Development of Online Learning Materials for Tensor Data Processing Exercises

Shota Abe, Akio Ishida, Jun Murakami, and Naoki Yamamoto

***Abstract*—Tensor decomposition is used in a wide range of research fields; however, its theory is difficult to understand. Therefore, basic education is essential when using it in programming. Currently, there are few Japanese universities that provide education on tensor decomposition; however, some overseas universities have already conducted it, and online learning materials are also substantial. Therefore, in this paper, we have developed online learning materials for basics and programming exercises of higher-order singular value decomposition (HOSVD), which is one of tensor decomposition, for the purpose of increasing the learning materials for tensor decomposition education. Our learning material is created on Microsoft Teams, and students can access this material channel and work on exercises on demand while watching explanatory videos including CG animation. As a result of the trial of this learning material, it was found that the students who used it can generally understand the processes related to tensor decomposition and can perform basic programming of them.**

***Index Terms*—Tensor decomposition, online learning materials, HOSVD, 3D puzzle, R language.**

I. Introduction

Tensors are one of the important data structures used as multidimensional arrays in data processing. Tensor decomposition, which is one of the tensor data processing, is applied in a wide range of research fields such as signal processing, numerical linear algebra, computer vision, numerical analysis, data mining, graph analysis, and neuroscience [1]. In order for undergraduate students to apply this tensor decomposition in their graduation research project, it will be important to educate them on the basics and application of the decomposition.

In Japan, tensor decomposition education is conducted at several higher education institutions. For example, at Rikkyo University Graduate School of Artificial Intelligence and Science, lectures on application cases of tensor decomposition such as recommender systems and location information are given in a course of Special Seminar on Artificial Intelligence [2]. In subjects of Physics Special Treatise Training I to VI conducted at Chuo University Graduate School of Science and Engineering, students read texts and resumes on variable selection by unsupervised learning using tensor decomposition and aim to apply it to bioinformatics by understanding the contents [3]. At Kyoto University Graduate School of Informatics, a lecture on the basics of tensor decomposition and tensor networks is given in the subject of Nonlinear Physics Special Lecture II [4]. As in these examples, in Japan, lectures on tensor decomposition are conducted at graduate schools of science and engineering.

Education on tensor decomposition in other countries is more common than in Japan, and online learning materials are also substantial. For example, at the University of Illinois, a detailed lecture on tensor decomposition and tensor network theory, algorithms, etc. is given in a subject called Tensor Computations [5]. Homework and quizzes are also provided online in this subject, and students can deepen their understanding by implementing algorithms using Python and answering quizzes. MIT OpenCourseWare of the Massachusetts Institute of Technology offers a course called Algorithmic Aspects of Machine Learning [6]. This course focuses on the design of basic machine learning algorithms such as non-negative matrix factorization and tensor decomposition, and lecture notes, assignments, and references are available online. Both the subjects described here are offered at the graduate level of each university. As shown in the example above, our survey revealed that most of the subjects dealing with tensor decomposition are offered at the graduate level, both in Japan and abroad.

We have been developing learning materials that are useful for understanding the concept of tensors and how to handle them. One of the developed learning materials is a set of R language scripts that express several types of 3D puzzles in tensors and solve them by matrix unfolding used in tensor decomposition [7]. Another learning material deals with 3D puzzles composed of MacMahon's cubes and is useful for understanding matrix unfolding and folding used in tensor decomposition [8]. The latter learning material was tried for junior high school students and technical college students, and the difficulty level, visibility, and required time were evaluated. Recently, we have developed online learning materials based on the results of our research so far [9]. This learning material expresses a 3D Puzzle called "Light Bulb Placement Puzzle" in tensors, and teaches its basic programming techniques, partial tensor sum, matrix unfolding, and folding, online with exercises.

In this paper, we added an explanation of the product of tensors and matrices called *n*-mode product to this latest learning material shown in ref. [9] so that we can teach tensor decomposition and made it available online. The points that have been changed or added in the new learning materials are as follows.

Manuscript received August 20, 2021; revised September 22, 2021. This work was supported in part by the JSPS KAKENHI (Grants-in Aid for Scientific Research) under Grant No. JP18K11596.

S. Abe is with Kumamoto College, National Institute of Technology, Koshi, Japan (e-mail: ae21abe@g.kumamoto-nct.ac.jp).

A. Ishida is with the Faculty of Liberal Arts, Kumamoto College, National Institute of Technology, Koshi, Japan (e-mail: ishida@kumamoto-nct.ac.jp).

J. Murakami and N. Yamamoto are with the Faculty of Electronics and Information Systems Engineering, Kumamoto College, National Institute of Technology, Koshi, Japan (e-mail: jun@kumamoto-nct.ac.jp, naoki@kumamoto-nct.ac.jp).

doi: 10.18178/ijiet.2022.12.3.1604

1) Explanations and exercises for *n*-mode product and tensor decomposition were introduced.
2) MacMahon's cube is mainly treated as a material to be expressed by tensor. Since there are many exercises for online learning in the new materials, a simpler puzzle was selected compared to the light bulb puzzle.
3) CG animations have been added to the video explanation of the slides to make process of tensor operation easier to understand visually.
4) Applied exercises that were previously given at the end of the material are now done at the end of each exercise.

The order of the chapters in this paper is as follows. Chapter 2 explains the theory related to tensors and tensor decomposition and the materials used in the learning materials. Chapter 3 describes the system configuration of the learning materials and detailed explanations of each exercise include in it. Furthermore, specific status of the exercises by students is explained. In Chapter 4, the results of the trial use of these materials by the students are concluded, and the consideration is given at the end.

II. Theory of Tensor Data Processing, Tensor Examples, and Programing Language Used for Exercises

In this chapter, we describe tensor and its operations and theory of tensor decomposition. Furthermore, we explain tensor examples used in learning material developed in this study and programming language used in exercises.

A. Theory of Tensor and Its Operations

In this study, the tensor means a multidimensional array, and its elements are real values. Now, a definition of an Nth order tensor is shown below.

[(Definition 1) *N*th Order Tensor]

Let $\mathcal{A} \in \mathbf{R}^{I_1 \times I_2 \times \cdots \times I_n \times \cdots \times I_N}$ be an *N*th order tensor of size $I_1 \times I_2 \times \cdots \times I_n \times \cdots \times I_N$ with real numbers. Then, let $a_{i_1 i_2 \cdots i_n \cdots i_N}$ be an $(i_1, i_2, \ldots, i_n, \ldots, i_N)$ element of $\mathcal{A}$, the tensor is defined by the following equation.

$$\mathcal{A} = (a_{i_1 i_2 \cdots i_n \cdots i_N}), \quad \begin{pmatrix} i_1 = 1,2,\ldots,I_1; i_2 = 1,2,\ldots,I_2; \cdots; \\ i_n = 1,2,\ldots,I_n; \cdots; i_N = 1,2,\ldots,I_N \end{pmatrix}. \tag{1}$$

(End of Definition)

From this definition, a 1st order tensor (a_{i_1}) and a 2nd order tensor ($a_{i_1 i_2}$) correspond to a vector and a matrix, respectively. A 3rd order or higher tensor is a 3-dimensional or higher multidimensional array.

The subscripts $i_n, (n = 1,2,\ldots,N)$ in (1) represent directions of the tensor and are called modes. In an example of a 3rd order tensor, there are three subscripts, i_1, i_2, i_3, which are called 1-mode, 2-mode, and 3-mode, respectively. We use a 3rd order tensor that correspond to the 1-mode, the 2-mode, and the 3-mode in vertical, horizontal, and depth directions of the tensor, respectively.

Next, we describe important operations of tensors, such as *n*-mode matrix unfolding, folding, and *n*-mode product. Firstly, the *n*-mode matrix unfolding is an operation to rearrange a tensor into a matrix, and the definition for an *N*th order tensor $\mathcal{A}$ is shown below [10].

[(Definition 2) *n*-Mode Matrix Unfolding]

Moving an $a_{i_1 i_2 \cdots i_n \cdots i_N}$ element of the *N*th order tensor $\mathcal{A}$ shown in Definition 1 to an (i_n, j_n) element of a matrix $\boldsymbol{A}_{(n)} \in \mathbf{R}^{I_n \times I_{n+1} I_{n+2} \cdots I_N I_1 I_2 \cdots I_{n-2} I_{n-1}}$, $(n = 1,2,\ldots,N)$ is called *n*-mode matrix unfolding. Where j_n is given by the following equation.

$$\begin{aligned} j_n = {} & (i_{n+1}-1) I_{n+2} \cdots I_N I_1 \cdots I_{n-2} I_{n-1} \\ & + (i_{n+2}-1) I_{n+3} \cdots I_N I_1 \cdots I_{n-2} I_{n-1} + \cdots \\ & + (i_N - 1) I_1 \cdots I_{n-2} I_{n-1} + (i_1 - 1) I_2 \cdots I_{n-2} I_{n-1} \\ & + \cdots + (i_{n-2} - 1) I_{n-1} + i_{n-1}. \end{aligned} \tag{2}$$

The above operation is performed by changing all subscripts $i_n, (n = 1,2,\ldots,N)$.

(End of Definition)

From this definition, the matrix unfolding of a 3rd order tensor $\mathcal{A}$ is as follows.

1-Mode Matrix Unfolding: An $a_{i_1 i_2 i_3}$ element of $\mathcal{A}$ is moved to an $(i_1, \{(i_2 - 1) I_3 + i_3\})$ element of $\boldsymbol{A}_{(1)}$.

2-Mode Matrix Unfolding: An $a_{i_1 i_2 i_3}$ element of $\mathcal{A}$ is moved to an $(i_2, \{(i_3 - 1) I_1 + i_1\})$ element of $\boldsymbol{A}_{(2)}$.

3-Mode Matrix Unfolding: An $a_{i_1 i_2 i_3}$ element of $\mathcal{A}$ is moved to an $(i_3, \{(i_1 - 1) I_2 + i_2\})$ element of $\boldsymbol{A}_{(3)}$.

That is, this unfolding is the operation to move *n*-mode of $\mathcal{A}$ to the row of $\boldsymbol{A}_{(n)}$ and remaining modes of $\mathcal{A}$ to the column of $\boldsymbol{A}_{(n)}$. In this learning material, implementation of matrix unfolding of a 3rd order tensor based on Definition 2 is given as an applied task of an exercise.

Secondly, folding is a reverse operation of matrix unfolding, which is the operation of converting an unfolded matrix into a tensor. This definition is shown below.

[(Definition 3) Folding]

Returning an (i_n, j_n) element of $\boldsymbol{A}_{(n)}$, $(n = 1,2,\ldots,N)$ in Definition 2 to an $a_{i_1 i_2 \cdots i_n \cdots i_N}$ element of an *N*th order tensor $\mathcal{A}$ is called folding. Where j_n is given by (2).

The above operation is applied while changing all subscripts of $\mathcal{A}$.

(End of Definition)

Thirdly, this learning material introduces an exercise of *n*-mode product, which is essential for calculation of tensor decomposition described in the next section. As for *n*-mode, it is represented an *n*th subscript i_n of a tensor. The *n*-mode product is an operation of a product of a tensor and a matrix for i_n. A definition of this product operation is shown below [10].

[(Definition 4) *n*-Mode Product]

The *n*-mode product of an *N*th order tensor $\mathcal{A}$ shown in Definition 1 and a size $J_n \times I_n$ matrix $\boldsymbol{U}^{(n)} \in \mathbf{R}^{J_n \times I_n}$ is defined by the following equation.

$$(\mathcal{A}\times_n \boldsymbol{U}^{(n)})_{i_1 i_2\cdots j_n\cdots i_N} = \sum_{i_n=1}^{I_n} a_{i_1 i_2\cdots i_n\cdots i_N} u_{j_n i_n}, \quad \begin{pmatrix} i_1 = 1,2,\ldots,I_1; i_2 = 1,2,\ldots,I_2;\cdots; i_n = 1,2,\ldots,I_n; \\ \cdots; i_N = 1,2,\ldots,I_N; j_n = 1,2,\ldots,J_n \end{pmatrix}. \tag{3}$$

Where $(\mathcal{A}\times_n \boldsymbol{U}^{(n)})_{i_1 i_2\cdots j_n\cdots i_N}$ is an $(i_1, i_2, \ldots, j_n, \ldots, i_N)$ element of a size $I_1\times I_2\times\cdots\times J_n\times\cdots\times I_N$ *N*th order tensor $\mathcal{A}\times_n \boldsymbol{U}^{(n)} \in \mathbf{R}^{I_1\times I_2\times\cdots\times J_n\times\cdots\times I_N}$, and $u_{j_n i_n}$ denotes a (j_n, i_n) element of $\boldsymbol{U}^{(n)}$.

(End of Definition)

As shown in (3), tensor operations are generally complicated in this way. In this learning material, implementation of the *n*-mode product using Definition 4 is an applied task of an exercise.

According to Definition 4, *n*-mode product can be calculated; however, it is difficult to understand it. Therefore, in order for learners to understand processing of this operation, implementation by algorithm shown below is also introduced in the exercise [11].

[(Algorithm 1) *n*-Mode Product]

Input: Size $I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N$ *N*th order tensor $\mathcal{A}$, size $J_n\times I_n$ matrix $\boldsymbol{U}^{(n)}$.

Output: Size $I_1\times I_2\times\cdots\times J_n\times\cdots\times I_N$ *N*th order tensor $\mathcal{A}\times_n \boldsymbol{U}^{(n)}$.

(Step 1) Apply *n*-mode matrix unfolding to $\mathcal{A}$ and unfold $\mathcal{A}$ to a matrix $\boldsymbol{A}_{(n)}$.

(Step 2) Calculate a matrix product $\boldsymbol{U}^{(n)}\boldsymbol{A}_{(n)}$ of $\boldsymbol{U}^{(n)}$ and $\boldsymbol{A}_{(n)}$ obtained in Step 1.

(Step 3) $\mathcal{A}\times_n \boldsymbol{U}^{(n)}$ is obtained by folding $\boldsymbol{U}^{(n)}\boldsymbol{A}_{(n)}$ calculated in Step 2 into an *N*th order tensor for *n*-mode.

(Step 4) Return $\mathcal{A}\times_n \boldsymbol{U}^{(n)}$ obtained in Step 3.

(End of Algorithm)

In addition, the following property is important in an operation of *n*-mode product [11].

[(Property 1) *n*-Mode Product]

Now $\mathcal{A}$ denotes an *N*th order tensor, and $\boldsymbol{U}^{(o)}$ and $\boldsymbol{U}^{(p)}$ are matrices. If $o \neq p$, then the following equation holds for operations of *o*-mode product and *p*-mode product.

$$\begin{aligned} \mathcal{A}\times_o \boldsymbol{U}^{(o)}\times_p \boldsymbol{U}^{(p)} &= (\mathcal{A}\times_o \boldsymbol{U}^{(o)})\times_p \boldsymbol{U}^{(p)} \\ &= (\mathcal{A}\times_p \boldsymbol{U}^{(p)})\times_o \boldsymbol{U}^{(o)} \end{aligned}. \tag{4}$$

(End of Property)

B. Theory of Tensor Decomposition

Tensor decomposition is important operation in tensor data processing. This is used to express high-dimensional tensor data as products or sums of lower-dimensional tensors. Higher Order Singular Value Decomposition (HOSVD) [10] is one of the tensor decompositions and can be practiced in this learning material. Furthermore, HOSVD is an extension of Singular Value Decomposition (SVD) of a matrix to the decomposition of the third or higher order tensors. HOSVD definition for an Nth order tensor and its algorithm are shown below [10].

[(Definition 5) HOSVD]

Now consider an *N*th order tensor $\mathcal{A}$ of Definition 1. The tensor $\mathcal{A}$ is decomposed as the following equation by *n*-mode product of size $I_n\times I_n$ orthonormal matrices $\boldsymbol{U}^{(n)} \in \mathbf{R}^{I_n\times I_n}$, $(n = 1,2,\ldots,N)$ and a core tensor $\mathcal{C} \in \mathbf{R}^{I_1\times I_2\times\cdots\times I_n\times\cdots\times I_N}$ of the same size as $\mathcal{A}$, and this is called HOSVD of the *N*th order tensor.

$$\mathcal{A} = \mathcal{C}\times_1 \boldsymbol{U}^{(1)}\times_2 \boldsymbol{U}^{(2)}\cdots\times_N \boldsymbol{U}^{(N)}. \tag{5}$$

Where operators $\times_n$, $(n = 1,2,\ldots,N)$ represent *n*-mode product.

(End of Definition)

[(Algorithm 2) HOSVD]

Input: Size $I_1\times I_2\times\cdots\times I_N$ *N*th order tensor $\mathcal{A}$.

Output: Size $I_n\times I_n$ orthonormal matrices $\boldsymbol{U}^{(n)}$, $(n = 1,2,\ldots,N)$, size $I_1\times I_2\times\cdots\times I_N$ core tensor $\mathcal{C}$.

(Step 1) Apply *n*-mode matrix unfolding to the input tensor $\mathcal{A}$ in order to unfold it into *N* matrices $\boldsymbol{A}_{(n)} \in \mathbf{R}^{I_n\times I_{n+1}I_{n+2}\cdots I_N I_1 I_2\cdots I_{n-2}I_{n-1}}$, $(n = 1,2,\ldots,N)$ of the size $I_n\times(I_{n+1}I_{n+2}\cdots I_N I_1 I_2\cdots I_{n-2}I_{n-1})$.

(Step 2) Applying SVD to each of $\boldsymbol{A}_{(n)}$ obtained in Step 1, the matrices are decomposed as follows.

$$\boldsymbol{A}_{(n)} = \boldsymbol{U}^{(n)}\boldsymbol{\Sigma}^{(n)}\boldsymbol{V}^{(n)\mathrm{T}}, (n = 1,2,\ldots,N). \tag{6}$$

Where $\boldsymbol{U}^{(n)}$ and $\boldsymbol{V}^{(n)}$ represent left and right singular matrices, respectively, and $\boldsymbol{\Sigma}^{(n)}$ is a diagonal matrix with singular values in the diagonal elements. The operator T means a transpose of a matrix.

(Step 3) A core tensor $\mathcal{C}$ can be calculated by the following equation from *n*-mode product of $\mathcal{A}$ and $\boldsymbol{U}^{(n)}$ obtained in Step 2.

$$\mathcal{C} = \mathcal{A}\times_1 \boldsymbol{U}^{(1)\mathrm{T}}\times_2 \boldsymbol{U}^{(2)\mathrm{T}}\cdots\times_N \boldsymbol{U}^{(N)\mathrm{T}}. \tag{7}$$

(Step 4) Return $\boldsymbol{U}^{(n)}$, $(n = 1,2,\ldots,N)$ and $\mathcal{C}$ calculated in Step 2 and 3, respectively.

(End of Algorithm)

Regarding exercises in HOSVD, a basic task using its function prepared in a library and an applied task to implement it following Algorithm 2 are assigned.

C. Material Used for Tensor Examples

In this learning material, from a viewpoint of making learners more interested in tensor data processing, in addition to a tensor example composed of simple numerical values, a 3D puzzle and a colored cube are also used as material for the examples.

As the 3D puzzle, Light Bulb Placement Puzzle introduced in ref. [9] is used. Fig. 1 shows an image of this puzzle expressed in a tensor. In this figure, light bulbs are placed on each element of a $3\times3\times3$ cube. Then, in this puzzle, $3^2 = 9$ out of $3^3 = 27$ light bulbs are lit well so that all the elements appear to be lit when viewed from 3 directions. In Fig. 1,

elements colored in red represent lighting of light bulbs, and a solution of this puzzle can be confirmed by taking a partial tensor sum of matrices sliced along each mode. Therefore, this is a basic task for practicing the sum.

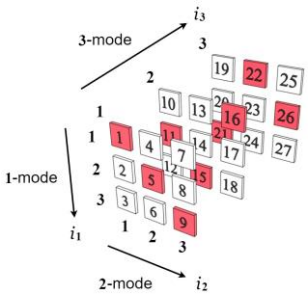

Fig. 1. Tensor representation of 3D puzzle.

Another material is a MacMahon's cube [12] as shown in Fig. 2. This cube is painted in 6 different colors on each side, and the colors used here are red(1), white(2), blue(3), green(4), yellow(5), and black(6). Note that the numbers in parentheses denote the numbers for each color.

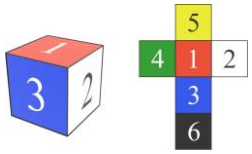

Fig. 2. Example of MacMahon's cube.

Fig. 3 shows a tensor representation devised by extending the cube. This extends the cube in Fig. 2 to a 3rd order tensor $\mathcal{A} \in \mathbf{R}^{3\times3\times3}$ of the size $3\times3\times3$, and the color number of each face shown in Fig. 2 is stored in central element of each side of $\mathcal{A}$. Note that the other elements of $\mathcal{A}$ are zero.

The tensor example shown in Fig. 3 appears in most of the exercises in this learning material. Specifically, it is used for the exercises of *n*-mode matrix unfolding, folding, *n*-mode product, and tensor decomposition.

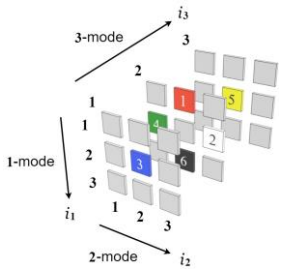

Fig. 3. Tensor representation of MacMahon's cube.

D. Programming Language and Package Used for Exercises

As programing language, R [13] is used, and rTensor [14] is installed and used in a package for tensor operation. In each exercise of this learning material, the following functions of rTensor are used.

as.tensor function: Function to create a tensor object that can be used with rTensor.

modeSum function: Function to calculate a partial tensor sum.

unfold function: Function to convert from a tensor to *n*-mode matrix unfolding.

fold function: Function to fold *n*-mode matrix unfolding into a tensor.

ttm function: Function to calculate *n*-mode product of a tensor and a matrix.

hosvd function: Function to calculate HOSVD of a tensor.

fnorm function: Function to calculate Frobenius norm of a tensor.

However, for implementation of *n*-mode matrix unfolding, folding, *n*-mode product, and HOSVD, there are also application tasks such as programming based on definitions and algorithms without directly using the above functions.

III. Contents of Developed Learning Material and Trial Results

A. System Configuration of Learning Material

In this study, we develop learning material for understanding the contents of tensor data processing described in Chapter 2 through programming in R language. Fig. 4 shows overall structure of the learning material. Students who are learners can watch explanation videos on Microsoft (MS) Teams, perform exercises, and submit assignments from Google Form. Exercises are basically performed in R environment, and some exercises are answered in MS Word files. The submitted assignments can be reviewed by teachers.

This learning material is created on a MS Teams channel. Fig. 5 shows an overview of the learning material that consists of each thread on that channel. The configuration is described below.

Firstly, Fig.6 shows a thread corresponding to the part (A) in Fig. 5. The first thread describes recommended environment in which the learning material can be properly executed. The second one gives preparations and precautions before using the learning material.

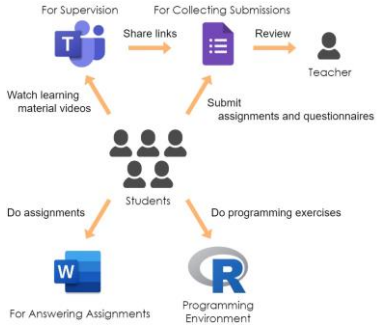

Fig. 4. Overall structure of the learning material.

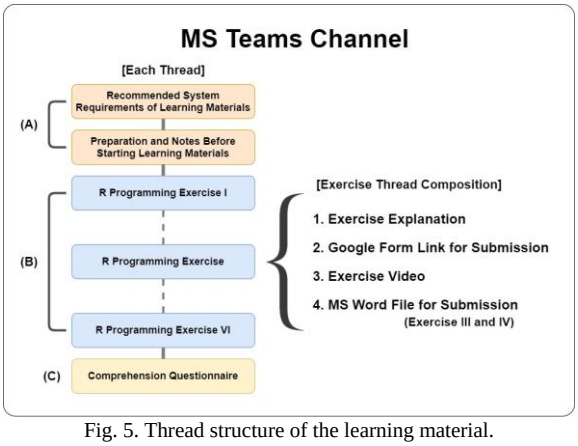

Fig. 5. Thread structure of the learning material.

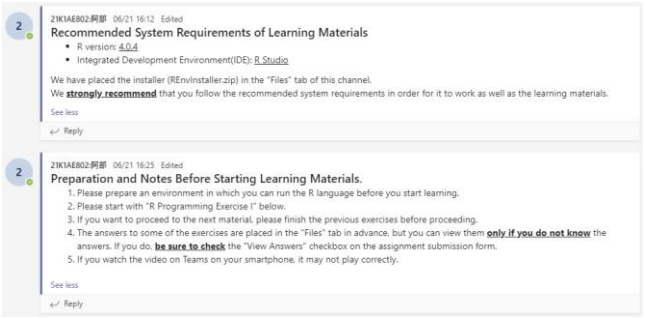

Fig. 6. Thread in part (A) of Fig. 5.

Fig. 7. Thread example of Exercise III.

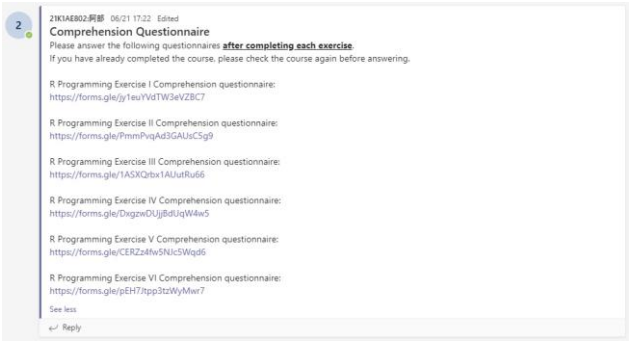

Fig. 8. Thread of comprehension questionnaire.

Secondly, the threads (B) in Fig. 5 are the part where exercises are performed, and six exercises are provided. As an example of this, a thread of Exercise III is shown in Fig. 7. This thread has shared links for an explanatory video, a Word file for answers, and Google Form for submitting assignments. As a supplement, some exercises have sample answers on the File tab. Therefore, to grasp the learner's practice situation, we asked them to check a questionnaire item on the submission form if they referred to the sample answers.

Thirdly, Fig. 8 shows the thread in the part (C) of Fig. 5. This thread contains comprehension questionnaires that learners answer after completing each exercise. The questionnaires are conducted to measure how well they understand important contents of each exercise. They perform a self-assessment on a five-point scale.

B. Contents of Learning Material and Results of Each Exercise

This learning material have been tried to use by five students. They were four 5th grade students (correspond to second-year university students) and one advanced course student (correspond to a third-year university student) belonging to the departments related information engineering in our college. All of them have an experience using R language, and one of formers has also used the rTensor package.

In the following sections, regarding (B) in Fig.5 explained in the previous section, the contents of each exercise and its trial results (exercise tasks and comprehension questionnaires) are shown in order from Exercise I.

1) R programming Exercise I

At first, we provide an exercise for learning the outline of tensors and how to install the rTensor package required to handle tensors in R language. The task in this exercise is to install the rTensor while watching a video explaining it.

Here, the result of the comprehension questionnaire in this exercise are shown in Fig. 9. As the result of this questionnaire, it was confirmed that the students had a good understanding of the tensor and its mode, because 80% of them answered 4 for each question.

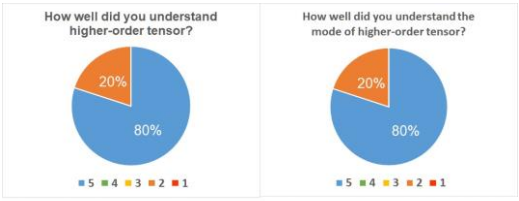

Fig. 9. Questionnaire results in Exercise I.

2) R programming Exercise II

In Exercise II, students can learn about the generation of tensors using the rTensor and the sum of partial tensors. From this exercise onward, they proceed with learning by implementing themselves the presented tasks. We used the Light Bulb Placement Puzzle described in Chapter 2 as the material of this example exercise. The task of this exercise is to implement the example exercise following the explaining video.

Regarding the performance of the exercise, it was confirmed that all the students could implement it from the review of submitted source codes. In addition, Fig. 10 shows the results of the comprehension questionnaire in this exercise. As a result, over 80% of them rated the comprehension level as 4 or higher in each item. From the above, it is considered that the meaning of the partial tensor sum and its implementation were well understood.

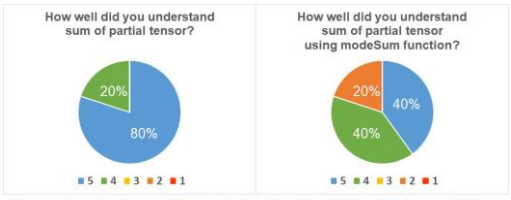

Fig. 10. Questionnaire results in Exercise II.

3) R programming Exercise III

Next, the learning content in this exercise is *n*-mode matrix unfolding. Fig. 11 shows a video explaining the implementation of the matrix unfolding on a 3rd order tensor using the unfold function.

In this learning material, as described in Chapter 1, students' understanding is improved by using CG animations for explaining the tensor operation which is difficult for the students to understand. For example, since it is difficult to understand the process of *n*-mode matrix unfolding, we created an animation as shown in Fig. 12 using Blender [15], which is 3DCG modeling software. The process of 2-mode matrix unfolding is shown in this figure. Furthermore, also in Exercises IV to VI below, process of tensor operations is

explained using CG animation.

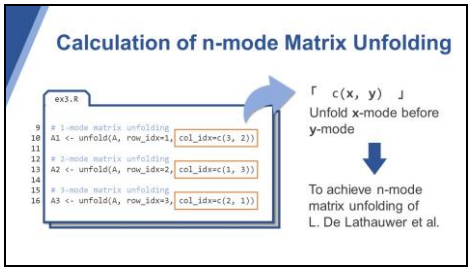

Fig. 11. Explanation of matrix unfolding calculation.

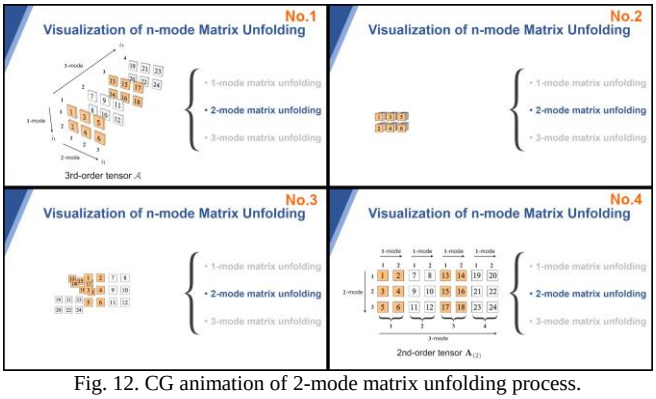

Fig. 12. CG animation of 2-mode matrix unfolding process.

Figs. 13 and 14 are parts of the exercise tasks in this material. Fig. 13 is a task that fills in elements and modes when matrix unfolding is applied to the presented 3rd order tensor. Here, the modified MacMahon's cube in Fig. 3 is used as the material of the tensor. In this task, a Word file for answering is attached with this thread, and students can download the file, put the answer, and submit it. Fig. 14 shows a task of implementing *n*-mode matrix unfolding from the definition without using the unfold function. As for exercises, there are a basic task and an applied task. Figs. 13 and 14 correspond to the basic one and the applied one, respectively. For the applied tasks, hints are provided below the question text as shown in Fig. 14.

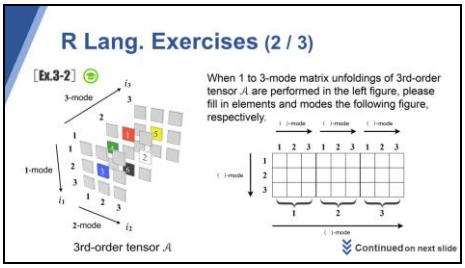

Fig. 13. Basic task of Exercise III.

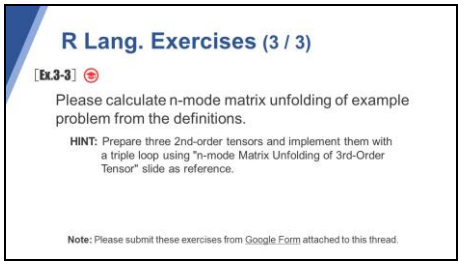

Fig. 14. Applied task of Exercise III.

Next, the results of this exercise are described. For the task in Fig. 13, all the students could correctly answer the elements and modes without reference to the answer example. In addition, four students implemented the task in Fig. 14 without viewing the answer example, however one student did the task with doing it. Furthermore, Fig. 15 shows the results of the comprehension questionnaire in this exercise. In this questionnaire, 80% of students answered that their understanding of each question was 4 or higher. From the above exercise results on matrix unfolding, it is considered that students had a good understanding of the definition and algorithm and the implementation using the unfold function.

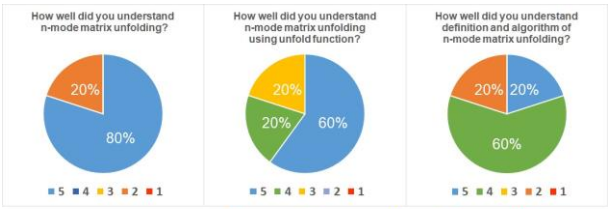

Fig. 15. Questionnaire results in Exercise III.

4) *R programming Exercise IV*

After learning the unfolding, students learn about folding operation. Firstly, they perform a basic task of folding matrices into tensors using the fold function, and then submit the result. It is confirmed from the submitted files that all of them could implement the task. Next, they work on a task folding a matrix of an unfolded MacMahon's cube into a 3rd order tensor. In this task, everyone could correctly answer how the elements of the matrix were arranged on the tensor by folding without viewing the example solution. Furthermore, we gave them an applied task of implementation based on the definition without using the fold function. As a result, three students could implement it without viewing the answer example, and the remaining two could do it with reference to the example.

The results of comprehension questionnaire of this exercise are shown in Fig. 16. For each question, more than 80% of students answered that their understanding level was 4 or higher. Therefore, from the above results, it is considered that they well understood about the folding algorithm and its implementation.

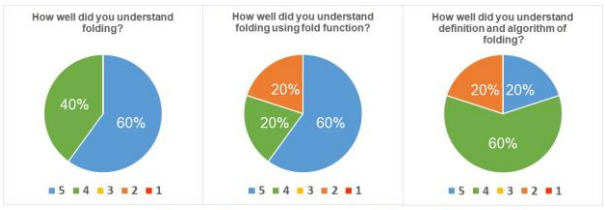

Fig. 16. Questionnaire results in Exercise IV.

5) *R programming Exercise V*

This exercise is for learning about *n*-mode product. At a beginning of this exercise, the definition and the property of the *n*-mode product are explained. Fig. 17 is part of a video describing the definition of 1-mode product of a 3rd order tensor.

However, it is considered difficult for first-time learners to calculate the *n*-mode product from the definition formula shown in Fig. 17. Therefore, as shown in Fig. 18, a method of calculating the *n*-mode product is described using Algorithm

1 in Chapter 2, which is considered easier for them to understand.

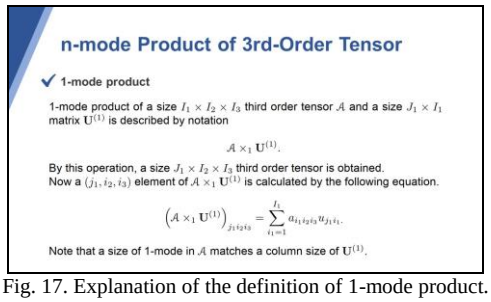

Fig. 17. Explanation of the definition of 1-mode product.

Fig. 19 shows a basic task of calculating the *n*-mode product of a 3rd order tensor and a matrix. As an example of the tensor, the MacMahon's cube is used. Furthermore, Fig. 20 shows an applied task of implementing the *n*-mode product from the definition without using ttm function.

Next, results of this exercise are described below. Regarding the task in Fig. 19, there were three students who implemented it without referring to the answer example, however one of them was incorrect due to a minor mistake. Remaining two students did the task with reference to it. For the task in Fig. 20, two students could implement it without referring to the answer example, and three students did it by referring to it. From this, it is found that difficulty of this task tends to be relatively high.

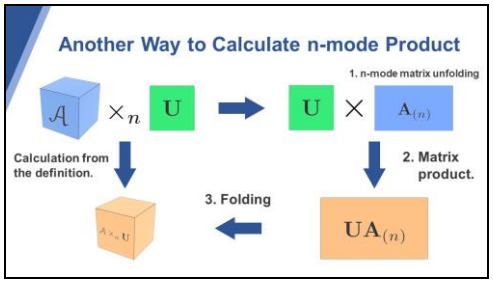

Fig. 18. Explanation of another calculation method of n-mode product.

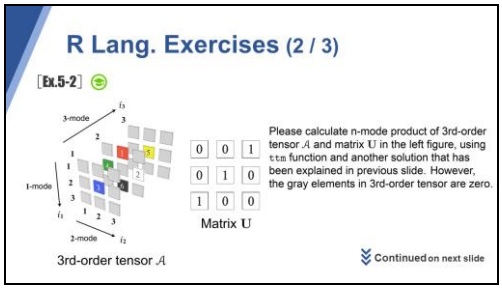

Fig. 19. Basic task of Exercise V.

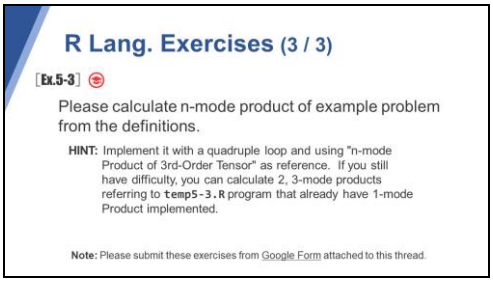

Fig. 20. Applied task of Exercise V.

Fig. 21 shows some of results of the comprehension questionnaire in this exercise. In each questionnaire, it is found that more than 60% of students evaluate their understanding as 3 points or more.

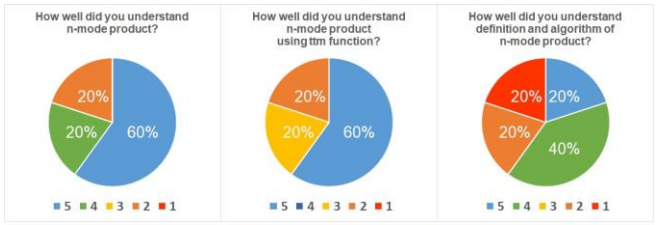

Fig. 21. Questionnaire results in Exercise V.

From these things, it is considered that they generally understand the definition of the *n*-mode product, its algorithm, and the implementation using the function.

1) *R Programming Exercise VI*

In this thread, the exercise on HOSVD is conducted. This exercise first provides an overview and a definition of HOSVD. Fig. 22 shows a portion of the video explaining the definition.

In this exercise, how to calculate with and without the hosvd function can be also learned. The method without it is based on the algorithm shown in Fig. 23 using the operations learned in the previous exercises III to V.

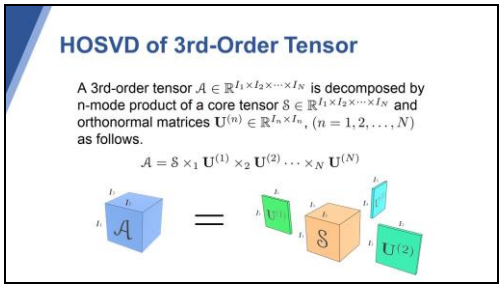

Fig. 22. Explanation of HOSVD definition.

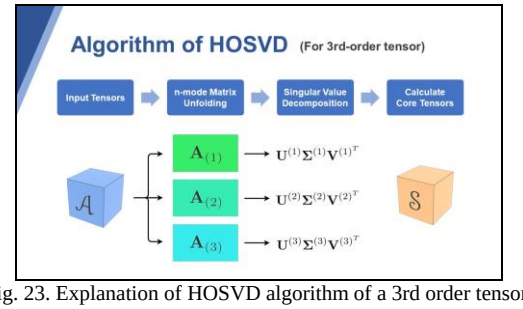

Fig. 23. Explanation of HOSVD algorithm of a 3rd order tensor.

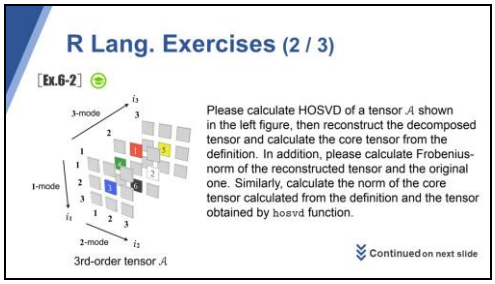

Fig. 24. Basic task of Exercise VI.

Fig. 24 is the basic task of calculating HOSVD, a reconstructed tensor, and residual norms of tensors using the hosvd and fnorm functions. In this task as well, the MacMahon's cube is used. Moreover, the problem shown in

Fig. 25 is the applied task of calculating HOSVD without using the function. This is implemented with reference to the algorithm described in Fig. 23.

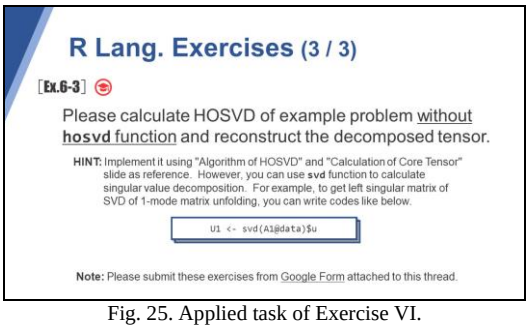

Fig. 25. Applied task of Exercise VI.

Next, status of implementation of this exercise is described. For the problem in Fig. 24, one student implemented it with reference to the answer example, however the other students could implement it without reference to it.

As for the problem in Fig. 25, two students could complete it without referring to the answer example, however remaining students performed it with reference to it. Therefore, it was found that the difficulty level of this problem was relatively high.

Fig. 26 shows results of the comprehension questionnaire of Exercise VI. In each questionnaire, 60% of students rated their comprehension as 3 or higher.

From the above situation and results, it is considered that the outline of HOSVD and the implementation using the function are generally understood.

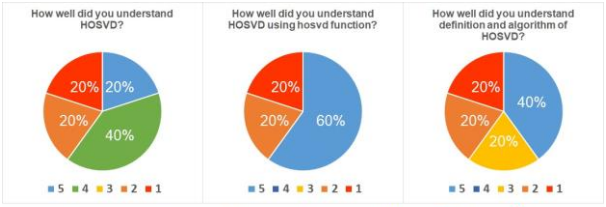

Fig. 26. Questionnaire results in Exercise VI.

C. Summary of Exercise Implementation Status

In this section, a summary of implementation status of each exercise is described. As shown in Fig. 27, 14 problems from Ex.1-1 to Ex.6-3 are given in this learning material. These results are summarized on a three-point scale based on A to C described in Fig. 27.

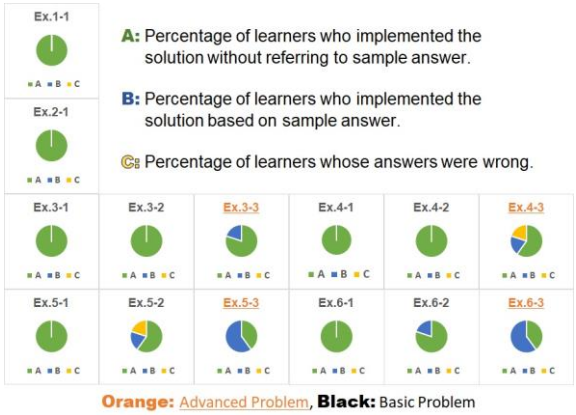

Fig. 27. Results of exercise status.

From the results, as for the basic problems, most of the students are rated A. Therefore, it can be found that they can solve them well. On the other hand, regarding advanced problems, many students are evaluated as B especially for Ex.5-3 and Ex.6-3. Thus, these are considered difficult for students because these are implemented based on the definitions and algorithm of *n*-mode product and HOSVD.

IV. Conclusions

This paper described the online learning materials developed for students to understand the theory and programming of HOSVD, which is one of the tensor decompositions. These learning materials are organized on Microsoft Teams, and students can proceed with learning and programming exercises on demand while watching videos. We used the 3D puzzle and MacMahon's cube for the exercises and expressed them in tensors so that the students could study with interest. Furthermore, in order to have a detailed understanding of the data processing process, we also created and used CG animations for explanation using Blender.

The following is a list of what we founded from the results of having our undergraduate students try these materials.

(1) Looking at the results of the exercises, about 60-100% of the students were able to perform the basic tasks of each exercise without referring to the answer examples. These tasks use the functions provided in the rTensor package, and it can be said that most students have mastered the use of those functions.

(2) On the other hand, about 40 to 80% of the students were able to perform the applied tasks without viewing the answer examples. These tasks are more difficult than the basic ones because they must be implemented without using packaged functions. Therefore, they should be optionally performed by students who want to solve more advanced tasks.

(3) It was found that it took about 3 to 6 hours from watching the explanation video to performing the exercises and finally answering the questionnaire.

(4) There were some students who practiced using R Studio Cloud [16], which is a cloud environment. From this, it was found that using a browser allows students to practice independently of the specifications of the machine.

(5) Several students commented that the explanation by CG animation was easy to understand. From this, it is considered that the animation created by Blender is useful for understanding the contents.

Based on these findings, we plan to continue improving the learning materials. In particular, we will immediately address the following points.

- Encourage many students to use this material and learn tensor decomposition.
- Motivate students to learn tensor decomposition by adding an explanation of the necessity and significance of it to our learning materials.
- Expand this learning materials so that programming exercises using application software libraries such as TensorFlow [17] can be performed.

Tensor decomposition is an important topic in fields such as big data processing and machine learning, and we think that it should be taught to undergraduate level students studying computer and data science. As mentioned in this paper, it is at a level that can be fully understood by undergraduate students. Therefore, we would like to improve our learning materials so that it can be widely used for that purpose.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Shota Abe implemented the learning materials and created the drawings of the manuscript; Akio Ishida shared the writing of the manuscript of the paper and made the final compilation; Jun Murakami shared the writing of the manuscript and oversaw the research plan; Naoki Yamamoto supervised the implementation of learning materials, shared the writing of the manuscript, and check the whole one; all authors had approved the final version.

References

[1] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455-500, 2009.
[2] Rikkyo University Syllabus. [Online]. Available: https://sy.rikkyo.ac.jp/timetable/stop.do.
[3] Chuo University Syllabus. [Online]. Available: https://syllabus.chuo-u.ac.jp/syllabus/.
[4] Kyoto University Syllabus. [Online]. Available: https://ocw.kyoto-u.ac.jp/syllabus/.
[5] University of Illinois Online Catalog. [Online]. Available: https://www.online.uillinois.edu/catalog/OnlineDegrees.asp
[6] MIT OpenCourseWare. [Online]. Available: https://ocw.mit.edu/
[7] A. Ishida, N. Yamamoto, J. Murakami, and N. Oishi, "Solving 3-D puzzles using tensor decomposition and application to education of multidimensional data analysis," *International Journal of Machine Learning and Computing*, vol. 8, no. 5, pp. 447-453, 2018.
[8] N. Yamamoto, A. Ishida, N. Oishi, and J. Murakami, "Development of teaching tool for supporting understanding of tensor decomposition using MacMahon's coloured cubes," *International Journal of Information and Education Technology*, vol. 10, no. 1, pp. 14-19, 2020.
[9] N. Yamamoto, A. Ishida, K. Ogitsuka, N. Oishi, and J. Murakami, "Development of online learning material for data science programming using 3D puzzle," *International Journal of Information and Education Technology*, vol. 11, no. 4, pp. 154-163, 2021.
[10] L. D. Lathauwer, B. Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253-1278, 2000.
[11] Lars Eldén, "Matrix methods in data mining and pattern recognition," *SIAM*, ch. 8, 2007.
[12] J. Beineke and J. Rosenhouse, *The Mathematics of Various Entertaining Subjects (Volume 2): Research in Games, Graphs, Counting, and Complexity*, Princeton University Press, 2017.
[13] R. C. Team, "R: A language and environment for statistical computing," *R Foundation for Statistical Computing*, Vienna, Austria.
[14] J. Li, J. Bien, and M. T. Wells, "rTensor: An R package for multidimensional array (tensor) unfolding, multiplication, and decomposition," *Journal of Statistical Software*, vol. 87, no. 10, pp. 1-31, 2018.
[15] L. Flavell, *Beginning Blender: Open Source 3D Modeling, Animation, and Game Design*, Apress, New York, 2010.
[16] J. M. Rosenberg, A. Edwards, and B. Chen, "Getting messy with data: Tools and strategies to help students analyze and interpret complex data sources," *The Science Teacher*, vol. 87, no. 5, pp. 30-34, 2020.
[17] N. Shukla, *Machine Learning with TensorFlow*, Manning Publications, 2018.

Shota Abe is currently an undergraduate student of the advanced course of electronics and information systems engineering, Kumamoto College, National Institute of Technology, Japan. His research interests are in the area of data analysis.

Akio Ishida received the M.S. and Ph.D. in science from Kumamoto University, Japan, in 2010 and 2014. He is currently an assistant professor of the Faculty of Liberal Arts, Kumamoto College, National Institute of Technology, Japan. His research interests are in the area of multi-dimensional data analysis and numerical calculation.

Jun Murakami received the Ph.D. in engineering from Toyohashi University of Technology, Japan, in 2000. He is currently a professor of the Faculty of Electronics and Information Systems Engineering, the Director of Library, Kumamoto College, National Institute of Technology, Japan. He is also a Board Member and Planning Chairman of the Research for Innovation and Synthesis of Technology in Kumamoto. His research area of interests includes statistical analysis, numerical calculation, and digital signal processing. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), and the Human Interface Society Japan (HISJ).

Naoki Yamamoto received the Ph.D. in engineering from Kyushu Institute of Technology, Japan, in 2001. He is currently a professor of the Faculty of Electronics and Information Systems Engineering, Kumamoto College, National Institute of Technology, Japan. His research interests are in the area of multidimensional data analysis and numerical calculation. He is a member of the Institute of Electronics, Information and Communication Engineers (IEICE) and Kyushu Society for Engineering Education.