix levels from 1 to 10). Each simulation provides a ground truth for comparison, allowing us to measure the accuracy of the solvers by calculating the residual error between the true solution and the computed solution. The accuracy is quantified using the largest absolute error across all components of the source vector divided by the average value of the source vector. This metric ensures that the accuracy assessment is independent of the data scale.

Performance Evaluation on Planck Data

For real-world evaluation, we apply both solvers to the Planck full-sky maps. Here, we focus on two main performance metrics:

- **Time Complexity:** The runtime of each solver is measured across different HEALPix levels and corresponding data sizes. The relationship between runtime

and problem size is assessed to determine the scalability of each method.

- **Memory Usage:** Memory consumption is a critical factor when dealing with large datasets. The total memory usage is recorded at different stages of the execution to understand the resource requirements of each solver. Memory profiling helps identify whether the solvers remain efficient in terms of space complexity as the data size grows.

6.3.2 Evaluation Results

Accuracy

Figure 6.1 (left) demonstrates the accuracy of our proposed methods by comparing the solutions obtained from both the Conjugate Gradient and Lanczos-Sylvester methods against the true source vector values from random-generated data. Using simulated data with HEALPix level $h = 4$ (corresponding to $N_{side} = 3072$), we observe that both methods produce solutions that closely align with the ground truth along the diagonal line, indicating excellent accuracy. Quantitatively, both methods achieve similar performance with an error margin of approximately 0.4% (defined as the largest absolute error across all components of the source vector divided by the average value of the source vector). The near-perfect overlap between the two solution methods confirms the numerical consistency of our approaches despite their different theoretical foundations. We observed comparable accuracy across various data sizes, confirming the robustness of our methods regardless of problem scale.

We attempt to interpret the pattern of residual values as shown in the right plot of Figure 6.1. Upon closer inspection, there does not appear to be anything particularly unusual or anomalous about the residual pattern. No obvious properties of the data seem to account for the peculiar shape observed. The residual values, while displaying a certain pattern, do not suggest that any specific aspect of the data, such as noise or structure, is causing the irregularity. This suggests that the observed pattern is likely a byproduct of the solver’s behavior rather than an inherent issue with the dataset

itself.

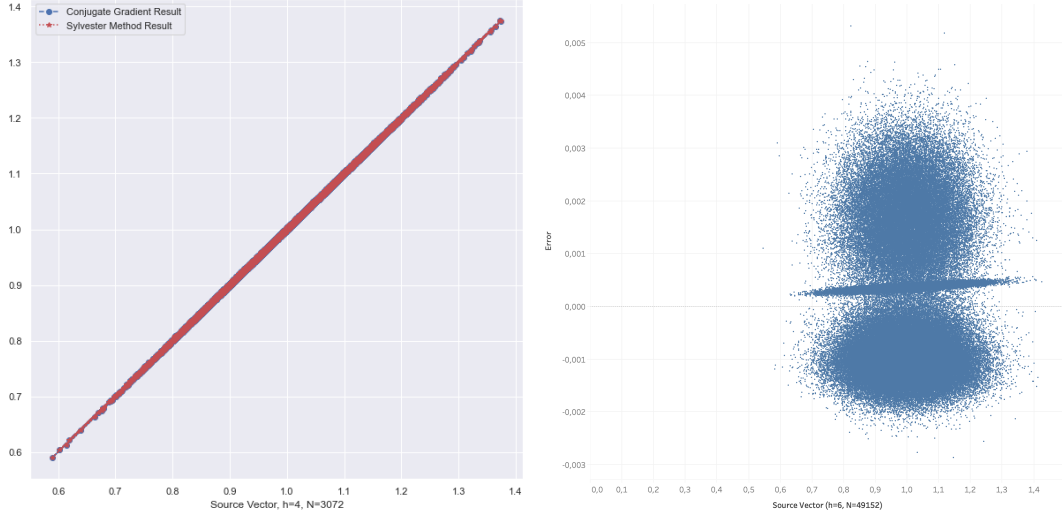


Figure 6.1: Solution accuracy. Scatter plot of components of the true source vector $\mathcal{S}$ versus the solution vector (left) and source vector versus the error (right).

Level	Planck Data (s)		Random Data (s)	
	CG	Lanczos-Sylvester	CG	Lanczos-Sylvester
1	0.002876	0.003313	0.002176	0.003922
2	0.006777	0.004710	0.005430	0.005069
3	0.016987	0.011501	0.013676	0.008950
4	0.047009	0.011501	0.031732	0.008950
5	0.113162	0.023558	0.066272	0.032834
6	0.374143	0.159481	0.295894	0.097409
7	1.574457	0.377765	1.191593	0.489172
8	10.260180	1.566114	10.006086	2.026547
9	53.972078	8.042352	48.300501	10.733038
10	422.819917	70.930965	426.981524	85.438139

Table 6.2: Running time comparisons between CG method and Lanczos-Sylvester method (seconds)

Time and memory usage

Table 6.2 and Figure 6.2 shows the time cost (in seconds) while running the Conjugate Gradient and Lanczos-Sylvester methods on simulated and Planck data, as a function of HEALPix level and the data size; Section 3.2.1 HEALPix level h means a data size of 4^h , hence signifies log data size. The difference between Planck and simulated data is due to the longer time that it took to load in the Planck data from a file than it took

to simulate data of the same size. However, the main difference is that the *Sylvester method is consistently faster, taking only about 15% of the time of the Conjugate Gradient approach for the larger data sizes*. In both cases, the algorithm demonstrates a close to linear complexity with the data size N , a substantial improvement over exact solution methods that are at least quadratic with N . We hypothesize that the superior performance of the Sylvester block Krylov method over the CG method applied directly to the Kronecker-structured problem is due both to conditioning issues reported in [Palitta and Simoncini, 2020b] as well as the superior data movement characteristics (higher computational intensity) often exhibited by block-Krylov-subspace-based methods; see, e.g., Baker et al. [2006].

Table 6.3 and Figure 6.3 show the total memory usage (in GB) while running the Conjugate Gradient and Lanczos-Sylvester methods on simulated and Planck data. We see that the Sylvester method’s memory requirements are about half of that of the Conjugate Gradient, the former needing about 46GB for HEALPix level 10, which is the size of the Planck satellite data.

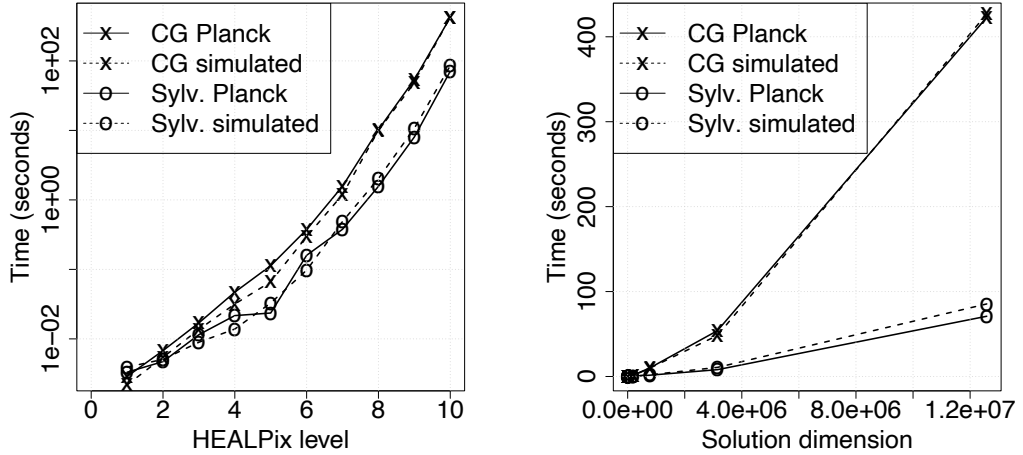


Figure 6.2: Running time (in seconds) by HEALPix level (left) and data size (right).

Comparisons of two Sylvester methods

As described in Chapter 4, we construct the sparse-dense form of the Sylvester equation and apply Algorithm 8 to the CMB source separation problem on Planck dataset. For

Level	Planck Data (GB)		Random Data (GB)	
	CG	Lanczos-Sylvester	CG	Lanczos-Sylvester
1	0.000601	0.000595	0.000559	0.000539
2	0.001505	0.001008	0.001285	0.000999
3	0.005181	0.003274	0.004216	0.003304
4	0.023935	0.011777	0.016631	0.010311
5	0.093979	0.032595	0.064889	0.039415
6	0.376907	0.206982	0.308007	0.153986
7	1.513498	0.630464	1.258506	0.739971
8	6.007464	2.476987	5.620569	2.906787
9	24.183870	10.058349	22.614844	11.755381
10	95.982568	45.844961	103.146484	46.228243

Table 6.3: Total memory usage comparisons between CG method and Lanczos-Sylvester method (GB)

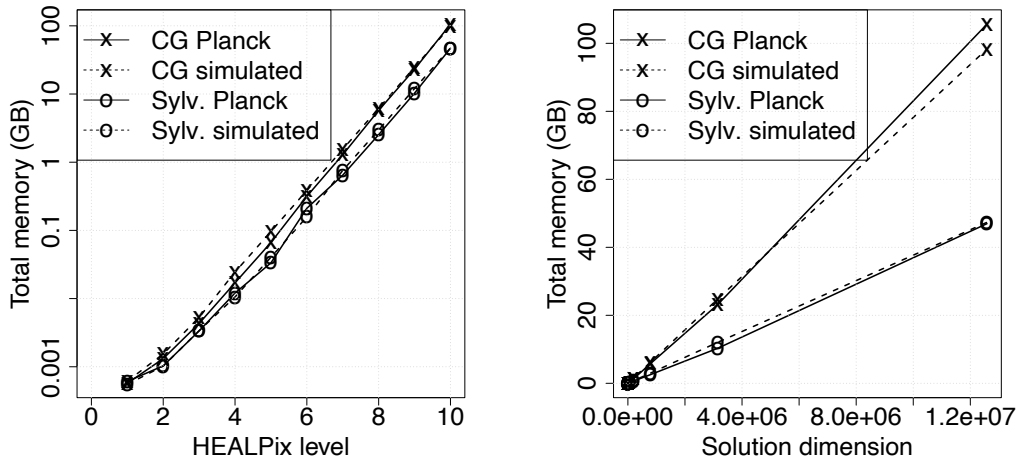


Figure 6.3: Total memory usage (in GB) by HEALPix level (left) and data size (right).

this dataset, $m = 4$ and $N = 12 \times 4^h, h = 1, 2, \dots, 10$. For the largest size of the problem, $\mathbf{H} \in \mathbb{R}^{12,582,912 \times 12,582,912}$, wherein we remind the reader that this matrix is the large, sparse matrix appearing in Equation 4.1.

The results in terms of accuracy and running time of our Lanczos-Sylvester approach in Algorithm 10 and the approach from Chapter 4, (i.e., Algorithm 8) are shown in 6.4 a and b, respectively. The Sparse-Dense method shows significant advantages in accuracy with a comparable running time. However, we observed that for larger Planck data sizes, the solve time for Algorithm 8 grew quadratically. Further investigation revealed that unlike for the problems discussed in [Benner et al., 2011], the SuperLU

Level (h)	N	Lanczos-Sylvester	Sparse-Dense Method
1	48	1.72e−06	2.23e−10
2	192	4.21e−06	3.28e−10
3	768	7.02e−06	4.73e−10
4	3072	4.86e−06	6.19e−10
5	12288	8.62e−06	5.85e−09
6	49152	1.08e−06	2.19e−10
7	196608	1.35e−06	2.6e−09
8	786432	1.58e−06	1.93e−09
9	3145728	1.74e−06	1.21e−09
10	12582912	4.23e−06	—

(a) Residual Comparison

Level (h)	N	Lanczos-Sylvester	Sparse-Dense Method
1	48	0.003	0.001
2	192	0.004	0.006
3	768	0.012	0.035
4	3072	0.021	0.3
5	12288	0.023	0.755
6	49152	0.159	4.35
7	196608	0.377	37.312
8	786432	1.566	301.698
9	3145728	8.042	6948.4228
10	12582912	70.930	—

(b) Running Time Comparison

Table 6.4: Comparison using Planck Data (with $m = 4$ sources), between the sparse-dense method introduced in Chapter 4 and the Lanczos-Sylvester method we proposed in 5.

inner linear solver was not effective at reducing solve time growth rate. We note that we did try different sparse direct solvers (UMFPACK, SuperLU) and an iterative method (Conjugate Gradient), all within the Scipy Python library [Virtanen et al., 2020]. The SuperLU solver proved to be the most *efficient* and accurate one for our specific data. We tried multiple approaches with this code to reduce the growth in run-time, but the growth in runtime remained quadratic. In comparison, our method is demonstrated to be efficient, even for the full size of the Planck dataset.

Function	Description
Calculate Dv	Implementation of Algorithm 2.
Find neighbours	Identify neighbouring pixel indices in the HEALPix partition.
Runtime iterator	A system function to determine the next parallelized item.
Key-value assignment	A system function to assign a value to a key (string).
Variable initialization	Define all variables for the problem and allocate memory.
Load y vector	Self explanatory.
Main solution	Managing the main loop of code.
Matvec products	Implementation of the matrix-vector operations.
Sylvester solution	Main while loop of Algorithm 10.
Write solution	Self explanatory.

Table 6.5: Definition of functions listed in Figure 6.4.

6.3.3 Code Profiling

Figure 6.4 is a break down of the computation time and memory use, as a function of problem size, by the functions in the code that used the most of these resources. A description of these functions is given in 6.5. For time, the top 4 functions at HEALPix level 10 are plotted, while for memory there were only 3 functions that accounted for almost all of the memory use. For memory, unsurprisingly it is the functions that load or make use of all of the data that dominate resources. For time, we see that mapping and computation around the D matrix dominate the computation time. This suggests that efforts to further speed up the algorithm should focus on this task. Encouragingly, operations around the D matrix are well suited to parallelisation, as explained in Algorithm 11.

Details about the code profiling process can be found in Appendix B.

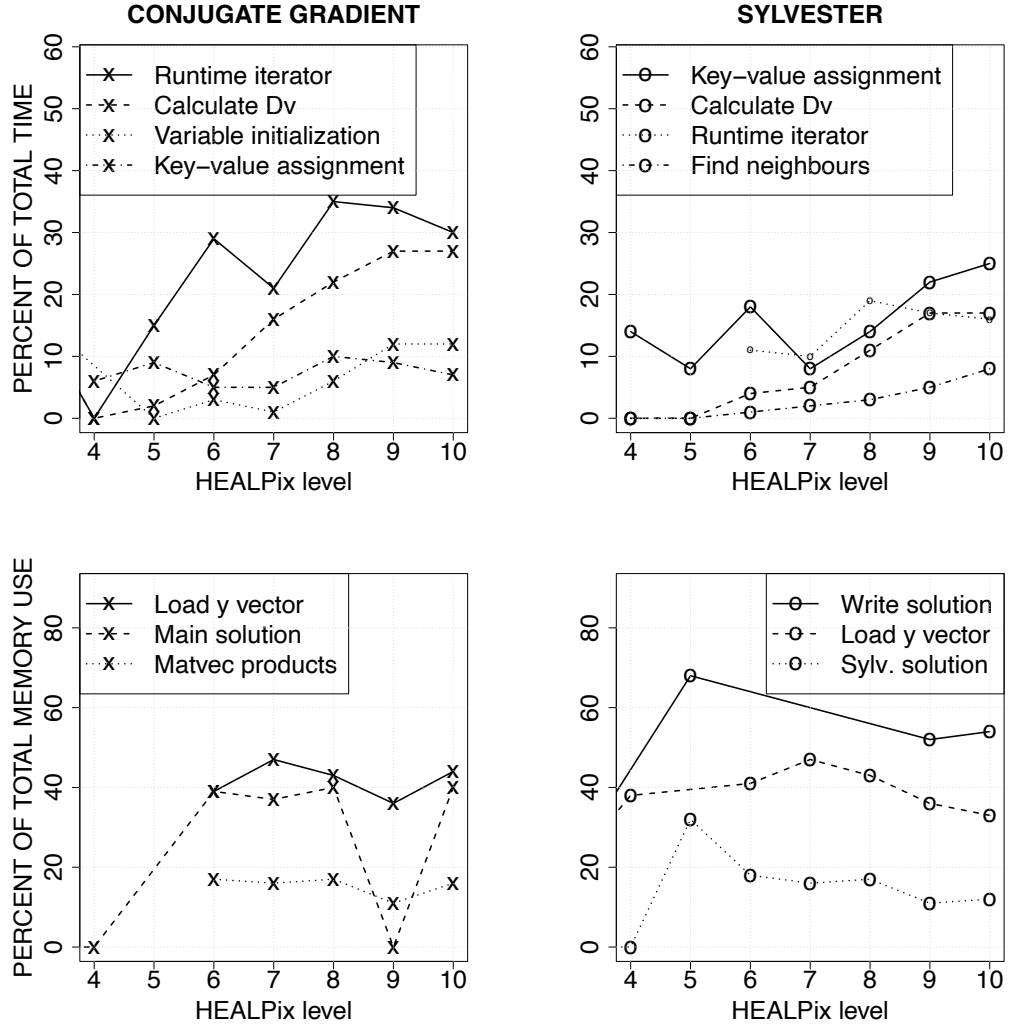


Figure 6.4: Code profiling. The functions that consume the most resources as a function of problem size. The top row is percentage of total computation time and the bottom row is percentage of maximum memory use. Table 6.5 provides more information on the functions. *Sylv.* refers to Algorithm 10.

7 Conclusions and Future Works

7.1 Conclusions

This thesis presents a comprehensive investigation into the development and application of approximate linear solvers for scalable statistical algorithms, with a primary focus on solving the Cosmic Microwave Background (CMB) source separation problem. From defining the problem in the context of large-scale, sparse linear systems to the implementation of novel solver methods, the research followed a clear and systematic approach.

The work began by identifying the challenges posed by large-scale linear systems, particularly in the context of high-dimensional data analysis like the CMB. The need for scalable, efficient algorithms that leverage problem-specific structures to approximate solutions became the central focus of the thesis. Motivated by the CMB source separation task, the research explored two main solvers: the Conjugate Gradient solver and the Lanczos-Sylvester. These methods were designed to exploit the underlying structures of the data, such as the Kronecker product structures and block Krylov subspaces, to deliver efficient and robust solutions.

The experimental results validated the effectiveness of these solvers. Specifically, the Lanczos-Sylvester solver demonstrated superior performance in terms of time complexity and memory efficiency, outperforming traditional methods and even the baseline sparse-dense Sylvester solver. The ability to solve large-scale problems with reduced computational resources, without compromising on accuracy, makes the Lanczos-Sylvester

solver particularly suitable for applications in cosmology and beyond.

Recalling the primary research aims discussed in the introduction, each has been achieved. In a theoretical perspective, this thesis explored and extended Krylov Subspaces techniques, particularly the Lanczos process, to handle large, sparse linear systems. We analyzed the theoretical foundation of a general Krylov Subspaces approach and offered new insights into the exploitation of special matrix structures appearing in the linear systems. Based on that theoretical foundation, we have successfully developed matrix-free algorithms for the CMB source separation problem. The proposed algorithms are highly generalizable and can be implemented in parallel settings, making them suitable for various applications. Finally, our experiments demonstrated that our approaches are not only effective for CMB data but also have the potential for broader application.

7.2 Future Works

7.2.1 CMB Source Separation with Polarization Data

The current work focuses on the separation of temperature signals in CMB data. A natural extension is to apply these methods to polarization data, which is essential for understanding the full CMB signal. Polarization introduces additional complexity, as the data is multidimensional, but the same block Krylov techniques could be adapted to handle the increased complexity of the system.

CMB polarization arises from the anisotropic scattering of CMB photons off free electrons during the era of recombination and reionization. These polarization patterns are essential for studying various cosmological phenomena, including the gravitational waves produced by inflation and the large-scale structure of the universe.

The polarization information is captured in two linear components:

- **Stokes-Q:** Describes the linear polarization along horizontal and vertical directions.

- **Stokes-U:** Describes the linear polarization along diagonal directions.

Both Stokes-Q and Stokes-U vary across the sky, just like temperature fluctuations, and are typically represented as maps. Together, the temperature (T), Stokes-Q, and Stokes-U form a vector that defines the total state of CMB radiation at each pixel on the sky.

Recall the source separation model in 3.1:

$$y_j = As_j + \epsilon_j; j = 1, \dots, N.$$

In the standard CMB source separation problem, the observed signal at pixel j is modeled as a mixture of different astrophysical sources for the temperature component alone. However, when including CMB polarization data, the model must account for the additional polarization components (Stokes-Q and Stokes-U), making the separation problem more complex. The new source vector s_j , representing the signal at pixel j , becomes a mixture of three components: temperature (T_j), Stokes-Q polarization (Q_j), and Stokes-U polarization (U_j).

$$s_j = \begin{bmatrix} T_j \\ Q_j \\ U_j \end{bmatrix}$$

In addition to the increased dimensions and computational complexity of the model, there are several challenges one should address for an efficient CMB polarization source separation model. Firstly, the mixing matrix $\mathbf{A}$, which previously handled only temperature data, must now account for the mixing of temperature and polarization signals, resulting in a more complex structure that reflects the relationships between different sources and these components. Additionally, polarization data is more susceptible to foreground contamination, such as galactic dust, which complicates the separation process. The algorithm must also address the correlation between temperature

and polarization signals, requiring modifications to jointly separate these components while preserving their inherent relationships. The noise model also becomes more intricate, as it must incorporate the noise present in both temperature and polarization measurements, adding further complexity to the source separation task.

7.2.2 Kronecker Product Exploitation in Matrix Compression

One significant attribute of the approaches proposed in this thesis is the exploitation of the Kronecker product structure that arises in the context of large, sparse linear systems. In our recent work [Pham and Wilson, 2023], we extended this concept to a different domain: the compression of auto-regressive language models. This method leverages the Kronecker product to decompose large weight matrices in neural networks, dramatically reducing the number of parameters while preserving model performance. Here, we introduce this approach and discuss potential extensions, including applying the Kronecker product technique to compress other neural network architectures.

Auto-regressive models have achieved state-of-the-art results in a wide variety of tasks, especially in Natural Language Processing. Recently, the Generative Pre-trained Transformer model (GPT) has attracted much attention because of its superior performance in multiple downstream tasks. However, due to their over-parameterized nature (e.g.: the GPT-3 model has 175 billion parameters), it is challenging to study and apply these models in resource-constraints environments. In [Pham and Wilson, 2023], we proposed an approach to reduce the number of parameters required for Transformer-based pre-trained language models like GPT using Kronecker product decomposition. We found that by representing the weight matrices of linear layers in the neural network by the Kronecker product of smaller matrices, the memory and computation overhead is reduced significantly. Moreover, we optimized the selection of the number of Kronecker factors as well as the size of factors in decomposition to balance between

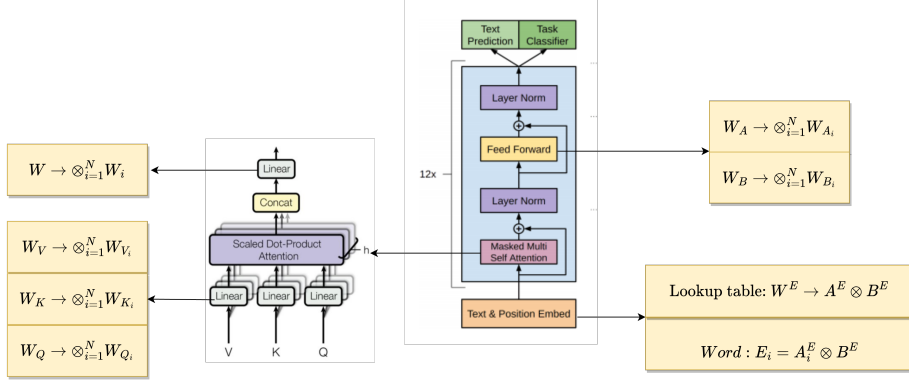


Figure 7.1: Kronecker decomposition on each component of the GPT architecture

saving resources and maintaining the performance of such models.

Methodology

We consider parameterizing each weight matrix $W \in \mathbb{R}^{m \times n}$ as a Kronecker product of N matrices $W_1, \dots, W_N$:

$$W = W_1 \otimes \dots \otimes W_N = \otimes_{i=1}^N W_i \quad (7.1)$$

where each *Kronecker factor* $W_i \in \mathbb{R}^{m_i \times n_i}$, $\prod_{i=1}^N m_i = m$ and $\prod_{i=1}^N n_i = n$. This decomposition reduces the number of parameters from mn to $\sum_{i=1}^N m_i n_i$. We can control the number of parameters by choosing both the number of factors and the size of each factor. That gives us the flexibility to find the balance between efficiency and performance in different real-world scenarios. Figure 1 shows the Kronecker decomposition for each component in the GPT architecture.

In the embedding layer, we represent the lookup table $W^E \in \mathbb{R}^{v \times d}$ by the Kronecker product of two smaller matrices $A^E \in \mathbb{R}^{v \times d/f}$ and $B^E \in \mathbb{R}^{1 \times f}$ where f is a factor of d . We can obtain the embedding E_i of a single word by $A_i^E \otimes B$ which is very efficient to compute. We also apply the Kronecker decomposition on the Multi-Head Attention (MHA) and the Feed Forward (FFN) layer of the Transformer architecture. The MHA module consists of three linear layers which calculate the Query, Key, and Value by multiplying the input by W_Q, W_K, W_V respectively. We use Equation 1 to decompose

	GPT-2 _{small}	KnGPT2	KroGPT
Parameters (million)	124	83	36
Training time (hours)	-	8.5	3.5
Dataset size (GB)	40	3.2	3.2
Test Perplexity	18.8	20.5	21.2

Table 7.1: Comparisons between KroGPT and two baseline models

these matrices as well as the subsequent linear mapping W the two matrices W_A, W_B in the following FFN block. Note that the weight matrices in a Transformer-based language model are not guaranteed to be block matrices, therefore Equation 1 is an approximation, which leads to information loss. To tackle that problem, we employ the knowledge distillation introduced by Tahaei et al. [2022] for the BERT.

Experimental Results

Our model KroGPT is a compressed version of the GPT-2_{small} model, which has 124 million parameters. The performance of this compressed model is significantly worse than the original one because of the information loss. We minimize that loss by pre-training the model on a relatively small dataset. We evaluate our model on the language modeling task using the Wikitext-103 dataset. We compare our model with the original GPT-2_{small} model and the similar Kronecker compression approach by Edalati et al. [2022] in terms of parameters saving, training cost, and performance. Results are presented in Table 7.1.

Future Works

This method leverages the Kronecker product to decompose large weight matrices in neural networks, dramatically reducing the number of parameters while preserving model performance. Future work could focus on the knowledge distillation techniques to recover the information loss due to compression or generalizing this technique to a broader range of deep learning models, particularly those with over-parameterized architectures, further advancing the field of neural network compression.

7.2.3 Generalized Algorithm for Gaussian Processes

In this thesis, we developed efficient solvers for large-scale linear systems by leveraging the specific structures present in the CMB source separation problem. These structures, such as the Kronecker product in the covariance matrix and the block structure in the system matrix, allowed for a significant reduction in computational complexity. However, the underlying framework—based on Gaussian processes—suggests that this approach can be generalized to other applications involving large-scale Gaussian Process (GP) models.

The general Gaussian process formulation involves inferring a latent Gaussian field $s \in \mathbb{R}^M$, which has a prior distribution:

$$s \sim \mathcal{N}(m, Q)$$

where $m \in \mathbb{R}^M$ is the mean vector (often zero), and Q is the covariance matrix. In many GP applications, including CMB, Q exhibits a Kronecker structure. The observation vector $y \in \mathbb{R}^N$ is also Gaussian-distributed, with a mean linearly dependent on s :

$$y \mid s \sim \mathcal{N}(Bs, C)$$

where B is an $N \times M$ matrix and C is the observation covariance matrix. The posterior distribution of s given y remains Gaussian, and the posterior mean μ is the solution to the system:

$$(B^T C^{-1} B + Q)\mu = B^T C^{-1} y$$

In the CMB source separation problem, specific structures in B , C , and Q allowed us to exploit computational efficiencies: B and Q have Kronecker structures, while C is diagonal. That leads us to a research question: how robust is this approach when

applied to more general Gaussian process models, where these matrices might take different forms?

Gaussian process models are widely used in various fields such as machine learning, spatial statistics, and time-series analysis. In many of these applications, covariance matrices exhibit special structures that can be exploited to reduce computational complexity, much like in the CMB problem. Some potential structures that could be leveraged include:

1. **Band Matrices:** In some GP applications, the observation covariance matrix C is not diagonal but has a banded structure, where non-zero elements are concentrated near the main diagonal. This occurs in spatial statistics, where measurements close to each other are correlated. Band matrices allow for efficient matrix-vector operations, which can be utilized in the solver to reduce memory usage and computational time.
2. **Toeplitz or Block-Toeplitz Matrices:** In time-series models, the covariance matrices are often Toeplitz or block-Toeplitz, reflecting the stationarity of the process. These matrices have a repeating structure that can be exploited to reduce storage requirements and improve the efficiency of matrix multiplications.
3. **Low-Rank Approximations:** For large datasets, especially in high-dimensional GP models, the covariance matrix Q may be approximated using a low-rank decomposition. This is particularly useful in kernel-based GPs, where the kernel matrix grows quadratically with the number of data points. Low-rank approximations, such as those obtained through the Nyström method, allow for scalable GP inference.
4. **Sparse Structures:** In cases where only a subset of the data points are highly correlated, the covariance matrix can be sparsified. This is common in models that use sparse Gaussian processes, where inducing points are introduced to approximate the full GP. Exploiting sparsity in the covariance matrix can drastically reduce computational costs.

Several studies have focused on leveraging matrix structures to improve the scalability of GP inference. Saatçi [2011] introduced structured kernel interpolation to exploit Toeplitz matrices in time-series Gaussian processes, allowing for scalable inference in high-dimensional settings. Similarly, Wilson et al. [2015] explored scalable kernel approximations using the Kronecker and Toeplitz structures in Gaussian processes for machine learning applications. In the context of sparse Gaussian processes, works by Quiñonero-Candela and Rasmussen [2005] have shown how inducing point methods can reduce the computational complexity of GPs by approximating the full covariance matrix with a smaller subset of data. This approach has been highly effective in scenarios with large datasets and has inspired further research into approximating GPs with sparse structures. By building on these advances, we aim to adapt and generalize our Kronecker-based solver to other large-scale GP models. The challenge lies in maintaining the balance between accuracy and efficiency as we move away from the specific structures of the CMB problem to more general Gaussian process models.

In conclusion, generalizing the solver developed in this thesis to other Gaussian process models represents a promising direction for future research. By exploring how the algorithm performs under different matrix structures—such as banded, sparse, or low-rank approximations—we can extend its applicability to a wide range of GP-based models in machine learning, spatial statistics, and beyond. This exploration will deepen our understanding of the algorithm’s robustness and its potential for solving complex, large-scale problems in various scientific and engineering domains.

Bibliography

Bin Yang, Hongwei Yang, and Shuang Li. Hierarchical approach for fast searching optimal launch opportunity in a wide range. *Advances in Space Research*, 63(1):572–588, 2019. ISSN 0273-1177. doi: <https://doi.org/10.1016/j.asr.2018.09.026>. URL <https://www.sciencedirect.com/science/article/pii/S0273117718307415>.

Peter Benner, Martin Köhler, and Jens Saak. Sparse-dense sylvester equations in h2-model order reduction. Preprint MPIMD/11-11, Max Planck Institute Magdeburg, December 2011. Available from <http://www.mpi-magdeburg.mpg.de/preprints/>.

Jon Cockayne, Chris Oates, Ilse Ipsen, and Mark Girolami. A Bayesian Conjugate Gradient Method. *arXiv e-prints*, art. arXiv:1801.05242, January 2018.

Martin Gutknecht. *A Brief Introduction to Krylov Space Methods for Solving Linear Systems*, pages 53–62. 01 2007. ISBN 978-3-540-46373-3. doi: 10.1007/978-3-540-46375-7_5.

Havard Rue and Leonhard Held. *Gaussian Markov Random Fields: Theory And Applications (Monographs on Statistics and Applied Probability)*. Chapman Hall/CRC, 2005a. ISBN 1584884320.

Aapo Hyvärinen and Erkki Oja. Independent component analysis by general nonlinear hebbian-like learning rules. *Signal Processing*, 64(3):301–313, 1998. ISSN 0165-1684. doi: [https://doi.org/10.1016/S0165-1684\(97\)00197-7](https://doi.org/10.1016/S0165-1684(97)00197-7). URL <https://www.sciencedirect.com/science/article/pii/S0165168497001977>.

- A. Hyvarinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3):626–634, 1999. doi: 10.1109/72.761722.
- Nilin Abrahamsen and Philippe Rigollet. Sparse gaussian ica. *ArXiv*, abs/1804.00408, 2018.
- Bobin, J., Sureau, F., and Starck, J.-L. Cosmic microwave background reconstruction from wmap and planck pr2 data. *AA*, 591:A50, 2016. doi: 10.1051/0004-6361/201527822. URL <https://doi.org/10.1051/0004-6361/201527822>.
- R H Dicke, R J.E. Peebles, P G Roll, and D T Wilkinson. Cosmic black-body radiation. *Astrophysical Journal (U.S.)*, (1). doi: 10.1086/148306. URL <https://www.osti.gov/biblio/4587058>.
- John C. Mather. The Cosmic Background Explorer (COBE). *Optical Engineering*, 21(4):769 – 774, 1982. doi: 10.1117/12.7972975. URL <https://doi.org/10.1117/12.7972975>.
- C. L. Bennett, M. Halpern, G. Hinshaw, N. Jarosik, A. Kogut, M. Limon, S. S. Meyer, L. Page, D. N. Spergel, G. S. Tucker, E. Wollack, E. L. Wright, C. Barnes, M. R. Greason, R. S. Hill, E. Komatsu, M. R. Nolta, N. Odegard, H. V. Peiris, L. Verde, and J. L. Weiland. First-year wilkinson microwave anisotropy probe/i (iWMAP/i) observations: Preliminary maps and basic results. *The Astrophysical Journal Supplement Series*, 148(1):1–27, sep 2003. doi: 10.1086/377253. URL <https://doi.org/10.1086%2F377253>.
- N. Aghanim, Y. Akrami, F. Arroja, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. J. Banday, R. B. Barreiro, N. Bartolo, S. Basak, R. Battye, K. Benabed, J.-P. Bernard, M. Bersanelli, P. Bielewicz, J. J. Bock, J. R. Bond, J. Borrill, F. R. Bouchet, F. Boulanger, M. Bucher, C. Burigana, R. C. Butler, E. Calabrese, J.-F. Cardoso, J. Carron, B. Casaponsa, A. Challinor, H. C. Chiang, L. P. L. Colombo, C. Combet, D. Contreras, B. P. Crill, F. Cuttaia, P. de Bernardis, G. de Zotti,

J. Delabrouille, J.-M. Delouis, F.-X. Désert, E. Di Valentino, C. Dickinson, J. M. Diego, S. Donzelli, O. Doré, M. Douspis, A. Ducout, X. Dupac, G. Efstathiou, F. Elsner, T. A. Enßlin, H. K. Eriksen, E. Falgarone, Y. Fantaye, J. Fergusson, R. Fernandez-Cobos, F. Finelli, F. Forastieri, M. Frailis, E. Franceschi, A. Frolov, S. Galeotta, S. Galli, K. Ganga, R. T. Génova-Santos, M. Gerbino, T. Ghosh, J. González-Nuevo, K. M. Górski, S. Gratton, A. Gruppuso, J. E. Gudmundsson, J. Hamann, W. Handley, F. K. Hansen, G. Helou, D. Herranz, S. R. Hildebrandt, E. Hivon, Z. Huang, A. H. Jaffe, W. C. Jones, A. Karakci, E. Keihänen, R. Keskitalo, K. Kiiveri, J. Kim, T. S. Kisner, L. Knox, N. Krachmalnicoff, M. Kunz, H. Kurki-Suonio, G. Lagache, J.-M. Lamarre, M. Langer, A. Lasenby, M. Lattanzi, C. R. Lawrence, M. Le Jeune, J. P. Leahy, J. Lesgourgues, F. Levrier, A. Lewis, M. Liguori, P. B. Lilje, M. Lilley, V. Lindholm, M. López-Caniego, P. M. Lubin, Y.-Z. Ma, J. F. Macías-Pérez, G. Maggio, D. Maino, N. Mandolesi, A. Mangilli, A. Marcos-Caballero, M. Maris, P. G. Martin, M. Martinelli, E. Martínez-González, S. Matarrese, N. Mauri, J. D. McEwen, P. D. Meerburg, P. R. Meinhold, A. Melchiorri, A. Mennella, M. Migliaccio, M. Millea, S. Mitra, M.-A. Miville-Deschênes, D. Molinari, A. Moneti, L. Montier, G. Morgante, A. Moss, S. Mottet, M. Münchmeyer, P. Natoli, H. U. Nørgaard-Nielsen, C. A. Oxborrow, L. Pagano, D. Paoletti, B. Partridge, G. Patanchon, T. J. Pearson, M. Peel, H. V. Peiris, F. Perrotta, V. Pettorino, F. Piacentini, L. Polastri, G. Polenta, J.-L. Puget, J. P. Rachen, M. Reinecke, M. Remazeilles, C. Renault, A. Renzi, G. Rocha, C. Rosset, G. Roudier, J. A. Rubiño-Martín, B. Ruiz-Granados, L. Salvati, M. Sandri, M. Savelainen, D. Scott, E. P. S. Shellard, M. Shiraishi, C. Sirignano, G. Sirri, L. D. Spencer, R. Sunyaev, A.-S. Suur-Uski, J. A. Tauber, D. Tavagnacco, M. Tenti, L. Terenzi, L. Toffolatti, M. Tomasi, T. Trombetti, J. Valiviita, B. Van Tent, L. Vibert, P. Vielva, F. Villa, N. Vittorio, B. D. Wandelt, I. K. Wehus, M. White, S. D. M. White, A. Zacchei, and A. Zonca. Planck2018 results. *Astronomy Astrophysics*, 641:A1, Sep 2020. ISSN 1432-0746. doi: 10.1051/0004-6361/201833880. URL <http://dx.doi.org/10.1051/0004-6361/201833880>.

- Jacques Delabrouille and J. Cardoso. Diffuse source separation in cmb observations. *Lecture Notes in Physics*, 665, 03 2007. doi: 10.1007/978-3-540-44767-2.6.
- R. Penrose. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, 51(3):406–413, 1955. doi: 10.1017/S0305004100030401.
- A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, 2000. ISSN 0893-6080. doi: [https://doi.org/10.1016/S0893-6080\(00\)00026-5](https://doi.org/10.1016/S0893-6080(00)00026-5). URL <https://www.sciencedirect.com/science/article/pii/S0893608000000265>.
- D. Maino, A. Farusi, C. Baccigalupi, F. Perrotta, A. J. Banday, L. Bedini, C. Burigana, G. De Zotti, K. M. Górski, and E. Salerno. All-sky astrophysical component separation with Fast Independent Component Analysis (FASTICA). , 334(1):53–68, July 2002. doi: 10.1046/j.1365-8711.2002.05425.x.
- Sebastian Wagner-Carena, Max Hopkins, Ana Diaz Rivero, and Cora Dvorkin. A novel cmb component separation method: hierarchical generalized morphological component analysis. *Monthly Notices of the Royal Astronomical Society*, 494(1):1507–1529, 2020.
- R. A. Fisher. On the Mathematical Foundations of Theoretical Statistics. *Phil. Trans. Roy. Soc. Lond. A*, 222:309–368, 1922. doi: 10.1098/rsta.1922.0009.
- J. Neyman. *Outline of a Theory of Statistical Estimation Based on the Classical Theory of Probability*. Philosophical transactions of the Royal Society of London / A. Royal Society, 1937. URL <https://books.google.com.vn/books?id=jqBtGwAACAAJ>.
- Harold Jeffreys. *Theory of Probability*. Clarendon Press, Oxford, England, 1939.
- Leonard Savage. *The Foundations of Statistics*. Wiley Publications in Statistics, 1954.
- C.P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer texts in statistics. Springer, 1999. ISBN 9780387987071. URL <https://books.google.com.vn/books?id=nlqVQgAACAAJ>.

- Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 0262018020.
- S.M. Kay. *Fundamentals of Statistical Signal Processing: Detection theory*. Fundamentals of Statistical Si. Prentice-Hall PTR, 1998. ISBN 9780133457117. URL <https://books.google.com.vn/books?id=vA9LAQAAIAAJ>.
- Roberto Trotta. Bayes in the sky: Bayesian inference and model selection in cosmology. *Contemporary Physics*, 49(2):71–104, March 2008. ISSN 1366-5812. doi: 10.1080/00107510802066753. URL <http://dx.doi.org/10.1080/00107510802066753>.
- L. J. M. Aslett. *MCMC for Inference on Phase-type and Masked System Lifetime Models*. PhD thesis, Trinity College Dublin, 2012.
- J.O. Berger and R.L. Wolpert. *The Likelihood Principle*. Institute of Mathematical Statistics. Lecture notes : monographs series. Institute of Mathematical Statistics, 1988. ISBN 9780940600133. URL <https://books.google.com.vn/books?id=7fz8JGLmWbgC>.
- A.O. Hagan and M. West. *The Oxford Handbook of Applied Bayesian Analysis*. Oxford Handbooks in Mathematics. OUP Oxford, 2010. ISBN 9780199548903. URL <https://books.google.com.vn/books?id=cwR8dm0nsokC>.
- J.M. Bernardo and A.F.M. Smith. *Bayesian Theory*. Wiley Series in Probability and Statistics. Wiley, 2009. ISBN 9780470317716. URL <https://books.google.com.vn/books?id=11nSgIcd7xQC>.
- H. Rue and L. Held. *Gaussian Markov Random Fields: Theory and Application*. Chapman and Hall/CRC, Boca Raton, Fl., 2005b.
- A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. ISBN 9781439840955. URL <https://books.google.com.vn/books?id=ZXL6AQAAQBAJ>.

- George Casella and Edward I. George. Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174, 1992. doi: 10.1080/00031305.1992.10475878. URL <https://www.tandfonline.com/doi/abs/10.1080/00031305.1992.10475878>.
- Gilles Moreau. On the solution phase of direct methods for sparse linear systems with multiple sparse right-hand sides. 2018.
- Yousef Saad. *Iterative Methods for Sparse Linear Systems*. 01 2003a. doi: 10.1137/1.9780898718003.ch4.
- Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of research of the National Bureau of Standards*, 49:409–435, 1952.
- Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, USA, 1996. ISBN 0801854148.
- Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357, 1983. doi: 10.1137/0720023. URL <https://doi.org/10.1137/0720023>.
- Yousef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986a. doi: 10.1137/0907058. URL <https://doi.org/10.1137/0907058>.
- R. Fletcher. Conjugate gradient methods for indefinite systems. In G. Alistair Watson, editor, *Numerical Analysis*, pages 73–89, Berlin, Heidelberg, 1976. Springer Berlin Heidelberg. ISBN 978-3-540-38129-7.
- H. A. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992. doi: 10.1137/0913035. URL <https://doi.org/10.1137/0913035>.

- Fokkema Diederik R. Sleijpen, Gerard L.G. Bcgstab() for linear equations involving unsymmetric matrices with complex spectrum. *ETNA. Electronic Transactions on Numerical Analysis [electronic only]*, 1:11–32, 1993. URL <http://eudml.org/doc/118652>.
- Shao-Liang Zhang. Gpbi-cg: Generalized product-type methods based on bi-cg for solving nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 18(2):537–551, 1997. doi: 10.1137/S1064827592236313. URL <https://doi.org/10.1137/S1064827592236313>.
- Cornelius Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Natl. Bur. Stand. B*, 45:255–282, 1950. doi: 10.6028/jres.045.026.
- K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann. Healpix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2):759, apr 2005. doi: 10.1086/427976. URL <https://dx.doi.org/10.1086/427976>.
- U. Fuskeland, I. K. Wehus, H. K. Eriksen, and S. K. Næss. Spatial variations in the spectral index of polarized synchrotron emission in the 9 yr wmap sky maps. *The Astrophysical Journal*, 790(2):104, jul 2014. doi: 10.1088/0004-637X/790/2/104. URL <https://dx.doi.org/10.1088/0004-637X/790/2/104>.
- Mathieu Remazeilles, Jacques Delabrouille, and Jean-François Cardoso. Cmb and sz effect separation with constrained internal linear combinations. *Monthly Notices of the Royal Astronomical Society*, 410(4):2481–2487, 2011. doi: <https://doi.org/10.1111/j.1365-2966.2010.17624.x>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2966.2010.17624.x>.
- John R. Gilbert and Robert Schreiber. Highly parallel sparse cholesky factorization. 13(5):1151–1172, 1992. doi: 10.1137/0913067. URL <https://doi.org/10.1137/0913067>.

Xiaoye S. Li and James W. Demmel. Superlu-dist: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. *ACM Trans. Math. Softw.*, 29(2):110–140, June 2003. ISSN 0098-3500. doi: 10.1145/779359.779361. URL <https://doi.org/10.1145/779359.779361>.

W. Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*. Applied mathematical sciences. Springer-Verlag, 1994. ISBN 9780387940649. URL <https://books.google.com.vn/books?id=xu0ZAQAAIAAJ>.

Davide Palitta and Valeria Simoncini. Optimality properties of galerkin and petrov–galerkin methods for linear matrix equations. *Vietnam Journal of Mathematics*, 48(4):791–807, Dec 2020a. ISSN 2305-2228. doi: 10.1007/s10013-020-00390-7. URL <https://doi.org/10.1007/s10013-020-00390-7>.

Peter Benner, Enrique S. Quintana-Ortí, and Gregorio Quintana-Ortí. State-space truncation methods for parallel model reduction of large-scale systems. *Parallel Computing*, 29(11):1701–1722, 2003. ISSN 0167-8191. doi: <https://doi.org/10.1016/j.parco.2003.05.013>. URL <https://www.sciencedirect.com/science/article/pii/S0167819103001534>. Parallel and distributed scientific and engineering computing.

Robert Granat and Bo Kågström. Parallel solvers for sylvester-type matrix equations with applications in condition estimation, part i: Theory and algorithms. *ACM Trans. Math. Softw.*, 37(3), September 2010. ISSN 0098-3500. doi: 10.1145/1824801.1824810. URL <https://doi.org/10.1145/1824801.1824810>.

Xiaoye S. Li. An overview of superlu: Algorithms, implementation, and user interface. *ACM Trans. Math. Softw.*, 31(3):302–325, September 2005. ISSN 0098-3500. doi: 10.1145/1089014.1089017. URL <https://doi.org/10.1145/1089014.1089017>.

Timothy A. Davis. Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):196–199, June 2004. ISSN

0098-3500. doi: 10.1145/992200.992206. URL <https://doi.org/10.1145/992200.992206>.

Yousef Saad and Martin H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986b. doi: 10.1137/0907058. URL <https://doi.org/10.1137/0907058>.

Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, Second edition, 2003b.

W.L. Briggs, V.E. Henson, and S.F. McCormick. *A Multigrid Tutorial: Second Edition*. Other Titles in Applied Mathematics. Society for Industrial and Applied Mathematics, 2000. ISBN 9780898714623. URL <https://books.google.com.vn/books?id=oSTGBm64o1AC>.

Michael L. Parks, Kirk Soodhalter, and Daniel B. Szyld. A block recycled GMRES method with investigations into aspects of solver performance. Technical Report 16-04-04, Department of Mathematics, Temple University, April 2016.

Erik G Boman, Andrew J Higgins, and Daniel B Szyld. Optimal size of the block in block gmres on gpus: computational model and experiments. *Numerical Algorithms*, 92(1):119–147, 2023. doi: 10.1007/s11075-022-01439-z.

Davide Palitta and Valeria Simoncini. Computationally enhanced projection methods for symmetric Sylvester and Lyapunov matrix equations. *Journal of Computational and Applied Mathematics*, 330:648–659, 2018. ISSN 0377-0427. doi: 10.1016/j.cam.2017.08.011.

William Pence. CFITSIO, v2.0: A New Full-Featured Data Interface. In David M. Mehringer, Raymond L. Plante, and Douglas A. Roberts, editors, *Astronomical Data Analysis Software and Systems VIII*, volume 172 of *Astronomical Society of the Pacific Conference Series*, page 487, January 1999.

Davide Palitta and Valeria Simoncini. Optimality properties of Galerkin and Petrov-Galerkin methods for linear matrix equations. *Vietnam J. Math.*, 48(4):791–807, 2020b. ISSN 2305-221X.

Allison H. Baker, John M. Dennis, and Elisabeth R. Jessup. On improving linear solver performance: a block variant of GMRES. *SIAM Journal on Scientific Computing*, 27(5):1608–1626, 2006. ISSN 1064-8275. doi: 10.1137/040608088.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Dung Pham and Simon Wilson. Compression of autoregressive language models using kronecker product decomposition. In *Proceedings of the 43rd Conference on Applied Statistics in Ireland*, pages 95–96, 2023. URL https://casi.ie/2023/wp-content/uploads/2023/06/CASI_2023_booklet.pdf.

Marzieh Tahaei, Ella Charlaix, Vahid Nia, Ali Ghodsi, and Mehdi Rezagholizadeh. KroneckerBERT: Significant compression of pre-trained language models through kronecker decomposition and knowledge distillation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2116–2127, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.154. URL <https://aclanthology.org/2022.naacl-main.154>.

Ali Edalati, Marzieh Tahaei, Ahmad Rashid, Vahid Nia, James Clark, and Mehdi

- Rezagholizadeh. Kronecker decomposition for GPT compression. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 219–226, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.24. URL <https://aclanthology.org/2022.acl-short.24>.
- Y. Saatçi. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011. PhD Thesis.
- Andrew G. Wilson, Elad Gilboa, Arye Nehorai, and John P. Cunningham. Fast kernel learning for multidimensional pattern extrapolation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, pages 3626–3634, 2015.
- Joaquín Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6: 1939–1959, 2005.

Appendix

Appendix A: Source Code

The source code is upload to a Github repository, including:

- Data processing source codes.
- Implementations of the Conjugate Gradient method, the baseline Sparse-Dense Sylvester method and the Lanczos-Sylvester method.
- Notebooks that describe code profiling and experimental results.

URL: <https://github.com/dungpt3003/cmb>

Appendix B: Code Profiling Details

0.1 1. Import necessary libraries

```
[302]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sys
import os
import re

import warnings
warnings.filterwarnings('ignore')
```

0.2 2. Data Processing

0.2.1 Reading raw file to dataframe

```
[303]: CURRENT_DIR = os.path.dirname(os.path.abspath('.'))
OUTPUT_PATH_PLANCK = "{}/output/profiling/planck/".format(CURRENT_DIR)
OUTPUT_PATH_RANDOM = "{}/output/profiling/random/".format(CURRENT_DIR)
MAX_HEALPIX_LEVEL = 10

[304]: df = pd.DataFrame(columns=['level', 'method', 'n_calls', 'total_time',
↳ 'per_call', 'cumulative_time', 'per_primitive_call', 'function'])
def process_output_file(filepath, level, method):
    with open(filepath, 'r') as f:
        for num, line in enumerate(f):
            #Skip the first five lines
            if num >= 5:
                content = re.sub("\s\s+", " ", line).split()
                contents = content[:5] + [" ".join(content[5:])]
                # Skip empty lines
                if len(contents) == 6:
                    df_length = len(df)
                    df.loc[df_length] = [level, method] + contents

[305]: for i in range(1, MAX_HEALPIX_LEVEL + 1):
    planck_file = OUTPUT_PATH_PLANCK + "profile_{}.txt".format(i)
    random_file = OUTPUT_PATH_RANDOM + "profile_random_{}.txt".format(i)
```

```
process_output_file(random_file, i, "random")
process_output_file(planck_file, i, "planck")
```

0.2.2 Re-format data

```
[307]: def split_n_calls(n_calls):
        lst = n_calls.split('/')
        if len(lst) == 1:
            return [lst[0], lst[0]]
        else:
            return lst

[308]: df["total_calls"] = df.apply(lambda row: split_n_calls(row['n_calls'])[0],
        ↪axis=1)
df["primitive_calls"] = df.apply(lambda row: split_n_calls(row['n_calls'])[1],
        ↪axis=1)

[309]: cols = [col for col in list(df.columns) if col not in ['function', 'n_calls',
        ↪'method']]
for col in cols:
    df[col] = pd.to_numeric(df[col], errors='coerce')
```

0.3 3. Profiling Analysis

0.3.1 3.1. Function call

```
[311]: df_group = df.groupby(['method', 'level'])

[312]: df_call = df_group['total_calls', 'primitive_calls'].sum().reset_index()

[313]: df_call
```

```
[313]:
```

	method	level	total_calls	primitive_calls
0	planck	1	80635	78419
1	planck	2	83201	80985
2	planck	3	93366	91150
3	planck	4	134936	132720
4	planck	5	301256	299040
5	planck	6	966380	964164
6	planck	7	3627287	3625071
7	planck	8	14270115	14267899
8	planck	9	56842282	56840067
9	planck	10	56842458	56840242
10	random	1	79434	77216
11	random	2	79434	77216
12	random	3	79434	77216
13	random	4	79434	77216
14	random	5	79434	77216

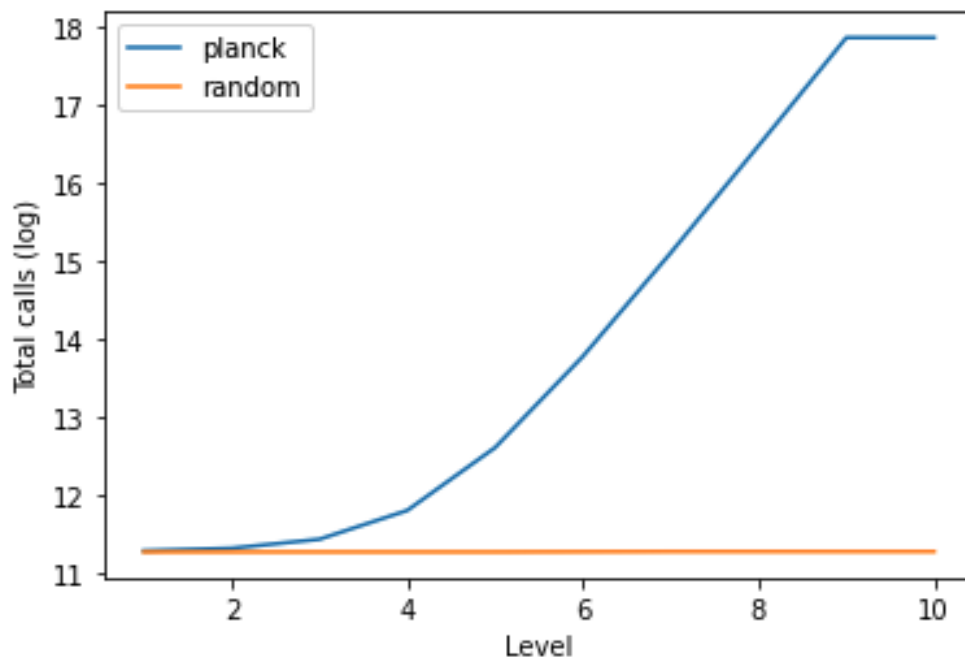
15	random	6	79661	77443
16	random	7	79888	77670
17	random	8	79888	77670
18	random	9	79888	77670
19	random	10	79888	77670

```
[314]: x = list(df_call[df_call['method'] == 'planck']['level'])
y_planck = [np.log(x) for x in list(df_call[df_call['method'] == 'planck']['total_calls'])]
y_random = [np.log(x) for x in list(df_call[df_call['method'] == 'random']['total_calls'])]

fig, ax = plt.subplots()

plt.plot(x, y_planck, label = "planck")
plt.plot(x, y_random, label = "random")
#plt.title("Total function calls")

ax.set_ylabel('Total calls (log)')
ax.set_xlabel('Level')
plt.legend()
plt.show()
```



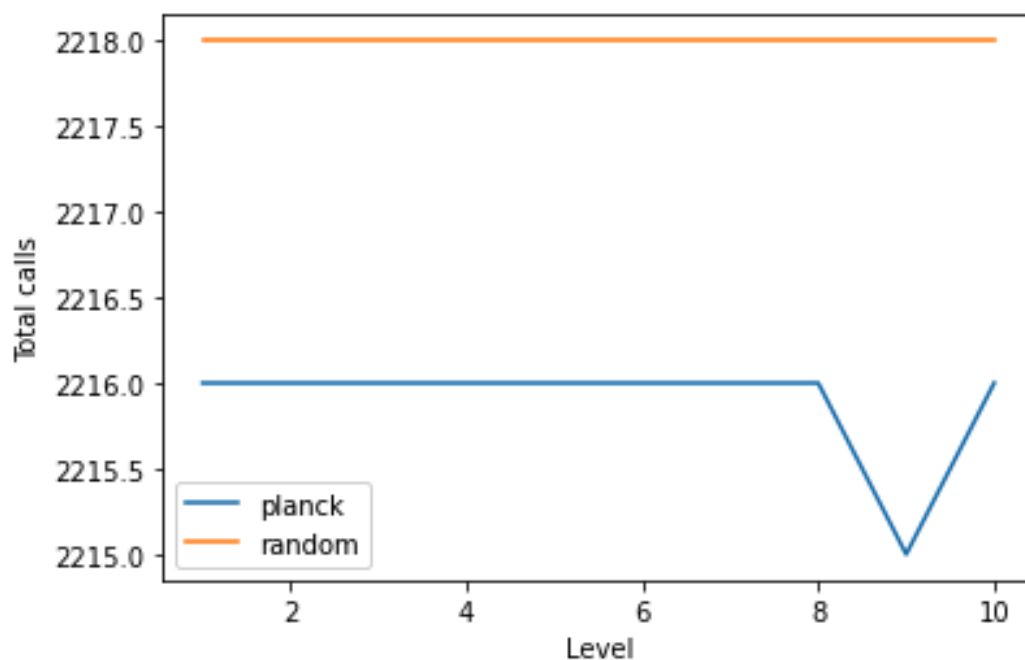
```
[315]: df_call['non_primitive_calls'] = df_call['total_calls'] -
↳ df_call['primitive_calls']
y_planck = list(df_call[df_call['method'] == 'planck']['non_primitive_calls'])
y_random = list(df_call[df_call['method'] == 'random']['non_primitive_calls'])

fig, ax = plt.subplots()

plt.plot(x, y_planck, label = "planck")
plt.plot(x, y_random, label = "random")

ax.set_ylabel('Total calls')
ax.set_xlabel('Level')

#plt.title("Total non-primitive function calls")
plt.legend()
plt.show()
```



0.3.2 3.2. Running time

Random data:

```
[ ]: df1 = df[(df['method'] == 'random') & (df['function'] == 'evaluation.py:
↳ 13(generate_random_sources)')]
df1
```

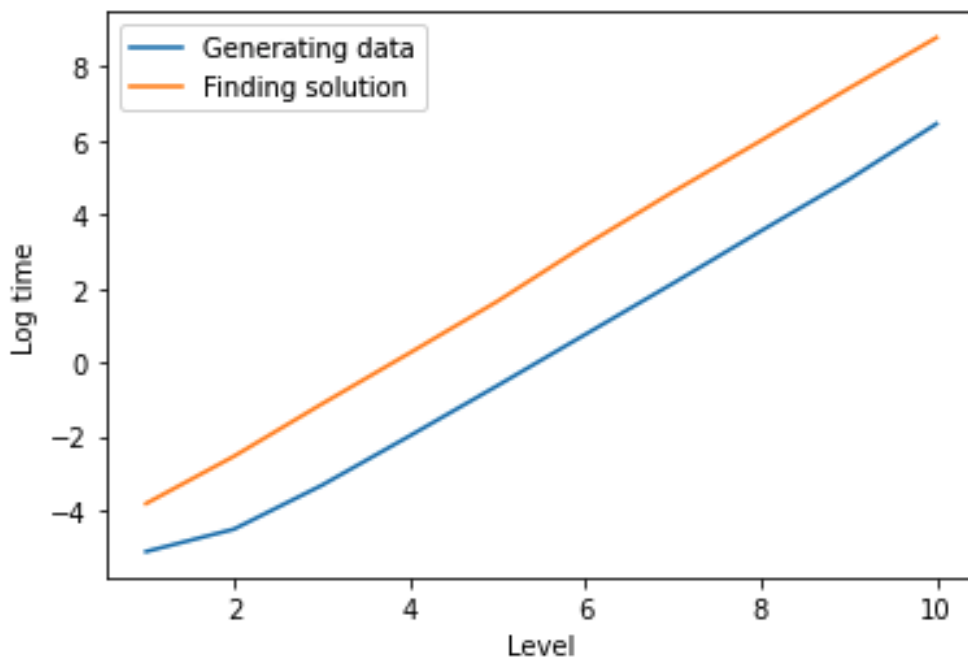
```
[ ]: df2 = df[(df['method'] == 'random') & (df['function'] == 'conjugate_solution.py:
↪81(findSolution)')]
df2
```

```
[318]: y_generate = [np.log(x) for x in list(df1['cumulative_time'])]
y_calculate = [np.log(x) for x in list(df2['cumulative_time'])]

fig, ax = plt.subplots()

plt.plot(x, y_generate, label = "Generating data")
plt.plot(x, y_calculate, label = "Finding solution")
#plt.title("Time spent on random data evaluation (log)")

ax.set_ylabel('Log time')
ax.set_xlabel('Level')
plt.legend()
plt.show()
```



```
[319]: labels = x
y_generate = list(df1['cumulative_time'])
y_calculate = list(df2['cumulative_time'])
y_total = [y_generate[i] + y_calculate[i] for i in range(len(y_generate))]
width = 0.35
```

```

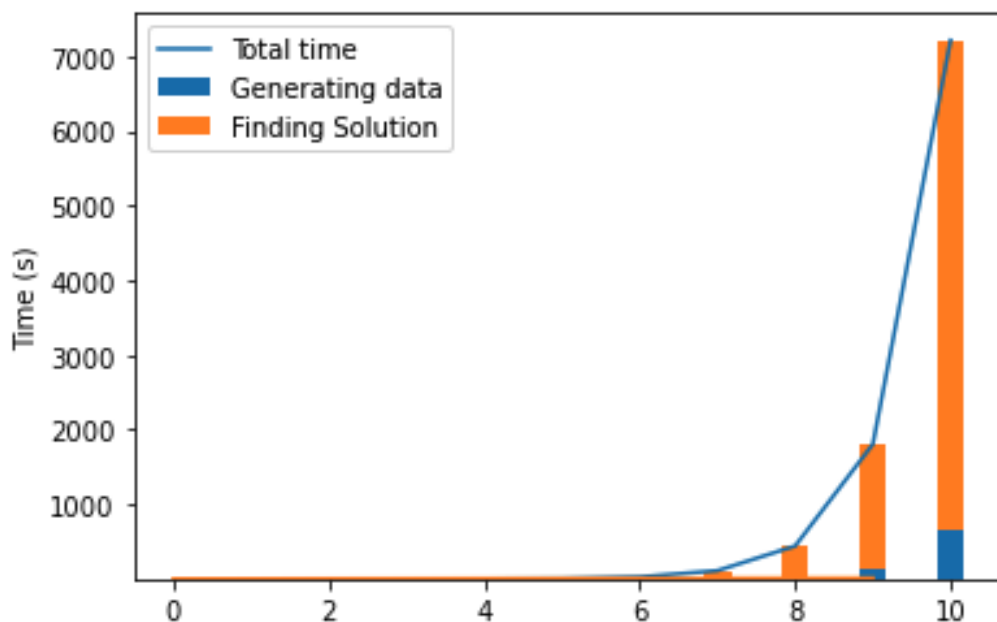
fig, ax = plt.subplots()

ax.bar(labels, y_generate, width, label='Generating data')
ax.bar(labels, y_calculate, width, bottom=y_generate, label='Finding Solution')
ax.plot(labels, y_total, label='Total time')
ax.plot(labels, )

ax.set_ylabel('Time (s)')
#ax.set_title('Time spent on random data evaluation')
ax.legend()

plt.show()

```



Planck data

```

[ ]: df3 = df[(df['method'] == 'planck') & (df['function'] == 'numpyio.py:
      ↳769(loadtxt)')]
df3

```

```

[ ]: df4 = df[(df['method'] == 'planck') & (df['function'] == 'conjugate_solution.py:
      ↳81(findSolution)')]
df4

```

```

[322]: y_generate = [np.log(x) for x in list(df3['cumulative_time'])]
y_calculate = [np.log(x) for x in list(df4['cumulative_time'])]

```

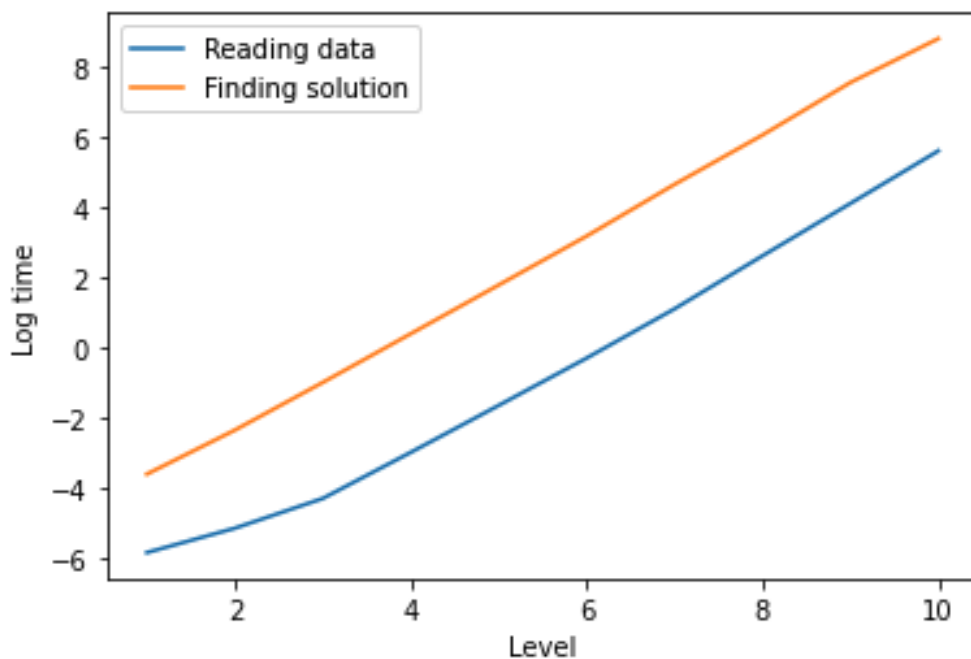
```

fig, ax = plt.subplots()

plt.plot(x, y_generate, label = "Reading data")
plt.plot(x, y_calculate, label = "Finding solution")
#plt.title("Time spent on Planck data evaluation (log)")

ax.set_ylabel('Log time')
ax.set_xlabel('Level')
plt.legend()
plt.show()

```



```

[324]: labels = x
y_generate = list(df3['cumulative_time'])
y_calculate = list(df4['cumulative_time'])
y_total = [y_generate[i] + y_calculate[i] for i in range(len(y_generate))]
width = 0.35

fig, ax = plt.subplots()

ax.bar(labels, y_generate, width, label='Reading data data')
ax.bar(labels, y_calculate, width, bottom=y_generate, label='Finding Solution')
ax.plot(labels, y_total, label='Total time')
ax.plot(labels, )

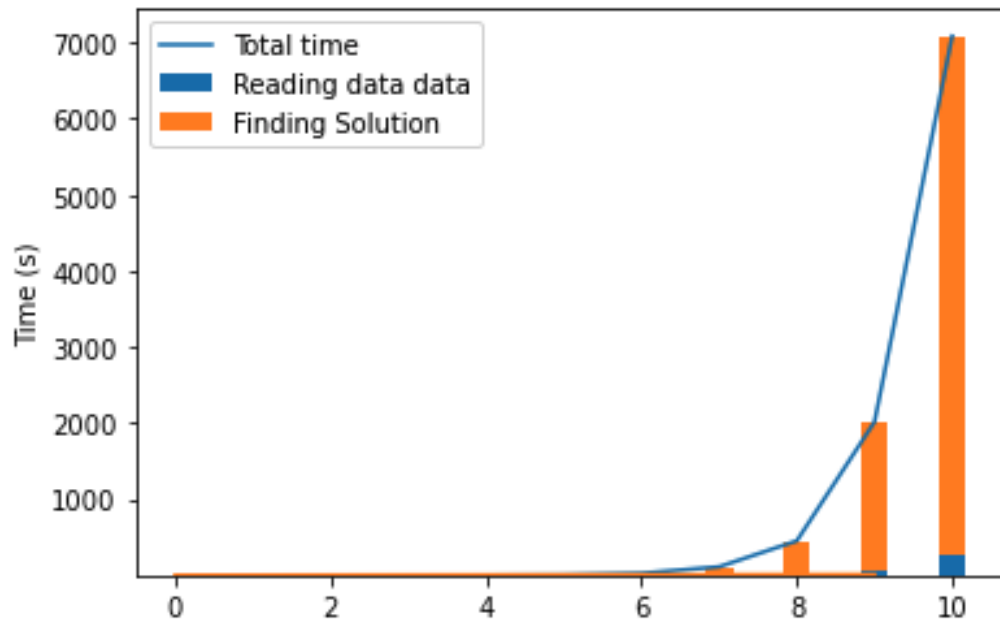
```

```

ax.set_ylabel('Time (s)')
#ax.set_title('Time spent on Planck data evaluation')
ax.legend()

plt.show()

```



Comparison

```

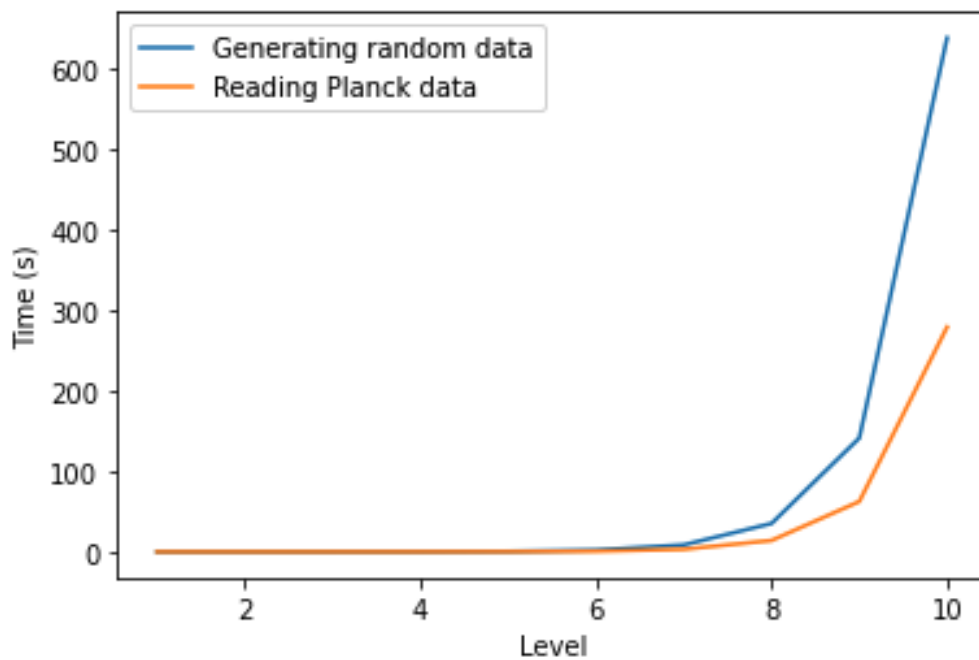
[325]: y_generate = list(df1['cumulative_time'])
y_read = list(df3['cumulative_time'])

fig, ax = plt.subplots()

plt.plot(x, y_generate, label = "Generating random data")
plt.plot(x, y_read, label = "Reading Planck data")
#plt.title("Comparison of time on data processing")

ax.set_ylabel('Time (s)')
ax.set_xlabel('Level')
plt.legend()
plt.show()

```



```
[326]: y_generate = [np.log(x) for x in list(df1['cumulative_time'])]
y_read = [np.log(x) for x in list(df3['cumulative_time'])]

fig, ax = plt.subplots()

plt.plot(x, y_generate, label = "Generating random data")
plt.plot(x, y_read, label = "Reading Planck data")
#plt.title("Comparison of time on data processing")

ax.set_ylabel('Time (log)')
ax.set_xlabel('Level')
plt.legend()
plt.show()
```

