Compression of Simulation Results Database Using Tensor Decomposition

Yixiang FENG*[1]

*1 Hitachi Research Laboratory
Hitachi, Ltd.
832-2 Horiguchi, Hitachinaka, Ibaraki 312-0034, JAPAN
yixiang.feng.cq@hitachi.com

Abstract

With the current trend of globalization, product design and development is often conducted collaboratively among several divisions, thus making it a huge challenge to transfer large CAE (Computer Aided Engineering) data between different locations. In this study, we have developed a data compression method based on TD (Tensor Decomposition). In this method, voxel simulation results data are represented as tensors and a tensor decomposition algorithm based on HOOI (Higher-Order Orthogonal Iteration) algorithm is applied to the tensors. After tensor decompression, the original tensor is decomposed into a core tensor and a series of basis matrices, whose summed size is considerably smaller than that of the original tensor. As a result, a compression ratio of over 60:1 is achieved for steady flow simulation results data and the error is below 5%. A compression ratio of over 70:1 is achieved for unsteady flow simulation results data and the error is below 5%. We have confirmed that at 5% error, no significant information is lost during the data compression process.

Keywords: CAE, product design, simulation, tensor decomposition, data compression

1 Introduction

Nowadays, CAE has become an indispensible tool for industrial product design and development. One of the trends with recent CAE is that models have become increasingly large-scaled. This is partly due to the increase in the functionality and complexity of products. The advancement in HPC (High Performance Computing) has made it possible to run large-scaled calculations in reasonably short time, which results in very large CAE results database. Meanwhile, in distributed computational environments such as the cloud systems, it is often necessary to transfer CAE results to client PCs for post-processing and/or visualization [1]. Since product design and development requires a speedy processing of CAE results data, it is important to reduce the data transfer time. To achieve this, it is important to compress the size of CAE results data before data transfer.

Besides the traditional mesh-based simulation, voxel –based simulation is often used to simulate actual industrial products due to the simplicity and robustness of grid generation [2], [3], [4]. In voxel simulations, the simulation models tend to be huge because of the fact that the voxel grids have to be divided at a high resolution to ensure precision [2]. Therefore, it is even more important to compress the voxel-based simulation results when transferring the database.

Data compression has long been studied in data-intensive disciplines such as image/video processing, signal processing, bioinformatics, *etc*. However, there were only few studies in the compression of CAE results data. In one of the studies, EZT (Embedded Zero-Tree) wavelet encoding method has been used to compress BCM (Building-Cube Method)-based CFD results data [5]. In another study, SVD (Singular Value Decomposition) method has been applied to particle simulation data [6]. Recently, high order SVD has been used to compress CFD results of the outer flow around a wing with hexahedral mesh [7].

Tensor decomposition, or tensor factorization, is the expansion of SVD to higher-dimensional arrays. In tensor decomposition, the original tensor is decomposed into a core tensor and several basis matrices whose total number is equal to the dimension of the input tensor [8][9][10]. **Figure 1** illustrates the image of tensor decomposition.

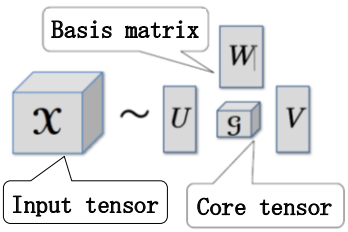

Fig. 1 An illustration of tensor decomposition

Tensor decomposition has been applied for image data compression [11], dimensionality reduction [12], and more recently for the compression of hexahedral mesh-based simulation result database [7]. However, to the best of our knowledge, there has been no application of tensor decomposition to the data compression of voxel-based simulation results.

In this study, we propose a data compression method for voxel simulation results based on tensor decomposition.

2 Tensor decomposition

2.1 Data representation using tensors

Simulation results data tend to be high-dimensional due to the fact that they have various design parameters and contain multiple physical quantities. Traditional

data compression methods such as SVD map the data into 2D plane and represent them as matrices [6]. It is obvious that the intrinsic correlation between dimensions will be lost during the mapping process. Therefore, we choose to use high-dimensional arrays, or tensors, to represent simulation results data. This way, the correlation between dimensions is preserved and high compression ratio can be expected by removing the redundancies in the data.

2.2 Tensor decomposition

Tensor decomposition can be formulated as an optimization problem of minimizing the distance between an input tensor and its approximate tensor of a lower rank. Mathematically, it is written as Eq. (1).

$$\begin{array}{ll} Given: & \mathcal{A} \in \Re^{I_1 \times I_2 \times .. \times I_N} \\ Minimize: & f(\hat{\mathcal{A}}) = \left\| \hat{\mathcal{A}} - \mathcal{A} \right\|_F \\ Subject\ to: & J_i < I_i, for\ i = 1 \sim N \end{array} \tag{1}$$

where $\mathcal{A}$ is the N^{th}-order input tensor consisting of real numbers. J_i is the rank of the i^{th}-mode.

Two popular tensor decomposition schemes are CP decomposition, where the core tensor is diagonal [13], and Tucker decomposition, where the core tensor is dense [14][15]. In this research, we use Tucker decomposition due to its flexibility in controlling the approximated tensor through adjusting the size of the core tensor. In Tucker decomposition, the approximated tensor $\hat{\mathcal{A}}$ is decomposed as shown in Eq. (2).

$$\hat{\mathcal{A}} = \mathcal{B} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \cdots \times_N \mathbf{U}^{(N)} \tag{2}$$

where $\mathcal{B}$ is the core tensor and $\mathcal{B} \in \Re^{J_1 \times J_2 \times .. \times J_N}$. $\mathbf{U}$ is the basis matrix and $\mathbf{U}^{(i)} \in \Re^{I_i \times J_i}$.

In this research, we adopt the HOOI (Higher-Order Orthogonal Iteration) tensor decomposition algorithm proposed by Lathauwer *et. al.* [16][17]. Using HOOI, the core tensor $\mathcal{B}$ is calculated using the following equation.

$$\mathcal{B} = \mathcal{A} \times_1 \mathbf{U}^{(1)^T} \times_2 \mathbf{U}^{(2)^T} \cdots \times_N \mathbf{U}^{(N)^T} \tag{3}$$

The basis matrix $\mathbf{U}$ is calculated using an ALS (Alternating Least Squares) method which is an expansion of the Least Squares method. The initial basis matrices are calculated using the HOSVD algorithm [16]. That is to say, $\mathbf{U}_0^{(i)}$ is calculated as the left singular vector of the i^{th} mode expansion of $\mathcal{A}$.

2.3 Tensor-decomposition-based data compression

One important property of tensor decomposition is that the overall size of the core tensor and basis matrices is usually much smaller than that of the original tensor. We take advantage of this property and use tensor decomposition for lossy data compression.

Compression ratio is defined as the number of elements before and after tensor decomposition.

$$CR_{Tensor} = \frac{\prod_{i=1}^{N} I_i}{\prod_{i=1}^{N} J_i + \sum_{i=1}^{N} (I_i \times J_i)} \tag{4}$$

The approximation error caused by tensor decomposition is defined as follows.

$$NORME = \frac{\left\| \hat{\mathcal{A}} - \mathcal{A} \right\|_F}{\left\| \mathcal{A} \right\|_F} \times 100\% \tag{5}$$

where *NORME* is the approximation error based on the Frobenius norm.

3 Test results

3.1 Test model

Figure 2 shows the voxel simulation model used for testing. In voxel simulation, the 3D simulation volume is automatically divided into orthogonal grids and then thermal or fluid simulations are performed. Our model is a simplified inverter model, which consists of fan unit, bus bar, resistors, *etc*. Thermal and fluid simulations are performed in the 3D volume containing these parts. The fan unit is regarded as fluid since it is the duct for air flow. The number of cross sections along the Z-axis is 251 and the number of voxels in the X and Y axes are 156 and 125, respectively. Therefore, the total number of voxels in the test model is 251*156*125=4,894,500. We performed simulation of air flow using in-house voxel simulation software and obtained results for eight physical quantities. **Table 1** shows a list of physical quantities of the test model.

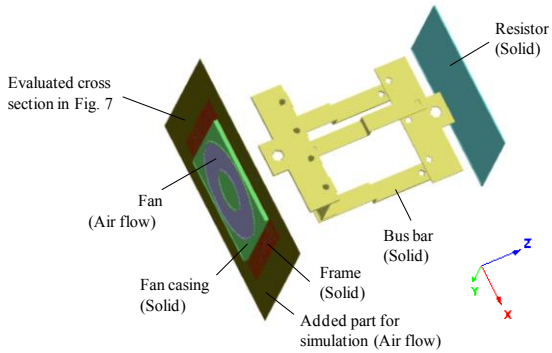

Fig. 2 Test model

Table 1 List of physical quantities of the test model

Physical quantity	Symbol	Unit
Humidity	h	%
Mass flow rate	m	Kg/s
Density	o	Kg/m3
Pressure	P	Pa
Temperature	T	K
Velocity component x	u	m/s
Velocity component y	v	m/s
Velocity component z	w	m/s

To test data compression of 4th–dimensional simulation data, we also perform unsteady flow simulation for 10 time steps.

All calculations are performed on an HP Z800 PC with CPU of Intel Xeon W5590 @ 3.33 GHz and 32GB physical memory.

3.2 Test results

First, we represent simulation results of one time step with a 3rd-order tensor and then perform tensor decomposition. The three dimensions of the 3rd-order tensor correspond to the X, Y and Z axes of the voxel model. For comparison, we also expand the input simulation results data to 2D matrix and perform SVD. **Figure 3** shows a comparison between TD and SVD in terms of compression ratio and approximation error. The X-axis is compression ratio and log scale is used. It is obvious that, at any error level, TD has higher compression ratios than those of SVD. For example, when approximation error is 5%, the compression ratio for SVD is about 5.4, while for TD is about 60.0, which is about 11 times over that of SVD.

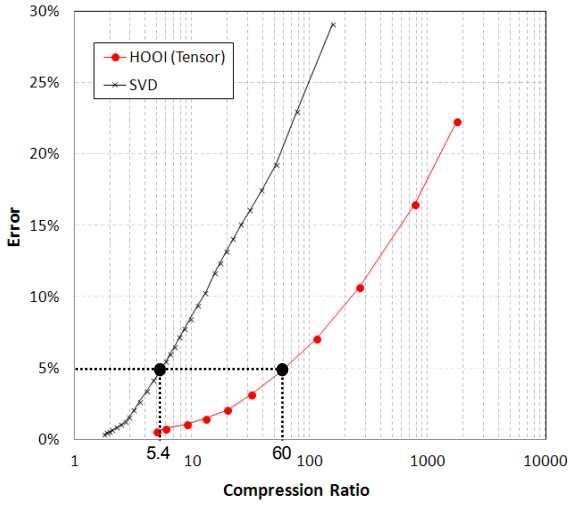

Fig. 3 Comparison between TD and SVD (3rd-order)

Next, we represent the 10-step time series of voxel simulation results with a 4th-order tensor and then perform tensor decomposition to compress the data. The four dimensions of the 4th-order tensor correspond to the X, Y, Z axes and t (time) of the voxel model. **Figure 4** shows a comparison between TD and SVD in terms of compression ratio and approximation error. The X-axis is compression ratio and log scale is used. Similar with the results of the 3rd-order tensor, when error level is the same, TD gains higher compression ratios than SVD does. For example, when error is 5%, the compression ratio for SVD is about 4.0, while for TD it is about 70.0, which is about 17 times over that of the SVD. Comparing with the results of 3rd-order tensor, the compression ratio for 4th-order tensor is higher than that of the 3rd-order tensor, which suggests that TD is suitable for the compression of large-scale and high-dimensional database.

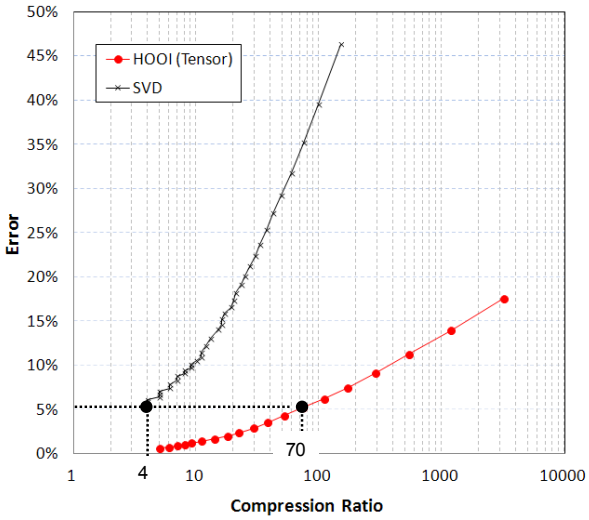

Fig. 4 Comparison between TD and SVD (4th-order)

3.3 Computational time

Figure 5 shows the comparison of computational time between TD and SVD in terms of compression ratio.

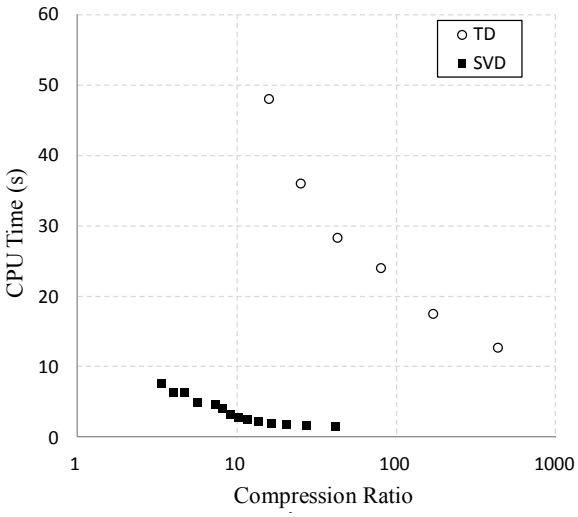

Fig. 5 Comparison of CPU time in terms of CR

As shown in **Figure 5**, the computational time of TD is about an order greater than that of SVD, which is due to the complexity of the HOOI algorithm. The HOOI algorithm requires SVD to be calculated in each mode and iterates until the ALS algorithm converges [17].

It can also be seen from **Figure 5** that TD and SVD operate at different zones. TD is more time-consuming, but the compression ratio is much higher than that of the SVD. However, it is not sufficient to look at the compression ratio alone, because there is a tradeoff between the compression ratio and approximation error.

Figure 6 shows a comparison of computational time between TD and SVD in terms of approximation error.

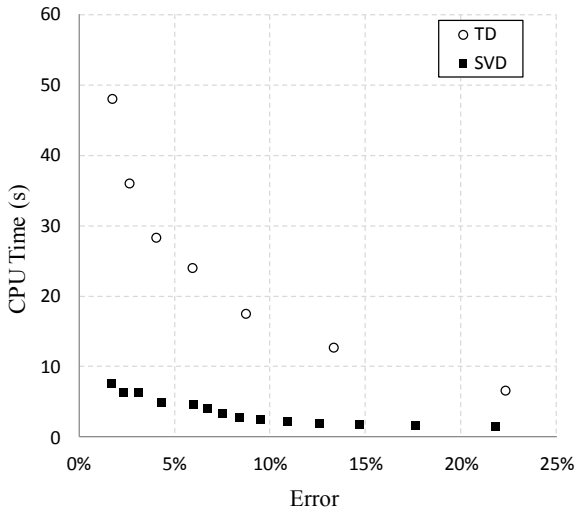

Fig. 6 Comparison of CPU time in terms of error

It can be seen from **Figure 6** that the computational time for TD is considerably greater than that for SVD. When the approximation error is 5%, the computational time for TD is about 25s, while for SVD it is about 5s. It should be noted that when the compression error is the same, the compression ratio with TD is much greater than that of SVD, as shown in **Figure 3** and **Figure 4**. We conclude that there is a tradeoff between the performance and computational time.

4 Discussion

To further investigate the information loss caused by data compression, we compare the data before and after compression by visualizing the data. **Figure 7** shows the pressure distributions in the cross section of the air flow inside the fan unit. The core sizes, compression ratio and error are listed in **Table 2**. It can be seen from **Fig. 7** and **Table 2** that, with the increase in core tensor size, the pressure distribution in the cross section gradually approximates that of the original data. When the approximation error is 10.7%, the overall feature of the pressure distribution is captured, but there are still noticeable differences in the color tone and some other details. Meanwhile, when the approximation error is decreased to 4.8%, there is no noticeable difference between the compressed data and the original data, which suggests that the compressed data can be used for further analysis or design.

We also compare the data before and after data compression of the unsteady flow simulation results. **Figure 8** shows pressure distributions in the cross section of the air flow inside the fan unit, which is the same cross section as used in **Figure 7**. In **Figure 8**, the n^{th} row corresponds to the simulation results of the n^{th} time step, where n is a number between one and ten. The first column is the original data without data compression. The second column is the data after data compression with TD with a compression ratio of 3174.

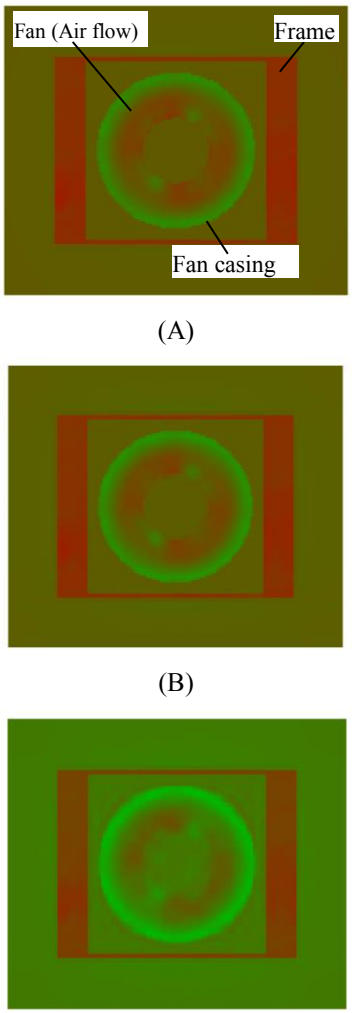

Fig. 7 Comparison of cross sections

Table 2 List of physical quantities of the test model

Data	Core size	CF	Error
(A)	Original data		
(B)	40*40*40	57	4.8%
(C)	20*20*20	263	10.7%

The third column is the data after data compression with TD with a compression ratio of 52. The fourth column is the data after data compression with SVD with a compression ratio of 10. It can be observed from **Figure 8** that, at compression ratio 52, the compressed data from TD yields almost the same pressure distributions as those from the original data. Meanwhile, for the compressed data from SVD, even at compression ratio 10, there are considerable differences between the compressed data and the original data. Even though the overall pressure distributions look similar, the data compressed from SVD lose many of the detailed characteristics in the pressure distribution. Therefore, it is confirmed that data compression using TD outperforms data compression using SVD.

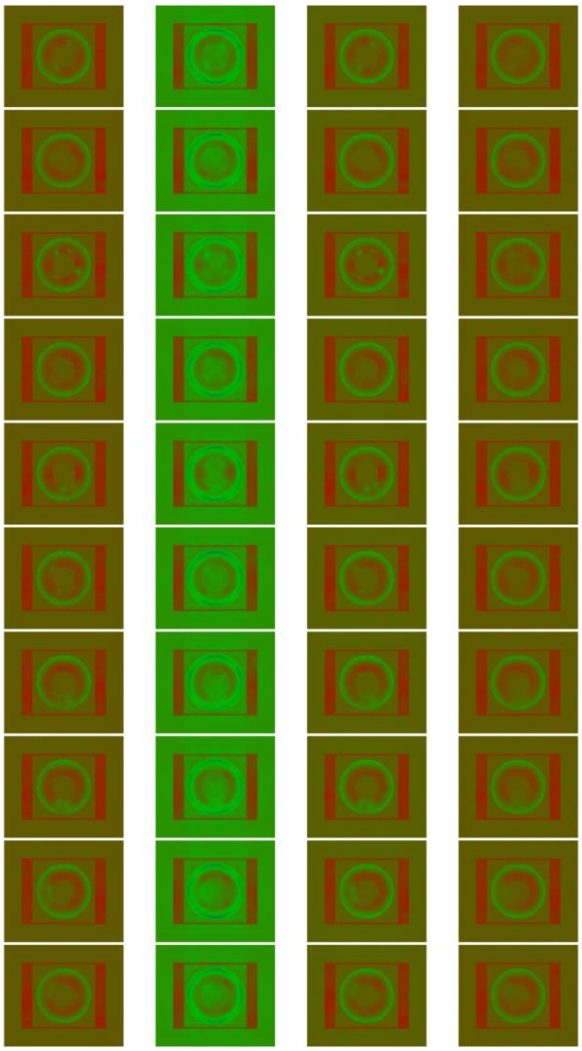

Fig. 8 Comparison of data compression results for unsteady flow simulation results data
Column 1: Original data;
Column 2: TD compressed data, CF=3174;
Column 3: TD compressed data, CF=52;
Column 4: SVD compressed data, CF=10.

5 Conclusions

We propose a data compression method for simulation results database using tensor decomposition. We test the method in a simplified inverter model and obtain the following conclusions.

(1) We represent single voxel simulation results data with 3rd-order tensor and decompose it. As a result, we obtain a compression ratio of 1:60 while the approximation error is about 5%. Compared with traditional method in which the compression ratio is about 5.4, the compression ratio of our method is more than 10 times higher.

(2) We represent a time-series of voxel simulation results data with 4th-order tensor and decompose it. As a result, we obtain a compression ratio of 1:70 while the approximation error is 5%. The proposed method is over 17 times higher in terms of compression ratio as compared with traditional method in which the value is about 1:4.

(3) Computation time for TD is greater than that for SVD. There is a tradeoff between performance and computational time.

References

[1] Matsuoka, D. and Araki, F., "Survey on Scientific Data Visualization for Large-scale Simulations", JAMSTEC Report of Research and Development, Vol. 13, (2011), pp. 35-63. (in Japanese)

[2] Tawara, T. and Ono, K., "Fast large scale voxelization using a pedigree", the 10th ISGG Conference on Numerical Grid Generation, Sep. 16-20, Forth, Crete, Greece, (2007).

[3] Ikegawa, M., Mukai, H., and Watanabe, M., "Airflow-simulation by voxel mesh method for complete hard disk drive structure", IEEE Trans. Magn. Vol. 45, No. 11, (2009), pp. 4918-4922.

[4] Hayashi, S., Watanabe, M., Iwase, Y., Kanno, K., and Fujimori, K., "Development of a household vacuum cleaner with a new cyclone dust collector", FEDSM2007-37014, (2007), pp. 1925-1932.

[5] Sakai, R., Sasaki, D., Obayashi, S. and Nakahashi, K., "Wavelet-based data compression for flow simulation on block-structured Cartesian mesh", International Journal for Numerical Methods in Fluids. Vol. 73, Issue 5, (2013), pp. 462–476.

[6] Wada, K. and Iwasaki, K., "Compression of Particle-based Fluid Simulation Data", Information Processing Society of Japan, Kansai Branch, (2011). (in Japanese)

[7] Lorente, L. S., Vega, J. M. and Velazquez, A., "Compression of aerodynamic databases using high-order singular value decomposition", Aerospace Science and Technology, Vol. 14, No. 3, (2010), pp. 168-177.

[8] Kolda, T. G. and Blder, B. W., "Tensor decomposition and applications", SIAM Review, Vol. 51, No. 3, (2009), pp. 455-500.

[9] Acar E. and Yener B., "Unsupervised multiway data analysis: a literature survey", IEEE Transactions on knowledge and data engineering", Vol. 21, No. 1, (2009), pp. 6-20.

[10] Qi, L., Sun, W., and Wang, Y., "Numerical multilinear algebra and its applications", Front. Math. China, Vol. 2, No. 4, (2007), pp. 501-526.

[11] D. Vlasic, M. Brand, H. Pfister, and J. Popovic, "Face transfer with multilinear models", ACM Trans. Graphics, Vol. 24 (2005), pp. 426-433.

[12] Wang, H. and Ahuja, N., "A tensor approximation approach to dimensionality reduction", Int J Comput Vis, Vol. 76, (2008), pp. 217-229.

[13] R.A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal factor analysis", UCLA Working Papers in Phonetics, Vol. 16 (1970), pp. 1-84.

[14] L.R. Tucker: The extension of factor analysis to three-dimensional matrices, in Contributions to Mathematical Psychology, H. Gulliksen and N. Frederiksen, eds., Holt, Rinehart & Winston, New York, (1964), pp. 109-127.

[15] L.R. Tucker: Some Mathematical Notes on Three-Mode Factor Analysis. Psychometrika, Vol.31,

No.3, (1966), pp. 279-311.
[16] Lathauwer, L. D., Moor, B. D. and Vanderwalle, J., “A multilinear singular value decomposition”, SIAM J. Matrix Anal. Appl., Vol. 21, No. 4, (2000), pp.1253-1278.
[17] Lathauwer, L. D., Moor, B. D. and Vanderwalle, J., “On the best rank-1 and rank-(R1,R2,..., RN) approximation of higher-order tensors”, SIAM J. Matrix Anal. Appl., Vol. 21, No. 4, (2000), pp.1324-1342.

Received on October 11, 2013
Accepted on February 4, 2014