NUMERICAL APPROXIMATIONS WITH TENSOR-BASED TECHNIQUES FOR HIGH-DIMENSIONAL PROBLEMS

AUTHOR: MARÍA MORA JIMÉNEZ

DIRECTED BY: J. ALBERTO CONEJERO CASARES
ANTONIO FALCÓ MONTESINOS

NOVEMBER 2023

A todas esas científicas que, por el hecho de ser mujeres, han sido ignoradas, apartadas o desmerecidas de sus descubrimientos. Gracias por abrirnos el camino.

Table of Contents

Abstract

The idea of following a sequence of steps to achieve a desired result is inherent in human nature: from the moment we start walking, following a cooking recipe or learning a new card game. Since ancient times, this scheme has been followed to organize laws, correct writings, and even assign diagnoses. In mathematics, this way of thinking is called an algorithm. Formally, an algorithm is a set of defined and unambiguous instructions, ordered and finite, that allows for solving a problem. From childhood, we face them when we learn to multiply or divide, and as we grow, these structures will enable us to solve different increasingly complex problems: linear systems, differential equations, optimization problems, etc.

There is a multitude of algorithms that allow us to deal with this type of problem, such as iterative methods, where we find the famous Newton Method to find roots; search algorithms to locate an element with specific properties in a more extensive set; or matrix decompositions, such as the LU decomposition to solve some linear systems. However, these classical approaches have limitations when faced with large-dimensional problems, a problem known as the 'curse of dimensionality'.

The advancement of technology, the use of social networks and, in general, the new problems that have appeared with the development of Artificial Intelligence, have revealed the need to handle large amounts of data, which requires the design of new mechanisms that allow its manipulation. This fact has aroused interest in the scientific community in tensor structures since they allow us to work efficiently with large-dimensional problems. However, most of the classic methods are not designed to be used together with these operations, so specific tools are required to allow their treatment, which motivates work like this.

This work is divided as follows: after reviewing some definitions necessary for its understanding, in Chapter 3, the theory of a new tensor decomposition for square matrices is developed. Next, Chapter 4 shows an application of said decomposition to regular graphs and small-world networks. In Chapter 5, an efficient implementation of the algorithm provided by the new matrix decomposition is proposed, and some order two PDEs are studied as an application. Finally, Chapters 6 and 7 present some brief conclusions and list some of the references consulted.

Resumen

La idea de seguir una secuencia de pasos para lograr un resultado deseado es inherente a la naturaleza humana: desde que empezamos a andar, siguiendo una receta de cocina o aprendiendo un nuevo juego de cartas. Desde la antigüedad se ha seguido este esquema para organizar leyes, corregir escritos, e incluso asignar diagnósticos. En matemáticas a esta forma de pensar se la denomina *algoritmo*. Formalmente, un algoritmo es un conjunto de instrucciones definidas y no-ambiguas, ordenadas y finitas, que permite solucionar un problema. Desde pequeños nos enfrentamos a ellos cuando aprendemos a multiplicar o dividir, y a medida que crecemos, estas estructuras nos permiten resolver diferentes problemas cada vez más complejos: sistemas lineales, ecuaciones diferenciales, problemas de optimización, etcétera.

Hay multitud de algoritmos que nos permiten hacer frente a este tipo de problemas, como métodos iterativos, donde encontramos el famoso Método de Newton para buscar raíces; algoritmos de búsqueda para localizar un elemento con ciertas propiedades en un conjunto mayor; o descomposiciones matriciales, como la descomposición LU para resolver sistemas lineales. Sin embargo, estos enfoques clásicos presentan limitaciones cuando se enfrentan a problemas de grandes dimensiones, problema que se conoce como 'la maldición de la dimensionalidad'.

El avance de la tecnología, el uso de redes sociales y, en general, los nuevos problemas que han aparecido con el desarrollo de la Inteligencia Artificial, ha puesto de manifiesto la necesidad de manejar grandes cantidades de datos, lo que requiere el diseño de nuevos mecanismos que permitan su manipulación. En la comunidad científica, este hecho ha despertado el interés por las estructuras tensoriales, ya que éstas permiten trabajar eficazmente con problemas de grandes dimensiones. Sin embargo, la mayoría de métodos clásicos no están pensados para ser empleados junto a estas operaciones, por lo que se requieren herramientas específicas que permitan su tratamiento, lo que motiva un proyecto como este.

El presente trabajo se divide de la siguiente manera: tras revisar algunas definiciones necesarias para su comprensión, en el Capítulo 3, se desarrolla la teoría de una nueva descomposición tensorial para matrices cuadradas. A continuación, en el Capítulo 4, se muestra una aplicación de dicha descomposición a grafos regulares y redes de mundo pequeño. En el Capítulo 5, se plantea una implementación eficiente del algoritmo que proporciona la nueva descomposición matricial, y se estudian como aplicación algunas EDP de orden dos. Por último, en los Capítulos 6 y 7 se exponen unas breves conclusiones y se enumeran algunas de las referencias consultadas, respectivamente.

Resum

La idea de seguir una seqüència de passos per a aconseguir un resultat desitjat és inherent a la naturalesa humana: des que comencem a caminar, seguint una recepta de cuina o aprenent un nou joc de cartes. Des de l'antiguitat s'ha seguit aquest esquema per a organitzar lleis, corregir escrits, i fins i tot assignar diagnòstics. En matemàtiques a aquesta manera de pensar se la denomina algorisme. Formalment, un algorisme és un conjunt d'instruccions definides i no-ambigües, ordenades i finites, que permet solucionar un problema. Des de xicotets ens enfrontem a ells quan aprenem a multiplicar o dividir, i a mesura que creixem, aquestes estructures ens permeten resoldre diferents problemes cada vegada més complexos: sistemes lineals, equacions diferencials, problemes d'optimització, etcètera.

Hi ha multitud d'algorismes que ens permeten fer front a aquesta mena de problemes, com a mètodes iteratius, on trobem el famós Mètode de Newton per a buscar arrels; algorismes de cerca per a localitzar un element amb unes certes propietats en un conjunt major; o descomposicions matricials, com la descomposició DL. per a resoldre sistemes lineals. No obstant això, aquests enfocaments clàssics presenten limitacions quan s'enfronten a problemes de grans dimensions, problema que es coneix com 'la maledicció de la dimensionalitat'.

L'avanç de la tecnologia, l'ús de xarxes socials i, en general, els nous problemes que han aparegut amb el desenvolupament de la Intel·ligència Artificial, ha posat de manifest la necessitat de manejar grans quantitats de dades, la qual cosa requereix el disseny de nous mecanismes que permeten la seua manipulació. En la comunitat científica, aquest fet ha despertat l'interés per les estructures tensorials, ja que aquestes permeten treballar eficaçment amb problemes de grans dimensions. No obstant això, la majoria de mètodes clàssics no estan pensats per a ser emprats al costat d'aquestes operacions, per la qual cosa es requereixen eines específiques que permeten el seu tractament, la qual cosa motiva un projecte com aquest.

El present treball es divideix de la següent manera: després de revisar algunes definicions necessàries per a la seua comprensió, en el Capítol 3, es desenvolupa la teoria d'una nova descomposició tensorial per a matrius quadrades. A continuació, en el Capítol 4, es mostra una aplicació d'aquesta descomposició a grafs regulars i xarxes de món xicotet. En el Capítol 5, es planteja una implementació eficient de l'algorisme que proporciona la nova descomposició matricial, i s'estudien com a aplicació algunes EDP d'ordre dos. Finalment, en els Capítols 6 i 7 s'exposen unes breus conclusions i s'enumeren algunes de les referències consultades, respectivament.

Chapter 1

Introduction and state of art

Rosalind Elsie Franklin (1920 – 1958)
Química y cristalógrafa británica, pionera en cristalografía de rayos x. Su imagen de una molécula de ADN resultó crítica para descifrar su estructura.

The use of algorithms goes back millennia, before the term itself existed. The Babylonians used algorithms to organise laws; ancient Latin teachers corrected grammar using algorithms; doctors have relied on algorithms to assign diagnoses; and countless people from all corners of the globe have tried to predict the future with algorithms. The idea of following a sequence of steps to achieve a desired outcome is inherent in human nature. However, the term "algorithm" is derived from the name of a Persian mathematician named Al-Khwarizmi, who lived in the 9th century. Al-Khwarizmi wrote a famous book called *Algoritmi de numero Indorum* [51].

In mathematics, logic, computer science and related disciplines, an algorithm is a finite, ordered, non-ambiguous, defined set of instructions or rules for solving a problem, performing a computation, processing data, and carrying out other tasks or activities. It can be considered as a recipe or a set of instructions that guide the process of solving a problem: given an initial state and an input, following successive steps leads to a final state and a solution is obtained [49].

From an early age, we are in contact with these typically mathematical structures, for example, when we start to multiply and divide and use the multiplication algorithm, the division algorithm or Euclid's algorithm. Also, when we use the Gaussian elimination method to solve a system of linear equations or to calculate the inverse of a square matrix. Even in other completely different disciplines, such as in language, when learning and using the grammatical rules of accentuation; in the chords of the refrain of a song; in card games, chess, and so forth. When solving more complex mathematical problems, such as differential equations or optimisation problems, we have several classical mechanisms at our disposal, such as iterative methods, among which we find the famous Newton's Method; correction algorithms; gradient descent algorithms; or matrix decompositions (such as LU, QR) among many others [18, 29, 52]. However, these classical approaches have limitations when faced with high-dimensional problems. As the problem dimension increases, the computational complexity of these methods increases significantly, leading to an increase in memory requirements and computational time. This problem is known as the *curse of dimensionality problem* [28].

Today, the advance of technology, Artificial Intelligence mechanisms and the rise of social networks have underlined the need to work with large amounts of data. So much so that the aforementioned methods are becoming obsolete in this new era, which is why the study and development of new mechanisms that can face these new challenges is so necessary. This has led to an interest in structures such as tensors, which, understood as a multidimensional generalisation of vectors and matrices, have proven to be a particularly useful tool in this context, allowing the representation and manipulation of high-dimensional data in a compact and structured way [40].

Tensors can be "measured" in different ways, for example, from the final size (dimension) of the object they define, or from the number of summands that compose it. For us, a tensor of order d will be an object defined on $\mathbb{R}^{N_1 \times N_2 \times \ldots N_d}$, and will have rank k if it can be written as the sum of k tensors of the same order.

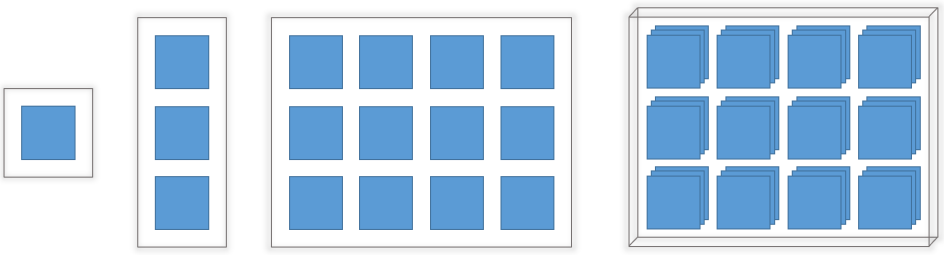

Figure 1: Representation of a tensor of order 0, in $\mathbb{R}$; 1, in $\mathbb{R}^3$; 2, in $\mathbb{R}^{3\times4}$; and 3, in $\mathbb{R}^{3\times4\times3}$, respectively.

For example, a $d-$dimensional tensor of rank k would be an object of the form

$$\mathbf{x} = \sum_{i=1}^{k} x_1^i \otimes x_2^i \otimes \cdots \otimes x_d^i.$$

In particular, a tensor of order 0 would be just a number; one of order 1 would be a vector; and of order 2 would be what we know as matrices [42].

Most existing mechanisms cannot be applied directly to tensor operations, requiring the design of proprietary tools for their use, which motivates work such as this. The use of tensor-based algorithms has proven to be highly beneficial in fields such as medicine, where accurate reconstruction of medical images from incomplete or noisy data is essential for diagnosing and treating diseases. Similarly, in machine learning, these algorithms have revolutionised information processing and feature extraction in massive datasets, improving prediction and classification capabilities.

In addition, tensor-based algorithms have found applications in physics, chemistry and engineering, where the simulation of complex systems and the solution of high-order differential equations require efficient and accurate numerical methods. Tensors provide a suitable structure for representing the properties and relationships of physical systems, making it possible to tackle previously computationally intractable problems [27].

Two of the most relevant strategies employed by these algorithms are tensor decomposition, which allows the representation of high-dimensional tensors in terms of lower-rank tensors, and low-rank iterative methods, which take advantage of the inherent structure of tensors to speed up computations.

These techniques are followed by the Tucker Decomposition, which allows a tensor to be expressed as a linear combination of kernels (a small central tensor) and mode factors (matrices), where each mode represents a dimension of the original tensor [54]; and by the Canonical Polyadic (CP) Decomposition, where a tensor is written as a sum of elementary tensors, each possessing relevant characteristics of the original tensor [8, 24].

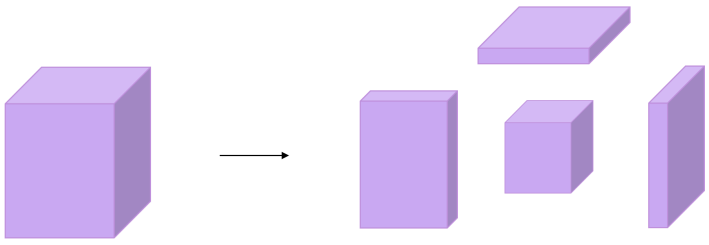

Figure 2: Graphic representation of the structure of Tucker Decomposition

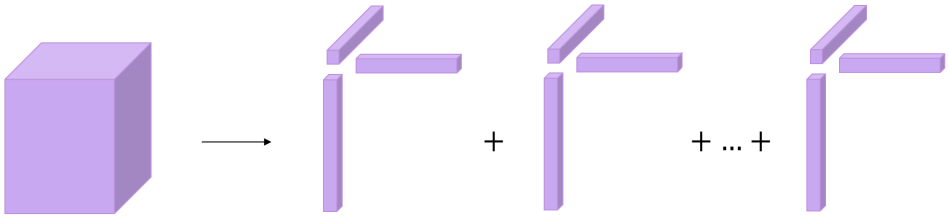

Figure 3: Graphic representation of the structure of CP Decomposition

Among the algorithms that use the above strategies, we will focus on two methods: the Proper Generalized Decomposition, PGD, and the Greedy Rank One Updated Algorithm, GROU. The **Proper Generalized Decomposition** technique approximates a complex multidimensional function by a linear combination of more straightforward, separable functions. This technique is based on the decomposition of the problem in different dimensions, which allows for reducing its complexity. These methods are mainly used in problems with multiple dependent variables, such as systems of partial differential equations or integral equations. The central idea of the PGD is to find a representation of the solution that is a separable function of each independent variable. To achieve this, optimization and approximation techniques are used to find the appropriate coefficients in the linear combination of separable functions [13, 43].

On the other hand, the **Greedy Rank One Updated Algorithm** is a method used to approximate the tensor decomposition of a high dimensional tensor in terms of rank one tensors. To do so, it uses the following iterative scheme: at each step, it selects a direction (or mode) that best approximates the structure of the original tensor. Then, it calculates the rank one tensor that best represents that direction and adds it to the current approximation. This process is repeated for each direction until a satisfactory approximation is obtained. Although the approximation obtained may not be exact, the algorithm seeks a good representation of the original tensor by focusing on the most significant directions and discarding the less relevant ones [1, 17].

In addition to studying the algorithms mentioned above, this work proposes a new tensor-based matrix decomposition algorithm that aims to improve the resolution of certain high-dimensional linear systems. The idea is to take advantage of the tensor structure of the matrix decomposition to improve the computational cost of tensor algorithms that solve these systems. As we will see, this method is particularly interesting when working with systems that come from the discretization of partial derivative equations.

The work is distributed as follows: in the Chapter 2, we will review the concepts and results we have used and need to know for the development of the thesis: elements of tensor calculus, notions of Lie algebras, etc.

Then, we will present the main result of the thesis in Chapter 3. This result is part of article *'Structure and approximation properties of Laplacian-like matrices'*, [7], and explains and demonstrates how to carry out a matrix decomposition in terms of the Laplacian matrix that best approximates it.

To study this new proposed decomposition, we have done some work with different graphs and small-world networks that we will show in Chapter 4. In it, we illustrate how we can use the Laplacian matrices to approximate the adjacency matrices in Watts-Strogatz networks. This study is part of the Preprint *On the tensor approximation of Watts-Strogatz networks.*

In view of the interesting results that we obtained as we progressed in the analysis of this structure, we decided to study in depth the algorithm that carried out this decomposition and, in Chapter 5, we explain how to perform an efficient implementation of it. This result is reported in the article *'A pre-processing method for the implementation of the Greedy-Rank One Algorithm for a class of linear systems'*, [6]. In it, we also study different problems in the form of a linear system, which come from the discretization of a PDE, as is the case of the Poisson, Helmholtz or Swift-Hohenberg equations. In these experiments, we can observe how our proposal improves computationally 'quite' much the $A\backslash b$ operation predefined in Matlab.

Finally, and by summary, we find the conclusions in Chapter 6 and the references consulted throughout the research.

Acknowledgements

María Mora Jiménez acknowledges funding from grant (ACIF/2020/269) funded by the Generalitat Valenciana and the European Social Found.

2 CHAPTER

Preliminaries and previous concepts

Ida Eva Tacke (1896 – 1978)
Química y Física alemana, fue la primera científica en mencionar la idea de la fisión nuclear. Encontró dos nuevos elementos -renio y masurium-.

To understand the content of the thesis, we will need to know or remember some concepts from different areas of mathematics, such as Tensor Calculus, Multilinear Algebra or Graphs.

The main results on which its content is based are discussed in each chapter, so we will only discuss some definitions and basic properties of concepts that will appear in one or more chapters, and we consider it essential to remember.

2.1 Notions of Tensor Calculus

Tensor Calculus refers to the operations and algorithms used to operate with tensors. A tensor is a multi-component algebraic 'entity' class that generalizes the concepts of scalar, vector and matrix in a way that is independent of any chosen coordinate system.

Once a vector basis is chosen, the components of a tensor in a basis will be given by a multimatrix. The order of a tensor will be the number of indices necessary to specify,

without ambiguity, a component of a tensor: a scalar will be considered as a tensor of order 0; a vector, a tensor of order 1; and, given a vector basis, second-order tensors can be represented by a matrix.

Kronecker's product

The operation par excellence with which we are going to work is the Kronecker product, which we will represent by the symbol $\otimes$. The Kronecker product of two matrices $A \in \mathbb{R}^{N\times M}$ and $B \in \mathbb{R}^{P\times Q}$ is defined as

$$A \otimes B = \begin{pmatrix} A_{1,1}B & A_{1,2}B & \dots & A_{1,M}B \\ A_{2,1}B & A_{2,2}B & \dots & A_{2,M}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{N,1}B & A_{N,2}B & \dots & A_{N,M}B \end{pmatrix} \in \mathbb{R}^{NP\times MQ},$$

or, element by element, from the expression

$$A \otimes B_{(n-1)P+p,(m-1)Q+q} = A_{n,m}B_{p,q}.$$

for $1 \le n \le N$, $1 \le m \le M$, $1 \le p \le P$ and $1 \le q \le Q$. Note that through this operation, we can obtain matrices (in general, tensors) of large sizes from much smaller ones.

On the other hand, some important properties of this operation that we are going to need, are:

1. Associative: $A \otimes (B \otimes C) = (A \otimes B) \otimes C$.
2. Distributive: $(A + B) \otimes C = (A \otimes C) + (B \otimes C)$.
3. $AB \otimes CD = (A \otimes C)(B \otimes D)$.
4. The inverse of the product is the product of the inverses, $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.
5. The transpose of the product is the product of the transposes, $(A \otimes B)^\top = A^\top \otimes B^\top$.
6. The trace of the product is the product of the traces, $\mathrm{tr}(A \otimes B) = \mathrm{tr}(A)\mathrm{tr}(B)$.

Another important concept that we are going to work with is the tensor norm. A norm $\|\cdot\|$ defined over $\mathbb{R}^{N\times N}$, where $N = n_1 \cdots n_d$, is called a tensor norm if and only if there exists a norm $\|\cdot\|_i$ over $\mathbb{R}^{n_i\times n_i}$ for $1 \le i \le d$, such that for any tensor $A = A_1 \otimes \cdots \otimes A_d \in \mathbb{R}^{N\times N}$, where $A_i \in \mathbb{R}^{n_i\times n_i}$ $(1 \le i \le d)$, it holds

$$\|A\| = \|A_1 \otimes \cdots \otimes A_d\| = \prod_{i=1}^{d} \|A_i\|_i.$$

In particular, the Frobenius norm $\|A\|_F = \sqrt{\langle A, A\rangle_{\mathbb{R}^{N\times N}}}$, is a tensor-norm, as we will see in Chapter 3.

2.1.1 Algorithms based on tensor decompositions

In this section, we will review the two main methods we have studied: the Greedy Rank One Algorithm - GROU - and the Family of PGD Methods. We will focus our attention on the GROU algorithm because it is, in turn, a starting point for PGD methods.

Greedy Rank One Updated Algorithm

Greedy algorithms are search algorithms that, at each step, choose the element that improves a certain objective function most in the hope of reaching a general optimal solution. This is the format that the Rank One Greedy algorithm adopts.

As commented in the introduction, the GROU Algorithm is a method used to approximate the tensor decomposition of a high dimensional tensor in terms of rank one tensors. If $\mathbf{u} \in \mathbb{R}^{N_1 \dots N_d \times N_1 \dots N_d}$, the GROU algorithm looks for an approximation of $\mathbf{u}$ of the form

$$\mathbf{u} = \sum_{j=1}^{k} \mathbf{x}_1^j \otimes \cdots \otimes \mathbf{x}_d^j,$$

where $\mathbf{x}_i \in \mathbb{R}^{N_i \times N_i}$, and k is the number of iterations of the algorithm. Following the notation in [1], the scheme followed by the algorithm is the following: let $\mathbf{y}_0 = \mathbf{x}_0 = 0$, for each $n \geq 1$ take

$$\begin{aligned} \mathbf{r}_{n-1} &= \mathbf{u} - \mathbf{x}_{n-1} \\ \mathbf{x}_n &= \mathbf{x}_{n-1} + \mathbf{y}_n \quad \text{where} \quad \mathbf{y}_n \in \underset{\text{rank}_{\otimes}\mathbf{y} \leq 1}{\arg\min} \|\mathbf{r}_{n-1} - \mathbf{y}\|. \end{aligned}$$

In this way,

$$\mathbf{u} \approx \mathbf{u}_n = \sum_{j=1}^{n} \mathbf{y}_j.$$

In the following chapters, we will see how to apply this technique to the resolution of linear systems.

PGD family

On the other hand, we find the family of Proper Generalized Decomposition -PGD- methods. Generalized Proper Decomposition is an a priori model reduction methodology based on the use of separate representations. Initially, it was developed to solve nonlinear structural problems in space-time, but it soon evolved towards its application in defined models in spaces with a high number of dimensions, and later, it was extended to general models in computational mechanics.

The main characteristic of the PGD is that it approximates the exact solution of the problem by imposing separation of variables. Thus, if $u(\mathbf{x}), \mathbf{x} \in \mathbb{R}^d$ is the solution to the

problem, it is approximated according to the PGD as:

$$u(\mathbf{x}) \approx \sum_{i=1}^{N} F_1^i(x_1) \cdot F_2^i(x_2) \cdot \cdots \cdot F_d^i(x_d),$$

where d is the number of dimensions of the problem. Unlike other methodologies, such as POD, the $F_n^s(x_n)$ functions are a priori unknown.

The strategy for determining the unknown functions F_j^i follows a greedy algorithm so that in each determination of each addend, a local optimum is selected with the perspective of finding the global optimum. One of the separate functions is obtained at each algorithm step, which are successively updated to minimize the residue.

2.2 Some Lie Algebra concepts

A Lie Algebra, G, is a vector space over a particular field F (typically the real or complex numbers), endowed with a binary operation called Lie Bracket, $[\cdot,\cdot] : G \times G \leftarrow G$, which satisfies the following properties:

1. is bilinear, that is, $[ax + by, z] = a[x, z] + b[y, z]$ and $[z, ax + by] = a[z, x] + b[z, y]$ for all $a, b \in F$ and all $x, y, z \in G$.

2. satisfies the Jacobi identity, that is, $[[x, y], z] + [[z, x], y] + [[y, z], x] = 0$ for all $x, y, z \in G$.

3. $[x, x] = 0$ for all $x \in G$.

From these properties, we can deduce the antisymmetry of the Lie Bracket, that is, $[x, y] = -[y, x]$ for all $x, y \in G$ and, in general, this operation is not associative.

On the other hand, a subalgebra of the Lie algebra G is a linear subspace H of G such that $[x, y] \in H$ for all $x, y \in H$ (i.e., $[h, h] \subset H$). The subalgebra is, then, a Lie algebra.

2.3 About Graphs and Small World Networks

Finally, we briefly recall some definitions of the Graphs area.

A **graph**, or equivalently, a **network**, is a structure formed by a tuple of the form $G = (V, E)$, where V is the set of *nodes*, $V = \{v_1, \ldots, v_n\}$, and E establishes the connections between the nodes, i.e., is the set of *edges*. We recall that the **adjacency matrix** $\mathscr{A}$ of G is defined as $\mathscr{A}_{ij} = 1$ if $(v_i, v_j) \in E$ and 0 elsewhere.

The degree of a vertex v_i, denoted by $\delta(v_i)$, is defined as the number of edges incident on it, counting the loops twice. We can define the **degree matrix** $\mathscr{D}$ of G as a diagonal matrix where $\mathscr{D}_{ii} = \delta(v_i)$ for all $1 \leq i \leq n$. Then, the **Laplacian matrix** (of the graph) $\mathscr{L}$ is defined as $\mathscr{L} = \mathscr{D} - \mathscr{A}$.

Small World networks

A network is k-regular if $\delta(v_i) = k$ for all $1 \leq i \leq n$. For example, circular graphs are 2-regular, and complete networks of $n+1$ nodes are n-regular.

A small-world network is a type of graph where most of the nodes are not neighbors of each other, but a relatively small edge path can connect two randomly chosen nodes. For example, those attending a wedding form a small world network: many will not know each other, but all of them will be connected through one of the bride and groom.

On the other hand, Watts and Strogatz introduce randomness into k-regular graphs to construct networks that preserve a high local clustering coefficient while reducing the average shortest length path between any pair of nodes to resemble the small world phenomenon. In Chapter 4, we remember how these networks are built from a regular graph.

3 CHAPTER

Structure and approximation properties of Laplacian-like matrices

Jocelyn Bell Burnell (1943 – Act)
Astrofísica norirlandesa, descubrió la primera radioseñal de un púlsar, gracias a lo cual se tuvo la primera evidencia de las ondas gravitacionales.

J. A. Conejero, A. Falcó, and M. Mora-Jiménez. *Structure and Approximation Properties of Laplacian-Like Matrices.* Results in Mathematics, 78 (184), 2023.

Abstract

Many of today's problems require techniques that involve the solution of arbitrarily large systems $A\mathbf{x} = \mathbf{b}$. A popular numerical approach is the so-called Greedy Rank-One Update Algorithm, based on a particular tensor decomposition.

The numerical experiments support the fact that this algorithm converges especially fast when the matrix of the linear system is Laplacian-Like. These matrices that follow the tensor structure of the Laplacian operator are formed by sums of Kronecker product of matrices following a particular pattern. Moreover, this set of matrices is not only a linear subspace it is a Lie sub-algebra of a matrix Lie Algebra.

In this Chapter, we characterize and give the main properties of this particular class of matrices. Moreover, the above results allow us to propose an algorithm to explicitly compute the orthogonal projection onto this subspace of a given square matrix $A \in \mathbb{R}^{N\times N}$.

3.1 Introduction

The study of linear systems is a problem that dates back to the time of the Babylonians, who used words like 'length' or 'width' to designate the unknowns without being related to measurement problems. The Greeks also solved some systems of equations, but using geometric methods [12]. Over the years, mechanisms to solve linear systems continued to be developed until the discovery of iterative methods, the practice of which began at the end of the 19th century, by the hand of the mathematician Gauss. The development of computers in the mid-20th century prompted numerous mathematicians to delve into the study of this problem [16, 17].

Nowadays, linear systems are widely used to approach computational models in applied sciences, for example, in mechanics, after the discretization of a partial differential equation. There are, in the literature, numerous mechanisms to deal with this type of problem, such as matrix decompositions (QR decomposition, LU decomposition), iterative methods (Newton, quasi-Newton, ...), and optimization algorithms (stochastic gradient descendent, alternative least squares,...), among others, see for instance [11, 19, 7]. However, most of them lose efficiency as the size of the matrices or vectors involved increases. This effect is known as the *curse of the dimensionality problem.*

To try to solve this drawback, we can use tensor-based algorithms [14], since their use significantly reduces the number of operations that we must employ. For example, we can obtain a matrix of size 100×100 (i.e. a total of 10.000 entries), from two matrices of size 10×10 multiplied, by means the tensor product, $100 + 100 = 200$ entries [8].

Among the algorithms based on tensor products strategies [18], the Proper Generalized Decomposition (PGD) family, based on the so-called Greedy Rank-One Updated (GROU) algorithm [1, 6], is one of the most popular techniques. PGD methods can be interpreted as 'a priori' model reduction techniques because they provide a way for the 'a priori' construction of optimally reduced bases for the representation of the solution. In particular, they impose a separation of variables to approximate the exact solution of a problem without knowing, in principle, the functions involved in this decomposition [4, 15]. The GROU procedure in the pseudocode is given in the Algorithm 1 (where $\otimes$ denotes the Kronecker product, that is briefly introduced in Section 3.2).

A good example is provided by the Poisson equation $-\Delta\phi = \mathbf{f}$. Let us consider the following problem in 3D,

$$\begin{cases} \dfrac{\partial^2\phi}{\partial x^2} + \dfrac{\partial^2\phi}{\partial y^2} + \dfrac{\partial^2\phi}{\partial z^2} = -\mathbf{f}(x,y,z), & \text{in} \quad \Omega = (0,1)^3, \\ \phi = 0 & \text{in} \quad \partial\Omega, \end{cases} \tag{1}$$

Algorithm 1 Greedy Rank-One Update

procedure GROU($\mathbf{b} \in \mathbb{R}^{n_1 \cdots n_d}$, $A \in \mathbb{R}^{n_1 \cdots n_d \times n_1 \cdots n_d}$, $\varepsilon > 0$, `tol`, `rank_max`)
  $\mathbf{r}_0 = \mathbf{b}$
  $\mathbf{x} = \mathbf{0}$
  for $i = 0, 1, 2, \ldots,$ `rank_max` **do**
    $\mathbf{y} = \arg\min_{\mathbf{y}=\mathbf{y}_1 \otimes \cdots \otimes \mathbf{y}_d} \|\mathbf{r}_i - A\mathbf{y}\|_2^2$
    $\mathbf{r}_{i+1} = \mathbf{r}_i - A\mathbf{y}$
    $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{y}$
    if $\|\mathbf{r}_{i+1}\|_2 < \varepsilon$ or $|\|\mathbf{r}_{i+1}\|_2 - \|\mathbf{r}_i\|_2| <$ `tol` **then goto** 13
    end if
  end for
  return u and $\|\mathbf{r}_{\texttt{rank\_max}}\|_2$.
  break
  return u and $\|\mathbf{r}_{i+1}\|_2$
end procedure

where $\mathbf{f}(x, y, z) = 3 \cdot (2\pi)^2 \cdot \sin(2\pi x - \pi)\sin(2\pi y - \pi)\sin(2\pi z - \pi)$. This problem has a closed form solution

$$\phi(x, y, z) = \sin(2\pi x - \pi)\sin(2\pi y - \pi)\sin(2\pi z - \pi).$$

By using the Finite Element Method (see [1] for more details), we can write the Poisson equation (1) in discrete form as a linear system $A \cdot \phi_{ijk} = -\mathbf{f}_{ijk}$, where the indices i, j, k correspond to the discretization of x, y and z respectively, and A is matrix having a particular representation, called Laplacian-like (see Definition 2 below), that allows to solve efficiently a high dimensional linear system. In Figure 4, we compare the CPU time employed in solving this discrete Poisson problem using the GROU Algorithm and the Matlab operator $\mathbf{x} = A \backslash \mathbf{b}$, for different numbers of nodes in $(0, 1)^3$. So, we will use this fact to study if, for a given generic square matrix, a characterization can be stated such that we can decide whether is either Laplacian-like or not. Clearly, under a positive answer, we expect that the analysis of the associated linear system $A\mathbf{x} = \mathbf{b}$ would be simpler. This kind of linear operator also exists in infinite dimensional vector spaces to describe evolution equations in tensor Banach spaces [2]. Its main property is that the associated dynamical system has an invariant manifold, the manifold of elementary tensors (see [3] for the details about its manifold structure).

Thus, the goal of this chapter is to obtain a complete description of this linear space of matrices, showing that is, in fact, a Lie subalgebra of $\mathbb{R}^{N \times N}$, and provide an algorithm in order to obtain the best approximation to this linear space, that is, to compute explicitly is the orthogonal projection on that space.

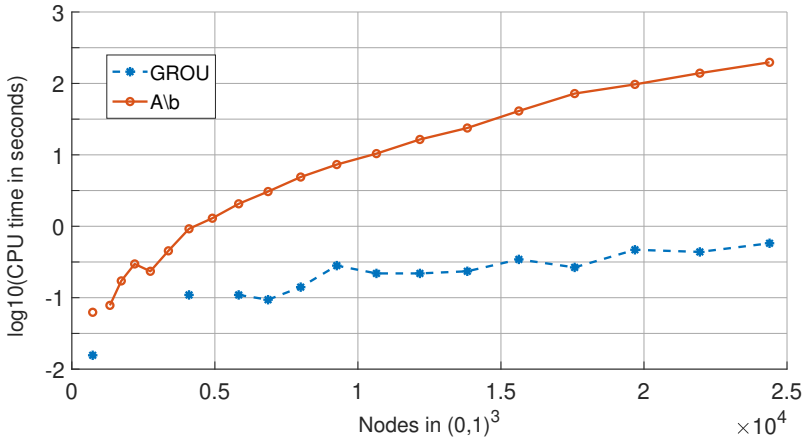

Figure 4: CPU time comparative to solve the discrete Poisson equation. For this numerical test, we have used a computer with the following characteristics: 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz, RAM 16,0 GB, 64 bit operating system; and a Matlab version R2021b [13].

The Chapter is organized as follows: in Section 3.2, we introduce the linear subspace of Laplacian-like matrices and prove that it is also a matrix Lie sub-algebra associated to a particular Lie group. Then, in Section 3.3, we prove that any matrix is uniquely decomposed as the sum of a Laplacian matrix and a matrix which is the subspace generated by the identity matrix, and we show that any Laplacian matrix is a direct sum of some particular orthogonal subspaces. Section 3.4 is devoted, with the help of the results of the previous section, to propose an algorithm to explicitly compute the orthogonal projection onto the subspace of Laplacian-like matrices. To illustrate this result, we also give two different numerical examples: the first one on the adjacency matrix of a simple graph; and the second on the numerical solution of the Poisson equation (1) by using a Finite Difference Scheme. Finally, in Section 3.5 some conclusions and final remarks are given.

3.2 The algebraic structure of Laplacian-Like matrices

First of all, we introduce some definitions, that will be used along this work.

Definition 1. *Let $A \in \mathbb{R}^{M\times N}$. Then, the Fröbenius norm (or the Hilbert–Schmidt norm) is defined as*

$$\|A\|_F = \sqrt{\sum_{i=1}^{M}\sum_{j=1}^{N} |a_{ij}|^2} = \sqrt{\operatorname{tr}\left(A^\top A\right)}.$$

The Fröbenius norm is the norm induced by the trace therefore, when $N = M$, we can work with the scalar product given by $\langle A, B\rangle = \operatorname{tr}\left(A^\top B\right)$. Let us observe that, in $\mathbb{R}^{N\times N}$,

1. $\langle A, B\rangle_{\mathbb{R}^{N\times N}} = \operatorname{tr}\left(A^\top B\right)$

2. $\langle A, \mathrm{id}_N \rangle_{\mathbb{R}^{N\times N}} = \mathrm{tr}(A) = \mathrm{tr}\left(A^\top\right)$

3. $\langle \mathrm{id}_N, \mathrm{id}_N \rangle_{\mathbb{R}^{N\times N}} = \|\mathrm{id}_N\|_F^2 = N$.

Given a linear subspace $\mathscr{U} \subset \mathbb{R}^{N\times N}$ we will denote:

(a) the orthogonal complement of $\mathscr{U}$ in $\mathbb{R}^{N\times N}$ by

$$\mathscr{U}^\perp = \left\{ V \in \mathbb{R}^{N\times N} : \langle U, V \rangle_{\mathbb{R}^{N\times N}} = 0 \text{ for all } U \in \mathscr{U} \right\},$$

and,

(b) the orthogonal projection of $\mathbb{R}^{N\times N}$ on $\mathscr{U}$ as

$$P_{\mathscr{U}}(V) := \arg\min_{U\in\mathscr{U}} \|U - V\|_F,$$

and hence

$$P_{\mathscr{U}^\perp} = \mathrm{id}_N - P_{\mathscr{U}}.$$

Before defining a Laplacian-like matrix, we recall that the *Kronecker product* of two matrices $A \in \mathbb{R}^{N_1\times M_1}$, $B \in \mathbb{R}^{N_2\times M_2}$ is defined by

$$A \otimes B = \begin{pmatrix} A_{1,1}B & A_{1,2}B & \dots & A_{1,M_1}B \\ A_{2,1}B & A_{2,2}B & \dots & A_{2,M_1}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{N_1,1}B & A_{N_1,2}B & \dots & A_{N_1,M_1}B \end{pmatrix} \in \mathbb{R}^{N_1N_2\times M_1M_2}.$$

Some of the well-known properties of the Kronecker product are:

1. $A \otimes (B \otimes C) = (A \otimes B) \otimes C$.

2. $(A + B) \otimes C = (A \otimes C) + (B \otimes C)$.

3. $AB \otimes CD = (A \otimes C)(B \otimes D)$.

4. $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.

5. $(A \otimes B)^\top = A^\top \otimes B^\top$.

6. $\mathrm{tr}(A \otimes B) = \mathrm{tr}(A)\mathrm{tr}(B)$.

From the example given in the introduction, we observe that there is a particular type of matrices to solve high-dimensional linear systems for which the GROU algorithm works particularly well: very fast convergence and also a very good approximation of the solution. These are the so-called Laplacian-Like matrices that we define below.

Definition 2. *Given a matrix $A \in \mathbb{R}^{N\times N}$, where $N = n_1 \cdots n_d$, we say that A is a Laplacian-like matrix if there exist matrices $A_i \in \mathbb{R}^{n_i \times n_i}$ for $1 \le i \le d$ be such that*

$$A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]} \doteq \sum_{i=1}^{d} \mathrm{id}_{n_1} \otimes \cdots \otimes \mathrm{id}_{n_{i-1}} \otimes A_i \otimes \mathrm{id}_{n_{i+1}} \otimes \cdots \otimes \mathrm{id}_{n_d}, \tag{2}$$

where id_{n_j} *is the identity matrix of size* $n_j \times n_j$.

Observe that for $1 < i < d$,

$$\mathrm{id}_{n_1} \otimes \cdots \otimes \mathrm{id}_{n_{i-1}} = \mathrm{id}_{n_1 \cdots n_{i-1}} \quad \text{and} \quad \mathrm{id}_{n_{i+1}} \otimes \cdots \otimes \mathrm{id}_{n_d} = \mathrm{id}_{n_{i+1} \cdots n_d},$$

hence

$$A_i \otimes \mathrm{id}_{[n_i]} = \mathrm{id}_{n_1 \cdots n_{i-1}} \otimes A_i \otimes \mathrm{id}_{n_{i+1} \cdots n_d}.$$

Moreover,

$$A_1 \otimes \mathrm{id}_{[n_1]} = A_1 \otimes \mathrm{id}_{n_2 \cdots n_d} \quad \text{and} \quad A_d \otimes \mathrm{id}_{[n_d]} = \mathrm{id}_{n_1 \cdots n_{d-1}} \otimes A_d.$$

It is not difficult to see that the set of Laplacian-like matrices is a linear subspace of $\mathbb{R}^{N\times N}$. From now on, we will denote by $\mathscr{L}\left(\mathbb{R}^{N\times N}\right)$ the subspace of Laplacian-like matrices in $\mathbb{R}^{N\times N}$ for a fixed decomposition of $N = n_1 \cdots n_d$.
These matrices can be easily related to the classical Laplacian operator [9, 10] by writing:

$$\frac{\partial^2}{\partial x_i^2} = \frac{\partial^0}{\partial x_1^0} \otimes \cdots \otimes \frac{\partial^0}{\partial x_{i-1}^0} \otimes \frac{\partial^2}{\partial x_i^2} \otimes \frac{\partial^0}{\partial x_{i+1}^0} \otimes \cdots \otimes \frac{\partial^0}{\partial x_d^0}$$

and where $\frac{\partial^0}{\partial x_j^0}$ is the identity operator for functions in the variable x_j for $j \neq i$.

As the next numerical example shows, matrices written as in (2) provides very good performance of the GROU algorithm. In Figure 5 we give a comparison of the speed of convergence to solve a linear system $A\mathbf{x} = \mathbf{b}$, where for each fixed size, we randomly generated two full-rank matrices: one given in the classical form and a Laplacian-like matrix. Both systems were solved following Algorithm 1.

The above results, together with the previous Poisson example given in the introduction, motivate the interest to know for a given matrix $A \in \mathbb{R}^{N\times N}$ how far it is from the linear subspace of Laplacian-like matrices. More precisely, we are interested in decomposing any matrix A as a sum of two orthogonal matrices L and $L^{\perp}$, where L is in $\mathscr{L}(\mathbb{R})$ and $L^{\perp}$ in $\mathscr{L}(\mathbb{R})^{\perp}$. Clearly, if we obtain that $L^{\perp} = 0$, that is, $A \in \mathscr{L}(\mathbb{R})$, then we can solve any associated linear system by means of the GROU algorithm.

Recall that the set of matrices $\mathbb{R}^{N\times N}$ is a Lie Algebra that appears as the tangent space at the identity matrix of the linear general group $\mathrm{GL}(\mathbb{R}^N)$, a Lie group composed by the non-singular matrices of $\mathbb{R}^{N\times N}$ (see [5]). Furthermore, the exponential map

$$\exp : \mathbb{R}^{N\times N} \longrightarrow \mathrm{GL}(\mathbb{R}^N), \quad A \mapsto \exp(A) = \sum_{n=0}^{\infty} \frac{A^n}{n!}$$

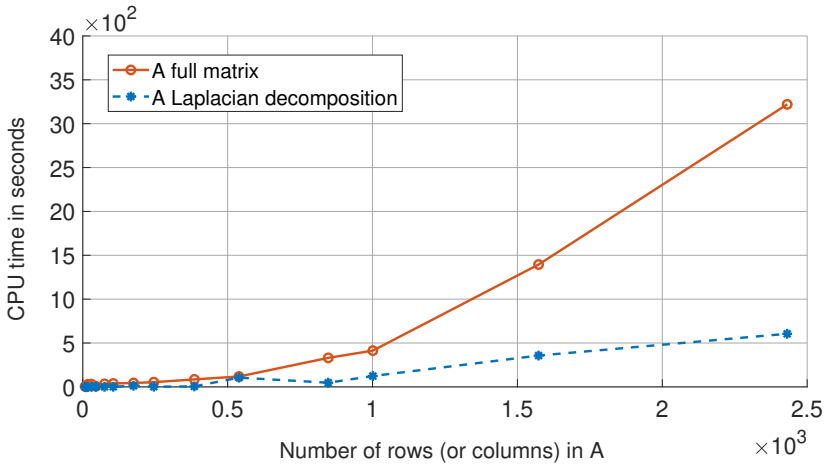

Figure 5: CPU time comparative to solve an $A\mathbf{x} = \mathbf{b}$ problem. This graph has been generated by using the following data in Algorithm 1: `tol` $= 2.22e-6$; $\varepsilon = 1.0e-06$; `rank_max` $= 3000$; (an `iter-max` $= 15$ was used to perform an ALS strategy); and the matrices have been randomly generated for each different size, in Laplacian and classical form. The characteristics of the computer used here are the same as in the case of Figure 1.

is well-defined, however it is not surjective because $\det(\exp(A)) = e^{\mathrm{tr}(A)} > 0$. Any linear subspace $\mathfrak{S} \subset \mathbb{R}^{N\times N}$ is a Lie-subalgebra if for all $A, B \in \mathfrak{S}$ its Lie crochet is also in $\mathfrak{S}$, that is, $[A,B] = AB - BA \in \mathfrak{S}$.
The linear space $\mathscr{L}(\mathbb{R}^{N\times N})$ is more than a linear subspace of $\mathbb{R}^{N\times N}$, it is also a Lie sub-algebra of $\mathbb{R}^{N\times N}$ as the next result shows.

Proposition 1. *Assume $\mathbb{R}^{N\times N}$, where $N = n_1 \cdots n_d$. Then the following statements hold.*

(a) The linear subspace $\mathscr{L}(\mathbb{R}^{N\times N})$ is a Lie subalgebra of the matrix Lie algebra $\mathbb{R}^{N\times N}$.

(b) The matrix group

$$\mathfrak{L}(\mathbb{R}^{N\times N}) = \left\{ \bigotimes_{i=1}^{d} A_i : A_i \in \mathrm{GL}(\mathbb{R}^{n_i}) \text{ for } 1 \le i \le d \right\}$$

is a Lie subgroup of $\mathrm{GL}\left(\mathbb{R}^N\right)$.

(c) The exponential map $\exp : \mathscr{L}(\mathbb{R}^{N\times N}) \longrightarrow \mathfrak{L}(\mathbb{R}^{N\times N})$ is well defined and it is given by

$$\exp\left(\sum_{i=1}^{d} A_i \otimes \mathrm{id}_{n_{[i]}}\right) = \bigotimes_{i=1}^{d} \exp(A_i).$$

Proof. (a) To prove the first statement, take $A, B \in \mathscr{L}(\mathbb{R}^{N\times N})$. Then there exist matrices $A_i, B_i \in \mathbb{R}^{n_i\times n_i}$ for $1 \le i \le d$ be such that

$$A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]} \quad \text{and} \quad B = \sum_{j=1}^{d} B_j \otimes \mathrm{id}_{[n_j]}.$$

Observe, that for $i < j$

$$(A_i \otimes \mathrm{id}_{[n_i]})(B_j \otimes \mathrm{id}_{[n_j]}) \quad \text{and} \quad (B_j \otimes \mathrm{id}_{[n_j]})(A_i \otimes \mathrm{id}_{[n_i]})$$

both products are equal to

$$\mathrm{id}_{n_1 \cdots n_{i-1}} \otimes A_i \otimes \mathrm{id}_{n_{i+1} \cdots n_{j-1}} \otimes B_j \otimes \mathrm{id}_{n_{j+1} \cdots n_d}.$$

A similar expression is obtained for $i > j$. Thus,

$$\left[A_i \otimes \mathrm{id}_{[n_i]}, B_j \otimes \mathrm{id}_{[n_j]} \right] = 0$$

for all $i \neq j$.
On the other hand, for $i = j$ we have

$$(A_i \otimes \mathrm{id}_{[n_i]})(B_i \otimes \mathrm{id}_{[n_i]}) = (A_i B_i \otimes \mathrm{id}_{[n_i]})$$

and

$$(B_i \otimes \mathrm{id}_{[n_i]})(A_i \otimes \mathrm{id}_{[n_i]}) = (B_i A_i \otimes \mathrm{id}_{[n_i]}).$$

Thus,

$$\left[A_i \otimes \mathrm{id}_{[n_i]}, B_i \otimes \mathrm{id}_{[n_i]} \right] = (A_i B_i - B_i A_i) \otimes \mathrm{id}_{[n_i]}.$$

that is,

$$\left[A_i \otimes \mathrm{id}_{[n_i]}, B_i \otimes \mathrm{id}_{[n_i]} \right] = [A_i, B_i] \otimes \mathrm{id}_{[n_i]}.$$

Here $[A_i, B_i]$ is the Lie crochet in $\mathbb{R}^{n_i \times n_i}$.

In consequence, from all said above, we conclude

$$[A, B] = \sum_{i=1}^{d} \sum_{j=1}^{d} \left[A_i \otimes \mathrm{id}_{[n_i]}, B_j \otimes \mathrm{id}_{[n_j]} \right] = \sum_{i=1}^{d} [A_i, B_i] \otimes \mathrm{id}_{[n_i]} \in \mathscr{L}\left(\mathbb{R}^{N \times N}\right).$$

This proves that $\mathscr{L}(\mathbb{R}^{N \times N})$ is a Lie sub-algebra of $\mathbb{R}^{N \times N}$.

(b) It is not difficult to see that $\mathfrak{L}(\mathbb{R}^{N \times N})$ is a subgroup of $\mathrm{GL}(\mathbb{R}^N)$. From Theorem 19.18 in [5], to prove that $\mathfrak{L}(\mathbb{R}^{N \times N})$ is a Lie subgroup of $\mathrm{GL}(\mathbb{R}^N)$ we only need to show that $\mathfrak{L}(\mathbb{R}^{N \times N})$ is a closed set in $\mathrm{GL}(\mathbb{R}^N)$. This follows from the fact that the map

$$\Phi : \mathrm{GL}(\mathbb{R}^{n_1}) \times \cdots \times \mathrm{GL}(\mathbb{R}^{n_d}) \longrightarrow \mathrm{GL}(\mathbb{R}^N) \quad (A_1, \cdots, A_d) \mapsto \bigotimes_{i=1}^{d} A_i$$

is continuous. Assume that there exists a sequence, $\{A_n\}_{n \in \mathbb{N}} \subset \mathfrak{L}(\mathbb{R}^{N \times N})$ convergent to $A \in \mathrm{GL}(\mathbb{R}^n)$. Then the sequence $\{A_n\}_{n \in \mathbb{N}}$ is bounded. Since there exists a sequence $\{(A_1^{(n)}, \ldots, A_d^{(n)})\}_{n \in \mathbb{N}} \subset \mathrm{GL}(\mathbb{R}^{n_1}) \times \cdots \times \mathrm{GL}(\mathbb{R}^{n_d})$ such that $A_n = \bigotimes_{j=1}^{d} A_j^{(n)}$, the sequence $\{(A_1^{(n)}, \ldots, A_d^{(n)})\}_{n \in \mathbb{N}}$ is also bounded. Thus, there exists a convergent sub-sequence, also denoted by $\{(A_1^{(n)}, \ldots, A_d^{(n)})\}_{n \in \mathbb{N}}$, to $(A_1, \ldots, A_d) \in \mathrm{GL}(\mathbb{R}^{n_1}) \times \cdots \times \mathrm{GL}(\mathbb{R}^{n_d})$. The continuity

of Φ, implies that $A = \bigotimes_{i=1}^{d} A_i$. Thus $\mathfrak{L}(\mathbb{R}^{N\times N})$ is closed in $\mathrm{GL}(\mathbb{R}^N)$, and hence a Lie subgroup.

(c) From Lemma 4.169(b)[8], the following equality

$$\exp\left(\sum_{i=1}^{d} \mathrm{id}_{n_1}\otimes\cdots\otimes\mathrm{id}_{n_{i-1}}\otimes A_i\otimes\mathrm{id}_{n_{i+1}}\otimes\cdots\otimes\mathrm{id}_{n_d}\right) = \bigotimes_{i=1}^{d}\exp(A_i)$$

holds. Thus, the exponential map is well defined. This ends the proof of the proposition. □

We conclude this section describing in a more detail the structure of matrices $A\in\mathbb{R}^{N\times N}$ for which there exists $A_i\in\mathbb{R}^{n_i\times n_i}$ for $1\le i\le d$ such that

$$A = \sum_{i=1}^{d} \mathrm{id}_{n_1\cdots n_{i-1}}\otimes A_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d}.$$

For dealing easily with Laplacian-like matrices, we introduce the following notation. For each $1 < i \le d$ consider the integer number $n_1n_2\cdots n_{i-1}$. Then, we will denote by

$$\begin{pmatrix} \star & \star & \cdots & \star \\ \star & \star & \cdots & \star \\ \vdots & \vdots & \ddots & \vdots \\ \star & \star & \cdots & \star \end{pmatrix}_{n_1n_2\cdots n_{i-1}\times n_1n_2\cdots n_{i-1}}$$

a block square matrix composed by $n_1n_2\cdots n_{i-1}\times n_1n_2\cdots n_{i-1}$-blocks.

Since $A_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} =$

$$\begin{pmatrix} (A_i)_{1,1}\mathrm{id}_{n_{i+1}\cdots n_d} & (A_i)_{1,2}\mathrm{id}_{n_{i+1}\cdots n_d} & \cdots & (A_i)_{1,n_i}\mathrm{id}_{n_{i+1}\cdots n_d} \\ (A_i)_{2,1}\mathrm{id}_{n_{i+1}\cdots n_d} & (A_i)_{2,2}\mathrm{id}_{n_{i+1}\cdots n_d} & \cdots & (A_i)_{2,n_i}\mathrm{id}_{n_{i+1}\cdots n_d} \\ \vdots & \vdots & \ddots & \vdots \\ (A_i)_{n_i,1}\mathrm{id}_{n_{i+1}\cdots n_d} & (A_i)_{n_i,2}\mathrm{id}_{n_{i+1}\cdots n_d} & \cdots & (A_i)_{n_i,n_i}\mathrm{id}_{n_{i+1}\cdots n_d} \end{pmatrix},$$

then $\mathrm{id}_{n_1\cdots n_{i-1}}\otimes A_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} =$

$$\begin{pmatrix} A_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} & O_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} & \cdots & O_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} \\ O_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} & A_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} & \cdots & O_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} \\ \vdots & \vdots & \ddots & \vdots \\ O_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} & O_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} & \cdots & A_i\otimes\mathrm{id}_{n_{i+1}\cdots n_d} \end{pmatrix}_{n_1n_2\cdots n_{i-1}\times n_1n_2\cdots n_{i-1}},$$

where O_i denotes the zero matrix in $\mathbb{R}^{n_i\times n_i}$ for $1\le i\le d$. To conclude, we have the following cases

$$A_1\otimes\mathrm{id}_{n_2\cdots n_d} = \begin{pmatrix} (A_1)_{1,1}\mathrm{id}_{n_2\cdots n_d} & (A_1)_{1,2}\mathrm{id}_{n_2\cdots n_d} & \cdots & (A_1)_{1,n_1}\mathrm{id}_{n_2\cdots n_d} \\ (A_1)_{2,1}\mathrm{id}_{n_2\cdots n_d} & (A_1)_{2,2}\mathrm{id}_{n_2\cdots n_d} & \cdots & (A_1)_{2,n_1}\mathrm{id}_{n_2\cdots n_d} \\ \vdots & \vdots & \ddots & \vdots \\ (A_1)_{n_1,1}\mathrm{id}_{n_2\cdots n_d} & (A_1)_{n_1,2}\mathrm{id}_{n_2\cdots n_d} & \cdots & (A_1)_{n_1,n_1}\mathrm{id}_{n_2\cdots n_d} \end{pmatrix}$$

and

$$\mathrm{id}_{n_1\cdots n_{d-1}} \otimes A_d = \begin{pmatrix} A_d & O_d & \cdots & O_d \\ O_d & A_d & \cdots & O_d \\ \vdots & \vdots & \ddots & \vdots \\ O_d & O_d & \cdots & A_d \end{pmatrix}_{n_1 n_2\cdots n_{d-1}\times n_1 n_2\cdots n_{d-1}}.$$

We wish to point out that the above operations are widely used in quantum computing.

3.3 A decomposition of the linear space of Laplacian-like matrices

We start by introducing some definitions and preliminary results needed to give an interesting decomposition of the linear space of Laplacian-like matrices. The next lemma lets us show how is the decomposition of $\mathbb{R}^{N\times N}$ as a direct sum of $\mathrm{span}\{\mathrm{id}_N\}$ and its orthogonal space $\mathrm{span}\{\mathrm{id}_N\}^\perp$, with respect the inner product $\langle A,B\rangle_{\mathbb{R}^{N\times N}} = \mathrm{tr}(A^TB)$. From now one, we will denote by

$$\mathfrak{h}_n := \mathrm{span}\{\mathrm{id}_n\}^\perp = \{A\in\mathbb{R}^{n\times n} : \mathrm{tr}(A) = 0\},$$

the linear subspace of null trace matrices in $\mathbb{R}^{n\times n}$.

Lemma 1. *Consider $(\mathbb{R}^{n\times n}, \|\cdot\|_F)$ as a Hilbert space. Then there exists a decomposition*

$$\mathbb{R}^{n\times n} = \mathrm{span}\{\mathrm{id}_n\}\oplus\mathfrak{h}_n,$$

Moreover, the orthogonal projection from $\mathbb{R}^{n\times n}$ on $\mathrm{span}\{\mathrm{id}_n\}$ is given by

$$P_{\mathrm{span}\{\mathrm{id}_n\}}(A) = \frac{\mathrm{tr}(A)}{n}\mathrm{id}_n,$$

and hence for each $A\in\mathbb{R}^{n\times n}$ we have the following decomposition,

$$A = \frac{\mathrm{tr}(A)}{n}\mathrm{id}_n + \left(A - \frac{\mathrm{tr}(A)}{n}\mathrm{id}_n\right),$$

where $\left(A-\frac{\mathrm{tr}(A)}{N}\mathrm{id}_N\right)\in\mathfrak{h}_n$.

Proof. The lemma follows from the fact that

$$P_{\mathrm{span}\{\mathrm{id}_n\}}(A) = \frac{\langle \mathrm{id}_n, A\rangle_{\mathbb{R}^{n\times n}}}{\|\mathrm{id}_N\|_F^2}\mathrm{id}_n = \frac{\mathrm{tr}(A)}{n}\mathrm{id}_n,$$

is the orthogonal projection onto $\mathrm{span}\{\mathrm{id}_n\}$. □

Now, we consider the matrix space $\mathbb{R}^{N\times N}$ where $N = n_1\cdots n_d$, and hence $\mathbb{R}^{N\times N} = \bigotimes_{i=1}^d \mathbb{R}^{n_i\times n_i}$ can be considered as a tensor space. A norm $\|\cdot\|$ defined over $\mathbb{R}^{N\times N}$ is called a tensor norm if and only if there exists a norm $\|\cdot\|_i$ over $\mathbb{R}^{n_i\times n_i}$ for $1\le i\le d$, such that for any tensor $A = A_1\otimes\cdots\otimes A_d\in\mathbb{R}^{N\times N}$, where $A_i\in\mathbb{R}^{n_i\times n_i}$ $(1\le i\le d)$, it holds

$$\|A\| = \|A_1\otimes\cdots\otimes A_d\| = \prod_{i=1}^d \|A_i\|_i.$$

We remark that for tensors $A = A_1 \otimes \cdots \otimes A_d$ and $B = B_1 \otimes \cdots \otimes B_d$ where $A_i, B_i \in \mathbb{R}^{n_i \times n_i}$, we have

$$\begin{aligned}\langle A, B\rangle_{\mathbb{R}^{N\times N}} &= \langle A_1 \otimes \cdots \otimes A_d, B_1 \otimes \cdots \otimes B_d\rangle_{\mathbb{R}^{N\times N}} \\ &= \operatorname{tr}((A_1 \otimes \cdots \otimes A_d)^T (B_1 \otimes \cdots \otimes B_d)) \\ &= \operatorname{tr}((A_1^T \otimes \cdots \otimes A_d^T)(B_1 \otimes \cdots \otimes B_d)) \\ &= \operatorname{tr}(A_1^T B_1 \otimes \cdots \otimes A_d^T B_d) \\ &= \prod_{i=1}^{d} \operatorname{tr}(A_i^\top B_i) = \prod_{i=1}^{d} \langle A_i, B_i\rangle_{\mathbb{R}^{n_i\times n_i}},\end{aligned}$$

and hence $\|A\|_F = \sqrt{\langle A, A\rangle_{\mathbb{R}^{N\times N}}}$, is a tensor-norm. In particular, the inner product $\langle\cdot,\cdot\rangle_{\mathbb{R}^{N\times N}}$ satisfies

$$\langle \mathrm{id}_{[n_i]} \otimes A_i, \mathrm{id}_{[n_i]} \otimes B_i\rangle_{\mathbb{R}^{N\times N}} = \operatorname{tr}(A_i^\top B_i) \prod_{\substack{j=1 \\ j\neq i}}^{d} n_j, \tag{3}$$

The next result gives a first characterization of the linear space $\mathscr{L}\left(\mathbb{R}^{N\times N}\right)$.

Theorem 1. *Let $\mathbb{R}^{N\times N}$, where $N = n_1 \cdots n_d$. Then*

$$\mathscr{L}\left(\mathbb{R}^{N\times N}\right) = \operatorname{span}\{\mathrm{id}_N\} \oplus \Delta, \tag{4}$$

where $\Delta = \mathfrak{h}_N \cap \mathscr{L}(\mathbb{R}^{N\times N})$. Furthermore, $\mathscr{L}(\mathbb{R}^{N\times N})^\perp$ is a subspace of $\mathfrak{h}_N$.

Proof. Assume that a given matrix $A \in \mathbb{R}^{N\times N}$ can be written as in (2) and denote each component in the sum representation of A, by $L_i = \mathrm{id}_{[n_i]} \otimes A_i$, where $A_i \in \mathbb{R}^{n_i\times n_i}$ for $1 \le i \le d$. Then $L_i \in \operatorname{span}\{\mathrm{id}_{[n_i]}\} \otimes \mathbb{R}^{n_i\times n_i}$ for $1 \le i \le d$, and in consequence,

$$\sum_{i=1}^{d} \operatorname{span}\{\mathrm{id}_{[n_i]}\} \otimes \mathbb{R}^{n_i\times n_i} = \mathscr{L}\left(\mathbb{R}^{N\times N}\right).$$

Thus, $\operatorname{span}\{\mathrm{id}_N\} \subset \mathscr{L}\left(\mathbb{R}^{N\times N}\right)$, and, by Lemma 1, we have the following decomposition

$$\mathscr{L}\left(\mathbb{R}^{N\times N}\right) = \Delta \oplus \operatorname{span}\{\mathrm{id}_N\}. \tag{5}$$

where $\Delta = \mathfrak{h}_N \cap \mathscr{L}(\mathbb{R}^{N\times N})$. The last statement is consequence of Lemma 1. This ends the theorem. □

Now, given any square matrix in $\mathbb{R}^{N\times N}$, we would like to project it onto $\mathscr{L}(\mathbb{R}^{N\times N})$ to obtain its Laplacian approximation. To compute this approximation explicitly, the following result, which is a consequence of the above theorem, will be useful.

Corollary 1. *Assume $\mathbb{R}^{N\times N}$, with $N = n_1 \cdots n_d \in \mathbb{N}$. Then*

$$P_{\mathscr{L}(\mathbb{R}^{N\times N})} = P_{\operatorname{span}\{\mathrm{id}_N\}} + P_\Delta,$$

that is, for all $A \in \mathbb{R}^{N\times N}$ it holds

$$P_{\mathscr{L}(\mathbb{R}^{N\times N})}(A) = \frac{\operatorname{tr}(A)}{N}\,\mathrm{id}_N + P_\Delta(A).$$

Next, we need to characterize Δ in order to explicitly construct the orthogonal projection onto $\mathscr{L}(\mathbb{R}^{N\times N})$. From the proof of the Theorem 1 we see that the linear subspaces given by

$$\mathfrak{h}_N \cap (\operatorname{span}\{\mathrm{id}_{[n_i]}\} \otimes \mathbb{R}^{n_i\times n_i}),$$

for $1 \le i \le d$, are of interest to characterize Δ as the next result shows.

Theorem 2. *Let $\mathbb{R}^{N\times N}$ with $N = n_1 \cdots n_d \in \mathbb{N}$, and let $\mathfrak{H}_i$ be the orthogonal complement of* $\operatorname{span}\{\mathrm{id}_N\}$ *in the linear subspace $\mathbb{R}^{n_i\times n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\}$ for $1 \le i \le d$. Then,*

$$\Delta = \bigoplus_{i=1}^{d} \mathfrak{H}_i. \tag{6}$$

Furthermore, a matrix A belongs to Δ if and only if it has the form

$$A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]}, \quad \textit{with} \quad \operatorname{tr}(A_i) = 0,\ i = 1,\dots,d.$$

Proof. First, we take into account that $\mathbb{R}^{n_i\times n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\}$ a linear subspace of $\mathscr{L}(\mathbb{R}^{N\times N})$ linearly isomorphic to the matrix space $\mathbb{R}^{n_i\times n_i}$. Thus, motivated by Lemma 1 applied on $\mathbb{R}^{n_i\times n_i}$, we write

$$\begin{aligned}\mathbb{R}^{n_i\times n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\} &= (\operatorname{span}\{\mathrm{id}_{n_i}\} \oplus \mathfrak{h}_{n_i}) \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\} \\ &= (\operatorname{span}\{\mathrm{id}_{n_i}\} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\}) \oplus (\mathfrak{h}_{n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\}).\end{aligned}$$

Now, we claim that $\mathfrak{h}_{n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\}$ it is the orthogonal complement of the linear subspace generated by the identity matrix $\mathrm{id}_N = \mathrm{id}_{n_i} \otimes \mathrm{id}_{[n_i]}$ in the linear subspace $\mathbb{R}^{n_i\times n_i} \otimes \mathrm{id}_{[n_i]}$. To prove the claim, observe that for $A_i \otimes \mathrm{id}_{[n_i]} \in \mathfrak{h}_{n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\}$ ($1 \le i \le d$), by using (3), it holds

$$\langle A_i \otimes \mathrm{id}_{[n_i]}, \mathrm{id}_N\rangle_{\mathbb{R}^{N\times N}} = \langle A_i \otimes \mathrm{id}_{[n_i]}, \mathrm{id}_{[n_i]} \otimes \mathrm{id}_{n_i}\rangle_{\mathbb{R}^{N\times N}} = \operatorname{tr}(A_i) \prod_{\substack{j=1\\ j\neq i}}^{d} n_j = 0,$$

because $A_i \in \mathfrak{h}_{n_i}$ and hence $\operatorname{tr}(A_i) = 0$, for $1 \le i \le d$. Thus, the claim follows and hence

$$\begin{aligned}\mathfrak{H}_i &= \mathfrak{h}_{n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\} \\ &= \{A_i \otimes \mathrm{id}_{[n_i]} \in \mathbb{R}^{n_i\times n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\} : \operatorname{tr}(A_i) = 0\} \\ &= \left\{A_i \otimes \mathrm{id}_{[n_i]} \in \mathbb{R}^{n_i\times n_i} \otimes \operatorname{span}\{\mathrm{id}_{[n_i]}\} : \langle A_i \otimes \mathrm{id}_{[n_i]}, \mathrm{id}_N\rangle_{\mathbb{R}^{N\times N}} = 0\right\}.\end{aligned}$$

To prove (6), we first consider $1 \le i < j \le d$, and take $A_k \otimes \mathrm{id}_{[n_k]} \in \mathfrak{H}_k$ for $k = i, j$. Then the inner product satisfies

$$\begin{aligned}\langle A_i \otimes \mathrm{id}_{[n_i]}, A_j \otimes \mathrm{id}_{[n_j]}\rangle_{\mathbb{R}^{N\times N}} &= \operatorname{tr}\left((A_i \otimes \mathrm{id}_{[n_i]})^T (A_j \otimes \mathrm{id}_{[n_j]})\right) \\ &= \operatorname{tr}\left((A_i^T \otimes \mathrm{id}_{[n_k]})(A_j \otimes \mathrm{id}_{[n_j]})\right) \\ &= \operatorname{tr}\left(\mathrm{id}_{n_1\cdots n_{i-1}} \otimes A_i^T \otimes \mathrm{id}_{n_{i+1}\cdots n_{j-1}} \otimes A_j \otimes \mathrm{id}_{n_{j+1}\cdots n_d}\right) \\ &= \prod_{\substack{\ell=1\\ \ell\neq i,j}}^{d} \langle \mathrm{id}_\ell, \mathrm{id}_\ell\rangle_{\mathbb{R}^{n_\ell\times n_\ell}} \operatorname{tr}(A_i)\operatorname{tr}(A_j) = 0,\end{aligned}$$

because $\mathrm{tr}(A_i) = \mathrm{tr}(A_j) = 0$. The same equality holds for $j < i$. Thus, we conclude that $\mathfrak{H}_i$ is orthogonal to $\mathfrak{H}_j$ for all $i \neq j$. So, the subspace

$$\Delta' = \bigoplus_{i=1}^{d} \mathfrak{H}_i,$$

is well defined and it is a subspace of $\mathscr{L}(\mathbb{R}^{N\times N})$.

To conclude the proof (6), we will show that $\Delta' = \Delta$. Since, for each $1 \leq i \leq d$, $\mathfrak{h}_i$ is orthogonal to $\mathrm{span}\{\mathrm{id}_N\}$ we have

$$\mathrm{span}\{\mathrm{id}_N\} \oplus \Delta' \subset \mathscr{L}(\mathbb{R}^{N\times N}).$$

To obtain the equality, take $A \in \mathscr{L}(\mathbb{R}^{N\times N})$. Then there exists $A_i \in \mathbb{R}^{n_i\times n_i}$ for $1 \leq i \leq d$ be such that

$$A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]}.$$

From Lemma 1 we can write

$$A_i = \frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i} + \left(A_i - \frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i}\right)$$

for each $1 \leq i \leq d$. Then,

$$\begin{aligned}
A &= \sum_{i=1}^{d} \left(\frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i} + \left(A_i - \frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i}\right)\right) \otimes \mathrm{id}_{[n_i]} \\
&= \sum_{i=1}^{d} \frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i} \otimes \mathrm{id}_{[n_i]} + \sum_{i=1}^{d} \left(A_i - \frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i}\right) \otimes \mathrm{id}_{[n_i]} \\
&= \sum_{i=1}^{d} \frac{\mathrm{tr}(A_i)}{n_i}\,(\mathrm{id}_{n_i} \otimes \mathrm{id}_{[n_i]}) + \sum_{i=1}^{d} \left(A_i - \frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i}\right) \otimes \mathrm{id}_{[n_i]} \\
&= \left(\sum_{i=1}^{d} \frac{\mathrm{tr}(A_i)}{n_i}\right) \mathrm{id}_N + \sum_{i=1}^{d} \left(A_i - \frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i}\right) \otimes \mathrm{id}_{[n_i]}.
\end{aligned}$$

Observe that $\left(\sum_{i=1}^{d} \frac{\mathrm{tr}(A_i)}{n_i}\right) \mathrm{id}_N \in \mathrm{span}\{\mathrm{id}_N\}$ and

$$\sum_{i=1}^{d} \left(A_i - \frac{\mathrm{tr}(A_i)}{n_i}\,\mathrm{id}_{n_i}\right) \otimes \mathrm{id}_{[n_i]} \in \Delta'.$$

Thus, $\mathscr{L}(\mathbb{R}^{N\times N}) \subset \mathrm{span}\{\mathrm{id}_N\} \oplus \Delta'$. In consequence $\Delta' = \Delta$, and this proves the theorem. □

A direct consequence of the above theorem is the next corollary.

Corollary 2. *Assume* $\mathbb{R}^{N\times N}$, *with* $N = n_1 \cdots n_d \in \mathbb{N}$. *Then*

$$P_{\mathscr{L}(\mathbb{R}^{N\times N})} = P_{\mathrm{span}\{\mathrm{id}_N\}} + \sum_{i=1}^{d} P_{\mathfrak{h}_i},$$

that is, for all $A \in \mathbb{R}^{N\times N}$ it holds

$$P_{\mathscr{L}(\mathbb{R}^{N\times N})}(A) = \frac{\operatorname{tr}(A)}{N}\,\mathrm{id}_N + \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]},$$

where $A_i \in \mathbb{R}^{n_i\times n_i}$ satisfies $\operatorname{tr}(A_i) = 0$ for $1 \le i \le d$.

3.4 A Numerical Strategy to perform a Laplacian-like decomposition

Now, in this section we will study some numerical strategies in order to compute, for a given matrix $A \in \mathbb{R}^{N\times N}$, with the help of Proposition 1 and Theorem 2, its best Laplacian-like approximation.Then, we will present two numerical examples to give consistency to the results obtained. In the first example, we will work with the adjacency matrix of a simple graph of 6 nodes; with it, we intend to show, step by step, the procedure to follow to calculate the Laplacian decomposition of a square matrix. The second example will complete the study of the discrete Poisson equation described in Section 3.1; with this example, we will illustrate the goodness of the Laplacian decomposition to solve PDEs in conjunction with the GROU Algorithm. We start with the following Greedy Algorithm.

Theorem 3. *Let A be a matrix in $\mathbb{R}^{N\times N}$, with $N = n_1\cdots n_d$, such that $\operatorname{tr}(A) = 0$. Consider the following iterative procedure:*

1. *Take $X_k^{(0)} = 0$ for $1 \le k \le d$.*
2. *For each $\ell \ge 1$ compute for $1 \le i \le d$ the matrix $U_i^{(\ell)}$ as*

$$U_i^{(\ell)} = \arg\min_{U_i\in\mathfrak{h}_i} \left\| A - \sum_{k=1}^{i-1} X_k^{(\ell)} \otimes \mathrm{id}_{[n_k]} - \xi(U_i)\otimes \mathrm{id}_{[n_i]} - \sum_{k=i+1}^{d} X_k^{(\ell-1)} \otimes \mathrm{id}_{[n_k]} \right\|,$$

where

$$\xi(U_i) = X_i^{(\ell-1)} + U_i,$$

and put $X_i^{(\ell)} = X_i^{(\ell-1)} + U_i^{(\ell)}$.

Then

$$\lim_{\ell\to\infty} \sum_{k=1}^{d} X_k^{(\ell)} \otimes \mathrm{id}_{[n_k]} = P_\Delta(A).$$

Proof. Recall that $P_\Delta(A)$ solves the problem

$$\min_{A^*\in\Delta} \left\| A - A^* \right\|.$$

To simplify notation put $P_\Delta^{(\ell)}(A) = \sum_{k=1}^{d} X_k^{(\ell)} \otimes \mathrm{id}_{[n_k]}$ for $\ell \ge 0$. By construction we have that

$$\|A - P_\Delta^{(1)}(A)\| \ge \|A - P_\Delta^{(2)}(A)\| \ge \cdots \ge \|A - P_\Delta^{(\ell)}(A)\| \ge \cdots \ge 0,$$

holds. Since the sequence $\{P_\Delta^{(\ell)}(A)\}_{\ell\in\mathbb{N}}$ is bounded, there is a convergent subsequence also denoted by $\{P_\Delta^{(\ell)}(A)\}_{\ell\in\mathbb{N}}$, so that

$$L_A = \lim_{\ell\to\infty} P_\Delta^{(\ell)}(A) \in \Delta.$$

If $L_A = P_\Delta(A)$, the theorem holds. Otherwise, assume that $L_A \neq P_\Delta(A)$, then it is clear that

$$\|A - P_\Delta(A)\| \leq \|A - L_A\|.$$

Suppose that $\|A - P_\Delta(A)\| < \|A - L_A\|$ and let $\lambda \in (0,1)$. Now, consider the linear combination $\lambda L_A + (1-\lambda)P_\Delta(A)$. Since $L_A, P_\Delta(A) \in \Delta$, they can be written as

$$L_A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]}, \quad \text{and} \quad P_\Delta(A) = \sum_{i=1}^{d} A_i^* \otimes \mathrm{id}_{[n_i]},$$

so $\lambda L_A + (1-\lambda)P_\Delta(A) = \sum_{i=1}^{d} \mathrm{id}_{[n_i]} \otimes \left(\lambda A_i + (1-\lambda)A_i^*\right) \in \Delta$. Hence,

$$\|A - P_\Delta(A)\| < \|A - (\lambda L_A + (1-\lambda)P_\Delta(A))\| < \|A - L_A\|.$$

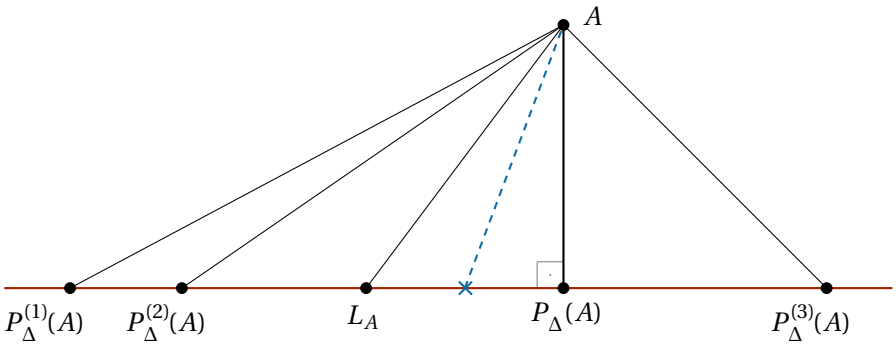

Figure 6: Situation described in reasoning by *Reductio ad absurdum*.

That is, we have found d matrices $Z_i = \lambda A_i + (1-\lambda)A_i^*$, $i = 1,\ldots,d$, such that

$$\|A - L_A\| = \left\| A - \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]} \right\| > \left\| A - \sum_{i=1}^{d} Z_i \otimes \mathrm{id}_{[n_i]} \right\|,$$

which is a contradiction with the definition of L_A. □

The previous result allows us to describe the procedure to obtain the Laplacian approximation of a square matrix, in the form of an algorithm. We can visualize the complete algorithm in the form of pseudocode in Algorithm 2.

Algorithm 2 Laplacian Decomposition Algorithm

procedure LAP(A^*, iter_max, tol)
  $A = A^* - (\mathrm{tr}(A)/N)\mathrm{id}_N$, iter = 1, Lap = 0
  while iter < iter_max **do**
    $A \leftarrow A - \mathrm{Lap}$
    for $k = 1, 2, \ldots, d$ **do**
      $P_k(A) = \mathrm{id}_{n_1} \otimes \cdots \otimes \mathrm{id}_{n_{k-1}} \otimes X_k \otimes \mathrm{id}_{n_{k+1}} \otimes \cdots \otimes \mathrm{id}_{n_d}$
      $X_k \leftarrow \min_{X_k} \|A - \sum_{i=1}^{k} P_i(A)\|$
      $\mathrm{Lap} = \mathrm{Lap} + P_k(A)$
    end for
    if $\|A - \mathrm{Lap}\| <$ tol **then goto** 14
    end if
    iter = iter + 1
  end while
  return Lap
end procedure

3.4.1 Numerical Examples

Example 1: The adjacency matrix of a simple graph

First, let us show an example in which the projection $P_\Delta(A)$ coincides with A and how the tensor representations is provided by the aforementioned proposed algorithm. Let us consider the simple graph $G(V,E)$, with $V = \{1,2,\ldots,6\}$ the set of nodes and $E = \{(1,2),(1,4),(2,3),(2,5),(3,6),(4,5),(5,6)\}$ the set of edges. Then, the adjacency matrix of G is

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}.$$

We want to find a Laplacian decomposition of the matrix $A \in \mathbb{R}^{6\times 6}$. Since $\mathrm{tr}(A) = 0$, we can do this by following the iterative scheme given by Theorem 3. So, we look for $X_1 \in \mathbb{R}^{2\times 2}$, $X_2 \in \mathbb{R}^{3\times 3}$ matrices such that

$$P_\Delta(A) = X_1 \otimes \mathrm{id}_{n_2} + \mathrm{id}_{n_1} \otimes X_2,$$

where $n_1 = 2, n_2 = 3$. We proceed according to the algorithm:

1. Computing X_1 by

$$\min_{X_1} \|A - X_1 \otimes \mathrm{id}_{n_2}\|,$$

we obtain

$$X_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

2. Computing X_2 by

$$\min_{X_2} \|A - X_1 \otimes \mathrm{id}_{n_2} - \mathrm{id}_{n_1} \otimes X_2\|,$$

we obtain

$$X_2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Since the residual value is $\|A - P_\Delta(A)\| = 0$, the matrix $A \in \Delta$. Thus, we can write it as

$$A = P_\Delta(A) = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \mathrm{id}_{n_2} + \mathrm{id}_{n_1} \otimes \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}.$$

Example 2: The Poisson's equation

Now, let us consider the Poisson's equation (1) with homogeneous boundary condition given in Section 3.1. For each $u \in \{x, y, z\}$ we fix $h = \frac{1}{n}$, where $n \in \mathbb{N}$, and take $u_\ell = (\ell - 1)h$ for $1 \leq \ell \leq n$. Next, we consider a derivative approximation scheme given by

$$\frac{\partial^2 \psi}{\partial u^2} \approx \frac{\psi(u_{\ell+1}) - 2\psi(u_\ell) + \psi(u_{\ell-1})}{h^2},$$

in (1) for $u \in \{x, y, z\}$. It allows to obtain a linear system written as

$$A\phi(x_i, y_j, z_k) = \mathbf{f}(x_i, y_j, z_k) \tag{7}$$

where the indices $1 \leq i, j, k \leq n$ correspond to the discretization of x, y and z respectively, and $A \in \mathrm{GL}\left(\mathbb{R}^{n^3}\right)$ is the matrix given by

$$A = \begin{pmatrix} T & -\mathrm{id}_{n^2} & & & \\ -\mathrm{id}_{n^2} & T & -\mathrm{id}_{n^2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\mathrm{id}_{n^2} & T & -\mathrm{id}_{n^2} \\ & & & -\mathrm{id}_{n^2} & T \end{pmatrix}, \tag{8}$$

where $T \in \mathbb{R}^{n^2 \times n^2}$ is the matrix

$$T = \begin{pmatrix} D & -\mathrm{id}_n & & & \\ -\mathrm{id}_n & D & -\mathrm{id}_n & & \\ & \ddots & \ddots & \ddots & \\ & & -\mathrm{id}_n & D & -\mathrm{id}_n \\ & & & -\mathrm{id}_n & D \end{pmatrix} \text{ with } D = \begin{pmatrix} 6 & -1 & 0 & \dots & 0 \\ -1 & 6 & -1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & -1 & 6 & -1 \\ 0 & \dots & 0 & -1 & 6 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

We can visualize a representation of the sparsity of the matrix A with the `spy` command from Matlab (see Figure 7). Since $\mathrm{tr}(A) = 6n^3 \neq 0$, instead of looking for the Laplacian approximation of A, we will look for $\hat{A} = \left(A - 6\,\mathrm{id}_{n^3}\right)$, which has null trace. Proceeding

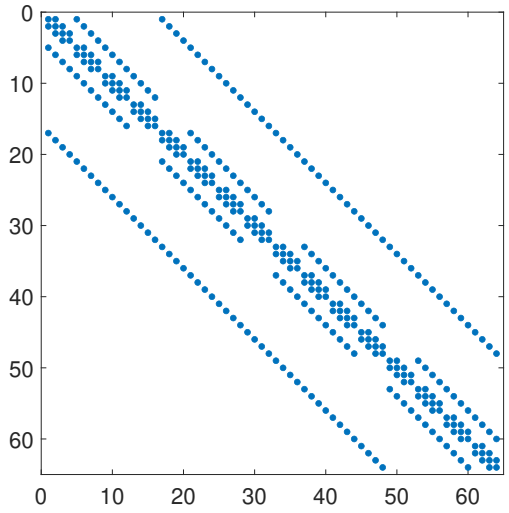

Figure 7: Matlab `spy(A)` representation of A given in (8) for $n = 4$.

according Algorithm 2 for sizes $n_1 = n_2 = n_3 = n$, we obtain the decomposition

$$\hat{A} = X \otimes \mathrm{id}_n \otimes \mathrm{id}_n + \mathrm{id}_n \otimes X \otimes \mathrm{id}_n + \mathrm{id}_n \otimes \mathrm{id}_n \otimes X,$$

where

$$X = \begin{pmatrix} 0 & -1 & 0 & \dots & 0 \\ -1 & 0 & -1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & -1 & 0 & -1 \\ 0 & \dots & 0 & -1 & 0 \end{pmatrix} \in \mathbb{R}^{n \times n},$$

and the residual of the approximation of $\hat{A}$ is $\|\hat{A} - P_\Delta(\hat{A})\| = 0$. Thus, following Corollary 2, we can write the original matrix A as

$$A = 6\,\mathrm{id}_{n^3} + X \otimes \mathrm{id}_n \otimes \mathrm{id}_n + \mathrm{id}_n \otimes X \otimes \mathrm{id}_n + \mathrm{id}_n \otimes \mathrm{id}_n \otimes X. \tag{9}$$

Note that the first term is $6 \cdot \mathrm{id}_{n^3} = 6 \cdot \mathrm{id}_n \otimes \mathrm{id}_n \otimes \mathrm{id}_n$, and hence A can be written as

$$A = Y \otimes \mathrm{id}_n \otimes \mathrm{id}_n + \mathrm{id}_n \otimes Y \otimes \mathrm{id}_n + \mathrm{id}_n \otimes \mathrm{id}_n \otimes Y,$$

where

$$Y = X + 2\,\mathrm{id}_n = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & -1 & 2 & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix}.$$

Next, we study the CPU-time used to solve the linear system (7) by using (a) the Matlab command $A\backslash b$, (b) the GROU Algorithm 1 with A in the compact form (8), and (c) the GROU Algorithm with A in Laplacian-Like form (9), previously obtained from the Laplacian Decomposition Algorithm 2.

We have used in the numerical experiments the following parameters values. For the GROU Algorithm 1: $\texttt{tol} = 2.2204e-16$; $\varepsilon = 2.2204e-16$; $\texttt{rank\_max} = 15$; (an $\texttt{iter-max} = 5$ was used to perform an ALS strategy); and the number of nodes in $(0,1)^3$ (that is, the number of rows or columns of the matrix A) increase from 10^3 to 35^3. For the Laplacian Decomposition Algorithm we fixed $\texttt{iter\_max} = 4$ and a tolerance $\texttt{tol} = 10^{-5}$. The characteristics of the computer are the same as we give in Section 3.1.

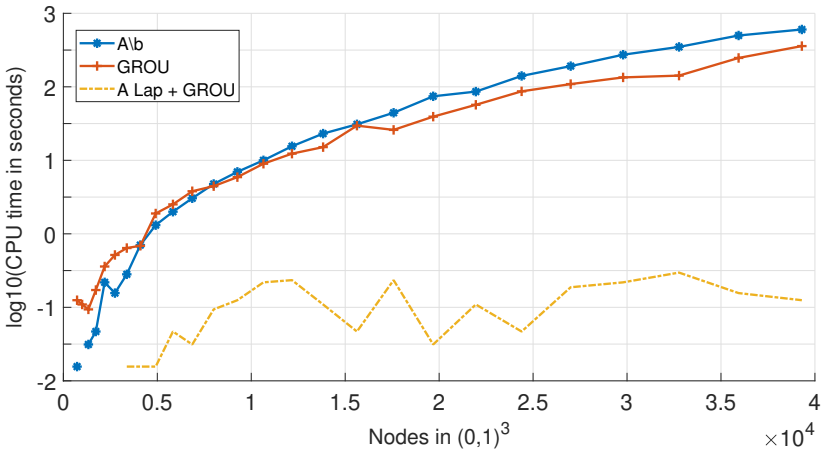

Figure 8: CPU Time, in seconds, employed to solve (7) by using the Matlab command $A\backslash b$, the GROU Algorithm 1, and the GROU Algorithm 1 with A written in Laplacian form, obtained from the Laplacian decomposition Algorithm 2. All algorithms were performed under the Matlab standard environment for basic matrix calculations.

In the first experiment (see Figure 8) the algorithms were implemented by means the Matlab standard environment to perform basic matrix calculations whereas we have done a second experiment to increase the size of the high dimensional matrices (see Figure 9). To this end, the algorithms were implemented under the Matlab environment for sparse matrices, which require less CPU memory. In this second experience we consider matrices with a number of rows (or columns) in a range from 10^3 to 100^3.

In both figures we observe how, for high-dimensional matrices, the GROU Algorithm 1 improves the CPU time of Matlab's command $A\backslash b$. But certainly the Laplacian Decomposition Algorithm 2 combined with the GROU Algorithm 1 significantly reduces the CPU time of the two previous methods.

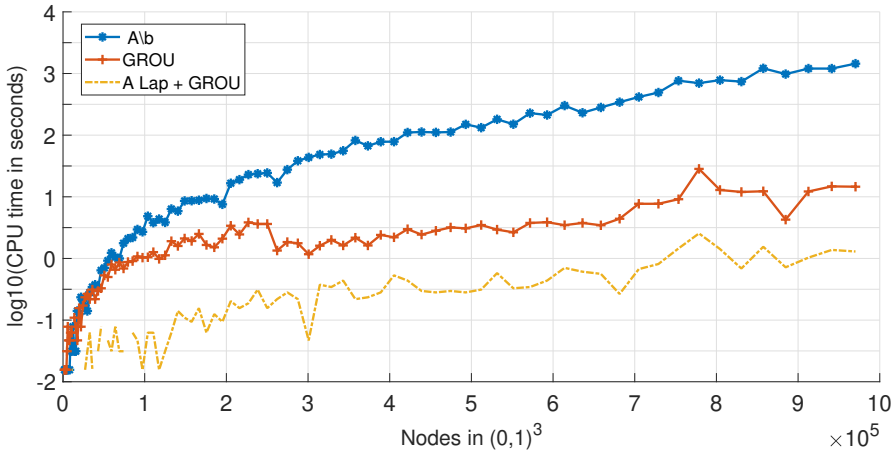

Figure 9: CPU Time, in second, employed to solve (7) by using the Matlab command $A\backslash b$, the GROU Algorithm 1, and the GROU Algorithm 1 with A written in Laplacian form, obtained from the Laplacian decomposition Algorithm 2. All algorithms were performed under the Matlab environment for matrices in sparse form.

3.5 Conclusions

We have presented a result to approximate a generic square matrix by its Laplacian form, and thus decompose it as the sum of two linearly independent matrices. This decomposition is motivated by the fact that tensor algorithms are more efficient when working with Laplacian matrices. We have also described the procedure to perform this approximation in the form of an algorithm and illustrated how it works on some basic examples.

With the proposed algorithm, we may provide an alternative way to solve linear systems, especially interesting if we combine algorithms 1 and 2, as shown in the second example presented above. Due to its structure, this matrix decomposition can be interesting for studying various types of matrices, such as sparse matrices, matrices resulting from the discretization of a PDE, adjacency matrices of simple graphs, and others. We will explore the computational benefits of this approach in future works.

References

[1] A. Ammar, F. Chinesta, and A. Falcó. On the convergence of a Greedy Rank-One Update algorithm for a class of linear systems. *Arch. Comput. Methods Eng.*, 17(4):473–486, 2010.

[2] A. Falcó. Tensor Formats Based on Subspaces are Positively Invariant Sets for Laplacian-Like Dynamical Systems. In A. Abdulle, S. Deparis, D. Kressner, F. Nobile, and M. Picasso, editors, *Numerical Mathematics and*

Advanced Applications - ENUMATH 2013, pages 315–323, Cham, 2015. Springer International Publishing.

[3] A. Falcó, W. Hackbusch, and A. Nouy. On the Dirac–Frenkel Variational Principle on Tensor Banach Spaces. *Foundations of Computational Mathematics*, 19(1):159–204, 2019.

[4] A. Falcó and A. Nouy. Proper generalized decomposition for nonlinear convex problems in tensor Banach spaces. *Numer. Math.*, 121:503–530, 2012.

[5] J. Gallier and J. Quaintance. *Differential Geometry and Lie Groups*. Springer, 2020.

[6] I. Georgieva and C. Hofreither. Greedy low-rank approximation in Tucker format of solutions of tensor linear systems. *J. Comput. Appl. Math.*, 358:206–220, 2019.

[7] G. Golub and C. Van Loan. *Matrix computations*. JHU press, 2013.

[8] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus (Second Edition)*. Springer Series in Computational Mathematics. Springer, 2019.

[9] W. Hackbusch, B. Khoromskij, S. Sauter, and E. Tyrtyshnikov. Use of tensor formats in elliptic eigenvalue problems. *Numer. Linear Algebra Appl.*, 19:133–151, 2012.

[10] G. Heidel, V. Khoromskaia, B. Khoromskij, and V. Schulz. Tensor product method for fast solution of optimal control problems with fractional multidimensional Laplacian in constraints. *J. Comput. Phys.*, 424:109865, 2021.

[11] C. Leiserson, R. Rivest, T. Cormen, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT press, 2009.

[12] D. Luzardo and A. J. Peña P. Historia del Álgebra Lineal hasta los Albores del Siglo XX. *Divulgaciones Matemáticas*, 14(2):153–170, 2006.

[13] MATLAB. *version R2021b*. The MathWorks Inc., Natick, Massachusetts, 2021.

[14] A. Nouy. *Chapter 4: Low-Rank Methods for High-Dimensional Approximation and Model Order Reduction.*, pages 171–226. Society for Industrial and Applied Mathematics, 2017.

[15] C. Quesada, G. Xu, D. González, I. Alfaro, A. Leygue, M. Visonneau, E. Cueto, and F. Chinesta. Un método de descomposición propia generalizada para operadores diferenciales de alto orden. *Rev. Int. Metod. Numer.*, 31(3):188–197, 2015.

[16] Y. Saad. Iterative methods for linear systems of equations: A brief historical journey. In S. C. Brenner, I. E. Shparlinski, C.-W. Shu, and D. B. Szyld, editors, *75 Years of Mathematics of Computation*, volume 754 of *Contemporary Mathematics*, pages 197–215. American Mathematical Society, 2020.

[17] Y. Saad and H. Van der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1):1–33, 2000.

[18] V. Simoncini. Numerical solution of a class of third order tensor linear equations. *Boll. Unione. Mat. Ital.*, 13:429–439, 2020.

[19] G. Strang. *Linear Algebra and its applications.* Belmont, CA: Thomson, Brooks/Cole, 2006.

4 Chapter

On the tensor approximation of Watts–Strogatz networks

Henrietta Swan Leavitt (1868 – 1921)
Astrónoma estadounidense, cambió la manera de observar el universo gracias a su descubrimiento sobre la luminosidad de las estrellas.

J. A. Conejero, A. Falcó, and M. Mora-Jiménez. *On the tensor approximation of Watts-Strogatz networks.* Preprint 2023.

Abstract

Small-world networks are characterized by the existence, on average, of shortest paths between any arbitrary pair of nodes with only a few edges. In order to preserve the local clustering while permitting the existence of the small-world phenomenon, Watts and Strogatz introduced their celebrated network model (Nature, 1998) exemplifying what they observed in different types of networks, such as the neural network of the *C. elegans*, the power grid network, or the collaboration network in cinema.

As the number of data increases, the networks and matrices that model it also do so, which makes it increasingly expensive to manipulate them. Recent work has shown the efficiency of tensor-based structures when performing, for example, matrix products, reducing the number of operations performed.

In the present work, we want to approximate the representative matrices of the Watts–Strogatz networks using tensor methods and compare the accuracy and the computational cost involved in operating with the original matrices and the matrices written in the approximate tensor form.

4.1 Introduction

Nowadays, networks are widely used to study and models different situations in real life, so they are a tool that gains importance in manipulating large data sets. For example, they have been used to analyze connections in a social network [10, 23, 15], the spreading of information in a network [25] to determine optimal transport routes [31, 18], and even to analyze medical images [5] or discover new drugs from chemical data [2, 19].

As the size of the matrices or vectors involved in these problems increases, most of the mechanisms we use to solve them (matrix decompositions, iterative methods, optimization algorithms, etc.) lose efficiency. This effect is known as the *curse of the dimensionality problem*. In the current situation, where massive data analysis is increasingly present, we need to search for and implement techniques that allow us to manipulate them. This is where tensor structures come into play since their use reduces and speeds up the number of operations to be carried out, as it has been recently seen [9]. Other works that illustrate the goodness of these structures in working with high-dimensional problems are [13, 14, 26].

One of the most intriguing properties in networks is the *small world phenomenon*, firstly conjectured by Milgram [21]. A small-world network is a type of graph where most of the nodes are not neighbors of each other, but a relatively small edge path can connect two randomly chosen nodes [4, 29]. In this work, we analyze the tensor decompositions of the Laplacian matrices of small-world networks in the sense of Watts and Strogatz, and compare the results obtained with the original Laplacian matrices to determine if this approach is helpful to study these networks.

The Laplacian matrix of a network is obtained as the difference between the degree and the adjacency matrix of the network. These matrices are commonly used in spectral graph theory to study the properties of the graph, in relation to its characteristic polynomial, and for representing diffusion processes on networks. They are also used to study some particular diffusion problems, as seen in [3]. Watts–Strogatz networks are constructed from a given k-regular graph, where some edges are rewired to connect a different pair of nodes with a certain probability p. For low values of p, the Laplacian matrices of these networks have a structure close to a $k+1$ diagonal matrix.

Motivated by the good properties of tensor products [12, 28], we can look for a tensor decomposition of a given Laplacian matrix $\mathscr{L}$ in the form of

$$\mathscr{L} \approx A_1 \otimes \mathrm{id}_{n_2} \otimes \cdots \otimes \mathrm{id}_{n_d} + \mathrm{id}_{n_1} \otimes A_2 \otimes \cdots \otimes \mathrm{id}_{n_d} + \cdots + \mathrm{id}_{n_1} \otimes \mathrm{id}_{n_2} \otimes \cdots \otimes A_d, \tag{10}$$

where $A_i \in \mathbb{R}^{n_i \times n_i}$, with $N = n_1 \dots n_d$. As shown in [6], we can define an algorithm that provides the best tensor approximation in form (10) of any square matrix. In

particular, for the Laplacian matrices of graphs with N nodes, this decomposition is an interesting approximation of $\mathscr{L}$, when calculating their eigenvalues or eigenvectors from algorithms that use tensors, and that, therefore, are more efficient in calculation times. So, we want to study the algorithm's performance in [6] over Laplacian matrices of Watts–Strogatz networks. In particular, we want to study how close the obtained tensor decomposition is to the Laplacian matrices of the k-regular network used for generating the Watts–Strogatz network. In order to measure how close are the initial and the tensor decompose networks, we compare the distribution of eigenvalues and eigenvectors of their corresponding Laplacian matrices [17].

Laplacian eigenvalues and eigenvectors are relevant to multi-aspects of complex network structures, like spanning trees [7], community structure [24] or algebraic connectivity [27]. Moreover, represent some significant physical or chemical properties of networks, and help to understand the relations between their topology and the dynamics. For example, in the framework of generalized Gaussian structures, the dynamics of polymer networks are fully described through the Laplacian eigenvectors, and eigenvalues [16, 22].

This Chapter is divided as follows: In Section 4.2, we briefly review the construction of Watts–Strogatz networks from k-regular networks. Then, in Section 4.3, we recall some concepts about tensors and tensor decompositions, and we will describe the methodology used for obtaining tensor decompositions in order to obtain approximations of the Laplacian matrices of Watts–Strogatz networks generated from circular 2-regular and 4-regular networks. Finally, we show our results and discussed them in Sections 4.4 and 4.5, respectively.

4.2 Regular networks and the Watts–Strogatz networks

Let us consider a non-directed simple connected network $G = (V, E)$, where V is the set of nodes, $V = \{v_1, \ldots, v_n\}$, and E is the set of edges. We recall that the adjacency matrix $\mathscr{A}$ of G is defined as $\mathscr{A}_{ij} = 1$ if $(v_i, v_j) \in E$ and 0 elsewhere. Since the network is simple, all elements in the diagonal satisfy $\mathscr{A}_{ii} = 0$, and since the network is non-directed $\mathscr{A}_{ij} = \mathscr{A}_{ji}$ for all $1 \leq i, j \leq n$.

The degree of a vertex v_i, denoted by $\delta(v_i)$, is defined as the number of edges incident on it, counting the loops twice. We can define the *degree matrix* $\mathscr{D}$ of G as a diagonal matrix where $\mathscr{D}_{ii} = \delta(v_i)$ for all $1 \leq i \leq n$. Then, the *Laplacian matrix* $\mathscr{L}$ is defined as $\mathscr{L} = \mathscr{D} - \mathscr{A}$. Since $\mathscr{A}$ is symmetric and $\mathscr{D}$ is diagonal, we have that $\mathscr{L}$ is symmetric, too. Laplacian matrices of simple networks satisfy that the sum of all the elements in a row (or column) is equal to 0. Therefore, $\lambda = 0$ is always an eigenvalue (with associated eigenvector $u^\top = (1, \ldots, 1)$), of the Laplacian matrix. On the other hand, for every node $v_i \in V$, we can define the state on this node at instant t as $u_i(t)$. So, we can express

the diffusion between adjacent nodes as the *Abstract Cauchy Problem* on ℓ_2 defined on (11).

$$\begin{cases} \dot{u}_i(t) = \mathscr{D} \sum\limits_{(i,j)\in E} \big(u_j(t) - u_i(t)\big), \quad i = 1, \ldots, n \\ u_i(0) = u_0(i), \quad u_0(i) \in \mathbb{R} \end{cases} \tag{11}$$

where D is the diffusion coefficient. If $u = (u_i(t))_{i=1}^n \in \ell_2$, where n is the number of nodes, this diffusion equation can be stated as $u(t)/dt = -D\mathscr{L}u(t)$, in terms of the Laplacian matrix of the network.

In the sequel, we deal with circular networks, in which we represent the nodes presented over a circumference and link them with their neighbors. We recall that a network is k-regular if $\delta(v_i) = k$ for all $1 \le i \le n$. So as to, a circular network C_{2n} is a 2-regular network with n vertices, $n \ge 3$. Up to isomorphism, there is a unique representation of C_{2n}, with the following Laplacian matrix:

$$\mathscr{L} = \begin{pmatrix} 2 & -1 & 0 & \ldots & 0 & -1 \\ -1 & 2 & -1 & \ldots & 0 & 0 \\ 0 & -1 & 2 & \ldots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ldots & -1 & 2 & -1 \\ -1 & 0 & \ldots & 0 & -1 & 2 \end{pmatrix}. \tag{12}$$

We can also consider 4-regular circular networks, denoted by C_{4n}, with n nodes, with $n \ge 5$. Here, a node is connected to the next two nodes on the right and the next two on the left until both cycles are completed (doubly circular graphs), see Figure 10.

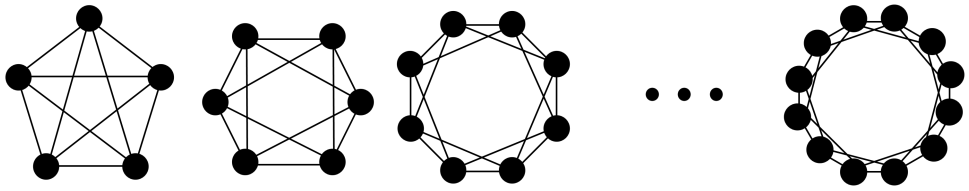

Figure 10: Examples of 4-regular graphs: $C_{4,5}, C_{4,6}, C_{4,7}$, and $C_{4,12}$.

The Laplacian matrix of 4-regular circular network has 4 in all the elements in the main diagonal, and -1 in the two diagonals above and below the main one, as we can see below

$$\mathscr{L} = \begin{pmatrix} 4 & -1 & -1 & 0 & \dots & -1 & -1 \\ -1 & 4 & -1 & -1 & \dots & 0 & -1 \\ -1 & -1 & 4 & -1 & \dots & 0 & 0 \\ 0 & -1 & -1 & 4 & \dots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & -1 & 4 & -1 & -1 \\ -1 & 0 & \dots & -1 & -1 & 4 & -1 \\ -1 & -1 & \dots & 0 & -1 & -1 & 4 \end{pmatrix}. \tag{13}$$

Watts and Strogatz introduce randomness into k-regular graphs to construct networks that preserve a high local clustering coefficient while reducing the average shortest length path between any pair of nodes to resemble the small world phenomenon. They set an algorithm in order to construct these networks. First, we start with a circular $2k$-regular network of n nodes, denoted by $C_{2k,n}$, where each node is connected with its closest 2k neighbor nodes. Then, we set a probability p, and for every existing edge, we decide whether to rewire a different pair of nodes, with probability p, or to leave it as it is.

The case $p = 0$ return the original $2k$-regular network $C_{2k,n}$. As we increase the value of p, we introduce randomness in the connectivity while maintaining the average degree equal to $2k$. In the case of $p = 1$, the generated network is completely random. We can appreciate how the network changes while increasing the value of p in Figure 11.

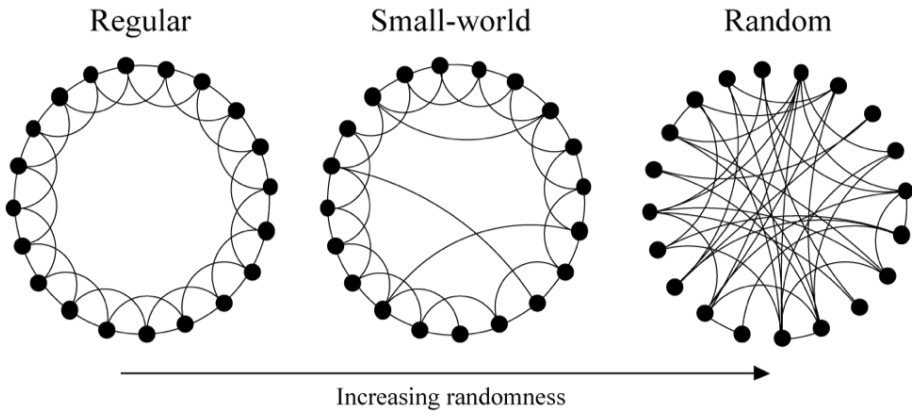

Figure 11: Watts–Strogatz networks constructed from a circular $C_{4,n}$ network for different values of p: $p = 0$ (left), $0 < p < 1$ (center), and $p = 1$ (right) [30].

4.3 The best tensor based decomposition

First of all, let's remember that the *Kronecker product* of two matrices $A \in \mathbb{R}^{N_1 \times M_1}$, $B \in \mathbb{R}^{N_2 \times M_2}$ is defined by

$$A \otimes B = \begin{pmatrix} A_{1,1}B & A_{1,2}B & \dots & A_{1,M_1}B \\ A_{2,1}B & A_{2,2}B & \dots & A_{2,M_1}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{N_1,1}B & A_{N_1,2}B & \dots & A_{N_1,M_1}B \end{pmatrix} \in \mathbb{R}^{N_1 N_2 \times M_1 M_2}.$$

And, also, some of the well-known properties of the Kronecker product are:

1. $A \otimes (B \otimes C) = (A \otimes B) \otimes C$.
2. $(A + B) \otimes C = (A \otimes C) + (B \otimes C)$.
3. $AB \otimes CD = (A \otimes C)(B \otimes D)$.
4. $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.
5. $(A \otimes B)^{\top} = A^{\top} \otimes B^{\top}$.
6. $\operatorname{tr}(A \otimes B) = \operatorname{tr}(A) \cdot \operatorname{tr}(B)$.

One of the most popular techniques among the algorithms based on tensor products strategies [28], is the Proper Generalized Decomposition (PGD) family, based on the so-called Greedy Rank-One Updated (GROU) algorithm [1, 11]. In particular, they impose a separation of variables to approximate the exact solution of a problem without knowing, in principle, the functions involved in this decomposition [8].

There is a particular type of matrices to solve high-dimensional linear systems for which the GROU algorithm works particularly well, those that are written in the form

$$A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]} \doteq \sum_{i=1}^{d} \mathrm{id}_{n_1} \otimes \cdots \otimes \mathrm{id}_{n_{i-1}} \otimes A_i \otimes \mathrm{id}_{n_{i+1}} \otimes \cdots \otimes \mathrm{id}_{n_d}, \tag{14}$$

where $A \in \mathbb{R}^{N \times N}$, with $N = n_1 \dots n_d$, $A_i \in \mathbb{R}^{n_i \times n_i}$ for $1 \le i \le d$, and id_{n_j} is the identity matrix of size $n_j \times n_j$. These matrices are called Laplacian-like matrices, since they can be easily related to the classical Laplacian operator [13, 14]. However, to avoid falling into a nomenclature ambiguity, we will refer to this structure as the tensor decomposition (of rank d) of a matrix.

So, we are interested in trying to achieve a structure similar to (14), to study the matrices of large-dimensional graphs. To do this, we will use the following theorem (Theorem 7 in [6]), which describes the Greedy Algorithm that calculates the best tensor decomposition (of rank d) of a given square matrix A.

Theorem 4. *Let A be a matrix in $\mathbb{R}^{N\times N}$, with $N = n_1\cdots n_d$, such that $tr(A) = 0$, and let*

$$\Delta = \{A \in \mathbb{R}^{N\times N} :\ A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[n_i]},\ \textit{with}\ \mathrm{tr}(A_i) = 0,\ i = 1,\ldots,d\}.$$

If we consider the following iterative procedure:

1. *Take $X_k^{(0)} = 0$ for $1 \le k \le d$.*
2. *For each $\ell \ge 1$ compute for $1 \le i \le d$ the matrix $U_i^{(\ell)}$ as*

$$U_i^{(\ell)} = \arg \min_{\substack{U_i \in \mathbb{R}^{n_i\times n_i} \\ \mathrm{tr}(U_i)=0}} \left\| A - \sum_{k=1}^{i-1} X_k^{(\ell)} \otimes \mathrm{id}_{[n_k]} - \xi(U_i) \otimes \mathrm{id}_{[n_i]} - \sum_{k=i+1}^{d} X_k^{(\ell-1)} \otimes \mathrm{id}_{[n_k]} \right\|,$$

where

$$\xi(U_i) = X_i^{(\ell-1)} + U_i,$$

and put $X_i^{(\ell)} = X_i^{(\ell-1)} + U_i^{(\ell)}$,

then

$$\lim_{\ell\to\infty} \sum_{k=1}^{d} X_k^{(\ell)} \otimes \mathrm{id}_{[n_k]} = P_\Delta(A),$$

where $P_\Delta(A)$ solves the problem

$$\min_{A^*\in\Delta} \left\| A - A^* \right\|.$$

The algorithm described in the previous theorem can be written in the form of the pseudocode 3.

4.3.1 Tensor decomposition of Watts-Strogatz networks

We study the tensor decomposition of Watts–Strogatz networks for low values of p. The Laplacian matrices of these networks will be pretty similar close to the Laplacian matrix of the associated circular k-regular network, with most of their non-null elements grouped in the main diagonals of the matrix.

We will consider the families of 2-regular and 4-regular networks, and we will modify them to get Watts–Strogatz networks for small probability values p, ranging between 0.05 and 0.2. Then, we will compute the Laplacian tensor-based approximation, with canonical rank d, as given in (10). With these numerical experiments, we want to determine whether, given a Watts–Strogatz network, its Laplacian matrix decomposition, written in the form (10), is a good approximation of the Laplacian matrix of the network and, if writing the matrix in tensor form, this speeds up the computations involving it, for example, in the computation of eigenvalues and eigenvectors.

For this purpose, we have analyzed networks of different sizes, from 100 to 2000 nodes per graph. For each network, we have conducted the following study, which consists

of two parts.

First part: Analysis of the k-regular network

1. We start with a 2 (or 4) regular circular network G and we obtain its Laplacian matrix $\mathscr{L}$.

2. We compute the tensor-based approximation of $\mathscr{L}$, and we denote it by $\mathscr{L}_{\otimes}$. This approximation is obtained using the algorithm described in [6], which is summarized in the pseudocode of Algorithm 3, using a maximum of 50 iterations and a tolerance of $2.22e{-}4$.

3. To determine how good such an approximation is, the resulting residual will be computed as $\texttt{res} = \texttt{norm}(\mathscr{L} - \mathscr{L}_{\otimes})/\texttt{norm}(\mathscr{L})$.

4. Lastly, we study the relationship between eigenvalues and eigenvectors of $\mathscr{L}$ and $\mathscr{L}_{\otimes}$, and compare the time to obtain them in each case using the command `eigs`. To measure the relative distance between the eigenvalues obtained, we will use the indicator $\texttt{dist}_{\lambda} = \texttt{norm}(\lambda - \lambda_{\otimes})/\texttt{norm}(\lambda)$; and we will measure the fetch time of λ and $\lambda_{\otimes}$, with the `tic-toc` instruction implemented in Matlab.

It is worth mentioning that each network has a different decomposition depending on the prime factor decomposition of the network size N. Thus, if $N = n_1 \cdots n_d$, we will look for matrices $A_i \in \mathbb{R}^{n_i \times n_i}$. We will call the vector $[n_1, \ldots, n_d]$ as *the vector of sizes.*

For the sake of completeness, we present the pseudocode of the algorithm used for this decomposition, which has been described in [6].

Algorithm 3 Laplacian decomposition Algorithm

procedure LAP(A^*, iter_max, tol)
  $A = A^* - (\mathrm{tr}(A)/N)\mathrm{id}_N$, iter = 1, Lap = 0
  while iter < iter_max **do**
    $A \leftarrow A - \mathtt{Lap}$
    for $k = 1, 2, \ldots, d$ **do**
      $P_k(A) = \mathrm{id}_{n_1} \otimes \cdots \otimes \mathrm{id}_{n_{k-1}} \otimes X_k \otimes \mathrm{id}_{n_{k+1}} \otimes \cdots \otimes \mathrm{id}_{n_d}$
      $X_k \leftarrow \min_{X_k} \|A - \sum_{i=1}^{k} P_i(A)\|$
      $\mathtt{Lap} = \mathtt{Lap} + P_k(A)$
    end for
    if $\|A - \mathtt{Lap}\| <$ tol **then goto** 14
    end if
    iter = iter + 1
  end while
  return Lap
end procedure

In this first part, there is no randomness introduced, so the tensor approximation $\mathscr{L}_\otimes$ of $\mathscr{L}$, for a concrete graph G is always the same.

Second part: Analysis of the Watts–Strogatz networks

1. We take the 2 (or 4) regular circular networks considered in the first part.
2. For each one of these networks, name it G, we convert it into different Watts–Strogatz networks G'_{p_i}, using four different probabilities: $p_1 = 0.05$, $p_2 = 0.1$, $p_3 = 0.15$, and $p_4 = 0.2$.
3. Once fixed p_i and for every edge $e \in E$, we generate a random r_e number from a uniform distribution on (0,1). This number r_e will be used for deciding if we rewire the nodes connected by e or not, using the following rule.
   - If $r_e < p_i$, then the edge e is rewired, connecting a new pair of nodes chosen randomly among all of them that were not already connected.
   - If $p_i \leq r_e$, the edge is maintained as it is.
4. We repeat the process described in the first part for each network G'_{p_i}. We first obtain the associated Laplacian matrices $\mathscr{L}'_{p_i}$, and then we compute the respective approximate Laplacians $\mathscr{L}'_{\otimes,p_i}$.

5. Finally, we obtain the eigenvalues and eigenvectors of matrices $\mathscr{L}'_{p_i}$ and $\mathscr{L}'_{\otimes,p_i}$, and we measure the difference between the eigenvectors and eigenvalues, as described at the end of the first part.

To make a more robust and consistent analysis in this second part, in order to avoid randomness effects, we repeat 1000 times this scheme. The charts and results in Section 4.4 will show the averaged values obtained from the 1000 repetitions.

4.4 Results analysis

We will now show and comment on the results of the experiment described in Section 4.3.1. The experiments were carried out on a computer with the following characteristics: 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz, RAM 16,0 GB, 64 bit operating system; and a Matlab version R2021b [20].

First, we compare the Laplacian matrices of the 2 and 4-regular graphs and their tensor approximations. In Figure 12, we compare the computation time taken to compute the eigenvalues and eigenvectors in the starting $\mathscr{L}$ Laplacian matrices and their approximate $\mathscr{L}_{\otimes}$ matrices in tensor form. As the number of nodes increases, the computation time for the original matrices increases much faster than the times required by the tensor-based approximate matrices.

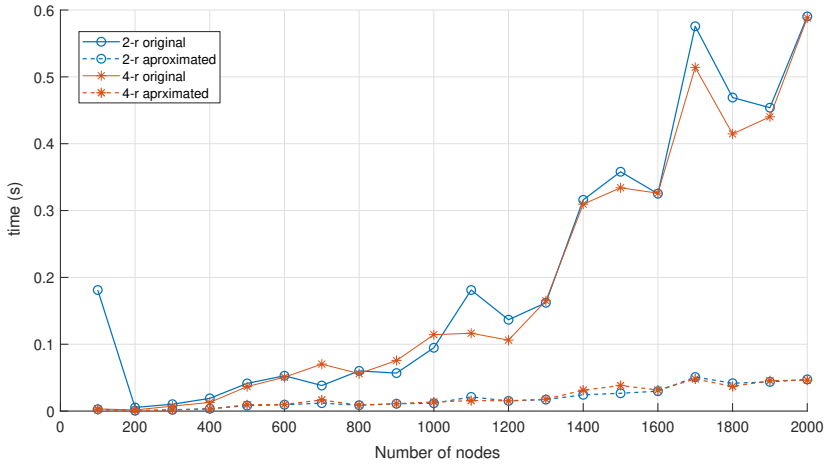

Figure 12: Eigenvalue calculation time

We also compare the eigenvalues of each of the matrices $\mathscr{L}$ and $\mathscr{L}_{\otimes}$, measuring the norm of their difference. In Figure 13, we see that the results oscillate between the values 0.03 and 0.12. So that, the eigenvalues of the $\mathscr{L}_{\otimes}$ approximate pretty well the eigenvalues of the original matrix $\mathscr{L}$. We observe that the distance between the

eigenvalues of the Laplacian matrix and its approximation depends on the network size but not on being a 2-regular or a 4-regular network.

Also noteworthy that there are cases where the approximations are much better than in the rest. This occurs for 700, 1100, 1300, 1700, and 1900 nodes, where prime numbers $7, 11, 13, 17$, and 19 are involved in the factorization of the network size, which yields vectors of sizes of shorter lengths. For example, if we look at $N = 1700$, the vector of sizes obtained from the factorization of N will be $\texttt{factor}(N) = [2,2,5,5,17]$; which is shorter than the vector associated to $N = 1800$, $\texttt{factor}(N) = [2,2,2,3,3,5,5]$. This fact illustrates the importance of the number (and which) prime factors appear in the decomposition of the network size into prime factors.

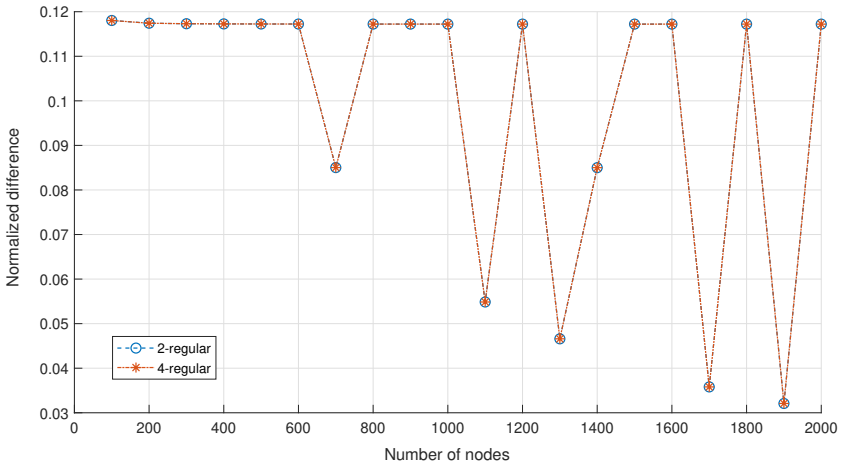

Figure 13: Distance (norm) between eigenvalues

In the second part, we study the approximations obtained of the Watts–Strogatz Laplacian matrices. In this case, we group the figures into two different groups: in Figures 14 and 15, we show the computation times of eigenvalues and eigenvectors of the original Laplacian matrices and those written in tensor form for the different probabilities $p_1, \ldots, p_4$.

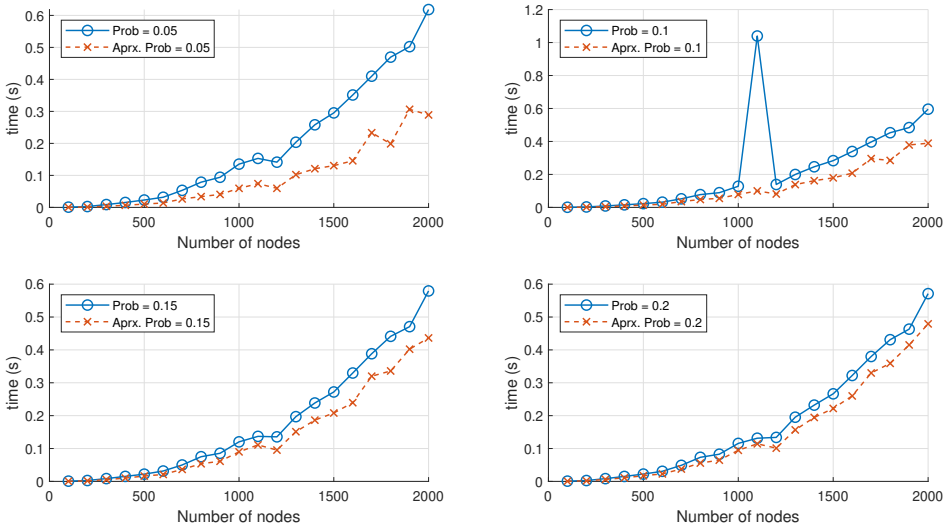

Figure 14: Mean time spent computing eigenvectors in Watts–Strogatz networks from 2-RG

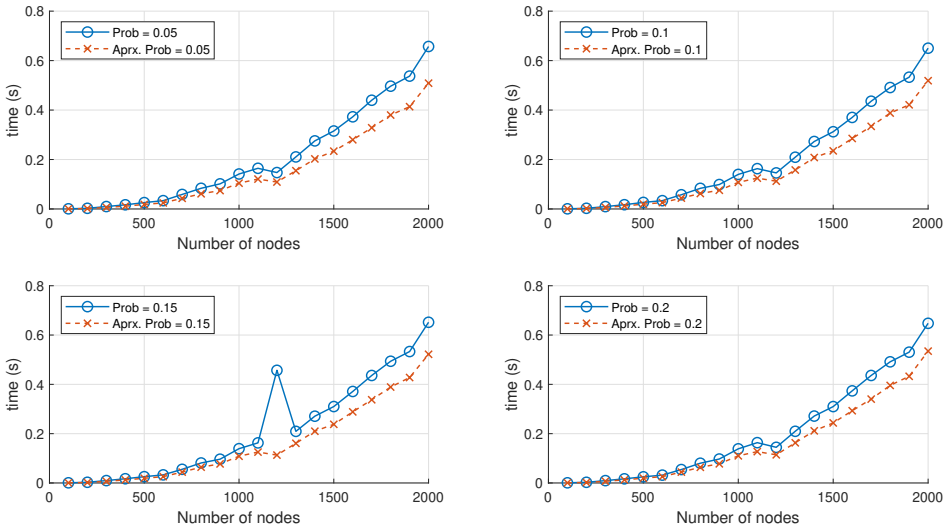

Figure 15: Mean time spent computing eigenvectors in Watts–Strogatz networks from 4-RG

Again, we see that we need less time to calculate eigenvalues and eigenvectors of the tensor approximate matrices than for the original Laplacian ones. Besides, the time increases as long as the network size does.

Finally, in Figures 16 and 17, we measure the difference between the eigenvalues of the original Laplacian matrices and those from the tensor-based approximations. The norms of the difference take small values, less than 0.26 for Watts–Strogatz networks obtained from 2-regular networks, and less than 0.19 for those obtained from

4-regular networks. As the probability increases, we obtain worse approximations, since networks become more *random* and less *regular*, which results in a structure that is harder to approximate in tensor form.

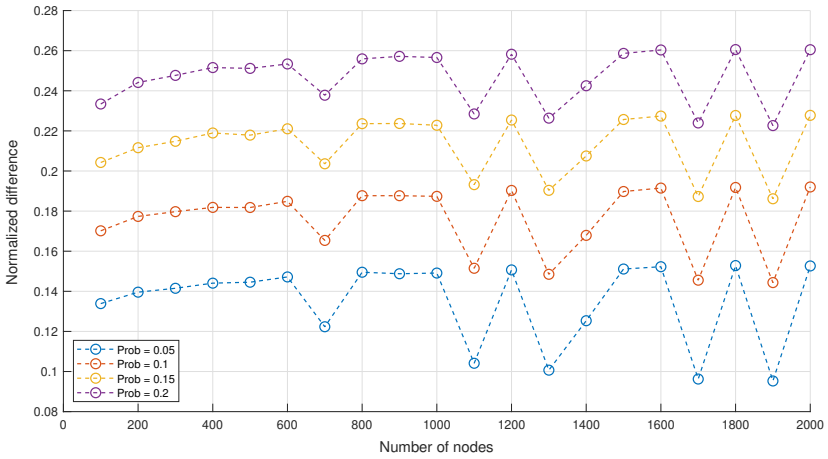

Figure 16: Difference (in norm) between the eigenvalues obtained in Watts–Strogatz networks from 2-regular graphs.

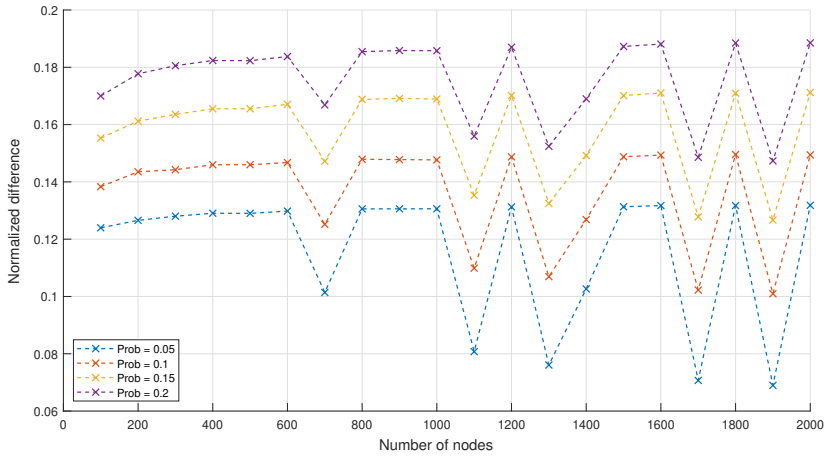

Figure 17: Difference (in norm) between the eigenvalues obtained in Watts–Strogatz networks from 4-regular graphs.

Nevertheless, we remark on the importance of the vectors of sizes and which prime numbers are involved in the factorization, as the spikes in the graph appeared similarly for each network size.

4.5 Conclusions

In this work, we have used tensor decompositions of the form (10) to approximate Laplacian matrices of 2 and 4-regular networks, and of some Watts–Strogatz networks

obtained from these regular networks.

By introducing the randomness in Watts–Strogatz networks, we lose the regularity in the structure of the Laplacian matrices, which is reflected in the approximations that we obtained: as the randomness increases, the matrices in tensor form approximate worse the original Laplacian matrices, although we still obtain interesting results. Based on the results obtained, we can affirm that this technique allows us to work with tensor approximations of the different networks, by reducing the costs of time spent in computations, as is the case for computing eigenvectors and eigenvalues.

It may be interesting to carry out similar experiments, using regular networks of higher degree $(6, 8, \ldots)$, to see if the 'repetition of patterns' is still maintained in the resulting graphs. It is also interesting to ask what would happen if we modified the *vector of sizes*, that is, the number of matrices that we use in each decomposition and what would be the optimal size of the matrices in that case. For example, if we only use two matrices, better results are obtained if the matrices have similar sizes, or if, on the contrary, they have the smallest and largest possible size, respectively? Therefore, there are still interesting studies to be carried out to estimate the goodness of the use of tensor decompositions.

References

[1] A. Ammar, F. Chinesta, and A. Falcó. On the convergence of a Greedy Rank-One Update algorithm for a class of linear systems. *Arch. Comput. Methods Eng.*, 17(4):473–486, 2010.

[2] M. Ay, K.-I. Goh, M. Cusick, A.-L. Barabasi, M. Vidal, and et al. Drug–target network. *Nat. Biotechnol.*, 25(10):1119–1127, 2007.

[3] J. Banasiak and M. Mokhtar-Kharroubi. *Evolutionary Equations with Applications in Natural Sciences*. Springer, 2015.

[4] A.-L. Barabási. *Network science*. Cambridge University Press, 2016.

[5] X. Chen and L. Pan. A survey of graph cuts/graph search based medical image segmentation. *IEEE Rev. Biomed. Eng.*, 11:112–124, 2018.

[6] J. A. Conejero, A. Falcó, and M. Mora-Jiménez. Structure and Approximation Properties of Laplacian-Like Matrices. *Results in Mathematics*, 78(184), 2023.

[7] R. Dobrin and P. M. Duxbury. Minimum spanning trees on random networks. *Phys. Rev. Lett.*, 86:5076–5079, May 2001.

[8] A. Falcó and A. Nouy. Proper generalized decomposition for nonlinear convex problems in tensor Banach spaces. *Numer. Math.*, 121:503–530, 2012.

[9] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 10 2022.

[10] L. Freeman. The development of social network analysis. *A study in the sociology of science*, 1(687):159–167, 2004.

[11] I. Georgieva and C. Hofreither. Greedy low-rank approximation in Tucker format of solutions of tensor linear systems. *J. Comput. Appl. Math.*, 358:206–220, 2019.

[12] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus (Second Edition)*. Springer Series in Computational Mathematics. Springer Cham, 2019.

[13] W. Hackbusch, B. Khoromskij, S. Sauter, and E. Tyrtyshnikov. Use of tensor formats in elliptic eigenvalue problems. *Numer. Lin. Algebra Appl.*, 19:133–151, 2012.

[14] G. Heidel, V. Khoromskaia, B. Khoromskij, and V. Schulz. Tensor product method for fast solution of optimal control problems with fractional multidimensional Laplacian in constraints. *J. Comput. Phys.*, 424:109865, 2021.

[15] W. Jiang, G. Wang, M. Bhuiyan, and J. Wu. Understanding graph-based trust evaluation in online social networks: Methodologies and challenges. *ACM Comput. Surv.*, 49(1), 2016.

[16] C. Liu, Y. Pan, J. Li, and L. Dai. The normalized laplacians on both two iterated constructions associated with graph and their applications. *Journal of Applied Mathematics and Physics*, 8:838–860, 2020.

[17] H. Liu, M. Dolgushev, Y. Qi, and Z. Zhang. Laplacian spectra of a class of small-world networks and their applications. *Scientific Reports*, 5(1):1–7, 2015.

[18] K. M. and K. Kavitha. A comparative study of transportation problem by graph theoretical algorithms. *Advances in Mathematics: Scientific Journal*, 9:6251–6260, 08 2020.

[19] F. MacLean. Knowledge graphs and their applications in drug discovery. *Expert Opinion on Drug Discovery*, 16:1–13, 04 2021.

[20] MATLAB. *version R2021b*. The MathWorks Inc., Natick, Massachusetts, 2021.

[21] S. Milgram. The small world problem. *Psychol. Today*, 2(1):60–67, 1967.

[22] P. Nakkirt. The eigenvalue distribution of the Watts-Strogatz random graph. *arXiv preprint arXiv:2009.00332*, 2020.

[23] M. Newman, D. Watts, and S. Strogatz. Random graph models of social networks. *Proc. Natl. Acad. Sci. USA*, 99(suppl_1):2566–2572, 2002.

[24] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006.

[25] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Let.*, 86(14):3200, 2001.

[26] C. Quesada, G. Xu, D. González, I. Alfaro, A. Leygue, M. Visonneau, E. Cueto, and F. Chinesta. Un método de descomposición propia generalizada para operadores diferenciales de alto orden. *Rev. Int. Metod. Numer.*, 31(3):188–197, 2015.

[27] B. RB. *Graphs and matrices*, volume 27. Springer, 2010.

[28] V. Simoncini. Numerical solution of a class of third order tensor linear equations. *Boll. Unione. Mat. Ital.*, 13:429–439, 2020.

[29] D. Watts. *Six degrees: The science of a connected age*. W. W. Norton & Company, 01 2003.

[30] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 11 1998.

[31] R. Żochowska and P. Soczówka. Analysis of selected transportation network structures based on graph measures. *Scientific Journal of Silesian University of Technology. Series Transport*, 98:223–233, 03 2018.

5 CHAPTER

A pre-processing method for the implementation of the GROU Algorithm for a class of linear systems

Esther Zimmer Lederberg (1922 – 2006)
Microbióloga estadounidense, pionera en genética bacteriana, desarrolló técnicas que contribuyeron al entendimiento de cómo funcionan los genes.

J. A. Conejero, A. Falcó, and M. Mora-Jiménez. *A pre-processing method for the implementation of the GROU Algorithm for a class of linear systems.* AIMS Mathematics, 2023, 8(11): 25633-25653.

Abstract

Algorithms that use tensors are increasingly important due to the goodness of this operation when performing calculations with large amounts of data. Among them, we find the algorithms that search for the solution of a linear system in separated form, where the Greedy Rank-One Update method stands out, the starting point of the famous Proper Generalized Decomposition family.

When the matrices of these systems have the particular structure of a Laplacian-type matrix, the convergence of the previous methods is faster and more accurate. The Laplacian Decomposition Algorithm calculates the Laplacian matrix that best approximates a given square matrix. When the residue of this approximation is small, we will be able to solve the linear system associated with a Laplacian-type matrix and thus obtain an approximation of the solution of the original system, with a lower computational cost.

In this chapter we prove that the discretization of a general PDE of the second order can be written as a linear system with a Laplacian-type matrix.

5.1 Introduction

Working with large amounts of data is one of the main challenges we face today. With the rise of social networks and rapid technological advances, we must develop tools that allow us to work with so much information. At this point the use of tensor products comes into play, since their use reduces and speeds up the number of operations to be carried out. Proof of this is the recent article [6], where tensor products are used to speed up the calculation of matrix products. Other articles that exemplify the goodness of this operation are [9], where the solution of 2,3-dimensional optimal control problems with spectral fractional Laplacian type operators is studied, and [12], where high-order problems are studied using proper generalized decomposition methods.

When we try to solve a linear system of the form $A\mathbf{x} = \mathbf{b}$, in addition to the classical methods, there are methods based on tensors that can be more efficient [11], since the classical methods face the *problem of the curse of dimensionality*, which makes them lose effectiveness as the size of the problem increases. The tensor methods look for the solution in separated form, that is, as the tensor combination

$$\mathbf{x} = \sum_{j=1}^{\infty} \mathbf{x}_1^j \otimes \cdots \otimes \mathbf{x}_d^j,$$

where $\mathbf{x}_i^j \in \mathbb{R}^{N_i}$ and d is the dimension of the problem. The main family of methods that solves this problem is PGD [5], and it is based on the GROU algorithm [1, 7]. This algorithm calculates the solution of the linear system $A\mathbf{x} = \mathbf{b}$ in separated form and, for this, in each iteration, it updates the approximation of the solution with the term resulting from minimizing the remaining residue. Furthermore, there are certain square matrices for which the GROU algorithm improves their convergence, matrices of the form

$$A = \sum_{i=1}^{d} \mathrm{id}_{N_1} \otimes \cdots \otimes \mathrm{id}_{N_{i-1}} \otimes A_i \otimes \mathrm{id}_{N_{i+1}} \otimes \cdots \otimes \mathrm{id}_{N_d}.$$

These matrices are called Laplacian-like matrices, because of their relationship with the Laplace operator.

However, it is not easy to obtain the matrix of problem A with that structure. To do this, we can use the Laplacian Decomposition Algorithm described in [2], which, given the value of d, calculates the best Laplacian approximation of the matrix A and returns it, L_A, and its residue, R_A. Thus, we can rewrite the linear system as $(L_A + R_A)\mathbf{x} = \mathbf{b}$, and when the value of the remainder is small, we can approximate the solution of the system $\mathbf{x}^*$ by the solution of the Laplacian system $\mathbf{x}_L$.

This fact is specially interesting in the case of Partial Differential Equation. We study the Laplacian decomposition of the matrix that comes from the discretization of a

general PDE of the second order of the form

$$\alpha \mathbf{u}_{xx} + \beta \mathbf{u}_{yy} + \gamma \mathbf{u}_x + \delta \mathbf{u}_y + \mu \mathbf{u} = \mathbf{f}.$$

Besides, to compare different methods to solve these equations, we consider some particular cases: the Helmholtz equation, which solve the eigenvalue problem for the Laplace operator; and the famous Poisson Equation. Furthermore, to illustrate that it is not necessary to be limited to the second order, we consider the 4-order Swift-Hohenberg equation

$$\frac{\partial u}{\partial t} = \varepsilon - \left(1 + \frac{\partial^2}{\partial x^2}\right)^2 u.$$

This equation is noted for its pattern-forming behaviour, and it was derived from the equations for thermal convection [13].

In this article, we review the Laplacian Decomposition Algorithm and show how it behaves in combination with the GROU Algorithm, Section 5.3. To do this, we begin by recalling some tensor concepts in Section 5.2. Finally, we show the numerical examples commented below in Section 5.5.

5.2 Preliminary definitions and results

First at all we introduce some notation that we use along the paper. We denote by $\mathbb{R}^{N\times M}$, the set of $N \times M$-matrices and by A^T the transpose of a given matrix A. As usual we use

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x}$$

to denote the Euclidean inner product in $\mathbb{R}^N$, and its corresponding 2-norm, by $\|x\|_2 = \langle \mathbf{x}, \mathbf{x} \rangle^{1/2}$. Let id_N be the $N \times N$-identity matrix and when the dimension is clear from the context, we simply denote it by I. Given a sequence $\{\mathbf{u}_j\}_{j=0}^{\infty} \subset \mathbb{R}^N$, we say that a vector $\mathbf{u} \in \mathbb{R}^N$ can be written as

$$\mathbf{u} = \sum_{j=0}^{\infty} \mathbf{u}_j$$

if and only if

$$\lim_{n\to\infty} \sum_{j=0}^{n} \mathbf{u}_j = \mathbf{u}$$

in the $\|\cdot\|_2$-topology.

The *Kronecker product* of two matrices $A \in \mathbb{R}^{N_1\times M_1}$, $B \in \mathbb{R}^{N_2\times M_2}$ is defined by

$$A \otimes B = \begin{pmatrix} A_{1,1}B & A_{1,2}B & \dots & A_{1,M_1}B \\ A_{2,1}B & A_{2,2}B & \dots & A_{2,M_1}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{N_1,1}B & A_{N_1,2}B & \dots & A_{N_1,M_1}B \end{pmatrix} \in \mathbb{R}^{N_1N_2\times M_1M_2}.$$

We can see some of the well-known properties of the Kronecker product in [1].

As we already said, we are interested solve a high-dimensional linear system $A\mathbf{x} = \mathbf{b}$ obtained from a discretization of a Partial Differential Equation. We are interested to solve it by using a tensor-based algorithm, so, we are going to look for an approximation of the solution in separated form. To see this, we assume that the coeffcient matrix A is a $(N_1 \cdots N_d) \times (N_1 \cdots N_d)$-dimensional invertible matrix, for some $N_1, \cdots, N_d \in \mathbb{N}$. Next, we look for an approximation (of rank n) of $A^{-1}\mathbf{b}$ of the form

$$A^{-1}\mathbf{b} \approx \sum_{j=1}^{n} \mathbf{x}_1^j \otimes \cdots \otimes \mathbf{x}_d^j. \tag{15}$$

To do this, given $\mathbf{x} \in \mathbb{R}^{N_1 \cdots N_d}$ we say that $\mathbf{x} \in \mathscr{R}_1 = \mathscr{R}_1(N_1, N_2, \ldots, N_d)$ if $\mathbf{x} = \mathbf{x}_1 \otimes \mathbf{x}_2 \otimes \cdots \otimes \mathbf{x}_d$, where $\mathbf{x}_i \in \mathbb{R}^{N_i}$, for $i = 1, \ldots, d$. For $n \geq 2$ we define inductively $\mathscr{R}_n = \mathscr{R}_n(N_1, N_2, \ldots, N_d) = \mathscr{R}_{n-1} + \mathscr{R}_1$, that is,

$$\mathscr{R}_n = \left\{ \mathbf{x} : \mathbf{x} = \sum_{i=1}^{k} \mathbf{x}^{(i)}, \mathbf{x}^{(i)} \in \mathscr{R}_1 \text{ for } 1 \leq i \leq k \leq n \right\}.$$

Note that $\mathscr{R}_n \subset \mathscr{R}_{n+1}$ for all $n \geq 1$.

To perform (15), what we will do is minimizing the difference

$$\left\| \mathbf{b} - A\left(\sum_{j=1}^{n} \mathbf{x}_d^j \otimes \cdots \otimes \mathbf{x}_d^j \right) \right\|_2,$$

that is, solve the problem

$$\underset{\mathbf{u} \in \mathscr{R}_n}{\arg\min} \|\mathbf{b} - A\mathbf{u}\|_2. \tag{16}$$

Here $\|\cdot\|_2$ is the 2-norm, or the Frobenius norm, defined by

$$\|A\|_2 = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{i,j}|^2} = \sqrt{\operatorname{tr}(A^\top A)}, \quad \text{for} \quad A \in \mathbb{R}^{m \times n}.$$

Unfortunately, from Proposition 4.1 (a) of [3], we have that the set $\mathscr{R}_n$ is not necessarily (or even usually) closed for each $n \geq 2$. In consequence, no best rank-n approximation exists, that is, (16) has no solution. However, from Proposition 4.2 of [3] it follows that $\mathscr{R}_1$ is a closed set in any norm-topology. This fact allows us to introduce the following algorithm.

5.2.1 Greedy Rank-One Update Algorithm

The Greedy Rank-One Update (GROU, in short) Algorithm is an iterative method to solve linear systems of the form $A\mathbf{x} = \mathbf{b}$ by using only rank-one updates. Thus, given

$A \in \mathrm{GL}(\mathbb{R}^{N\times N})$ with $N = N_1 \cdots N_d$, and $\mathbf{b} \in \mathbb{R}$ we can obtain an approximation of the form

$$A^{-1}\mathbf{b} \approx \mathbf{u}_n = \sum_{j=1}^{n} \mathbf{x}_1^j \otimes \cdots \otimes \mathbf{x}_d^j$$

for some $n \geq 1$ and $\mathbf{x}_i^j \in \mathbb{R}^{N_i}$, for $i = 1,2,\ldots,d$ and $j = 1,2,\ldots,n$ [1]. We proceed with the following iterative procedure (see Algorithm 4 below): let $\mathbf{u}_0 = \mathbf{y}_0 = 0$, and for each $n \geq 1$ take

$$\mathbf{r}_{n-1} = \mathbf{b} - A\mathbf{u}_{n-1}, \tag{17}$$

$$\mathbf{u}_n = \mathbf{u}_{n-1} + \mathbf{y}_n \quad \text{where} \quad \mathbf{y}_n \in \underset{\mathbf{u}\in\mathscr{R}_1}{\arg\min} \|\mathbf{r}_{n-1} - A\mathbf{u}\|_2. \tag{18}$$

Since $\mathbf{u}_n \approx A^{-1}\mathbf{b}$, we can define the $\mathrm{rank}_\otimes$ for $A^{-1}\mathbf{b}$ obtained by the GROU Algorithm as

$$\mathrm{rank}_\otimes(A^{-1}\mathbf{b}) = \begin{cases} \infty & \text{if} \quad \{j \geq 1 : \mathbf{y}_j = 0\} = \emptyset, \\ \min\{j \geq 1 : \mathbf{y}_j = 0\} - 1 & \text{otherwise.} \end{cases}$$

The next result, presented at [1], give us the convergence of the sequence $\{\mathbf{u}_n\}_{n\geq 0}$ to the solution $A^{-1}\mathbf{b}$ of the linear system.

Theorem 5. *Let $\mathbf{b} \in \mathbb{R}^{N_1\cdots N_d}$ and $A \in \mathbb{R}^{N_1\cdots N_d \times N_1\cdots N_d}$ be an invertible matrix. Then, by using the iterative scheme (17)-(18), we obtain that the sequence $\{\|\mathbf{r}_n\|_2\}_{n=0}^{\mathrm{rank}_\otimes(A^{-1}\mathbf{b})}$, is strictly decreasing and*

$$A^{-1}\mathbf{b} = \lim_{n\to\infty} \mathbf{u}_n = \sum_{j=0}^{\mathrm{rank}_\otimes(A^{-1}\mathbf{b})} \mathbf{y}_j. \tag{19}$$

Note that the updates in the previous scheme works under the assumption that in the line 5 of Algorithm 4 we have a way to obtain

$$\mathbf{y} \in \underset{\mathbf{x}\in\mathscr{R}_1}{\arg\min} \|\mathbf{r}_i - A\mathbf{x}\|_2^2. \tag{20}$$

(equation (18)). To compute $\mathbf{y}$, we can use an Alternating Least Squares (ALS, in short) approach, (see [1, 4]).

The idea below the ALS strategy to solve (20) is the following: for each $1 \leq k \leq d$ we proceed as follows. Assume that the values $\mathbf{x}_1,\ldots,\mathbf{x}_{k-1},\mathbf{x}_{k+1},\ldots,\mathbf{x}_d$ are given. Then, we look for the unknown $\mathbf{x}_k$, satisfying,

$$\mathbf{x}_k \in \underset{\mathbf{z}_k\in\mathbb{R}^{N_k\times N_k}}{\arg\min} \|\mathbf{b} - A(\mathbf{x}_1 \otimes \cdots \otimes \mathbf{x}_{k-1} \otimes \mathbf{z}_k \otimes \mathbf{x}_{k+1} \otimes \cdots \otimes \mathbf{x}_d)\|_2,$$

where we can write

$$A(\mathbf{x}_1 \otimes \cdots \otimes \mathbf{x}_{k-1} \otimes \mathbf{z}_k \otimes \mathbf{x}_{k+1} \otimes \cdots \otimes \mathbf{x}_d) = A(\mathbf{x}_1 \otimes \cdots \otimes \mathbf{x}_{k-1} \otimes \mathrm{id}_{N_k} \otimes \mathbf{x}_{k+1} \otimes \cdots \otimes \mathbf{x}_d)\mathbf{z}_k.$$

Algorithm 4 Greedy Rank-One Update Algorithm

procedure GROU($\mathbf{f}, A, \varepsilon, \texttt{tol}, \texttt{rank\_max}$)
  $\mathbf{r}_0 = \mathbf{f}$
  $\mathbf{u} = \mathbf{0}$
  for $i = 0, 1, 2, \ldots, \texttt{rank\_max}$ **do**
    $\mathbf{y} =$ **procedure** $(\min_{\mathbf{x}\in\mathscr{R}_1} \|\mathbf{r}_i - A\mathbf{x}\|_2^2)$
    $\mathbf{r}_{i+1} = \mathbf{r}_i - A\mathbf{y}$
    $\mathbf{u} \leftarrow \mathbf{u} + \mathbf{y}$
    if $\|\mathbf{r}_{i+1}\|_2 < \varepsilon$ or $|\|\mathbf{r}_{i+1}\|_2 - \|\mathbf{r}_i\|_2| < \texttt{tol}$ **then goto** 13
    end if
  end for
  return u and $\|\mathbf{r}_{\texttt{rank\_max}}\|_2$.
  break
  return u and $\|\mathbf{r}_{i+1}\|_2$
end procedure

In consequence, by using a Least Squares approach [4], we can obtain $\mathbf{x}_k$ by solving the following $N_k \times N_k$-dimensional linear system:

$$Z_k \mathbf{z}_k = \mathbf{b}_k \tag{21}$$

where

$$Z_k := (\mathbf{x}_1^T \otimes \cdots \otimes \mathbf{x}_{k-1}^T \otimes \mathrm{id}_{N_k} \otimes \mathbf{x}_{k+1}^T \otimes \cdots \otimes \mathbf{x}_d^T) A^T A (\mathbf{x}_1 \otimes \cdots \otimes \mathbf{x}_{k-1} \otimes \mathrm{id}_k \otimes \mathbf{x}_{k+1} \otimes \cdots \otimes \mathbf{x}_d)$$

and

$$\mathbf{b}_k := (\mathbf{x}_1^T \otimes \cdots \otimes \mathbf{x}_{k-1}^T \otimes \mathrm{id}_{N_k} \otimes \mathbf{x}_{k+1}^T \otimes \cdots \otimes \mathbf{x}_d^T) A^T \mathbf{b}.$$

Here id_{N_k} denotes the identity matrix of size $N_k \times N_k$. Clearly,

$$\|\mathbf{b} - A(\mathbf{x}_1 \otimes \cdots \otimes \mathbf{x}_{k-1} \otimes \mathbf{z}_k \otimes \mathbf{x}_{k+1} \otimes \cdots \otimes \mathbf{x}_d)\|_2 \leq \|\mathbf{b} - A(\mathbf{x}_1 \otimes \cdots \otimes \mathbf{x}_{k-1} \otimes \mathbf{x}_k \otimes \mathbf{x}_{k+1} \otimes \cdots \otimes \mathbf{x}_d)\|_2$$

holds for all $\mathbf{z}_k \in \mathbb{R}^{N_k \times N_k}$. However, it is well-known (see Section 4 in [4]) that the performance of the ALS strategy can be improved (see Algorithm 5 below) when the shape of the matrix $A^T A \in \mathbb{R}^{N\times N}$, with $N = N_1 \ldots N_d$, can be written in the form

$$A^T A = \sum_{i=1}^{r} \bigotimes_{j=1}^{d} A_j^{(i)} \tag{22}$$

where $\bigotimes_{j=1}^{d} A_j^{(i)} = A_1^{(i)} \otimes \cdots \otimes A_d^{(i)}$, here $A_j^{(i)} \in \mathbb{R}^{N_j \times N_j}$ for $1 \leq j \leq d$ and $1 \leq i \leq r$. In particular, when the matrix A is given by

$$A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[N_i]} \doteq \sum_{i=1}^{d} \mathrm{id}_{N_1} \otimes \cdots \otimes \mathrm{id}_{N_{i-1}} \otimes A_i \otimes \mathrm{id}_{N_{i+1}} \otimes \cdots \otimes \mathrm{id}_{N_d}, \tag{23}$$

where $A_i \in \mathbb{R}^{N_i \times N_i}$ for $1 \leq i \leq d$, and id_{N_j} is the identity matrix of size $N_j \times N_j$, then the matrix $A^T A$ can be easily written in the form (22). These matrices has been introduced in [2] as Laplacian-like matrices, since they can be easily related to the classical Laplacian operator [8, 9]. The next section will be devoted to the study of this class of matrices.

Algorithm 5 An Alternated Least Squares Algorithm for matrices in the form (22) [4, Algorithm 2]

Given $A^T A = \sum_{i=1}^{r} \bigotimes_{j=1}^{d} A_j^{(i)} \in \mathbb{R}^{N \times N}$ and $\mathbf{b} \in \mathbb{R}^N$.
Initialize $\mathbf{x}_i^{(0)} \in \mathbb{R}^{N_i}$ for $i = 1, 2 \ldots, d$.
Introduce $\varepsilon > 0$ and `itermax`, `iter` = 1.
while `distance` $> \varepsilon$ and `iter` $<$ `itermax` **do**
    for $k = 1, 2, \ldots, d$ **do**
        $\mathbf{x}_k^{(1)} = \mathbf{x}_k^{(0)}$
        for $i = 1, 2, \ldots, r$ **do**
            $\alpha_k^{(i)} = \left(\prod_{j=1}^{k-1} (\mathbf{x}_j^{(0)})^T A_j^{(i)} \mathbf{x}_j^{(0)} \right) \left(\prod_{j=k+1}^{d} (\mathbf{x}_j^{(1)})^T A_j^{(i)} \mathbf{x}_j^{(1)} \right)$
        end for
        $\mathbf{x}_k^{(0)}$ solves $\left(\sum_{i=1}^{r} \alpha_k^{(i)} A_k^{(i)} \right) \mathbf{x}_k = (\mathbf{x}_1^{(0)} \otimes \cdots \otimes \mathbf{x}_{k-1}^{(0)} \otimes \mathrm{id}_{N_k} \otimes \mathbf{x}_k^{(0)} \otimes \cdots \otimes \mathbf{x}_d^{(0)})^T \mathbf{b}$
    end for
    `iter` = `iter` + 1.
    `distance` $= \max_{1 \leq i \leq d} \|\mathbf{x}_i^{(0)} - \mathbf{x}_i^{(1)}\|_2$.
end while

5.3 On the best Laplacian matrix approximation

As we said in the Introduction, the Proper Orthogonal Decomposition, is a popular numerical strategy in the engineering to solve high-dimensional problems. It is based on the GROU algorithm (17)–(18) and it can be considered as a tensor-based decomposition algorithm.

There is a particular type of matrices to solve high-dimensional linear systems for which these methods work particularly well, those that satisfy the property (22). To this end we introduce the following definition.

Definition 3. *Given a matrix $A \in \mathbb{R}^{N \times N}$, where $N = N_1 \cdots N_d$, we say that A is a Laplacian-like matrix if there exist matrices $A_i \in \mathbb{R}^{N_i \times N_i}$ for $1 \leq i \leq d$ be such that*

$$A = \sum_{i=1}^{d} A_i \otimes \mathrm{id}_{[N_i]} \doteq \sum_{i=1}^{d} \mathrm{id}_{N_1} \otimes \cdots \otimes \mathrm{id}_{N_{i-1}} \otimes A_i \otimes \mathrm{id}_{N_{i+1}} \otimes \cdots \otimes \mathrm{id}_{N_d}, \tag{24}$$

where id_{N_j} is the identity matrix of size $N_j \times N_j$.

It is not difficult to see that the set of Laplacian-like matrices is a linear subspace $\mathbb{R}^{N\times N}$ of matrices satisfying the property (22). From now on, we will denote by $\mathscr{L}\left(\mathbb{R}^{N\times N}\right)$ the subspace of Laplacian-like matrices in $\mathbb{R}^{N\times N}$ for a fixed decomposition of $N = N_1\cdots N_d$.

Now, given a matrix $A\in\mathbb{R}^{N\times N}$, our goal is to solve the following optimization problem:

$$\min_{L\in\mathscr{L}\left(\mathbb{R}^{N\times N}\right)} \|A-L\|_2. \tag{25}$$

Clearly, if we denote by $\Pi_{\mathscr{L}\left(\mathbb{R}^{N\times N}\right)}$ the orthogonal projection onto the linear subspace $\mathscr{L}\left(\mathbb{R}^{N\times N}\right)$ then $L_A := \Pi_{\mathscr{L}\left(\mathbb{R}^{N\times N}\right)}(A)$ is the solution of (25). Observe that $\|A-L_A\|_2 = 0$, if and only if $A\in\mathscr{L}\left(\mathbb{R}^{N\times N}\right)$.

Since, we are interested in trying to achieve a structure similar to (24), to study the matrices of large-dimensional problems. We search an algorithm that allows to construct, for a given matrix A, its Laplacian-like best approximation L_A.

To do this, we will use the following theorem which describes a particular decomposition of the space of matrices $\mathbb{R}^{N\times N}$. Observe that the linear subspace $\operatorname{span}\{\mathrm{id}_N\}$ in $\mathbb{R}^{N\times N}$ has as orthogonal space the null trace matrices:

$$\operatorname{span}\{\mathrm{id}_n\}^{\perp} = \{A\in\mathbb{R}^{n\times n} : \operatorname{tr}(A) = 0\},$$

with respect the inner product $\langle A,B\rangle_{\mathbb{R}^{N\times N}} = \operatorname{tr}(A^TB)$.

Theorem 6. *Consider $\left(\mathbb{R}^{N\times N}, \|\cdot\|_2\right)$ as a Hilbert space where $N = N_1\cdots N_d$. Then there exists a decomposition*

$$\mathbb{R}^{N\times N} = \operatorname{span}\{\mathrm{id}_N\}\oplus\mathfrak{h}_N = \mathscr{L}\left(\mathbb{R}^{N\times N}\right)\oplus\mathscr{L}\left(\mathbb{R}^{N\times N}\right)^{\perp},$$

where $\mathfrak{h}_N = \operatorname{span}\{\mathrm{id}_N\}^{\perp}$ is the orthogonal complement of the linear subspace generated by the identity matrix. Moreover,

$$\mathscr{L}\left(\mathbb{R}^{N\times N}\right) = \operatorname{span}\{\mathrm{id}_N\}\oplus\Delta, \tag{26}$$

where $\Delta = \mathfrak{h}_N\cap\mathscr{L}(\mathbb{R}^{N\times N})$. Furthermore, $\mathscr{L}(\mathbb{R}^{N\times N})^{\perp}$ is a subspace of $\mathfrak{h}_N$ and

$$\Delta = \bigoplus_{i=1}^{d} \operatorname{span}\{\mathrm{id}_{N_1}\}\otimes\cdots\otimes\operatorname{span}\{\mathrm{id}_{N_{i-1}}\}\otimes\operatorname{span}\{\mathrm{id}_{N_i}\}^{\perp}\otimes\operatorname{span}\{\mathrm{id}_{N_{i+1}}\}\otimes\cdots\otimes\operatorname{span}\{\mathrm{id}_{N_d}\}.$$

Proof. It follows from Lemma 3.1, Theorem 3.1 and Theorem 3,2 in [2]. □

The above theorem allows us to compute the projection of matrix A onto $\mathscr{L}(\mathbb{R}^{N\times N})$ as follows. Denote by Π_i the orthogonal projection of $\mathbb{R}^{N\times N}$ onto the linear subspace

$$\operatorname{span}\{\mathrm{id}_{N_1}\}\otimes\cdots\otimes\operatorname{span}\{\mathrm{id}_{N_{i-1}}\}\otimes\operatorname{span}\{\mathrm{id}_{N_i}\}^{\perp}\otimes\operatorname{span}\{\mathrm{id}_{N_{i+1}}\}\otimes\cdots\otimes\operatorname{span}\{\mathrm{id}_{N_d}\}$$

for $1 \leq i \leq d$. Thus, $\sum_{i=1}^{k} \Pi_i$ is the orthogonal projection of $\mathbb{R}^{N\times N}$ onto the linear subspace Δ. In consequence, by using (26), we have

$$\frac{\operatorname{tr}(A)}{N}\mathrm{id}_N + \sum_{i=1}^{d} \Pi_i(A) = \underset{L\in\mathscr{L}(\mathbb{R}^{N\times N})}{\arg\min} \|A-L\|_2. \tag{27}$$

If we analyze a little more (27), we observe that the second term on the left, is of the form

$$\sum_{i=1}^{d} \Pi_i(A) = \sum_{i=1}^{d} \mathrm{id}_{N_1} \otimes \cdots \otimes \mathrm{id}_{N_{i-1}} \otimes X_i \otimes \mathrm{id}_{N_{i+1}} \otimes \cdots \otimes \mathrm{id}_{N_d},$$

and it has only $(N_1^2 + \cdots + N_d^2 - d)$-degrees of freedom (recall that $\dim \operatorname{span}\{\mathrm{id}_{N_i}\}^{\perp} = N_i^2 - 1$). In addition, due to the tensor structure of the products, the unknowns x_l of X_k are distributed in the form of a block, so that we can calculate which will be the entries of the matrix A that we can approximate. Therefore, to obtain the value of the different x_l we only need to calculate which is the value that best approximates the entries (i, j) of the original matrix that are in the same position as x_l.

In our next result, we will see how to carry out this procedure. To do this, we make the following observation. Given a matrix $A = (a_{i,j}) \in \mathbb{R}^{KL\times KL}$ for some integers $K, L > 1$, we can write A as a matrix block

$$A = \begin{pmatrix} A_{1,1}^{(K,L)} & A_{1,2}^{(K,L)} & \cdots & A_{1,L}^{(K,L)} \\ A_{2,1}^{(K,L)} & A_{2,2}^{(K,L)} & \cdots & A_{2,L}^{(K,L)} \\ \vdots & \vdots & \ddots & \vdots \\ A_{L,1}^{(K,L)} & A_{L,2}^{(K,L)} & \cdots & A_{L,L}^{(K,L)} \end{pmatrix} \tag{28}$$

where the block $A_{i,j}^{(K,L)} \in \mathbb{R}^{K\times K}$ for $1 \leq i, j \leq L$ is given by

$$A_{i,j}^{(K,L)} = \begin{pmatrix} a_{(i-1)K+1,(j-1)K+1} & \cdots & a_{(i-1)K+1,jK} \\ \vdots & \ddots & \vdots \\ a_{iK,(j-1)K+1} & \cdots & a_{iK,jK} \end{pmatrix}.$$

Moreover,

$$\|A\|_{\mathbb{R}^{KL\times KL}}^2 = \sum_{i=1}^{KL}\sum_{j=1}^{KL} a_{i,j}^2 = \sum_{r=1}^{L}\sum_{s=1}^{L} \|A_{r,s}^{(K,L)}\|_{\mathbb{R}^{K\times K}}^2.$$

Observe that K and L can easily interchanged. To simplify notation, from now one given $N = N_1 N_2 \cdots N_d$ we denote by $N_{[k]} = N_1 \cdots N_{k-1} N_{k+1} \cdots N_d$ for each $1 \leq k \leq d$.

Theorem 7. *Let $A \in \mathbb{R}^{N\times N}$ with $N = N_1 \cdots N_d$. For each fixed $1 \leq k \leq d$ consider the linear function $P_k : \mathbb{R}^{N_k\times N_k} \longrightarrow \mathbb{R}^{N\times N}$ given by*

$$P_k(X_k) := \mathrm{id}_{N_1} \otimes \cdots \otimes \mathrm{id}_{N_{k-1}} \otimes X_k \otimes \mathrm{id}_{N_{k+1}} \otimes \cdots \otimes \mathrm{id}_{N_d}.$$

Then, the solution of the minimization problem

$$\min_{X_k \in \mathbb{R}^{N_k \times N_k}} \|A - P_k(X_k)\|_2 \tag{29}$$

is given by

$$(X_k)_{i,j} = \begin{cases} \dfrac{1}{N_{[1]}} \displaystyle\sum_{n=1}^{N_{[1]}} a_{(i-1)N_{[1]}+n,(j-1)N_{[1]}+n} & \text{if} \quad k=1, \\ \dfrac{1}{N_{[k]}} \displaystyle\sum_{m=1}^{N_{k+1}\cdots N_d} \left(\sum_{n=1}^{N_1\cdots N_{k-1}} A_{n,n}^{(N_k\cdots N_d, N_1\cdots N_{k-1})} \right)_{(i-1)N_{k+1}\cdots N_d+m,(j-1)N_{k+1}\cdots N_d+m} & \text{if} \quad 1<k<d, \\ \dfrac{1}{N_{[d]}} \left(\displaystyle\sum_{n=1}^{N_{[d]}} A_{n,n}^{(N_d,N_{[d]})} \right)_{i,j} & \text{if} \quad k=d. \end{cases}$$

Proof. First, let us observe that $\mathrm{id}_{N_1} \otimes \cdots \otimes \mathrm{id}_{N_k} = \mathrm{id}_{N_1 \cdots N_k}$, so, we can find three different situations in the calculation of the projections:

1. $P_1(A) = X_1 \otimes \mathrm{id}_{N_{[1]}}$; in this case,

$$P_1(X_1) = \begin{pmatrix} (X_1)_{1,1}\mathrm{id}_{N_{[1]}} & (X_1)_{1,2}\mathrm{id}_{N_{[1]}} & \dots & (X_1)_{1,N_1}\mathrm{id}_{N_{[1]}} \\ (X_1)_{2,1}\mathrm{id}_{N_{[1]}} & (X_1)_{2,2}\mathrm{id}_{N_{[1]}} & \dots & (X_1)_{2,N_1}\mathrm{id}_{N_{[1]}} \\ \vdots & \vdots & \ddots & \vdots \\ (X_1)_{N_1,1}\mathrm{id}_{N_{[1]}} & (X_1)_{N_1,2}\mathrm{id}_{N_{[1]}} & \dots & (X_1)_{N_1,N_1}\mathrm{id}_{N_{[1]}} \end{pmatrix} \in \mathbb{R}^{N_{[1]}N_1 \times N_{[1]}N_1}.$$

2. $P_d(X_d) = \mathrm{id}_{N_{[d]}} \otimes X_d$; in this case,

$$P_d(X_d) = \begin{pmatrix} X_d & O_d & \cdots & O_d \\ O_d & X_d & \cdots & O_d \\ \vdots & \vdots & \ddots & \vdots \\ O_d & O_d & \cdots & X_d \end{pmatrix} \in \mathbb{R}^{N_d N_{[d]} \times N_d N_{[d]}},$$

   where O_d denotes the zero matrix in $\mathbb{R}^{N_d \times N_d}$.

3. $P_i(X_i) = \mathrm{id}_{N_1 \cdots N_{i-1}} \otimes X_i \otimes \mathrm{id}_{N_{i+1}\cdots N_d}$, for $i = 2, \dots, d-1$; in this case for a fixed $2 \leq$

$i \le d-1$, we write $N_\ell = N_1 \cdots N_{i-1}$, and $N_r = N_{i+1} \cdots N_d$. Thus,

$$
\begin{aligned}
P_i(X_i) &= \mathrm{id}_{N_\ell} \otimes X_i \otimes \mathrm{id}_{N_r} \\
&= \mathrm{id}_{N_\ell} \otimes \begin{pmatrix} (X_i)_{1,1}\mathrm{id}_{N_r} & (X_i)_{1,2}\mathrm{id}_{N_r} & \dots & (X_i)_{1,N_1}\mathrm{id}_{N_r} \\ (X_i)_{2,1}\mathrm{id}_{N_r} & (X_i)_{2,2}\mathrm{id}_{N_r} & \dots & (X_i)_{2,N_1}\mathrm{id}_{N_r} \\ \vdots & \vdots & \ddots & \vdots \\ (X_i)_{N_1,1}\mathrm{id}_{N_r} & (X_i)_{N_1,2}\mathrm{id}_{N_r} & \dots & (X_i)_{N_1,N_1}\mathrm{id}_{N_r} \end{pmatrix} \\
&= \begin{pmatrix} X_i \otimes \mathrm{id}_{N_r} & O_i \otimes \mathrm{id}_{N_r} & \cdots & O_i \otimes \mathrm{id}_{N_r} \\ O_i \otimes \mathrm{id}_{N_r} & X_i \otimes \mathrm{id}_{N_r} & \cdots & O_i \otimes \mathrm{id}_{N_r} \\ \vdots & \vdots & \ddots & \vdots \\ O_i \otimes \mathrm{id}_{N_r} & O_i \otimes \mathrm{id}_{N_r} & \cdots & X_i \otimes \mathrm{id}_{N_r} \end{pmatrix} \in \mathbb{R}^{(N_i N_r)N_\ell \times (N_i N_r)N_\ell}
\end{aligned}
$$

In either case, a difference of the form

$$\min_{X_k \in \mathbb{R}^{N_k \times N_k}} \| A - P_k(A) \|_2$$

must be minimized. To this end, we will consider on each case A as a block matrix $A \in \mathbb{R}^{KL \times KL}$ in the form (28).

Case 1: For $P_1(X_1)$ we take $K = N_{[1]}$, $L = N_1$, and hence

$$
A - P_1(X_1) = \begin{pmatrix} A^{(K,L)}_{1,1} - (X_1)_{1,1}\mathrm{id}_{N_{[1]}} & A^{(K,L)}_{1,2} - (X_1)_{1,2}\mathrm{id}_{N_{[1]}} & \dots & A^{(K,L)}_{1,N_1} - (X_1)_{1,N_1}\mathrm{id}_{N_{[1]}} \\ A^{(K,L)}_{2,1} - (X_1)_{2,1}\mathrm{id}_{N_{[1]}} & A^{(K,L)}_{2,2} - (X_1)_{2,2}\mathrm{id}_{N_{[1]}} & \dots & A^{(K,L)}_{2,N_1} - (X_1)_{2,N_1}\mathrm{id}_{N_{[1]}} \\ \vdots & \vdots & \ddots & \vdots \\ A^{(K,L)}_{N_1,1} - (X_1)_{N_1,1}\mathrm{id}_{N_{[1]}} & A^{(K,L)}_{N_1,2} - (X_1)_{N_1,2}\mathrm{id}_{N_{[1]}} & \dots & A^{(K,L)}_{N_1,N_1} - (X_1)_{N_1,N_1}\mathrm{id}_{N_{[1]}} \end{pmatrix}.
$$

In this situation we have

$$\| A - P_1(X_1) \|^2_{\mathbb{R}^{N \times N}} = \sum_{i=1}^{N_1} \sum_{j=1}^{N_1} \| A^{(K,L)}_{i,j} - (X_1)_{i,j}\mathrm{id}_{N_{[1]}} \|^2_{\mathbb{R}^{N_{[1]} \times N_{[1]}}},$$

hence we wish for each $1 \le i, j \le N_1$ to find

$$(X_1)_{i,j} = x \in \arg\min_{x \in \mathbb{R}} \| A^{(K,L)}_{i,j} - x\,\mathrm{id}_{N_{[1]}} \|^2_{\mathbb{R}^{N_{[1]} \times N_{[1]}}} = \arg\min_{x \in \mathbb{R}} \sum_{n=1}^{N_{[1]}} (a_{(i-1)N_{[1]}+n,(j-1)N_{[1]}+n} - x)^2.$$

Thus, it is not difficult to see that

$$(X_1)_{i,j} = \frac{1}{N_{[1]}} \sum_{n=1}^{N_{[1]}} a_{(i-1)N_{[1]}+n,(j-1)N_{[1]}+n},$$

for $1 \le i, j \le N_1$.

Case 2: For $P_d(X_d)$ we take $K = N_d$, $L = N_{[d]}$, and hence

$$
A - P_d(X_d) = \begin{pmatrix} A^{(K,L)}_{1,1} - X_d & A^{(K,L)}_{1,2} - O_d & \cdots & A^{(K,L)}_{1,N_{[d]}} - O_d \\ A^{(K,L)}_{2,1} - O_d & A^{(K,L)}_{2,2} - X_d & \cdots & A^{(K,L)}_{2,N_{[d]}} - O_d \\ \vdots & \vdots & \ddots & \vdots \\ A^{(K,L)}_{N_{[d]},1} - O_d & A^{(K,L)}_{N_{[d]},2} - O_d & \cdots & A^{(K,L)}_{N_{[d]},N_{[d]}} - X_d \end{pmatrix}
$$

Now, we have

$$\|A - P_d(X_d)\|^2_{\mathbb{R}^{N\times N}} = \sum_{i=1}^{N_{[d]}} \|A^{(K,L)}_{i,i} - X_d\|^2_{\mathbb{R}^{N_d\times N_d}} + \sum_{i=1,j=1,i\neq j}^{N_{[d]}} \|A^{(K,L)}_{i,i}\|^2_{\mathbb{R}^{N_d\times N_d}}.$$

Thus, $X_d \in \mathbb{R}^{N_d\times N_d}$ minimizes $\|A - P_d(X_d)\|^2_{\mathbb{R}^{N\times N}}$ if and only if

$$X_d \in \arg\min_{X\in\mathbb{R}^{N_d\times N_d}} \sum_{i=1}^{N_{[d]}} \|A^{(K,L)}_{i,i} - X\|^2_{\mathbb{R}^{N_d\times N_d}}.$$

In consequence,

$$X_d = \frac{1}{N_{[d]}} \sum_{i=1}^{N_{[d]}} A^{(K,L)}_{i,i}.$$

Case 3: For $P_i(X_i)$ we take $K = N_i N_r$, $L = N_\ell$ and hence

$$A - P_i(X_i) = \begin{pmatrix} A^{(K,L)}_{1,1} - X_i\otimes \mathrm{id}_{N_r} & A^{(K,L)}_{1,2} - O_i\otimes \mathrm{id}_{N_r} & \cdots & A^{(K,L)}_{1,N_\ell} - O_i\otimes \mathrm{id}_{N_r} \\ A^{(K,L)}_{2,1} - O_i\otimes \mathrm{id}_{N_r} & A^{(K,L)}_{2,2} - X_i\otimes \mathrm{id}_{N_r} & \cdots & A^{(K,L)}_{1,N_\ell} - O_i\otimes \mathrm{id}_{N_r} \\ \vdots & \vdots & \ddots & \vdots \\ A^{(K,L)}_{N_\ell,1} - O_i\otimes \mathrm{id}_{N_r} & A^{(K,L)}_{N_\ell,2} - O_i\otimes \mathrm{id}_{N_r} & \cdots & A^{(K,L)}_{N_\ell,N_\ell} - X_i\otimes \mathrm{id}_{N_r} \end{pmatrix}$$

In this case

$$\|A - P_i(X_i)\|^2_{\mathbb{R}^{N\times N}} = \sum_{n=1}^{N_\ell} \|A^{(K,L)}_{n,n} - X_i\otimes \mathrm{id}_{N_r}\|^2_{\mathbb{R}^{N_iN_r\times N_iN_r}} + \sum_{n=1,j=1,n\neq j}^{N_\ell} \|A^{(K,L)}_{n,j}\|^2_{\mathbb{R}^{N_iN_r\times N_iN_r}},$$

so we need to solve the following problem:

$$\min_{X\in\mathbb{R}^{N_i\times N_i}} \sum_{n=1}^{N_\ell} \|A^{(K,L)}_{n,n} - X\otimes \mathrm{id}_{N_r}\|^2_{\mathbb{R}^{N_iN_r\times N_iN_r}}. \tag{30}$$

Since $X\otimes \mathrm{id}_{N_r} \in \mathbb{R}^{N_i\times N_i}\otimes \mathrm{span}\{\mathrm{id}_{N_r}\}$ we can write (30) as

$$\min_{Z\in\mathbb{R}^{N_i\times N_i}\otimes \mathrm{span}\{\mathrm{id}_{N_r}\}} \sum_{n=1}^{N_\ell} \|A^{(K,L)}_{n,n} - Z\|^2_{\mathbb{R}^{N_iN_r\times N_iN_r}}. \tag{31}$$

Observe that

$$A^* = (a^*_{u,v}) = \frac{1}{N_\ell}\sum_{n=1}^{N_\ell} A^{(K,L)}_{n,n} = \arg\min_{U\in\mathbb{R}^{N_iN_r\times N_iN_r}} \sum_{n=1}^{N_\ell} \|A^{(K,L)}_{n,n} - U\|^2_{\mathbb{R}^{N_iN_r\times N_iN_r}}.$$

To simplify notation, we write $\mathscr{U} := \mathbb{R}^{N_i\times N_i}\otimes \mathrm{span}\{\mathrm{id}_{N_r}\}$. Then we have the following orthogonal decomposition $\mathbb{R}^{N_iN_r\times N_iN_r} = \mathscr{U}\oplus\mathscr{U}^\perp$. Denote by $\Pi_{\mathscr{U}}$ the orthogonal projection onto the linear subspace $\mathscr{U}$. Then for each $Z\in\mathscr{U}$ we have

$$\begin{aligned} \|A^{(K,L)}_{n,n} - Z\|^2 &= \|(\mathrm{id} - \Pi_{\mathscr{U}})(A^{(K,L)}_{n,n}) + \Pi_{\mathscr{U}}(A^{(K,L)}_{n,n}) - Z\|^2 \\ &= \|(\mathrm{id} - \Pi_{\mathscr{U}})(A^{(K,L)}_{n,n})\|^2 + \|\Pi_{\mathscr{U}}(A^{(K,L)}_{n,n}) - Z\|^2, \end{aligned}$$

because $(\mathrm{id}-\Pi_{\mathscr{U}})(A^{(K,L)}_{n,n}) \in \mathscr{U}^{\perp}$ and $\Pi_{\mathscr{U}}(A^{(K,L)}_{n,n}) - Z \in \mathscr{U}$. In consequence, solve (31) is equivalent to solve the following optimization problem

$$\min_{Z\in\mathscr{U}} \sum_{n=1}^{N_\ell} \|\Pi_{\mathscr{U}}(A^{(K,L)}_{n,n}) - Z\|^2_{\mathbb{R}^{N_i N_r \times N_i N_r}}. \tag{32}$$

Thus,

$$Z^* = \frac{1}{N_\ell}\sum_{n=1}^{N_\ell} \Pi_{\mathscr{U}}(A^{(K,L)}_{n,n}) = \arg\min_{Z\in\mathscr{U}} \sum_{n=1}^{N_\ell} \|\Pi_{\mathscr{U}}(A^{(K,L)}_{n,n}) - Z\|^2_{\mathbb{R}^{N_i N_r \times N_i N_r}},$$

that is, $Z^* = \Pi_{\mathscr{U}}(A^*)$ and hence

$$Z^* = \arg\min_{Z\in\mathscr{U}} \|A^* - Z\|^2 = X_i \otimes \mathrm{id}_{N_r} = \arg\min_{X\in\mathbb{R}^{N_i\times N_i}} \|A^* - X\otimes \mathrm{id}_{N_r}\|^2_{\mathbb{R}^{N_i N_r \times N_i N_r}}.$$

Proceeding in a similar way as in Case 1, we obtain

$$(X_i)_{u,v} = \frac{1}{N_r}\sum_{m=1}^{N_r} a^*_{(u-1)N_r+m,(v-1)N_r+m} = \frac{1}{N_r}\frac{1}{N_l}\sum_{m=1}^{N_r}\left(\sum_{n=1}^{N_l} A^{(K,L)}_{n,n}\right)_{(u-1)N_r+m,(v-1)N_r+m},$$

for $1 \le u, v \le N_i$. This concludes the proof of the theorem. □

To conclude we obtain the following useful corollary.

Corollary 3. *Let $A \in \mathbb{R}^{N\times N}$ with $N = N_1 \cdots N_d$. For each fixed $1 \le k \le d$ consider the linear function $P_k : \mathbb{R}^{N_k\times N_k} \longrightarrow \mathbb{R}^{N\times N}$ given by*

$$P_k(X_k) := \mathrm{id}_{N_1} \otimes \cdots \otimes \mathrm{id}_{N_{k-1}} \otimes X_k \otimes \mathrm{id}_{N_{k+1}} \otimes \cdots \otimes \mathrm{id}_{N_d}.$$

For each $1 \le k \le d$, let $X_k \in \mathbb{R}^{N_k\times N_k}$ be the solution of the optimization problem (29). Then

$$L_A = \frac{\mathrm{tr}(A)}{N}\mathrm{id}_N + \sum_{k=1}^{d} P_k\left(X_k - \frac{\mathrm{tr}(X_k)}{N_k}\mathrm{id}_{N_k}\right) = \arg\min_{L\in\mathscr{L}(\mathbb{R}^{N\times N})} \|A - L\|_2. \tag{33}$$

Proof. Observe that for $1 \le k \le d$, the matrix X_k satisfies

$$P_k(X_k) = \arg\min_{Z\in\mathfrak{h}^{(k)}} \|A - Z\|_2,$$

where

$$\mathfrak{h}^{(k)} := \mathrm{span}\{\mathrm{id}_{N_1}\} \otimes \cdots \otimes \mathrm{span}\{\mathrm{id}_{N_{k-1}}\} \otimes \mathbb{R}^{N_k\times N_k} \otimes \mathrm{span}\{\mathrm{id}_{N_{k+1}}\} \otimes \cdots \otimes \mathrm{span}\{\mathrm{id}_{N_d}\}.$$

is a linear subspace of $\mathbb{R}^{N\times N}$ linearly isomorphic to $\mathbb{R}^{N_k\times N_K}$. Since $\mathbb{R}^{N_k\times N_k} = \mathrm{span}\{\mathrm{id}_{N_k}\} \oplus \mathrm{span}\{\mathrm{id}_{N_k}\}^{\perp}$, then

$$X_k = \frac{\mathrm{tr}(X_k)}{N_k}\mathrm{id}_{N_k} + \left(X_k - \frac{\mathrm{tr}(X_k)}{N_k}\mathrm{id}_{N_k}\right),$$

and hence

$$P_k(X_k) = P_k\left(\frac{\mathrm{tr}(X_k)}{N_k}\mathrm{id}_{N_k}\right) + P_k\left(X_k - \frac{\mathrm{tr}(X_k)}{N_k}\mathrm{id}_{N_k}\right) = \frac{\mathrm{tr}(X_k)}{N_k}\mathrm{id}_N + P_k\left(X_k - \frac{\mathrm{tr}(X_k)}{N_k}\mathrm{id}_{N_k}\right).$$

We can conclude, that $\Pi_k(A) = P_k\left(X_k - \frac{\mathrm{tr}(X_k)}{N_k}\mathrm{id}_{N_k}\right)$, recall that Π_k is the orthogonal projection of $\mathbb{R}^{N\times N}$ onto the linear subspace

$$\mathrm{span}\{\mathrm{id}_{N_1}\}\otimes\cdots\otimes\mathrm{span}\{\mathrm{id}_{N_{k-1}}\}\otimes\mathrm{span}\{\mathrm{id}_{N_k}\}^{\perp}\otimes\mathrm{span}\{\mathrm{id}_{N_{k+1}}\}\otimes\cdots\otimes\mathrm{span}\{\mathrm{id}_{N_d}\}.$$

From (27) the corollary is proved. □

5.4 The best Laplacian approximation for the discretization of a second order PDEs without mixing derivatives

In this section we consider the general equation of a generic second order PDE without mixing derivatives with homogeneous boundary conditions. More precisely, let

$$\alpha\mathbf{u}_{xx} + \beta\mathbf{u}_{yy} + \gamma\mathbf{u}_x + \delta\mathbf{u}_y + \mu\mathbf{u} = \mathbf{f}, \quad \text{for } (x,y)\in(0,1)\times(0,1) \tag{34}$$

$$\mathbf{u}(x,0) = \mathbf{u}(x,1) = \mathbf{u}(0,y) = \mathbf{u}(1,y) = 0, \quad \text{for all } 0\le x\le 1 \text{ and } 0\le y\le 1. \tag{35}$$

We discretize (34) by the help of the following derivative approximations

$$\mathbf{u}_x(x,y) \approx \frac{\mathbf{u}(x_{i+1},y_j) - \mathbf{u}(x_{i-1},y_j)}{2h}, \quad \mathbf{u}_y(x,y) \approx \frac{\mathbf{u}(x_i,y_{j+1}) - \mathbf{u}(x_i,y_{j-1})}{2k},$$

and

$$\mathbf{u}_{xx}(x,y) \approx \frac{\mathbf{u}(x_{i+1},y_j) - 2\mathbf{u}(x_i,y_j) + \mathbf{u}(x_{i-1},y_j)}{h^2},$$
$$\mathbf{u}_{yy}(x,y) \approx \frac{\mathbf{u}(x_i,y_{j+1}) - 2\mathbf{u}(x_i,y_j) + \mathbf{u}(x_i,y_{j-1})}{k^2},$$

for $i = 1,\ldots,N$, $j = 1,\ldots,M$. From (35) we have $\mathbf{u}(x,y_0) = \mathbf{u}(x,y_{M+1}) = \mathbf{u}(x_0,y) = \mathbf{u}(x_{N+1},y) = 0$ for all $0\le x\le 1$ and $0\le y\le 1$.

Next, in order to obtain a linear system we put $\mathbf{u}_\ell := \mathbf{u}(x_i,y_j)$ and $\mathbf{f}_\ell := \mathbf{f}(x_i,y_j)$ where $\ell := (i-1)M + j$ for $1\le i\le N$ and $1\le j\le M$. In this way, the represented mesh is traversed as shown in Figure 18, and the elements $U = (\mathbf{u}_\ell)_{\ell=1}^{MN}$ and $F = \{\mathbf{f}_\ell\}_{\ell=1}^{MN}$ are column vectors. It allows us to represent (34)-(35) as the linear system $AF = U$, where A is the $MN\times MN$-block matrix

$$A = \begin{pmatrix} T & D_1 & & & \\ D_2 & T & D_1 & & \\ & \ddots & \ddots & \ddots & \\ & & D_2 & T & D_1 \\ & & & D_2 & T \end{pmatrix}, \tag{36}$$

for $T \in \mathbb{R}^{M\times M}$ given by

$$T = \begin{pmatrix} 0 & 2\beta h^2+\delta h^2 k & 0 & \dots & 0 \\ 2\beta h^2-\delta h^2 k & 0 & 2\beta h^2+\delta h^2 k & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 2\beta h^2-\delta h^2 k & 0 \end{pmatrix} + (2\mu h^2 k^2 - 4\alpha k^2 - 4\beta h^2)\mathrm{id}_M$$

and $D_1, D_2 \in \mathbb{R}^{M\times M}$ are the diagonal matrices

$$D_1 = (2\alpha k^2 + \gamma h k^2)\mathrm{id}_M, \quad D_2 = (2\alpha k^2 - \gamma h k^2)\mathrm{id}_M.$$

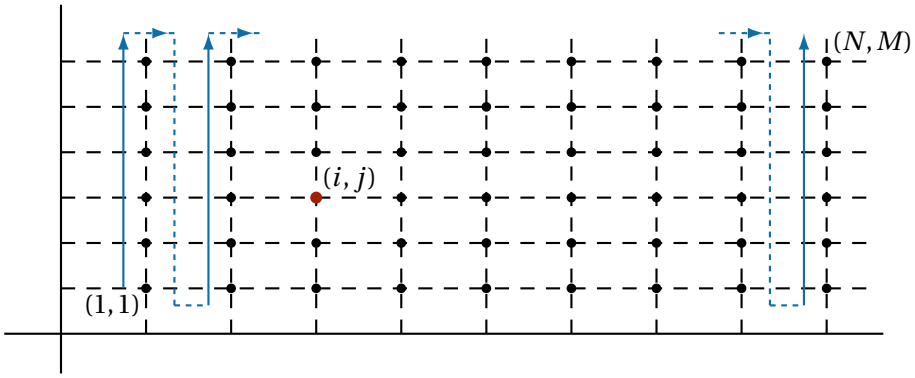

Figure 18: Starting at $(1,1)$ to $(1,M)$; $(2,1),\dots,(2,M)$; and ending at $(N,1),\dots,(N,M)$.

In this case, $\mathrm{tr}(A) = NM(2\mu h^2 k^2 - 4\alpha k^2 - 4\beta h^2)$, so instead of looking for L_A as in (33) we will look for $L_{\hat{A}}$ where

$$\hat{A} = \left(A - \frac{\mathrm{tr}(A)}{NM}\mathrm{id}_{NM}\right),$$

has null trace. Proceeding according Theorem 7 for sizes $N_1 = N$ and $N_2 = M$, we obtain the following decomposition:

$$X_1 = \begin{pmatrix} 0 & 2\alpha k^2+\gamma h k^2 & 0 & \dots & 0 \\ 2\alpha k^2-\gamma h k^2 & 0 & 2\alpha k^2+\gamma h k^2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 2\alpha k^2-\gamma h k^2 & 0 & 2\alpha k^2+\gamma h k^2 \\ 0 & \dots & 0 & 2\alpha k^2-\gamma h k^2 & 0 \end{pmatrix} \in \mathbb{R}^{N\times N},$$

and

$$X_2 = \begin{pmatrix} 0 & 2\beta h^2+\delta h^2 k & 0 & \dots & 0 \\ 2\beta h^2-\delta h^2 k & 0 & 2\beta h^2+\delta h^2 k & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 2\beta h^2-\delta h^2 k & 0 & 2\beta h^2+\delta h^2 k \\ 0 & \dots & 0 & 2\beta h^2-\delta h^2 k & 0 \end{pmatrix} \in \mathbb{R}^{M\times M}.$$

We remark that $\text{tr}(X_1) = \text{tr}(X_2) = 0$. Moreover, the residual of the approximation $L_{\hat{A}}$ of $\hat{A}$ is $\|\hat{A} - L_{\hat{A}}\| = 0$. In consequence, we can write the original matrix A as

$$A = \frac{\text{tr}(A)}{NM}\text{id}_{NM} + X_1 \otimes \text{id}_M + \text{id}_N \otimes X_2.$$

Recall that the first term is

$$\frac{\text{tr}(A)}{NM}\text{id}_{NM} = \left(2\mu h^2 k^2 - 4\alpha k^2 - 4\beta h^2\right) \cdot \text{id}_{NM} = \left(2\mu h^2 k^2 - 4\alpha k^2 - 4\beta h^2\right) \cdot \text{id}_N \otimes \text{id}_M,$$

and hence A can be written as

$$A = Z_1 \otimes \text{id}_M + \text{id}_N \otimes Z_2,$$

where Z_1 is the $N \times N$-matrix

$$\begin{pmatrix} \mu h^2 k^2 - 2\alpha k^2 - 2\beta h^2 & 2\alpha k^2 + \gamma h k^2 & 0 & \dots & 0 \\ 2\alpha k^2 - \gamma h k^2 & \mu h^2 k^2 - 2\alpha k^2 - 2\beta h^2 & 2\alpha k^2 + \gamma h k^2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 2\alpha k^2 - \gamma h k^2 & \mu h^2 k^2 - 2\alpha k^2 - 2\beta h^2 & 2\alpha k^2 + \gamma h k^2 \\ 0 & \dots & 0 & 2\alpha k^2 - \gamma h k^2 & \mu h^2 k^2 - 2\alpha k^2 - 2\beta h^2 \end{pmatrix},$$

and Z_2 the $M \times M$-matrix

$$\begin{pmatrix} \mu h^2 k^2 - 2\alpha k^2 - 2\beta h^2 & 2\beta h^2 + \delta h^2 k & 0 & \dots & 0 \\ 2\beta h^2 - \delta h^2 k & \mu h^2 k^2 - 2\alpha k^2 - 2\beta h^2 & 2\beta h^2 + \delta h^2 k & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 2\beta h^2 - \delta h^2 k & \mu h^2 k^2 - 2\alpha k^2 - 2\beta h^2 & 2\beta h^2 + \delta h^2 k \\ 0 & \dots & 0 & 2\beta h^2 - \delta h^2 k & \mu h^2 k^2 - 2\alpha k^2 - 2\beta h^2 \end{pmatrix}.$$

Now, we can use this representation of A to implement the GROU Algorithm 4 together the ALS strategy given by Algorithm 5 to solve linear system

$$AU = (Z_1 \otimes \text{id}_M + \text{id}_N \otimes Z_2)U = F.$$

This study can be extended to high-dimensional equations, as occurs in [2] with the three-dimensional Poisson equation.

5.5 Numerical examples

Next, we are going to consider some particular equations to analyze their numerical behavior. In all cases, the characteristics of the computer used are: 11th Gen Intel(R) Core(TM) i7-11370H @ 3.30GHz, RAM 16 GB, 64 bit operating system; and a Matlab version R2021b [10].

5.5.1 The Helmholtz equation

Let us consider the particular case of the second order PDE, $\alpha = \beta = 1$, $\mu = c^2$ and $\mathbf{f} = 0$, that is

$$\mathbf{u}_{xx} + \mathbf{u}_{yy} + c^2\mathbf{u} = 0.$$

This is the 2D-Helmholtz equation. To obtain the linear system associated to the discrete problem, we need some boundary conditions, for example

$$\begin{cases} \mathbf{u}(x,0) = \sin(\omega x) + \cos(\omega x) & \text{for} \quad 0 \le x \le L \\ \mathbf{u}(0,y) = \sin(\omega y) + \cos(\omega y) & \text{for} \quad 0 \le y \le T \end{cases}$$

and

$$\begin{cases} \mathbf{u}(x,T) = \sin\left(\omega(x+T)\right) + \cos\left(\omega(x+T)\right) & \text{for} \quad 0 \le x \le L \\ \mathbf{u}(L,y) = \sin\left(\omega(y+L)\right) + \cos\left(\omega(y+L)\right) & \text{for} \quad 0 \le y \le T. \end{cases}$$

This IVP has a closed solution for $\omega = \dfrac{c}{\sqrt{2}}$,

$$\mathbf{u}(x,y) = \sin\left(\omega(x+y)\right) + \cos\left(\omega(x+y)\right).$$

From the above operations, and taking $h = k$ for simplicity, we can write the matrix of the discrete linear system associated to the equation of Helmholtz as

$$A = \begin{pmatrix} 2c^2h^4 - 8h^2 & 2h^2 & 0 & \dots & 0 \\ 2h^2 & 2c^2h^4 - 8h^2 & 2h^2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 2h^2 & 2c^2h^4 - 8h^2 \end{pmatrix} \otimes \mathrm{id}_M + \mathrm{id}_N \otimes \begin{pmatrix} 0 & 2h^2 & 0 & \dots & 0 \\ 2h^2 & 0 & 2h^2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 2h^2 & 0 \end{pmatrix}$$

or, equivalently,

$$A = \begin{pmatrix} c^2h^4 - 4h^2 & 2h^2 & 0 & \dots & 0 \\ 2h^2 & c^2h^4 - 4h^2 & 2h^2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 2h^2 & c^2h^4 - 4h^2 \end{pmatrix} \otimes \mathrm{id}_M + \mathrm{id}_N \otimes \begin{pmatrix} c^2h^4 - 4h^2 & 2h^2 & 0 & \dots & 0 \\ 2h^2 & c^2h^4 - 4h^2 & 2h^2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 2h^2 & c^2h^4 - 4h^2 \end{pmatrix}.$$

If we solve this linear system $A\mathbf{u}_l = \hat{\mathbf{f}}_l$ for the case $c = \sqrt{2}$, $L = T = 1$ and with $N = M$, we obtain the temporary results shown in Figure 19. To carry out this experiment, we have used the following parameters values: for the GROU Algorithm 4: `tol` $= 2.2204e-16$; $\varepsilon = 2.2204e-16$; `rank_max` $= 10$; (an `iter-max` $= 5$ and $\varepsilon = 2.2204e-16$ was used to

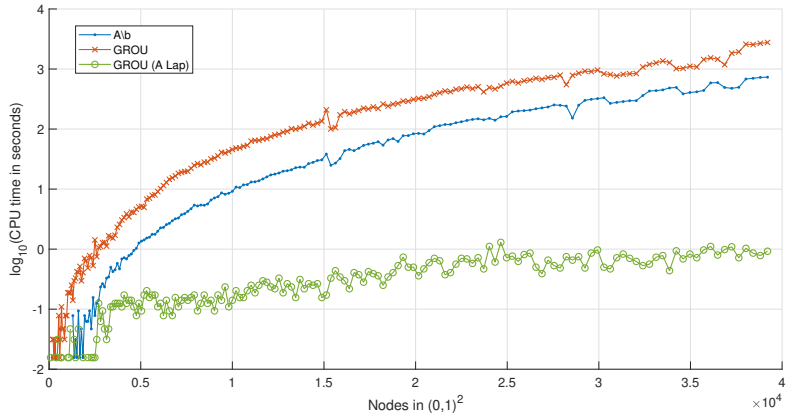

Figure 19: CPU Time, in second, employed to solve the discrete Helmholtz IPV by using the Matlab command $A\backslash b$, the GROU Algorithm 4, and the GROU Algorithm 4 with A written as L_A, obtained from Corollary 26.

perform Algorithm 5); and the number of nodes in $(0,1)^2$ (that is, the number of rows or columns of the matrix A) increase from 10^2 to 200^2.

To measure the goodness of the approximations obtained, we have calculated the *normalized errors*, that is, the value of the difference, in absolute value, of the results obtained and the real solution, between the length of the solution, i.e.

$$\varepsilon = \frac{|\text{exact solution} - \text{approximate solution}|}{N^2}.$$

for the different approximations obtained. The value of these errors is of the order of 10^{-4}, and can be seen in Figure 20.

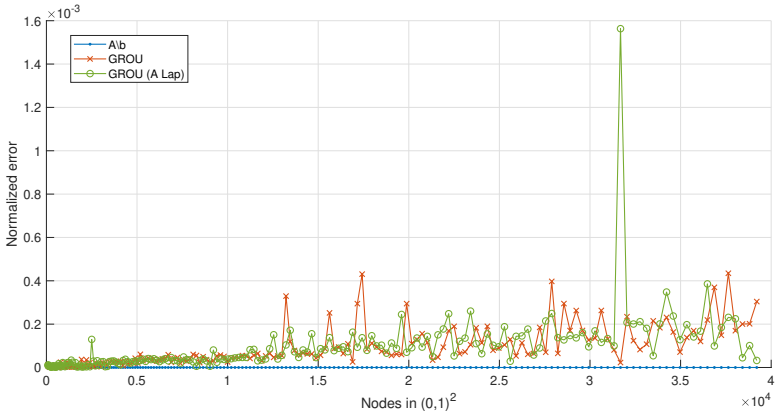

Figure 20: Normalized error between the solution of the discrete Helmholtz IPV and the solutions obtained by using the Matlab command $A\backslash b$, the GROU Algorithm 4, and the GROU Algorithm 4 with A written as L_A, obtained from Corollary 26.

5.5.2 The Swift-Hohenberg equation

Now, let we consider the PDE of order 4

$$\frac{\partial u}{\partial t} = \varepsilon - \left(1 + \frac{\partial^2}{\partial x^2}\right)^2 u \tag{37}$$

with the boundary conditions

$$\begin{cases} u(x,0) = \sin(kx) \\ u(x,T) = \sin(kx)e^T, \end{cases} \quad \text{for} \quad 0 \le x \le L, \tag{38}$$

and

$$u(0,t) = u(L,t) = 0, \quad \text{for} \quad 0 \le t \le T. \tag{39}$$

For $k = \sqrt{1+\sqrt{\varepsilon-1}}$, $L = 2\pi/k$, the IVP (37)-(39) has as a solution

$$u(x,t) = \sin(kx)e^t.$$

If we discretize the (37)-(39) problem as in the previous example with the same step size in both variables, h, we obtain a linear system of the form $A\mathbf{u}_l = \hat{\mathbf{f}}_l$, where A, in Laplacian-Like form, is the matrix

$$A = \left((12 - 8h^2 + (2-2\varepsilon)h^4)\mathrm{id}_N + \begin{pmatrix} 0 & 4h^2-8 & 2 & 0 & \dots & 0 \\ 4h^2-8 & 0 & 4h^2-8 & 2 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 2 & 4h^2-8 & 0 \end{pmatrix} \right) \otimes \mathrm{id}_M$$
$$+ \mathrm{id}_N \otimes \begin{pmatrix} 0 & h^3 & 0 & \dots & 0 \\ -h^3 & 0 & h^3 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -h^3 & 0 \end{pmatrix},$$

and $l = (i-1)M + j$ is the order established for the indices, with $1 \le i \le N$, $1 \le j \le M$.

To perform a numerical experiment, we set $\varepsilon = 2$, $L = T = 2\pi$, and the same number of points in the two variables. At this point, we can solve the linear system associated to the Swift-Hohenberg discrete problem with our tools: the Matlab command $A\backslash b$, the GROU Algorithm 4, and the GROU Algorithm 4 together the ALS Algorithm 5 with A write in Laplacian-like form. In this case we have used the following parameters values in the algorithms: `tol` $= 2.2204e-16$; $\varepsilon = 2.2204e-16$; `rank_max` $= 10$ for the GROU Algorithm 4, with `iter-max` $= 5$ for the ALS step; and the number of nodes in $(0,2\pi)^2$ increase from 10^2 to 200^2. Figure 21 shows the results obtained.

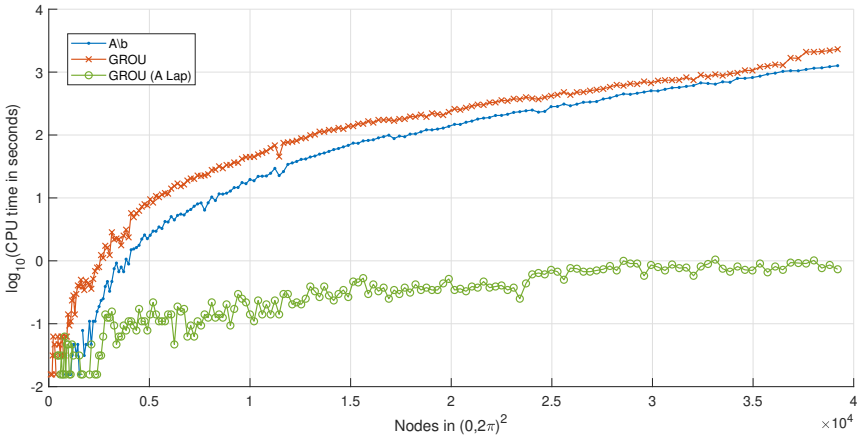

Figure 21: CPU Time, in second, employed to solve the discrete Swift-Hohenberg IPV by using the Matlab command $A\backslash b$, the GROU Algorithm 4, and the GROU Algorithm 4 with A written in Laplacian form.

Again, we calculated the normalized errors to estimate the goodness of the approximations, Figure 22.

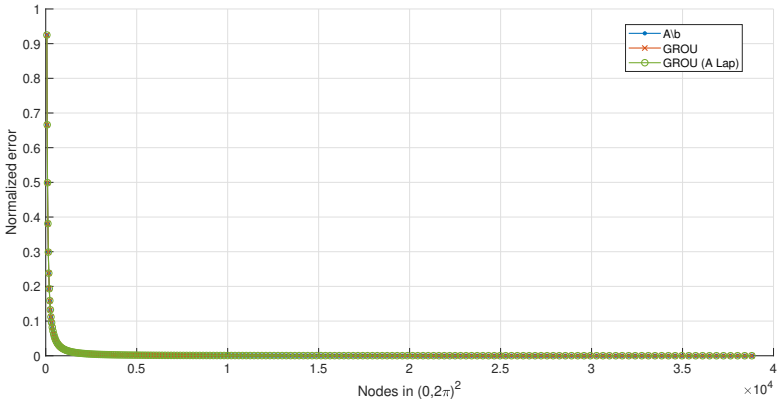

Figure 22: Normalized error between the solution of the discrete Swift-Hohenberg IPV and the solutions obtained by using the Matlab command $A\backslash b$, the GROU Algorithm 4, and the GROU Algorithm 4 with A written in Laplacian form.

5.6 Conclusions

In this work, we have studied the Laplacian Decomposition Algorithm which, given any square matrix, calculates its best Laplacian approximation. Furthermore, in Theorem 7, we have shown how it is implemented optimally.

For us, the greatest interest in this algorithm lies in the computational improvement of combining it with the Greedy Rank-One Updated Algorithm 4 to solve linear systems

from the discretization of a Partial Derivative Equation. Said improvement can be seen in the different numerical examples shown, where we have compared this procedure with the standard resolution of Matlab by means of the instruction $A\backslash b$.

This proposal proposes a new way of dealing with certain large-scale problems, where classical methods prove to be more inefficient.

References

[1] A. Ammar, F. Chinesta, and A. Falcó. On the convergence of a Greedy Rank-One Update algorithm for a class of linear systems. *Arch. Comput. Methods Eng.*, 17(4):473–486, 2010.

[2] J. A. Conejero, A. Falcó, and M. Mora-Jiménez. Structure and Approximation Properties of Laplacian-Like Matrices. *Results in Mathematics*, 78(184), 2023.

[3] V. de Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, Jan. 2008.

[4] A. Falcó, L. Hilario, N. Montés, and M. Mora. Numerical strategies for the galerkin–proper generalized decomposition method. *Mathematical and Computer Modelling*, 57(7-8):1694–1702, Apr. 2013.

[5] A. Falcó and A. Nouy. Proper generalized decomposition for nonlinear convex problems in tensor Banach spaces. *Numer. Math.*, 121:503–530, 2012.

[6] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 10 2022.

[7] I. Georgieva and C. Hofreither. Greedy low-rank approximation in Tucker format of solutions of tensor linear systems. *J. Comput. Appl. Math.*, 358:206–220, 2019.

[8] W. Hackbusch, B. Khoromskij, S. Sauter, and E. Tyrtyshnikov. Use of tensor formats in elliptic eigenvalue problems. *Numer. Lin. Algebra Appl.*, 19:133–151, 2012.

[9] G. Heidel, V. Khoromskaia, B. Khoromskij, and V. Schulz. Tensor product method for fast solution of optimal control problems with fractional multidimensional Laplacian in constraints. *J. Comput. Phys.*, 424:109865, 2021.

[10] MATLAB. *version R2021b*. The MathWorks Inc., Natick, Massachusetts, 2021.

[11] A. Nouy. *Chapter 4: Low-Rank Methods for High-Dimensional Approximation and Model Order Reduction.*, pages 171–226. Society for Industrial and Applied Mathematics, 2017.

[12] C. Quesada, G. Xu, D. González, I. Alfaro, A. Leygue, M. Visonneau, E. Cueto, and F. Chinesta. Un método de descomposición propia generalizada para operadores diferenciales de alto orden. *Rev. Int. Metod. Numer.*, 31(3):188–197, 2015.

[13] J. Swift and P. C. Hohenberg. Hydrodynamic fluctuations at the convective instability. *Phys. Rev. A*, 15:319–328, Jan 1977.

6 CHAPTER

Conclusions

Maria Margarethe Winckelmann (1670 – 1720)
Astrónoma austriaca, contribuyó al establecimiento de la Academia de las Ciencias de Berlín y fue la primera mujer en descubrir un cometa, el C/1702 H1.

The development of this doctoral thesis arose as a response to the need to work with problems where large amounts of data appear. In particular, a change of focus was pursued when working with large-dimensional linear systems: abandoning classical tools and opting for an approach where tensor techniques are used. As we have already commented, these techniques are more effective and efficient when we face large-dimensional problems, and they are 'the remedy' to solve the dimensionality curse problem suffered by classical mechanisms. A simple example is the one recently shown by the magazine *Nature (2022)* in [14], where a new way to perform matrix products using the Kronecker product is explained, speeding up current calculation times.

Following this idea, we have deepened the study of methods and algorithms that use tensors to solve optimization problems, such as the famous Greedy Rank-One Updated Algorithm and the family of PGD methods. The analysis of these tools has

led us to develop a new matrix decomposition algorithm, which takes advantage of the structure of certain square matrices and approximates them to their Laplacian form. In Chapter 3, we have presented and demonstrated the result that explains this procedure and the algorithm that describes the steps necessary to carry it out.

The most surprising result is found in the application of this algorithm together with the GROU algorithm when solving certain Partial Derivative Equations. In particular, in Chapter 5, we have studied the Laplacian structure of the matrices that come from the discretization of a second-order PDE, and we have shown how the convergence to the solution is faster when we solve these linear systems by the aforementioned tensor techniques, surpassing Matlab's own operator, $A\backslash b$.

However, we have also studied other possible applications, such as in the field of graphs (chapter 4). In this case, we have studied the similarity between the adjacency matrices of regular graphs or Watts–Strogatz networks and their corresponding approximations in Laplacian form.

This new approach opens up a wide range of possibilities for dealing with problems that have been challenging up to now.

Chapter 7

References

[1] A. Ammar, F. Chinesta, and A. Falcó. On the convergence of a Greedy Rank-One Update algorithm for a class of linear systems. *Arch. Comput. Methods Eng.*, 17(4):473–486, 2010.

[2] M. Ay, K.-I. Goh, M. Cusick, A.-L. Barabasi, M. Vidal, and et al. Drug–target network. *Nat. Biotechnol.*, 25(10):1119–1127, 2007.

[3] J. Banasiak and M. Mokhtar-Kharroubi. *Evolutionary Equations with Applications in Natural Sciences*. Springer, 2015.

[4] A.-L. Barabási. *Network science*. Cambridge University Press, 2016.

[5] X. Chen and L. Pan. A survey of graph cuts/graph search based medical image segmentation. *IEEE Rev. Biomed. Eng.*, 11:112–124, 2018.

[6] J. A. Conejero, A. Falcó, and M. Mora-Jiménez. A pre-processing method for the implementation of the Greedy-Rank One Algorithm for a class of linear systems. *AIMS Mathematics*, 8:25633–25653, 2023.

[7] J. A. Conejero, A. Falcó, and M. Mora-Jiménez. Structure and Approximation Properties of Laplacian-Like Matrices. *Results in Mathematics*, 78(184), 2023.

[8] F. Cong, Q.-H. Lin, L.-D. Kuang, X.-F. Gong, P. Astikainen, and T. Ristaniemi. Tensor decomposition of EEG signals: A brief review. *Journal of Neuroscience Methods*, 248:59–69, 2015.

[9] R. Dobrin and P. M. Duxbury. Minimum spanning trees on random networks. *Phys. Rev. Lett.*, 86:5076–5079, May 2001.

[10] A. Falcó. Tensor Formats Based on Subspaces are Positively Invariant Sets for Laplacian-Like Dynamical Systems. In *Numerical Mathematics and Advanced Applications ENUMATH 2013*, Lecture Notes in Computational Science and Engineering, pages 315–323. Springer, 2015.

[11] A. Falcó, W. Hackbusch, and A. Nouy. On the Dirac–Frenkel Variational Principle on Tensor Banach Spaces. *Foundations of Computational Mathematics*, 19(1):159–204, 2018.

[12] A. Falcó, L. Hilario, N. Montés, M. Mora, and E. Nadal. Towards a Vector Field Based Approach to the Proper Generalized Decomposition (PGD). *Mathematics*, 9(1):34, 2020.

[13] A. Falcó and A. Nouy. Proper generalized decomposition for nonlinear convex problems in tensor Banach spaces. *Numer. Math.*, 121:503–530, 2012.

[14] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47–53, 10 2022.

[15] L. Freeman. The development of social network analysis. *A study in the sociology of science*, 1(687):159–167, 2004.

[16] J. Gallier and J. Quaintance. *Differential Geometry and Lie Groups*. Springer International Publishing, 2020.

[17] I. Georgieva and C. Hofreither. Greedy low-rank approximation in Tucker format of solutions of tensor linear systems. *J. Comput. Appl. Math.*, 358:206–220, 2019.

[18] G. Golub and C. Van Loan. *Matrix computations*. JHU press, 2013.

[19] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus (Second Edition)*. Springer Series in Computational Mathematics. Springer Cham, 2019.

[20] W. Hackbusch, B. Khoromskij, S. Sauter, and E. Tyrtyshnikov. Use of tensor formats in elliptic eigenvalue problems. *Numer. Lin. Algebra Appl.*, 19:133–151, 2012.

[21] G. Heidel, V. Khoromskaia, B. Khoromskij, and V. Schulz. Tensor product method for fast solution of optimal control problems with fractional multidimensional Laplacian in constraints. *J. Comput. Phys.*, 424:109865, 2021.

[22] G. Heidel, V. Khoromskaia, B. Khoromskij, and V. Schulz. Tensor product method for fast solution of optimal control problems with fractional multidimensional Laplacian in constraints. *J. Comput. Phys.*, 424:109865, 2021.

[23] D. Hincapié and J. Ospina. Algebraic analysis of social networks for bio-surveillance: The cases of sars-beijing-2003 and ah1n1 influenza-méxico-2009. In H. R. Arabnia and Q.-N. Tran, editors, *Software Tools and Algorithms for Biological Systems*, pages 751–761. Springer New York, New York, NY, 2011.

[24] D. Hong, T. G. Kolda, and J. A. Duersch. Generalized Canonical Polyadic Tensor Decomposition. *SIAM Review*, 62(1):133–163, 2020.

[25] W. R. Inc. Mathematica, Version 13.1. Champaign, IL, 2022.

[26] W. Jiang, G. Wang, M. Bhuiyan, and J. Wu. Understanding graph-based trust evaluation in online social networks: Methodologies and challenges. *ACM Comput. Surv.*, 49(1), 2016.

[27] T. G. Kolda and B. W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500, 2009.

[28] F. Kuo and I. Sloan. Lifting the Curse of Dimensionality. *Notices of the AMS*, 52:1320–1328, 12 2005.

[29] C. Leiserson, R. Rivest, T. Cormen, and C. Stein. *Introduction to algorithms*, volume 3. MIT press - McGraw Hill, 2001.

[30] C. Liu, Y. Pan, J. Li, and L. Dai. The normalized laplacians on both two iterated constructions associated with graph and their applications. *Journal of Applied Mathematics and Physics*, 8:838–860, 2020.

[31] H. Liu, M. Dolgushev, Y. Qi, and Z. Zhang. Laplacian spectra of a class of small-world networks and their applications. *Scientific Reports*, 5(1):1–7, 2015.

[32] D. Luzardo and A. Peña P. Historia del Álgebra Lineal hasta los Albores del Siglo XX. *Divulgaciones Matemáticas*, 14(2):153–170, 2006.

[33] K. M. and K. Kavitha. A comparative study of transportation problem by graph theoretical algorithms. *Advances in Mathematics: Scientific Journal*, 9:6251–6260, 08 2020.

[34] F. MacLean. Knowledge graphs and their applications in drug discovery. *Expert Opinion on Drug Discovery,* 16:1–13, 04 2021.

[35] MATLAB. *version R2021b*. The MathWorks Inc., Natick, Massachusetts, 2021.

[36] S. Milgram. The small world problem. *Psychol. Today*, 2(1):60–67, 1967.

[37] P. Nakkirt. The eigenvalue distribution of the Watts-Strogatz random graph. *arXiv preprint arXiv:2009.00332*, 2020.

[38] M. Newman, D. Watts, and S. Strogatz. Random graph models of social networks. *Proc. Natl. Acad. Sci. USA*, 99(suppl_1):2566–2572, 2002.

[39] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E*, 74:036104, Sep 2006.

[40] A. Nouy. *Chapter 4: Low-Rank Methods for High-Dimensional Approximation and Model Order Reduction.*, pages 171–226. Society for Industrial and Applied Mathematics, 2017.

[41] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Phys. Rev. Let.*, 86(14):3200, 2001.

[42] W. Qin, H. Wang, F. Zhang, J. Wang, X. Luo, and T. Huang. Low-Rank High–Order Tensor Completion with applications in Visual Data. *IEEE Transactions on Image Processing*, 31:2433–2448, 2022.

[43] C. Quesada, G. Xu, D. González, I. Alfaro, A. Leygue, M. Visonneau, E. Cueto, and F. Chinesta. Un método de descomposición propia generalizada para operadores diferenciales de alto orden. *Rev. Int. Metod. Numer.*, 31(3):188–197, 2015.

[44] B. RB. *Graphs and matrices*, volume 27. Springer, 2010.

[45] G. Robins, T. Snijders, P. Wang, M. Handcock, and P. Pattison. Recent developments in exponential random graph (p*) models for social networks. *Social Networks*, 29(2):192–215, 2007. Special Section: Advances in Exponential Random Graph (p*) Models.

[46] S. Robledo, N. Duque, and J. Zuluaga-Giraldo. Difusión de productos a través de redes sociales: una revisión bibliográfica utilizando la teoría de grafos. *Respuestas*, 18:28–42, 07 2013.

[47] Y. Saad. Iterative methods for linear systems of equations: A brief historical journey. In S. Brenner, I. Shparlinski, C. Shu, and D. Szyld, editors, *75 Years of Mathematics of Computation*, volume 754 of *Contemporary Mathematics*. American Mathematical Society, 2020.

[48] Y. Saad and H. Van der Vorst. Iterative solution of linear systems in the 20th century. *Journal of Computational and Applied Mathematics*, 123(1):1–33, 2000.

[49] O. Sadykova and G. Il'bahtin. The definition of Algorithmic Thinking. 01 2020.

[50] V. Simoncini. Numerical solution of a class of third order tensor linear equations. *Boll. Unione. Mat. Ital.*, 13:429–439, 2020.

[51] C. Steiner. *Automate This: How Algorithms Came to Rule Our World*. Penguin Group, Westminster, 2012.

[52] G. Strang. *Linear algebra and its applications*. Belmont, CA: Thomson, Brooks/Cole, 2006.

[53] I. Takigawa and H. Mamitsuka. Graph mining: Procedure, application to drug discovery and recent advances. *Drug discovery today*, 18, 08 2012.

[54] L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[55] D. Watts. *Six degrees: The science of a connected age*. W. W. Norton & Company, 01 2003.

[56] D. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 11 1998.

[57] R. Żochowska and P. Soczówka. Analysis of selected transportation network structures based on graph measures. *Scientific Journal of Silesian University of Technology. Series Transport*, 98:223–233, 03 2018.