THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF ELECTRICAL ENGINEERING

MODELING AND SIMULATION OF RENEWABLE ENERGY SYSTEMS THROUGH TENSOR DECOMPOSITION

PIERCE J. MCKELVEY
SPRING 2022

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Electrical Engineering
with honors in Electrical Engineering

Reviewed and approved* by the following:

Yan Li
Assistant Professor of Electrical Engineering
Thesis Supervisor

Julio Urbina
Associate Professor of Electrical Engineering
Honors Adviser

*Electronic approvals are on file in the Schreyer Honors College.

Abstract

As the world's technology becomes more advanced, the need for efficient data analytics becomes an increasingly pressing issue. The advent of Distributed Energy Resources (e.g. Photovoltaics, Microturbines, Supercapacitors, Home battery walls, etc.) and their integration into the power grid introduces additional complexity to their control and operation problem.

Tensor decompositions, though invented in the early 20th century play a very important role in reducing this high dimensional data from modern systems into easily understandable low dimensional insights. These tools have been very popular in the chemmometric and signal processing communities, but there has been little research conducted with an application to power engineering.

In this work, we will introduce tensors and some common tensor decompositions. Then, we will introduce the new challenges raised by the transition of the power grid to a microgrid model. Finally, we will combine these concepts to show an application of tensor decomposition in the modelling of a microgrid.

Table of Contents

List of Figures

List of Tables

Acknowledgements

Thank you to Dr. Yan Li for helping me through this process and continuously keeping me thinking about the next problem to solve. Thank you to Dr. Julio Urbina for helping me with a myriad of questions throughout my tenor as a student here and during my time working on this project. I'd like to thank Penn State and Schreyer Honors College for allowing me the opportunity to learn about power systems more deeply through this work. Lastly I'd like to thank my family for supporting me always.

Chapter 1

Literature Review

1.1 What are tensors?

Tensors are the generalized form of an array. "An N- way or Nth-order tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system." [1] See Table 1.1 for a list of the lower order tensors and their more common names.

Tensor Dimension	Name
0-D Tensor	Scalar
1-D Tensor	Vector
2-D Tensor	Matrix
3-D Tensor	3-way Array
...	...
N-D Tensor	N-way Array

Table 1.1: Tensor dimensions to common name translation

Use visuals from [2] and [1] here to help explain.

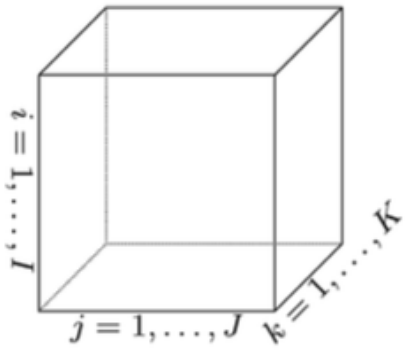

Figure 1.1: 3-D tensor [1]

Tensor decompositions are tools used to extract low-dimensional subspaces from high dimensional data. First introduced in 1927 by Hitchcock [3] the tools of tensor decomposition have been the subject of a flurry of research in fields ranging from applied mathematics to psychology to quantum molecular chemistry [1] [4].

The Tucker decomposition, the main focus of this work, is an extension of the Singular Value Decomposition. A brief explanation of the Singular Value Decomposition and Principle Component Analysis follows.

The Singular Value Decomposition (SVD) is one of the most commonly used and useful tools in data processing. It is used for data reduction and is often a first step in processing. For an introductory video series to SVD and Principle Component Analysis with code examples see [5].

These are extensions of tools used in introductory Linear Algebra classes. Essentially, they find the dominant modes of a data set (matrix) to help with modeling. Use `[U,S,V] = svd(X)`

command in MATLAB to compute the SVD. Where,

$$X = U * 8 * V^T. \tag{1.1}$$

Some useful properties of the SVD:
U, V are unitary,

$$U * U^T = U^T * U = I \tag{1.2}$$

$$V * V^T = V^T * V = I \tag{1.3}$$

Where I is the identity matrix of the same size as U or V.

U: columns of U have the same shape as columns of X. The columns are hierarchically aranged eigen-representations of the data you are representing. For example, if we have a data matrix X where each column represents a face as in [5]. The U matrix would have a ranked series of eigenfaces. Where column 1 of U would describe the most variance in X, column 2 would describe less, etc. The U matrix is known as the left-singular vectors.

S: Diagonal, non-negative, hierarchical descending order. Tells us the weight of each eigen-column in U. Know as the singular values.

$\mathbf{V}^T$: The columns of V are in the rowspace of X. In the face example from [5] the row of V^T represent the amounnts of each column of U required to make up the original face X_n where X_n is a column n of X. This matrix is known as the right singular vectors.

The values of the S matrix tell us how important each column of U is in representing our data. If the later columns of S approach zero we can represent most of the variation of X without including them in the calculation. This is where the data compression element of the SVD comes in.
SVD allows us to generalize

$$Ax = b \tag{1.4}$$

for non-square A.
This generalization allows us to compute the pseudo-inverse of matrices when they are non-invertible, e.g. in the case of over-determined systems. This pseudo-inverse is called the Moore-Penrose pseudo-inverse, represented by A^+ where A is our original matrix.

$$A^+ = V * \Sigma^{-1} * V^T \tag{1.5}$$

Principle Component Analysis (PCA): Similar to SVD but for statistical data sets. Can be used to visualize high dimensional data into human view-able 2D or 3D. Introduced in 1901, but still widely used. Please see [6] for a presentation explaining PCA.

While both of these methods are great in the ways they have been used. They don't work as the dimensions of the data passed in start to increase. One drawback of SVD is he non-uniqueness of the factor matrices. Each can be subject to rotations and still represent the original data. These drawbacks form the motivation for using the tensor decomposition.

1.2 Why use Tensor Decompostions?

Using tensors over matrices allows one to apply the tools to more complex data with a large number of dimensions.

Most modern data sources involve large amounts of data and a high number of dimensions. Some examples of this high order systems include social media data, financial data, and network traffic data (cited examples in [4]). Using decomposition tools allows for a speeding up of data analysis. Additional advantages include robustness to noise and uniqueness [2] [7].

1.3 Typical Applications

Tensor decomposition techniques have been applied to problems in Signal Processing [8], Psychology [9], Computer Vision, Physics, and Chemmometrics [7]. The techniques are used to compress data to make the processing faster. The goal of this paper is to apply these techniques to simulation and modelling of power electronics, specifically microgrids.

1.4 Selecting a Decomposition Method

CANDECOMP/PARAFAC (CP):

The CP method took three introductions in literature to become popular. The first by Hitchcock in 1927, then by Cattell in 1944, it gained steam after its introduction to the psycometrics community in 1970 by Carroll and Chang [1]. The abbreviated name CP was coined by Kiers in 2000 [10]. The purpose of the CP decomposition is to factorize a tensor into a finite sum of component rank-one tensors. A dive into the mathematics behind CP can be found in [1]. An advantage of CP is the few parameters it requires as compared to other decompositions.

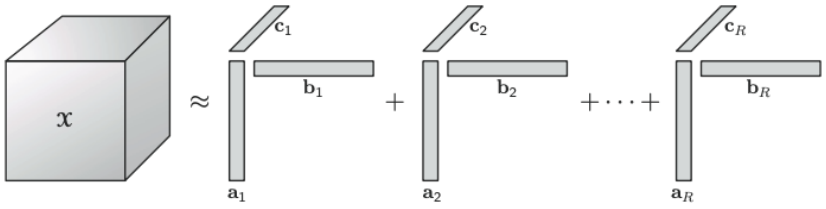

Figure 1.2: CANDECOMP visualization in the three-way case [1]

Some drawbacks of using CP include it being an NP hard problem that it doesn't always converge to the global minimum [4]. The NP hardness of the problem discourages the use of this decomposition as the dimensions get large. With the increasing complexity of modern data sets the dimensionality in turn also increases. This makes CP a less appealing option in modern data applications [11].

TUCKER:

This method was introduced in 1963 by L.R. Tucker [12]. A further refinement by Tucker in 1966 is noted by Kolda as the most comprehensive of early literature [13]. The Tucker decomposition decomposes a tensor into a core tensor G as well as factor matrices, one for each mode.

$$X \approx G \times_1 U^{(1)T} \times_2 U^{(2)T} ... \times_d U^{(d)T} \tag{1.6}$$

A compact version of equation 1.6 for the 3-way case introduced by Kolda is,

$$X \approx [[\mathbf{G}; \mathbf{A}, \mathbf{B}, \mathbf{C}]] \tag{1.7}$$

In Equation 1.6, The U tensors are factor matrices. In Figure 1.3 there are 3 factor matrices (A, B, C) each corresponding to a particular mode in the original tensor X.

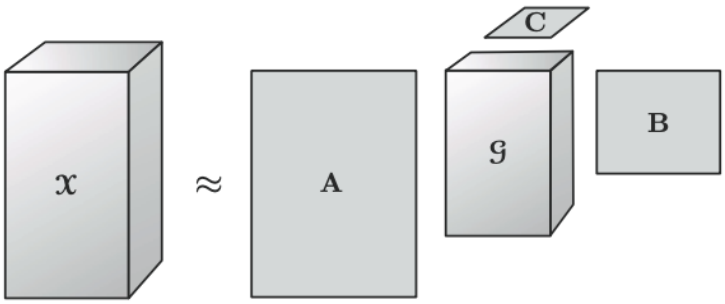

Figure 1.3: Tucker decomposition visualization in the three-way case [1]

The computational complexity of the Tucker decomposition of a $C^{nx...xn}$ tensor is given by

$$O(dnr + r^d) \tag{1.8}$$

Where d is the number of modes in the tensor [4]. Note that the complexity grows exponentially with d so Tucker is not appropriate with high dimensional data.

Throughout academic literature the Tucker decomposition is known by a few names (N-mode PCA, Higher-order SVD, N-Mode SVD, etc.) which were summarized by Kolda in [1]. This is due to a few researchers discovering the method in different fields of study.

In [2] one can see experimental results comparing the performance of a few different tensor decompositions (Tucker, CANDECOMP/PARAFAC, Hierarchical Tucker , Tensor-Train) on 3 data sets. While CP provides the optimal compression it does not always converge. HOOI (Tucker) provides the best compression vs. error rate for the data sets with less than 4 dimensions [2]. The Hierarchical (HT) and Tensor Train (TT) decomposition methods work increasingly well as the dimensions start to increase [2].

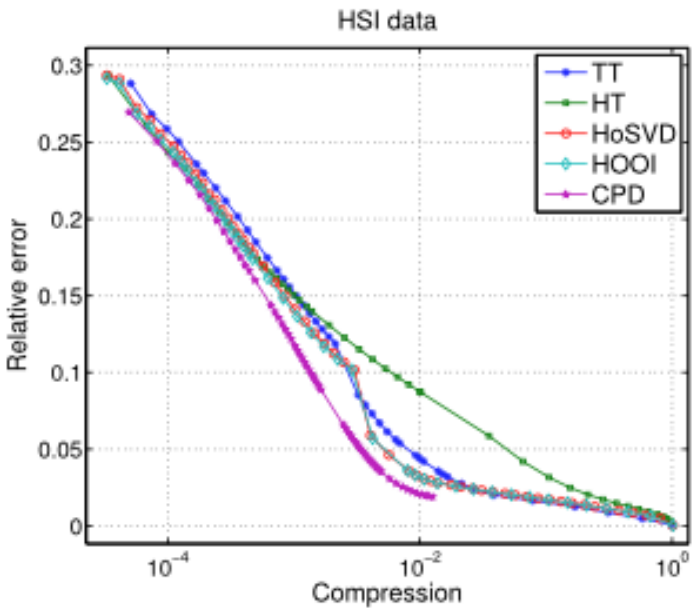

Figure 1.4: Zare experimental decomposition results [2]

An explanation as to why HT and TT decompositions perform better in higher dimensions can be found in the work of Grasedyck [14] and Oseledets [15].

1.5 Computational Tools

In [1], Kolda mentions two toolboxes for tensor computation in MATLAB. The MATLAB Tensor Toolbox by Bader and Kolda [16] as well as the N-way Toolbox for MATLAB developed by Andersson and Bro [17].

Zare mentions the TensorLab package [18], the TP tool [19], the Tensor-Train Toolbox [20], and the Hierarchical Tucker Toolbox [21].

The Tensor toolbox for MATLAB provides ALS-based algorithms for CP and Tucker decompositions. Kolda mentions the specific support of this toolbox for sparse tensors. In the use case of Distrubuted Energy Resources (DER) computation, the data presents itself in a sparse tensor format. For this reason as well as the ease of use, the MATLAB Tensor Toolbox was selected as the primary computational tool for this work. The ease of use is aided by the automatic determination of parameters. The manual selection and optimization processes of these parameters is detailed in [9]. This process is done automatically in the Tensor Toolbox for MATLAB developed by Kolda and Bader [16].

Chapter 2

Power Systems

2.1 Intro to Power Systems + Traditional Grid Overview

The purpose of this work is to apply the tools of tensor mathematics to problems in power pngineering. First we must learn about what problems we can solve in power engineering.

The traditional power grid is composed of three main parts, the Generation, Transmission and Distribution systems. Power is generated in the generation system at power plants 15kV - 25kV. Power is transmitted long distances across high voltage lines in the transmission system 69kV - 500kV. The distribution system includes a local substation where voltage is stepped down to 12kV before being routed to customers. This grid follows a unidirectional power flow model. With power flowing from generation to transmission to distribution in that order exclusively. This is known as the Centralized Unidirectional System [22].

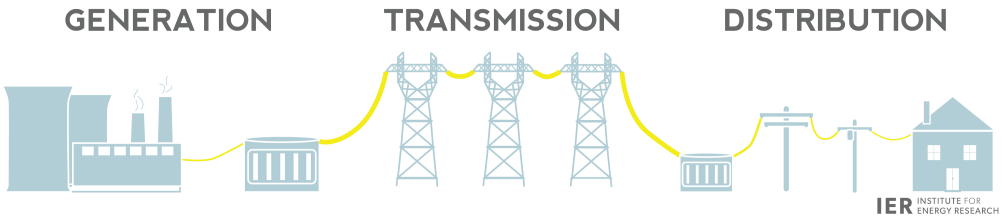

Figure 2.1: Three Stages of the Traditional Power Grid [23]

The increasing popularity and adoption of renewable energy sources introduces new operating challenges to the grid and power companies that operate them. One example of a newfound problem is the duck curve of demand due to photovoltaics (solar panels) inability to provide power once the sun goes down. This creates a very large change in power demanded as soon as the sun goes down and it also presents the risk of over-generation during the day.

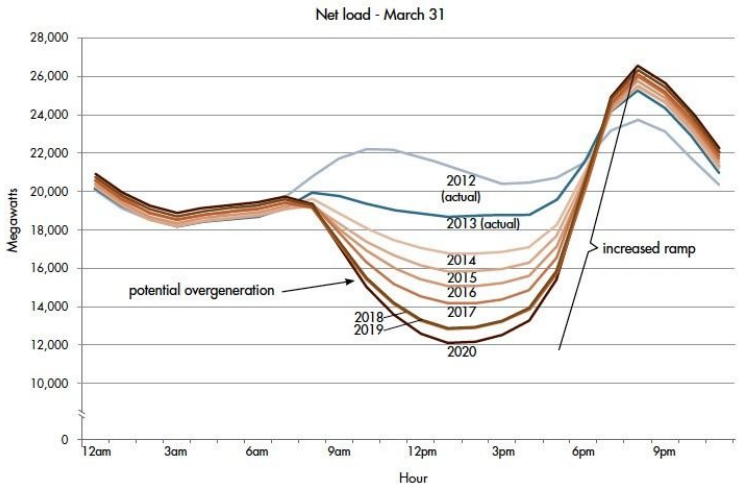

Figure 2.2: The Duck Curve from a 2013 published study by the California Independent System Operator [24]

This duck curve can be compensated for with by coupling photovoltaics with energy storage

devices such as flywheels or large capacity battery storage [24]. This would reduce the demand on power companies allowing them to operate in a more consistent fashion.

2.2 Microgrids + DERs

The term microgrid refers to a small electrical grid with the ability to connect and disconnect from the traditional power grid.

Distributed Energy Resources or DERs are items used to improve power supply reliability and reduce a local energy grid's dependence on the main power grid. Examples of common distributed energy resources include, Photovoltaics, Microturbines, Fuel Cells, Battery Storage, Super Capacitors, and Flywheels. The introduction of local power sources also reduces power waste due to long-range transmission losses.

Type	Voltage Output Type	Storage/Generation
Photovoltaic	DC	Generation
Microturbines	AC	Generation
Flywheel	AC	Storage
Supercapacitor	DC	Storage
Fuel Cell	DC	Storage

Table 2.1: A quick overview of the different sources of power in a microgrid.

Microgrids can also help to get power back to areas quicker in times of blackouts as local power storage devices can be used to restart local systems or keep essential functions working.

2.3 The Admittance Matrix

The admittance matrix is a matrix used to represent the admittances present at each node of a power system. The admittance matrix will be a sparse, square matrix and diagonally symmetric.

Go through basic admittance matrix example

Admittance matrix construction:

$$Y = \begin{bmatrix} Y_{11} & Y_{12} & ... & Y_{1n} \\ Y_{21} & Y_{22} & ... & Y_{2n} \\ ... & ... & ... & ... \\ Y_{n1} & Y_{n2} & ... & Y_{nn} \end{bmatrix} \tag{2.1}$$

The main diagonal values from Y_{11} to Y_{nn} are known as self-admittances.

$$Y_{ij} = \begin{pmatrix} y_i + \sum_{k=1;i\neq k}^{N} y_{ik}, & if\ i = j \\ -y_{ij}, & if\ i \neq j \end{pmatrix} \tag{2.2}$$

$$Y_{bus}V_{bus} - I_{bus} = 0 \tag{2.3}$$

Often in real world problems the Ibus matrix includes some of the unknowns in the Vbus matrix resulting in an equation with no closed form solution. We can use methods such as the Newton-Raphson method to solve these equations for each unknown value in V_{bus}.

Additionally, in real world applications the size of these admittance matrix is very large. These can be used to model each different PJM interconnection zone (multi-state) or the entire United States Power grid. As one can imagine these matrices could easily reach multiple hundred thousand in a conservative estimate.

Chapter 3

My Work

3.1 The problem

In order to demonstrate the power of tensor decompositions we show this example. In a Microgrid with 8 nodes we want to know the net power flow. This grid consists of 1 generator object (represented by the windmill, node: 8) and 5 load objects (nodes: 1,4,5,6,7).

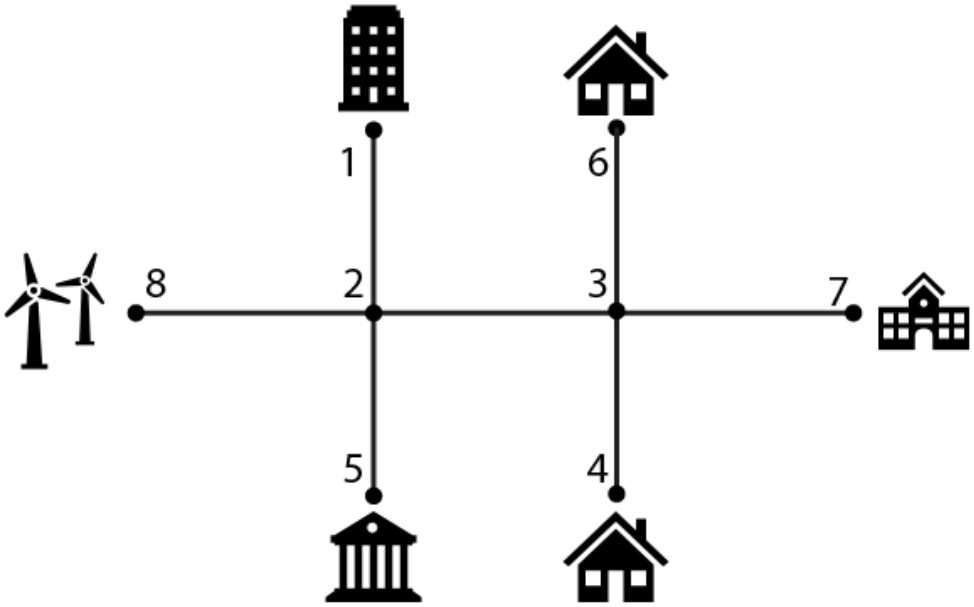

Figure 3.1: Microgrid Example Toplogy with labelled nodes

While a 2-dimensional system is demonstrated here the true benefit of using tensor decompositions will be realized once this technique is applied to data of higher dimensions. Some additional dimensions could be time, power from generator 1 / generator 2, etc.

3.2 Results

In this section we detail experimental results using the Tucker tensor decomposition with the Tensor Toolbox MATLAB toolbox [16].

$$Y_{bus}V_{bus} - I_{bus} = 0 \tag{3.1}$$

Now as in real world applications admittances, currents, and voltages are complex-valued. This extend the real valued 8x8 admittance matrix for an 8 node system to a 16x16 node system. There are two nodes for each physical node, one for the real and the other for the imaginary component.

The Y_{bus} used in this problem was built in MATLAB using the following command.

```
>>Ybus_Hermitian = [imag(Ybus), real(Ybus);real(Ybus),-imag(Ybus)];
```

Below is our given data of Y_{bus} (16 x 16) and I_{bus} (16 x 1).

$$Y_{bus} = \begin{bmatrix}
-0.07332 & 0.0628 & 0 & 0 & 0 & 0 & 0 & 0 & 1.42 & -1.323 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.0628 & -2.759 & 2.167 & 0 & 0.3007 & 0 & 0 & 0.2288 & -1.323 & 14.63 & -7.415 & 0 & -2.038 & 0 & 0 & -3.851 \\
0 & 2.167 & -3.156 & 0.5646 & 0 & 0.332 & 0.09284 & 0 & 0 & -7.415 & 16.53 & -6.071 & 0 & -1.914 & -1.131 & 0 \\
0 & 0 & 0.5646 & -0.5966 & 0 & 0 & 0 & 0 & 0 & 0 & -6.071 & 6.122 & 0 & 0 & 0 & 0 \\
0 & 0.3007 & 0 & 0 & -0.348 & 0 & 0 & 0 & 0 & -2.038 & 0 & 0 & 2.114 & 0 & 0 & 0 \\
0 & 0 & 0.332 & 0 & 0 & -0.3633 & 0 & 0 & 0 & 0 & -1.914 & 0 & 0 & 1.964 & 0 & 0 \\
0 & 0 & 0.09284 & 0 & 0 & 0 & -0.1026 & 0 & 0 & 0 & -1.131 & 0 & 0 & 0 & 1.147 & 0 \\
0 & 0.2288 & 0 & 0 & 0 & 0 & 0 & -5.229 & 0 & -3.851 & 0 & 0 & 0 & 0 & 0 & 3.851 \\
1.42 & -1.323 & 0 & 0 & 0 & 0 & 0 & 0 & 0.07332 & -0.0628 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1.323 & 14.63 & -7.415 & 0 & -2.038 & 0 & 0 & -3.851 & -0.0628 & 2.759 & -2.167 & 0 & -0.3007 & 0 & 0 & -0.2288 \\
0 & -7.415 & 16.53 & -6.071 & 0 & -1.914 & -1.131 & 0 & 0 & -2.167 & 3.156 & -0.5646 & 0 & -0.332 & -0.09284 & 0 \\
0 & 0 & -6.071 & 6.122 & 0 & 0 & 0 & 0 & 0 & 0 & -0.5646 & 0.5966 & 0 & 0 & 0 & 0 \\
0 & -2.038 & 0 & 0 & 2.114 & 0 & 0 & 0 & 0 & -0.3007 & 0 & 0 & 0.348 & 0 & 0 & 0 \\
0 & 0 & -1.914 & 0 & 0 & 1.964 & 0 & 0 & 0 & 0 & -0.332 & 0 & 0 & 0.3633 & 0 & 0 \\
0 & 0 & -1.131 & 0 & 0 & 0 & 1.147 & 0 & 0 & 0 & -0.09284 & 0 & 0 & 0 & 0.1026 & 0 \\
0 & -3.851 & 0 & 0 & 0 & 0 & 0 & 3.851 & 0 & -0.2288 & 0 & 0 & 0 & 0 & 0 & 5.229
\end{bmatrix} \tag{3.2}$$

$$I_{bus} = \begin{bmatrix}
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
-5.1077 \\
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
0.0000 \\
0.2626
\end{bmatrix} \tag{3.3}$$

Step 1 of this process is to compress the $Ybus$ tensor.

Tucker definition

The G tensor, aka the core from our Tucker decomposition definition above can be viewed as compressed version of our data tensor Y_{bus} as per [25]. Compression ratio comparison for different tolerances can be found in the appendix.

Step 2, we must then invert this compressed Ybus in order to solve.

$$V_{bus} = (Y_{bus})^{-1} I_{bus} \tag{3.4}$$

Since $Y_{buscompressed}$ is a non-square matrix we use the Moore-Penrose pseudo-inverse in this step. A mathematical description of this operation is described in Section 1.1.

The $Y^{-1}_{buscompressed}$ has dimensions (8 x 10).

$$Y^{-1}_{buscompressed} = \begin{bmatrix} 0.03908 & 0.006147 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.006147 & 0.03908 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -0.03688 & 0.07217 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.07217 & 0.03688 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.08298 & 0.1529 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.1529 & 0.08298 & 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -0.1035 & 0.2894 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.2894 & 0.1035 & 0.0000 & 0.0000 \end{bmatrix} \tag{3.5}$$

Step 3, we must compress the I_{bus} tensor so that the dimensions of $Ybus^{-1}$ and I_{bus} are the compatible values to be multiplied. We compress the I_{bus} tensor to a (10 x 1) tensor. Here this compression may not be possible. One method of overcoming this problem is to change the tolerance of the `hosvd()` command. This would result in a larger sized core G. That new core size may be compatible with a compressed I_{bus}.

$$I_{bus_compressed} = \begin{bmatrix} 5.114 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix} \tag{3.6}$$

In Step 4, we multiply these two compressed tensors to retrieve a compressed version of V_{bus}.

$$V_{buscompressed} = \begin{bmatrix} 0.1999 \\ -0.03144 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix} \tag{3.7}$$

Now, we may be able to use the factor matrices from both decompositions to decompress our matrix. In this work, I was not able to figure out how to decompress this tensor correctly. I outline some potential math to do so below.

$$Y_{bus}V_{bus} - I_{bus} = 0 \tag{3.8}$$

In the 2D Ybus case we can compress Y_{bus} and I_{bus}

$$Y_{bus} \approx G_Y \times_1 U_Y^{(1)T} \times_2 U_Y^{(2)T} \tag{3.9}$$

$$I_{bus} \approx G_I \times_1 U_I^{(1)T} \times_2 U_I^{(2)T} \tag{3.10}$$

$$(G_Y \times_1 U_Y^{(1)T} \times_2 U_Y^{(2)T}) V_{bus} = G_I \times_1 U_I^{(1)T} \times_2 U_I^{(2)T} \tag{3.11}$$

Using properties of the inverse we might be able to find a relationship that translates the compressed product $G_Y^{-1} * G_I$ into the decompressed version.

$$V_{bus} = (G_Y \times_1 U_Y^{(1)T} \times_2 U_Y^{(2)T})^{-1} (G_I \times_1 U_I^{(1)T} \times_2 U_I^{(2)T}) \tag{3.12}$$

At this stage I believe using properties of inverses there may be a factor of U_Y's and U_I's that will translate $V_{buscompressed}$ into V_{bus}.
Code for this example can be found in section A.1 of the appendix.

3.3 Future Work

The first step would be figuring out how to decompress our result so it can be used to finish the power flow calculation.

Following this work one could expand it to higher dimensional problems. With the computation complexity of the Tuck decomposition scaling exponentially one could look into using the TT or HT decompositions mentioned in chapter 1.

While this work was done using standard tensor objects with the Tensor toolbox for MATLAB. Using sparse tensor objects could serve to increase computational efficiency once the data sets become large. Documentation on sparse tensor objects can be found on the tensor toolbox website found here: [16]

Appendix

A.1: Microgrid Decomposition Example

```
%% Pierce McKelvey
% Tucker decomposition example for undergrad thesis
% Spring 2022

% clear the workspace
clear

% import data
load('Ybus_Hermitian.mat','Ybus_Hermitian');
load('Ibus.mat', 'f2_realnumber');


%% Traditional Method (Linear Algebra)

Vbus_trad = inv(Ybus_Hermitian) * f2_realnumber


%% Tensor compression method

% Convert data into tensor format
Ibus = tensor(f2_realnumber);
Ybus = tensor(Ybus_Hermitian);


% Perform HOSVD on Ybus matrix
T = hosvd(Ybus, 0.1);


% Check the size of the core of Ybus compressed
coresize = size(T.core);

% Check error
relativeError = norm(Ybus-full(T))/norm(Ybus);


% TAMU CODE
% Factor matrices (U,V,W and core tensor)
core = T.core;


% if compressed matrix is nonsquare use moore penrose pseudo inverse
coreInv = pinv(double(core));

% find 2nd dimension size so we know what size to compress Ibus to
% Vbus = Ybus^-1 * Ibus compressed dims must agree for ...
   multiplication to
% work

coreInvdims = size(coreInv);

```

```
50
51
52
53      % Try and shape Ibus matrix to be the same size as the compressed Ybus
54
55      Ibus_compressed = hosvd(Ibus, 0.1, 'rank', [coreInvdims(2) 1]);
56
57      % Check the size of core
58      coresize = size(Ibus_compressed.core);
59
60
61      % Compute the compressed Vbus result
62      Vbus = coreInv *  double(Ibus_compressed.core)
```

A.2: Tolerance's Effect on Compression Ratio Code

[26]
Compression rate for T1 with tolerance = 0.1 is 93.46%
Compression rate for T2 with tolerance = 0.2 is 98.86%

```
1       %% Code by Keyla Gonzalez, edited slightly by Pierce McKelvey
2
3       % Example from tensortoolbox.org on how to use tucker decomp
4       % ttps://www.tensortoolbox.org/hosvd_doc.html
5
6
7       % Seismic example from Keyla Gonzalez
8       % https://github.com/keylagonzalezabad/HOSVD-Microseismic-Data
9       %/blob/master/3D%20HOSVD/Matlab_HOSVD/HOSVD.m
10      % Gives compression data
11
12      clear
13
14      % Create 50x40x30 rand tensor with an optimal compression size of ...
            [5,4,3]
15      info = create_problem('Type', 'Tucker', 'Num_Factors', [5,4,3], ...
            'Size', [50, 40, 30]);
16      X = info.Data;
17
18      % Compute HOSVD with 0.1 tolerance
19      % increasing tolerance decreases core size, increases %compression, ...
            and increases error
20      T = hosvd(X, 0.1);
21
22      % Compute HOSVD with 0.2 tolerance
23      T2 = hosvd(X, 0.2);
24
25      % Check the sise of core
26      coresize = size(T.core)
27
```

```
28      % Check error
29      relativeError = norm(X-full(T))/norm(X)
30
31
32      % TAMU CODE
33      % Factor matrices (U,V,W and core tensor)
34      core = T.core;
35      U = T.U{1};
36      V = T.U{2};
37      W = T.U{3};
38
39
40      % Compression Ratio Print results
41      s_core = size(core);
42      s_x = size(X);
43
44      fprintf('Decomp with error = 0.1')
45      wh_X = whos('bytes', 'X');
46      siz_X = wh_X.bytes*10e-6;
47      wh_T=whos('bytes','T');
48      siz_T=wh_T.bytes*10e-06;
49      CR=(1-(wh_T.bytes/wh_X.bytes))*100
50
51
52
53      % T2 Results
54      fprintf('Decomp with error = 0.2')
55      wh_X = whos('bytes', 'X');
56      siz_X = wh_X.bytes*10e-6;
57      wh_T=whos('bytes','T2');
58      siz_T=wh_T.bytes*10e-06;
59      CR=(1-(wh_T.bytes/wh_X.bytes))*100
```

Bibliography

[1] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[2] Ali Zare, Alp Ozdemir, Mark A Iwen, and Selin Aviyente. Extension of pca to higher order data structures: An introduction to tensors, tensor decompositions, and tensor pca. *Proceedings of the IEEE*, 106(8):1341–1358, 2018.

[3] Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[4] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[5] Steve Brunton. Singular value decomposition (svd): Overview. `https://www.youtube.com/watch?v=gXbThCXjZFM.`, Jan 2020.

[6] Praneeth Netrapalli. An introduction to pca. 2015.

[7] Frederic Estienne, Nele Matthijs, DL Massart, Ph Ricoux, and D Leibovici. Multi-way modelling of high-dimensionality electroencephalographic data. *Chemometrics and Intelligent Laboratory Systems*, 58(1):59–72, 2001.

[8] Gérard Favier, C Alexandre R Fernandes, and André LF de Almeida. Nested tucker tensor decomposition with application to mimo relay systems using tensor space–time coding (tstc). *Signal Processing*, 128:318–331, 2016.

[9] Eva Ceulemans and Henk AL Kiers. Selecting among three-mode principal component models of different types and complexities: A numerical convex hull based method. *British journal of mathematical and statistical psychology*, 59(1):133–150, 2006.

[10] Henk AL Kiers. Towards a standardized notation and terminology in multiway analysis. *Journal of Chemometrics: A Journal of the Chemometrics Society*, 14(3):105–122, 2000.

[11] Johan Håstad. Tensor rank is np-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[12] Ledyard R Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in measuring change*, 15(122-137):3, 1963.

[13] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[14] Lars Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM journal on matrix analysis and applications*, 31(4):2029–2054, 2010.

[15] Ivan V Oseledets and Eugene E Tyrtyshnikov. Algebraic wavelet transform via quantics tensor train decomposition. *SIAM Journal on Scientific Computing*, 33(3):1315–1328, 2011.

[16] Brett W. Bader, Tamara G. Kolda, et al. Tensor toolbox for matlab, version 3.2.1. `https://www.tensortoolbox.org/`.

[17] Claus A Andersson and Rasmus Bro. The n-way toolbox for matlab. *Chemometrics and intelligent laboratory systems*, 52(1):1–4, 2000.

[18] Tensorlab: A matlab toolbox for tensor computations. `https://www.tensorlab.net`.

[19] Tp tool. `https://www.mathworks.com/matlabcentral/fileexchange/25514-tp-tool`.

[20] Tt-toolbox. `https://github.com/oseledets/TT-Toolbox`.

[21] Hierarchical tucker toolbox. `https://anchp.epfl.ch/htucker`.

[22] Yan Li. *Cyber-Physical Microgrids*. Springer, 2021.

[23] Institute for Energy Research. Electricity transmission. `https://www.instituteforenergyresearch.org/electricity-transmission/`.

[24] Office of Energy Efficiency Renewable Energy. Confronting the duck curve: How to address over-generation of solar energy. `https://www.energy.gov/eere/articles/confronting-duck-curve-how-address-over-generation-solar-energy`, October 12 2017.

[25] Keyla Gonzalez. Hosvd.m. `https://www.youtube.com/watch?v=79AaVLAFaK4`, Tensor Decomposition 2020.

[26] Keyla Gonzalez. Hosvd.m. `https://github.com/keylagonzalezabad/HOSVD-Microseismic-Data/blob/master/3D%20HOSVD/Matlab_HOSVD/HOSVD.m`, June 2021.

ACADEMIC VITA

Pierce J. McKelvey

Education

Bachelors of Science in Electrical Engineering, May 2022
The Pennsylvania State University
Schreyer Honors College, Honors in Electrical Engineering

Relevant Coursework:

- Fundamentals of Remote Sensing
- Communication Systems
- Engineering Electromagnetics
- Renewable Energy Systems
- Linear Control Systems
- Data Structures

Technical Experience

Undergraduate Research Assistant, Sept 2021 - Present
PSU Renewable Energy and Microgrid Power Lab, Supervisor: Dr. Yan Li

- Honors undergraduate work culminating with an honors thesis

EAHI Flight Test Engineering Intern, May - Aug 2021
Boeing Test & Evaluation, The Boeing Company, Seattle, WA

- Member of the 2021 ecoDemonstrator Instrumentation team.

Electrical Engineering Co-op, Jan - Aug 2020
Kimberley-Clark Corp., New Milford, CT

- Designed, built, tested, and implemented new data collection system utilizing an Android application (written in C) with Excel and OSIPI integration (VBA) used for analysis/condition monitoring.
- Data collection system garnered attention of employees in other manufacturing sites leading to sharing of project details with corporate headquarters.
- Authored training resources for end users in collaboration with site training coordinator.

Awards

- Madden Memorial Honors Scholarship in Engineering, 2021
- IEEE Power and Energy Society Scholarship Plus, 2019, 2020
- Dean's List, all semesters
- Eagle Scout

Extracurricular

- Schreyer Deloitte Leadership Development Center, Fall 2021
- Member of PSU IEEE chapter
- EECS Peer tutor for Eta Kappa Nu Epsilon chapter
- Engineering Orientation Network Mentor, The Pennsylvania State University, 2020