Project-Based Teaching of Matrix Operations in Linear Algebra: Insights from AI Frontier Applications

Yan Liang, Simin Wu*
School of Science, Guangdong University of Petrochemical Technology, Maoming, Guangdong, China
**Corresponding Author*

Abstract: With the rapid advancement of Artificial Intelligence (AI) technologies, matrices and their operations， which are core topics in linear algebra, play a pivotal role in AI frontiers. However, in current linear algebra course, the teaching of matrix operations remains largely centered on abstract theoretical explanations and traditional applications, with insufficient integration of cutting-edge AI technologies. This gap weakens students' perception of the connection between theory and real-world applications. This study examines the principles of basic matrix operations and their representative AI application scenarios, selecting two typical cases for analysis: matrix operations in basic image processing and matrix factorization in recommender systems. For the image processing case, a practical project-based instructional design are proposed,as well as its analysis. The results demonstrate that integrating frontier AI applications into linear algebra teaching yields positive teaching and learning outcomes, offering valuable insights for innovating teaching models and enriching instructional practices.

Keywords: AI Frontier Application Cases; Teaching Innovation; Matrix Operations; Problem-chain Teaching Design

1. Introduction

In the past decade, the rapid advancement of Artificial Intelligence (AI) technologies has underscored the central role of mathematical foundations. Among these, linear algebra, with its rigorous framework for vector spaces, linear transformations, and matrix operations, serves as a cornerstone for the design and implementation of AI theories and algorithms. As a fundamental construct, the matrix not only provides a universal form for data representation and storage but also functions as the computational vehicle for algorithm design and model training.

Matrix operations permeate nearly every facet of AI. Images are represented as pixel matrices [1]; textual data are transformed into high-dimensional matrices[2]; graph data are modeled using adjacency matrices[3] and so on. In deep neural networks, model parameters are stored as matrices or tensors, and both forward and backward propagation reduce to core operations such as matrix multiplication and addition[4,5]. Additionally, matrix factorization methods, including singular value decomposition (SVD), eigenvalue decomposition, and non-negative matrix factorization, play critical roles in feature extraction, dimensionality reduction, and recommendation algorithms[6-8].

The co-evolution of matrix theory and computational technology has been mutually reinforcing. Advances in high-performance computing (HPC) and GPU-accelerated libraries (e.g., cuBLAS, MKL) have enabled large-scale matrix computation, laying the hardware foundation for training deep models at scale. Conversely, AI's emerging demands have inspired new research in matrix theory, including efficient algorithms for sparse matrices, low-rank approximation techniques, and structured matrices for neural network compression[9].

Despite these developments, the teaching of matrix operations in higher education often remains anchored in abstract derivations and traditional applications[10], leaving a gap between theory and contemporary AI practice. Integrating AI frontier application cases into linear algebra courses can bridge this gap, enabling students to appreciate the practical significance of matrix theory while cultivating interdisciplinary modeling skills and algorithmic implementation capabilities.

Motivated by this need, this study proposes an innovative approach to teaching matrix operations through AI-oriented application cases. Two representative scenarios are examined in increasing complexity: matrix operations in basic

image processing and matrix factorization in recommendation systems. Section 2 outlines the theoretical foundations and AI relevance of matrices and their operations. Section 3 presents detailed modeling and computational analysis of the selected cases. Section 4 discusses pedagogical strategies for embedding these cases into linear algebra instruction. Section 5 concludes with key findings and directions for future research.).

2. Theoretical Foundations: Matrix Operations and Their Relevance of AI

A matrix is a rectangular array of complex numbers, real numbers, or other mathematical objects, with wide applications in mathematics, physics, engineering, and computer science. In AI, matrices serve not only as carriers of data but also as the core computational tools for implementing algorithms. This section starts from the fundamental operations of matrices and explains their connections to AI applications.

2.1 Matrix Operations and Their Relevance of AI

Let $A \in R^{m\times n}$ be a matrix with elements a_{ij}, where a_{ij} denotes the entry in the *i*-th row and *j*-th column. The fundamental operations of matrices and their typical application scenarios in AI algorithms is going to be introduced. In teaching practice, teachers may design AI application cases by linking matrix operations with their roles in AI algorithms.

2.1.1 Matrix linear operations (addition and scalar multiplication)

Matrix addition and scalar multiplication are collectively referred to as linear operations, expressed as:

$$A + B = (a_{ij} + b_{ij}),$$
$$\alpha A = (\alpha a_{ij}) \quad (1)$$

In AI, this type of operation is commonly used for updating model parameters, such as parameter adjustment in gradient descent:

$$W^{(t+1)} = W^{(t)} - \eta \nabla L(W^{(t)}) \quad (2)$$

2.1.2 Matrix multiplication

If $A \in R^{m\times p}$, $B \in R^{p\times n}$, then the product $C \in R^{m\times n}$ is

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} \quad (3)$$

Matrix multiplication is widely used. For example, in the core computation of neural network forward propagation: multiplying the input vector with the weight matrix to obtain a new feature representation.

2.1.3 Transpose and conjugate transpose

The transpose of a matrix is denoted as A^T. In AI, it is used for transformations of feature matrices and for similarity computations. For example, in the attention mechanism, it is necessary to compute the product of the Query matrix and the transpose of Key matrix.

2.1.4 Inverse matrix and generalized inverse

For an invertible matrix A, it holds that $AA^{-1} = A^{-1}A = I$. In AI algorithms, direct matrix inversion is rarely used; however, the generalized inverse is often applied in solving least squares problems and computing closed-form solutions of models, such as in the normal equation for linear regression.

2.1.5 Block matrices and parallel computation

The block matrix form facilitates parallel computation in large-scale data processing, such as mini-batch training on GPUs.

2.2 Matrix Factorization and Feature Extraction

Matrix factorization is the process of decomposing a matrix into multiple matrices with special structures, widely used in AI for dimension reduction, feature extraction, and model compression.

2.2.1 Singular value decomposition (SVD)

For a matrix $A \in R^{m\times n}$, singular value decomposition is expressed as:

$$A = U\Sigma V^T \quad (4)$$

2.2.2 Eigenvalue decomposition

Eigenvalue decomposition is given by:

$$A = P\Lambda P^{-1} \quad (5)$$

where Λ is the eigenvalue matrix. Algorithms such as Markov chains and spectral clustering in AI rely on eigenvalue decomposition.

2.2.3 Non-negative matrix factorization (NMF)

A non-negative matrix A can be decomposed into the product of two non-negative matrices W and H:

$$A \approx WH \quad (6)$$

Non-negative matrix factorization plays an important role in topic modeling and pattern recognition.

2.3 High-Dimensional Extensions of Matrix Operations

In AI, much of the data takes the form of three-dimensional or higher-order tensors, such as

color images (height × width × channels). A matrix is a special case of a tensor with two dimensions, and many matrix operations can be directly extended to tensor operations, such as convolution, dot product, and tensor decomposition. Understanding matrix operations forms the foundation for mastering high-dimensional tensor computations.

3. Matrix operations in AI Frontier Application Case Studies

3.1 Case 1: Matrix Operations in Basic Image Processing

3.1.1 Background and application scenarios

A digital image is essentially a matrix whose element values represent pixel brightness or color. In AI image-related tasks, data preprocessing often involves direct matrix operations. Examples include image rotation, brightness adjustment, and watermark addition. Although these operations are relatively simple, they play important roles in data augmentation before model training, copyright protection, and feature visualization. In the following, we illustrate these concepts through image modeling and computational implementation.

3.1.2 Image modeling and processing

A grayscale image can be represented as a matrix by dividing it into pixels. The more pixels the image is divided into, the finer the resolution and the clearer the image. Each pixel block can be represented by a single grayscale value, and these grayscale values, arranged according to their pixel positions, form a matrix. The greater the number of pixel divisions, the higher the order of the corresponding matrix.

Let the grayscale image matrix be

$$A \in R^{m \times n} \tag{7}$$

where the element a_{ij} represents the pixel value at the i-th row and j-th column. In a typical computer grayscale image, pixel values range from [0,255], with 0 representing black and 255 representing white. The parameter $m \times n$ indicates the number of pixel divisions, that is, the resolution of the image.

3.1.2.1 Rotation

Let A denote the matrix representing the original image, and B denote the matrix representing the rotated image. To achieve image rotation, a permutation matrix with all 1's on its secondary diagonal can be used in matrix multiplication. When the image needs to be rotated 90∘to the left, the transformation can be expressed as:

$$B = PA^T \tag{8}$$

When the image needs to be rotated 90∘ to the right

$$B = A^T P \tag{9}$$

When the image needs to be rotated 180∘

$$B = PAP \tag{10}$$

When the image needs to be rotated 90∘ to the left while performing a left-right mirror reflection—that is, reversing along the top-left to bottom-right diagonal—the transformation can be achieved directly by transposing the matrix:

$$B = A^T \tag{11}$$

3.1.2.2 Highlighting

Highlighting refers to increasing the pixel values, without exceeding the maximum value (usually 255). Highlighting can take various forms, such as overall highlighting, proportional highlighting, and local highlighting. Below is the matrix operation representation for highlighting, where B denotes the matrix corresponding to the new image.

When applying overall highlighting to an image, matrix addition can be used:

$$b_{ij} = \min(a_{ij} + k, 255) \tag{12}$$

where a_{ij}, b_{ij} represent the elements in the i-th row and j-th column of matrices A and B, respectively. When applying proportional highlighting to an image, matrix multiplication can be used:

$$b_{ij} = \min(ka_{ij}, 255) \tag{13}$$

where k is the highlight ratio. When applying local highlighting to an image:

$$B = A + M \circ \Delta \tag{14}$$

where M is the model matrix of the highlighted region, and Δ is the highlight matrix, which is usually taken as a constant matrix. The operation $\circ$ is represented by Hadamard element-wise multiplication. It should be noted that the elements' value of the matrix B must also be ensured not to exceed 255.

3.1.2.3 Adding a watermark

An digital image processing, image watermarking refers to embedding additional information (such as text, graphics, or logos) into the original image to indicate copyright, source, or authorship when displayed or distributed. Let the watermark image be represented by the matrix $W \in R^{m \times n}$, then adding the watermark can be expressed as:

$$B = A + \alpha W \quad (15)$$

where $0 \le \alpha \le 1$ is used to control the transparency of the watermark. Similarly, the elements' value of the matrix B must be ensured not to exceed 255.

3.1.3 Analysis and discussion

From the above process of image modeling and operations, we can see that even very basic matrix addition, scalar multiplication, and matrix multiplication play a significant role in image processing. These fundamental matrix operations form the foundation for understanding advanced image processing techniques such as convolution and filtering. In practical AI models, similar operations are part of data preprocessing, which can directly affect model accuracy. Therefore, matrix operations have broad and important applications in image processing and serve as an essential basis for cutting-edge image processing technologies, such as face recognition and image restoration.

3.2 Case 2: Matrix Factorization in Recommendation Systems

3.2.1 Background and application scenarios

In e-commerce, film and television, music and other recommendation platforms, the system needs to predict users' interest in non-contact items, which can be abstracted as a sparse matrix completion problem. How to learn a model that can reasonably estimate missing items from a small set of observed values in a given user item rating matrix? A common method to solve this problem is matrix factorization. The Matrix Factorization (MF) method represents a high-dimensional sparse matrix as the product of two low dimensional matrices through low rank approximation, thereby extracting "latent factors" and widely used in collaborative filtering recommendations.

3.2.2 Mathematical modeling and low rank hypothesis

Assuming there are m users and n items. Let the rating matrix

$$R \in R^{m \times n} \quad (16)$$

where its element r_{ij} represents the rating value of user i for the item j (if not rated, the position is empty). We use two low dimensional matrices to approximate R:

$$R \approx PQ^T \quad (17)$$

where $P \in R^{m \times k}$ represents the latent feature preference matrix of users. The i-th row vector p_i^T indicates the degree of preference (weight) of user i for each of the k latent feature factors; $Q \in R^{n \times k}$ represents the latent feature matrix of items, where the j-th row vector q_i^T indicates the feature attributes of item j across the k latent feature factors. In general, k is much less than m, n, thus forming R a low-rank matrix.

Next, for the set of known ratings $\Omega = \{ (i,j) | r_{ij} \quad is \quad known \}$,define the predicted value as

$$\hat{r}_{ij} = p_i^T q_j \quad (18)$$

The above intuitive explanation is: there exist k latent feature factors, and each user and item can be represented as a vector in this k-dimensional space. The rating is given by the inner product of the two vectors (geometrically, this corresponds to the degree of matching in terms of vector dot product).

Then, P, Q (only fitted at observed positions) are estimated by constructing a regularized least squares objective function:

$$J(P,Q) = \sum_{(i,j)\in\Omega} (r_{ij} - p_i^T q_j)^2 + \lambda(\|P\|_F^2 + \|Q\|_F^2) \quad (19)$$

The first term in Equation (19) represents the fitting error (summed only over the observed entries), while λ in the second term is the regularization coefficient, which is used to prevent overfitting and improve numerical stability.

3.2.3 Optimization objective function and solution

This problem can be solved using the Alternating Least Squares (ALS) method. The idea of ALS is to fix one matrix and solve for the other, then alternate between them, iterating until convergence.

When Q is fixed, for each user i, we only consider the set of items they have rated, denoted by $\Omega_i = \{ j | r_{ij} \quad \text{is} \quad know \}$. Then arrange the feature vectors of these items into the matrix Q_i ($Q_i \in R^{|\Omega_i| \times k}$), with the corresponding ratings forming the vector r_i, $\mathrm{r}_i \in R^{|\Omega_i|}$.

At this point, the subproblem is:

$$\min_{p_i} \left\| \mathrm{r}_i - Q_i p_i \right\|_2^2 + \lambda \left\| p_i \right\|_2^2 \quad (20)$$

Taking the derivative with respect to p_i and

setting the gradient to zero gives:

$$-2Q_i^T(\mathrm{r}_i - Q_i\, p_i) + 2\lambda p_i = 0 \quad (21)$$

Rearranging Equation (21) yields the normal equation:

$$(Q_i^T Q_i + \lambda I_k) p_i = Q_i^T \mathrm{r}_i \quad (22)$$

Solving for p_i gives:

$$p_i = (Q_i^T Q_i + \lambda I_k)^{-1} Q_i^T \mathrm{r}_i \quad (23)$$

Similarly, when P is fixed, for each item j the normal equation is:

$$(P_j^T P_j + \lambda I_k) q_j = P_i^T \mathrm{r}_j \quad (24)$$

where P_j and r_j are the user feature matrix and rating vector corresponding to item j, respectively. Solving for q_j yields:

$$q_j = (P_j^T P_j + \lambda I_k)^{-1} P_i^T \mathrm{r}_j \quad (25)$$

The advantage of this method is that each subproblem has a closed-form solution, is numerically stable, and allows for parallel updates of users or items.

3.2.4 Analysis and discussion

A recommendation system leverages matrix factorization to transform the recommendation problem into a matching model in a latent space, where the similarity between user vectors and item vectors (e.g., defined by the inner product) determines the predicted ratings.

The core advantage of this approach lies in transforming the originally sparse and high-dimensional rating matrix into the product of two dense, low-dimensional matrices, thereby significantly reducing computational complexity and improving prediction accuracy.

The modeling framework of matrix factorization exhibits remarkable scalability. By adjusting the matrix structure and constraints, it can be directly adapted to other AI tasks, such as social network link prediction (determining whether a connection between two nodes is likely to occur) and knowledge graph completion. Essentially, these tasks can all be abstracted as matrix (or higher-dimensional tensor) completion problems, for which matrix factorization provides a unified and efficient solution. Teachers can also design additional AI application cases based on this idea.

4. Integration Strategies and Insights for Linear Algebra Teaching

4.1 The Necessity of Introducing AI Cases into Linear Algebra Teaching

Traditional linear algebra courses often focus on definitions, theorems, proofs, and a small number of applications in physics and engineering, leaving students with an insufficient perception of the "real-world significance" of matrix operations. In the AI era, matrices are not only theoretical tools but also the core carriers of data and the driving force of algorithms. Introducing AI cases such as CNNs, matrix factorization, and attention mechanisms into the classroom can: enhance students' awareness of the connection between mathematics and cutting-edge technologies; stimulate learning interest and motivation; and cultivate cross-disciplinary modeling and computational thinking.

The introduction of AI cases is not merely a transfer of knowledge but also a form of ability training. In the modeling stage, students need to abstract real-world problems into mathematical models in the form of matrices, thereby strengthening their mathematical modeling skills. In the implementation stage, they use tools such as Python and MATLAB to perform matrix operations and program design, enhancing their computational experiment capabilities. In the results processing stage, they must interpret, compare, and optimize based on the computational results, fostering rigorous analytical skills.

Furthermore, incorporating AI cases can create a positive cycle of teaching–research interaction: university instructors can feed AI scenarios and matrix optimization methods accumulated in research back into classroom teaching, allowing students to directly engage with the latest applications of mathematics in cutting-edge research. At the same time, new ideas or improvements proposed by students in class may, in turn, inspire research directions, forming a virtuous cycle.

4.2 Case-Driven Problem Chain Design

To systematically integrate AI-related cases into linear algebra teaching, teachers can adopt teaching framework such as the "Four-steps Problem Chain,"[11] breaking down complex AI tasks into a series of progressive learning problems that advance from mathematical foundations to technology implementation. Using the basic image processing case as an example, we present a problem chain and corresponding teaching activity design

recommendations.

4.2.1 Example of problem chain design

Modeling Starting Problem: How can a grayscale image be represented as a matrix? What is the relationship between pixel value ranges and matrix elements? How are matrix transpose and permutation matrices defined, and how do they affect images?

Theoretical Transition Problem: How can a permutation matrix be used to rotate an image by 90°, 180°, or 270°? How do matrix addition and scalar multiplication change image brightness? How can matrix addition be used to embed an image watermark? Explore the underlying operational properties and invertibility of these operations.

Application Transformation Problem: Given an image, how can rotation, brightness enhancement, and watermark embedding operations be abstracted into a unified matrix operation model? Can multiple image processing steps (e.g., rotation + brightness + watermark) be expressed using a single formula?

Engineering Implementation Problem: How can these operations be efficiently implemented using NumPy or MATLAB? How does the time/space complexity of these matrix operations vary with different image sizes? Can sparse matrix storage be used to reduce memory consumption?

This problem chain design ensures theoretical rigor while guiding students to map abstract mathematics onto concrete AI technology problems

4.2.2 Teaching activities design

Lecturing and Derivation (Theory Class): Introduce the method of representing images as matrices; derive the construction of permutation matrices and explain how they achieve image rotation; use examples to demonstrate how matrix addition/scalar multiplication affects brightness; analyze the matrix operation formulas for watermark overlay.

Mini Experiments (Lab / Assignments): Provide students with a grayscale image matrix and require them to perform a 90° left rotation, increase brightness by 20%, and overlay a watermark with 30% transparency. Students should include the corresponding matrix operation formulas in their experiment reports.

Students should successfully use the theories to achieve the requirement in technology and show the outcomes in their report. For example,

Figure 1 shows the rotation of a picture and Figure 2 shows adding a watermark.

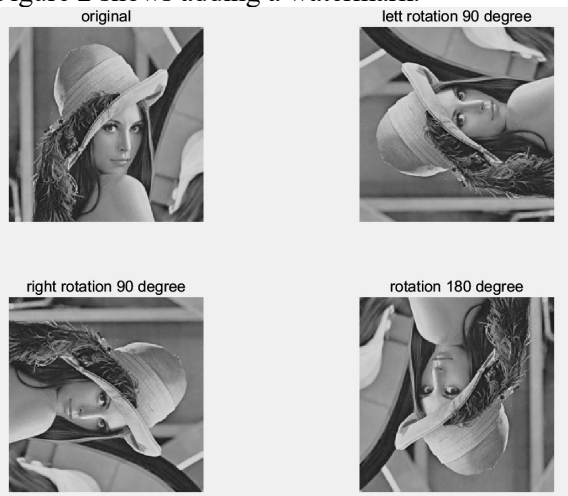

Figure 1. Outcomes of Rotation. Generated by the Authors Using Matlab

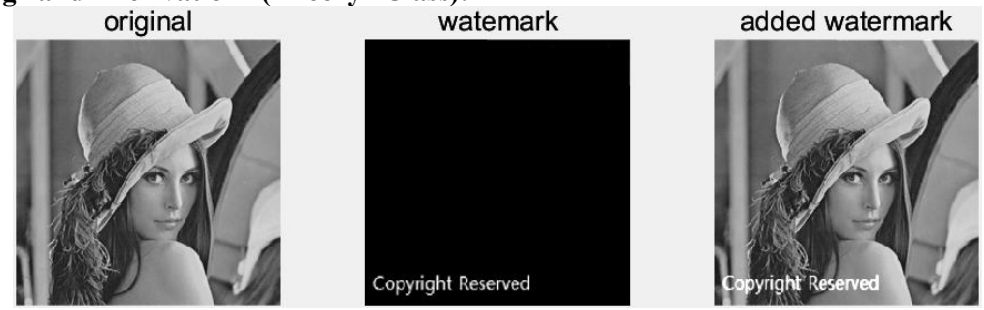

Figure 2. Adding a Watermark. Generated by the Authors Using Matlab

Project-Based Learning (PBL): Groups complete an "Image Processing Tool" project where users can input a grayscale image and choose a combination of operations (rotation, brightness adjustment, watermark embedding) to output the processed image. Final achievements include source code, processed examples, and an implementation explanation.

Class Discussion and Research Extension: Explore how matrix operations extend to color images (RGB three-channel); discuss how basic image processing connects to computer vision tasks such as edge detection and convolutional feature extraction.

Through these AI-driven, project-based tasks, students experience the full process from data-to-

matrix conversion, to matrix computation, and finally to technology project. This allowing them to appreciate the real-world application value of matrix operations through practice.

5. Conclusions

This paper investigates innovative teaching approaches for integrating cutting-edge AI application cases into the teaching of matrix operations in linear algebra. It summarizes the fundamental principles of matrix operations and their connections to AI algorithms. Based on these principles and their relevance to AI, two AI application cases were designed in increasing order of complexity: matrix operations in basic image processing and matrix decomposition in recommendation systems. The study then analyzes the rationale and models for integrating AI application cases into teaching and, using the basic image processing case, develops a project-based teaching design for practical classroom implementation. The results provide a reference for innovating teaching models and enriching teaching practice.

Acknowledgment

This work was supported by Guangdong Association of Higher Education "14th Five-Year Plan" 2022 Higher Education Research Project (No. 22GQN34); and 2023 Guangdong Provincial Educational Science Planning Project (Higher Education Special Program) (No. 2023GXJK404).

References

[1] Liu, X., Toh, K. A., & Allebach, J. P. (2019). Pedestrian detection using pixel difference matrix projection. IEEE transactions on intelligent transportation systems, 21(4), 1441-1454.

[2] Gao, L., Song, J., Liu, X., Shao, J., Liu, J., & Shao, J. (2017). Learning in high-dimensional multimedia data: the state of the art. Multimedia Systems, 23(3), 303-313.

[3] Chiaselotti, G., Gentile, T., Infusino, F. G., & Oliverio, P. A. (2017). The adjacency matrix of a graph as a data table: A geometric perspective. Annali di Matematica Pura ed Applicata (1923-), 196(3), 1073-1112.

[4] Gao, Z. F., Cheng, S., He, R. Q., Xie, Z. Y., Zhao, H. H., Lu, Z. Y., & Xiang, T. (2020). Compressing deep neural networks by matrix product operators. Physical Review Research, 2(2), 023-300.

[5] Liang, Y., Ibrahim, A., & Omar, Z. (2023). Richardson Iterative Method for Solving Multi-Linear System with M-Tensor. Malaysian Journal of Mathematical Sciences, 17(4), 645–671.

[6] Zarzour, H., Al-Sharif, Z., Al-Ayyoub, M., & Jararweh, Y. (2018, April). A new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques. In 2018 9th international conference on information and communication systems (ICICS) (pp. 102-106). IEEE.

[7] Jia, W., Sun, M., Lian, J., & Hou, S. (2022). Feature dimensionality reduction: a review. Complex & Intelligent Systems, 8(3), 2663-2693.

[8] Qiao, K., Yu, K., Qu, B., Liang, J., Yue, C., & Ban, X. (2022). Feature extraction for recommendation of constrained multiobjective evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 27(4), 949-963.

[9] Li, Y., Yu, Y., Zhang, Q., Liang, C., He, P., Chen, W., & Zhao, T. (2023, July). Losparse: Structured compression of large language models based on low-rank and sparse approximation. In International Conference on Machine Learning (pp. 20336-20350). PMLR.

[10] Dorier, J. L., & Sierpinska, A. (2001). Research into the teaching and learning of linear algebra. In The teaching and learning of mathematics at university level: An ICMI study (pp. 255-273). Dordrecht: Springer Netherlands.

[11] Liang Y. (2024) Research on the Four-Step Teaching Model of Objective-Problem-Oriented Teaching Method in Linear Algebra under the Context of "Four New Trends". Proceedings of the Seminar on Reform and Innovation in University Mathematics Teaching in the New Era, Beijing: Higher Education Press, May, 61-66.

