Mathematical Biosciences
and Engineering

http://www.aimspress.com/journal/mbe

MBE, 20(9): 16551–16595.
DOI: 10.3934/mbe.2023738
Received: 16 May 2023
Revised: 03 August 2023
Accepted: 09 August 2023
Published: 17 August 2023

Review

Investigating difficulties and enhancing understanding in linear algebra: Leveraging SageMath and ChatGPT for (orthogonal) diagonalization and singular value decomposition

Natanael Karjanto[*]

Department of Mathematics, University College, Natural Science Campus, Sungkyunkwan
University, Suwon 16419, Republic of Korea

* **Correspondence:** Email: natanael@skku.edu.

Abstract: We explored some common challenges faced by undergraduate students when studying linear algebra, particularly when dealing with algorithmic thinking skills required for topics such as matrix factorization, focusing on (orthogonal) diagonalization and singular value decomposition (SVD). To address these challenges, we introduced SageMath, a Python-based open-source computer algebra system, as a supportive tool for students performing computational tasks despite its static output nature. We further examined the potential of dynamic ChatGPT, an AI-based chatbot, by requesting examples or problem-solving assistance related to (orthogonal) diagonalization or the SVD of a specific matrix. By reinforcing essential concepts in linear algebra and enhancing computational skills through effective practice, mastering these topics can become more accessible while minimizing mistakes. Although static in nature, SageMath proved valuable for confirming calculations and handling tedious computations because of its easy-to-understand syntax and accurate solutions. However, although dynamic ChatGPT may not be fully reliable for solving linear algebra problems, the errors it produces can serve as a valuable resource for improving critical thinking skills.

Keywords: linear algebra; (orthogonal) diagonalization; singular value decomposition; computer algebra system; AI-based chatbot; SageMath; ChatGPT; algorithmic thinking skills

1. Introduction

Linear algebra is a branch of mathematics that deals with linear equations, linear functions, and their representations in vectors and matrices. This involves the study of vector spaces, linear transformations, matrices, determinants, eigenvalues, and eigenvectors. Linear algebra is a fundamental tool in many areas of mathematics, including geometry, calculus, optimization, and numerical analysis. It also has practical applications in various fields such as physics, engineering, computer science, and economics [1–20].

Similar to other fields of mathematics, algorithmic thinking in linear algebra is a crucial skill for understanding and mastering the topics. Often also called computational thinking, algorithmic thinking is the process of breaking down complex mathematical problems into smaller, more manageable parts, and then solving them systematically using a sequence of steps or algorithms. This process involves identifying patterns, constructing algorithms, and developing logical and analytical skills to solve problems [21–24]. Teaching and learning linear algebra for both mathematics and non-mathematics majors provides an opportunity for cultivating and embracing problem-solving, logical reasoning, and computational thinking skills, which are essential in various areas of science and engineering requiring solid knowledge of linear algebra, such as operational research, computer science, data science, and machine learning, among others [3, 20, 25–33]. Strengthening computational thinking among future teachers will also be useful when they eventually train future student generations [34, 35].

In addition to being able to manipulate a matrix algebraically and perform matrix operations, some essential concepts in linear algebra include but are not limited to understanding vector spaces and subspaces, applying linear transformation, working with norms and inner products, finding least-squares solutions, and the ability to perform matrix factorization. In this study, we focused on the latter. Occasionally also called decomposition, factorization in linear algebra refers to the process of decomposing a matrix into several, often simpler, matrices that can be analyzed and manipulated more easily. These simpler matrices may have special properties or structures that make them easier to analyze or implement in computations. Mastering algorithmic thinking in performing matrix factorization is essential for every linear algebra learner because this topic not only requires other basic concepts in linear algebra, but it is also an essential building block for understanding other more complicated topics in linear algebra.

There are various types of matrix decompositions, such as LU decomposition, QR decomposition, diagonalization, orthogonal diagonalization, singular value decomposition (SVD), and eigendecomposition. The latter refers to the factorization of a matrix into its canonical form, where the matrix is represented by its eigenvalues and eigenvectors. Each type of decomposition has its own specific properties and applications. For example, LU decomposition is useful for solving systems of linear equations, QR decomposition is used for least-squares problems, SVD is used for data compression and feature extraction, and eigendecomposition is used for analyzing the behavior of linear operators. Although matrix diagonalization only applies to square matrices and orthogonal diagonalization only applies to a special class of symmetric square matrices, SVD can be implemented for any size of the rectangular matrix, which does not necessarily have to be square.

In short, SVD is a factorization method in linear algebra that decomposes a real- or complex-valued matrix into three components: a diagonal matrix of singular values and two unitary (orthogonal) matrices. Mathematical applications of SVD encompass the calculation of the (Moore-Penrose) pseudoinverse, approximating a matrix, and determining the rank, range, and null space of a matrix. SVD is a widely utilized technique in data analysis and machine learning for reducing the dimensionality of data and extracting important features. SVD is also used in various applications in science and engineering, including signal and image processing, text mining, data least-squares fitting, process control, and recommendation systems [36–46]. The singular values obtained from SVD represent the importance of each feature in the original data and can be used to reconstruct the data with different levels of accuracy.

A common curriculum in linear algebra outlines that students ought to know how to calculate both matrix diagonalization and orthogonal diagonalization before being introduced to SVD. However, we observed that students often encountered some difficulties with these decompositions. For orthogonal diagonalization, several students often forget to normalize the associated eigenvectors to become unit vectors. In other cases, when one eigenvalue yields two associated eigenvectors, some students also forget to transform the set of eigenvectors into an orthonormal set, which can be easily done by orthogonal projection via the Gram-Schmidt process, keeping one of them while projecting the other.

For SVD, a common mistake is in finding the singular values of a matrix. Let A be a matrix. Instead of finding the eigenvalues of $A^T A$, some students calculated the eigenvalues of A, took their square root, and designated them as singular values. Another difficulty occurs when the matrix does not have a full rank, for which the students need to find the missing one or more orthonormal eigenvectors in one of the matrices by implementing the orthogonal property or cross product vector operation. In the absence of these eigenvectors, the resulting factorization is called "reduced SVD" instead of (full) SVD.

To the best of our knowledge, this issue of common mistakes and learners' struggles in understanding (orthogonal) diagonalization and SVD has not been fully addressed in the body of published literature. The closest article that is tangentially related to our study is Yildiz Ulus' (2013) study, in which the author investigated teaching diagonalization using advanced calculators and observed that such technological tools are beneficial for learners' acquisition of algorithmic mathematical knowledge for this particular topic of linear algebra [47].

Although Lazar (2012) argued in his master's thesis that a solid understanding of fundamental concepts in linear algebra is essential for mastering SVD, the author did not inquire the participants in his study regarding the difficulty or common mistakes they encountered when studying the topic [48]. Finally, Zandieh et al. (2017) explored student learning in linear algebra, where they touched briefly on symbolizing the diagonalization equation $A = PDP^{-1}$ but did not discuss students' difficulty in acquiring algorithmic thinking skills [49].

This study attempts to fill this gap by investigating common mistakes and challenges when students learn linear algebra, particularly (orthogonal) diagonalization and SVD. Furthermore, because teaching and learning in contemporary mathematics cannot be dismantled by utilizing technological tools, we are also interested in investigating whether computer software or another newly arrived technology can assist and enhance students' learning, instead of disrupting and marring it. The scope of our computer algebra system (CAS) is SageMath, which has a static nature, and we will consider ChatGPT as an artificial intelligence (AI)-assisted tool, which features dynamic interaction output. In the following paragraphs, we provide a brief overview of the static CAS SageMath and dynamic ChatGPT.

What is a CAS? A CAS is a software program that allows the manipulation and computation of mathematical expressions and symbols, including algebraic equations, calculus, and other mathematical functions. It is designed to perform symbolic manipulation, numerical computations, and graphics, as well as to provide tools for solving equations, manipulating matrices and vectors, and performing other mathematical operations. CASs are commonly used in scientific research, engineering, and education, and they can be used to solve complex mathematical problems that may be too difficult or time-consuming to solve by hand [50–65]. Examples of popular CASs include (wx)Maxima [66–70], Wolfram Mathematica [71–76], Maplesoft Maple [77–82], and MathWorks MATLAB [83–87].

What is SageMath? SageMath (also known as Sage) is a free open-source mathematical software system that uses Python as its primary programming language. It aims to provide an alternative to commercial mathematical software systems, such as Wolfram Mathematica and Maplesoft Maple, while also providing an interface to other popular mathematics software systems, such as MathWorks MATLAB and GAP, the latter being a system for computational discrete algebra. SageMath has a wide range of capabilities, including algebraic and numerical computations, graphics, symbolic manipulation, and combinatorics. Its development is community-driven and it has been released under the GNU General Public License [**?**, 89–93].

Remarkably, SageMath offers robust support not only for linear algebra but also for a wide range of related subjects, boasting numerous built-in functions and capabilities dedicated to matrix operations, vector manipulations, and linear transformations. SageMath provides a variety of algorithms and methods for solving linear algebra problems, such as performing matrix operations, solving linear systems of equations, finding eigenvalues and eigenvectors, and computing matrix decompositions (e.g., LU, QR, SVD). SageMath is a powerful tool for symbolic linear algebra, which allows exact computations with variables and expressions. Despite being static in nature, we are convinced that SageMath can be useful for teaching and learning linear algebra, not only in terms of assisting computational tasks but also with regard to check whether our hand calculations were correct. In addition, the time saved on computations can be channeled toward other purposes, such as understanding deeper mathematical concepts in linear algebra or exploring various problems within the subject matter.

What is ChatGPT? The Chat Generative Pre-trained Transformer, that is, ChatGPT, is a language model developed by OpenAI, a USA-based AI research laboratory consisting of a team of researchers and engineers dedicated to creating safe and beneficial AI. Although OpenAI was founded in 2015, ChatGPT, as one of its products, was launched as a prototype in November 2022. Amazingly, it reached one million users within five days after its launch. An AI tool built on top of the unsupervised transformer language model GPT-3, ChatGPT was trained on a large dataset of text and could generate responses to questions, write coherent paragraphs, and even conduct a conversation with users. Its purpose is to assist with various tasks, such as answering questions, providing information, and generating text in a conversational manner.

Interestingly, as an AI language model, ChatGPT has been trained on a large corpus of texts, including mathematical concepts and problems. Thus, it is capable of solving mathematics problems, at least that is what it claims. ChatGPT admits that its ability to solve mathematical problems may be limited by its training data, although it can certainly provide assistance and guidance on various mathematical topics. Specifically, when asked whether it can solve problems in linear algebra, ChatGPT claimed that it "possesses knowledge and understanding of linear algebra concepts and can provide solutions to problems in this field." During the past few months, the number of published articles related to ChatGPT and its capabilities has increased steadily, including many that appear in various preprint servers. The following examples cover only a few articles on ChatGPT related to mathematics.

Recently, Frieder et al. (2023) investigated the mathematical capabilities of ChatGPT by asking a wide range of questions, and although it understood the question, it often failed to provide correct solutions. They concluded that the mathematical abilities of ChatGPT were significantly lower than those of average mathematics graduate students [94]. Shakarian et al. (2023) also evaluated ChatGPT for mathematical word problems and discovered that its performance was dramatically altered based on the requirement to show its work. When the chatbot was not required to reveal the complete solution, it failed 84% of the time, whereas it failed by only 20% when revealing the detailed solution was requested [95]. Azaria (2022) tested the numerical literacy of ChatGPT and noticed that when it comes to using number digits, ChatGPT is rather biased, with 7 being the most frequent digit generated by the machine, which also turns out to be many people's favorite number [96]. Borji (2023) focused on ChatGPT's failures in mathematics, including in areas of arithmetic, logic, and reasoning [97].

Certainly, ChatGPT is not the only chatbot available on the market. There are numerous alternatives to ChatGPT and the AI race among different companies is becoming fierce. Some examples include but are not limited to Google Bard AI, Microsoft Bing Chat, Amazon Codewhisperer, Github Copilot, Chatsonic, Character AI, Quora Poe, etc. Some early findings suggest that when it comes to solving mathematics problems at the high school level, the Vietnamese students still performed better than both ChatGPT and Bing Chat [98]. Another study from Vietnam on the mathematics test for its national high school graduation examination indicated that Google Bard's performance was lagging behind its competitors, that is, ChatGPT and Bing Chat [99].

In terms of output production, SageMath and other CASs tend to be static, whereas ChatGPT and its competitors are dynamic. The latter can provide a step-by-step explanation of a solution to a particular problem. For matrix factorization in linear algebra, it is essential for learners to understand not only the technical details of a calculation but also to grasp the algorithm behind any particular problem task. Certainly, matrix factorization is a broad topic in itself, and any attempt to discuss other types of matrix factorization should be addressed separately elsewhere. Presently, our focal point for the topic of "matrix factorization" covers only diagonalization, orthogonal diagonalization, and SVD. Based on the literature mentioned earlier in this introduction, we consider the following research questions:

- What are some common mistakes and difficulties that students encounter when learning matrix factorization in linear algebra?

- How can a static CAS such as SageMath assist students in learning linear algebra, particularly in (orthogonal) diagonalization and SVD?

- Can we rely on the dynamic ChatGPT to better understand (orthogonal) diagonalization and SVD?

The remainder of this article is organized as follows. After this introduction, Section 2 features some common mistakes that students often make when learning (orthogonal) diagonalization and SVD. Section 3 continues with SageMath and its ability to assist students in learning (orthogonal) diagonalization and SVD. Section 4 features some examples in which we asked the chatbot to solve problems related to (orthogonal) diagonalization and SVD. We also discuss where ChatGPT makes mistakes and encounters troubles in completing assigned tasks. Section 5 provides some applications of (orthogonal) diagonalization and SVD in life sciences and engineering. Finally, Section 6 discusses the results and concludes the study.

2. Common mistakes in matrix factorization

2.1. Matrix diagonalization

In this section, we consider some common mistakes that students encounter when learning to diagonalize both non-symmetric and symmetric matrices. We also consider similar aspects of SVD. For the former, if A is an $n \times n$ square matrix, then A is diagonalizable whenever it possesses n linearly independent eigenvectors. Usually, the students did not encounter difficulties in diagonalizing a non-symmetric diagonalizable matrix provided they understood the procedure for finding it. A typical algorithm starts with finding the eigenvalues and their associated eigenvectors, constructing a diagonal matrix D, constructing an invertible matrix P, finding its inverse P^{-1}, and expressing its diagonalization, that is, $A = PDP^{-1}$. The following example illustrates this algorithm.

Suppose that there exists a 3×3 matrix A with integer entries, given as follows:

$$A = \begin{bmatrix} 1 & 2 & 2 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}.$$

To diagonalize a matrix, we must determine its eigenvalues and their corresponding eigenvectors. The former can be obtained by solving the characteristic equation $\det(A - \lambda I) = 0$ or $(\lambda - 1)(\lambda - 3)(\lambda - 1) = 0$, which gives $\lambda_1 = 1$, $\lambda_2 = 3$, and $\lambda_3 = 1$. Note that some students might attempt to express the characteristic equation in cubic form, that is, $\lambda^3 - 5\lambda^2 + 7\lambda - 3 = 0$, and solve this for λ. Although in other cases, this might be a necessary step and even inevitable, we do not have to do this particular step in this specific example. Up to this point, we can form a diagonal matrix D by placing the eigenvalues along the diagonal:

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

However, this was not the only option. Depending on the choice of eigenvalues designated as the first, second, third, and so on, we might obtain a different expression for D. Coincidentally, both the algebraic multiplicity and geometric multiplicity for the eigenvalue $\lambda = 1$ from the matrix in this example are equal, that is, two. In this case, we are guaranteed to obtain three linearly independent eigenvectors and thus, the matrix is diagonalizable.

Let $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ be the corresponding eigenvectors, we found that

$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}, \qquad \mathbf{p}_2 = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

An invertible matrix P can then be formed by stacking these eigenvectors as its columns:

$$P = \begin{bmatrix} 0 & 2 & 1 \\ 1 & 1 & 0 \\ -1 & 1 & 0 \end{bmatrix}.$$

Its inverse P^{-1} is given by

$$P^{-1} = \begin{bmatrix} 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & -1 & -1 \end{bmatrix}.$$

Finally, we can write a diagonalization of A by expressing it as PDP^{-1} and verify that the products are reduced to the original matrix A:

$$\underbrace{\begin{bmatrix} 1 & 2 & 2 \\ 0 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0 & 2 & 1 \\ 1 & 1 & 0 \\ -1 & 1 & 0 \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & -1 & -1 \end{bmatrix}}_{P^{-1}}.$$

2.2. *Orthogonal diagonalization*

The following example illustrates an orthogonal diagonalization for a symmetric matrix, whereby one of the eigenvalues has both algebraic and geometric multiplicities of two; thus, the resulting eigenvectors are not orthogonal. Consider the following 3 × 3 matrix B:

$$B = \begin{bmatrix} 3 & -4 & -4 \\ -4 & 3 & -4 \\ -4 & -4 & 3 \end{bmatrix}.$$

This matrix has a characteristic polynomial of $\lambda^3 - 9\lambda^2 - 21\lambda + 245$, which yields three real-valued eigenvalues $\lambda_1 = -5$ and $\lambda_2 = 7 = \lambda_3$ upon solving the characteristic equation. A diagonal matrix D is given by

$$D = \begin{bmatrix} -5 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 7 \end{bmatrix}.$$

Let $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_3$ be the eigenvectors corresponding to these eigenvalues, we have

$$\mathbf{q}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \qquad \mathbf{q}_2 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \qquad \mathbf{q}_3 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}.$$

A common mistake we often encounter is that some students construct an invertible matrix P directly from these eigenvectors without normalizing them, and then they find its inverse by simply transposing it, that is, $P^T = P^{-1}$. However, the correct step would be to construct an orthonormal set of eigenvectors that form the column of P. We observe that $\mathbf{q}_1$ is orthogonal to both $\mathbf{q}_2$ and $\mathbf{q}_3$. However, $\mathbf{q}_2$ and $\mathbf{q}_3$ are not orthogonal because they originate from the same eigenspace. Only the eigenvectors associated with the distinct eigenvalues are orthogonal. Applying the Gram-Schmidt process to the set $\{\mathbf{q}_2, \mathbf{q}_3\}$, we can obtain a new set of orthogonal eigenvectors $\left\{\mathbf{q}_2, \mathbf{q}_3'\right\}$, where

$$\mathbf{q}_3' = \mathbf{q}_3 - \frac{\langle \mathbf{q}_2, \mathbf{q}_3 \rangle}{\langle \mathbf{q}_2, \mathbf{q}_2 \rangle}\mathbf{q}_2 = \frac{1}{2}\begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}.$$

Angle brackets denote the usual inner (dot) product. By normalizing these vectors, we obtain an orthonormal set of eigenvectors and construct matrix P accordingly. Because P is now an orthogonal matrix, its inverse is simply its transpose, given as follows:

$$P = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \end{bmatrix}, \qquad P^{-1} = P^T = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix}.$$

We can now express an orthogonal diagonalization of B and again confirm that the right-hand side will be reduced to the original matrix upon multiplication and simplification, as shown by the following computaton:

$$\underbrace{\begin{bmatrix} 3 & -4 & -4 \\ -4 & 3 & -4 \\ -4 & -4 & 3 \end{bmatrix}}_{B} = \underbrace{\begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & -\frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \end{bmatrix}}_{P} \underbrace{\begin{bmatrix} -5 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 7 \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{bmatrix}}_{P^{-T}}.$$

2.3. SVD

Although matrix diagonalization and orthogonal diagonalization can be implemented in a square matrix, SVD encompasses both the square and rectangular matrices. Let M be an $m \times n$ real-valued matrix with rank r; then, an SVD of M is given by $M = U\Sigma V^T$, where U is an $m \times m$ orthogonal matrix, Σ is an $m \times n$ rectangular "diagonal" matrix with non-negative real numbers on the diagonal, and V is an $n \times n$ orthogonal matrix. The columns of U and V in such a decomposition are called the left- and right-singular vectors of M, respectively. Regarding complex-valued matrices, U and V are complex unitary matrices, and instead of V^T, we would have the conjugate transpose of V, that is, V^*. In general, Σ is composed of matrix blocks and admits the following form:

$$\Sigma = \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix},$$

where D is an $r \times r$ diagonal matrix for $r \leq \min\{m, n\}$. The numbers of (zero) rows and columns in the second row and column of the block are $(m - r)$ and $(n - r)$, respectively. For $r = m$, $r = n$, or $r = m = n$, some or all of the zero blocks would disappear. Furthermore, the diagonal entries of D are the first r singular values of M, that is, $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

Similar to obtaining matrix diagonalization and orthogonal diagonalization, constructing an SVD of a matrix requires a step-by-step algorithm and an understanding of (orthogonal) diagonalization. For square matrices with full rank, many learners usually encounter no difficulty when constructing an SVD of a matrix, although one must be careful when calculating eigenvalues and the corresponding eigenvectors of a symmetric matrix $M^T M$ instead of the original matrix M. However, for rectangular matrices or those with rank deficiencies, many students often could not complete the construction of an SVD, partly because one or more singular values can be zero. The following example illustrates the construction of an SVD of a square matrix with rank deficiency. Let

$$M = \begin{bmatrix} 3 & -1 \\ -6 & 2 \end{bmatrix}$$

be a matrix with rank $r = 1$; the eigenvalues of $A = M^T M$ are given by $\lambda_1 = 50$ and $\lambda_2 = 0$, which gives only one singular value $\sigma_1 = 5\sqrt{2}$ because the second eigenvalue is zero. Thus, $D = \sigma_1$ and matrix Σ contain one block of zeros in each row and column below and to the right of D, respectively, given as follows:

$$\Sigma = \begin{bmatrix} 5\sqrt{2} & 0 \\ 0 & 0 \end{bmatrix}.$$

By finding the corresponding eigenvectors of λ_1 and λ_2, we obtain $\mathbf{v}_1$ and $\mathbf{v}_2$ and set up an orthogonal matrix V:

$$V = \begin{bmatrix} 3/\sqrt{10} & 1/\sqrt{10} \\ -1/\sqrt{10} & 3/\sqrt{10} \end{bmatrix}.$$

The first column of matrix U, that is, $\mathbf{u}_1$, can be calculated using the following formula:

$$\mathbf{u}_1 = \frac{1}{\sigma_1} M\mathbf{v}_1 = \frac{1}{5\sqrt{2}} \begin{bmatrix} 3 & -1 \\ -6 & 2 \end{bmatrix} \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ -2 \end{bmatrix}.$$

The second column of matrix U, that is, $\mathbf{u}_2$, must be orthonormal to $\mathbf{u}_1$:

$$\mathbf{u}_2 = \frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

An SVD of M can be expressed as follows:

$$\underbrace{\begin{bmatrix} 3 & -1 \\ -6 & 2 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} 2/\sqrt{5} & 1/\sqrt{5} \\ 1/\sqrt{5} & -2/\sqrt{5} \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 5\sqrt{2} & 0 \\ 0 & 0 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 3/\sqrt{10} & -1/\sqrt{10} \\ 1/\sqrt{10} & 3/\sqrt{10} \end{bmatrix}}_{V^T}.$$

3. (Orthogonal) diagonalization and SVD using SageMath

To use SageMath online, we can utilize Sage Cell Server or SageMathCell, accessible online at the URL `https://sagecell.sagemath.org/`. In addition to an open-source, scalable, and easy-to-use web interface for SageMath, this cell server also allows the embedding of SageMath computations into any webpage. Without loading up any program, this is one way to conduct one-off computations using SageMath, with the idea of accessing the computation in the cloud as simply as possible, as long as one has an internet connection.

3.1. Matrix diagonalization using SageMath

The SageMath commands for a matrix diagonalization discussed in Subsection 2.1 is presented in Appendix A.1. Observe that the presence of the “`print`” command might be excessive for first-time readers who are new to SageMath. However, the expressions inside the double quotation marks, as well as an empty vertical spacing, might be omitted if one wishes.

After constructing matrix A, we asked SageMath to display it, finding an expression for its characteristic polynomial using “`A.charpoly()`”, calculating its eigenvalues using “`A.eigenalues()`”, and acquiring the corresponding eigenvectors using “`A.eigenvectors_right()`”. The outputs for the first four essential commands are given as follows:

```
Matrix A =
[1 2 2]
[0 2 1]
[0 1 2]

Characteristic polynomial of A: p(x) = x^3 - 5*x^2 + 7*x - 3

Eigenvalues of A = [3, 1, 1]

Eigenvalue, eigenvector, and geometric multiplicity:
[(3, [(1, 1/2, 1/2)], 1), (1, [(1, 0, 0),(0, 1, -1)], 2)]
```

Observe that the command "`A.eigenvectors_right()`" provides information not only about eigenvectors but also on the associated eigenvalue and their geometric multiplicity.

The next step is extracting the eigenvalues of A using the command "`A.eivenvalues()[n]`", which correspond to λ_{n+1}, where $n = 0, 1, 2$. Once we have obtained the eigenvalues, we can construct a diagonal matrix D manually by inserting each value of λ_n. Alternatively, we can also construct D directly using the command "`diagonal_matrix(A.eigenvalues())`", as shown in our code. The outputs of these two essential commands are given as follows:

```
Extracting eigenvalues:
lambda1 = 3
lambda2 = 1
lambda3 = 1

Diagonal matrix D =
[3 0 0]
[0 1 0]
[0 0 1]
```

To construct an invertible matrix P, we also need to extract the associated eigenvectors, that is, using the command "`A.eigenvectors_right()[m][1]`", where $m = 0, 1$, which corresponds to the first and second sets of the outputs in the same command in the absence of both square brackets. The second square bracket [1] indicates the second entry of each output, that is, the eigenvector(s). We can multiply by 2 to obtain the eigenvector corresponding to $\lambda_1 = 3$ with integer entries, which is given by $\mathbf{p}_1 = (2, 1, 1)$. Because the eigenvalues $\lambda_2 = 1 = \lambda_3$ have geometric multiplicities of 2, we again need to extract by adding the brackets [0] and [1], which correspond to $\mathbf{p}_2$ and $\mathbf{p}_3$, respectively. Furthermore, because these vectors appear as row columns, we need to apply a transpose command to construct P, that is, "`P=matrix([p1,p2,p3]).tranpose()`". The outputs for eigenvector extraction and P are given as follows:

```
Extracting eigenvectors:
p1 = (2, 1, 1)
p2 = (1, 0, 0)
p3 = (0, 1, -1)
```

```
Invertible matrix P =
[ 2  1  0]
[ 1  0  1]
[ 1  0 -1]
```

The final two steps are to determine the inverse matrix P^{-1} and confirm that the diagonalization process is correct. The former can be achieved using the command "`P.inverse()`", whereas the latter is performed by simply taking the product of PDP^{-1}, that is, using the command "`P*D*P.inverse()`", which should result in product simplification and a return to the original matrix A, thus confirming that the diagonalization is indeed correct. The outputs are given as follows:

```
Inverse of P, P^(-1) =
[   0  1/2  1/2]
[   1   -1   -1]
[   0  1/2 -1/2]

Calculate PDP^(-1) =
[1 2 2]
[0 2 1]
[0 1 2]
= A
```

3.2. Orthogonal diagonalization using SageMath

The SageMath commands for an orthogonal diagonalization of the matrix example considered in Subsection 2.2 are given in Appendix A.2. Similar to the previous example, we use standard SageMath commands to display the matrix, find its characteristic polynomial, compute its eigenvalues, and construct a diagonal matrix D. We have the following outputs:

```
Matrix A =
[ 3 -4 -4]
[-4  3 -4]
[-4 -4  3]

Characteristic polynomial of A: p(x) = x^3 - 9*x^2 - 21*x + 245

Eigenvalues of A = [-5, 7, 7]

Extracting eigenvalues:
lambda1 = -5
lambda2 = 7
lambda3 = 7

Diagonal matrix D =
[-5  0  0]
```

```
[ 0  7  0]
[ 0  0  7]
```

We also calculated the corresponding eigenvectors $\mathbf{q}_1$, $\mathbf{q}_2$, and $\mathbf{q}_3$, and extracted each of them accordingly, that is, using the following commands:

```
q1=A.eigenvectors_right()[0][1][0],
q2=A.eigenvectors_right()[1][1][0],
q3=A.eigenvectors_right()[1][1][1].
```

The first square bracket indicates the set of solutions containing the eigenvalue, eigenvector(s), and geometric multiplicity. Because there are two distinct eigenvalues, the entry in the first set of square brackets only takes a value of either 0 or 1. The entry inside the second set of square brackets is always 1, because we would like to extract the eigenvector. The entry in the third set of square brackets indicates which eigenvector to extract. For the first case, there is only one; thus, the value is 0. For the second case, because there are two eigenvectors, the values 0 and 1 are taken for the second and third eigenvectors, respectively. We have the following outputs:

```
Eigenvalue, eigenvector, and geometric multiplicity:
[(-5, [(1, 1, 1)], 1), (7, [(1, 0, -1),(0, 1, -1)], 2)]

Extracting eigenvectors:
q1 = (1, 1, 1)
q2 = (1, 0, -1)
q3 = (0, 1, -1)
```

However, before constructing an orthogonal matrix P, we must ensure that the set of eigenvectors is orthonormal. To check the orthogonality between vectors $\mathbf{q}_m$ and $\mathbf{q}_n$, we use the command

```
qm.inner_product(qn),
```

where $m, n = 1, 2, 3$ and $m \neq n$. If the result is zero, then both vectors are orthogonal to each other; otherwise, they are not orthogonal. We observed that $\mathbf{q}_2$ and $\mathbf{q}_3$ are not orthogonal. To obtain an orthogonal matrix, we apply the Gram-Schmidt process, using the command

```
q3-q2.inner_product(q3)/q2.inner_product(q2)*q2.
```

After obtaining the new eigenvector $\mathbf{q}_3'$, we verified that it is now orthogonal to $\mathbf{q}_2$. We obtain the following outputs:

```
Checking orthogonality:
<q1,q2> = 0
<q1,q3> = 0
<q2,q3> = 1

q2 and q3 are not orthogonal; apply the Gram-Schmidt process:

q3' = (-1/2, 1, -1/2)

Check that now q2 and q3' are orthogonal: <q2,q3'> = 0
```

The next step was to obtain an orthonormal set of eigenvectors by normalizing this orthogonal set of vectors. The norm of any vector can be calculated using the command “`sqrt(qn.inner_product(qn))`”. Thus, the unit eigenvectors $\mathbf{p}_n$ can be obtained using the command “`qn/sqrt(qn.inner_product(qn))`”, where in both instances, $n = 1, 2, 3$. The outputs are given as follows:

```
Normalize all orthogonal eigenvectors:

p1 = (1/3*sqrt(3), 1/3*sqrt(3), 1/3*sqrt(3))

p2 = (1/2*sqrt(2), 0, -1/2*sqrt(2))

p3 = (1/6*sqrt(6), -1/3*sqrt(6), 1/6*sqrt(6))
```

We can now construct an orthogonal matrix P, which is also invertible. The commands are similar to those in the previous example. We also further verified that $P^{-1} = P^T$ and the calculations for both PDP^{-1} and PDP^T reduce to the original matrix A, which confirms that an orthogonal diagonalization for A is indeed correct. Readers are presented the following outputs:

```
Construct an invertible and orthogonal matrix P:
P =
[ 1/3*sqrt(3)  1/2*sqrt(2)  1/6*sqrt(6)]
[ 1/3*sqrt(3)            0 -1/3*sqrt(6)]
[ 1/3*sqrt(3) -1/2*sqrt(2)  1/6*sqrt(6)]

Inverse of P, P^(-1) =
[ 1/3*sqrt(3)  1/3*sqrt(3)  1/3*sqrt(3)]
[ 1/2*sqrt(2)            0 -1/2*sqrt(2)]
[ 1/6*sqrt(6) -1/3*sqrt(6)  1/6*sqrt(6)]

Calculate PDP^(-1) =
[ 3 -4 -4]
[-4  3 -4]
[-4 -4  3]
= A

Transpose of P, P^T =
[ 1/3*sqrt(3)  1/3*sqrt(3)  1/3*sqrt(3)]
[ 1/2*sqrt(2)            0 -1/2*sqrt(2)]
[ 1/6*sqrt(6) -1/3*sqrt(6)  1/6*sqrt(6)]

Calculate PDP^T =
[ 3 -4 -4]
[-4  3 -4]
[-4 -4  3]
```

```
= A
```

3.3. SVD using SageMath

The SageMath commands for constructing an SVD of the matrix discussed in Subsection 2.3 are displayed in Appendix A.3.

After constructing matrix M, we calculate a symmetric matrix A, compute its eigenvalues, take the square root, and obtain the singular values of M. This process can be achieved using the following commands:

```
sigma1=sqrt(A.eigenvalues()[0]),
sigma2=sqrt(A.eigenvalues()[1]).
```

Matrix Σ is constructed manually using the command

```
Sigma=matrix([[sigma1,0],[0,sigma2],[0,0]]).
```

We have the following ouputs:

```
Matrix M =
[ 2 -2]
[-3 -4]
[-4 -3]

Matrix M^T M =
[29 20]
[20 29]

Eigenvalues of M^T M = [49, 9]

Singular value of M:
sigma1 = 7
sigma2 = 3

Matrix Sigma =
[7 0]
[0 3]
[0 0]
```

We then calculate the eigenvectors of A, normalize them, and construct an orthogonal matrix V, which can be achieved using the following commands:

```
w1=A.eigenvectors_right()[0][1][0],
v1=w1/w1.norm(),
w2=A.eigenvectors_right()[1][1][0],
v2=w2/w2.norm().
```

We have the following outputs, leading to V:

```
Eigenvalue, eigenvector, and geometric multiplicity:
[(49, [(1, 1)], 1), (9, [(1, -1)], 1)]

Eigenvectors of M^T M:
v1 = (1/2*sqrt(2), 1/2*sqrt(2))
v2 = (1/2*sqrt(2), -1/2*sqrt(2))

Matrix V =
[ 1/2*sqrt(2)  1/2*sqrt(2)]
[ 1/2*sqrt(2) -1/2*sqrt(2)]
```

To construct an orthogonal matrix U, we require three linearly independent orthonormal vectors $\mathbf{u}_1$, $\mathbf{u}_2$, and $\mathbf{u}_3$. In this example, both vectors $\mathbf{u}_1$ and $\mathbf{u}_2$ are calculated using a formula that involves a singular value σ_i, the original matrix M, and the eigenvectors of A, that is, $\mathbf{v}_i$, $i = 1, 2$, given by

$$\mathbf{u}_i = \frac{1}{\sigma_i} M\mathbf{v}_i, \qquad \text{for} \quad i = 1, 2.$$

The SageMath commands for this computation are straightforward:

```
u1=1/sigma1*M*v1,
u2=1/sigma2*M*v2.
```

However, because the third singular value σ_3 does not exist, we must find $\mathbf{u}_3$ using other techniques. To ensure its orthogonality with $\mathbf{u}_1$ and $\mathbf{u}_2$, $\mathbf{u}_3$ can be calculated using either a dot or cross product operation. In this example, we implemented the latter, that is, "`u3=u1.cross_product(u2)`". As always, we should not forget to verify that these vectors are orthogonal, and we can do this by simply calculating the dot product between the two vectors "`um.dot_product(un)`", where $m, n = 1, 2, 3$, but $m \neq n$. We present the following outputs:

```
Left singular vectors of M:
u1 = (0, -1/2*sqrt(2), -1/2*sqrt(2))
u2 = (2/3*sqrt(2), 1/6*sqrt(2), -1/6*sqrt(2))
u3 = (1/3, -2/3, 2/3)

Checking orthogonality
u1.u2 = 0
u1.u3 = 0
u2.u3 = 0
```

Because vectors $\mathbf{u}_i$, $i = 1, 2$, and 3 appear as row vectors, matrix U is constructed by taking the transpose of these row vectors. Finally, we confirm that the construction of our SVD is correct because the product of each matrix returns to the original matrix M, that is, $M = U\Sigma V^T$. The outputs are given as follows:

```
Matrix U =
[            0  2/3*sqrt(2)          1/3]
```

```
[-1/2*sqrt(2)  1/6*sqrt(2)         -2/3]
[-1/2*sqrt(2) -1/6*sqrt(2)          2/3]

SVD of M = U*Sigma*V^T =
[ 2 -2]
[-3 -4]
[-4 -3]
= M
```

We observed that some students did not know how to find the third vector $\mathbf{u}_3$. Very often, they simply ignored and abandoned it entirely, constructing a factorization using the first two vectors in matrix U instead, that is, $\mathbf{u}_1$ and $\mathbf{u}_2$. In this situation, they were confused between “reduced SVD” and “(complete, non-reduced) SVD”. The former depends on the rank of M, that is, r, and instead of employing Σ, we admit the diagonal matrix D instead. Thus, the reduced SVD for M in the example considered in this subsection is given by

$$\underbrace{\begin{bmatrix} 2 & -2\\ -3 & -4\\ -4 & -3\end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} 0 & 2\sqrt{2}/3\\ -1/\sqrt{2} & \sqrt{2}/6\\ -1/\sqrt{2} & -\sqrt{2}/6\end{bmatrix}}_{U_2} \underbrace{\begin{bmatrix} 7 & 0\\ 0 & 3\end{bmatrix}}_{D} \underbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix}}_{V_2^T}.$$

This reduced SVD is beneficial for calculating the pseudoinverse M^+ (also called the Moore-Penrose inverse) of the matrix M. For the particular M considered in this subsection, its pseudoinverse is given by

$$\underbrace{\frac{1}{63}\begin{bmatrix} 14 & -1 & -8\\ -14 & -8 & -1\end{bmatrix}}_{M^+} = \underbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2}\\ 1/\sqrt{2} & -1/\sqrt{2}\end{bmatrix}}_{V_2} \underbrace{\begin{bmatrix} 1/7 & 0\\ 0 & 1/3\end{bmatrix}}_{D^{-1}} \underbrace{\begin{bmatrix} 0 & -1/\sqrt{2} & -1/\sqrt{2}\\ 2\sqrt{2}/3 & \sqrt{2}/6 & -\sqrt{2}/6\end{bmatrix}}_{U_2^T}.$$

SageMath can certainly compute the pseudoinverse of a matrix in a matter of seconds using the command “`print(M.pseudoinverse())`”. However, it is important to exercise discretion and only use this step to check the final calculation, rather than as a substitute for an appropriate algorithm to find it.

4. (Orthogonal) diagonalization and SVD in ChatGPT

We used the version from March 14, 2023, for any responses generated by ChatGPT in this study.

4.1. Matrix diagonalization in ChatGPT

When asking the chatbot to provide an example of matrix diagonalization without specifying any size of the matrix, ChatGPT provided an answer by selecting a simple 2 × 2 matrix, as illustrated in the following example. We input an inquiry “Provide an example of a matrix diagonalization.” ChatGPT responded as follows:

```
Sure, here's an example of matrix diagonalization:
Suppose we have the following matrix A:
```

$$\begin{bmatrix} 3 & 1\\ 0 & 2\end{bmatrix}.$$

We want to diagonalize A, which means finding a diagonal matrix D and an invertible matrix P such that $A = PDP^{-1}$, where D has the eigenvalues of A along the diagonal.

To diagonalize A, we first find the eigenvalues of A. The characteristic polynomial of A is:

$$\begin{aligned} \det(A - \lambda I) &= \det\begin{pmatrix} 3-\lambda & 1 \\ 0 & 2-\lambda \end{pmatrix} \\ &= (3-\lambda)(2-\lambda) = 6 - 5\lambda + \lambda^2. \end{aligned}$$

Solving for the roots of the characteristic polynomial, we get $\lambda = 3$ and $\lambda = 2$. These are the eigenvalues of A.

Next, we find the eigenvectors corresponding to each eigenvalue. For $\lambda = 3$, we solve the equation $(A - 3I)\mathbf{x} = \mathbf{0}$:

$$\begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This gives us the equation $x_2 = 0$ and x_1 can be any nonzero value. So an eigenvector corresponding to $\lambda = 3$ is $[1 \quad 0]^T$.

For $\lambda = 2$, we solve the equation $(A - 2I)\mathbf{x} = \mathbf{0}$:

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This gives us the equation $x_1 + x_2 = 0$, or $x_1 = -x_2$. So an eigenvector corresponding to $\lambda = 2$ is $[-1 \quad 1]^T$.

We can normalize these eigenvectors to get:

$$\begin{aligned} \mathbf{v}_1 &= \frac{[1 \quad 0]^T}{\sqrt{1^2 + 0^2}} = [1 \quad 0]^T, \\ \mathbf{v}_2 &= \frac{[-1 \quad 1]^T}{\sqrt{1^2 + 1^2}} = \left[-1/\sqrt{2} \quad 1/\sqrt{2}\right]^T. \end{aligned}$$

We can then form the matrix P by stacking the eigenvectors as columns:

$$P = \begin{bmatrix} 1 & -1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix}.$$

We can also form the diagonal matrix D by putting the eigenvalues along the diagonal:

$$D = \begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}.$$

Finally, we can verify that $A = PDP^{-1}$:

$$PDP^{-1} = \begin{bmatrix} 1 & -1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{bmatrix}\begin{bmatrix} 3 & 0 \\ 0 & 2 \end{bmatrix}\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 0 & 2 \end{bmatrix},$$

which is equal to A. So we have successfully diagonalized A.

As we observed in this example, ChatGPT came up with an excellent diagonalization example with a step-by-step explanation that can help readers better understand the proper algorithm. Although the calculation for finding p^{-1} was not shown, the machine assumed that the readers have working knowledge of it, particularly for an easy, simple 2 × 2 matrix where the computation is relatively straightforward. As some examples in the following subsection illustrate, ChatGPT encountered difficulty in (orthogonally) diagonalizing a larger matrix size, such as a 3 × 3 matrix.

4.2. Orthogonal diagonalization in ChatGPT

When we asked ChatGPT to produce an example of a diagonalization of a 3 × 3 symmetric matrix, it gave the following response:

Sure, let's consider the symmetric matrix:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 9 \end{bmatrix}.$$

To diagonalize this matrix, we need to find a matrix P and a diagonal matrix D such that:

$$A = PDP^T,$$

where P^T denotes the transpose of P. First, we need to find the eigenvalues of A by solving the characteristic equation:

$$\det(A - \lambda I) = 0,$$

where λ is the eigenvalue and I is the identity matrix. This gives us:

$$\begin{aligned} &\det \begin{bmatrix} 1-\lambda & 2 & 3 \\ 2 & 5-\lambda & 6 \\ 3 & 6 & 9-\lambda \end{bmatrix} = 0 \\ &= (1-\lambda)\left[(5-\lambda)(9-\lambda) - 6 * 6\right] - 2\left[2 * (9-\lambda) - 6 * 3\right] + 3\left[2 * 6 - \underline{\mathbf{5}} * 3\right] \\ &= (1-\lambda)\left(\lambda^2 - \underline{\mathbf{15}}\lambda + \underline{\mathbf{36}}\right) + 2\underline{(\mathbf{3}\lambda - \mathbf{12})} - 3\underline{(\mathbf{6} - \mathbf{15})} \\ &= \lambda^3 - 15\lambda^2 + \underline{\mathbf{54}\lambda - \mathbf{36}} \end{aligned}$$

We observed that ChatGPT made calculation mistakes starting with the second expression, where it wrote 5 instead of $5 - \lambda$. The subsequent terms, including the final expression for the characteristic equation, are undoubtedly incorrect. These are represented by the underlined boldface terms. The correct characteristic equation is $\lambda^3 - 15\lambda^2 + 10\lambda = 0$. Interestingly, when we asked ChatGPT to orthogonally diagonalize the given 3 × 3 symmetric matrix A or find its characteristic equation, it still produced an incorrect answer although it differed from the original version. Both queries yielded $\lambda^3 - 15\lambda^2 + \underline{\mathbf{16}\lambda - \mathbf{33}} = 0$. In all three cases, both the cubic and quadratic terms were correct, whereas both the linear and constant terms were incorrect, as indicated by the underlined boldface terms.

We continue with what ChatGPT offered after this computational step.

This polynomial has a repeated root at $\lambda = 0$, and distinct roots at $\lambda = 3$ and $\lambda = 12$. Therefore, the eigenvalues of A are $\lambda_1 = 0$, $\lambda_2 = 3$, and $\lambda_3 = 12$. Next, we need to find the eigenvectors of A for each eigenvalue. For $\lambda_1 = 0$, we have:

$$A - \lambda_1 I = A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 6 \\ 3 & 6 & 9 \end{bmatrix}.$$

Reducing this matrix to row echelon form, we get:

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 1 & \underline{\mathbf{1}} \\ 0 & 0 & 0 \end{bmatrix},$$

which gives us the eigenvector $v_1 = [-2,\ 1,\ 0]$. For $\lambda_2 = 3$, we have:

$$A - \lambda_2 I = \begin{bmatrix} -2 & 2 & 3 \\ 2 & 2 & 6 \\ 3 & 6 & 6 \end{bmatrix}.$$

Reducing this matrix to row echelon form, we get:

$$\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 0 \end{bmatrix},$$

which gives us the eigenvector $v_2 = [-1,\ -2,\ 1]$. For $\lambda_3 = 12$, we have:

$$A - \lambda_3 I = \begin{bmatrix} -11 & 2 & 3 \\ 2 & -7 & 6 \\ 3 & 6 & -3 \end{bmatrix}.$$

Reducing this matrix to row echelon form, we get:

$$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -2 \\ 0 & 0 & 0 \end{bmatrix},$$

which gives us the eigenvector $v_3 = [1,\ 2,\ \underline{\mathbf{1}}]$.

ChatGPT stopped at this stage, even without completing the final component for the third eigenvector, as indicated by the red color. In other instances, ChatGPT could continue the process by constructing a diagonal matrix D, an invertible matrix P, and finalizing the answer with "Finally, we can check that $A = PDP^{-1}$." However, this did not occur in this example.

Interestingly, however, ChatGPT obtained one correct eigenvalue, that is, $\lambda_1 = 0$, although one entry in the row echelon form matrix was incorrect, that is, $a_{23} = 0$ instead of 1. The correct associated eigenvector is $\mathbf{v}_1 = (3,\ 0,\ -1)$. Furthermore, contrary to other positive integer eigenvalues obtained by ChatGPT, λ_2 and λ_3 of matrix A in this example are not integers, but irrational numbers instead, although both are still positive. Arranged ascendingly, they are

$$\lambda_2 = \frac{1}{2}\left(15 - \sqrt{185}\right), \qquad \text{and} \qquad \lambda_3 = \frac{1}{2}\left(15 + \sqrt{185}\right).$$

A possible choice for the corresponding eigenvectors would be

$$\mathbf{v}_2 = \left(\frac{1}{3}, \frac{2\left(-15 + \sqrt{185}\right)}{3\left(-13 + \sqrt{185}\right)}, 1\right), \qquad \text{and} \qquad \mathbf{v}_3 = \left(\frac{1}{3}, \frac{2\left(15 + \sqrt{185}\right)}{3\left(13 + \sqrt{185}\right)}, 1\right).$$

If we use the eigenvalues and associated eigenvectors delivered by ChatGPT to construct the original matrix A, the resulting matrix is neither symmetric nor has all integer entries. Instead, we obtain the following matrix:

$$PDP^{-1} = \frac{1}{10}\begin{bmatrix} 15 & 30 & 45 \\ 30 & 60 & 90 \\ 9 & 18 & 75 \end{bmatrix} = \begin{bmatrix} \frac{3}{2} & 3 & \frac{9}{2} \\ 3 & 6 & 9 \\ \frac{9}{10} & \frac{9}{5} & \frac{15}{2} \end{bmatrix}.$$

To further test its capability, we investigated ChatGPT to orthogonally diagonalize a particular 3×3 symmetric matrix. This is the same matrix B that we explored in Subsection 2.2:

$$B = \begin{bmatrix} 3 & -4 & -4 \\ -4 & 3 & -4 \\ -4 & -4 & 3 \end{bmatrix}.$$

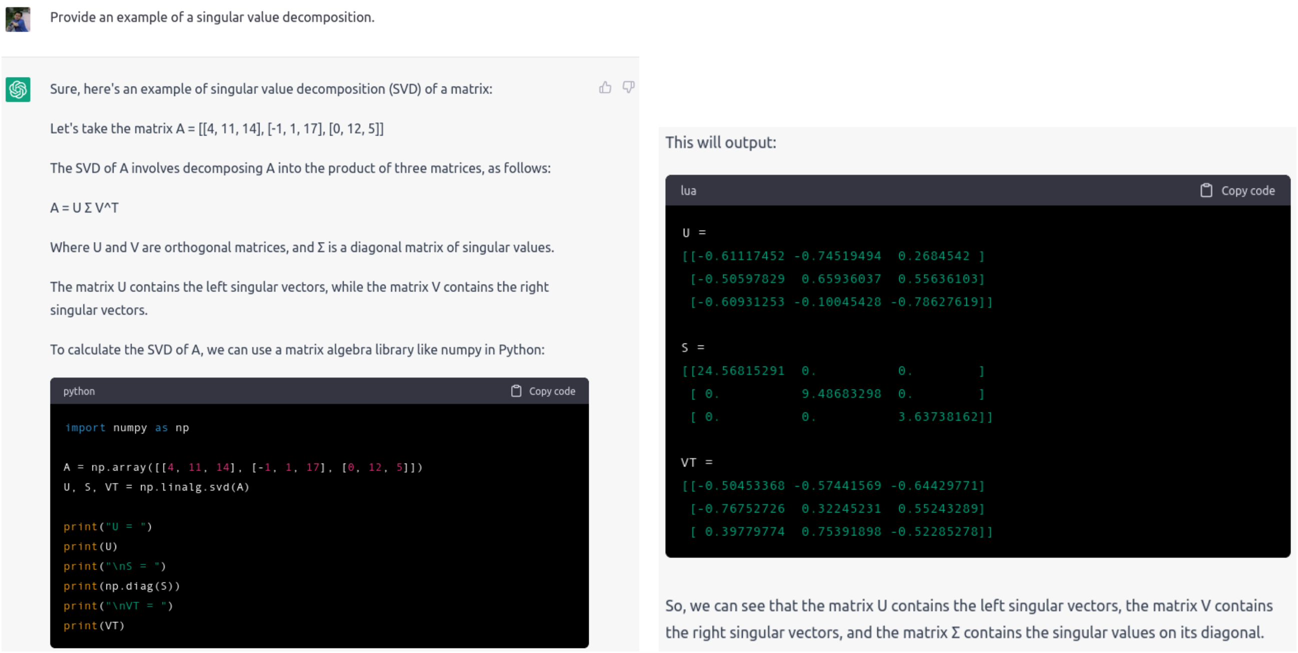
Unfortunately, ChatGPT provided an incorrect answer for this matrix diagonalization. Its computation for the characteristic polynomial was erroneous: $\lambda^3 - 9\lambda^2 + 24\lambda$, which results in a bogus set of eigenvalues: $\lambda_1 = 0$, $\lambda_2 = 3$, and $\lambda_3 = 6$. The calculation was continued to seek the corresponding eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$, but ChatGPT was unable to complete the calculation for $\mathbf{v}_3$. The computation stopped abruptly with some missing entries in the matrix equation. By further analyzing each step, we discovered that neither the row-reduction process of the augmented matrix for the first and second eigenvalues nor the obtained eigenvector was correct. Requesting ChatGPT to "regenerate response" was also ineffective because it once again produced inaccurate and partial replies. From these examples, we observe that the current version of ChatGPT does not seem ready for the (orthogonal) diagonalization of a simple matrix beyond a size of 2×2.

4.3. SVD in ChatGPT

We asked ChatGPT to "provide an example of a singular value decomposition." Interestingly, ChatGPT provided a relatively advanced example of a 3×3 matrix and utilized the matrix algebra library `numpy` in Python, which SageMath also employs for as its primary language solver. The chosen matrix is given by

$$A = \begin{bmatrix} 4 & 11 & 14 \\ -1 & 1 & 17 \\ 0 & 12 & 5 \end{bmatrix}.$$

In the previous two cases, ChatGPT explained the algorithm to perform an (orthogonal) diagonalization of a matrix. In this case, it directly provided a Python code and its corresponding output. Furthermore, although the original matrix contains integer entries, the matrices in its SVD consist of real numbers, which can be difficult to check manually.



Provide an example of a singular value decomposition.

Sure, here's an example of singular value decomposition (SVD) of a matrix:

Let's take the matrix A = [[4, 11, 14], [-1, 1, 17], [0, 12, 5]]

The SVD of A involves decomposing A into the product of three matrices, as follows:

A = U Σ V^T

Where U and V are orthogonal matrices, and Σ is a diagonal matrix of singular values.

The matrix U contains the left singular vectors, while the matrix V contains the right singular vectors.

To calculate the SVD of A, we can use a matrix algebra library like numpy in Python:

```python
import numpy as np

A = np.array([[4, 11, 14], [-1, 1, 17], [0, 12, 5]])
U, S, VT = np.linalg.svd(A)

print("U = ")
print(U)
print("\nS = ")
print(np.diag(S))
print("\nVT = ")
print(VT)
```

This will output:

```lua
U =
[[-0.61117452 -0.74519494  0.2684542 ]
 [-0.50597829  0.65936037  0.55636103]
 [-0.60931253 -0.10045428 -0.78627619]]

S =
[[24.56815291  0.          0.        ]
 [ 0.          9.48683298  0.        ]
 [ 0.          0.          3.63738162]]

VT =
[[-0.50453368 -0.57441569 -0.64429771]
 [-0.76752726  0.32245231  0.55243289]
 [ 0.39779774  0.75391898 -0.52285278]]
```

So, we can see that the matrix U contains the left singular vectors, the matrix V contains the right singular vectors, and the matrix Σ contains the singular values on its diagonal.

Figure 1. An output of ChatGPT to the inquiry "Provide an example of a singular value decomposition."

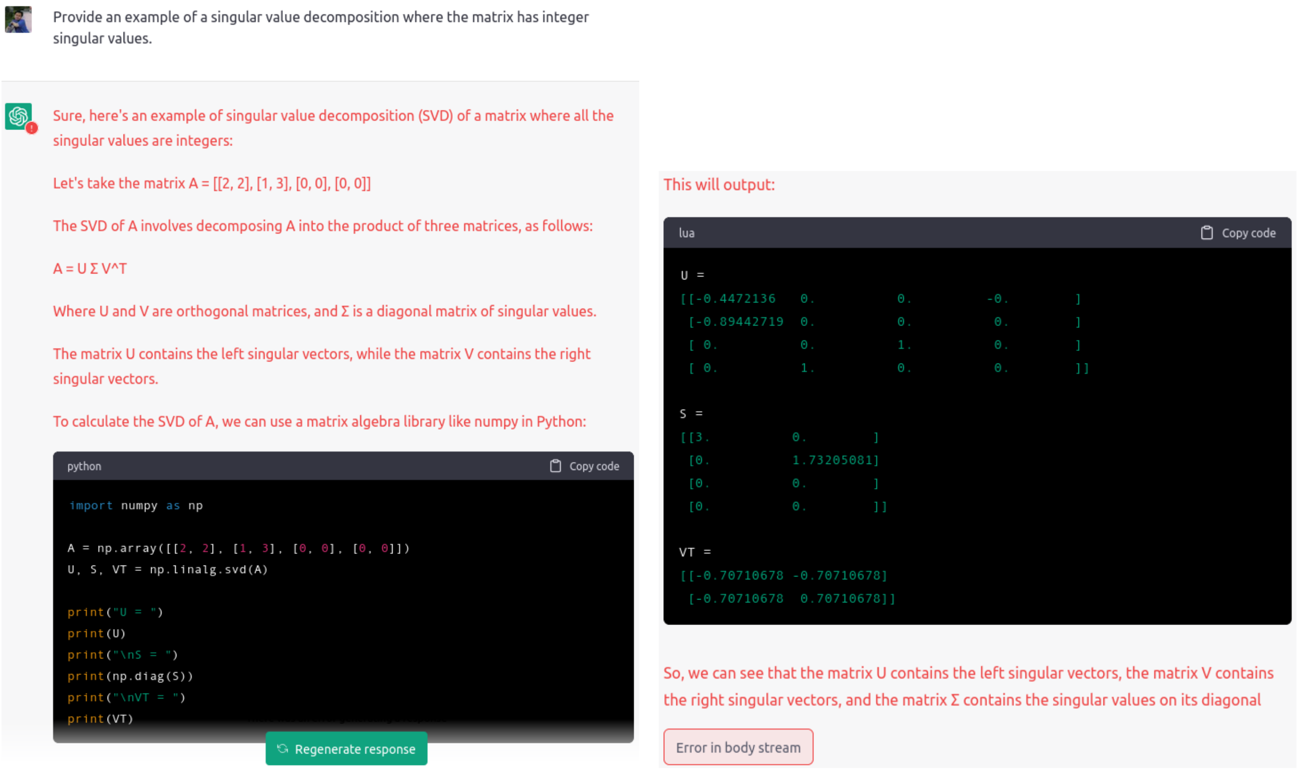
Unfortunately, ChatGPT got it wrong for its own given example, as shown in Figure 1. It dispensed singular values of $\sigma_1 = 24.57$, $\sigma_2 = 9.49$, and $\sigma_3 = 3.63$, whereas the correct singular values are $\sigma_1 = 25.58$, $\sigma_2 = 11.36$, and $\sigma_3 = 3.13$. Although the error for the first singular value was less than 4%, the errors for the second and third singular values were approximately 16%, which is too large to be considered acceptable. Using the SageMath commands `A=matrix(RDF,[[4,11,14],[-1,1,17],[0,12,5]])` and `A.SVD()`, we obtained the following output:

```
(
[-0.7020788202506024 0.21713691817311598 -0.6781864706124737]
[-0.5829773893610252 -0.7221727564849129  0.3722954113130941]
[-0.4089287146304667  0.6567481012970376  0.6336081105805151],

[25.575254913906992                0.0                0.0]
[               0.0 11.363233131054077                0.0]
[               0.0                0.0 3.1278217497152005],

[-0.08701136700816689  0.13998836518016763  -0.9863226042353253]
[ -0.5166317611363659   0.8401931429966347  0.16482446980522636]
[ -0.8517749969308255  -0.5239071865088518 0.0007839182813947777]
),
```

where the first, second, and third matrices correspond to U, Σ, and V^T, respectively. Here, RDF stands for Real Double Field, which is an approximation to a real number using double-precision floating point numbers.



Provide an example of a singular value decomposition where the matrix has integer singular values.

Sure, here's an example of singular value decomposition (SVD) of a matrix where all the singular values are integers:

Let's take the matrix A = [[2, 2], [1, 3], [0, 0], [0, 0]]

The SVD of A involves decomposing A into the product of three matrices, as follows:

A = U Σ V^T

Where U and V are orthogonal matrices, and Σ is a diagonal matrix of singular values.

The matrix U contains the left singular vectors, while the matrix V contains the right singular vectors.

To calculate the SVD of A, we can use a matrix algebra library like numpy in Python:

```python
import numpy as np

A = np.array([[2, 2], [1, 3], [0, 0], [0, 0]])
U, S, VT = np.linalg.svd(A)

print("U = ")
print(U)
print("\nS = ")
print(np.diag(S))
print("\nVT = ")
print(VT)
```

This will output:

```lua
U =
[[-0.4472136   0.          0.         -0.        ]
 [-0.89442719  0.          0.          0.        ]
 [ 0.          0.          1.          0.        ]
 [ 0.          1.          0.          0.        ]]

S =
[[3.          0.        ]
 [0.          1.73205081]
 [0.          0.        ]
 [0.          0.        ]]

VT =
[[-0.70710678 -0.70710678]
 [-0.70710678  0.70710678]]
```

So, we can see that the matrix U contains the left singular vectors, the matrix V contains the right singular vectors, and the matrix Σ contains the singular values on its diagonal

Error in body stream

Figure 2. An output of ChatGPT to the inquiry "Provide an example of a singular value decomposition where the matrix has integer singular values."

We further asked ChatGPT to "provide an example of an SVD where the matrix has integer singular values." ChatGPT responded with the same wording but chose a different matrix, and this time it was a 4 × 2 matrix, given as follows:

$$A = \begin{bmatrix} 2 & 2 \\ 1 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

However, the result was still incorrect and the computation stopped nearly at the end of the answer with the message "Error in body stream" and an offer to "Regenerate response." See Figure 2. Requesting another response multiple times did not yield any results. For the given matrix, the correct singular values are not integers, although they can be expressed analytically, that is, $\sigma_{1,2} = \sqrt{9 \pm \sqrt{65}} \approx 4.13$ and 0.97, respectively.

Using a new chatbot and asking for an identical inquiry, we obtained an improved result, even though there were still some conceptual mistakes in the answer. See Figure 3. Another 3 × 3 matrix was selected, but this time it was symmetric:

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 3 \end{bmatrix}.$$

Instead of calculating singular values, ChatGPT computed the eigenvalues of matrix A. Coincidentally, they are identical for this particular example, that is, $\lambda_1 = 5 = \sigma_1$, $\lambda_2 = \lambda_3 = 2 = \sigma_2 = \sigma_3$. This is a rather rare finding, as it occurs only for positive-definite and symmetric (or Hermitian, for complex-valued) matrices. Even though matrices U and V appeared to be correct, matrix Σ did not because ChatGPT displayed the square root of the eigenvalues instead of correctly calculating the singular values.

5. Applications

There are many applications of (orthogonal) diagonalizations and SVD in various scientific, engineering, and computational fields. Their significance provides valuable tools for solving complex problems and understanding the underlying structures of various systems. In this section, we only provide brief coverage of applications in life sciences and engineering.

5.1. Applications in life sciences

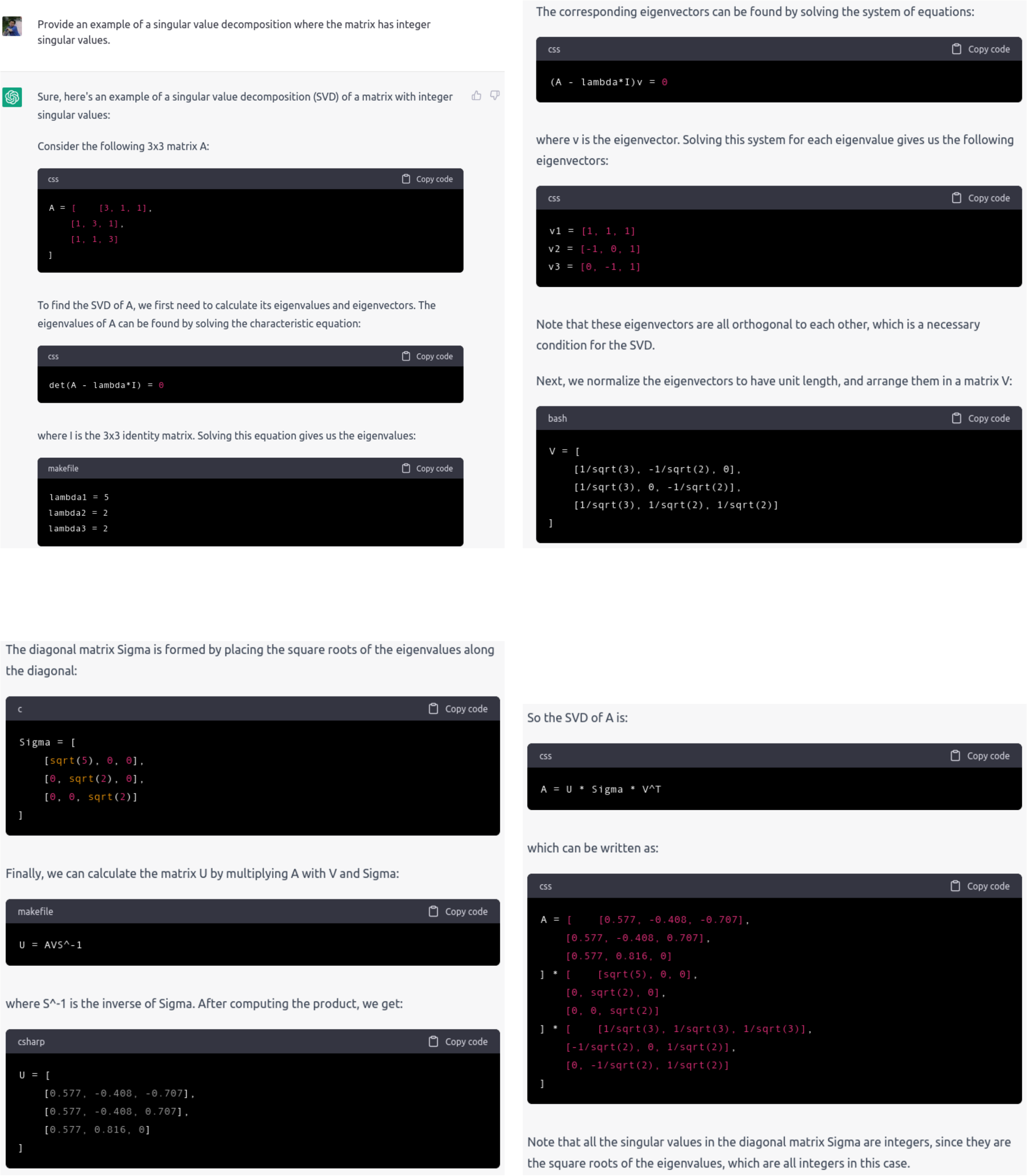
The study of eigenvalues, eigenvectors, and (orthogonal) diagonalization has useful applications in the solutions of systems of ordinary differential equations as well as in discrete dynamical systems, for which such models appear not only in biology but also other branches of science and engineering. In particular, in the field of population ecology and dynamics, with applications such as the predator-prey system [17] and competing species model [100], mathematical modeling often involves the use of matrices [101].

When solving a coupled system of differential equations, diagonalization allows for the transformation of a system of linear differential equations into a set of decoupled equations, which are easier to solve. Observe that the system $\mathbf{x}' = A\mathbf{x}$ admits the solution $\mathbf{x}(t) = \mathbf{x}(0)\, e^{A\,t}$, where $e^{A\,t}$ is the matrix exponential. If A is diagonalizable, that is $A = PDP^{-1}$, then the matrix exponential can be calculated according to

$$e^{A\,t} = P\, e^{D\,t}\, P^{-1},$$

where $e^{D\,t}$ is a diagonal matrix with entries $e^{\lambda_i t}$, $i = 1, 2, \cdots, n$, on its diagonal [102].

Matrix diagonalization also finds applications for investigating sex-linked genes, such as recessive color blindness, for which it would be easier to describe the situation mathematically using a matrix model. One such model that groups the population based on sex, that is males and females, can be described by $\mathbf{x}^{(n)} = A^n\, \mathbf{x}^{(0)}$, where $\mathbf{x}^{(n)}$ denotes the proportion of color blindness-related genes in the male and female populations of the $(n + 1)$st generation, and A denotes a coefficient matrix related to the model. To find the proportions of genes for color blindness in a particular population for multiple generations, we need to calculate the limit of $\mathbf{x}^{(n)}$, which also requires calculating the limit of the power matrix on the right-hand side, that is, A^n, which will be easier if we know an expression for its diagonalization [103].



Provide an example of a singular value decomposition where the matrix has integer singular values.

Sure, here's an example of a singular value decomposition (SVD) of a matrix with integer singular values:

Consider the following 3x3 matrix A:

```
A = [    [3, 1, 1],
    [1, 3, 1],
    [1, 1, 3]
]
```

To find the SVD of A, we first need to calculate its eigenvalues and eigenvectors. The eigenvalues of A can be found by solving the characteristic equation:

```
det(A - lambda*I) = 0
```

where I is the 3x3 identity matrix. Solving this equation gives us the eigenvalues:

```
lambda1 = 5
lambda2 = 2
lambda3 = 2
```

The corresponding eigenvectors can be found by solving the system of equations:

```
(A - lambda*I)v = 0
```

where v is the eigenvector. Solving this system for each eigenvalue gives us the following eigenvectors:

```
v1 = [1, 1, 1]
v2 = [-1, 0, 1]
v3 = [0, -1, 1]
```

Note that these eigenvectors are all orthogonal to each other, which is a necessary condition for the SVD.

Next, we normalize the eigenvectors to have unit length, and arrange them in a matrix V:

```
V = [
    [1/sqrt(3), -1/sqrt(2), 0],
    [1/sqrt(3), 0, -1/sqrt(2)],
    [1/sqrt(3), 1/sqrt(2), 1/sqrt(2)]
]
```

The diagonal matrix Sigma is formed by placing the square roots of the eigenvalues along the diagonal:

```
Sigma = [
    [sqrt(5), 0, 0],
    [0, sqrt(2), 0],
    [0, 0, sqrt(2)]
]
```

Finally, we can calculate the matrix U by multiplying A with V and Sigma:

```
U = AVS^-1
```

where S^-1 is the inverse of Sigma. After computing the product, we get:

```
U = [
    [0.577, -0.408, -0.707],
    [0.577, -0.408, 0.707],
    [0.577, 0.816, 0]
]
```

So the SVD of A is:

```
A = U * Sigma * V^T
```

which can be written as:

```
A = [    [0.577, -0.408, -0.707],
     [0.577, -0.408, 0.707],
     [0.577, 0.816, 0]
] * [    [sqrt(5), 0, 0],
     [0, sqrt(2), 0],
     [0, 0, sqrt(2)]
] * [    [1/sqrt(3), 1/sqrt(3), 1/sqrt(3)],
     [-1/sqrt(2), 0, 1/sqrt(2)],
     [0, -1/sqrt(2), 1/sqrt(2)]
]
```

Note that all the singular values in the diagonal matrix Sigma are integers, since they are the square roots of the eigenvalues, which are all integers in this case.

Figure 3. An output of ChatGPT to the same inquiry as in Figure 2. The reading order starts from the top left, then moves to the top right, followed by the bottom left, and finally, the bottom right.

Diagonalization and SVD play significant roles in biomedical signal processing by enabling dimensionality reduction, noise reduction, feature extraction, and improvement of the signal analysis and interpretation. Diagonalization techniques, such as principal component analysis (PCA), are widely employed to process and analyze biomedical signals, such as EEG (electroencephalogram), ECG (electrocardiogram), and fMRI (functional magnetic resonance imaging) data [104–107]. By extracting meaningful features and reducing noise, these methods significantly contribute to diagnosis and our comprehension of brain activity and cardiac patterns. Through these applications, diagonalization plays a vital part in advancing our understanding of biological processes, disease diagnosis, and the development of personalized healthcare solutions.

In the field of evolutionary biology, two crucial symmetric matrices underlie the understanding of microevolutionary change. The first matrix describes the individual fitness surface, referred to as the matrix of nonlinear selection gradients γ. The second matrix affects a multivariate response to selection, known as the genetic variance-covariance matrix G. By applying (orthogonal) diagonalization to both γ and G, biologists not only gain deeper insights into the form and strength of nonlinear selection, but they also assess the availability of genetic variance for multiple traits. This powerful technique enhances our understanding of evolutionary processes, shedding light on the interplay between selection pressures and genetic variation in shaping the traits of organisms [108].

Eigenanalysis and SVD find valuable applications in population genetics, enabling the study of genetic variation and evolutionary processes within populations. SVD, in particular, plays a prominent role in analyzing high-dimensional biological data, such as gene expression profiles, protein-protein interaction networks, and genomic sequencing data. By identifying significant patterns, correlations, and reducing noise, SVD helps extract biologically relevant information from intricate datasets. Additionally, SVD aids in identifying population substructure and detecting signatures of natural selection, further enhancing our understanding of genetic diversity and evolutionary dynamics. Moreover, in the realm of data visualization, SVD proves particularly advantageous in simplifying data complexity and fostering a clearer visualization of biological datasets. This capability empowers researchers to gain valuable insights into the intricate relationships and processes underlying complex biological systems [109–111].

In summary, both (orthogonal) diagonalization and SVD are powerful and versatile tools with diverse applications in biology and life sciences. Their capacity to analyze, integrate, and extract crucial information from complex datasets plays a pivotal role in advancing our understanding of biological systems, disease mechanisms, and personalized healthcare solutions. These techniques are indispensable in the field of biosciences, making significant contributions to our knowledge and applications across various areas of biology and life sciences. As researchers continue to harness their potential, the impact of orthogonal diagonalization and SVD is likely to extend even further, driving innovation and discoveries in the quest to unravel the complexities of life.

5.2. Applications in engineering

Diagonalization and SVD play a pivotal role in electrical engineering, particularly in power system analysis, load flow calculations, and fault diagnosis. These techniques empower engineers to efficiently study and optimize the behavior of electrical power systems, allowing them to gain a comprehensive understanding of power grid dynamics and optimize power distribution. By offering valuable insights into stability, dynamic responses, and control strategies, diagonalization and SVD serve as versatile tools that significantly enhance power system analysis. Moreover, they contribute to ensuring the reliable, secure, and efficient operation of electrical power systems, facilitating the seamless integration of renewable energy sources and driving advancements in smart grid technologies. The integration of diagonalization and SVD in electrical engineering proves essential for promoting sustainable and resilient power systems, which will foster a more efficient and eco-friendly energy landscape [112–116].

In mechanical engineering, (orthogonal) diagonalization and SVD have strong applications, particularly in mechanical and structural analysis. Diagonalization techniques are employed to analyze the dynamic behavior of mechanical systems, enabling engineers to analyze the vibration modes, modal frequencies, and dynamic responses of such systems. It facilitates understanding the system behavior, structural integrity, structural design optimization, stability, and performance of mechanical components and systems. SVD finds application in structural analysis for the dimensionality reduction of large-scale data, allowing engineers to extract dominant modes and efficiently study complex structures. Additionally, SVD is used in model reduction to create reduced-order models, facilitating faster and more accurate simulations of mechanical systems. These techniques contribute significantly to enhance mechanical engineering design, optimization, and reliability, paving the way for innovative and efficient mechanical systems [117–119].

(Orthogonal) diagonalization and SVD play a crucial role in control engineering, offering powerful tools to analyze the dynamic behavior of linear systems and design effective controllers. Specifically, diagonalization serves as a key technique for stability analysis in linear control systems. By transforming the system's transfer function into a diagonal form, engineers gain valuable insights into stability and response characteristics, facilitating the design and optimization of control systems. Through diagonalization, engineers can easily determine the system's stability based on the eigenvalues of the diagonal matrix. Moreover, both diagonalization and SVD are indispensable in controller design. By diagonalizing the system's state-space representation, engineers can thoroughly understand the system's dynamics and efficiently optimize controller parameters. This enables the design of feedback controllers that stabilize the system and meet stringent performance specifications, leading to robust and reliable control solutions [120–125].

(Orthogonal) diagonalization and SVD find diverse and valuable applications in image and signal processing, as we have seen in Subsection 5.1. In image processing, diagonalization techniques are utilized for dimensionality reduction, denoising, and feature extraction. By transforming image data into their principal components, diagonalization enables efficient compression and denoising while preserving essential image features. SVD plays a pivotal role in image compression and data analysis, allowing for efficient representation of images with reduced storage requirements. Moreover, SVD is employed in image and signal denoising, where it separates noise from the underlying signal, improving the quality and clarity of images and signals. Additionally, SVD is used in feature extraction, pattern recognition, and image registration, aiding in tasks such as object detection, face recognition, and medical image alignment. These techniques have proven indispensable in various image and signal processing applications, contributing to advancements in computer vision, medical imaging, multimedia processing, and many other fields [126–133].

Linear algebra serves as a vital bridge between theoretical mathematical concepts and their practical applications, particularly in data science and machine learning. A solid understanding of this subject opens doors to comprehending complex machine learning algorithms more effectively. SVD stands out as a prominent technique commonly used for dimensional reduction in data science. Moreover, SVD finds extensive use in matrix factorization and data compression. In collaborative filtering and recommendation systems, SVD uncovers latent features and predicts missing values in sparse datasets, while also contributing to techniques like latent semantic analysis for natural language processing tasks. These techniques greatly aid in data preprocessing, enhancing the performance, interpretability, and scalability of machine learning models, thus making them invaluable tools in diverse applications. Similarly, diagonalization finds application in various machine learning algorithms, enabling PCA for dimensionality reduction. By transforming data into principal components, diagonalization enables efficient data representation and visualization, preserving vital information while reducing computational complexity. Together, these linear algebra techniques play pivotal roles in the success of data science and machine learning, empowering researchers and practitioners to tackle complex real-world problems with greater clarity and efficiency [134–142].

In summary, mastering (orthogonal) diagonalization and SVD provides students with immense benefits, enabling efficient problem-solving in diverse science and engineering domains. These techniques enhance computational efficiency, reduce data complexity, and provide deeper insights into various scientific and engineering phenomena. By harnessing their power, innovative solutions are unlocked, driving advancements and revolutionizing multiple disciplines. Indeed, (orthogonal) diagonalization and SVD continue to inspire innovation, deepen our understanding of complex phenomena, and pave the way for future advancements in science and engineering.

6. Discussion and conclusion

This study admits several limitations. First, we only focused on the topics of (orthogonal) diagonalization and SVD, whereas algorithmic thinking in matrix factorization encompasses a broader range of topics. For example, we did not consider decompositions related to solving systems of linear equations, such as LU, QR, and Cholesky decompositions. Other eigenvalue-based decompositions are related to our study, but we did not cover them in this article; they include Jordan, Schur, Takagi, and QZ decompositions, among others. Second, we particularly selected SageMath to obtain a better understanding of matrix factorization instead of another CAS. The main reason is that it is a free and open-source CAS, and as stated on its website, its mission is to "create a viable free open source alternative to Magma, Maple, Mathematica, and Matlab" [143]. Additionally, we can perform our computation directly on the server without the hassle of installing the program on our computer, as long as we have an internet connection.

Third, the outcome that ChatGPT presented disappointingly contains numerous mistakes. On the one hand, this can be confusing for many students who are academically weak and just willing to accept what is handed out by the chatbot. On the other hand, this can be an excellent opportunity for further discussion on cultivating critical thinking among learners. Additionally, because our study was conducted at the initial stage of ChatGPT development, the responses considered in this discussion might no longer be relevant in the future, thus making it challenging for other researchers to replicate a similar study. However, we are aware that updated versions will be released in the future and the responses will be improved accordingly. (In fact, at the time of this writing, ChatGPT Plus subscribers can already enjoy the more advanced version of the chatbot, that is the GPT-4 model.) Thus, we can be confident that incorrect answers related to matrix factorization in particular or other mathematics problems in general will eventually be minimized [94].

Many mathematics educators and linear algebra instructors have likely observed that while some students grasp the subject with ease, others struggle to establish connections between different concepts, leading to a sense of confusion and disorientation. This feature makes teaching linear algebra challenging for undergraduate students. To investigate the first research question, we identified some common mistakes and difficulties that students encountered in particular topics of linear algebra, that is, (orthogonal) diagonalization and SVD. Indeed, these topics can be challenging for many learners because they require previous, more basic knowledge of linear algebra, such as proficiency in performing elementary row operations and finding orthogonal projections. At the same time, a lack of understanding of matrix diagonalization will lead to more difficulties in the subsequent topics, such as orthogonal diagonalization and SVD, as well as other topics related to diagonalization, which are usually not covered in a standard linear algebra course but have some applications in data science or machine learning.

Learners of linear algebra should be aware of the fact that not all square matrices can be diagonalized. Having a misconception that every matrix can be diagonalized can lead to confusion in subsequent topics. To master computational thinking in the area of solving diagonalization problems, in addition to understanding an algorithmic procedure to diagonalize a particular matrix, students should have an intuitive feeling for less concrete objects, such as eigenvalues and their associated eigenvectors. This also means that they should know how to find the characteristic equation of a matrix, compute the eigenvalues, and find the corresponding eigenvectors via elementary row operations. Learners who struggle with row reduction usually do not go far from figuring out how to express eigenvectors in a simple way. Certainly, those who struggle with this more basic matrix algebra should practice sufficiently until they are comfortable with the computational process.

Another difficulty arises when students learn orthogonal diagonalization for symmetric matrices. Although the procedure is similar to previously learned matrix diagonalization, some students often forget to normalize the obtained eigenvectors and check that these eigenvectors must be orthonormal to each other. The absence of this step and simply establishing that $P^{-1} = P^T$ will not yield a correct orthogonal diagonalization, that is, PDP^T will not return to the original matrix A. Further difficulty will occur when one needs to construct a set of orthogonal eigenvectors, whereas the resulting eigenvectors are not orthogonal because they arise from the same eigenspace. Connecting to another concept of vector projection and an orthogonalization procedure of the Gram-Schmidt algorithm requires a solid understanding of related concepts, in addition to proficiency in the calculation of the orthogonal projection itself. By becoming competent in these computational thinking skills, any learner in the field of linear algebra will be able to solve (orthogonal) diagonalization problems.

When learning SVD, a common mistake is the confusion between singular values and eigenvalues. Even if learners are aware of the definition of singular values as the square root of eigenvalues, many of them often forget that they should compute the eigenvalues of the symmetric matrix $M^T M$ instead of the original matrix M. A small mistake in this early step of the algorithm will not lead to a correct SVD, even though they understand the procedure—and implement it correctly—required to calculate an SVD of a matrix. Another difficulty is in constructing matrix Σ, particularly when M is neither square nor possesses a full rank. Indeed, we must correctly determine the number of blocks of zeros that we should embed along the diagonal matrix D to construct Σ. An easy rule to remember is that

Σ must be the same size as M. If M has more rows than columns, the square orthogonal matrix U will be larger than the square orthogonal matrix V, and vice versa. A challenge in constructing U occurs when either M has a deficient rank or $m > n$, where m and n denote the number of rows and columns, respectively. When we attempted to express a decomposition in the absence of the missing eigenvector, we obtained a reduced SVD instead of a full, non-reduced one. Solid knowledge of both inner (dot) and cross products, as well as not forgetting to normalize the vectors, will be sufficient to master algorithmic thinking skills when attempting to find an SVD of a matrix.

This leads us to discuss the second research question, that is, how can a static CAS such as SageMath assist learners in mastering topics in linear algebra that require algorithmic thinking skills, particularly (orthogonal) diagonalization and SVD? Using a CAS for assisting computational processes, particularly in linear algebra, can be a tremendous help for many of us, including both learners and teachers. SageMath in particular, being more powerful than a graphic calculator, is easier to use owing to its offer web browser cell access to users with an internet connection. Similar to other symbolic computation-type CASs, SageMath can assist many of us in solving many mathematical problems, aiding us to investigate whether our calculations done by pen and paper are correct, which would save a lot of time on lengthy and complex calculations that are otherwise too difficult or time-consuming to perform by hand.

As we have shown in Section 3, despite its static nature, SageMath can assist us in finding the (orthogonal) diagonalization and SVD of a given matrix. Calculating the characteristic polynomial, eigenvalues, and eigenvectors is straightforward and quick. This is helpful for a larger-sized matrix, where obtaining eigenvalues and eigenvectors may take a significant amount of time. Without worrying about this step, learners in the subject of linear algebra can directly focus on the main business of constructing matrices for (orthogonal) diagonalization and SVD. Once we have all of the matrices, SageMath can help us to quickly determine whether those matrices are correct by multiplying the products in the (orthogonal) diagonalization or SVD. Any correct factorization that does not have to be unique will return the product to the original matrix. The static nature of SageMath also demands that learners carefully observe outputs. For example, the command `A.eigenvectors_right()` gives an output comprising of an eigenvalue, eigenvector(s), and its geometric multiplicity in that particular order. Constructing a diagonal matrix D and invertible matrix P also requires hand intervention, which requires some dynamics from the users' side. This interaction between static SageMath and dynamic learners makes the CAS a powerful tool for learning mathematics. Furthermore, the interactive feature of SageMath allows users to experiment with distinct matrices, or matrices of different sizes or characteristics. Although we do not really feature the visualization tools of SageMath in this study, its ability to present plots or graphs will assist learners further in explaining more difficult concepts encountered in mathematics.

As knowledge improves and technology progresses, the presence of a static CAS will increasingly be accompanied by the emergence of dynamic chatbots that make use of the power of AI, such as ChatGPT, Google Bard, and Bing Chat. Although ChatGPT is not primarily designed for responding to inquiries related to mathematical problems, some attempts in this study and other researchers suggest that it has the capability to handle mathematical problems. It is just a matter of time before the responses will improve and get better. This triggers us to address the third and final research question, that is, whether we can rely on ChatGPT, despite its dynamic nature, in understanding linear algebra, particularly (orthogonal) diagonalization and SVD. The short answer is no. A quick and simple reason for this argument is that ChatGPT often provides an incorrect answer. In our study, except for the first inquiry, all other responses from ChatGPT contained mistakes in one way or another. If this is the case, should we simply abandon ChatGPT and other similar AI-generated chatbots when it comes to learning mathematics or understanding linear algebra, perhaps going all the way to forbidding the students to access it entirely during the course of their study? How can we strike a balance between utilizing AI-generated chatbots such as ChatGPT in learning mathematics and understanding linear algebra, while still ensuring that students develop a deep and thorough understanding of the subject matter without solely relying on the technology?

Although many would not agree, we have some reservations regarding abandoning or forbidding ChatGPT, not only in terms of learning mathematics but also for other subjects. Whether we like it or not, this "technological train" has left its compound and accelerated ever since, reaching all corners of the earth where everyone has access to the Internet. Its dynamic features have captivated many researchers. And, this is just the beginning. Since its release at the end of the last year, it has disrupted many industries, and academia and education are no exception. Thus, instead of abandoning or forbidding its use, we should embrace and integrate it into our teaching and learning. It is similar to the period when the calculator or CAS infiltrated the traditional chalk-and-board classrooms. Although in the beginning there was some resistance toward integrating technology in mathematics teaching and learning, nowadays, it is standard practice, and aspiring educators are even encouraged to do so [144–147]. After all, this is one way to connect the divide between digital natives and digital migrants [148–151].

This early stage of ChatGPT development provides a tremendous opportunity for developing algorithmic thinking skills among learners, especially when the output contains many mistakes. When inquired, instead of providing an answer in one go, ChatGPT released the answer letter by letter, literally like someone else typing out the answer from the other side of the computer. This dynamic feature makes ChatGPT superior to static CASs and attractive to many new users. In our study on matrix factorization, ChatGPT demonstrated a proper step-by-step algorithm on how to diagonalize or find an SVD of a given matrix. In the absence of specifying a particular matrix, it can even provide its own choice of matrix example, usually with a relatively small matrix size.

As observed in Section 4 on orthogonal diagonalization, ChatGPT shows how to calculate the characteristic polynomial of a matrix and solve the corresponding characteristic equation to find eigenvalues, albeit with a computational error. This provides an opportunity for learners to critically examine the provided answers and identify where mistakes occur. The same principle can also be applied to other mistakes when finding the associated eigenvectors, and when it abruptly stopped without completing the calculation, thus making the results incomplete. For the SVD, the first two attempts provided examples in which a detailed algorithm appeared to be eschewed. Instead, ChatGPT employed a matrix algebra library in Python to perform the task. It then provided the final answers of the three matrices involved in the decomposition, that is, U, Σ, and V. Our third attempt seems to be an improvement, where ChatGPT showed a step-by-step algorithm for computing singular values, finding eigenvectors, constructing an orthogonal matrix V that contains right singular vectors on its columns, forming matrix Σ, and calculating another orthogonal matrix U that contains left singular vectors on its columns. The fact that ChatGPT is confused between eigenvalues and singular values, we can glean a valuable lesson from this output because this issue is also commonly discovered among linear algebra learners, as addressed in Section 2.

From our brief coverage of the applications of (orthogonal) diagonalization and SVD, it becomes evident that students stand to gain immense benefits when they master algorithmic thinking skills and develop a solid understanding of these topics. As they delve into fields like control engineering, image processing, data science, and beyond, their proficiency in (orthogonal) diagonalization and SVD empowers them to efficiently tackle complex problems and optimize system behavior. Not only do these techniques enhance computational efficiency and reduce data complexity, they also provide deeper insights into the underlying structures of various scientific and engineering phenomena. By harnessing the power of (orthogonal) diagonalization and SVD, students and researchers alike can uncover hidden patterns, streamline analyses, and unlock innovative solutions across diverse disciplines, propelling advancements in science and engineering to new heights.

The versatility and power of (orthogonal) diagonalization and SVD have revolutionized various fields of science and engineering. Their ability to efficiently extract crucial information, reduce data dimensionality, and optimize system behavior has made them indispensable tools in linear algebra, signal processing, control engineering, data science, and more. From enabling stable control of complex systems to enhancing image and signal analysis, these techniques have proven invaluable in the advancement of research and technology. Moreover, their applications in population genetics, structural analysis, and biomedical signal processing have further extended their impact, contributing to breakthroughs in life sciences and healthcare. With the potential to solve intricate problems and reveal hidden patterns, (orthogonal) diagonalization and SVD continue to drive innovation, deepen our understanding of complex phenomena, and pave the way for future advancements in science and engineering.

In conclusion, we have considered the relationship between algorithmic thinking skills in linear algebra and some common mistakes among learners when they study particular topics such as (orthogonal) diagonalization and SVD, as well as how technological tools such as the static CAS SageMath and dynamic AI ChatGPT can contribute to enhancing algorithmic comprehension of these topics. Understanding a procedure for matrix factorization and the ability to perform the computational process accurately requires a solid understanding of other topics in linear algebra, including but not limited to solving a system of linear equations using elementary row operations, finding eigenvalues and the corresponding eigenvectors, orthogonality and vector projection (the Gram-Schmidt process), and dot (inner) and cross products.

We observed that some common mistakes that students make can be eliminated with more practice and by strengthening the basic concepts in linear algebra and their interrelationships. Although static by nature, utilizing CAS SageMath can provide tremendous help in verifying calculation results done by hand, handling a larger matrix size, checking whether the obtained factorizations were correct, and other required computational activities for understanding the materials. Its free and open-source characteristics, together with a web-based interface using cells and Python, are the primary strengths of SageMath in comparison to other costly CASs such as Maple, Matlab, and Mathematica. The current stage of ChatGPT, despite its dynamic and attractive features, can be used for complementing the study, but neither as a primary tool nor to be relied upon, owing to its numerous output mistakes. This has exciting future implications, particularly in the field of education, where updated versions of ChatGPT will be improved and released in the future. One question remains as to how to utilize a chatbot effectively for teaching and learning mathematics without compromising academic integrity.

Regarding this, we also hope that a sequence of follow-ups of this study from other researchers will appear accordingly.

Use of AI tools declaration

The author declares that he has used the AI tool of ChatGPT in the creation of this article. The tool was used to obtain responses to inquiries related to matrix (orthogonal) diagonalization and singular value decomposition, which can be found in Section 4.

Acknowledgments

This research was supported by the National Research Foundation (NRF) of South Korea, as funded by the South Korean Ministry of Science, Information, Communications, and Technology (MSICT) through Grant No. NRF-2022-R1F1A1-059817 under the scheme of Broadening Opportunities Grants–General Research Program in Basic Science and Engineering.

Conflict of interest

The author declares that he has no conflict of interest to disclose.

References

1. S. Andrilli, D. Hecker, *Elementary Linear Algebra*, Sixth edition, Academic Press, Cambridge, Massachusetts, US, 2022. https://doi.org/10.1016/C2019-0-03227-X
2. H. Anton, C. Rorres, *Elementary Linear Algebra: Applications Version*, 12th edition, John Wiley & Sons, New York, US, 2013.
3. S. Axler, *Linear Algebra Done Right*, Third edition, Springer, Berlin Heidelberg, Germany, 2015. https://doi.org/10.1007/978-3-319-11080-6
4. R. Baker, K. L. Kuttler, *Linear Algebra with Applications*, World Scientific, Singapore, 2021. https://doi.org/10.1142/9111
5. T. S. Blyth, E. F. Robertson, *Basic Linear Algebra*, Springer Science & Business Media, Berlin, Germany, 2002. https://doi.org/10.1007/978-1-4471-0681-4
6. O. Bretscher, *Elementary Linear Algebra with Applications*, Fifth edition, Pearson Education, London, England, UK, 2018.
7. S. Boyd, L. Vandenberghe, *Introduction to Applied Linear Algebra: Vectors, Matrices, and Least Squares*, Cambridge University Press, Cambridge, England, UK, 2018. https://doi.org/10.1017/9781108583664
8. S. H. Friedberg, A. J. Insel, L. E. Spence, *Linear Algebra*, Fifth edition, Pearson Education, London, England, UK, 2013.
9. R. O. Hill, *Elementary Linear Algebra*, Academic Press, Cambridge, Massachusetts, US, 2014.
10. K. Hoffman, R. Kunze, *Linear Algebra*, Second edition, Pearson Education, India, 2015.

11. L. Johnson, D. Riess, J. Arnold, *Introduction to Linear Algebra*, Fifth edition, Pearson Education, London, England, UK, 2017.
12. B. Kolman, D. Hill, *Elementary Linear Algebra with Applications*, Ninth edition, Pearson Education, London, England, UK, 2017.
13. K. L. Kuttler, *Elementary Linear Algebra*, Independently published, 2021.
14. S. Lang, *Introduction to Linear Algebra*, Second edition, Springer Science & Business Media, Berlin Heidelberg, Germany, 1997. https://doi.org/10.1007/978-1-4612-1070-2
15. R. Larson, *Elementary Linear Algebra*, Eight edition, Cengage Learning, Boston, Massachusetts, US, 2016.
16. P. D. Lax, *Linear Algebra and Its Applications*, Second edition, John Wiley & Sons, New York, US, 2007.
17. D. C. Lay, S. R. Lay, J. McDonald, *Linear Algebra and its Applications*, Sixth edition, Pearson Education, London, England, UK, 2021.
18. L. Mirsky, *An Introduction to Linear Algebra*, Dover Publications, Mineola, New York, US, 2013.
19. L. Spence, A. Insel, S. Friedberg, *Elementary Linear Algebra*, Second edition, Pearson Education, London, England, UK, 2017.
20. G. Strang, *Linear Algebra and Its Applications*, Fourth edition, Thomson, Brooks/Cole, Belmont, California, US, Cengage Learning, Boston, Massachusetts, US, 2006.
21. T. S. Barcelos, R. Muñoz-Soto, R. Villarroel, E. Merino, I. F. Silveira, Mathematics learning through computational thinking activities: A systematic literature review, *J. Universal Comput. Sci.*, **24** (2018), 815–845.
22. M. Stephens, D. M. Kadijevich, Computational/algorithmic thinking, in *Encyclopedia of Mathematics Education* (Ed., S. Lerman), Springer, Cham, Switzerland, (2020), 117–123. https://doi.org/10.1007/978-3-030-15789-0_100044
23. W. Sung, J. Ahn, J. B. Black, Introducing computational thinking to young learners: Practicing computational perspectives through embodiment in mathematics education, *Technol. Knowled. Learn.*, **22** (2017), 443–463. https://doi.org/10.1007/s10758-017-9328-x
24. D. Weintrop, E. Beheshti, M. Horn, K. Orton, K. Jona, L. Trouille, U. Wilensky, Defining computational thinking for mathematics and science classrooms, *J. Sci. Educ. Technol.*, **25** (2016), 127–147. https://doi.org/10.1007/s10956-015-9581-5
25. S. Boyd, L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, England, UK, 2004. https://doi.org/10.1017/CBO9780511804441
26. L. N. Trefethen, D. Bau, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, Pennsylvania, US, 1997. https://doi.org/10.1137/1.9780898719574
27. S. R. Bennett, *Linear Algebra for Data Science with Examples in R*, Github, San Francisco, California, US, 2021. Available from `https://shainarace.github.io/LinearAlgebra/`. Retrieved August 17, 2023.
28. M. Cohen, *Practical Linear Algebra for Data Science: From Core Concepts to Applications Using Python*, O'Reilly Media, Sebastopol, California, US, 2022.

29. G. H. Golub, C. F. Van Loan, *Matrix Computations*, John Hopkins University Press, Charles Village, Baltimore, Maryland, US, 2013. https://doi.org/10.56021/9781421407944
30. P. N. Klein, *Coding the Matrix: Linear Algebra through Applications to Computer Science*, Newtonian Press, Newton, Massachusetts, US, 2013.
31. G. Strang, *Linear Algebra and Learning from Data*, Wellesley-Cambridge Press, Wellesley, Massachusetts, US, 2019.
32. C. C. Aggarwal, *Linear Algebra and Optimization for Machine Learning: A Textbook*, Springer, Cham, Switzerland, 2020. https://doi.org/10.1007/978-3-030-40344-7
33. R. Yoshida, *Linear Algebra and Its Applications with R*, CRC Press, Boca Raton, Florida, US, 2021. https://doi.org/10.1201/9781003042259
34. G. Gadanidis, R. Cendros, L. Floyd, I, Namukasa, Computational thinking in mathematics teacher education, *Contempor. Issues Technol. Teacher Educ.*, **17** (2017), 458–477. https://doi.org/10.1163/9789004418967_008
35. A. Yadav, C. Stephenson, H. Hong, Computational thinking for teacher education, *Commun. ACM*, **60** (2017), 55–62. https://doi.org/10.1145/2994591
36. H. Abdi, Singular value decomposition (SVD) and generalized singular value decomposition (GSVD), in *Encyclopedia of Measurement and Statistics* (Ed., N. J. Salkind), Sage Publications, Thousand Oaks, California, US, (2007), 907–912.
37. A. G. Akritas, G. I. Malaschonok, Applications of singular-value decomposition (SVD), *Math. Comput. Simul.*, **67** (2004), 15–31. https://doi.org/10.1016/j.matcom.2004.05.005
38. H. Andrews, C. Patterson, Singular value decompositions and digital image processing, *IEEE Transact. Acoust. Speech Signal Process.*, **24** (1976), 26–53. https://doi.org/10.1109/TASSP.1976.1162766
39. E. Biglieri, K. Yao, K. Some properties of singular value decomposition and their applications to digital signal processing, *Signal Process.*, **18** (1989), 277–289. https://doi.org/10.1016/0165-1684(89)90039-X
40. J. Bisgard, *Analysis and Linear Algebra: The Singular Value Decomposition and Applications*, American Mathematical Society, Providence, Rhode Island, US, 2021. https://doi.org/10.1090/stml/094
41. S. L. Freire, T. J. Ulrych, Application of singular value decomposition to vertical seismic profiling, *Geophysics*, **53** (1988), 778–785. https://doi.org/10.1190/1.1442513
42. E. R. Henry, J. Hofrichter, Singular value decomposition: Application to analysis of experimental data, in *Essential Numerical Computer Methods* (Eds., L. Brand, M. L. Johnson), volume 210 of Methods in Enzymology, Academic Press, Burlington, Massachusetts, US, (1992), 129–192. https://doi.org/10.1016/0076-6879(92)10010-B
43. V. Klema, A. Laub, A. The singular value decomposition: Its computation and some applications, *IEEE Transact. Autom. Control*, **25**(1980), 164–176. https://doi.org/10.1109/TAC.1980.1102314
44. K. Lange, Singular value decomposition, in *Numerical Analysis for Statisticians* (Ed., K. Lange), Statistics and Computing, Springer, New York, US, (2010), 129–142. https://doi.org/10.1007/978-1-4419-5945-4_9

45. A. A. Maciejewski, C. A. Klein, The singular value decomposition: Computation and applications to robotics, *Int. J. Robot. Res.*, **8** (1989), 63–79. https://doi.org/10.1177/027836498900800605
46. J. Mandel, Use of the singular value decomposition in regression analysis, *Am. Statist.*, **36** (1982), 15–24. https://doi.org/10.1080/00031305.1982.10482771
47. A. Yildiz Ulus, Teaching the "diagonalization concept" in linear algebra with technology: A case study at Galatasaray University, *Turkish Online J. Educ. Technology-TOJET*, **12** (2013), 119–130.
48. Z. Lazar, *Teaching the Singular Value Decomposition of Matrices: A Computational Approach*, Masters' thesis, Concordia University, Montreal, Quebec, Canada, 2012.
49. M. Zandieh, M. Wawro, C. Rasmussen, An example of inquiry in linear algebra: The roles of symbolizing and brokering, *PRIMUS*, **27**(2017), 96–124. https://doi.org/10.1080/10511970.2016.1199618
50. B. Buchberger, G. E. Collins, R. Loos, R. Albrecht, (Eds.), *Computer Algebra: Symbolic and Algebraic Computation*, Second edition, Springer Science & Business Media, Berlin Heidelberg, Germany, 1983.
51. V. Chudnovsky, R. D. Jenks, (Eds.), *Computers in Mathematics*, CRC Press, Boca Raton, Florida, US, 1990.
52. J. S. Cohen, *Computer Algebra and Symbolic Computation: Elementary Algorithms*, CRC Press, Boca Raton, Florida, US, 2002. https://doi.org/10.1201/9781439863695
53. J. S. Cohen, *Computer Algebra and Symbolic Computation: Mathematical Methods*, CRC Press, Boca Raton, Florida, US, 2003. https://doi.org/10.1201/9781439863701
54. J. H. Davenport, Y. Siret, É. Tournier, *Computer Algebra: Systems and Algorithms for Algebraic Computation*, Second edition, Academic Press, Cambridge, Massachusetts, US, 1993.
55. J. T. Fey, (Ed.), *Computer Algebra Systems in Secondary School Mathematics Education*, National Council of Teachers of Mathematics (NCTM), Reston, Virginia, US, 2003.
56. K. J. Fuchs, Computer algebra systems in mathematics education: Teacher training programs, challenges and new aims, *Zentralblatt für Didaktik der Mathematik*, **35** (2003), 20–23. https://doi.org/10.1007/BF02652762
57. K. O. Geddes, S. R. Czapor, G. Labahn, *Algorithms for Computer Algebra*, Springer Science & Business Media, Berlin Heidelberg, Germany, 1992. https://doi.org/10.1007/b102438
58. J. Grabmeier, E. Kaltofen, V. Weispfenning, (Eds.), *Computer Algebra Handbook: Foundations, Applications, Systems*, Springer, Berlin Heidelberg, Germany, 2003. https://doi.org/10.1007/978-3-642-55826-9
59. D. Harper, C. Wooff, D. Hodgkinson, *A Guide to Computer Algebra Systems*, John Wiley & Sons, New York, US, 1991.
60. W. Koepf, *Computer Algebra: An Algorithm-Oriented Introduction*, Springer Nature, Berlin Heidelberg, Germany, 2021. https://doi.org/10.1007/978-3-030-78017-3
61. E. A. Lamagna, *Computer Algebra: Concepts and Techniques*, CRC Press, Boca Raton, Florida, US, 2019. https://doi.org/10.1201/9781315107011

62. G. Simon, *Interoperability Between Computer Algebra Systems*, Wilhelm-Schickard-Institut für Informatik (WSI), Tübingen, Germany, 1996.
63. N. M. Soiffer, *The Design of A User Interface for Computer Algebra Systems*, PhD thesis, University of California, Berkeley, California, US, 1992.
64. J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, Third edition, Cambridge University Press, Cambridge, England, UK, 2013. https://doi.org/10.1017/CBO9781139856065
65. M. J. Wester, *Computer Algebra Systems: A Practical Guide*, John Wiley & Sons, New York, US, 1999.
66. Z. Hannan, *wxMaxima for Calculus I*, *wxMaxima for Calculus II*, Solano Community College, Fairfield, California, US, 2015. Available from `https://wxmaximafor.wordpress.com/`. Last accessed August 17, 2023.
67. M. Kanagasabapathy, *Introduction to wxMaxima for Scientific Computations*, BPB Publications, New Delhi, India, 2018.
68. S. Kadry, P. Awad, *Mathematics for Engineers and Science Labs Using Maxima*, CRC Press, Boca Raton, Florida, US, 2019. https://doi.org/10.1201/9780429469718
69. F. Senese, *Symbolic Mathematics for Chemists: A Guide for Maxima Users*, John Wiley & Sons, Hoboken, New Jersey, US, 2019.
70. T. K. Timberlake, J. W. Mixon, *Classical Mechanics with Maxima*, Springer, New York, US, 2016. https://doi.org/10.1007/978-1-4939-3207-8
71. M. L. Abell, J. P. Braselton, *Mathematica by Example*, Sixth edition, Academic Press, London, England, UK and Cambridge, Massachusetts, US, 2022. https://doi.org/10.1016/C2013-0-10266-8
72. A. Grozin, *Introduction to Mathematica® for Physicists*, Springer, Cham, Switzerland, 2014. https://doi.org/10.1007/978-3-319-00894-3
73. R. Maeder, *Programming in Mathematica*, Second edition, Addison-Wesley Longman Publishing, Boston, Massachusetts, US, 1991.
74. M. Trott, *The Mathematica Guidebook for Symbolics*, Springer Science & Business Media New York, US, 2007. https://doi.org/10.1007/0-387-28815-5
75. S. Wagon, *Mathematica in Action*, Second edition, Springer-Verlag, New York, US, 1999. https://doi.org/10.1007/978-0-387-75477-2
76. S. Wolfram, *The MATHEMATICA® Book*, Fifth edition, Wolfram Media, Champaign, Illinois, US, 2003.
77. M. L. Abell, J. P. Braselton, *Maple by Example*, Third edition, Elsevier, Burlington, Massachusetts, US, 2005.
78. W. P. Fox, W. Bauldry, *Advanced Problem Solving Using Maple: A First Course*, Chapman and Hall/CRC Press, Boca Raton, Florida, US, 2019. https://doi.org/10.1201/9780429469633
79. W. P. Fox, W. Bauldry, *Advanced Problem Solving Using Maple: Applied Mathematics, Operations Research, Business Analytics, and Decision Analysis*, Chapman and Hall/CRC Press, Boca Raton, Florida, US, 2020. https://doi.org/10.1201/9780429469626

80. F. Garvan, *The Maple Book*, Chapman and Hall/CRC Press, Boca Raton, Florida, US, 2001. https://doi.org/10.1201/9781420035605
81. J. Carette, Understanding expression simplification, in *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation*, ISSAC'04, July 4–7, 2004, Santander, Spain, (2004), 72–79. https://doi.org/10.1145/1005285.1005298
82. A. Heck, *Introduction to Maple*, Third edition, Springer, New York, US, 2003. https://doi.org/10.1007/978-1-4613-0023-6
83. S. Attaway, *MATLAB: A Practical Introduction to Programming and Problem Solving*, Sixth edition, Butterworth-Heinemann, Oxford, England, UK, 2013. https://doi.org/10.1016/C2011-0-07060-6
84. T. A. Davis, *MATLAB Primer*, Eight edition, CRC Press, Boca Raton, Florida, US, 2010. https://doi.org/10.1201/9781439828632
85. D. M. Etter, *Introduction to MATLAB*, Fourth edition, Pearson, New York, US, 2017.
86. D. J. Higham, N. J. Higham, *MATLAB Guide*, Third edition, SIAM, Philadelphia, Pennsylvania, US, 2017. https://doi.org/10.1137/1.9781611974669
87. D. T. Valentine, B. Hahn, *Essential MATLAB for Engineers and Scientists*, Eight edition, Academic Press, Cambridge, Massachusetts, US, 2022.
88. G. V. Bard, *Sage for Undergraduates*, American Mathematical Society, Providence, Rhode Island, US, 2015. https://doi.org/10.1090/mbk/143
89. C. Finch, *Sage Beginner's Guide*, Packt Publishing, Birmingham, England, UK, 2011.
90. D. Joyner, W. Stein, *Sage Tutorial*, CreateSpace Independent Publishing Platform, Scotts Valley, California, US, 2008.
91. V. Kumar, *Basic of SageMath: Mathematics (Practical)*, Amazon Kindle Direct Publishing, Seattle, Washington, US, 2022.
92. P. Szabó, J. Galanda, Sage math for education and research, in *2017 15th International Conference on Emerging eLearning Technologies and Applications (ICETA)*, Institute of Electrical and Electronics Engineers (IEEE), Manhattan, New York, US, (2017), 1–4. https://doi.org/10.1109/ICETA.2017.8102535
93. P. Zimmermann, A. Casamayou, N. Cohen, G. Connan, T. Dumont, L. Fousse, et al., in *Computational Mathematics with SageMath*, SIAM, Philadelphia, Pennsylvania, US, 2018. https://doi.org/10.1137/1.9781611975468
94. S. Frieder, L. Pinchetti, R. R. Griffiths, T. Salvatori, T. Lukasiewicz, P. C. Petersen, et al., Mathematical capabilities of ChatGPT, arXiv preprint, (2023). `arXiv:2301.13867`.
95. P. Shakarian, A. Koyyalamudi, N. Ngu, L. Mareedu, An independent evaluation of ChatGPT on mathematical word problems (MWP), arXiv preprint, `arXiv:2302.13814`.
96. A. Azaria, ChatGPT usage and limitations, HAL preprint, `hal-03913837`, 2022. https://doi.org/10.31219/osf.io/5ue7n
97. A. Borji, A categorical archive of ChatGPT failures, arXiv preprint, `arXiv:2302.03494`.

98. X. Q. Dao, N. B. Le, ChatGPT is good but Bing Chat is better for Vietnamese students, arXiv preprint, `arXiv:2307.08272`.

99. P. Nguyen, P. Nguyen, P. Bruneau, L. Cao, J. Wang, H. Truong, H. Evaluation of mathematics performance of Google Bard on the mathematics test of the Vietnamese national high school graduation examination, preprint, 2023. https://doi.org/10.36227/techrxiv.23691876

100. M. M. Meerschaert, *Mathematical Modeling*, Fourth edition, Academic Press, Waltham, Massachusetts, US, 2013. https://doi.org/10.1016/C2010-0-66940-9

101. J. M. Cushing, Matrix models and population dynamics, *Math. Biol.*, **14** (2009), 47–150. https://doi.org/10.1090/pcms/014/04

102. W. E. Boyce, R. C. DiPrima, D. B. Meade, *Elementary Differential Equations and Boundary Value Problems*, 12th edition, John Wiley & Sons, New York, US, 2022.

103. S. J. Leon, L. de Pillis, *Linear Algebra with Applications*, 10th edition, Pearson Education, Upper Saddle River, New Jersey, US, 2020.

104. M. P. S. Chawla, PCA and ICA processing methods for removal of artifacts and noise in electrocardiograms: A survey and comparison, *Appl. Soft Comput.*, **11** (2011), 2216–2226. https://doi.org/10.1016/j.asoc.2010.08.001

105. A. Cichocki, S. I. Amari, *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*, John Wiley & Sons, Hoboken, New Jersey, US, 2002. https://doi.org/10.1002/0470845899

106. M. Ringnér, What is principal component analysis?, *Nat. Biotechnol.*, **26** (2008), 303–304. https://doi.org/10.1038/nbt0308-303

107. S. Sanei, J. A. Chambers, *EEG Signal Processing and Machine Learning*, John Wiley & Sons, Hoboken, New Jersey, US, 2021. https://doi.org/10.1002/9781119386957

108. M. W. Blows, A tale of two matrices: multivariate approaches in evolutionary biology, *J. Evolut. Biol.*, **20** (2007), 1–8. https://doi.org/10.1111/j.1420-9101.2006.01164.x

109. G. Abraham, M. Inouye, Fast principal component analysis of large-scale genome-wide data, *PloS One*, **9** (2014), e93766. https://doi.org/10.1371/journal.pone.0093766

110. N. Duforet-Frebourg, K. Luu, G. Laval, E. Bazin, M. G. Blum, Detecting genomic signatures of natural selection with principal component analysis: Application to the 1000 genomes data, *Molecular Biol. Evolut.*, **33** (2016), 1082–1093. https://doi.org/10.1093/molbev/msv334

111. X. Zheng, B. S. Weir, Eigenanalysis of SNP data with an identity by descent interpretation, *Theor. Population Biol.*, **107** (2016), 65–76. https://doi.org/10.1016/j.tpb.2015.09.004

112. N. Abu-Shikhah, F. Elkarmi, Medium-term electric load forecasting using singular value decomposition, *Energy*, **36** (2011), 4259–4271. https://doi.org/10.1016/j.energy.2011.04.017

113. L. Cai, N. F. Thornhill, B. C. Pal, Multivariate detection of power system disturbances based on fourth order moment and singular value decomposition, *IEEE Transactions on Power Systems*, **32** (2017), 4289–4297. https://doi.org/10.1109/TPWRS.2016.2633321

114. K. Ellithy, M. Shaheen, M. Al-Athba, A. Al-Subaie, S. Al-Mohannadi, S. Al-Okkah, S. Abu-Eidah, Voltage stability evaluation of real power transmission system using singular value de-

composition technique, in *2008 IEEE Second International Power and Energy Conference*, IEEE, Manhattan, New York, US, (2008), 1691–1695. https://doi.org/10.1109/PECON.2008.4762751

115. A. M. A. Hamdan, An investigation of the significance of singular value decomposition in power system dynamics, *Int. J. Electr. Power Energy Syst.*, **21** (1999), 417–424. https://doi.org/10.1016/S0142-0615(99)00011-3
116. C. Madtharad, S. Premrudeepreechacharn, N. R. Watson, Power system state estimation using singular value decomposition, *Electr. Power Syst. Res.*, **67** (2003), 99–107. https://doi.org/10.1016/S0378-7796(03)00080-4
117. G. Kerschen, J. C. Golinval, Physical interpretation of the proper orthogonal modes using the singular value decomposition, *J. Sound Vibr.*, **249** (2002), 849–865. https://doi.org/10.1006/jsvi.2001.3930
118. N. K. Mani, E. J. Haug, K. E. Atkinson, Application of singular value decomposition for analysis of mechanical system dynamics, *J. Mechan. Design*, **107** (1985), 82–87. https://doi.org/10.1115/1.3258699
119. G. Sun, W. Li, Q. Luo, Q. Li, Modal identification of vibrating structures using singular value decomposition and nonlinear iteration based on high-speed digital image correlation, *Thin-Walled Structures*, **163** (2021), 107377. https://doi.org/10.1016/j.tws.2020.107377
120. C. Cloud, G. Li, E. H. Maslen, L. E. Barrett, W. C. Foiles, Practical applications of singular value decomposition in rotordynamics, *Australian J. Mechan. Eng.*, **2** (2005), 21–32. https://doi.org/10.1080/14484846.2005.11464477
121. D. W. Gu, P. Petkov, M. M. Konstantinov, *Robust Control Design with MATLAB®*, Springer Science & Business Media, London, England, UK, 2005. https://doi.org/10.1007/978-1-4471-4682-7
122. F. Lin, *Robust Control Design: An Optimal Control Approach*, John Wiley & Sons, Hoboken, New Jersey, US, 2007. https://doi.org/10.1002/9780470059579
123. J. Ringwood, Multivariable control using the singular value decomposition in steel rolling with quantitative robustness assessment, *Control Eng. Pract.*, **3** (1995), 495–503. https://doi.org/10.1016/0967-0661(95)00021-L
124. C. R. Smith III, *Multivariable Process Control using Singular Value Decomposition*, PhD dissertation, The University of Tennessee, Knoxville, Tennessee, US, 1981.
125. G. Tao, *Adaptive Control Design and Analysis*, John Wiley & Sons, Hoboken, New Jersey, US, 2003. https://doi.org/10.1002/0471459100
126. S. Gai, G. Yang, M. Wan, L. Wang, Denoising color images by reduced quaternion matrix singular value decomposition, *Multidimen. Syst. Signal Process.*, **26** (2015), 307–320. https://doi.org/10.1007/s11045-013-0268-x
127. E. Ganic, A. M. Eskicioglu, Robust embedding of visual watermarks using discrete wavelet transform and singular value decomposition, *J. Electron. Imag.*, **14** (2005), 043004. https://doi.org/10.1117/1.2137650
128. R. C. Gonzalez, R. E. Woods, *Digital Image Processing*, Fourth edition, Pearson Education, New York, US, and Harlow, Essex, UK, 2018.

129. C. C. Lai, C. C. Tsai, Digital image watermarking using discrete wavelet transform and singular value decomposition, *IEEE Transact. Instrument. Measur.*, **59** (2010), 3060–3063. https://doi.org/10.1109/TIM.2010.2066770

130. S. Malini, R. S. Moni, Image denoising using multiresolution singular value decomposition transform, *Proced. Computer Sci.*, **46** (2015), 1708–1715. https://doi.org/10.1016/j.procs.2015.02.114

131. J. P. Pandey, S. L. Umrao, Digital image processing using singular value decomposition, in *Proceedings of Second International Conference on Advanced Computing and Software Engineering (ICACSE)*, February 8–9, 2019, Kamla Nehru Institute of Technology, Sultanpur, India, (2019), 3. https://doi.org/10.2139/ssrn.3350278

132. A. Rajwade, A. Rangarajan, A. Banerjee, Image denoising using the higher order singular value decomposition, *IEEE Transact. Pattern Anal. Machine Intell.*, **35** (2012), 849–862. https://doi.org/10.1109/TPAMI.2012.140

133. F. Renault, D. Nagamalai, M. Dhanuskodi, Advances in digital image processing and information technology, in *Proceedings of the First International Conference in Digital Image Processing and Pattern Recognition*, September 23–25, 2011, Tirunelveli, Tamil Nadu, India, (2011), 23–25. https://doi.org/10.1007/978-3-642-24055-3

134. J. Bisgard, *Analysis and Linear Algebra: The Singular Value Decomposition and Applications*, American Mathematical Society, Providence, Rhode Island, US, 2020. https://doi.org/10.1090/stml/094

135. A. Blum, J. Hopcroft, R. Kannan, *Foundations of Data Science*, Cambridge University Press, Cambridge, England, UK, 2020. https://doi.org/10.1017/9781108755528

136. S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, R. Harshman, Indexing by latent semantic analysis, *Journal of the American society for Information Science*, **41** (1990), 391–407. https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

137. T. Hastie, R. Tibshirani, J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer, New York, US, 2009. https://doi.org/10.1007/978-0-387-84858-7

138. Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, *Computer*, **42** (2009), 30–37. https://doi.org/10.1109/MC.2009.263

139. X. Li, S. Wang, Y. Cai, Tutorial: Complexity analysis of singular value decomposition and its variants, arXiv preprint, `arXiv:1906.12085`.

140. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al., Scikit-learn: Machine learning in Python, *J. Machine Learn. Res.*, **12** (2011), 2825–2830.

141. J. B. Tenenbaum, V. D. Silva, J. C. Langford, A global geometric framework for nonlinear dimensionality reduction, *Science*, **290** (2000), 2319–2323. https://doi.org/10.1126/science.290.5500.2319

142. Z. Zhang, The singular value decomposition, applications and beyond, arXiv preprint, `arXiv:1510.08532`.

143. `https://www.sagemath.org/`

144. S. M. D'Souza, L. N. Wood, Secondary students' resistance toward incorporating computer technology into mathematics learning, *Math. Comput. Educ.*, **37** (2003), 284–295. https://doi.org/10.1007/978-94-6300-761-0_8

145. M. L. Niess, Guest Editorial: Preparing teachers to teach mathematics with technology, *Contempor. Issues Technol. Teacher Educ.*, **6** (2006), 195–203. https://doi.org/10.1007/978-0-387-35596-2_69

146. Q. Li, Student and teacher views about technology: A tale of two cities?, *J. Res. Center Educ. Technol.*, **39** (2007), 377–397. https://doi.org/10.1080/15391523.2007.10782488

147. H. Crompton, Mathematics in the age of technology: There is a place for technology in the mathematics classroom, *J. Res. Center Educ. Technol.*, **7** (2011), 54–66.

148. M. Prensky, Digital natives, digital immigrants Part 1, *On the Horizon*, **9** (2001), 1–6. https://doi.org/10.1108/10748120110424816

149. M. Prensky, Digital natives, digital immigrants Part 2: Do they really think differently?, *On the Horizon*, **9** (2001), 2–6. https://doi.org/10.1108/10748120110424843

150. M. Prensky, H. sapiens digital: From digital immigrants and digital natives to digital wisdom, *Innovate J. Online Educ.*, **5** (2009), 1–9.

151. M. Prensky, *Teaching Digital Natives: Partnering for Real Learning*, Corwin Press, Thousand Oaks, California, US, 2010.

A. SageMath code

This appendix provides SageMath code for matrix diagonalization, orthogonal diagonalization, and SVD.

A.1. Matrix diagonalization

```
A=matrix([[1,2,2],[0,2,1],[0,1,2]])
print("Matrix A = ")
print(A)
print()
print("Characteristic polynomial of A: p(x) =", A.charpoly())
print()
print("Eigenvalues of A =", A.eigenvalues())
print()
print("Eigenvalue, eigenvector, and geometric multiplicity:", )
print(A.eigenvectors_right())
print()
lambda1 = A.eigenvalues()[0]
lambda2 = A.eigenvalues()[1]
lambda3 = A.eigenvalues()[2]
print("Extracting eigenvalues:")
print("lambda1 =", lambda1)
print("lambda2 =", lambda2)
print("lambda3 =", lambda3)
print()
D=diagonal_matrix(A.eigenvalues())
print("Diagonal matrix D = ")
print(D)
print()
r1=A.eigenvectors_right()[0][1]
p1=2*r1[0]
r2=A.eigenvectors_right()[1][1]
p2=r2[0]
r3=A.eigenvectors_right()[1][1]
p3=r3[1]
print("Extracting eigenvectors:")
print("p1 =", p1)
print("p2 =", p2)
print("p3 =", p3)
print()
P=matrix([p1,p2,p3]).transpose()
print("Invertible matrix P = ")
print(P)
print()
print("Inverse of P, P^(-1) = ")
print(P.inverse())
print()
print("Calculate PDP^(-1) = ")
print(P*D*P.inverse())
print("= A")
```

A.2. Orthogonal diagonalization

```
A=matrix([[3,-4,-4],[-4,3,-4],[-4,-4,3]])
print("Matrix A = ")
print(A)
print()
print("Characteristic polynomial of A: p(x) =", A.charpoly())
print()
print("Eigenvalues of A =", A.eigenvalues())
print()
lambda1 = A.eigenvalues()[0]
lambda2 = A.eigenvalues()[1]
lambda3 = A.eigenvalues()[2]
print("Extracting eigenvalues:")
print("lambda1 =", lambda1)
print("lambda2 =", lambda2)
print("lambda3 =", lambda3)
print()
D=diagonal_matrix(A.eigenvalues())
print("Diagonal matrix D = ")
print(D)
print()
print("Eigenvalue, eigenvector, and geometric multiplicity:", )
print(A.eigenvectors_right())
print()
q1=A.eigenvectors_right()[0][1][0]
q2=A.eigenvectors_right()[1][1][0]
q3=A.eigenvectors_right()[1][1][1]
print("Extracting eigenvectors:")
print("q1 =", q1)
print("q2 =", q2)
print("q3 =", q3)
print()
print("Checking orthogonality:")
print("<q1,q2> =", q1.inner_product(q2))
print("<q1,q3> =", q1.inner_product(q3))
print("<q2,q3> =", q2.inner_product(q3))
print()
print("q2 and q3 are not orthogonal; apply the Gram-Schmidt process:")
print()
q31 = q3 - q2.inner_product(q3)/q2.inner_product(q2)*q2
print("q3’ =", q31)
print()
print("Check that now q2 and q3’ are orthogonal: <q2,q3’> =", q2.inner_product(q31))
print()
print("Normalize all orthogonal eigenvectors:")
print()
p1 = q1/sqrt(q1.inner_product(q1))
print("p1 =", p1)
print()
p2 = q2/sqrt(q2.inner_product(q2))
print("p2 =", p2)
print()
q32 = 2*q31
p3 = -q32/sqrt(q32.inner_product(q32))
print("p3 =", p3)
print()
print("Construct an invertible and orthogonal matrix P:")
```

```
P = matrix([p1,p2,p3]).transpose()
print("P = ")
print(P)
print()
print("Inverse of P, P^(-1) = ")
print(P.inverse())
print()
print("Calculate PDP^(-1) = ")
print(P*D*P.inverse())
print("= A")
print()
print("Transpose of P, P^T = ")
print(P.transpose())
print()
print("Calculate PDP^T = ")
print(P*D*P.transpose())
print("= A")
```

A.3. SVD

```
M = matrix([ [2,-2], [-3,-4], [-4,-3] ])
print("Matrix M = ")
print(M)
print()
A = M.transpose()*M
print("Matrix M^T M = ")
print(A)
print()
print("Eigenvalues of M^T M =", A.eigenvalues())
print()
print("Eigenvalue, eigenvector, and geometric multiplicity:", )
print(A.eigenvectors_right())
print()
sigma1 = sqrt(A.eigenvalues()[0])
sigma2 = sqrt(A.eigenvalues()[1])
print("Singular value of M:")
print("sigma1 =", sigma1)
print("sigma2 =", sigma2)
print()
Sigma = matrix([[sigma1, 0], [0, sigma2], [0,0]])
print("Matrix Sigma =")
print(Sigma)
print()
w1 = A.eigenvectors_right()[0][1][0]
```

```
v1 = w1/w1.norm()
w2 = A.eigenvectors_right()[1][1][0]
v2 = w2/w2.norm()
print("Eigenvectors of M^T M:")
print("v1 =", v1)
print("v2 =", v2)
print()
print("Matrix V =")
V = matrix([ v1, v2 ])
print(V)
print()
u1 = 1/sigma1*M*v1
u2 = 1/sigma2*M*v2
u3 = u1.cross_product(u2)
print("Left singular vectors of M:")
print("u1 =", u1)
print("u2 =", u2)
print("u3 =", u3)
print()
print("Checking orthogonality")
print("u1.u2 =", u1.dot_product(u2))
print("u1.u3 =", u1.dot_product(u3))
print("u2.u3 =", u3.dot_product(u2))
print()
UT = matrix([u1, u2, u3])
U = UT.transpose()
print("Matrix U =")
print(U)
print()
print("SVD of M = U*Sigma*V^T =")
print(U*Sigma*V.transpose())
print("= M")
```

AIMS Press

© 2023 the author, licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0)