

**POLITECNICO DI MILANO**  
**Industrial and Information Engineering School**  
**Master Degree of Electrical Engineering**



**TENSOR BASED PROCESSING OF ELECTRICAL GRID  
DATA**

Relatore: Prof. Martin Haardt

Correlatore: Prof. Alessandro Ferrero

Tesi di Laurea Magistrale di:  
Danial Jafarigiv  
Matr. 803647

Academic Year 2014-2015

## **Acknowledgments**

First of all, I owe my deepest gratitude to my advisor Prof. Ferrero for the continuous support of my master thesis and related research, for his patience, motivation, and immense knowledge. His encouragement helped me in all the time of research and writing of this thesis.

My sincere thanks also goes to Prof. Haardt and Mrs. Naskovska at Ilmenau University of Technology who provided me an opportunity to join their team as intern, and who gave access to the research facilities. I have acquired from their expertise in the fields of array signal processing.

A special thanks to my family. Words cannot express how appreciative I am to my mother, and father for all of the sacrifices that they have made on my behalf. I also would like to thank my brother and sister for supporting me spiritually throughout writing this thesis. At the end I would like express my sincere gratitude to my beloved love Haniyyeh who always support me in the moments when there was no one to answer my queries.

---

# INDEX

<b>INDEX</b> .....	1
<b>INDEX OF FIGURE</b> .....	4
<b>INDEX OF TABLE</b> .....	6
<b>ABSTRACT</b> .....	7
<b>1 INTRODUCTION</b> .....	9
<b>2 HIGHER ORDER TENSOR</b> .....	12
2.1 NOTATION AND PRELIMINARIES.....	12
2.1.2 MATRIX REPRESENTATION OF A HIGHER ORDER TENSOR .....	14
2.1.3 RANK PROPERTIES OF A HIGHER ORDER TENSOR .....	15
2.1.4 ORTHOGONALITY CONCEPT, SCALAR PRODUCT DEFINITION, AND NORM OF HIGHER ORDER-TENSORS.....	17
2.1.5 MATRIX KRONECKER, KHATRI-RAO, AND HADAMARD PRODUCTS.. .....	18
2.1.6 MULTIPLICATION AND PRODUCT OF A TENSOR BY MATRIX .....	19
2.2 A HIGH ORDER SINGULAR VALUE DECOMPOSITION.....	22
2.3 CANDECOMP/PARAFAC DECOMPOSITION.....	25
2.3.1 TENSOR RANK .....	27
2.3.2 UNIQUENESS .....	27
2.4 COMPUTING THE CP DECOMPOSITION.....	28

---

---

<b>3</b>	<b>ELECTRICAL GRID DATA ANALYSIS</b> .....	31
3.1	ELECTRICAL GRID DAT.....	31
3.2	PREPORCESSING.....	34
3.2.1	SELECT INTERESTING CHUNK.....	35
3.2.2	INTERPOLATING MISSING VALUES.....	36
3.3	EVALUATION OF THE BENEFITS OF USING TENSOR BASED PROCESSING.....	37
3.4	RESHAPE TIME AXIS.....	41
3.5	ANALYZING THE DATA.....	45
3.6	PREPORCESSING OF THE SECOND CASE.....	48
3.6.1	SELECT INTERESTING CHUNK.....	49
3.6.2	INTERPOLATING MISSING VALUES.....	49
3.7	RESHAPE TIME AXIS.....	50
3.8	ANALYZING THE DATA.....	52
<b>4</b>	<b>WEGHTED CP DECOMPOSITION DATA RECOVERY</b> .....	55
4.1	FACTORIZATIONS IN PRESENCE OF MISSING DATA.....	60
4.1.1	TENSOR FACTORIZATIONS.....	60
4.2	CP WEIGHTRD OPTIMIZATION ALGORITM.....	60
4.2.1	WEIGHTED OBJECTIVE FUNCTION.....	61
4.2.2	GRADIENT.....	62
4.3	NONLINEAR CONJUGATE GRADIENT.....	63
4.4	EXPERIMENTS.....	65
4.4.1	SIMULATED DATA.....	65

---

---

4.4.2	CAPTURING ORIGINAL STRUCTURE OF POWER GRID DATA.....	70
<b>5</b>	<b>CONCLUSION</b> .....	<b>74</b>
<b>6</b>	<b>REFERENCES</b> .....	<b>76</b>
<b>7</b>	<b>APPENDICES</b> .....	<b>78</b>

---

## INDEX OF FIGURE

<b>Figure 1:</b> Fibers of a 3rd-order tensor.....	13
<b>Figure 2:</b> Slices of a 3rd-order tensor.....	13
<b>Figure 3:</b> Unfolding of the $(I_1 \times I_2 \times I_3)$ -tensor $A$ .....	15
<b>Figure 4:</b> Tensor rank visualization.....	16
<b>Figure 5:</b> Visualization of the multiplication of a third-order tensor.....	21
<b>Figure 6:</b> Visualization of the matrix SVD.....	23
<b>Figure 7:</b> Visualization of the HOSVD for a third-order tensor.....	24
<b>Figure 8:</b> Matrices SVD vs tensor HOSVD.....	25
<b>Figure 9:</b> CP decomposition of a three-way array.....	26
<b>Figure 10:</b> ALS algorithm to compute a CP decomposition.....	30
<b>Figure 11:</b> EU energy network.....	32
<b>Figure 12:</b> Hourly sample rates for the 1000 channels.....	34
<b>Figure 13:</b> Interesting chunk.....	35
<b>Figure 14:</b> Interesting chunk.....	36
<b>Figure 15:</b> Channel 151 interpolating.....	37
<b>Figure 16:</b> 10*10*10 random tensor.....	40
<b>Figure 17:</b> 10*10*10 random tensor rank 5.....	40
<b>Figure 18:</b> 10*10*10 random tensor rank 5+ noise (10 dB SNR).....	41
<b>Figure 19:</b> R-D data representation.....	42
<b>Figure 20:</b> Singular values.....	43
<b>Figure 21:</b> Distortion vs compression (3-D, 4-D tensor).....	44
<b>Figure 22:</b> Time signature of dominant (rank-one) component 2D.....	46
<b>Figure 23:</b> Time signature of dominant (rank-one) component 3D.....	46
<b>Figure 24:</b> Time signature of dominant (rank-one) component 4D.....	47
<b>Figure 25:</b> Daily time Signature.....	47
<b>Figure 26:</b> Interesting chunk.....	48

---

---

<b>Figure 27:</b> Interesting chunk.....	49
<b>Figure 28:</b> Singular values.....	50
<b>Figure 29:</b> Distortion vs compression (3-D,4-D tensor).....	51
<b>Figure 30:</b> Time signature of dominant (rank-one) component 2D.....	53
<b>Figure 31:</b> Time signature of dominant (rank-one) component 3D.....	53
<b>Figure 32:</b> Time signature of dominant (rank-one) component 4D.....	54
<b>Figure 33:</b> CP-WOPT computation of function value $f^w$ and gradient $G^{(n)} = \frac{\partial f^w}{\partial A^{(N)}}$ .....	62
<b>Figure 34:</b> Accuracy versus D.....	69
<b>Figure 35:</b> Interesting channels.....	71
<b>Figure 36:</b> Reconstruction error.....	71
<b>Figure 37:</b> Time signature of dominant (rank-one) component 4D.....	73
<b>Figure 38:</b> Time signature of dominant (rank-one) component 3D.....	73

---

# INDEX OF TABLE

**Table 1:** Hourly sample rate for the bus and generator number 1.....33  
**Table 2:** Conjugate direction updates available.....64  
**Table 3:** Accuracy for randomly missing entries.....67  
**Table 4:** Accuracy for randomly missing fibers.....67



## ABSTRACT

*The problem of missing data is appearing in some areas such as electrical grid data, biomedical signal processing, traffic network analysis, social network services, image processing and communication systems in which data set is aim to uncommon errors. Moreover, these data collections can be quite large and have more than two axes of variation, e.g., frequency, amplitude, time. Many applications in those domains object to capture the underlying hidden structure of the data; in other words, factorizing data sets with missing entries get a better vision of data structures. If we cannot settle the issue of missing data, many important data sets will be discarded or improperly analyzed. By considering previous studies which have only reflected on matrices, we focus here on the problem of electrical grid data in multi-way arrays (tensors) because the data most of the time have more than two modes of variation and are therefore best represented as multi-way arrays. For instance, in electrical grid, data of each record from a bus can be represented as a time-bus matrix; thus, data from multiple channels is three-dimensional (time, bus, and voltage or power) and forms a three-way array. Therefore, we need a robust and reliable approach for factorizing multi-way arrays (i.e., tensors) in the presence of missing data. In this work we present a proper analysis of multi-ways power gird data to evaluate the benefits of using tensor-based processing by computing an approximate Canonical Polyadic decomposition and High Order Singular Value decomposition. After interpolation of missing data, there is a low-rank structure in form of a quite dominant rank-one component (periodic) on top of the rest of the data. Moreover, in the last chapter we employ one of the most applicable tensor factorizations, Canonical Polyadic Decomposition (CP), and work out the CP model as a weighted least squares problem that models only the known entries as the second method for factorizing the missing data. We apply an algorithm called Canonical Polyadic Weighted Optimization using a first-order optimization approach to solve the weighted least squares problem. Based on numerical experiments that we use over the sampled data, this algorithm is shown to successfully factor matrices with noise and up to 70% missing data. We then*

---

*illustrate the comparison results between these two methods over our power grid data which prove CP weighted optimization algorithm has a better factorization with respect to interpolation method.*

# 1 INTRODUCTION

Research scheme of signal processing problems increase with involvement the quantities of which the elements are arranged by more than two indices. In mathematics vector present a first order element and matrix apply for second order elements. Meanwhile, higher-order equivalents of vectors and matrices are called higher-order tensors, multidimensional matrices, or multiway arrays which has three or more indices. This notion generally referred to tensor fields in mathematics the same as physics and engineering. In some engineering domains such as electrical grid data, biomedical signal processing, power grid data analysis, image processing and mobile communication in which we involve multiway arrays, the utilizing framework of vector and matrix algebra demonstrates to be unfavorable and inappropriate.

Electrical grid which is defined as an interconnected network for delivering electricity from generating stations to power consumers. This grid is a combination of generating stations which produce electrical power, high-voltage transmission lines and buses that transfer power to demand centers, and also distribution lines and low voltage buses that connect home customers [12].

At each power bus we can read the power components such as active power, reactive power and voltage which measure the usable and necessary information to track, design, analyze, monitor and compensate electrical systems. Active and reactive powers are thus used to define system characteristics as power factor, installed power capacity, or load. In this work, we consider power grid data of buses in multi-way arrays framework as the data often have more than two modes of variation. For instance, in one bus, data of each record can be represented as a time-bus matrix; thus, data from multiple buses or channels is three-dimensional (time, bus, and voltage or power) and forms a three-way arrays. Further processes

techniques needed to evaluate the benefits of using power grid data as multi-way arrays.

An increasing electric transmission systems interconnections force to develop necessary methods for real-time monitoring and processing of the system variables and quantities. Fortunately, this mandatory demand is simultaneous with the advantage of more powerful and reliable computers and system processors and remarkable developments in analytical techniques which are the tools necessary to do this job [12].

We will employ a power system consist different buses which gain from using the data in multi-way arrays (tensor model). This kind of data framework provides a wide range of analytical techniques for further processing such as weighted least squares method which suppress bad data in power system state estimation based on the model discussed in [12],[13],[14] or adjust nonlinear iterative methods such as newton raphson for power flow calculation [15].

In this work, two analytical methods based on proper application of multiway arrays will be present. To a large extent, multiway arrays are currently starting to be considered important due to the advances in the field of signal processing [1]. A brief example of some opportunities offered by mutiway arrays gives an idea of the promising role of multi-way arrays in signal processing field [1].

In chapter one we aim to provide an overview of multi-way arrays, their characteristics and also their decomposition algortims. Search a data structure via high order singular value decomposition (HOSVD) and canonical polyadic decomposition to get an idea that there is some low-rank (tensor) structure in electrical grid data and gaining from using multi-way arrays are discussed in chapter two. Also interpolation method which recovers the missing data in multi-way arrays is the base of this chapter.

Moreover, In chapter three we consider a more general method of tensor factorization in the presence of missing electrical grid data. Formulating the canonical polyadic tensor decomposition for incomplete multi-way arrays as a weighted least squares problem and recovering the underlying matrix factors even

with high percent of missing data. We will discuss related work in matrix and tensor factorizations. We then compared the results between these two methods over our power grid data. Results proved that CP weighted optimization algorithm has a better factorization with respect to interpolation method.

## 2 Higher Order Tensors

### 2.1. Basic definitions

#### Mathematical Notation and Preliminaries

In this chapter, we have attempted to review tensor terms that would be applied in mathematics and engineering and became familiar with the terminology of previous works in the field of multi-way arrays. The *order* of a multi-way array (tensor) is the number of dimensions, also known as ways or modes. Vectors (tensors of order one) are denoted by lowercase letters, e.g.,  $\mathbf{a}$ . Matrices (tensors of order two) are denoted by boldface capital letters, e.g.,  $\mathbf{A}$ . Higher-order tensors (order three or higher) are denoted by boldface Euler script letters, e.g.,  $\mathcal{X}$ .

Scalars are defined by lowercase Euler script letters, e.g.,  $a$ . The  $i$ -th entry of a vector  $\mathbf{a}$  is denoted by  $a_i$ , element  $(i, j)$  of a matrix  $\mathbf{A}$  is indicated by  $a_{ij}$ , and element  $(i, j, k)$  of a third-order tensor  $\mathcal{X}$  is denoted by  $x_{ijk}$ .

Indices typically differ from 1 to their capital version, e.g.,  $i = 1, \dots, I$ .

The  $n$ -th element in a data series is denoted by a superscript in parentheses, e.g.,  $A^{(n)}$  indicates the  $n$ th matrix in a sequence.

Subarrays are formulated when a subset of the indices is fixed. For matrices, these are the rows and columns. A colon is used to define all elements of a mode.

Thus, the  $j$ th column of  $\mathbf{A}$  is indicated by  $\mathbf{a}_{:j}$ , and the  $i$ th row of a matrix  $\mathbf{A}$  is denoted by  $\mathbf{a}_i$ . These are the main notations that we employ in this work.

*Fibers* are the higher-order analogue of matrix rows and columns. A fiber is indicated by fixing every index but one could ambulate. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. Third-order tensors have column,

row, and tube fibers, denoted by  $x_{:jk}$ ,  $x_{i:k}$ , and  $x_{ij}$ , respectively; see Figure 1. Fibers are always supposed to be lied on column vectors direction by extracting from the tensor [2].

*Slices* are two-dimensional sections of a tensor, defined by fixing all indices but two indices can ambulate.

In Figure 2, we can easily exhibit the horizontal, lateral, and frontal slides of a third-order tensor  $X$ , indicated by  $x_{i::}$ ,  $x_{:j:}$ , and  $x_{::k}$ , respectively. Moreover, the  $k$ th frontal slice of a third-order tensor,  $x_{::k}$ , may be defined more compactly as  $x_k$ .

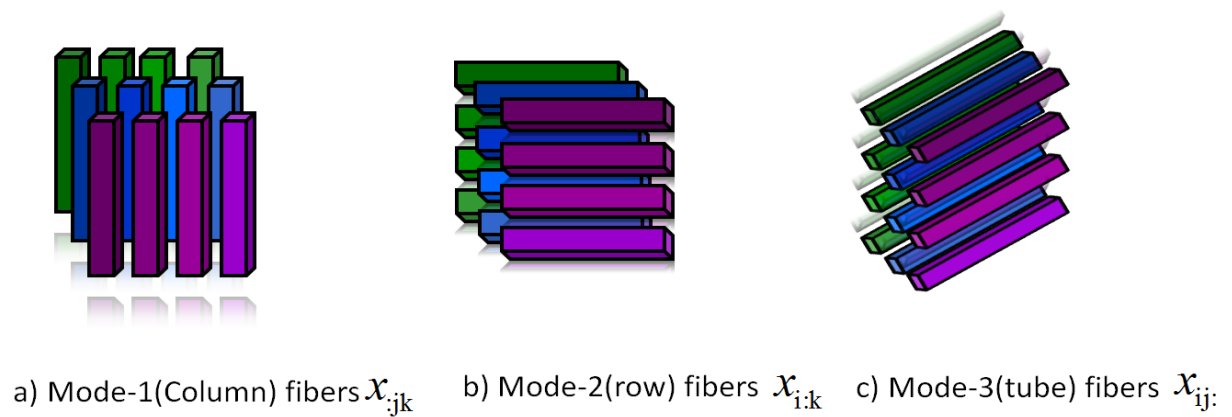


Figure 1. Fibers of a 3rd-order tensor

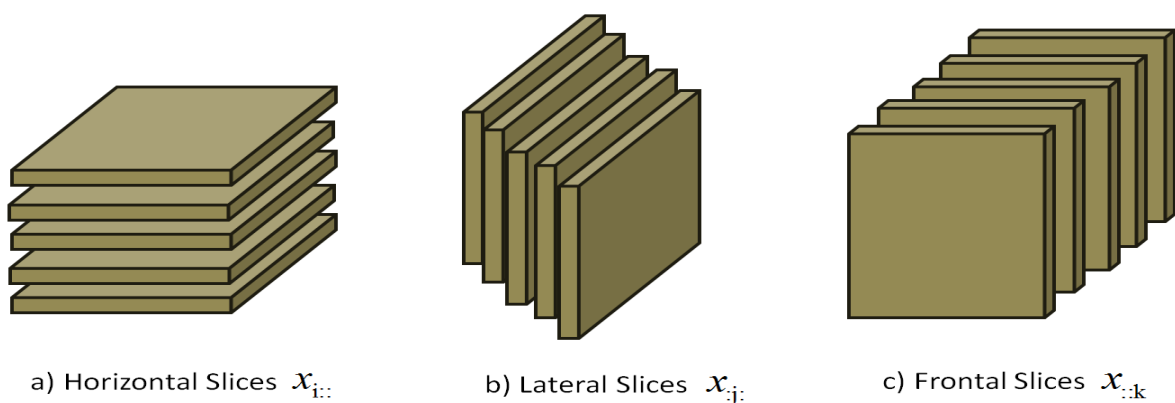


Figure 2. Slices of a 3rd-order tensor

The *norm* of a tensor  $X \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N}$  is the square root of the sum of the squares of all its elements, i.e.,

$$\|x\| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \dots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \dots i_N}^2}$$

### 2.1.2. Matrix representation of a higher-order tensor

High order singular value decomposition definition can be explained by considering a proper generalization of the interconnection between the column (row) vectors and the left or right singular vectors of a matrix.

This idea can be formulated if we define “matrix unfoldings” of a given tensor, i.e., matrix representations of that tensor in which all the column (row) vectors are accumulate one after the other. Matricization, also known as unfolding or flattening, is the process of rearranging the elements of an  $N$ -way array into a matrix.

Play it safe, here is an example of one particular ordering of the column (row, . . .) vectors; for order three, these unfolding procedures are illustrated in Figure 3. Consider that the definitions of the matrix unfoldings for the tensor dimensions  $I_1, I_2, I_3$  in a cyclic way and that, when involving with an unfolding of dimensionality  $I_c \times I_a I_b$ , we formally propose that the index  $i_b$  varies more faster than  $i_a$ .

In general, we have the following definition [3].

**Definition 1.** Assume an  $N$ th-order tensor  $A \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$ .

The matrix unfolding  $\mathbf{A}_{(n)} \in \mathbb{C}^{I_n \times (I_{n+1} I_{n+2} \dots I_N I_1 I_2 \dots I_{n-1})}$  contains the element  $a_{i_1 i_2 \dots i_N}$  at the position with row number  $i_n$  and column number equal to:

$$\begin{aligned} & (i_{n+1} - 1) I_{n+2} I_{n+3} \dots I_N I_1 I_2 \dots I_{n-1} + (i_{n+2} - 1) I_{n+3} I_{n+4} \dots I_N I_1 I_2 \dots I_{n-1} + \dots \\ & + (i_N - 1) I_1 I_2 \dots I_{n-1} + (i_1 - 1) I_2 I_3 \dots I_{n-1} + (i_2 - 1) I_3 I_4 \dots I_{n-1} + \dots + i_{n-1} \end{aligned}$$



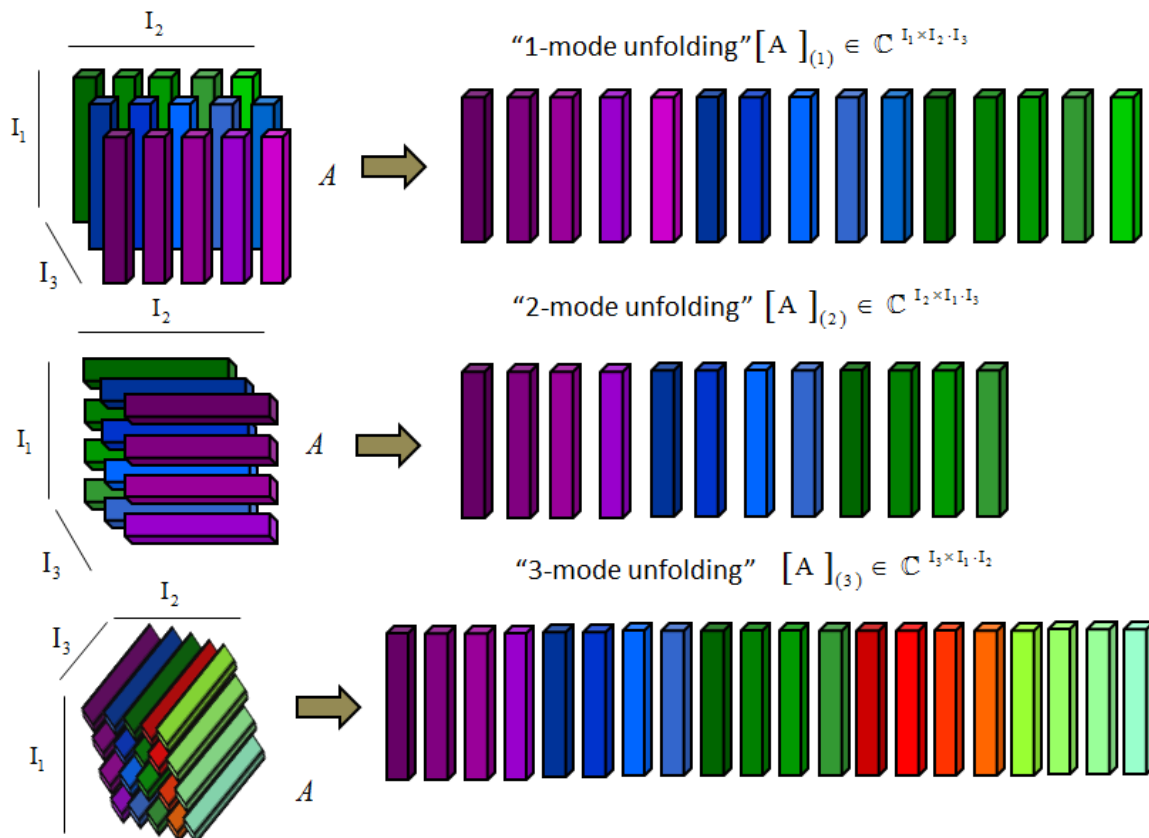


Figure 3. Unfolding of the  $(I_1 \times I_2 \times I_3)$ -tensor  $A$

### 2.1.3. Rank definitions of a higher-order tensor

There are major differences between matrices and higher-order tensors based on rank definitions and properties. Alternatively, as we will explain in section 1.3, these differences directly change the root an SVD generalization could obtain. As a matter of fact, there is not a unique and only one way to generalize the rank meaning.

The starting point is to generalize the d of column and row rank. To formulate this problem, if we refer in general to the row (column) vectors of an  $N$ th-order tensor or multi-way array  $A \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$  as its " $n$ -mode vectors," dedicated as the  $I_n$ -dimensional vectors obtained from  $A$  by changing the index  $i_n$  and keeping the other indices fixed, then we have the following definition. This definition is based on matrix techniques which is called matrix unfoldings.

Definition 2. The  $n$ -rank of  $A$ , defined by  $R_n = \text{rank}_n(A)$  is the dimension of the vector space covered by the  $n$ -mode vectors.

The  $n$ -rank of a given tensor  $A \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$  can be formulated by means of matrix techniques as the following expression.

Note: The  $n$ -mode vectors of  $A$  are the column vectors of the matrix unfolding  $A_{(n)}$  and

$$\text{rank}_n(A) = \text{rank}(A_{(n)})$$

The fact that the different  $n$ -ranks of a higher-order tensor are not certainly the same is a major difference with the matrix case as proof easily by giving some examples. The rank of a higher-order tensor is usually defined in porportion with the fact that a rank- $R$  matrix can be decomposed as a sum of  $R$  rank-1 terms; see Figure 4.

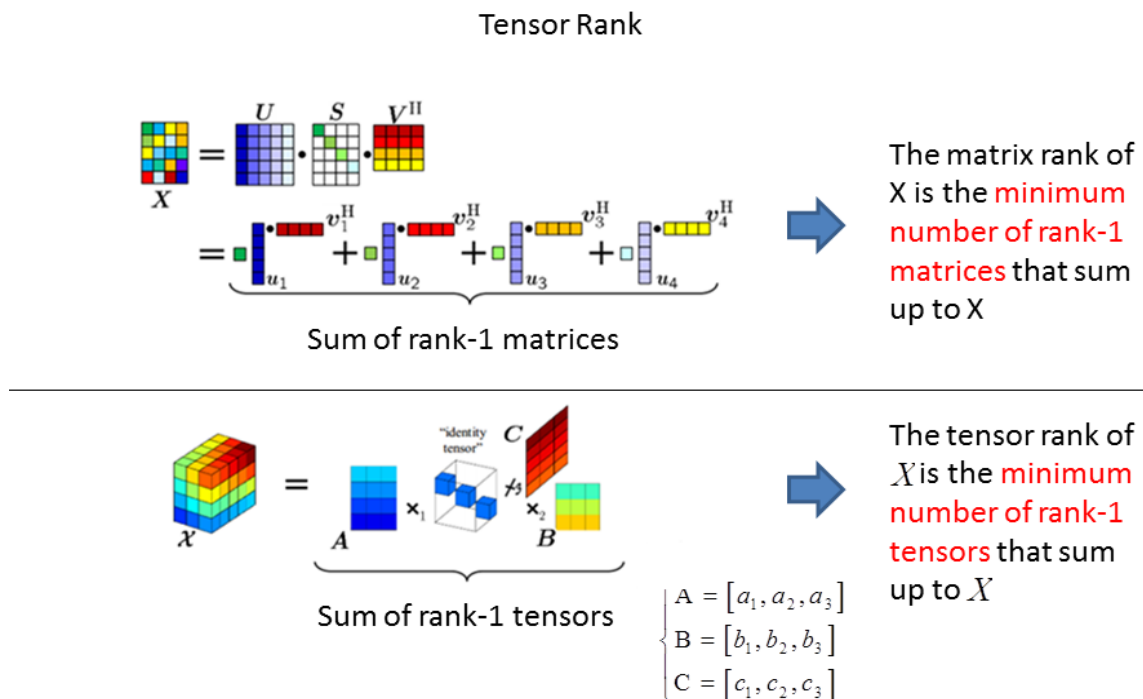


Figure 4. Tensor rank visualization

Definition 3. An  $N$  th-order tensor  $A$  has rank 1 when it equals the outer product of  $N$  vectors  $\mathbf{u}^{[1]}, \mathbf{u}^{[2]}, \dots, \mathbf{u}^{[N]}$ , i.e.,

$$a_{i_1 i_2 \dots i_N} = u_{i_1}^{[1]} u_{i_2}^{[2]} \dots u_{i_N}^{[N]}$$

for all values of the indices.

Definition 4. The rank of an arbitrary  $N$ th-order tensor  $A$ , denoted by  $R = \text{rank}(A)$  is the minimum number of rank-1 tensors that produce  $A$  in a linear combination.

A second and more important difference between matrices and higher-order tensors rank is the fact that the rank is not certainly equal to an  $n$ -rank, even when all the  $n$ -ranks are the same. Based on the previous definitions, it is clear that always  $R_n \leq R$ .

Example 1. Consider the  $(2 \times 2 \times 2)$  - tensor  $A$  defined as a

$$\begin{cases} a_{211} = a_{121} = a_{112} = 1 \\ a_{111} = a_{222} = a_{122} = a_{212} = a_{221} = 0 \end{cases}$$

The 1-rank, 2-rank, and 3-rank are equal to 2. The rank, however, equals 3, since

$$A = X_2 \circ Y_1 \circ Z_1 + X_1 \circ Y_2 \circ Z_1 + X_1 \circ Y_1 \circ Z_2$$

in which,

$$X_1 = Y_1 = Z_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, X_2 = Y_2 = Z_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

is a tensor decomposition in a minimum linear combination of three rank-1 tensors.

#### 2.1.4. Orthogonality concept, Scalar product definition, and Frobenius-norm of tensors

In the high order singular value decomposition (HOSVD) definition of section 2.2, the structure limitation of diagonality of singular values matrix in the second-order

case study will be substituted by a number of geometrical conditions. For this purpose we need a generalization of the well-known definitions of orthogonality concept, scalar product and Frobenius-norm of multi-way arrays. These generalizations and their definitions can be indicated in a straight forward way as follows.

Definition 5. The scalar product  $\langle A, B \rangle$  of two tensors  $A, B \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$  is defined as:

$$\langle A, B \rangle \underline{\underline{def}} \sum_{i_1} \sum_{i_2} \dots \sum_{i_N} b_{i_1 i_2 \dots i_N}^* a_{i_1 i_2 \dots i_N}$$

in which  $*$  denotes the complex conjugation.

Definition 6. Arrays of which the scalar product of two tensors equals 0 are orthogonal.

Definition 7. The Frobenius-norm of a tensor  $A$  can be defined as:

$$\|A\|_F \underline{\underline{def}} \sqrt{\langle A, A \rangle}$$

### 2.1.5. Matrix Kronecker products, Khatri–Rao products, and Hadamard Products

In this section we will discuss several matrix products that are important, so we briefly define them here [2]. The Kronecker product of matrices  $A \in \mathbb{R}^{I \times J}$  and  $B \in \mathbb{R}^{K \times L}$  is defined as  $A \otimes B$ . The result is a matrix of size  $(IK) \times (JL)$  and denoted by:

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \dots & a_{IJ}B \end{bmatrix}$$

$$= [a_{1\otimes} b_1 \quad a_{1\otimes} b_2 \quad a_{1\otimes} b_3 \cdots a_{J\otimes} b_{L-1} \quad a_{J\otimes} b_L]$$

The Khatri–Rao product is the “matching columnwise” Kronecker product.

Given matrices  $A \in \mathbb{R}^{I \times K}$  and  $B \in \mathbb{R}^{J \times K}$ , their Khatri–Rao product is defined by  $A \diamond B$ . The result is a matrix of size  $(I J) \times K$  defined by

$$A \diamond B = [a_{1\otimes} b_1 \quad a_{2\otimes} b_2 \quad \cdots \quad a_{K\otimes} b_K]$$

If  $a$  and  $b$  are vectors, then we can conclude that the Khatri–Rao products and Kronecker products are identical i.e.,

$$a \diamond b = a \otimes b$$

The Hadamard product is the elementwise matrix product. Given matrices  $A$  and  $B$ , both of size  $I \times J$ , their Hadamard product is defined by  $A * B$ . The result can be in size  $I \times J$  and defined by

$$A * B = \begin{bmatrix} a_{11} b_{11} & a_{12} b_{12} & \cdots & a_{1J} b_{1J} \\ a_{21} b_{21} & a_{22} b_{22} & \cdots & a_{2J} b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1} b_{I1} & a_{I2} b_{I2} & & a_{IJ} b_{IJ} \end{bmatrix}$$

### 2.1.6. Multiplication and product of tensor by a matrix

The High Order Singular Value Decomposition (HOSVD) of a multi-way tensor  $A \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$  will be indicated by searching orthogonal coordinate transformations of  $\mathbb{R}^{I_1}, \mathbb{R}^{I_2}, \dots, \mathbb{R}^{I_N}$  which enforce a particular display of the higher-order tensor, as the same for matrices. We will discuss below in more details. Actually in this section, we will introduce a notation for the multiplication of a higher-order tensor by a matrix and their products. This representation will allow us to analyze the effect of basis transformations and their properties. Starting point

is to have a look at the matrix product  $\mathbf{G} = \mathbf{U} \times \mathbf{F} \times \mathbf{V}^T$ , involving matrices  $\mathbf{F} \in \mathbb{R}^{I_1 \times I_2}$ ,  $\mathbf{U} \in \mathbb{R}^{J_1 \times I_1}$ ,  $\mathbf{V} \in \mathbb{R}^{J_2 \times I_2}$ , and  $\mathbf{G} \in \mathbb{R}^{J_1 \times J_2}$ .

To avoid confusion, with “generalized transposes” in the multilinear case, we observed that the relationship between  $\mathbf{U}$  and  $\mathbf{F}$  and the relationship between  $\mathbf{V}$  (not  $\mathbf{V}^T$ ) and  $\mathbf{F}$  are in fact completely similar and show the same behaviors. In other words, in the same way as  $\mathbf{U}$  makes linear combinations of the rows of  $\mathbf{F}$ ,  $\mathbf{V}$  makes linear combinations of the columns of  $\mathbf{F}$ ;

This typical relationship will be indicated by expressing and using the  $X_n$ -symbol:

$$\mathbf{G} = \mathbf{U} \times_1 \mathbf{F} \times_2 \mathbf{V}$$

(For complex matrices the product  $\mathbf{U} \times \mathbf{F} \times \mathbf{V}^H$  is consequently denoted as  $\mathbf{F} \times_1 \mathbf{U} \times_2 \mathbf{V}^*$  in which H denotes matrix hermitian).

In general, we have the following definition and expression.

**Definition 8.** *The  $n$ -mode product of a tensor  $\mathbf{A} \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$  by a matrix  $\mathbf{U} \in \mathbb{C}^{J_n \times I_n}$ , indicated by  $\mathbf{A} \times_n \mathbf{U}$ , is an  $I_1 \times I_2 \times \dots \times I_{n-1} \times J_n \times I_{n+1} \times \dots \times I_N$ -tensor of which the entries are defined by*

$$(\mathbf{A} \times_n \mathbf{U})_{i_1 i_2 \dots i_{n-1} i_{n+1} \dots i_N} \stackrel{\text{def}}{=} \sum_{i_n} \mathbf{a}_{i_1 i_2 \dots i_{n-1} i_n i_{n+1} \dots i_N} \mathbf{u}_{j_n i_n}$$

The  $n$ -mode product of a tensor by a matrix states the following properties [2].

**Property 2.** *Given the tensor  $\mathbf{A} \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$  and the matrices  $\mathbf{F} \in \mathbb{C}^{I_n \times I_n}$ ,  $\mathbf{G} \in \mathbb{C}^{J_m \times I_m}$  ( $n \neq m$ ), one has*

$$(\mathbf{A} \times_n \mathbf{F}) \times_m \mathbf{G} = (\mathbf{A} \times_m \mathbf{G}) \times_n \mathbf{F} = \mathbf{A} \times_n \mathbf{F} \times_m \mathbf{G}$$

**Property 3.** *Given the tensor  $\mathbf{A} \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$  and the matrices  $\mathbf{F} \in \mathbb{C}^{I_n \times I_n}$ ,  $\mathbf{G} \in \mathbb{C}^{K_n \times J_n}$ , one has*

$$(A \times_n F) \times_n G = A \times_n (G \cdot F)$$

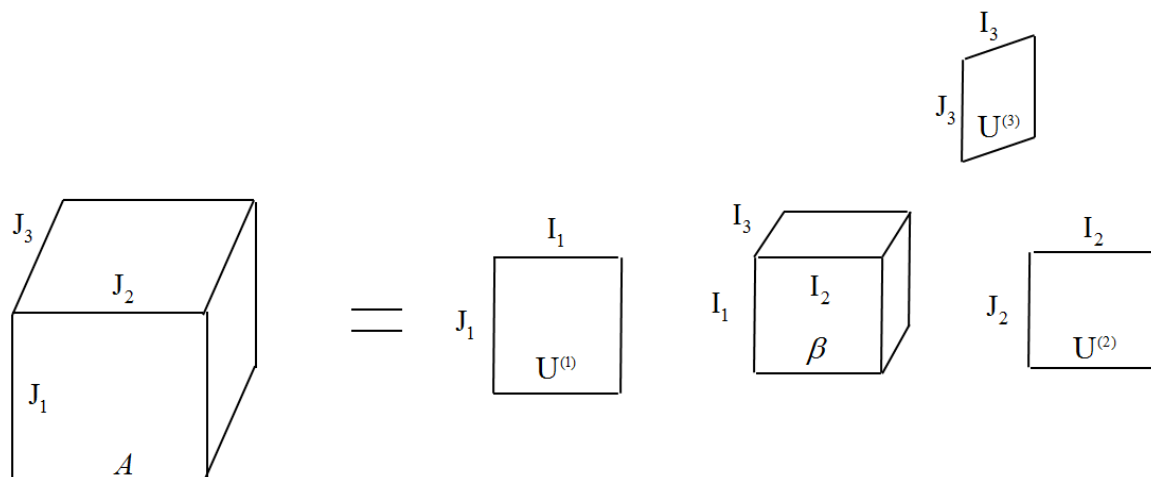


Figure 5. Visualization of the multiplication of a third-order tensor  $B \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$  [2]

Figure 5 visualizes the equation  $A = B \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)}$  for third-order tensors  $A \in \mathbb{C}^{J_1 \times J_2 \times J_3}$  and  $B \in \mathbb{C}^{I_1 \times I_2 \times I_3}$ . Unlike the regular way to illustrate second-order matrix products, for reasons of symmetry,  $U^{(2)}$  has not been transposed. Multiplication with  $U^{(1)}$  involves linear combinations of the “horizontal matrices” (index  $i_1$  fixed) in  $B$ . Stated otherwise, multiplication of  $B$  with  $U^{(1)}$  means that every column of  $B$  (indices  $i_2$  and  $i_3$  fixed) has to be multiplied from the left with  $U^{(1)}$ . Similarly, we can state that multiplication with  $U^{(2)}$ , resp.,  $U^{(3)}$ , involves linear combinations of matrices, obtained by fixing  $i_2$ , resp.,  $i_3$ .

This can be presented as a multiplication or product, from the left of the vectors sequentially obtained by fixing the indices  $i_3$  and  $i_1$ , resp.,  $i_1$  and  $i_2$ .

Visualization frameworks like figure 5 have proven to be very useful to gain insight in tensor techniques and find the relationships of different slices which create tensors.

The  $n$ -mode product of a multi-way arrays (tensor) and a matrix is a special case of the inner product in multilinear algebra and multi-way arrays analysis.

## 2.2. A High Order Singular Value Decomposition (HOSVD)

In this section, a singular value decomposition model is proposed for  $N$  th-order tensors. To simplify the comparison, first we repeat the matrix decomposition in the same notation, as follows.

Theorem 1 (matrixSVD). *Every complex  $(I_1 \times I_2)$ -matrix  $F$  can be written as the product*

$$(1) \quad F = U^{(1)} \cdot S \cdot V^{(2)H} = S \times_1 U^{(1)} \times_2 V^{(2)*} = S \times_1 U^{(1)} \times_2 U^{(2)}$$

in which

1.  $U^{(1)} = (u_1^{[1]} u_2^{[1]} \dots u_{I_1}^{[1]})$  is a unitary  $(I_1 \times I_1)$ -matrix,
2.  $U^{(2)} = (u_1^{[2]} u_2^{[2]} \dots u_{I_2}^{[2]}) (= V^{(2)*})$  is a unitary  $(I_2 \times I_2)$ -matrix,
3.  $S$  is an  $(I_1 \times I_2)$ -matrix with the properties of

(i) pseudodiagonality:

$$(3) \quad S = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_{\min(I_1, I_2)})$$

(ii) ordering:

$$(4) \quad \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(I_1, I_2)} \geq 0$$

The  $\sigma_i$  are singular values of  $F$  and the vectors  $u_i^{[1]}$  and  $u_i^{[2]}$  are, resp., an  $i$ -th left and an  $i$ -th right singular vector. The decomposition is visualized in Figure 6. Now let's state the following theorem for a High Order Singular Value Decomposition (HOSVD). The same procedures like matrix, but the matrix SVD will be changed to core tensor which can be seen as the following steps.



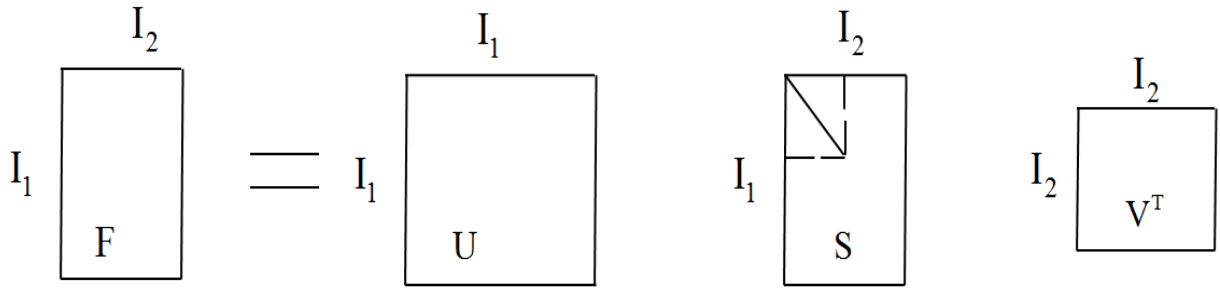


Figure 6. Visualization of the matrix SVD

Theorem 2 (HOSVD). Every complex tensor  $A \in \mathbb{C}^{I_1 \times I_2 \times \dots \times I_N}$  can be written as the product

$$(5) \quad A = S \times_1 U^{(1)} \times_2 U^{(2)} \dots \times_N U^{(N)},$$

in which

1.  $U^{(n)} = (\mathbf{u}_1^{[n]} \mathbf{u}_2^{[n]} \dots \mathbf{u}_{I_n}^{[n]})$  is a unitary  $(I_n \times I_n)$ -matrix,

2.  $S$  is a complex  $(I_1 \times I_2 \times \dots \times I_N)$ -tensor of which the subtensors  $S_{i_n} = a$ , obtained by fixing the  $n$ -th index to  $a$ , have the properties of:

(i) all-orthogonality: two subtensors  $S_{i_n} = a$  and  $S_{i_n} = \beta$  are orthogonal for all possible values of  $n$ ,  $a$  and  $\beta$  subject to  $a \neq \beta$ :

$$(6) \quad \langle S_{i_n} = a, S_{i_n} = \beta \rangle = 0 \quad \text{when } a \neq \beta,$$

(ii) ordering:

$$(7) \quad \|S_{i_n} = 1\| \geq \|S_{i_n} = 2\| \geq \dots \geq \|S_{i_n} = I_n\| \geq 0$$

for all possible values of  $n$  from 1 to  $N$ .

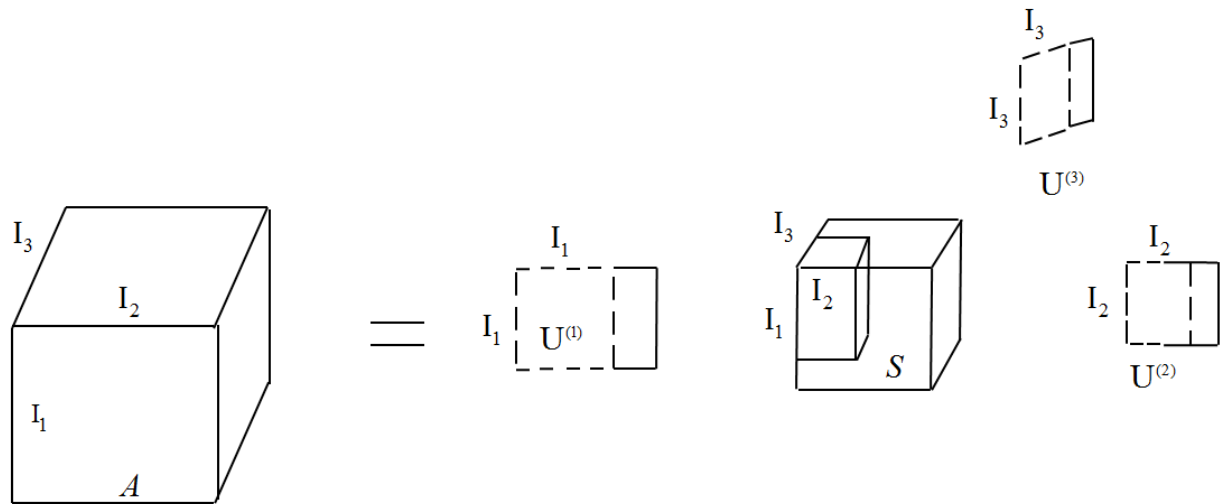


Figure 7. Visualization of the HOSVD for a third-order tensor

The Frobenius-norms  $\|S_{i_n} = i\|$ , symbolized by  $\sigma_i^{(n)}$ , are  $n$ -mode singular values of  $A$  and the vector  $u_i^{(n)}$  is an  $i$ -th  $n$ -mode singular vector.

The decomposition is visualized for third-order tensors in Figure 7. The difference between matrix singular value decomposition (SVD) and tensor high order singular value decomposition (HOSVD) are illustrated in Figure 8.

**Discussion.** Applied to a tensor  $A \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ , Theorem 2 states that it is always possible to find orthogonal transformations of the column, row, and 3-mode space in order that  $S = A \times_1 U^{(1)H} \times_2 U^{(2)H} \times_3 U^{(3)H}$  is all-orthogonal and ordered (the new basis vectors are the columns of  $U^{(1)}$ ,  $U^{(2)}$ , and  $U^{(3)}$ ).

All-orthogonality need a deep understanding and it means that the different “horizontal matrices” of  $S$  (the first index  $i_1$  is kept fixed, while the two other indices,  $i_2$  and  $i_3$ , are free) are mutually orthogonal with respect to the scalar product of matrices. In other words, the sum of the products of the corresponding entries vanishes. Meanwhile, the different “frontal” matrices ( $i_2$  fixed) and the different “vertical” matrices ( $i_3$  fixed) should be mutually orthogonal as well as before.

## Properties of the HOSVD

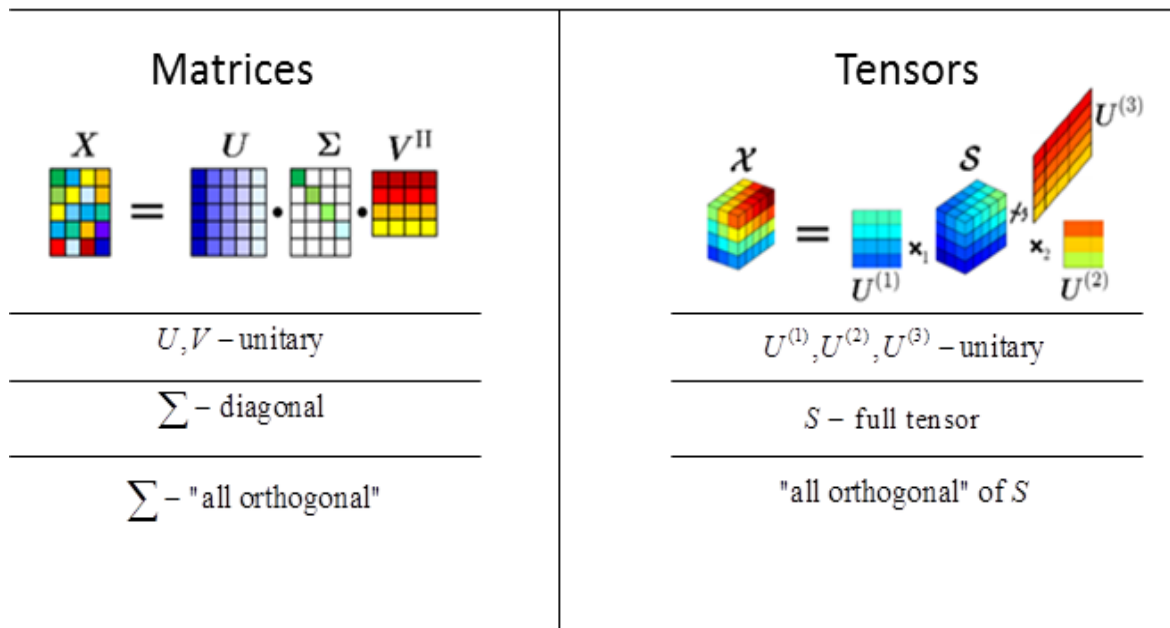


Figure 8. Difference between matrices SVD and tensor HOSVD

### 2.3. The Canonical Polyadic Decomposition (CANDECOMP/PARAFAC)

In 1927, Hitchcock was the first one who proposed the idea of the polyadic form of a tensor [3]. In other words, it is a kind of generalization of the matrix singular value decomposition to multi-way arrays by expressing a tensor as the sum of a finite number of rank-one tensors; Such decompositions represent a tensor as the sum of the  $n$ -fold outer products of rank-1 tensors, where  $n$  is the dimension of the tensor indices. We refer to the Canonical Polyadic Decomposition (CANDECOMP) as CP Decomposition.

The CP decomposition is a simple way to reformulate a tensor into a sum of component rank-one tensors. For example, if we consider a noiseless third-order tensor  $X \in \mathbb{R}^{I \times J \times K}$ , we can write it as sum of  $R$  rank one tensors. Later, we can find how to define rank of a tensor by applying this decomposition.

$$(8) \quad x = \sum_{r=1}^R a_r \circ b_r \circ c_r$$

where  $R$  is a positive integer and  $a_r \in \mathbb{R}^I$ ,  $b_r \in \mathbb{R}^J$ , and  $c_r \in \mathbb{R}^K$  for  $r = 1, \dots, R$ . Elementwise, (8) is written as:

$$(9) \quad x_{ijk} = \sum_{r=1}^R a_{ir} b_{jr} c_{kr} \quad i = 1, \dots, I, j = 1, \dots, J, k = 1, \dots, K$$

This is illustrated for noisy tensor in Figure 9.

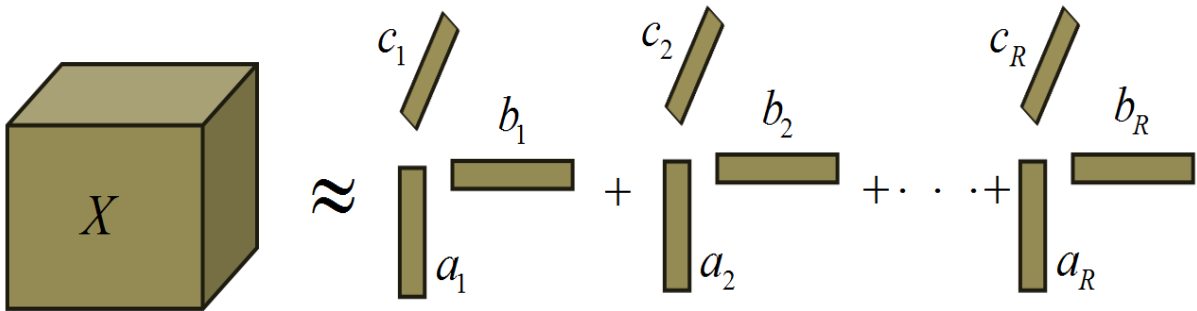


Figure 9. CP decomposition of a three-way array

The factor matrices defined as a combination of the vectors from the rank-one components, i.e.,  $\mathbf{A} = [\mathbf{a}_1 \ \mathbf{a}_2 \ \dots \ \mathbf{a}_R]$  and likewise for  $\mathbf{B}$  and  $\mathbf{C}$ . Using these definitions, (8) can be written in matricized form (one per mode):

$$(10) \quad \begin{cases} \mathbf{X}_{(1)} \approx \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T \\ \mathbf{X}_{(2)} \approx \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T \\ \mathbf{X}_{(3)} \approx \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T \end{cases}$$

Remember that  $\odot$  indicates the Khatri–Rao product from section 2.1.5. The three-way model is sometimes written in terms of the frontal slices of  $X$  :

$$X_k \approx \mathbf{A} \mathbf{D}^{(k)} \mathbf{B}^T, \quad \text{where } \mathbf{D}^{(k)} \equiv \text{diag}(\mathbf{C}_{k:}) \quad \text{for } k = 1, \dots, K$$

Similar equations can be written for the horizontal and lateral slices. In general, though, slice form expressions can not easily extract beyond three dimensions. Following the CP model can be briefly presented as

$$X \approx \llbracket A, B, C \rrbracket \equiv \sum_{r=1}^R a_r \circ b_r \circ c_r$$

### 2.3.1. Tensor Rank

The rank of a tensor  $X$ , denoted  $\text{rank}(X)$ , is indicated as the smallest number of rank-one tensors (see section 2.1.3) that generate or make  $X$  as their sum. In other words, this is the minimum number of factorized components in an exact CP decomposition, where “exact” means that there is equality in (8). An exact CP decomposition with  $R = \text{rank}(X)$  components is called the rank decomposition. The definition of tensor rank is similar to the definition of matrix rank, but the properties and characteristics of matrix and tensor ranks are quite different. One difference which we can mention is that the rank of a real-valued multi-way arrays may quite be different over  $\mathbb{R}$  and  $\mathbb{C}$ .

### 2.3.2. Uniqueness

Unlike the matrix decomposition, rank decompositions of higher order tensors are often unique which is an interesting property of higher-order tensors. By considering the fact that matrix decompositions are not unique, let  $X \in \mathbb{R}^{I \times J}$  be a matrix of rank  $R$ . Then a rank decomposition of this matrix is

$$X = AB^T \equiv \sum_{r=1}^R a_r \circ b_r$$

If the SVD of  $X$  is  $U\Sigma V^T$ , then we can define  $A=U\Sigma$  and  $B=V$ . However, it is equally valid to indicate  $A=U\Sigma W$  and  $B=VW$ , where  $W$  is some  $R \times R$  orthogonal matrix. Put differently, we can easily generate two completely different sets of  $R$  rank-one matrices that sum to the original matrix. The SVD of a matrix is unique (assuming all the singular values are distinct) only because of the addition

of orthogonality limitations (The diagonal matrix of singular values is located in the middle). For the CP decomposition, these properties can be apply but under weaker condistions. Let  $X \in \mathbb{R}^{I \times J \times K}$  be a three-way tensor of rank  $R$ , i.e.,

$$X = \llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket \equiv \sum_{r=1}^R a_r \circ b_r \circ c_r$$

To generate tensor  $X$ , there is just only one possible summation of rank-one tensors, which means Uniqueness. We must consider the exception of the elementary uncertainty of scaling and permutation.

Given a sequence of matrices  $\mathbf{A}^{(n)}$  of size  $I_n \times R$  for  $n = 1, \dots, N$ ,  $\llbracket \mathbf{A}^{(1)}, \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket$  indicates an  $I_1 \times I_2 \times \dots \times I_N$  tensor whose elements are given by:

$$\left( \llbracket \mathbf{A}^{(1)}, \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \rrbracket \right)_{i_1 i_2 \dots i_N} = \sum_{r=1}^R \prod_{n=1}^N a_{i_n}^{(n)} \quad \text{all } i_n \in \{1, \dots, I_n\}$$

#### 2.4. Computing the CP Decomposition (ALS Method)

As we discussed before, there is no confined algorithm for determining the rank of a multi-way arrays. Accordingly, the way for choosing the number of rank-one components is the first problem that appear in computing a CP decomposition. Most algorithms perform some multiple CP decompositions with different numbers of components until one combination is “fit”. Ideally, if the noise can be neglect from the data and have a reliable procedure for computing CP with a given number of components, then we can do that calculation for  $R = 1, 2, 3, \dots$  components and the convergence criterion is the first value of  $R$  that obtains a fit of full accuracy. However, there are many issues with this method.

Considering the number of components is fixed, there are many algorithms to calculate a CP decomposition. In this work we focus on the “workhorse” method what is today the most simple way to compute CP decomposition: the alternating

least squares (ALS) method proposed in the original papers by Carroll and Chang [9]. To avoid confusion, we only apply the method for the third-order case, but the full algorithm is presented for a multi-way array in Figure 10.

Let's assume  $X \in \mathbb{R}^{I \times J \times K}$  be a third-order tensor. The goal is to compute a CP decomposition with  $R$  components that best approximates of tensor  $X$ , i.e., to solve

$$\widehat{X} = \llbracket \lambda; \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket = \sum_{r=1}^R \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \min_{\widehat{X}} \left\| X - \widehat{X} \right\|$$

The Alternating Least Squares (ALS) approach fixes  $\mathbf{B}$  and  $\mathbf{C}$  to solve for  $\mathbf{A}$ , then fixes  $\mathbf{A}$  and  $\mathbf{C}$  to solve for  $\mathbf{B}$ , then fixes  $\mathbf{A}$  and  $\mathbf{B}$  to solve for  $\mathbf{C}$ , and follow the same procedure until some stopping criterion is satisfied. Assuming that all the matrices fixed, just one can move, the problem re-formulates to a linear least-squares problem. For example, consider that  $\mathbf{B}$  and  $\mathbf{C}$  are fixed. Then, from (10), we can formulate the above minimization problem in matrix form as

$$\min_{\widehat{\mathbf{A}}} \left\| \mathbf{X}_{(1)} - \widehat{\mathbf{A}} (\mathbf{C} \diamond \mathbf{B})^T \right\|_{\text{F}}$$

where  $\widehat{\mathbf{A}} = \mathbf{A} \cdot \text{diag}(\lambda)$ . The optimal solution is then calculated by

$$\widehat{\mathbf{A}} = \mathbf{X}_{(1)} \left[ (\mathbf{C} \diamond \mathbf{B})^T \right]^{\dagger}$$

Because the Khatri–Rao product pseudoinverse has the special properties, it is common to formulate the solution as

$$\widehat{\mathbf{A}} = \mathbf{X}_{(1)} (\mathbf{C} \diamond \mathbf{B}) (\mathbf{C}^T \mathbf{C} \diamond \mathbf{B}^T \mathbf{B})^{\dagger}$$

The full Alternating Least Squares (ALS) procedure for a multi-way arrays is illustrated in Figure 10. Let's assume that the number of  $R$  components of the CP

decomposition is specified. We can set the initial values for factor matrices in any way, such as randomly or by defining

$$A^{(n)} = R \text{ leading left singular vectors of } X_{(n)} \text{ for } n = 1, \dots, N.$$

At each inner iteration, the pseudoinverse of a matrix  $V$  (see Figure 10) must be computed, but it is only of size  $R \times R$ . The iterations repeat until some combination of stopping criterion is satisfied. Different stopping conditions can be consider as follows: small or no betterment in the objective function, little or no obvious change in the factor matrices, the objective value reaches zero or near it, and exceeding a predefined maximum number of iterations.

```

Procedure CP-ALS ( $X, R$ )
  initialize  $A^{(n)} \in \mathbb{R}^{I_n \times R}$  for  $n = 1, \dots, N$ 
  repeat
    for  $n = 1, \dots, N$  do
       $V \leftarrow A^{(1)T} A^{(1)} * \dots * A^{(n-1)T} A^{(n-1)} * A^{(n+1)T} A^{(n+1)} * \dots * A^{(N)T} A^{(N)}$ 
       $A^{(n)} \leftarrow X^{(n)} (A^{(N)} \odot \dots \odot A^{(n+1)} \odot A^{(n-1)} \odot \dots \odot A^{(1)}) V^\dagger$ 
      normalize columns of  $A^{(n)}$  (storing norms as  $\lambda$ )
    end for
  until fit ceases to improve or maximum iterations exhausted
  return  $\lambda, A^{(1)}, A^{(2)}, \dots, A^{(N)}$ 
end procedure

```

Figure 10. ALS algorithm to compute CP decomposition with  $R$  components for an  $N$ th-order tensor  $X$  of size  $I_1 \times I_2 \times \dots \times I_N$



## 3 ELECTRICAL GRID DATA ANALYSIS

### 3.1. Electrical grid data

In this chapter, first we apply interpolation method to recover missing data and then we present a proper analysis of multi-way power grid data to evaluate the benefits of using tensor-based processing by computing an approximate CP decomposition and High Order Singular Value decomposition (HOSVD).

The tensor analysis is inspired by (subspace-based) low-rank modeling. Put differently, we typically gain from using tensors if the tensor has a low-rank structure that can be exploited. We can present this work in two steps, first we try to capture the latent structure of the data via multiway arrays analysis in presence of missing data and the second step is to find a low rank structure in our power grid data.

At each power bus we can read the power components such as active power, reactive power and voltage which measure the usable information to track, design, analyze, monitor and compensate electrical systems. Active and reactive powers are thus used to indicate system characteristics as power factor, power compensation, installed power capacity, or load behaviour. In this work, we consider power grid data of buses in multi-way arrays framework as the data often have more than two modes of variation. In other words, in one bus, data of each record can be represented as a time-bus matrix; thus, data from multiple buses or channels is three-dimensional (time, bus, and voltage or power) and forms a three-way arrays.

The data have read from a set power meters from EU energy network (Figure 11). Each record per hour from the buses can be represented as a time-voltage matrix; thus, data from multiple buses is three-dimensional (time, bus, and voltage) or

measured data of loads and generators of a huge power system. Hourly sample rate for the bus and generator number 1 is given in Table 1.

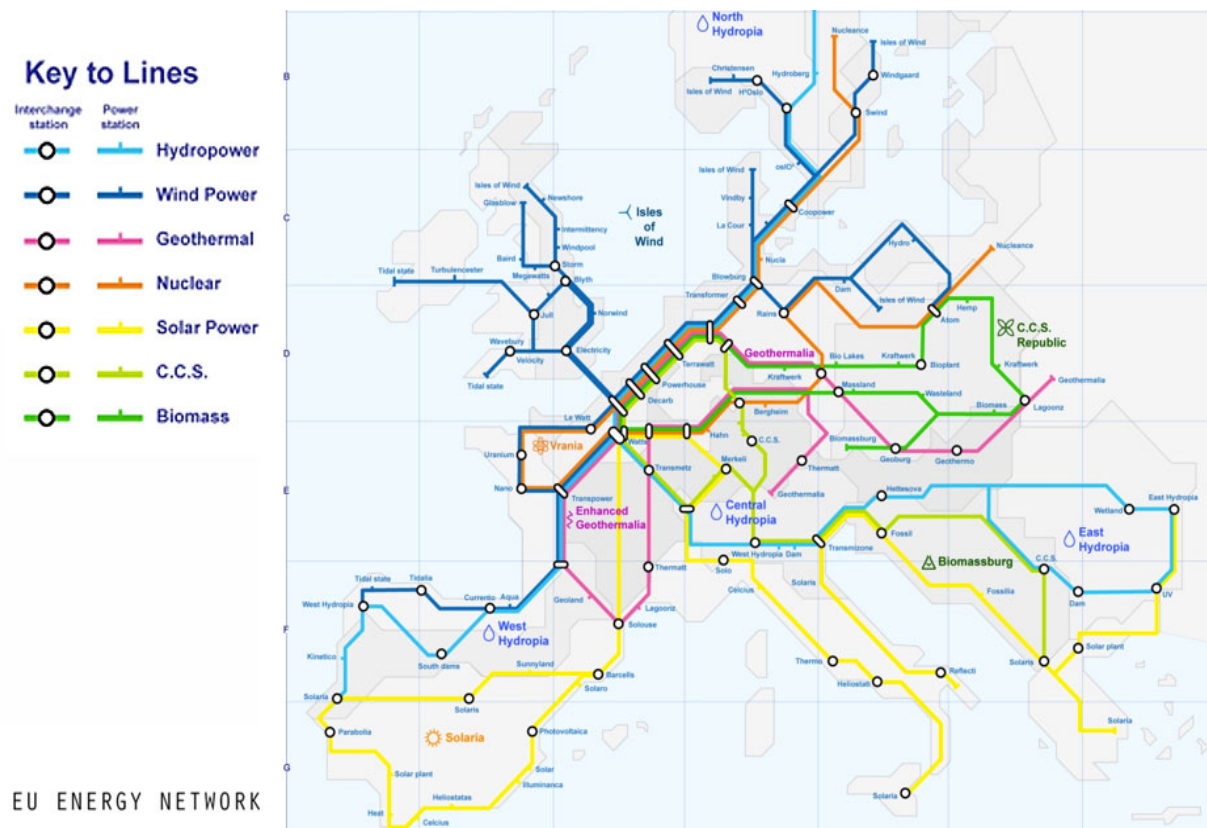


Figure 11. EU energy network

The data set contains multi-channel buses (12418 channels) recorded from 36 country nodes. For each measurement, the data from each bus can be presented by time domains (forming a vector of length 692); In other words, each measurement can be presented by a bus over time with 5 bus properties (Active power, reactive power, voltage, country node, number of buses). The data of all measurements can then be arranged as a buses over time include the bus properties, which generate a tensor of size  $692 \times 12418 \times 5$ . To simplify the concept, first we consider the data set 1 with the following properties:

Data set 1:

⇒ 12418 channels, 29 days x 24 hours

Table 1. Hourly sample rate for the bus and generator number 1

<b>Data of bus number 1</b>							
	Bus 1		Generator 1		Voltage KV	Country node	Number of bus
	Active Power	Reactive Power	Active Power	Reactive Power			
	P MW	Q MVAR	P MW	Q MVAR			
31-Dec- 2012 23:30:00	129.2	6.1	129.2	6.1	240.5	7	1
31-Dec- 2012 22:30:00	126.1	12.3	126.1	12.3	240.1	7	1
31-Dec- 2012 21:30:00	132.1	24.4	132.1	24.4	239.6	7	1
31-Dec- 2012 20:30:00	130.3	25.7	130.3	25.7	239.4	7	1

Hourly sample rates of active power for the 1000 buses plotted, as illustrated in Figure 12. In power system, if the set power meters that are used to measure the data along the buses loose or disconnected, the data is either lost or discarded. To recover the missing data or suppress the data mixed with noise, we need to apply some analytical techniques.

To reflect such cases of missing data, let's define a nonnegative weight tensor  $W$  of the same size as  $X$  such that:

$$w_{i_1 i_2 \dots i_N} = \begin{cases} 1 & \text{if } x_{i_1 i_2 \dots i_N} \text{ is known, Red} \\ 0 & \text{if } x_{i_1 i_2 \dots i_N} \text{ is missing, Blue} \end{cases}$$

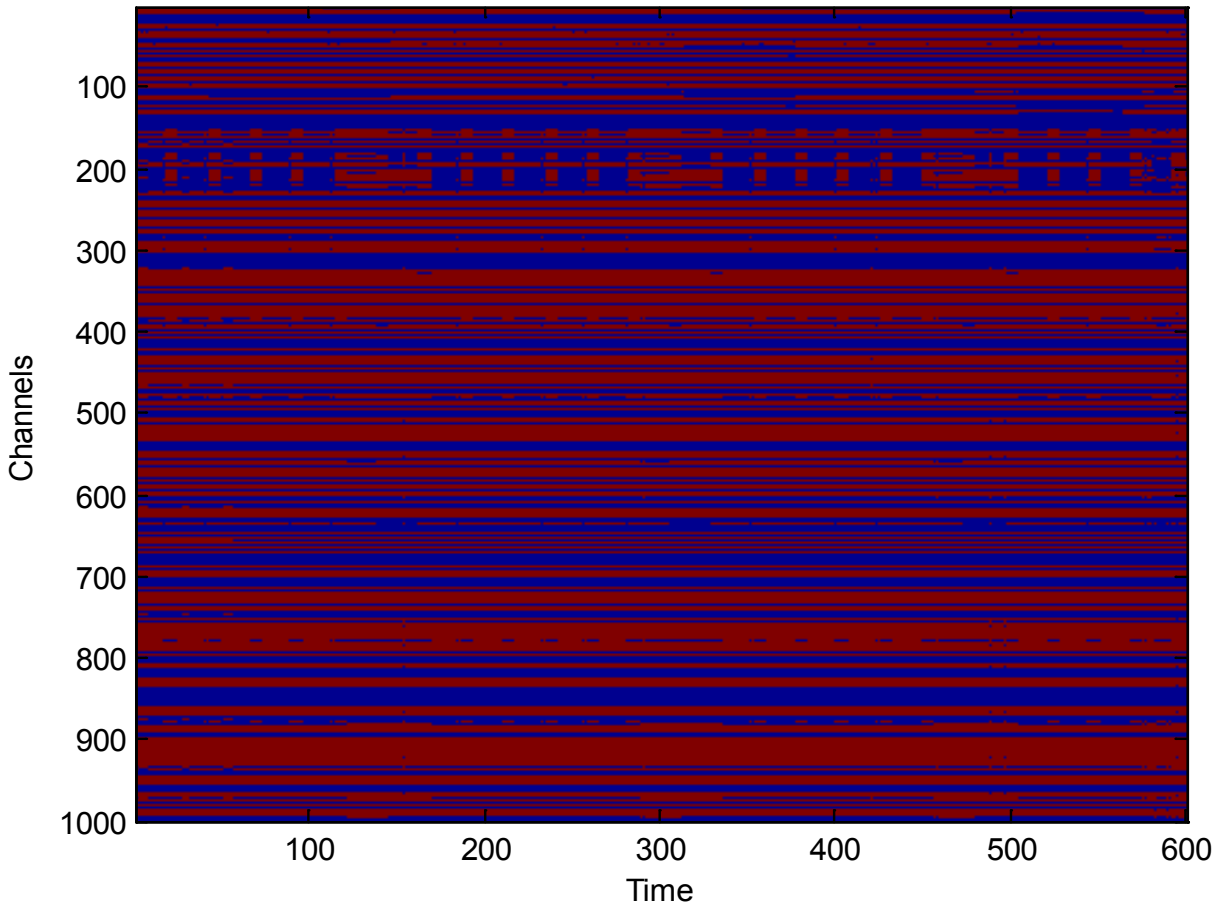


Figure 12. Hourly sample rates for the 1000 channels, missing values shown by blue

### 3.2. Preprocessing

The starting point is to find a reliable method for selecting the interesting channels and data which provide the necessary information for further processing. Between all the methods, we propose a more simple method based on interpolation of missing values that can be divided in two steps as below:

- ⇒ Select interesting chunk (channels, days): MANUAL!
- ⇒ Interpolate missing values: linear (so far)

### 3.2.1. Select interesting chunk (channels, days):

In this section, we try to select more proper channels of the first data set for further analyzing with the minimum missing data. As it is illustrated in Figure 13, we select the 600 hours sample rates of active power for 100 buses with the minimum missing data out of 12418 channels.

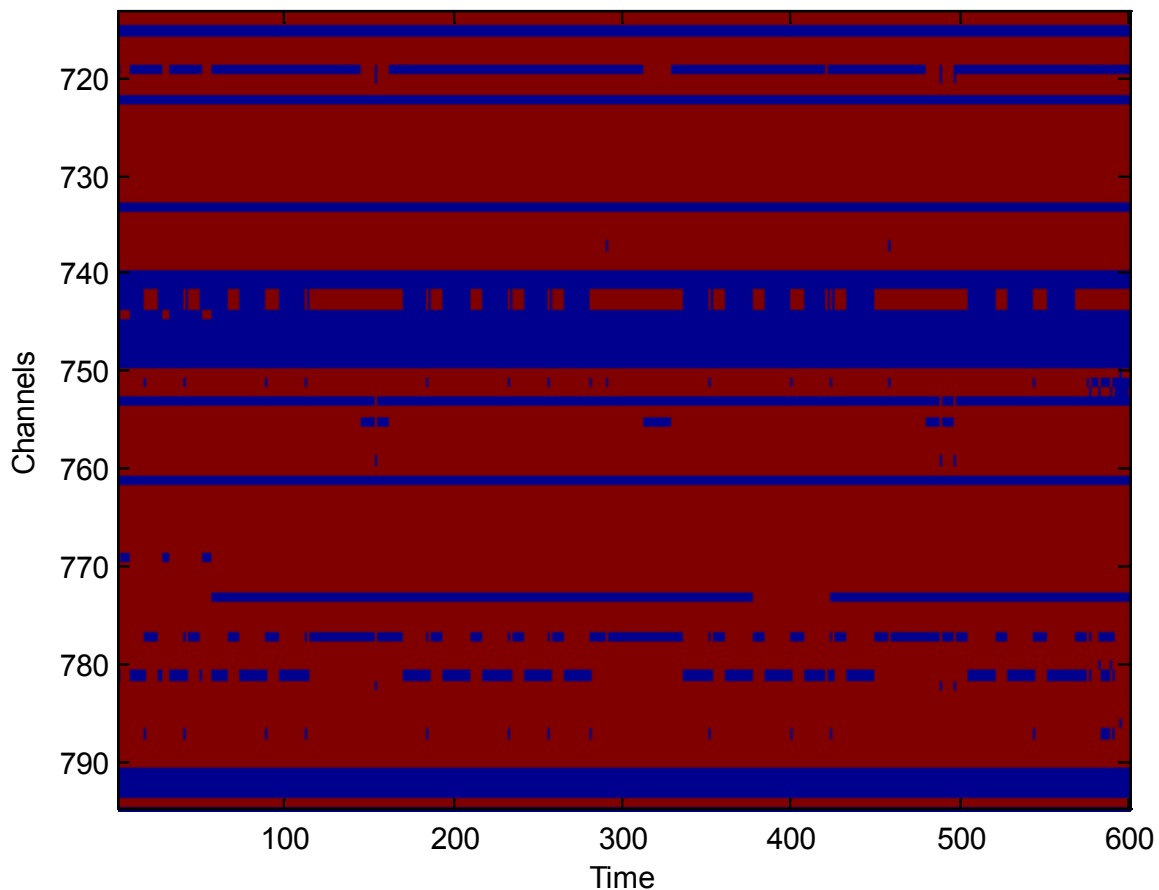


Figure 13. Interesting channels

Based on the weight tensor that we select, the missing values are shown with blue color in Figure 13. Further manual selection lead to choose 175 “interesting” channels and remove the channels with data which all are missed. At the end, the interesting chunk is consist of 175 channels (175 channels, 600 hours) of active power data that is shown in Figure 14.

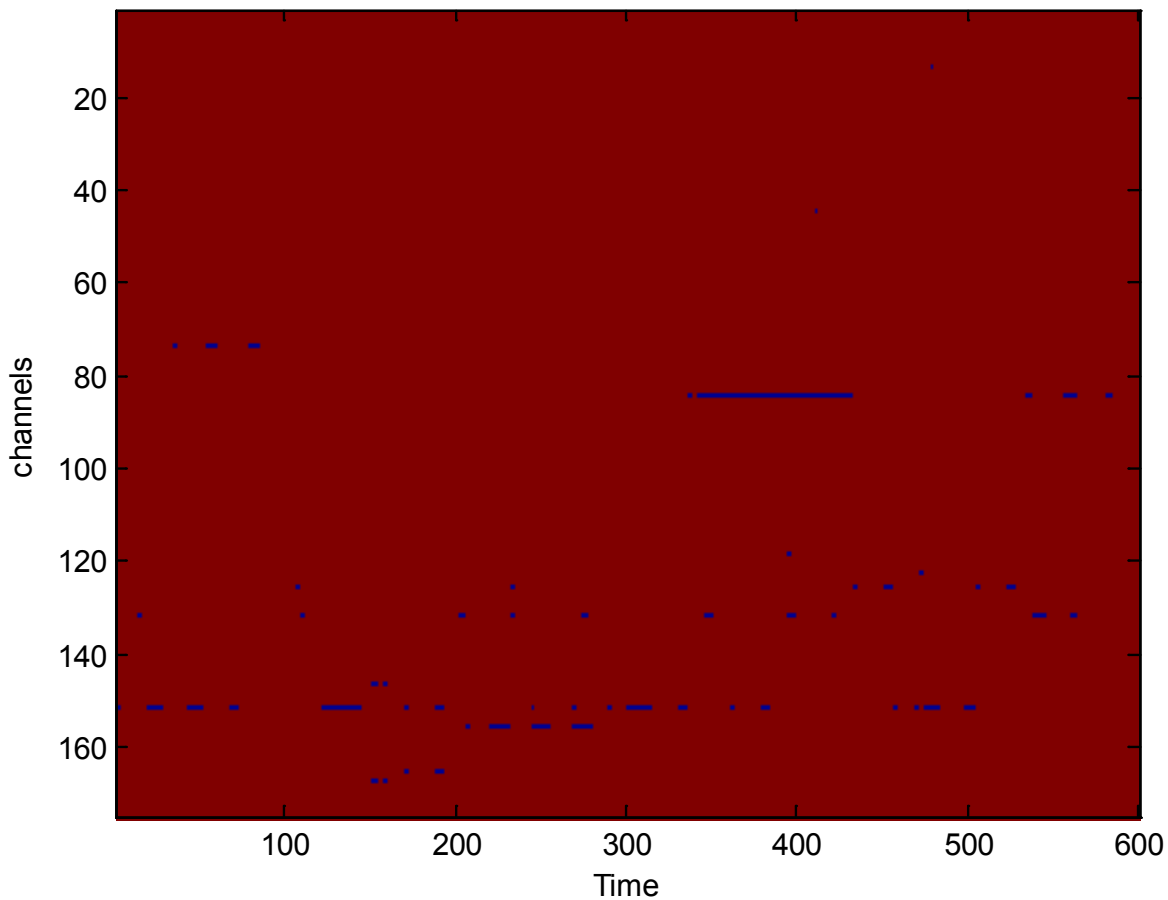


Figure 14. Interesting chunk

### 3.2.2. Interpolate missing values:

Interpolation is a method of constructing missing data points within the range of a discrete set of known data points. One of the simplest methods is linear interpolation (sometimes known as lerp). Although it seems a simple way to recover the missing values, we applied this method in this chapter because just we need to find a tensor low rank structure of power grid data.

After choosing the interesting buses, next step will be interpolating of the remained missing data. As an example, for the bus number 151 the linear interpolation of the first 600 hours can be seen in Figure 15.

To overcome the problem of first/last values missing data in the interpolation, we assume zero for them.

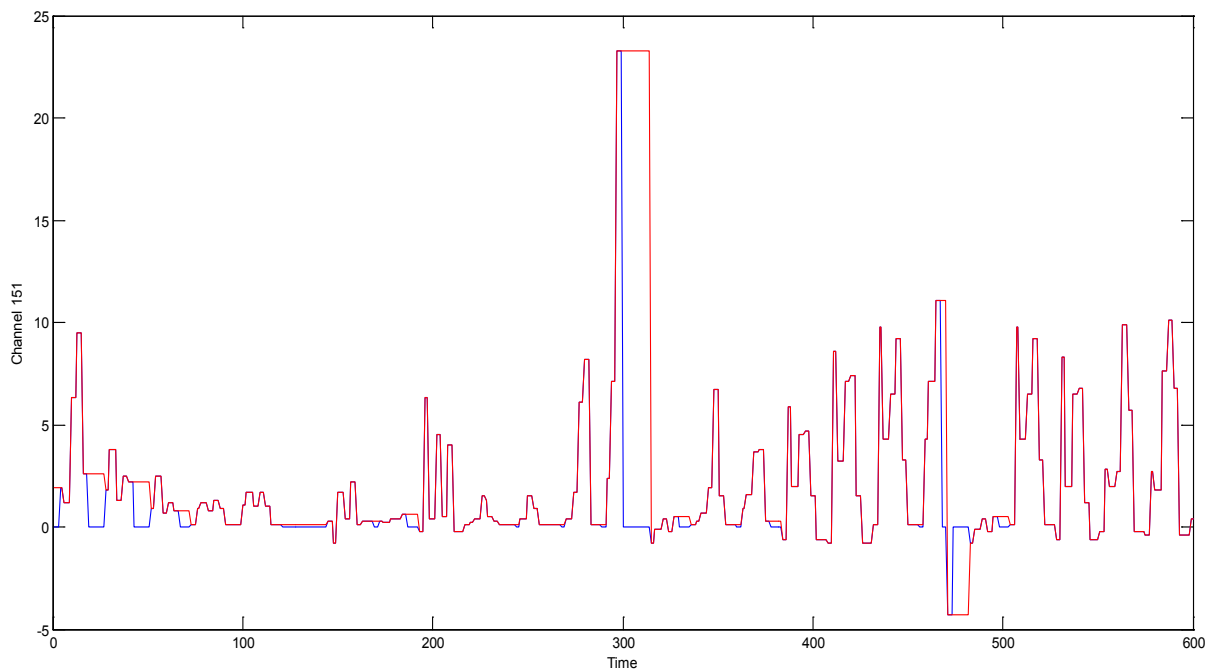


Figure 15. Channel 151 interpolating, red color is the interpolated one

### 3.3. Evaluation of the benefits of using tensor-based processing

#### Is using tensors has any benefit?

To answer this question, first we must find an analytical method to analyze the electrical grid data such that a low-rank modeling in their structure can be fined.

In this work, the tensor analysis that we apply is inspired by (subspace-based) low-rank modeling. we typically gained from using tensors if the tensor has a low-rank structure that we can exploited.e.g., HOSVD-based subspace estimate only better than SVD if the tensor has low n-ranks in more than one mode. This is natural, otherwise we could just reshape an arbitrary matrix into a tensor and gain from it. So there must be an underlying structure. In order to find this low-rank structure in our power grid data, we employ Mean Square Error (MSE) algorithm to reconstruct a tensor with different rank. Let's discuss it with more details in the next section.

---

### How do we check whether there is a low-rank (tensor) structure in the data?

The idea is to use the “compression graphs” that shows reconstruction error of data by applying CP decomposition and High Order Singular Value Decomposition (HOSVD) versus compression. coarsely resembles rate/distortion graphs, well known in compression theory.

The goal is to perform low rank (tensor) approximations in order to find low rank structure in our data and plotting distortion (reconstruction error) versus compression (number of parameters in the low rank model / number of data samples). The idea will be more obvious if this curve comparing to one obtained from unstructured data, shows a better fractions. Then we concluded that there is some low-rank structure. Reconstruction error in this method can be calculated via this formula:

$$\text{relative MSE} = \mathbb{E} \left\{ \frac{\|X_{rec} - X_0\|_H^2}{\|X_0\|_H^2} \right\}$$

There are different low rank approximations for a given tensor  $X \in \mathbb{C}^{M_1 \times \dots \times M_n}$  with  $M$  elements for both CP decomposition and HOSVD. The number of parameters in the low rank model for each decomposition mentioned below:

$$M = \prod_{r=1}^R M_r \text{ elements}$$

$$\Rightarrow \text{CP rank-d: } \prod_{r=1}^R M_r \times d$$

$$\Rightarrow \text{Truncated HOSVD, rank } (d, d, \dots, d): d^R + \prod_{r=1}^R M_r \times d$$

$$\Rightarrow \text{Truncated HOSVD, rank } (d_1, d_2, \dots, d_R): \prod_{r=1}^R d_r + \prod_{r=1}^R M_r \times d$$



---

Note: truncating HOSVD with  $d$  elements is a way to short the core tensor to  $d$  elements, in addition the singular vector matrices has been truncated to  $d$  columns.

### **Simulated Unstructured data:**

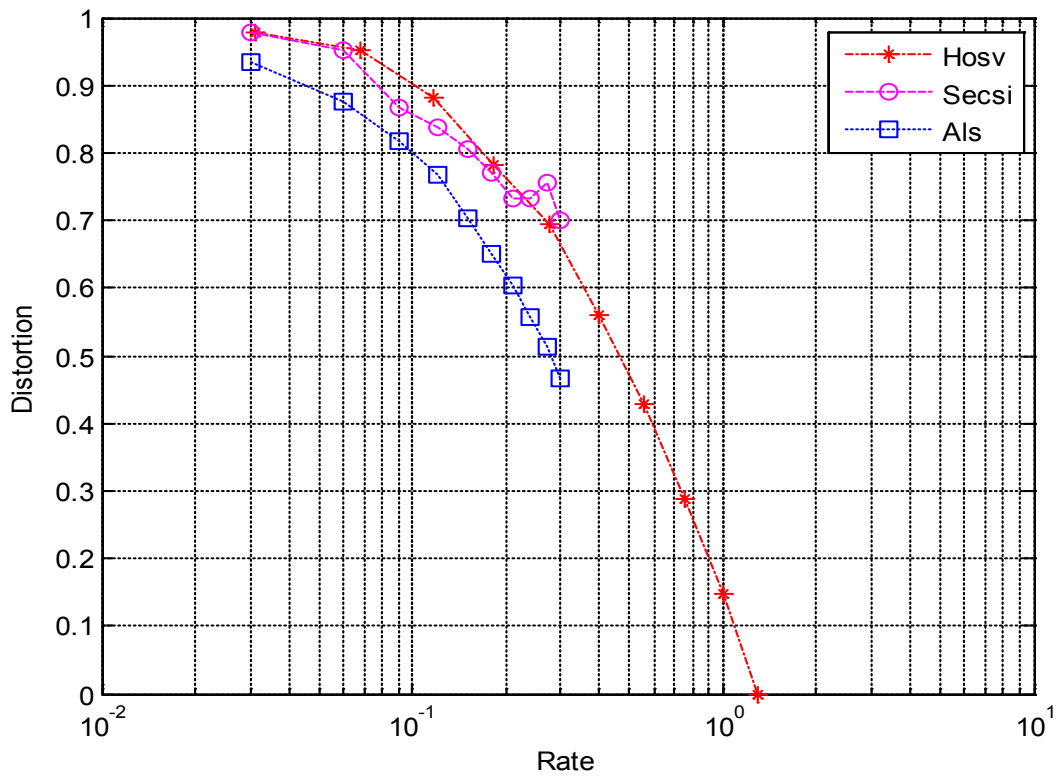
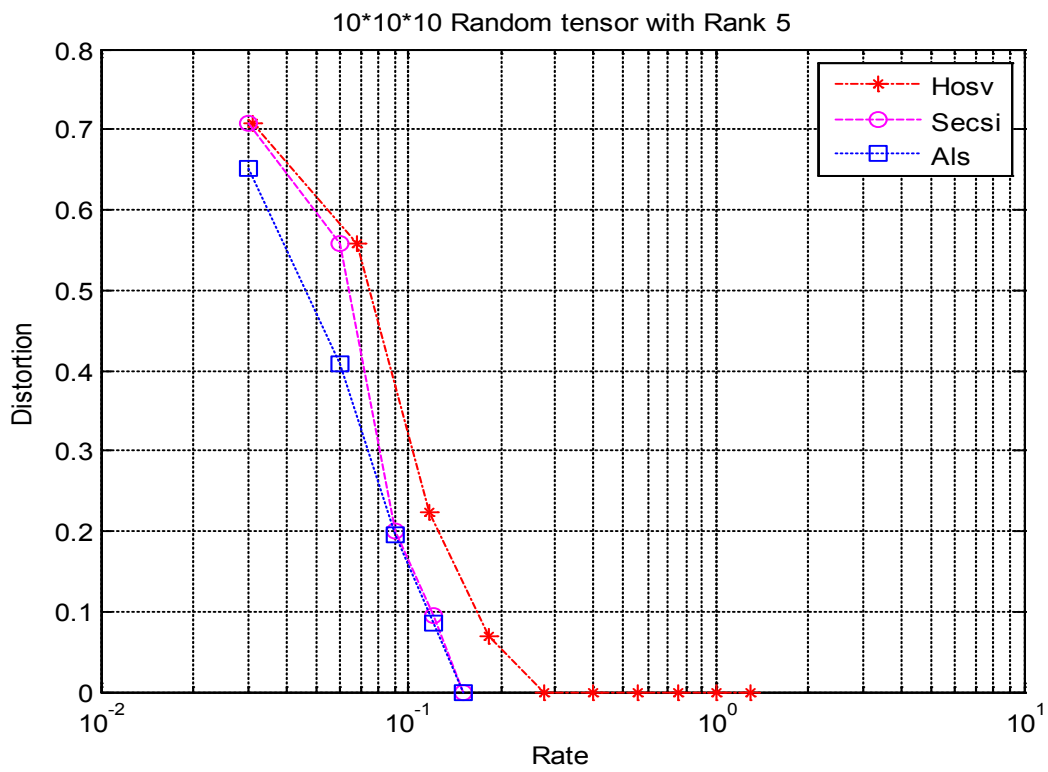
To have a simulated unstructured data and perform low rank approximations, first we generate a pseudo random factor matrices  $U$  and their associated full tensor  $T$  with rank specification 5. Furthermore a cell array of pseudo random factor matrices  $U = \{U^{(1)}, U^{(2)}, \dots\}$  corresponding to a CPD of a tensor in rank-one terms can be generated.

Assume a cell array of factor matrices  $U = \{U^{(1)}, U^{(2)}, \dots\}$ , it is associated full tensor  $T$  that can be computed.

We randomly generate three way tensors with size  $(10 \times 10 \times 10)$ , using tensor-based processing by computing an approximate CP decomposition, SECSI algorithm (calculated by Semi-algebraic framework for approximate CP decompositions via Simultaneous matrix diagonalization) [10] and HOSVD decomposition. The procedure for an  $N$ -way tensor is shown in Figure 16. As it can seen, there is a uniform distortion for different number of parameters in low-rank modeling.

The points on the curve show different ranks ( $n$ -ranks). As it is shown in Figure 16, there is no structure (no fraction) on simulated data, so we can concluded that there is no low rank modeling in this case.

Now let's generate a random tensor with specific rank 5 (Figure 17) to search low-rank modeling in our simulated data. Accordingly, by plotting distortion for different ranks, the fraction in the simulated tensor structure can be found. The horizontal axis considered as a rate for the number of parameters at different ranks over the sampled data. As remembered, the vertical axis nominated for the reconstruction error in CP decomposition and High Order Singular Value Decomposition. low rank (tensor) approximations performed ideally with compression graphs.

Figure 16.  $10*10*10$  random tensorFigure 17.  $10*10*10$  random tensor rank 5

In Figure 17, easily it can be illustrated that there is a fraction on rank 5 of CP decomposition and also in 5-rank of HOSVD, so the low-rank tensor structure clearly visible in this case study. As far as, was considered an ideal case (noise free), now let's assume a model with noise in our simulated data.

The next study case is a random tensor with rank 5 which is added with 10 dB noise (Figure 18).

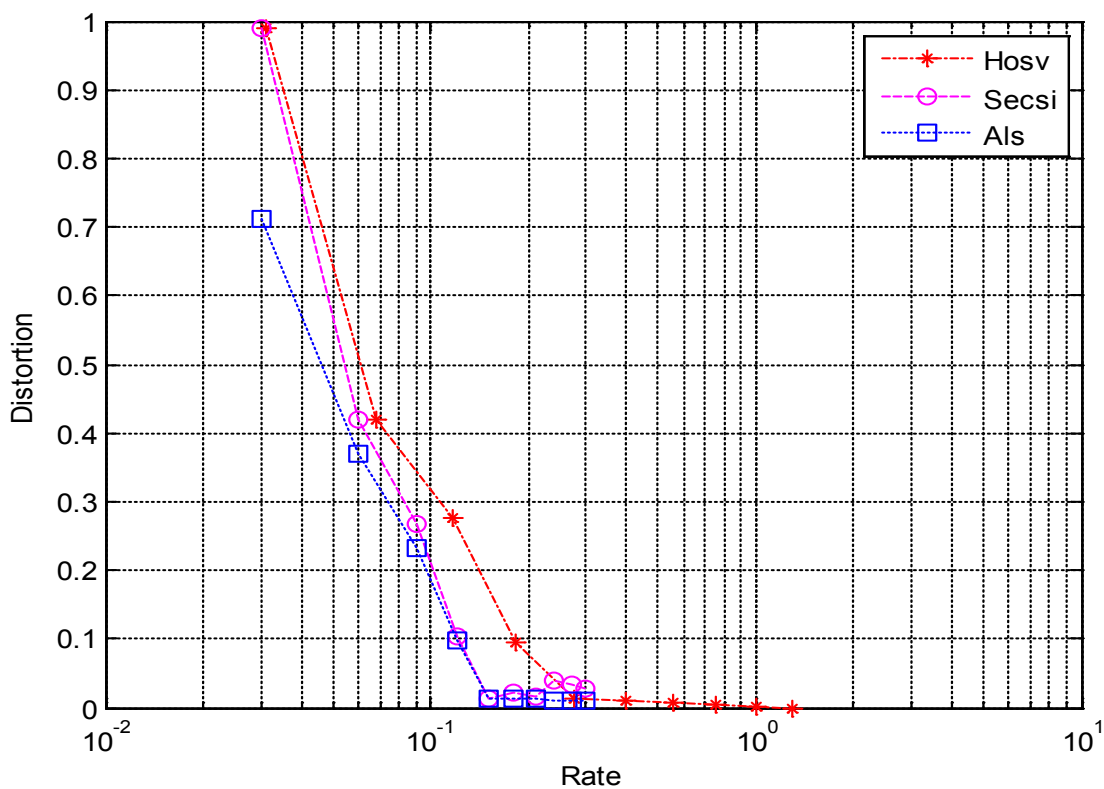


Figure 18.  $10 \times 10 \times 10$  random tensor rank 5+ noise (10 dB SNR)

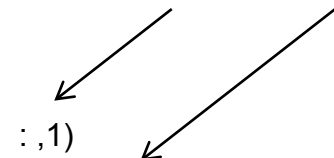
As it can be shown in Figure 18, low-rank tensor structure is clearly visible. In other words, the fraction of data in CP-rank 5 and 5-rank HOSVD is visible.

Through compression graphs, we searched for a rough idea whether there is some low-rank (tensor) structure in some given data set or no. The last case with noise was close to our data structure. Now, the same procedure can be done over power grid data set!

It would be more appropriate if we broke data set into 3D and 4D for further processing.

### 3.4. Reshape time axis of the power grid data to 3D and 4D structure

The data set of power grid can be arranged as buses over time in a matrix of size  $600 \times 175$  for active powers. Therefore, the time axis can be broken into two dimensional space (24 hours x 21 days) or three dimensional space (24 hours x 7 days x 3 weeks) to generate respectively a 3-D tensor or 4-D tensor. For instance, to avoid confusion R-D representation of data illustrated in Figure 19.

- active power      selected buses
- 
- Data set (time , : , 1)
  - 3-D = Reshape (24,21,175);
  - 4-D = Reshape (24,7,3,175);

Example:  $R = 2, M_1 = 4, M_2 = 3, N = 5$

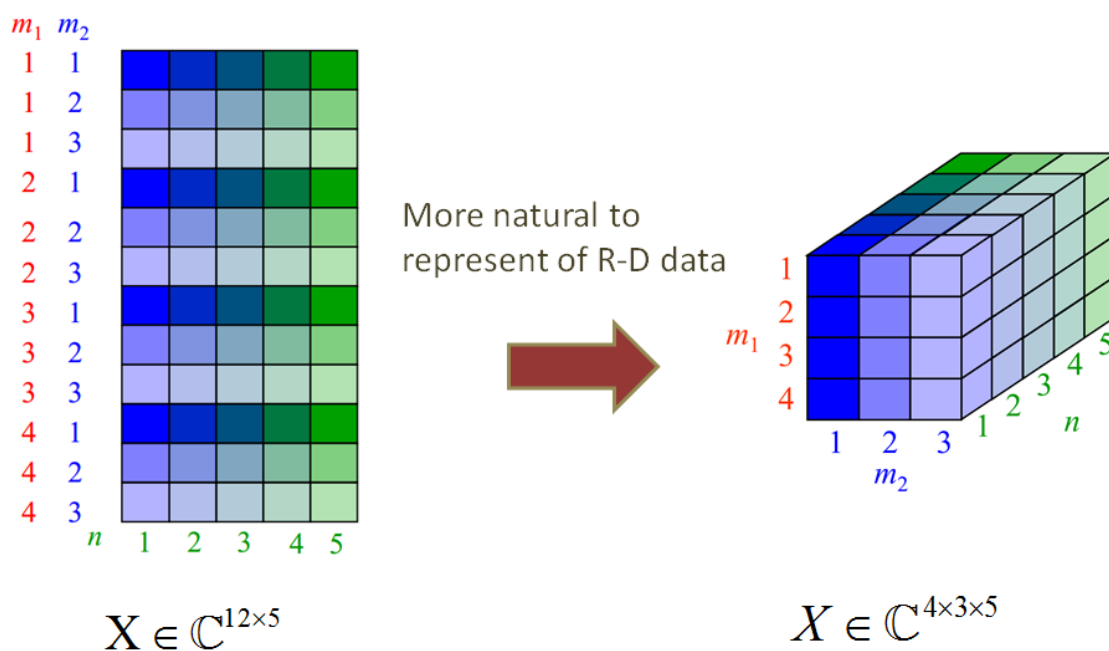


Figure 19. R-D data representation

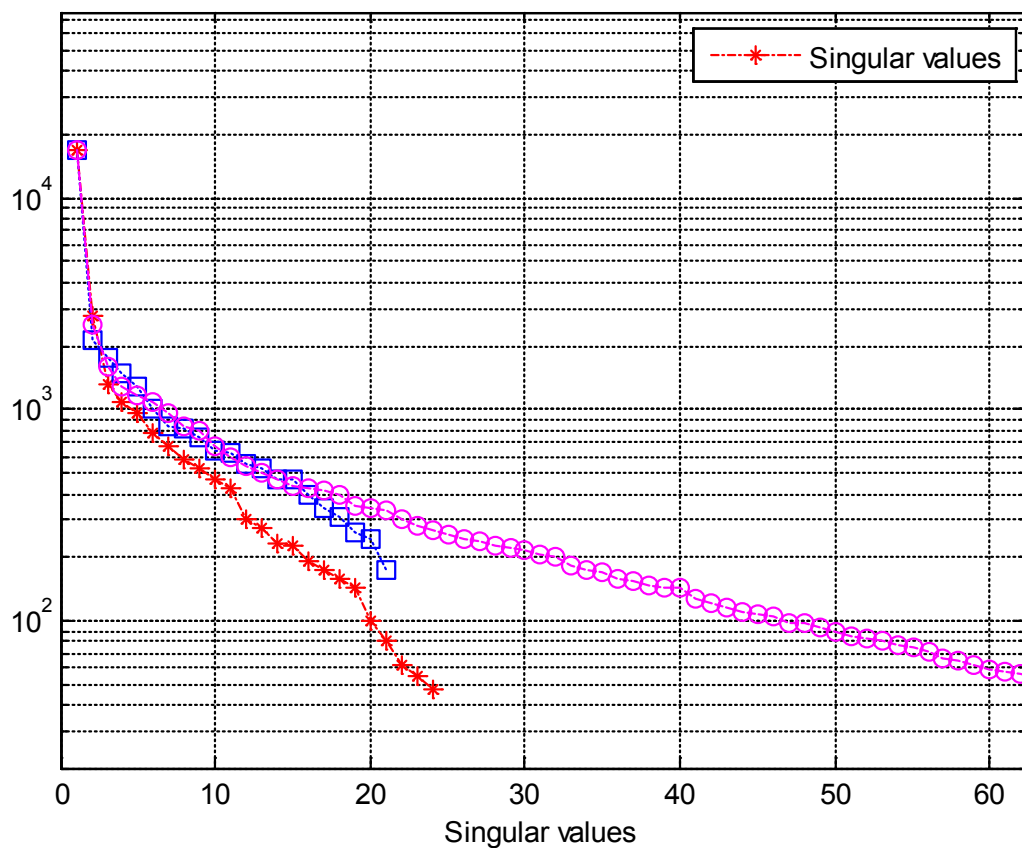


Figure 20. Singular values of CP decomposition rank one

The fraction at the first singular values of 3D tensor is a verification to find a low rank structure (Figure 20). The horizontal axis considered as singular values of three rank one vector which generate the main tensor. Also, the vertical axis nominated for singular values number. After calculating distortion for different number of parameters, rank-one components explained a significant fraction of data in both HOSVD and CP decomposition; rest seems to be mostly unstructured as it can be shown in Figure 21.

Via compression graphs we can get a rough idea that there is some low-rank (tensor) structure in our electrical grid data set, therefore we can typically gained from using tensors as the data has a low-rank structure that we can exploited. As illustrated, low rank (tensor) approximations performed ideally with compression graphs.

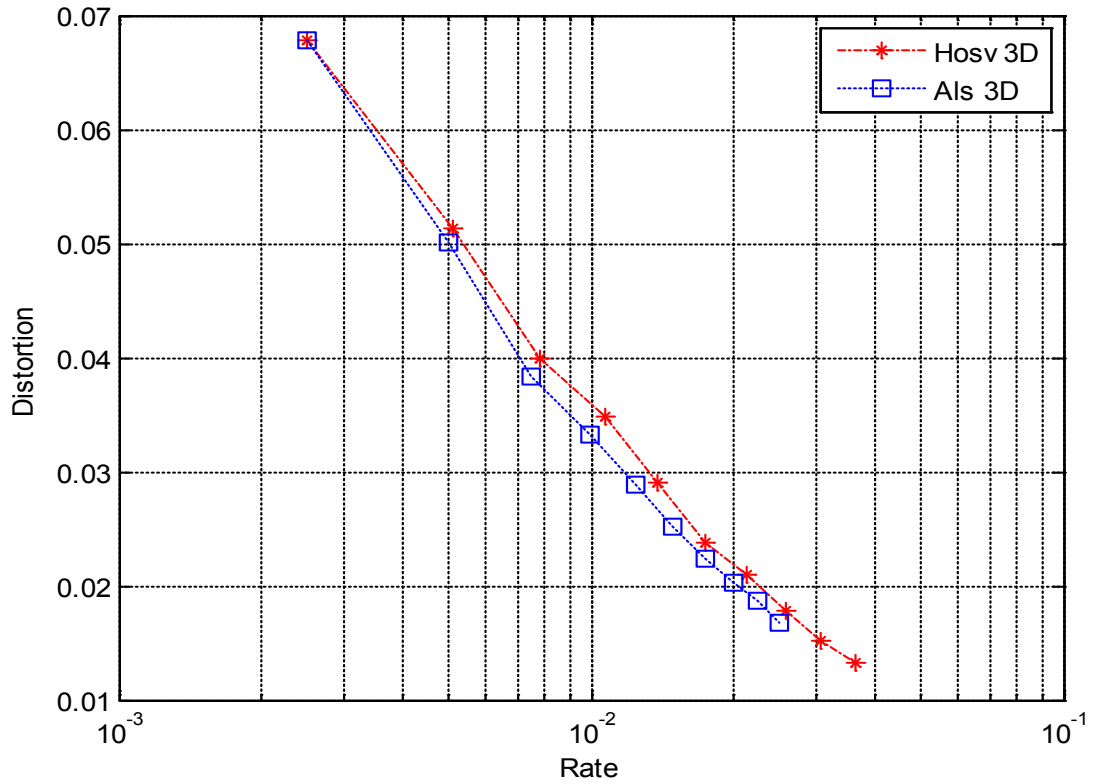
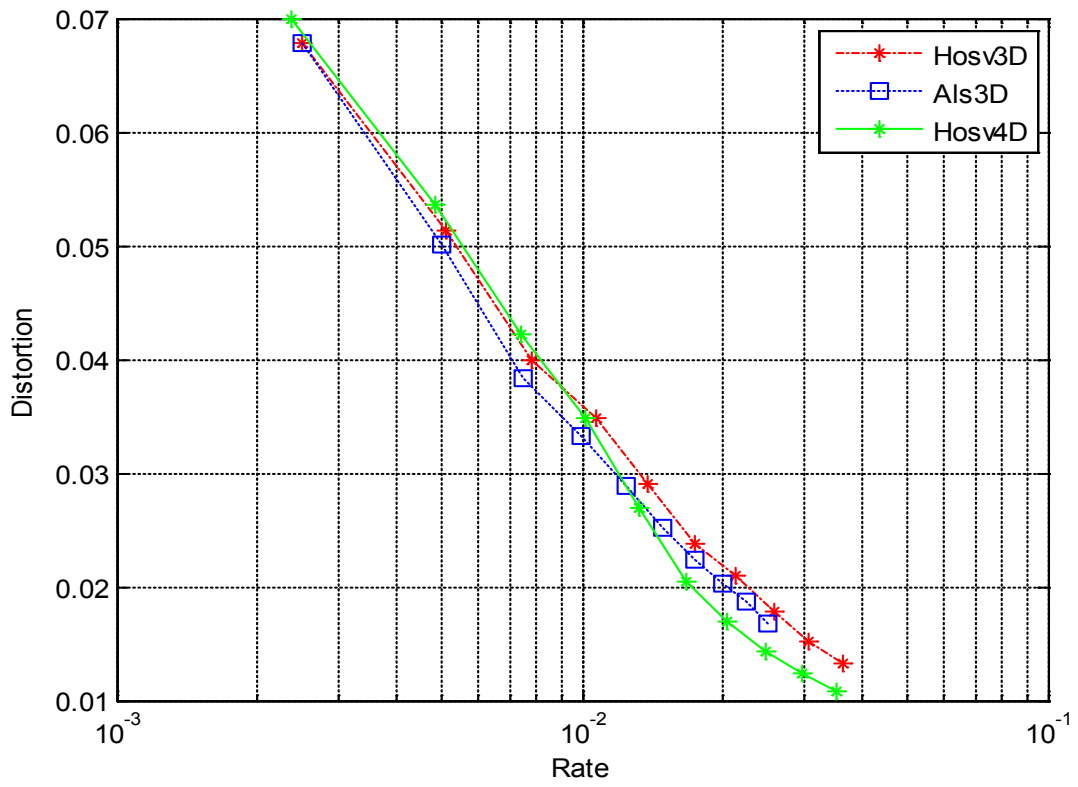


Figure 21. Distortion vs compression (3-D, 4-D tensor)

---

### 3.5. Analyzing the data:

We observed that there is a low-rank structure in form of a quite dominant rank-one component (periodic) on top of the rest of the data (which seems to be not periodic) by performing low rank tensor approximations, reconstruction error vs. compression.

#### **Time signature of dominant (rank-one) component:**

Convention which is used to specify how many samples (pulses) contained in each time is called the *time signature*.

Now we try to perform time signature of dominant (rank-one) component of the data, which is calculated by kronecker product of factor matrices obtained by CP decomposition-rank one. The CP decomposition calculated by Semi-algebraic framework for approximate CP decompositions via Simultaneous matrix diagonalization (SECSI) [10].

As mentioned before, the data set of power grid can be arranged as buses over time in a matrix of size  $600 \times 175$  for active powers. Therefore, the time axis can be broken into two dimensional space (24 hours x 21 days) or three dimensional space (24 hours x 7 days x 3 weeks) to generate respectively a 3-D tensor or 4-D tensor.

As it can be illustrated at the following Figures:

- ⇒ 2-D enforces no periodicity (Figure 22)
- ⇒ 3-D enforces periodicity over days (Figure 23)
- ⇒ 4-D enforces periodicity over days and weeks (Figure 24)

After careful consideration, we found that there is a multi-dimensional periodicity in 2D, 3D and also 4D data structure. There is an appropriate result because it means that tensors can potentially help in this method. Data structure in 2D, 3D and 4D illustrated a multi-dimensional periodicity which is can be useful for finding error location in our system. Time and bus number can be easily extracted from this kind of structure which is applicable to find error specifications.

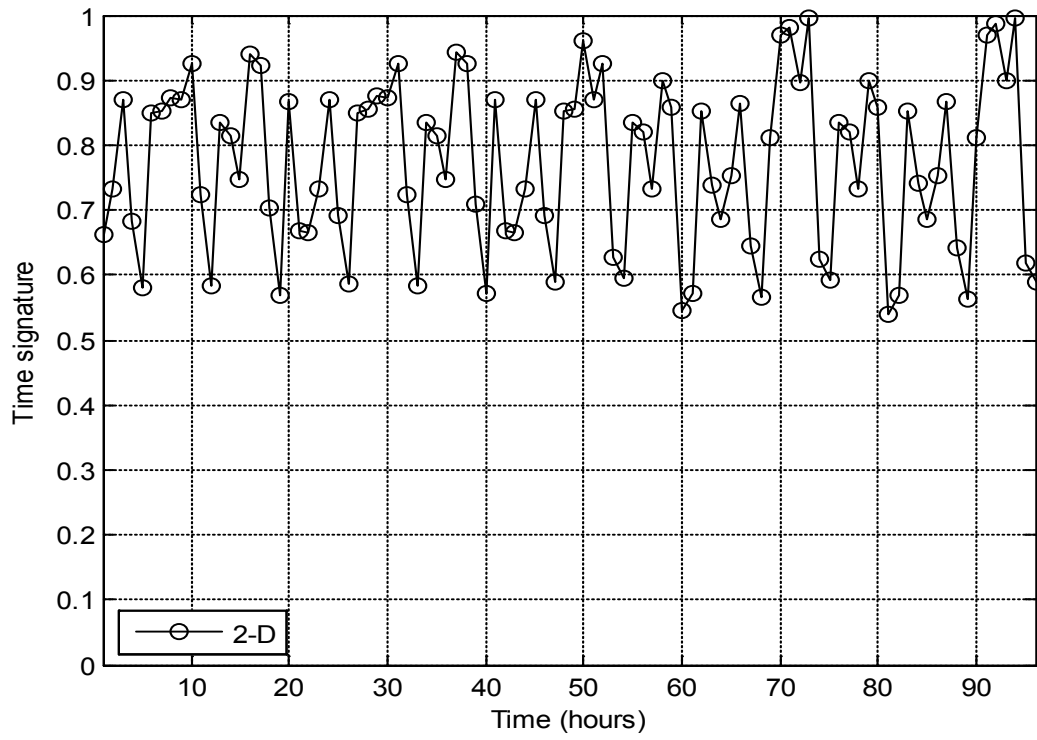


Figure 22. Time signature of dominant (rank-one) component 2D, CP decomposition

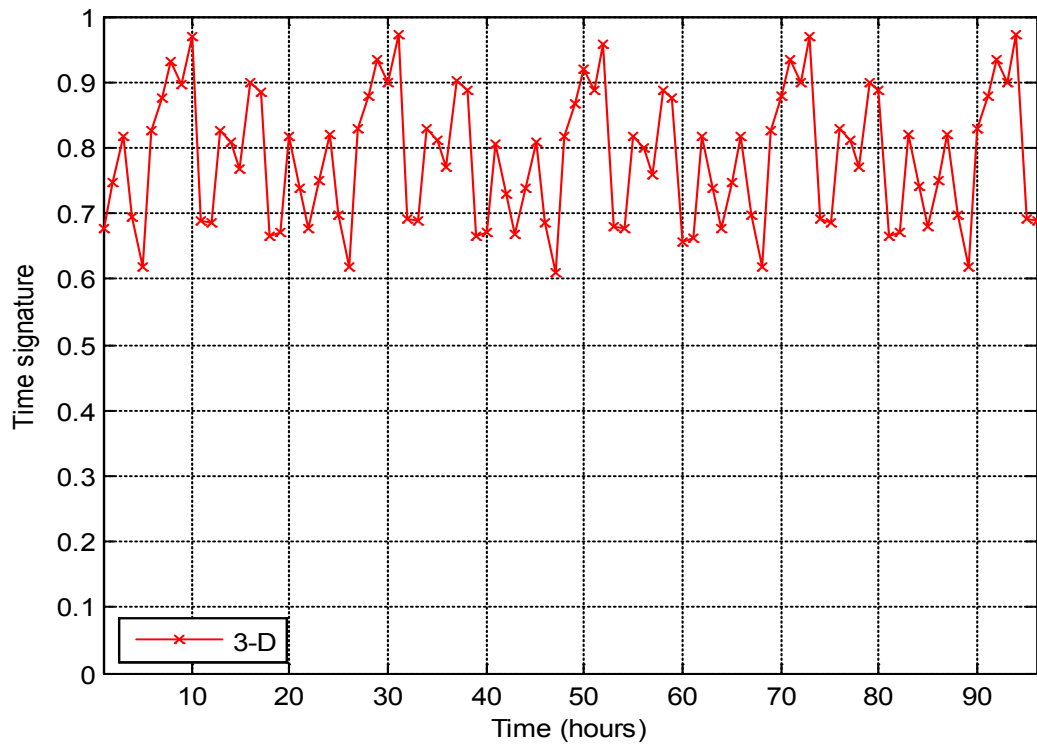


Figure 23. Time signature of dominant (rank-one) component 3D, CP decomposition



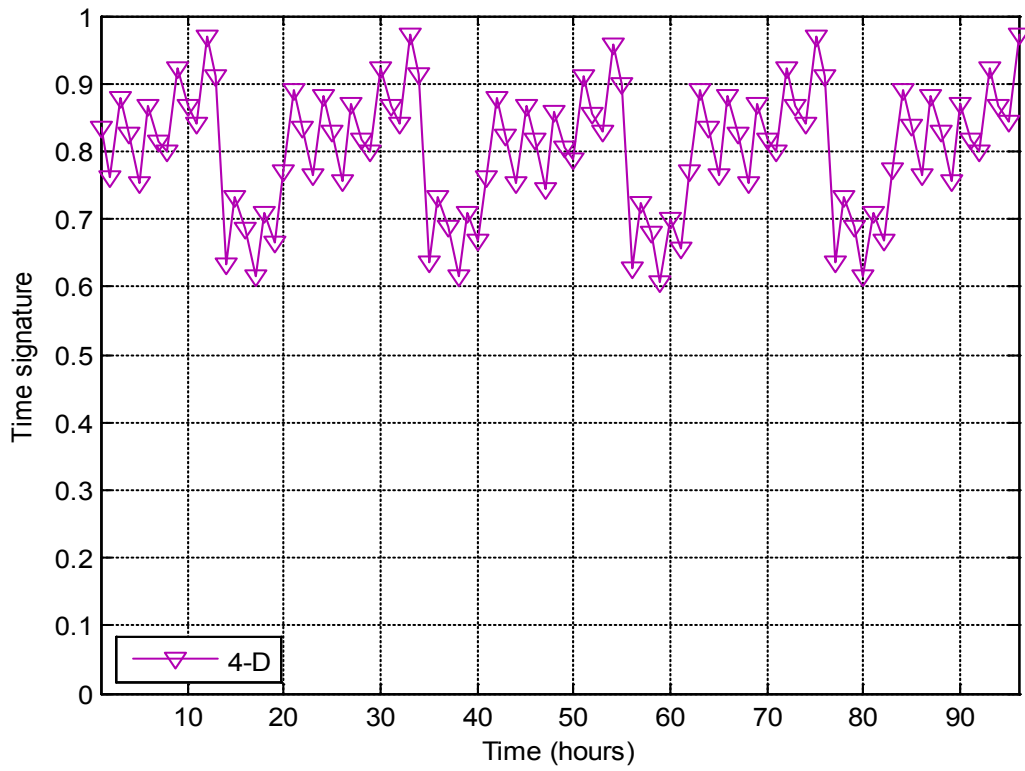


Figure 24. Time signature of dominant (rank-one) component 4D, CP decomposition

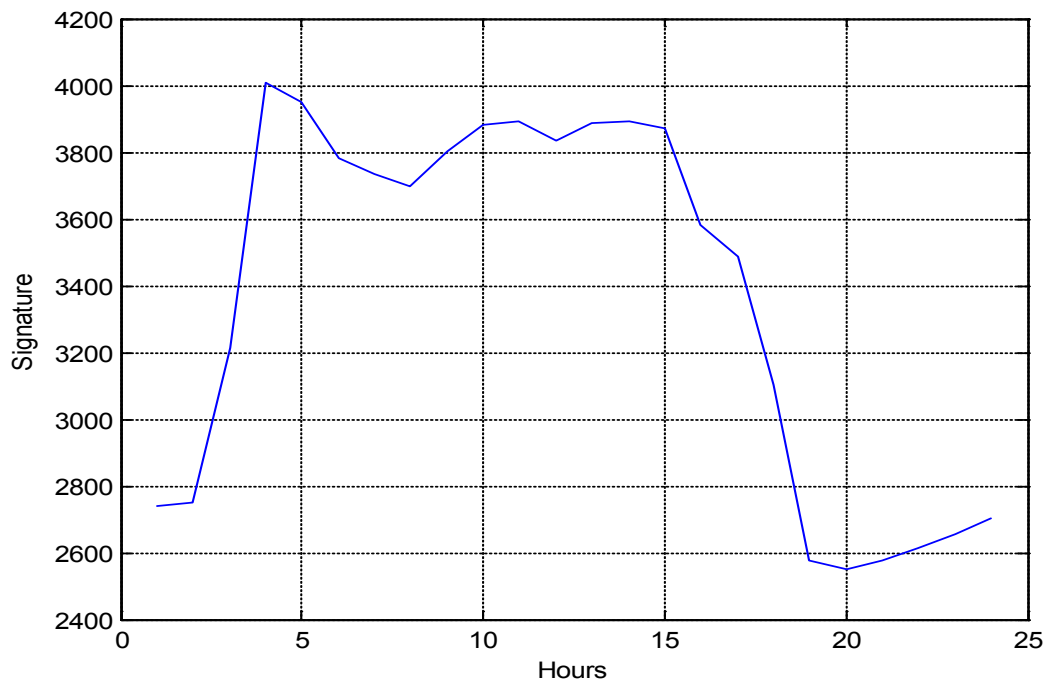


Figure 25. Daily time signature

As it can be seen in Figure 25, daily time signature of 175 buses versus 24 hours plotted. With a short glance we can find how many samples (pulses) are to be contained in each time. The peak is at 4 am, which is shown the consumption of active power is so high.

### Generalizing the method

Now to generalize the method, we perform all the previous steps for hourly sample rates of voltage instead of power for the 1000 buses that is shown in Figure 26. As before, data set can be arranged as time with different buses, which is generate a matrix of size  $600 \times 290$  for voltages.

### 3.6. Preprocessing

The starting point is to choose the interesting buses and hours which provide proper information without missing values for further processing.

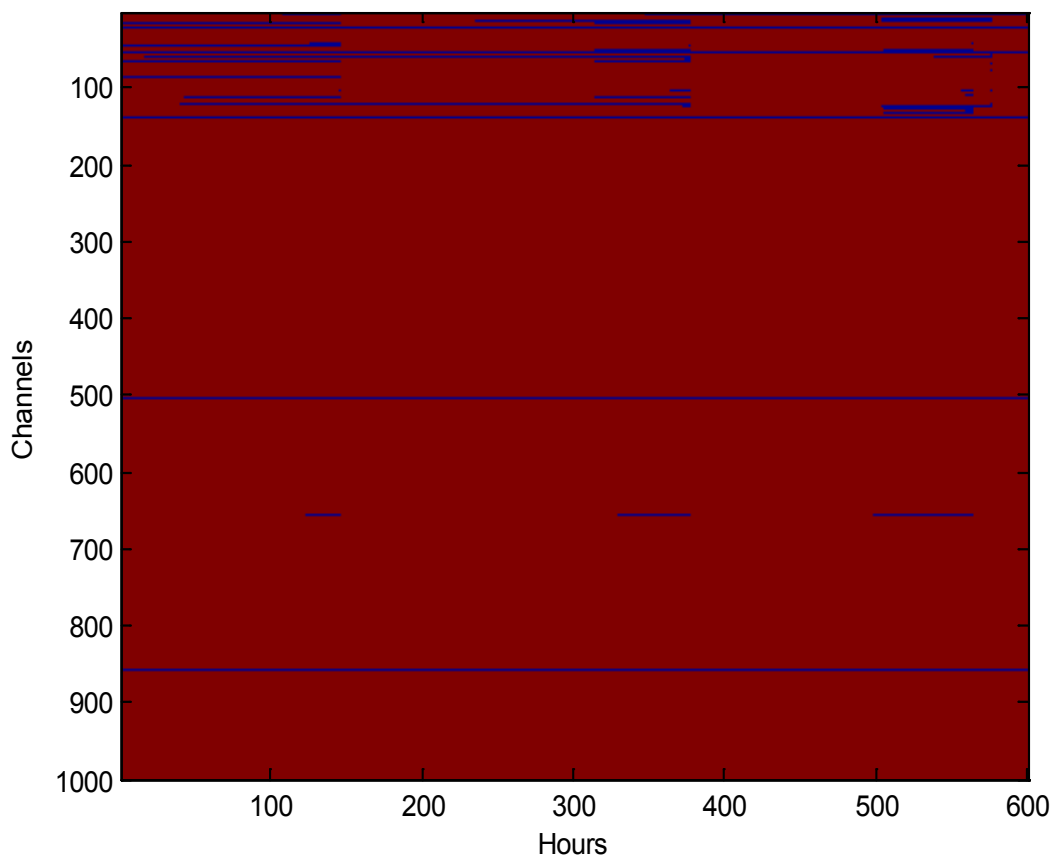


Figure 26. Interesting channels

### 3.6.1. Select interesting chunk (channels, days):

As it is illustrated in Figure 27, final selection resulted to choose 290 buses without any missing values out of 1000 buses, from bus number 159 to 448. Unlike the previous case, there is no need to remove any buses between this final selection. Therefore, we have a better data correlation between the buses. The known data painted with red color. The interesting data chunk is a third way tensor (290 channels, 600 hours) of voltages.

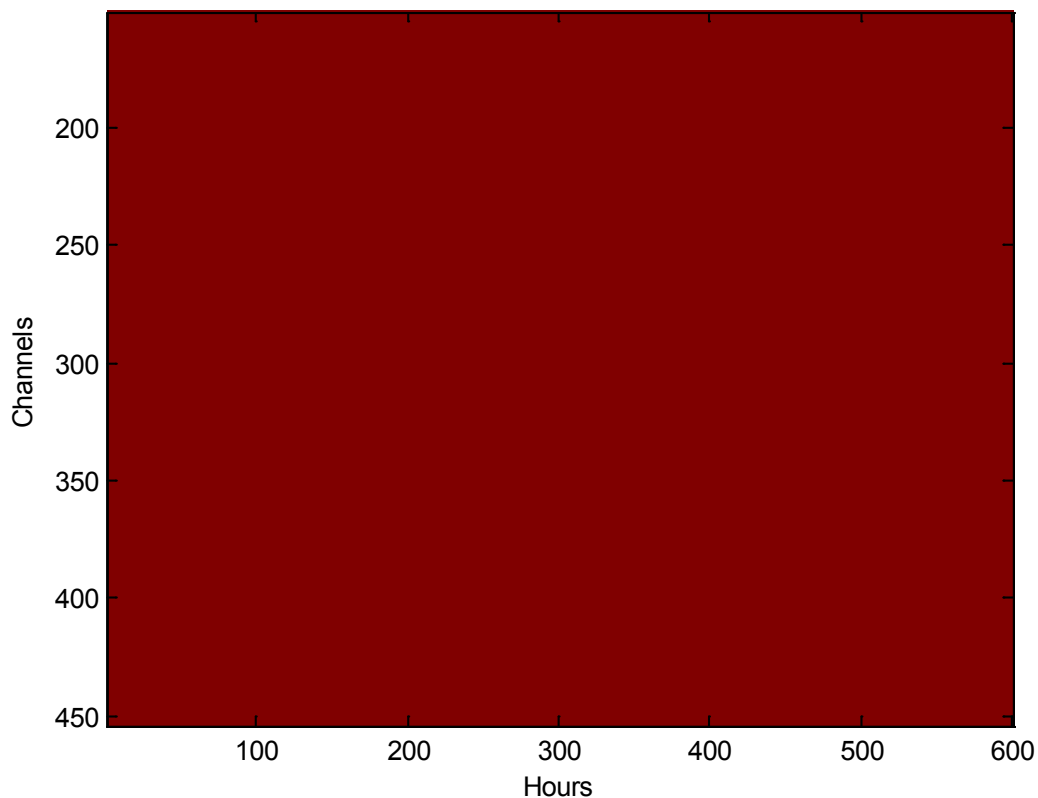


Figure 27. Interesting chunk

### 3.6.2 Interpolate missing values:

In this part we don't need any interpolation, as the interesting channels has no missing data and all the values are known. Once again, in this case we can reshape our tensors to 3-D and 4-D structure, because it would be more appropriate if the data set presented in R-D structure for further processing.

### 3.7. Reshape time axis of the data to 3D and 4D

Actually, the time axis can be broken into two dimensional space (24 hours x 21 days) to generate a 3-D tensor or can be reshaped to three dimensional space (24 hours x 7 days x 3 weeks) to generate a 4-D tensor.

- Selected buses                      voltage  
 ↓                                              ↙
- Data set (time ,159:448 ,3)
  - 3-D = Reshape (24,21,290);
  - 4-D = Reshape (24,7,3,290);

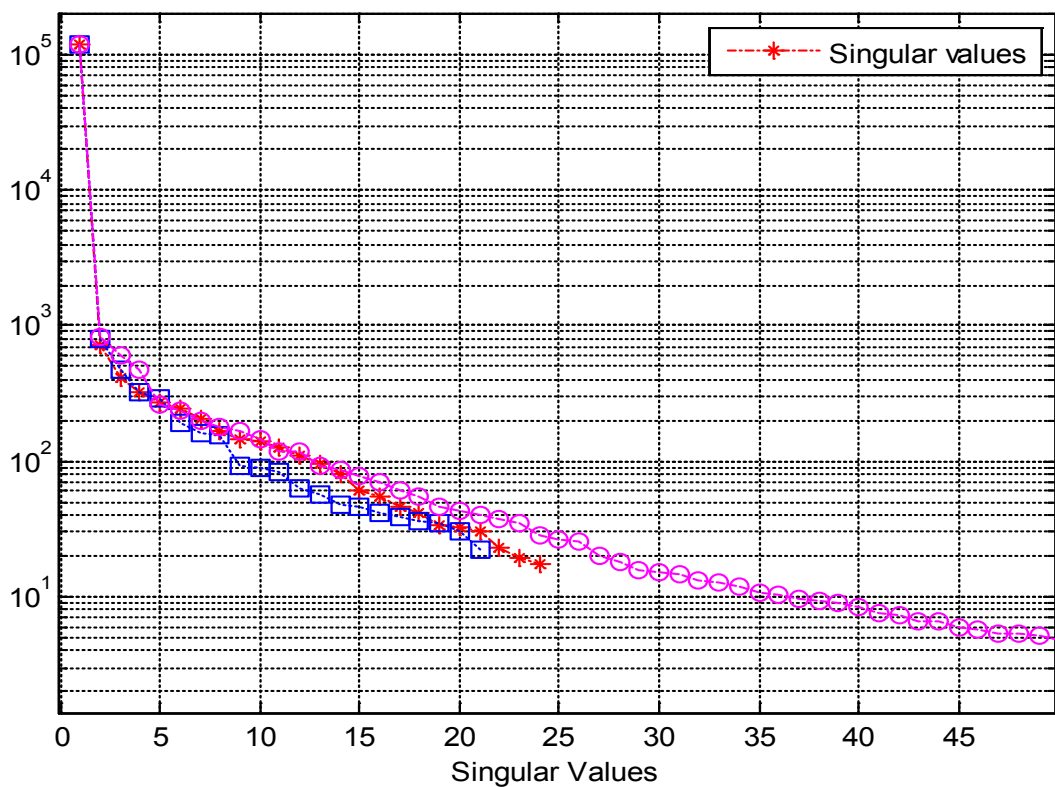


Figure 28. Singular values CP decomposition rank one

The fraction at the first singular values of 3D tensor is a verification to find a low rank structure (Figure 28). As it can be shown in Figure 29, the low rank tensor structure is visible. In other words, the fraction of data in CP-rank 1 and 1-rank HOSVD is visible.

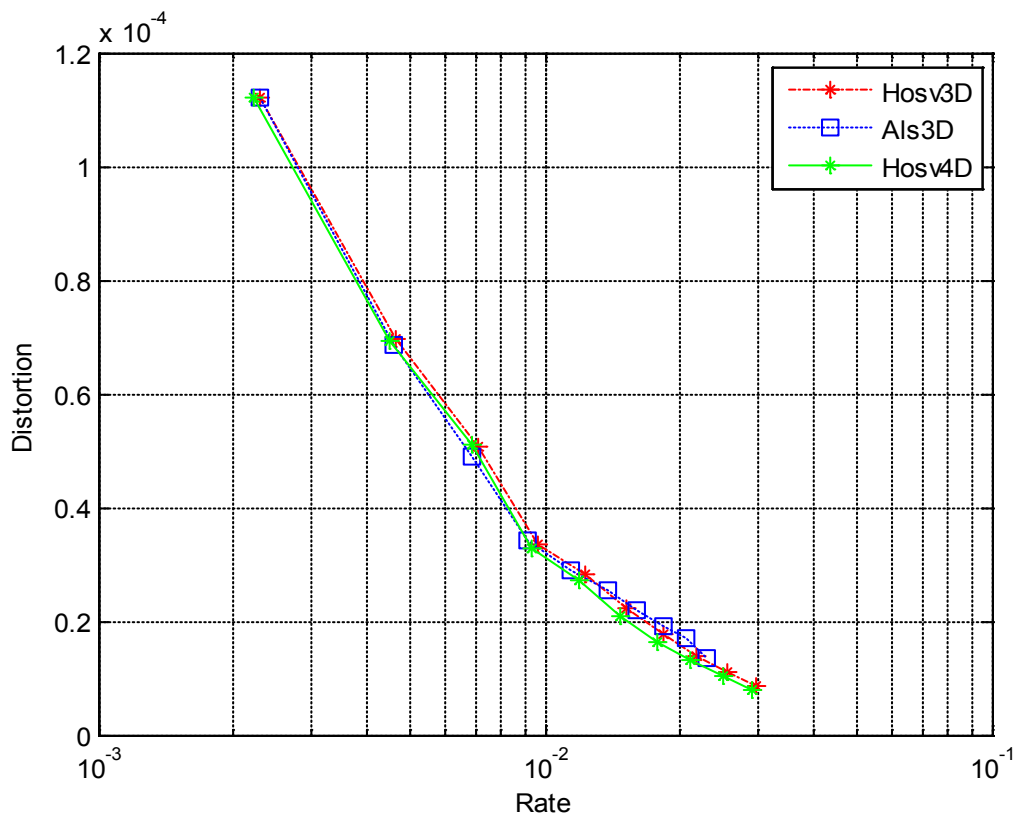
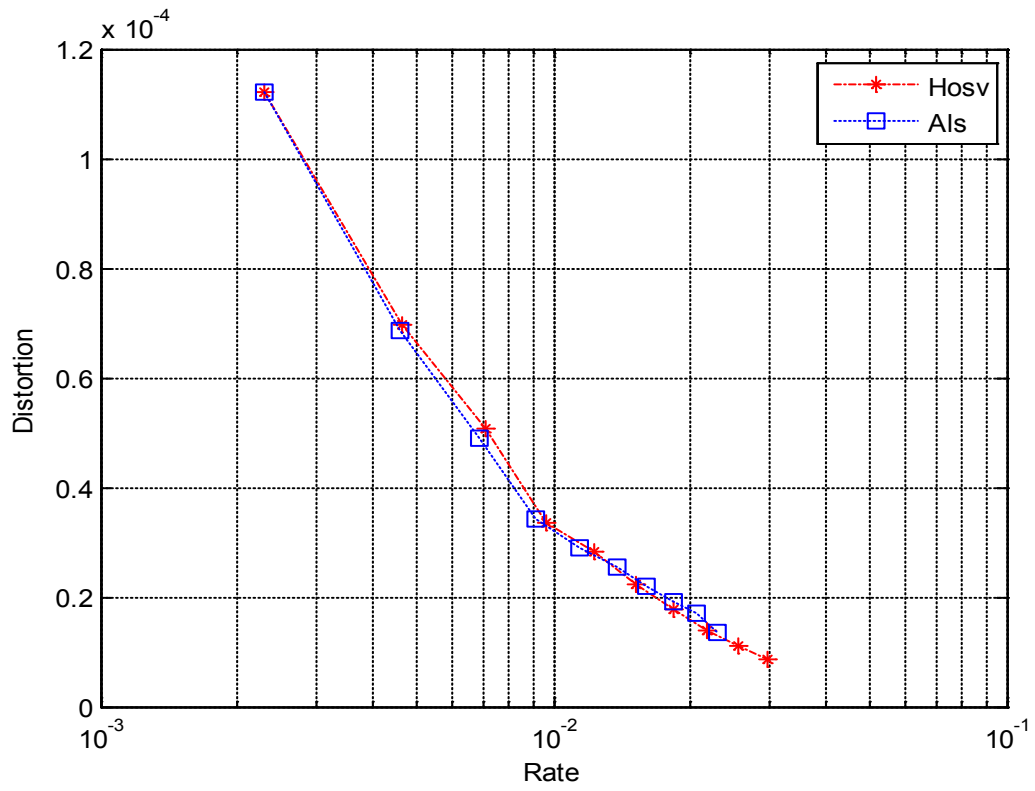


Figure 29. Distortion vs compression (3-D,4-D tensor)

---

Through these graphs we can get a rough idea that there is some low-rank (tensor) structure in our data set, therefore we typically gained from using tensors as the data has a low-rank structure that we can exploited.

### **3.8. Analyzing the data:**

We observed that there is a low-rank structure in form of a quite dominant rank-one component (periodic) on top of the rest of the data (which seems to be not periodic). After careful consideration, we found that there is a multi-dimensional periodicity in 2D, 3D and also 4D data structure. We can conclude that it is an appropriate result because it means that tensors can potentially help in this method. Data structure in 2D, 3D and 4D illustrated a multi-dimensional periodicity which is can be useful for finding error location in our system. Time and bus number can be easily extracted from this kind of structure which is applicable to find error specifications.

#### **Time signature of dominant (rank-one) CP decomposition components:**

As we can see in the following Figures:

- ⇒ 2-D enforces no periodicity (Figure 30)
- ⇒ 3-D enforces periodicity over days (Figure 31)
- ⇒ 4-D enforces periodicity over days and weeks (Figure 32)

To review this chapter, first we applied interpolation method to recover missing values from the selected channels. That interpolation provided a proper data structure to analyze multi-way power grid data. Then the evaluation of the benefits of using tensor-based processing by computing an approximate CP decomposition and High Order Singular Value decomposition (HOSVD) was performed sucessfully with compression graphs. After carefully studying both cases, we found that there is a multi-dimensional periodicity in 2D, 3D and also 4D data structure which is applicable to find system mulfunction by searching the time and bus of the error location in this periodic structure.

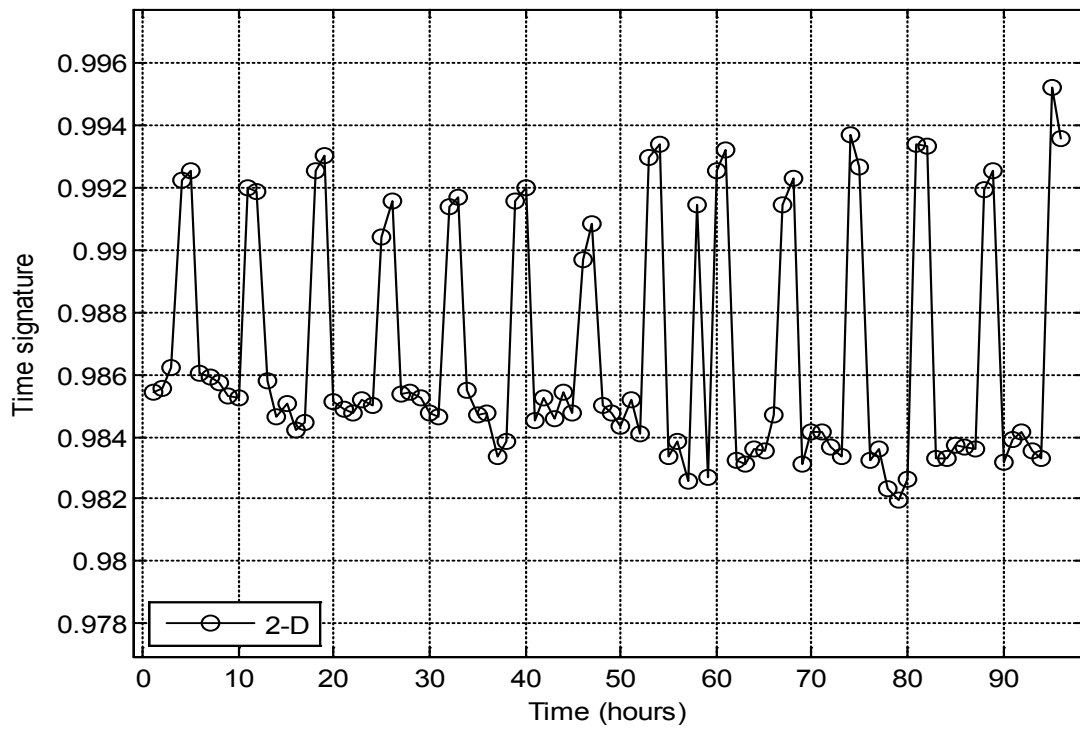


Figure 30. Time signature of dominant (rank-one) component 2D, CP decomposition

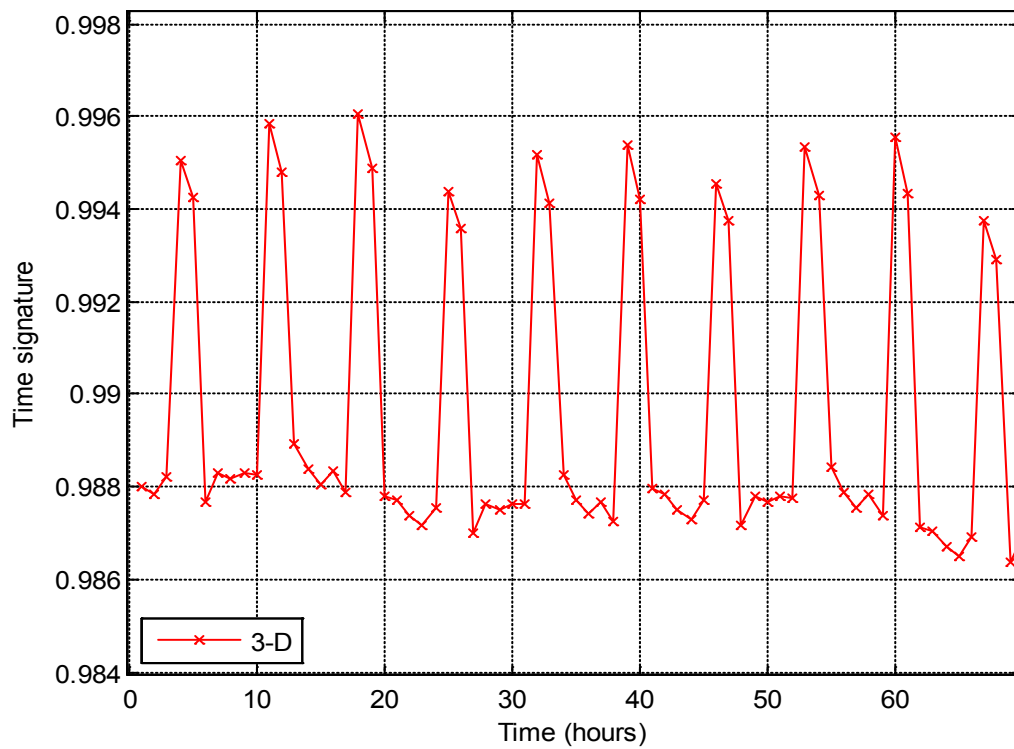


Figure 31. Time signature of dominant (rank-one) component 3D, CP decomposition

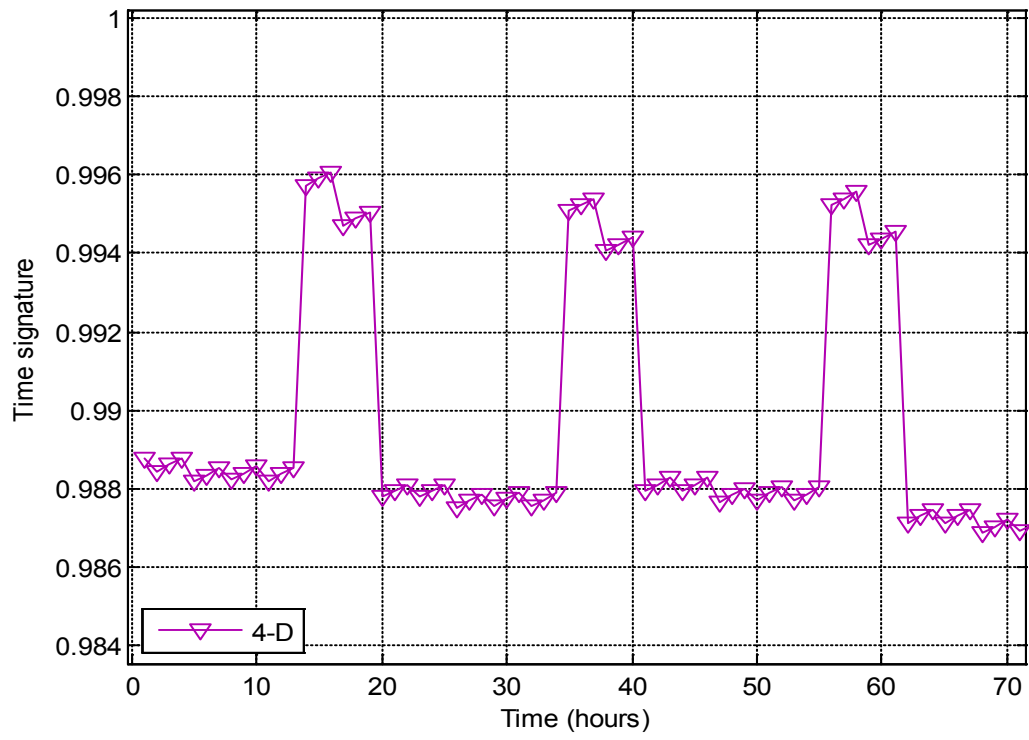


Figure 32. Time signature of dominant (rank-one) component, CP decomposition



## **4 WEIGHTED CP DECOMPOSITION DATA RECOVERY**

A central power system control can be used to improve the security (reliability) of power system generation and an extra high voltage (EHV) transmission line. For instance, we can consider the operation of such a central control system in two steps: 1) Primary information is processed in real time by a digital computer into an applicable data form; and 2) control decisions are handled from the processed information either by the digital console processor or by a human command.

In particular, real-time digital computer programs (algorithms) for converting the available raw informations (such as meter readings plus the other useful information), including possibly incorrect or missing data is the case that we will discuss in this work. Uncertainties appear in power system because of meter and communication link errors, incomplete metering process, errors in system models and unpredictable system changes. In power system, the goal of state estimator is to "analyze" the incoming data (such as real measurements, system status updation, and possibly pseudo measurements) and provide the rest of the control system with a "reliable" set of measurements (the state estimate) which are truly present the actual system status. Thus there is studies that can be use for system planning and monitoring. Moreover, it must be consider that no control system can effectively plan a proper state for power system to go in the future without some knowledge of its present state, so it shows a progresive demand for a correct system metering and monitoring.

In real power system, occasionally "bad data" will occur. That is, a data point will be very inaccurate or noisy. This can happen for a number of reasons. Perhaps there is a temporary failure in a communications link. Perhaps a meter is out of it's

measuring fault tolerance. Perhaps, the state of the system has changed quickly and the pseudo-measurements are very far from the change point. Any of a number of cases can happen which will cause a given reading data to be very inaccurate.

Missing data issue is a hidden problem in some areas. For instance, in biomedical signal processing, missing data can be happened during Electroencephalogram (EEG) measurement, where neuroheadset electrodes are placed to record the brain activity along the scalp. By losing or disconnecting one of the electrodes during the record, the signal can be lost or ignored because of impurity with high noise range. Missing data problem is not limited to one area, in other areas related to data mining, we also faced the same problem, such as packet losses in data communication along fibers or destructive correlation in images in computer vision.

For a power system which is controlled by a human being, recorder will automatically ignore or report a large missing error that is "obviously" incorrect. Unfortunately, computers don't perform the same task as people, and while it is partly easy for a control system operator to recognize certain types of bad data, it is quite difficult to program a computer to ignore and report the same task. These limitations, forced us to develop necessary methods for real-time monitoring and processing of the system variables and quantities. Furthermore, this mandatory demand is simultaneous with the advantage of more reliable computers and system processors and remarkable developments in analytical techniques which are the tools necessary to do this job.

If we cannot settle the issue of missing data, many important data sets will be discarded or improperly analyzed. But, this work contributes to the growing evidence that such data can be analyzed. By considering previous studies which have only reflected on matrices, we focus here on the problem of electrical grid data in multi-way arrays (tensors) as the data often have more than two modes of variation and are therefore best represented as multi-way arrays. For instance, in power grid, data of each record from a bus can be represented as a time-bus matrix; thus, data from multiple channels is three-dimensional (time, bus, and voltage or power) and forms a three-way array. These data can have multiple

dimensions, are often massively large, and generally have at least some missing data. Therefore, we need a robust and reliable approach for factorizing multi-way arrays in the presence of missing data.

In previous chapter, we found that via tensor decomposition we can get a rough idea that there is some low-rank (tensor) structure in our data set, so we typically gained from using tensors as the data has a low-rank structure that we can exploited. Moreover, we illustrated a multi-dimensional periodicity in the data, which is good because it means that tensors can potentially help.

This chapter goal sets to capture the underlying structure of the data through a higher-order tensor factorization. Tensor factorization performed in the presence of missing data. The algorithm formulated missing data in the context of tensor factorizations. There is a close relationship between tensor factorization and matrix completion from columns and rows, where we aim to recover the missing entries.

Tensor factorization (higher order factorization) algorithm emerged as an applicable method for information analysis. Instead of arranging data in tabular (matrix unfolding) multi-way arrays as matrices and applying matrix factorization techniques to recover the missing entries, tensor models keep multi-way nature structure of the data and generate the underlying latent factors in each mode (dimensions in Figure 2) of a higher-order array.

We focus here on the Canonical Polyadic Decomposition (CP decomposition), which is “workhorse” and commonly-used tensor decomposition model in various applications [1],[3]. Let  $X \in \mathbb{R}^{I \times J \times K}$  be a noisy three-way tensor of size  $I \times J \times K$ , and assume its rank is  $R$ . With complete data, the CP decomposition is indicated by factor matrices  $A$ ,  $B$ , and  $C$  of sizes  $I \times R$ ,  $J \times R$ , and  $K \times R$ , respectively and can be defined for a multiplication of three as following:

$$(11) \quad x_{ijk} \approx \sum_{r=1}^R a_{ir} b_{jr} c_{kr} \quad i = 1, \dots, I, j = 1, \dots, J, k = 1, \dots, K$$

In the presence of noise, the real tensor  $X$  is not objective and it is an approximate equality. Instead, the CP decomposition should minimize the error function which is indicated as following:

$$f(A, B, C) = \sum_{i=1}^I \sum_{j=1}^J \sum_{k=1}^K (x_{ijk} - \sum_{r=1}^R a_{ir} b_{jr} c_{kr})^2$$

An illustration of CP for third-order tensors is given in Figure 9 section 1.3. The CP decomposition easily can be extended to  $N$ -way tensors for  $N \geq 3$ . There are many algorithms to calculate a CP decomposition.

The starting point in case of missing data, is to find a way to attribute missing values in different way which can be perform by filling the entries with an estimation method such as mean. Next step summerized in a way to employ some analytical factorization technique such as multi-way factorization to impute again the missing values.

Another method which we used in this work, is to weight error function in order to ignore missing values and consider only the known values. Alternating technique, which first calculate the factor matrices and then combined them with iterative attribution can be practibale and robust as easy to perform. We must consider that by increasing the number of missing data, alternating method may go out of convergence and have a wrong result as the initialization and primary model used to attribute the missing values, discussed at [7]. In this work, to avoid wrong convergence of alternating techniques by increasing the number of missing values, we applied nonlinear conjugate gradient optimization to compute the weighted least squares problem which is formulated as CP model.

The weighted version of (11) is:

$$(12) \quad f(A, B, C) = \sum_{i=1}^I \sum_{j=1}^J \sum_{k=1}^K \left\{ w_{ijk} \left( x_{ijk} - \sum_{r=1}^R a_{ir} b_{jr} c_{kr} \right) \right\}^2$$

Where  $W$  is a nonnegative weight multi-way array, the same size as  $X$ , by considering 1 for known entries and 0 for missing one:

$$w_{ijk} = \begin{cases} 1 & \text{if } x_{ijk} \text{ is known} \\ 0 & \text{if } x_{ijk} \text{ is missing} \end{cases}$$

Using the notation defined in section 1.3.2, (12) can be rewritten as:

$$f(A, B, C) = \left\| W * (X - \llbracket A, B, C \rrbracket) \right\|^2$$

Let  $X \in \mathbb{R}^{I \times J \times K}$  be a three-way tensor of size  $I \times J \times K$ , and assume its rank is  $R$ . In the presence of noise, the actual tensor  $X$  is not observable and there is an approximate equality.

Let's present the algorithm in three steps, first we will discuss about tensor factorization and its properties, then define a method to factorize by CP decomposition and the last step is to extend it to weighted optimization with nonlinear conjugate gradient.

The main concepts and their functions are summarized as follows:

- Using a scalable algorithm called CP Weighted Optimization (CP-WOPT) which is applicable for tensor factorizations in the presence of missing values. CP Weighted Optimization applies first order optimization to solve the weighted least squares error function (12)[1].
- Performing numerical experiments for simulated data sets, which shown that CP Weighted Optimization can properly recover tensors with missing values.
- Applying CP\_WOPT algorithm over power grid multi-way data to recover actual tensor in presence of missing values, then comparing the result with reference to interpolation method where we aimed to find low rank tensor structure in our power grid data, as illustrated in previous chapter.

#### 4.1. Factorization method in presence of missing data

To summarize this section, based on matrix factorization first we presented research approaches for tensor factorizations to overcome missing values problem, then discussed the techniques for extending tensor factorizations to weighted least squares.

##### 4.1.1 Generalize matrix factorizations to tensor factorizations

The matrix factorizations can be considered as a main concept to re-formulate it for tensor factorizations in presence of missing data and recover the actual tensor values. If we consider the perfect tensor model as  $\llbracket A, B, C \rrbracket$ , then recover the missing entries of  $X$  to generate a complete tensor according to

$$(13) \quad \overline{X} = W * X + (1 - W) * \llbracket A, B, C \rrbracket$$

Where 1 defined a multi-way array where all entries are equal to one. Using alternating least squares (ALS), we performed some multiple CP decompositions with different numbers of components of factor matrices until one combination is fit  $X$ .

#### 4.2. CP Weighted Optimization algorithm

Let's consider a general N-way structure for CP factorization to generate a complete tensor in presence of missing values. Assume  $X$  be a tensor of size  $I_1 \times I_2 \times \dots \times I_N$  and consider R for it's rank. Suppose the model influenced by a weight tensor  $W$  of the same size as  $X$  such that:

$$(14) \quad w_{i_1 i_2 \dots i_N} = \begin{cases} 1 & \text{if } x_{i_1 i_2 \dots i_N} \text{ is known} \\ 0 & \text{if } x_{i_1 i_2 \dots i_N} \text{ is missing} \end{cases}$$

We aim to find factor matrices  $A^{(n)} \in \mathbb{R}^{I_n \times R}$  for  $n = 1, \dots, N$  which is minimize the weighted objective function §4.2.1.

In other words, our mission is to solve:

$$\min_{A^{(1)}, \dots, A^{(N)}} fw(A^{(1)}, \dots, A^{(N)})$$

In the following, we will discuss the weighted error function in §4.2.1 and the procedure for calculating its gradient in §4.2.2 in more details. As soon as we found its gradient, we can apply different gradient-based optimization techniques [9] to solve the optimization problem.

#### 4.2.1. Error function

The N-way objective function is indicated as:

$$(15) \quad fw(A^{(1)}, \dots, A^{(N)}) = \left\| W * (X - \llbracket A^{(1)}, \dots, A^{(N)} \rrbracket) \right\|^2 = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \dots \sum_{i_N=1}^{I_N} w^2_{i_1 i_2 \dots i_N} \dots \left\{ x^2_{i_1 i_2 \dots i_N} - 2x_{i_1 i_2 \dots i_N} \sum_{r=1}^R \prod_{n=1}^N a_{i_n r}^{(n)} + \left( \sum_{r=1}^R \prod_{n=1}^N a_{i_n r}^{(n)} \right)^2 \right\}$$

Using tensor operation techniques we can efficiently calculate this function, therefore there is no need for element-wise computations [2].

If we pre-compute  $Y = W * X$  and  $Z = W * \llbracket A^{(1)}, \dots, A^{(N)} \rrbracket$ , then we can conclude that:

$$fw = \|y - z\|^2$$

### 4.2.2. Gradient (Partial derivatives)

Deriving the gradient of (15) by computing the partial derivatives of  $f_W$  with respect to each element of the factor matrices, i.e.,  $a_{i_n r}^{(n)}$  for all  $i_n = 1, \dots, I_n$ ,  $n = 1, \dots, N$ , and  $r = 1, \dots, R$ .

The partial derivatives of the objective function  $f_W$  in (15) are given by

$$\frac{\partial f_W}{\partial a_{i_n r}^{(n)}} = 2 \sum_{i_1=1}^{I_1} \dots \sum_{i_{n-1}=1}^{I_{n-1}} \sum_{i_{n+1}=1}^{I_{n+1}} \dots \sum_{i_N=1}^{I_N} w_{i_1 i_2 \dots i_N}^2 \left( -x_{i_1 i_2 \dots i_N} + \sum_{l=1}^R \prod_{m=1}^N a_{i_m l}^{(m)} \right) \prod_{\substack{m=1 \\ m \neq n}}^N a_{i_m r}^{(m)}$$

```

% assume Y = W * X is precomputed
Z = W * [A^(1), ..., A^(N)]
% function computation
fW = ||Y||^2 - 2<Y, Z> + ||Z||^2
% gradient computation
FOR n = 1 TO N
    G^(n) = -2Y_(n)A^(-n) + 2Z_(n)A^(-n)
END

```

Figure 33. CP-WOPT computation of function value  $f_W$  and gradient  $G^{(n)} = \frac{\partial f_W}{\partial A^{(N)}}$ . It is possible to make

this implementation more efficient by computing  $G^{(n)} = -2(Y_{(n)} - Z_{(n)})A^{(-n)}$



Once again, by using tensor operation techniques we can efficiently calculate gradient, therefore there is no need for element-wise computations. In matrix notation, we can rewrite the gradient equation as

$$(16) \quad \frac{\partial f_w}{\partial \mathbf{A}^{(N)}} = 2 \left( \mathbf{Z}_{(n)} - \mathbf{Y}_{(n)} \right) \mathbf{A}^{(-n)}$$

Where,

$$\mathbf{A}^{(-n)} = \mathbf{A}^{(N)} \diamond \dots \diamond \mathbf{A}^{(n+1)} \diamond \mathbf{A}^{(n-1)} \diamond \dots \diamond \mathbf{A}^{(1)}$$

for  $n = 1, \dots, N$ . Remembered (chapter one) that the symbol  $\diamond$  denotes the Khatri-Rao product and the star  $*$  denotes Hadamad product.

The computations in (16) exploit the fact that  $W$  is binary, see (14), such that:

$$W^2 * X = W * X = Y,$$

$$Z = W^2 * \left[ \left[ \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \right] \right] = W * \left[ \left[ \mathbf{A}^{(1)}, \dots, \mathbf{A}^{(N)} \right] \right]$$

Matricized tensor times Khatri-Rao product is the primary computation in (16) and can be computed efficiently [5]. The algorithm is summarized in Figure 33. Now that we have the gradient, we can apply nonlinear conjugate gradient (NCG) [9] as a gradient-based optimization technique to solve the optimization problem.

### 4.3. Nonlinear conjugate gradient

Nonlinear conjugate gradient (NCG) methods are used to solve unconstrained nonlinear optimization problems. It is based on conjugate gradient iterative method for solving linear systems which is adapted to solve unconstrained nonlinear optimization problems. The Poblano function which is applicable for the nonlinear conjugate gradient methods is called NCG. There are some general steps of the NCG method which are given below in pseudo code:

1. Input:  $x_0$ , a starting point
2. Evaluate  $f_0 = f(x_0)$ ,  $g_0 = \nabla f(x_0)$
3. Set  $p_0 = -g_0, i = 0$
4. while  $\|g_i\| > 0$ 
  5. Compute a step length  $a_i$  and set  $x_{i+1} = x_i + a_i p_i$
  6. Set  $g_i = \nabla f(x_{i+1})$
  7. Compute  $\beta_{i+1}$
  8. Set  $p_{i+1} = -g_{i+1} + \beta_{i+1} p_i$
  9. Set  $i = i + 1$
10. end while
11. Output:  $x_i \approx x^*$

Take into consideration to set  $\beta_{i+1}$  zero in cases where the update coefficient is negative to avoid directions that are not descent directions.

The choice of  $\beta_{i+1}$  in Step 7 leads to different NCG methods. The update methods for  $\beta_{i+1}$  are listed in Table 2. The special case of  $\beta_{i+1} = 0$  leads to the steepest descent method:

Tabel 2. Conjugate direction updates available

Update Type	Update Formula
Fletcher-Reeves	$\beta_{i+1} = \frac{g_{i+1}^T g_{i+1} - g_{i+1}^T g_i}{g_i^T g_i}$
Polak-Ribiere	$\beta_{i+1} = \frac{g_{i+1}^T (g_{i+1} - g_i)}{g_i^T g_i}$
Hestenes-Stiefel	$\beta_{i+1} = \frac{g_{i+1}^T (g_{i+1} - g_i)}{p_i^T (g_i - g_i)}$

---

## 4.4. Experiments

In this experiment we aim to analyze the performance of the CP Weighted Optimization algorithm and estimate its capability to recover a multi-way array which contains missing values. The experiment proved that CP factor matrices can efficiently recover multi-way arrays even if we encountered a significant percentage of missing entries or fibers.

CP Weighted Optimization is formulated using tensor operation techniques (Tensor Toolbox) [4], based on the partial derivatives and objective function calculations shown in Figure 33. This algorithm used the nonlinear conjugate gradient (NCG) method with Hestenes-Stiefel updates [9] for optimization problem. CP Weighted Optimization is an iterative method.

Replacing the missing values of left singular vectors  $X_{(n)}$  (matrix unfolding) with zero is the starting point of this algorithm.

In this experiment, the stopping criteria are considered as follows [1]:

1. Set stopping condition ( $10^{-6}$ ) for the related change in the function value  $f_w$  in (12).
2. Set the value of  $10^{-8}$  for the tolerance ratio between two-norm of the gradient and the number of entries in the gradient.
3. The maximum number of iterations and the maximum number of function evaluations are set to  $10^3$  and  $10^4$ , respectively.

### 4.4.1. Simulated data

In this section we aim to introduce multi-way arrays structure which are recovered. Three way tensors generated with ranks 5 and 10, and different sizes. The other fact that we employed was missing values percentage and also the patterns which can be single entries or fibers. The tensor sizes considered as  $50 \times 50 \times 50$  or  $150 \times 150 \times 150$ . The true tensor generated randomly with factor matrices A, B and C with related sizes.

We aimed to find factor matrices which is minimized the weighted objective function. From each set of factor matrices, a third-order tensor,  $T = \llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket$ , is generated.

To add noise we considered another tensor,  $N \in \mathbb{R}^{I \times J \times K}$ , the same size as  $T$ . the entries generated randomly and have a normal distribution as follows:

$$X = T + (100 / \eta - 1)^{-1/2} \frac{\|T\|}{\|N\|} N$$

$\eta\%$  defined as the noise percentage. For sake of simplicity, the value  $\eta = 2$  is used in our experiments [7].

The last step is to remove randomly some tensor entries in order to generate missing values. The patterns of missing values can be single entries or fibers. We considered three different missing values percentage 10%, 40%, and 70%.

To remove some tensor entries, we ignore the cases which we miss a complete tensor slice because it is not possible to recover the factor matrices in such cases. The same problem as we have in matrix completion, which recover matrix from a column or row is our goal, so we ignore the cases when the entire row or column turns out to be missing, as the matrix cannot be correctly recover due to lack of information.

Let's say that the factor matrices should be efficiently recovered if the following equation satisfied. Let  $\bar{\mathbf{A}}, \bar{\mathbf{B}}, \bar{\mathbf{C}}$  be the recovered factor matrices. We require, for all  $r \in \{1, \dots, R\}$ :

$$\text{sim}(\mathbf{r}) = \frac{|\mathbf{a}_r^T \bar{\mathbf{a}}_r|}{\|\mathbf{a}_r\| \|\bar{\mathbf{a}}_r\|} \times \frac{|\mathbf{b}_r^T \bar{\mathbf{b}}_r|}{\|\mathbf{b}_r\| \|\bar{\mathbf{b}}_r\|} \times \frac{|\mathbf{c}_r^T \bar{\mathbf{c}}_r|}{\|\mathbf{c}_r\| \|\bar{\mathbf{c}}_r\|} \succ 0.97 \approx (0.99)^3$$

Unlike the matrix decomposition, CP decompositions are often unique which is an interesting property of higher-order tensors. In other words, to generate tensor  $T$ , there is just only one possible summation of factor matrices, which means Uniqueness. The uniqueness property of CP decomposition gives an applicable tool to compare the recovered factor matrices with the generated factor matrices. The percentage of successful recovery out of different running times, gives us the method accuracy.

Table 3 and 4 demonstrate the CP Weighted Optimization accuracy. Ten sets of factor matrices were generated for different tensor rank (5,10). Each entry in the table nominated as the percentage of completely recovered underlying factors out of the ten corresponding CP models. In the case of randomly missing entries, CP Weighted Optimization can completely recover the factor matrices even in presence of 70% missing data. It can be shown through Table 3 that high percentage of accuracy achieved for different sizes and ranks in case of missing entries.

Tabel 3. Accuracy for randomly missing entries

Missing Data	10%		40%		70%	
	R=5	R=10	R=5	R=10	R=5	R=10
$50 \times 50 \times 50$	100.0	100.0	100.0	100.0	100.0	100.0
$150 \times 150 \times 150$	100.0	100.0	100.0	100.0	100.0	100.0

Tabel 4. Accuracy for randomly missing fibers

Missing Data	10%		40%		70%	
	R=5	R=10	R=5	R=10	R=5	R=10
$50 \times 50 \times 50$	100.0	100.0	100.0	100.0	89.3	81.5
$150 \times 150 \times 150$	100.0	100.0	100.0	100.0	100.0	100.0

On the other hand, Table 4 demonstrates the percentage of accuracy which is achieved by recovering true tensor out of ten corresponding factor matrices for missing fibers pattern. As it can be shown, CP Weighted Optimization only has less accuracy when it has 70% missing fibers for a tensor with size  $50 \times 50 \times 50$ .

Here's something we should pay attention to that is the smaller problem size are more difficult to converge due to the following reason.

If we consider a tensor of size  $I \times I \times I$  with proportion  $M$  of missing data. Let  $D$  be indicated by

$$D = \frac{\text{Number of known tensor entries}}{\text{Number of variables(degree of freedom)}} = \frac{(1-M)I^3}{3RI}$$

$D$  is inversely proportional to the difficulty of the problem. We have an easier problem to solve if  $I$  increases as long as  $M$  and  $R$  constant. Figure 34 illustrates how accuracy can be change with respect to  $D$  for both ranks 5 and 10.

For instance,  $D$  is 50 for smaller size tensors in case of missing fibers for  $M = 70\%$  missing data and  $R = 5$ , and accuracy of algorithm is 89.3%; on the other hand  $D$  is 450 for tensors of size  $150 \times 150 \times 150$ , and accuracy of CP Weighted Optimization algorithm reaches to 100%. It is worth noting that we do not use the any initialization in the results proposed here, which is suggested in [7].

The experiments prove that the factor matrices can be recovered even in presence of significant amount of missing values. This is because the tensor analysis that we applied is inspired by (subspace-based) low-rank modeling.

A rank- $R$  tensor of size  $I \times J \times K$  has  $R(I + J + K)$  degrees of freedom. If the numbers of data are more than variables, then suggested tensors can be recover even with high percentage of missing values. In other words, if the size of data defined as  $0.3IJK$ , then the number of variables  $R(I + J + K)$  is much less than the data size. As it is a nonlinear problem, the number of entries which we need to recover the CP models inspired by (subspace-based) low-rank modeling are unknown.

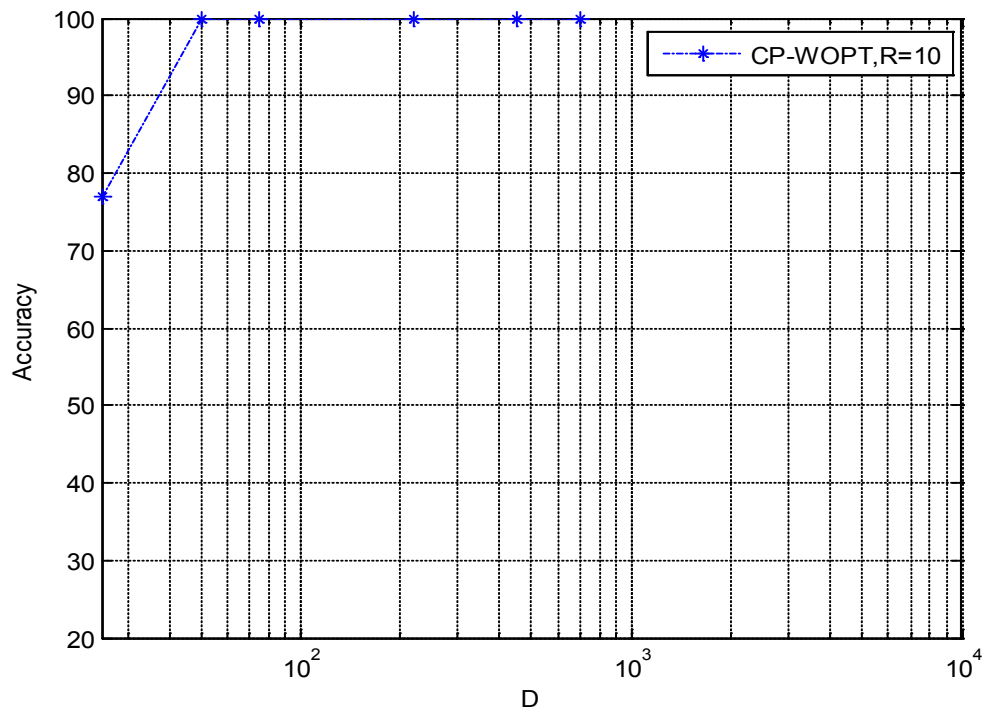
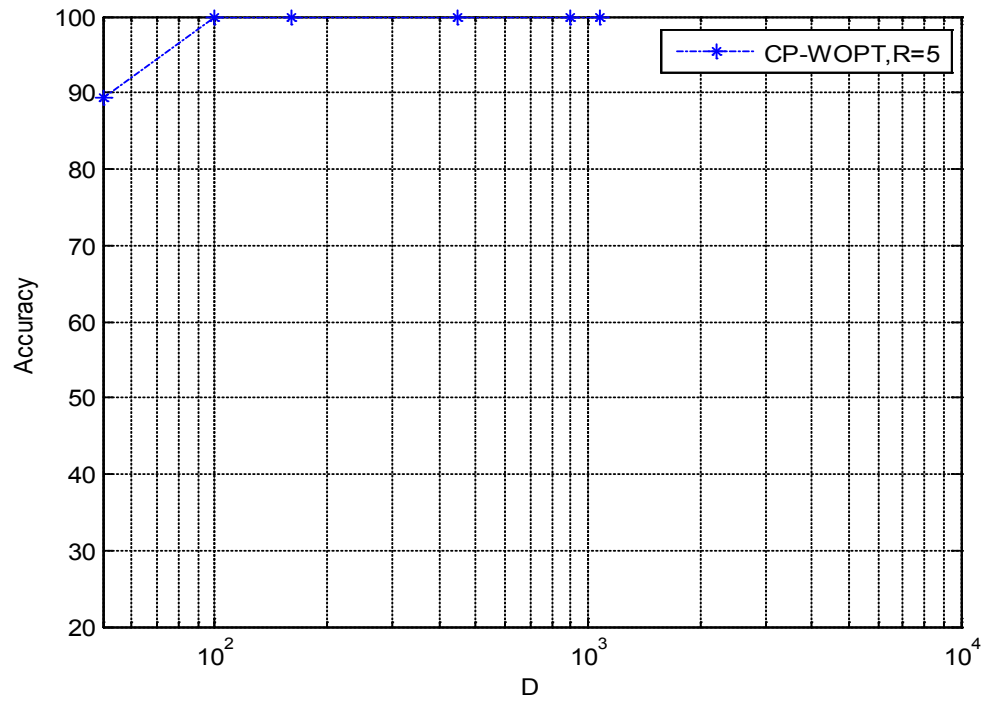


Figure 34. Accuracy versus  $D$ , i.e., the ratio of the number of known tensor entries to the number of variables, for CP-WOPT algorithm for tensors of size  $50 \times 50 \times 50$  and  $150 \times 150 \times 150$ . The plots are for the randomly missing fiber case in Table 3,4

---

#### 4.4.2. Capturing underlying factors of power grid data

If we cannot solve the problem of missing data, many important data sets will be discarded or improperly analyzed. But, this work focuses to the growing research that such data can be analyzed. As we mentioned in chapter 2, data of each record from a bus can be represented as a time-bus matrix; thus, data from multiple channels is three-dimensional (time, bus, and voltage or power) and forms a three-way array. These data can have multiple dimensions, are often massively large, and generally have at least some missing data. Therefore, we need a robust and reliable approach for factorizing multi-way arrays in the presence of missing data.

In previous chapter, we found that via tensor decomposition we can get a rough idea that there is some low-rank (tensor) structure in our data set, so we typically gained from using tensors as the data has a low-rank structure that we can exploited. Moreover, we illustrated a multi-dimensional periodicity in the data, which is good because it means that tensors can potentially help.

In this section, we aim to capture the underlying structure of the data through a higher-order tensor factorization instead of using interpolation method. Tensor factorization performe in the presence of missing data. The algorithm formulate missing values in the context of tensor factorizations.

We arrange the data using a CP model, denoting A, B and C as the extracted factor matrices corresponding to the time, selected buses, active power, respectively. It is common in power system that the data from buses are ignore due to malfunction of measurement instruments or quick change in the state of the system. This happening generate missing entries and fibers when the data arrange in CP model, as shown in Figure 35. Instead of using interpolation method, we apply CP Weighted Optimization algoritm to capture the underlying data structure. We select the same case study that we discussed in chapter 2, The data contains the 600 hours sample rates of active power for 100 buses out of 12418 channels (Figure 35).



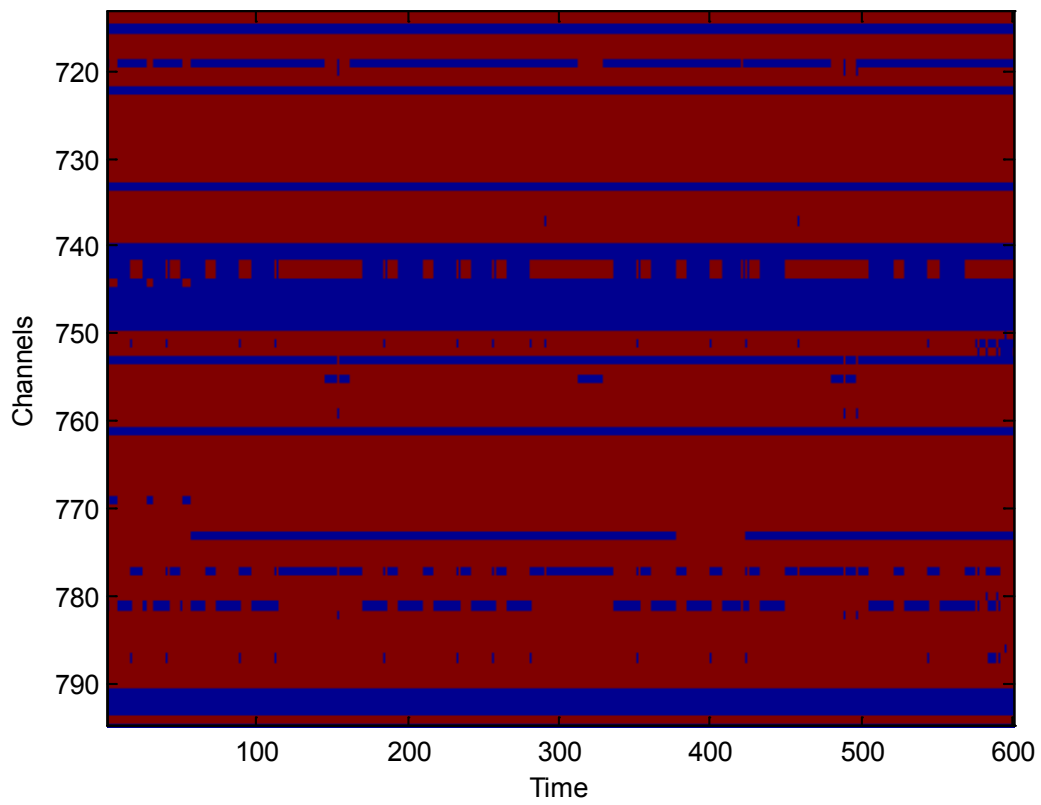


Figure 35. Interesting channels

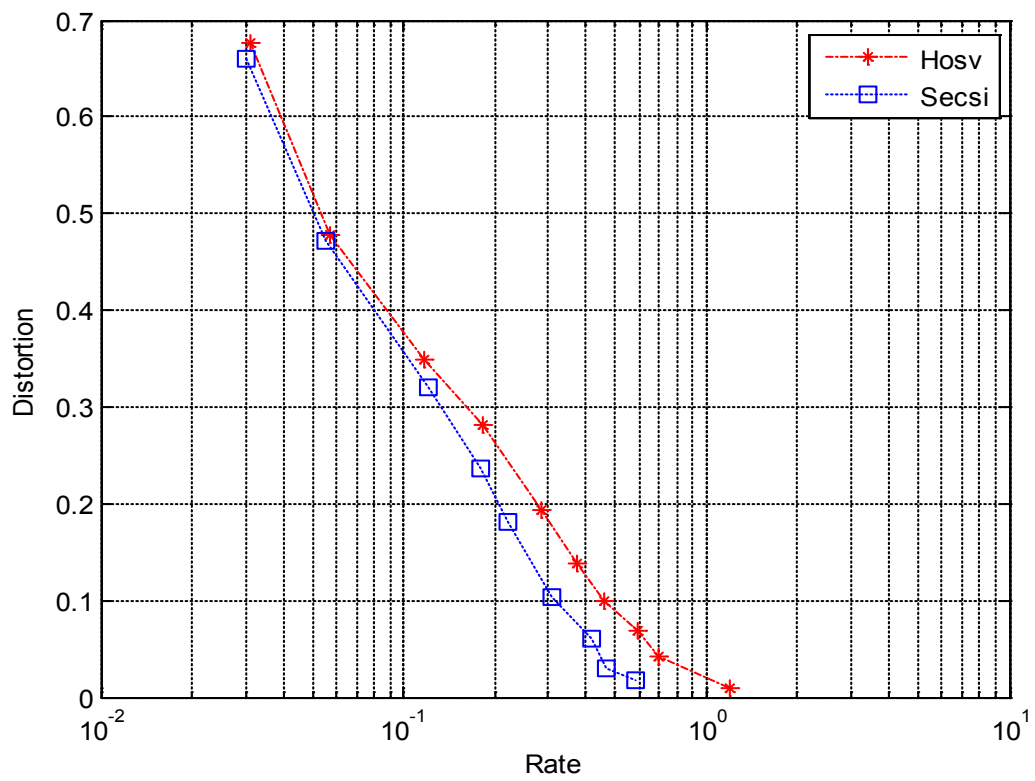


Figure 36. Reconstruction error

---

First we applied CP Weighted Optimization algorithm to find factor matrices and recover the missing values, then the benefits of tensor-based processing evaluated by computing an approximate CP decomposition and High Order Singular Value decomposition (HOSVD) which is performed successfully via compression graphs. The horizontal axis considered as a rate for the number of parameters at different ranks over the sampled data. As remembered, the vertical axis also nominated for the reconstruction error in CP decomposition and High Order Singular Value Decomposition. As it can be shown in Figure 36, unlike the interpolation method, rank-one components in CP Weighted Optimization method explained a significant fraction of data in both HOSVD and CP decomposition.

Now we performed the time signature of dominant (rank-one) component of the data, which is calculated by kronecker product of factor matrices obtained by CP decomposition-rank one. The CP decomposition calculated by Semi-algebraic framework for approximate CP decompositions via Simultaneous matrix diagonalization (SECSI) [10].

As mentioned before, the data set of power grid can be arranged as buses over time in a matrix of size  $600 \times 175$  for active powers. Therefore, the time axis can be broken into two dimensional space (24 hours x 21 days) or three dimensional space (24 hours x 7 days x 3 weeks) to generate respectively a 3-D tensor or 4-D tensor.

After careful consideration, we found that there is a multi-dimensional periodicity in 3D and also 4D data structure, the same as before (Figure 37, 38). As we can see the periodic behaviour doesn't change we respect to interpolation method. The reason of similarity, can be the number of missing data, which are not more than 15%, so the periodic behaviour is the same with respect to the interpolation method.

In conclusion, therefore, the CP Weighted Optimization algorithm can scale even to larger data sets and observe that it recovers the underlying tensors structure for a data set of size  $500 \times 500 \times 500$  properly [1]. So, it can also apply for power grid data sets contained more buses.

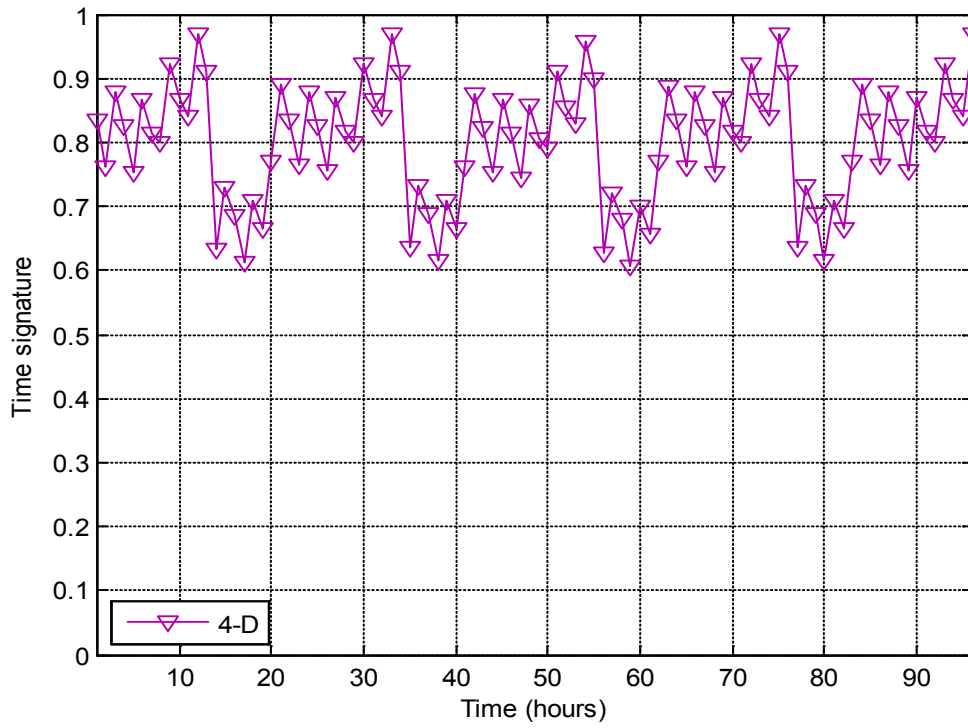


Figure 37. Time signature of 4-D tensor, CP decomposition rank one

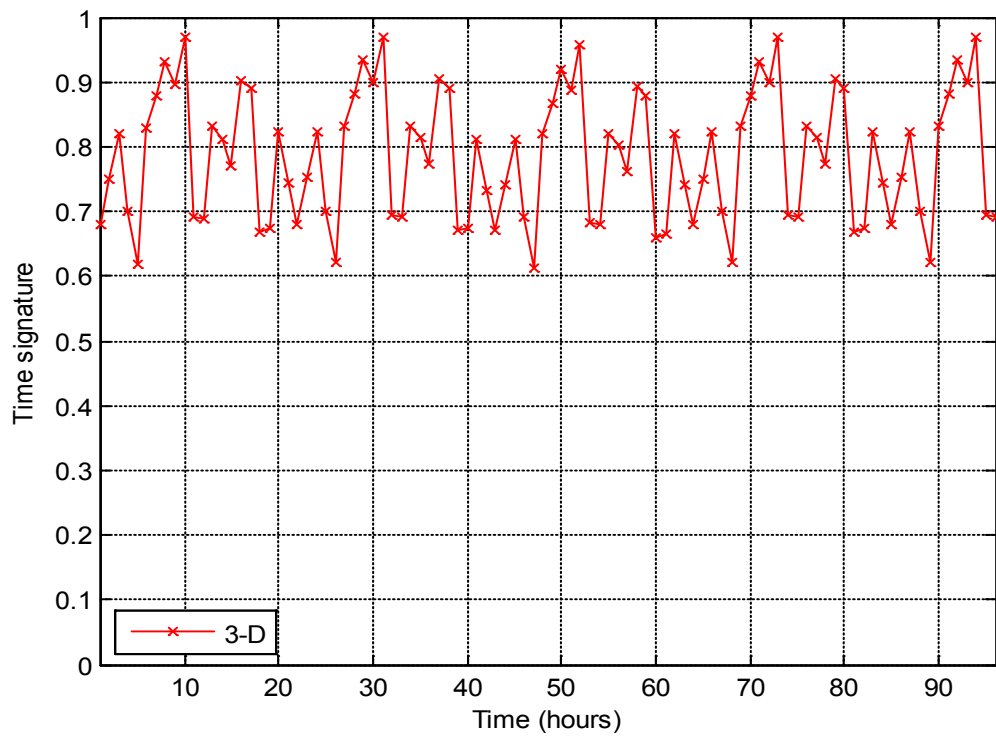


Figure 38. Time signature of 3-D tensor, CP decomposition rank one

---

## 5 CONCLUSION

In this work we focused to solve the problem of missing values. By considering previous studies which have only reflected on matrices, we focus here on the problem of electrical grid data in multi-way arrays (tensors). Data from multiple channels is three-dimensional (time, bus, and voltage or power) and forms a three-way array.

After recovering missing values with interpolation method, we observed that there is a low-rank structure in form of a quite dominant rank-one component (periodic) on top of the rest of the data (which seems to be not periodic). Then the evaluation of the benefits of using tensor-based processing by computing an approximate CP decomposition and High Order Singular Value decomposition (HOSVD) was performed successfully with compression graphs. We found that there is a multi-dimensional periodicity in 2D, 3D and also 4D data structure that's good because it means that tensors can potentially help.

In the last chapter, we aimed to capture the underlying structure of the data through a higher-order tensor factorization instead of interpolation method and performed the same procedure to find low rank modeling in our data structure. Tensor factorization performed in the presence of missing data. Formulating the canonical tensor decomposition for incomplete tensors as a weighted least squares problem. Our numerical studies proved that the proposed CP Weighted Optimization is so accurate and reliable in both simulated data and power grid data.

In future studies, we aim at extending these approaches through the power data structure with more than 3 dimensions and also with more than three variables. Also it can be done for the buses with much more missing values to test the accuracy of the algorithm in higher resolution. On the other hand, a simulated power system data can be more reliable to test the accuracy of the CP weighted Optimization algorithm, as we can remove randomly some tensor entries in order to generate missing values. In other words, a power grid with no missing values is

much more reliable to check how much the CP Weighted Optimization algorithm can successfully capture the original data structure.

Finally as the data structure gained from using tensor (CP) model, in future works we can use this data structure to solve some nonlinear problem in power system, such as weighted least squares method for bad data suppression in power system state estimation which is discussed in [12],[13],[14] and also use some nonlinear iterative methods, i.e. Newton Raphson to solve power flow.

In [15] the authorities suggested an approximate model for power system static state estimation in matrix notation which can be extended as a proper model for multi-way arrays.

---

## 6 REFERENCES

- [1] Acar, Evrim, et al. "Scalable tensor factorizations for incomplete data." *Chemometrics and Intelligent Laboratory Systems* 106.1 (2011): 41-56.
- [2] T. G. Kolda and B. W. Bader. "Tensor decompositions and applications". *SIAM Rev.*, 51(3):455–500, 2009.
- [3] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition", *SIAM J. Matrix Anal. Appl.*, 21 (2000), pp. 1253–1278.
- [4] B. W. Bader and T. G. Kolda. *Tensor toolbox for matlab*, version 2.2, last accessed November, 2008. <http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/>.
- [5] B. W. Bader and T. G. Kolda. "Efficient MATLAB computations with sparse and factored tensors". *SIAM J. Sci. Comput.*, 30(1):205–231, Dec. 2007.
- [6] A. M. Buchanan and A. W. Fitzgibbon. "Damped Newton algorithms for matrix factorization with missing data". In *CVPR'05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 316–322. IEEE Computer Society, 2005.
- [7] G. Tomasi and R. Bro. "PARAFAC and missing values". *Chemometr. Intell. Lab.*, 75(2):163–180, Feb. 2005
- [8] J. D. Carroll and J. J. Chang, "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition", *Psychometrika*, 35 (1970), pp. 283–319.
- [9] J. Nocedal and S. J. Wright. "Numerical Optimization". Springer, 1999.
- [10] Roemer, Florian, and Martin Haardt. "A semi-algebraic framework for approximate CP decompositions via simultaneous matrix diagonalizations (SECSI)" *Signal Processing* 93.9 (2013): 2722-2738.
- [11] F. Roemer and M. Haardt, "A closed-form solution for Parallel Factor (PARAFAC) analysis", in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008)*, pp. 2365-2368, Las Vegas, NV, USA, Apr. 2008.
- [12] Merrill, Hyde M., and Fred C. Schweppe. "Bad data suppression in power system static state estimation." *Power Apparatus and Systems, IEEE Transactions on* 6 (1971): 2718-2725.

- [13] Salberg, Arnt-Børre, Alfred Hanssen, and Louis L. Scharf. "Robust multidimensional matched subspace classifiers based on weighted least-squares." *Signal Processing, IEEE Transactions on* 55.3 (2007): 873-880.
- [14] Handschin, E., et al. "Bad data analysis for power system state estimation." *Power Apparatus and Systems, IEEE Transactions on* 94.2 (1975): 329-337.
- [15] Schweppe, Fred C., and Douglas B. Rom. "Power system static-state estimation, Part II: Approximate model." *power apparatus and systems, iee transactions on* 1 (1970): 125-130.

## 7 APPENDICES

### %%% Calculate Matrix Unfoldings

```

%%% |-----
% |
% |
% |   Danial Jafarigiv
% |   % computes all unfoldings of the given tensor T
% |
% |   out = unfoldings(T, order[, Ndim])
% |
% |   Inputs: T    - tensor
% |
% |   Output: out - matrix unfouldings of T
% |-----

```

```
function out = unfoldings (T, order, Ndim)
```

```
% get dimensions
```

```
sizes = size(T);
dimension = length(sizes);
```

```
if nargin > 2
```

```
    if dimension < Ndim
        sizes = [sizes,ones(Ndim-dimension)];
        dimension = Ndim;
    end
end
```

```
% Set standard Lathauwer unfolding
```

```
if nargin == 1
    order = 3;
end
```

```
% initialize out
```

```
out = cell(1, dimension);
```

```
% compute all unfoldings
```

```
for n = 1:dimension
```

```
    % permute tensor T for reshape - command
```

```
    switch order
```

```
        case 1
```

```
            P = permute(T, [n, 1:(n-1), (n+1):dimension]); % indices go faster with
            increasing index
        end
    end
end
```



---

```

    case 2
        P = permute(T, [n, fliplr( [1:(n-1), (n+1):dimension] )]); % indices go slower
with increasing index
    case 3
        P = permute(T, [n, fliplr( 1:(n-1) ), fliplr( (n+1):dimension )]); % Lathauwer:
indices go slower with I_n+1, ... I_N, I_1, ... I_n-1
    case 4
        P = permute(T, [n, fliplr( [ fliplr( 1:(n-1) ), fliplr( (n+1):dimension ) ])]); %
flipped Lathauwer
    otherwise
        disp('Error: unknown ordering for n--mode vectors');
        return
    end

    % compute n'th unfolding
    out{n} = reshape(P, [sizes(n), prod(sizes)./sizes(n)]);

end

```

### %%% Generate random tensor with rank, tensor decomposition

```

%%% |-----
% |
% |
% |   Danial Jafarigiv
% |   Generate random tensor with rank 5 to search low-rank modeling in our
% |   simulated data
% |-----

```

```

clear all;
clc

```

```

% generate random tensor with rank 5
size_tens = [10 10 10]; R = 5;
U = cpd_rnd(size_tens,R);
X = cpdgen(U);

```

```

X1 = double (X);

```

```

f = size(X1);
element = f(:,1)*f(:,2)*f(:,3);

```

### %% Add Noise

```

PS=1;
SNR = 10;
X1 = addnoise(X1,SNR,PS);

```

```

for d=1:10 %Diff Ranks

```

---

```

%HOSVD
[S, U_Cell, SD_Cell] = hosvd(X1);
[S_c, U_c] = CUTHOSVD(S,U_Cell,d); %Truncated HOSVD
T2 = nmode_product(S_c, U_c{1}, 1);
T2 = nmode_product(T2, U_c{2}, 2);
T2 = nmode_product(T2, U_c{3}, 3);

out_Hosv = ho_norm(X1-T2);
rec_err_H(d)=(out_Hosv.^2)./(ho_norm(X1).^2);

%Horiz
Hosv(d)=d^3+sum(f(:,1)*d+f(:,2)*d+f(:,3)*d);
l1(d)=Hosv(d)/element;

%% SECSI
Fhat = SECSI(X1,d);

%% ALS, multilinear alternating least squares ("plain vanilla")
method='MALS';
[Fac_est_ALS, Xrec_ALS, amplitudes_ALS] = solve_parafac(method,X1,d);

%% calculate the reconstructed tensors
Xrec_secsi = cp_construct(Fhat);

%% calculate forbenious norm;
out_secsi = ho_norm(X1-Xrec_secsi);
out_als = ho_norm(X1-Xrec_ALS);

rec_err_secsi_br(d)=(out_secsi.^2)./(ho_norm(X1).^2);
rec_err_als_br(d)=(out_als.^2)./(ho_norm(X1).^2);

%%Horz
se(d)= sum(f(:,1)*d+f(:,2)*d+f(:,3)*d);
l(d)=se(d)/element;
end

%%Plot
figure

semilogx(l1,rec_err_H,'-r*');
hold on
semilogx(l,rec_err_secsi_br,'--mo');
semilogx(l,rec_err_als_br,'bs');

grid on
legend('Hosv','Secsi','Als');

```

---

---

**%%% Calculate Tensor Decomposition to find rank one approximation**

```

%%% |-----
% |
% |
% |   Danial Jafarigiv
% |   HOSVD, SECSI, ALS methods for 3D and 4D structure
% |-----

```

```

load export_data_load11.mat
load dataset2

```

```

for d = 1:10 %Diff Ranks
%%data set 2
a = dataset2(1:504, :, 1);
c = reshape(a, 24, 21, 175);
dd = reshape(a, 24, 7, 3, 175);

```

```

X = c;
X2 = dd;

```

```

%% HOSVD 3D
X1 = double (X);
[S, U_Cell, SD_Cell] = hosvd(X1);
[S_c, U_c] = cuthosvd(S, U_Cell, d); %Truncated HOSVD
T22 = nmode_product(S_c, U_c{1}, 1);
T22 = nmode_product(T22, U_c{2}, 2);
T22 = nmode_product(T22, U_c{3}, 3);

```

```

%% HOSVD 4D
X3 = double (X2);
[S, U_Cell, SD_Cell] = hosvd(X3);
[S_c, U_c] = cuthosvd(S, U_Cell, d); %Truncated HOSVD
T2 = nmode_product(S_c, U_c{1}, 1);
T2 = nmode_product(T2, U_c{2}, 2);
T2 = nmode_product(T2, U_c{3}, 3);
T3 = nmode_product(T2, U_c{4}, 4);

```

```

%% SECSI 3D
Fhat = SECSI(X1, d);

```

```

%% ALS, multilinear alternating least squares ("plain vanilla") 3D
method='MALS';
[Fac_est_ALS, Xrec_ALS, amplitudes_ALS] = solve_parafac(method, X1, d);

```

```

%% calculate the reconstructed tensors

```

```

Xrec_secsi = cp_construct(Fhat);

```

---

---

```

%% calculate forbenious norm;
out_secsi = ho_norm(X1-Xrec_secsi);
out_als = ho_norm(X1-Xrec_ALS);
out_Hosv = ho_norm(X1-T22);

rec_err_secsi_br(d)=(out_secsi.^2)./(ho_norm(X1).^2);
rec_err_als_br(d)=(out_als.^2)./(ho_norm(X1).^2);
rec_err_H(d)=(out_Hosv.^2)./(ho_norm(X1).^2);

%Horiz
f=size(X1);
sec(d)= sum(f(:,1)*d+f(:,2)*d+f(:,3)*d);
element= f(:,1)*f(:,2)*f(:,3);
l(d)=sec(d)/element;
Hosv(d)=d^3+sum(f(:,1)*d+f(:,2)*d+f(:,3)*d);
l1(d)=Hosv(d)/element;

%% SECSI 4D
Fhat1 = SECSI(X3,d);

%% ALS, multilinear alternating least squares ("plain vanilla") 4D
% method='MALS';
% [Fac_est_ALS, Xrec_ALS, amplitudes_ALS] = solve_parafac(method,X3,d);

%% calculate the reconstructed tensors

Xrec_secsi1 = cp_construct(Fhat1);

%% calculate forbenious norm;
out_secsi1 = ho_norm(X3-Xrec_secsi1);
out_Hosv1 = ho_norm(X3-T3);
% out_als1 = ho_norm(X3-Xrec_ALS);

rec_err_secsi_br1(d)=(out_secsi1.^2)./(ho_norm(X3).^2);
rec_err_H1(d)=(out_Hosv1.^2)./(ho_norm(X3).^2);
% rec_err_als_br1(d)=(out_als1.^2)./(ho_norm(X3).^2);

%Horiz
f1=size(X3);
sec1(d)= sum(f1(:,1)*d+f1(:,2)*d+f1(:,3)*d+f1(:,4)*d);
element1 = f1(:,1)*f1(:,2)*f1(:,3)*f1(:,4);
l11(d)=sec1(d)/element1;
Hosv1(d)=d^3+sum(f1(:,1)*d+f1(:,2)*d+f1(:,3)*d+f1(:,4)*d);
l111(d)=Hosv1(d)/element1;

end

```

---

```
%% plots
figure

semilogx(l1,rec_err_H,'-r*');
hold on
semilogx(l,rec_err_als_br,':bs');
hold on
semilogx(l111,rec_err_H1,'-g*');
hold on

grid on
legend('Hosv3D','Als3D','Hosv4D');
```

---

```

%% % Test for SECSI and ALS
% We want to investigate the SECSI algorithm for different tensors of
% different sizes.
%%% |-----
% |
% |
% |   Danial Jafarigiv
% |   Apply SECSI algorithm and Alternating Least Squares method
% |   to decompose the tensors
% |-----

clear all;
clc;
for L=3:20
%% generate random tensor.
% L=3; % size1
% M=3; % size2
% N=3; % size3
d=3; %assumed rank
for SNR=20:1:50
PS=1;
% X= randn(L,M,N);
A=randn(L,d);
B=randn(L,d);
C=randn(L,d);
I=supereye(3,d);
X=nmode_product(I, A, 1);
X=nmode_product(X, B, 2);
X=nmode_product(X, C, 3);
for br=1:50
X = addnoise(X,SNR,PS);

%% SECSI , plain BD default for R=3;
Fhat = SECSI(X,d);

%% ALS, multilinear alternating least squares ("plain vanilla")
method='MALS';
[Fac_est_ALS, Xrec_ALS, amplitudes_ALS] = solve_parafac(method,X,d);

%% COMFAC (Sidiropoulos, Bro)
method='COMFAC';
[Fac_est_comfac, Xrec_comfac, amplitudes_comfac] =
solve_parafac(method,X,d);

%% calculate the reconstructed tensors

Xrec_secsi = cp_construct(Fhat);

%% calculate forbenious norm;

```

---

---

```

out_secsi = ho_norm(X-Xrec_secsi);
out_als = ho_norm(X-Xrec_ALS);
out_comfac = ho_norm(X-Xrec_comfac);

rec_err_secsi_br(br)=(out_secsi.^2)./(ho_norm(X).^2);
rec_err_als_br(br)=(out_als.^2)./(ho_norm(X).^2);
rec_err_comfac_br(br)=(out_comfac.^2)./(ho_norm(X).^2);
end

%% calculate normalized reconstruction error
rec_err_secsi(L-2,SNR+6)=mean(rec_err_secsi_br);
rec_err_als(L-2,SNR+6)=mean(rec_err_als_br);
rec_err_comfac(L-2,SNR+6)=mean(rec_err_comfac_br);
end
end

%% plots
figure
semilogy(3:20,rec_err_secsi(:,23),'r');
hold on
semilogy(3:20,rec_err_als(:,23),'b');
semilogy(3:20,rec_err_comfac(:,23),'k');
grid on
legend('SECSI','ALS','COMFAC');

%% Compare rank one 2D , 3D and 4D structure
%% |-----
% |
% |
% |   Danial Jafarigiv
% |   Using SECSI algorithm to calculate the rank one components of 3D and 4D
% |   structure
% |-----

show_3d = 1;
show_4d = 1;
load export_data_load11.mat
%% reshape 2D structure to 3D and 4D
a = dataset2(1:504, :, 1);
c = reshape(a,24,21,175);
d = reshape(a,24,7,3,175);

%% rank one decomposition
Fhat_3 = SECSI(c,1);
Fhat_4 = SECSI(d,1);
[U,S,V] = svd(unfolding(c,3));

```

---

---

```
%%%calculate time signature
```

```
timesig_2d = -V(:,1);
timesig_3d = kron(Fhat_3{1},Fhat_3{2});
timesig_4d = kron(kron(Fhat_4{1},Fhat_4{2}),Fhat_4{3});
```

```
%%% Plot time signature for 2D,3D and 4D structure
```

```
start = 8*24;
cnt = 4*24;
leg = {};
figure(1);
clf;
plot(1:cnt,timesig_2d(start:start+cnt-1)/max(abs(timesig_2d)),'ko-');leg{end+1} = '2-D';
hold on;
if show_3d
plot(1:cnt,-1*timesig_3d(start:start+cnt-1)/max(abs(timesig_3d)),'rx-');leg{end+1} = '3-D';
end
if show_4d
plot(1:cnt,-1*timesig_4d(start:start+cnt-1)/max(abs(timesig_4d)),'mv-','Color',[0.7,0,0.7]);leg{end+1} = '4-D';
end
hold on;
axis([1,cnt,0,1]);
xlabel('Time (hours)');
ylabel('Time signature');
legend(leg,3);
grid on;
```

```
%%% Plot the interpolated tensors
```

```
%%% |-----
% |
% |
% |   Danial Jafarigiv
% |
% |-----
```

```
load export_data_load11.mat
```

```
%%% bus_ts is a grid data which is formulated as a tensor structure
```

```
for b = 1:1000
```

```
    for a = 1:600
```



---

```

    z = abs(bus_ts(a,b,1));

    if z > 0
        g(a,b)=1;
    else
        g(a,b)=0;

    end
end
end

figure

imagesc(g');figure(gcf);
set(gca,'YLim',[1 1000])

hold on

%%% Interpolate the missing values of the tensors
%%% |-----
% |
% |
% |   Danial Jafarigiv
% |   bus_ts is a grid data which is formulated as a tensor structure,
% |   blue color define the missing values
% |-----

load export_data_load11.mat

zz17 = bus_ts(1:600,905:913,:);

l=1;
for b=1:175
    i = 1;

    for a=1:600

        z = abs(zz17(a,b,1));

        if z > 0

            k(l,i) = (zz17(a,b,1));

```

---

---

```
        i = i+1 ;  
  
    end  
  
    end  
    l = l+1;  
end  
  
for b= 1:175  
  
    for a = 1:600  
  
        z = abs(zz17(a,b,1));  
  
        if z > 0  
            g(b,a)= 1;  
  
        else  
  
            ZI = interp1(1:520,k(b,1:520),zz17(a,b,1),'linear');  
  
            zz17(a,b,1)= (ZI);  
  
            if abs (ZI) > 0  
                g(b,a) = 1;  
  
            else  
                g(b,a)= 0;  
  
            end  
  
        end  
  
    end  
  
end  
end
```

figure (1)

```
imagesc(g);figure(gcf);  
set(gca,'YLim',[1 175])
```

```
hold on
```

---

**%% Weighted CP tensor decomposition inorder to recover the missing data**

```

%% %% |-----
% |
% |
% |   Danial Jafarigiv
% |   Three way tensors generated with ranks 5 and 10, and different sizes
% |-----

```

```
load export_data_load11.mat
```

```

%% Create 50*50*50 problem with 10% missing data.
% Here we have 10% missing data and 10% noise.
R = 5;
info = create_problem('Size', [50 50 50], 'Num_Factors', R, ...
    'M', 0.10, 'Noise', 0.10);
X = info.Data;
P = info.Pattern;
M_true= info.Soln;

```

```

%% Create initial guess using 'nvecs'
M_init = create_guess('Data', X, 'Num_Factors', R, ...
    'Factor_Generator', 'nvecs');

```

```
%% Set up the optimization parameters
```

```

% Tighten the stop tolerance (norm of gradient)
ncg_opts.StopTol = 1.0e-8;
% Tighten relative change in function value tolearnce
ncg_opts.RelFuncTol = 1.0e-6;
% The number of iterations.
ncg_opts.MaxIters = 10^3;
% Only display every 10th iteration
ncg_opts.DisplayIters = 10;
% Display the final set of options
ncg_opts;

```

```

%% Call the |cp_wopt| method
% prints the least squares fit function value (being minimized) and the
% norm of the gradient. The meaning of any line search warnings

```

```

[M,~,output] = cp_wopt(X, P, R, 'init', M_init, ...
    'alg', 'ncg', 'alg_options', ncg_opts);

```

```

%% Check the output
exitflag1 = output.ExitFlag;

```

```
%% Evaluate the output
```

---

---

%The |score| function on ktensor's gives a score in [0,1] with 1 indicating a perfect match.

```
scr1 = score(M,M_true);
```

```
%% Create a 150*150*150 problem with 40% missing data.
```

```
% Here we have 40% missing data and 10% noise.
```

```
R = 10;
```

```
info = create_problem('Size', [150 150 150], 'Num_Factors', R, ...  
    'M', 0.40, 'Sparse_M', true, 'Noise', 0.10);
```

```
X = info.Data;
```

```
P = info.Pattern;
```

```
M_true= info.Soln;
```

```
%% Create initial guess using 'nvecs'
```

```
M_init = create_guess('Data', X, 'Num_Factors', R, ...  
    'Factor_Generator', 'nvecs');
```

```
%% Set up the optimization parameters
```

```
ncg_opts.StopTol = 1.0e-8;
```

```
ncg_opts.RelFuncTol = 1.0e-6;
```

```
ncg_opts.MaxIters = 10^3;
```

```
ncg_opts.DisplayIters = 10;
```

```
ncg_opts;
```

```
%% Call the |cp_wopt| method
```

```
[M,~,output] = cp_wopt(X, P, R, 'init', M_init, ...  
    'alg', 'ncg', 'alg_options', ncg_opts);
```

```
%% Check the output
```

```
exitflag2 = output.ExitFlag;
```

```
%% Evaluate the output
```

```
scr2 = score(M,M_true);
```