THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF ELECTRICAL ENGINEERING

MODELING AND SIMULATION OF RENEWABLE ENERGY SYSTEMS THROUGH TENSOR DECOMPOSITION

Jidapa Chalermkit
SPRING 2023

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

Renewable energy needs to be utilized more heavily in the production of electricity since the United States government is to reach its goal of a net-zero emissions economy. Due to its quick installation growth and the quantity of solar energy, solar photovoltaics (PV) is one of the renewable energy sources that attract the most attention from the general public. Solar power generation, however, varies with the time of day and the seasons. It is necessary to estimate the solar power output over time so that we can assess the reliability of solar PV energy and make appropriate plans for when to rely on energy from other sources.

Since the solar PV power output depends on the temperature and solar irradiance, the temperature and irradiance data are to be obtained to determine the power output. However, the data collected over the years will take up a lot of space. Tensor decomposition is a technique that can be used to minimize data storage and forecast future data.

Without having to keep all of the original data, we may predict the temperature and solar irradiance at various places over time by computing Tucker decomposition using the Tensor Toolbox. Outputting the predictions to the two-stage PV integration in Simulink, power generation can then be forecasted over time in accordance with the fluctuating temperature and solar irradiation.

Table of Contents

List of Figures

Acknowledgements

The thesis could not have been completed without help from Dr. Yan Li. She has been lighting up my interest in renewable energy systems since I took her sustainable energy course last year. She has been such a great thesis supervisor: guiding me on what I should do, offering help throughout the thesis completion steps, and mentally supporting me. I very much appreciate her help.

I would like to thank Dr. Julio Urbina, my thesis advisor, for his support throughout my college studies. Also, many thanks to all the authors of research papers and other materials I have used as references in this thesis for contributing such meaningful work.

Chapter 1

Literature Review

1.1 History of Tensor Decomposition

Hitchcock first introduced tensor decomposition in 1927; Cattell later developed the concept of a multiway model in 1944 [1]. However, tensor decomposition had not become popular until Tucker, Carroll and Chang, and Harshman included this concept in their psychometric literature. Following that, tensor decompositions were initially used in chemometrics by Appellof and Davidson, and since then, tensors have gained enormous popularity in that area [1]. Recently, tensor decompositions have been employed in a variety of applications, including signal processing, numerical linear algebra, computer vision, numerical analysis, data mining, graph analysis, neuroscience, etc. Tensor decomposition is also beneficial for power system applications, and a detailed explanation of this will be provided later. Tensors have a wide range of applications because they can offer a clear and intuitive mathematical framework for expressing and resolving problems [4].

1.2 Definition of Tensor

For a brief explanation, a tensor is usually described as a multi-dimensional array. A tensor is similar to a matrix in the sense that they both store mathematical data; however, tensor and matrix have different properties that help us distinguish them apart, which will be explained in Section 1.3. For now, important terms along with an example of tensors will be introduced.

1.2.1 Order

The term ”order” is used to describe the number of dimensions of tensors. This concept might be referred to using different terms, such as degree. Tensors in different orders are similar to the simpler concepts as follows:

- Order 0 tensor: scalar
- Order 1 tensor: 1-dimensional array (vector)
- Order 2 tensor: 2-dimensional array (matrix)
- Order n tensor: n-dimensional array

1.2.2 Examples of Tensor by Orders

Following are examples of how tensors in different orders can be written.

- Order 1 tensor:

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

- Order 2 tensor:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

- Order 3 tensor:

$$\begin{bmatrix} [1 & 2] & [3 & 4] & [5 & 6] \\ [7 & 8] & [9 & 10] & [11 & 12] \\ [13 & 14] & [15 & 16] & [17 & 18] \end{bmatrix}$$

- Order 4 tensor:

$$\begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} & \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} & \begin{bmatrix} 9 & 0 \\ 1 & 2 \end{bmatrix} \\ \begin{bmatrix} 7 & 9 \\ 3 & 4 \end{bmatrix} & \begin{bmatrix} 1 & 8 \\ 5 & 4 \end{bmatrix} & \begin{bmatrix} 6 & 1 \\ 4 & 9 \end{bmatrix} \\ \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} & \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} & \begin{bmatrix} 1 & 5 \\ 6 & 7 \end{bmatrix} \end{bmatrix}$$

1.3 Properties of Tensor and Their Advantages Over Matrix

Even though the tensor is often confused with the term ”matrix,” the tensor has exceptional properties that make them more suitable for applications.

1.3.1 Dynamical Property

Unlike matrices, tensors do not just store entries but also allow each entry in the tensor to interact with each other. This property is very beneficial in our energy systems modeling as the data will be automatically corrected to correspond to external changes. If we stored data in a matrix, we would have to change the entire matrix when a change occurred.

1.3.2 Flexibility in Tensor

Tensors provide more flexibility in data analysis compared to matrices as they can deal with data in different dimensions. This property allows us to save data in multiple ways: decimal format, binary format, etc. Furthermore, as a tensor is not regulated by a fixed dimension, it can work in situations where new data comes in a never-ending manner, which will continuously expand the dimension of the tensor.

1.3.3 Method of Moments

Tensors come into use as other algorithms do not work well in high dimensions. When a mathematical entity stores big data in high dimensions, the entity will take up large spaces, causing a lag in data analysis. However, tensors use the ”method of moments” in tensor decomposition, allowing the system to determine data configuration. The method of moments computes the statistical information corresponding to the data stored: mean, variance, and skewness. This method causes tensors to deal with data in a more organized way, which makes tensors work well at high dimensions.

1.4 Tensor Decomposition Methods and its Applications

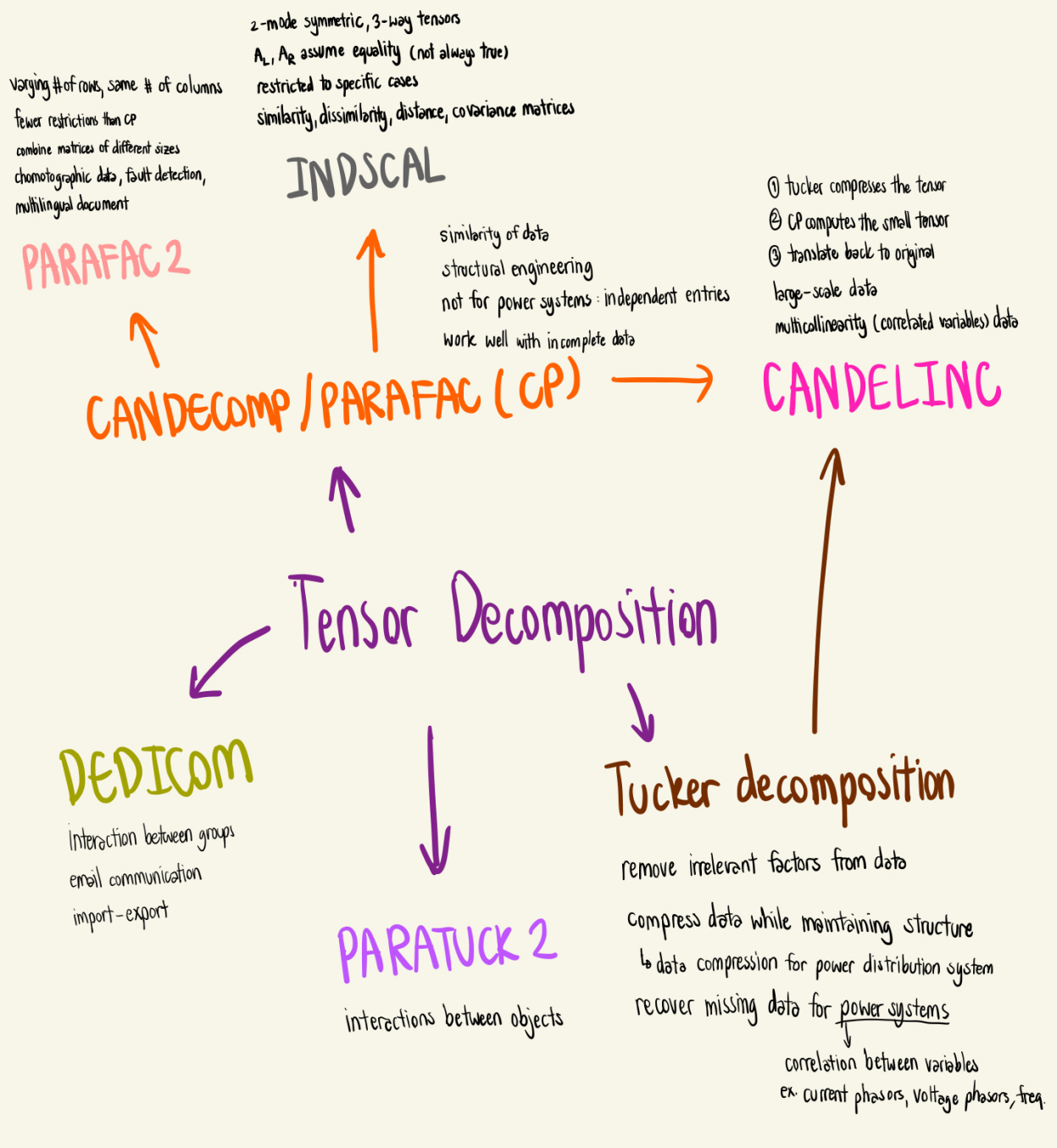

Figure 1.1: Key features of each decomposition method

The picture above depicts tensor decomposition methods and their key properties. It is shown that PARAFAC2 and INDSCAL are special cases of CANDECOMP/PARAFAC (CP) decomposition. Similarly, CANDELINC has arrows drawn from CP and Tucker decompositions since it

uses CP and Tucker decompositions in the calculation. A detailed description of each method is as follows.

1.4.1 CANDECOMP/PARAFAC (CP)

With the CP decomposition, we would be able to write a tensor in a form of the sum of rank-one tensors. To clarify this, the third-order tensor $\chi \in \Re^{IxJxK}$ can be written as follows.

$$\chi = \sum_{r=1}^{N} a_r \circ b_r \circ c_r$$

when n is a positive integer, $a_r \in \Re^I$, $b_r \in \Re^J$, and $c_r \in \Re^K$ for r = 1, 2, ..., N.

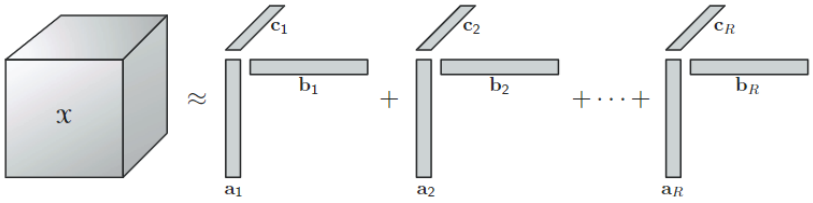

Figure 1.2: Representation of a CP decomposition [1]

One of the algorithms used to compute a CP decomposition is the Alternating Least Squares (ALS) method. The ALS method will determine a matrix at a time when the rest matrices are fixed, i.e., for a third-order tensor, when solving for a matrix A, matrices B and C are fixed. Similarly, matrices A and C are fixed when solving for B, and matrices A and B are fixed when solving for C.

Carroll and Chang introduced CANDECOMP to determine the similarity of data. Harshman then introduced PARAFAC to reduce the vagueness brought on by the two-dimensional Principal Component Analysis (PCA). With these two principles, CP decomposition is useful in many applications, including the modeling of fluorescence excitation-emission data in chemometrics, sensor array processing, telecommunications, independent component analysis (ICA), and neuroscience applications. Furthermore, CP decomposition works effectively even when some of the data is unknown or missing. [1]

1.4.2 Tucker Decomposition

Tucker decomposition is often called as HOSVD (Higher-order Singular Value Decomposition) With Tucker Decomposition, the tensor will be decomposed to a core tensor and factor matrices. For example, a third-order tensor $\chi \in \Re^{IxJxK}$ can be represented as a core tensor $g \in \Re^{PxQxR}$ and three factor matrices, $A \in \Re^{IxP}$, $B \in \Re^{JxQ}$, and $C \in \Re^{KxR}$ as shown in the figure below.

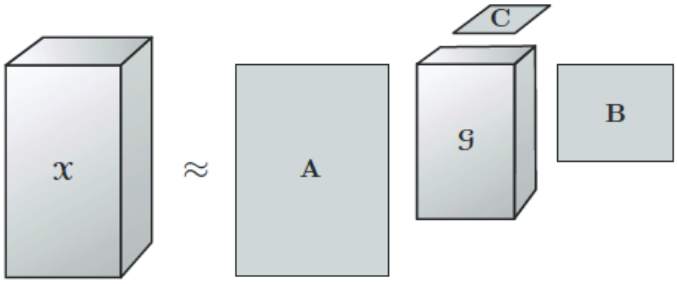

Figure 1.3: Representation of a Tucker decomposition [1]

Tucker decomposition is used in many applications, such as chemical analysis, signal processing, and facial image analysis. One of the great advantages of using the Tucker decomposition in facial image analysis is its ability to remove irrelevant effects from the data. Vasilescu and Terzopoulos introduced Tucker decomposition in computer vision for facial image data analysis called "TensorFaces". They took into account face image data from numerous subjects, where each person had a number of photos taken under various lighting conditions and camera angles. Then, the data would be categorized into several modes: person, lighting conditions, and camera angles. The Tucker decomposition's ability to remove irrelevant factors makes it beneficial in applications, including data compression for the power distribution system. [1]

1.4.3 INDSCAL

A specific case of CP decomposition for two-mode symmetric, three-way tensors is known as individual differences in scaling (INDSCAL). Without an explicit constraint mandating equality, the two "A matrices" are updated individually and considered separate factors (A_L and A_R). The estimations are different at the beginning of the process, but it is expected that the fundamental symmetry of the data will eventually allow the two factors to converge to be equal. However, equality does not always hold true in practice, informing that this method is not the best way to compute tensor decomposition. Although the INDSCAL is frequently beneficial when working with similarity, dissimilarity, distance, or covariance matrices, it is not widely used because it is limited to specific cases. [1]

1.4.4 PARAFAC2

PARAFAC2 is a special case of CP decomposition that can be used to for a set of matrices with varying numbers of rows but the same number of columns. Advantages of PARAFAC2 include fewer restrictions than CP and the ability to combine matrices of different sizes in one mode, such as those with the same column dimension but different numbers of rows. Applications of PARAFAC2 include spectrum detection for chromatographic data resolution, fault detection in semiconductor etching, and multilingual document clustering. [1]

1.4.5 CANDELINC

Canonical decomposition with linear constraints (CANDELINC) uses Tucker compression to perform CP computation on large-scale data. The process would begin by compressing the tensor using Tucker decomposition. Then, the compressed core tensor will be computed using CP decomposition. After that, it will be converted to the original tensor's CP decomposition. A few CP-ALS iterations on the entire tensor would be used to achieve the final results of the CANDELINC. As the CANDELINC is capable of dealing with large-scale data, it is used to deal with multicollinearity (the principle that many independent variables in a model are correlated) in chemometrics data. [1]

1.4.6 DEDICOM

Decomposition into directional components (DEDICOM) is a type of tensor decomposition. The great advantage of DEDICOM is that it determines if there is an interaction between modes of a matrix. Assuming I objects with a matrix $\chi \in \Re^{IxI}$, then

$$\chi = ARA^T$$

when matrix $A \in \Re^{IxR}$ represents a latent component and matrix $R \in \Re^{RxR}$ represents the interaction between components.

DEDICOM is normally used in applications that have interactions between groups, for example, asymmetric measures of import-export data, and email communication graphing. [1]

1.4.7 PARATUCK2

PARATUCK2 is the decomposition method that can be described as a combination of CP and Tucker decompositions. It is very beneficial in determining the interactions between two sets of items. Applications of PARATUCK2 often involve with the use that has interactions between objects. [1]

1.5 Applications of Tensor Decomposition in Power Systems

For power system applications, power systems have used tensor decomposition for model optimization and load predictions. Furthermore, potential power system applications in data compression, event detection, and cyber security analysis are made possible because of tensor decomposition's ability in recovering missing data [5]. According to Nuño-Ayon, et.al., the PARAFAC decomposition is typically used in structural engineering but not in the power system field [6]. If the PARAFAC decomposition is to be dealt with power system data, many assumptions have to be made. For example, it has to be assumed that entries in a column are linearly independent, and that the data from the ambient system and noise are not linked. [6] On the other hand, Tucker decomposition is generally used for applications related to power systems, for example, the data compression for power distribution systems as shown in the "Historical Multi-Station SCADA Data Compression of Distribution Management System Based on Tensor Tucker Decomposition" [7], or the data recovery for phasor measurement units as presented on "PMU Missing Data Recovery Using Tensor Decomposition" [5].

Tucker decomposition is suitable for applications because of its ability to remove irrelevant factors from the data. Moreover, it works very well when the size of the original tensor is large. If the core tensor g got from the first round of the Tucker decomposition is too large to represent the data, it can go through more loops of the Tucker decomposition until the preferred size of the core tensor g is obtained.

Chapter 2

Tensor Decomposition

2.1 Plans and Expectations

The goal of the thesis is to model PV integration and predict the solar power output associated with varying temperatures and solar irradiance. Unlike other software available, the result is not just an average amount of power generated per year. The power output associated with time is expected. It will be able to provide more details about solar power reliability.

Temperature and solar irradiance data will be measured at different locations over time. Because of the large data size, tensor decomposition will be used to reduce the data storage size. Moreover, tensor decomposition can be used to predict the missing temperature and irradiance data, so that the solar power generated can be predicted accordingly. It can also be applied to predict future power generation.

2.2 Choosing a Decomposition Method

According to section 1.5, Tucker decomposition is the most suitable decomposition method for this thesis as Tucker decomposition works well with large tensors. Additionally, unlike other decomposition methods–including PARAFAC and INDSCAL–Tucker does not have a lot of limitations that might cause the thesis results to be impractical. Moreover, there are research papers in the power area that made use of Tucker decomposition, ensuring Tucker decomposition is appropriate for the modeling of renewable energy systems in this thesis.

2.3 Tensor Toolbox

2.3.1 Introduction to Tensor Toolbox

Tensor toolbox is a toolbox available on MATLAB. It provides tools to work with multidimensional arrays. The functionalities of the Tensor toolbox include CP decomposition and Tucker decomposition. The toolbox was developed by Brett W. Bader, Tamara G. Kolda, and others. Since the Tensor toolbox is an open source, people can access it freely through www.tensortoolbox.org.

With the Tensor toolbox, a tensor can be easily created using the function **tensor**. Then, the function **tucker_als** can compute the Tucker decomposition [8]. The **tucker_als** function will take two inputs: a tensor, and the preferred size of the core tensor [8].

2.3.2 Computing Tucker Decomposition Using Tensor Toolbox

Data Gathering

For the purpose of this thesis, PV integration will be modeled on MATLAB. Since the PV array takes temperature and solar irradiance data as its inputs, these data are obtained from the National Solar Radiation Database (NSRDB) on the National Renewable Energy Laboratory (NREL) website [9]. The data obtained are the temperature in degrees Celsius and the Global Horizontal Irradiance (GHI) measured hourly for the entire year in 2021–the most recent year with available data–at different locations.

The Global Horizontal Irradiance (GHI) is of particular interest to photovoltaic installations

[10]. The Global Horizontal Irradiance (GHI) is the total amount of shortwave radiation since it includes both Direct Normal Irradiance (DNI) and Diffuse Horizontal Irradiance (DHI).

The locations picked to represent irradiance and temperature data are State College, PA; Sacramento, CA; Mesa, AZ; Honolulu, HI; Las Vegas, NV; and New Bedford, MA. Sacramento and Mesa are locations of interest since they are ones of the cities that receive the highest solar irradiance in the United States [11]. Honolulu and Las Vegas were ranked as the first and second cities that produce the most solar power [12]. Despite the little sunshine and considerable system loss due to snow in New Bedford, installing solar arrays there has a significant financial benefit. At \$0.20/kWh—more than 50% above the national average costs—New Bedford's energy costs rank among the highest in the USA [13]. Thus, solar power generation can save a lot of money on electricity bills. In addition, the city provides a number of incentives, such as tax benefits and renewable energy credits [13].

Importing Data

Once the data are exported to **Excel**, the **Import Data** tool on MATLAB generates six matrices–named statecollege, sacramento, mesa, honolulu, lasvegas, and newbedford–corresponding to six locations of interest. Each matrix is in the size of 8760*2. The data are measured hourly for the entire year, so there are $24\,hours/day * 365\,days/year = 8760\,entries$. The number 2 represents the number of data types, which are temperature and solar irradiance (GHI).

A	B
Temperature	GHI
9.7	3
9.3	0
9.1	0
8.5	0
7.8	0
7.3	0
6.9	0
6.3	0
5.7	0
5.3	0
4.9	0
4.4	0
4	0
3.7	0
5.3	5
8.6	142
12.2	320
15.4	466
17.2	562

Figure 2.1: First 20 rows of temperature and GHI data measured at Mesa, AZ stored in an Excel file

Creating a Tensor

Each of the six matrices storing temperature and irradiance data for each location will be stored in each slice of **data**. In this case, the data obtained at State College is stored in the first frontal slice, data obtained at Sacramento is stored in the second frontal slice, data obtained at Mesa is stored in the third frontal slice, and so on. Even though the variable **data** is in three-dimension with a size of 8760*2*6, it is stored as **double**, so the tensor decomposition cannot be completed through **data**.

```
data(:,:,1) = statecollege;
data(:,:,2) = sacramento;
data(:,:,3) = mesa;
data(:,:,4) = honolulu;
data(:,:,5) = lasvegas;
data(:,:,6) = newbedford;
```

Figure 2.2: MATLAB code used to create three-dimensional data storage

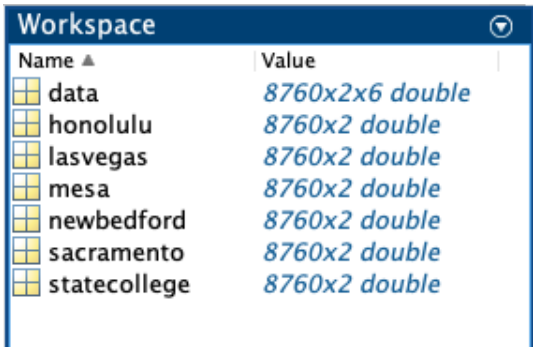

Figure 2.3: Variables with their data type and size shown on Workspace

The command **tensorX = tensor(data)** is used to convert data type from **double** for **data** to **tensor** under the variable named **tensorX**. Notice that even though **data** and **tensorX** are in the same size, their data type is different.

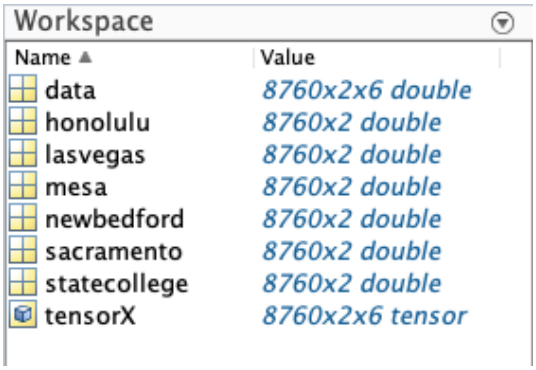

Figure 2.4: Variable named tensorX shown on Workspace after implementing the tensor toolbox command

Tucker Decomposition

The function **tucker_als** is used to calculate the Tucker decomposition. The function takes two inputs: an original tensor (tensorX) and the preferred size of the core tensor. In this case, the preferred size of the core tensor is n*2*3, which means the tensor will have n rows, 2 columns, and 3 slices when n can be any integer ranging from 2 to 8759. So, the code used is **tuckerX = tucker_als(tensorX, [n 2 3]);** when n is an integer between 2 and 8759.

The number of rows of the core tensor represents the number of categories used to group data in the first dimension of the original tensor. As the first dimension of the original tensor specifies the time throughout the year when the temperature and solar irradiance are measured, we would need to figure out how many groups we need to categorize the timing into. Since the temperature and GHI greatly fluctuate throughout the year in some locations because of seasonal changes but way less diverge in some locations, it is challenging to come up with a specific number of groups needed for categorization. However, it is known that the size of the core tensor is smaller than the size of the original tensor. The original tensor is in the size of 8760*2*6, thus the number of rows n will be less than 8760. Additionally, the number of rows must be greater than 1. Otherwise, it means all the data can fall into just one category, and there is no difference between them. Putting all data into just one category will not benefit us in data analysis. So, the possible number of rows ranges between 2 to 8759. The smaller number is preferred as the core tensor will be simpler. All possible numbers, starting from 2, will be implemented in the code until we obtain the core tensor and factor matrices that efficiently represent all data in the original tensor.

The original tensor has two columns, specifying temperature and solar irradiance. The size of the core tensor will not exceed the size of the original tensor, so the core tensor must have 1 or 2 columns. However, as discussed previously, the number of columns must be greater than 1. Consequently, the core tensor is expected to have two columns.

The number of slices of the core tensor represents the number of categories used to group data in the third dimension of the original tensor, which are locations. In this case, a 3-slice is expected since the locations can be categorized into three groups: snow areas, non-snow areas with high-temperature variations, and non-snow areas with less notable temperature variations. However, the 3-slice expected is just an assumption made by determining the raw data from NREL. The number of slices of the core tensor can actually vary from 2 to 5, which is the greatest number that is still smaller than the number of locations of interest which is 6. It is necessary to verify that the factor

matrix got from Tucker decomposition agrees with the 3-slice assumption. The verification will be explained in the Chapter 4.

Obtaining the Decomposition Results

As discussed in Section 1.4.2, the results of a Tucker decomposition are a core tensor and three-factor matrices. The code **tensorG = tuckerX.core** is used to obtain the core tensor. Three-factor matrices are named A, B, and C obtained using the command **tuckerX.U{1}**, **tuckerX.U{2}**, and **tuckerX.U{3}**, respectively.

```
tuckerX = tucker_als(tensorX, [2 2 3]);     %% tensor decomposition
tensorG = tuckerX.core;     %% core tensor
A = tuckerX.U{1};           %% factor matrix
B = tuckerX.U{2};           %% factor matrix
C = tuckerX.U{3};           %% factor matrix
```

Figure 2.5: MATLAB code used to calculate Tucker decomposition

Obtaining Values in Original Tensor

Instead of storing a large original tensor, we can store a smaller tensor—core tensor—and three factor matrices. With the function **M = ttm(tensorG,A,B,C);**, a tensor named M will be created when M contains entries close to the original tensor **tensorX**.

The original tensor is in the size of 8760*2*6, which means it contains 105,120 entries in total. On the contrary to that, the core tensor, with the size of n*2*3, contains only 6n entries. Factor matrix A contains 8760*n entries. Factor matrix B contains $2 * 2 = 4\,entries$. Factor matrix C contains $6 * 3 = 18\,entries$. The core tensor together with factor matrices contains only $22 + 8776n\,entries$. The tensor decomposition will save data storage space as long as n is less than or equal to 11.

Chapter 3

Modeling of Renewable Energy Systems

3.1 Solar Energy

By 2035, the U.S. government aims to establish a carbon-pollution-free power sector, and by 2050, it hopes to achieve a net-zero emissions economy [14]. To reach these goals, we must rely more on renewable energy sources. Solar PV is one of the renewable energy sources that is experiencing rapid growth in installations since the residential sector can also help install solar PVs at their homes. Moreover, the building of solar arrays does not seem to experience a shortage of materials since silicon is the most abundant material on Earth, and 95% of the solar PV on the market are silicon solar cells [15]. Furthermore, solar energy can provide more than enough energy—720 times more than global energy consumption. Solar energy measured on the Earth's surface is claimed to be as high as $5.5 * 10^{17}$ kWh/year when the global energy use is only $4 * 10^{14}$ kWh/year [16]. Consequently, solar PV technology becomes the main focus of this thesis, with the belief that it will play an even more important role in power systems.

3.2 MPPT

The output power of a solar array depends heavily on solar irradiance and temperature. It is found that when solar irradiance increases, the short circuit current (the current flows when there is no voltage across the solar cell) increases significantly. Since $P_{array} = I_{array} * V_{array}$, the increasing current causes a higher power output at higher solar irradiances. Similarly, it is claimed that the open circuit voltage (the voltage across the solar cell when there is no current flowing through it) increases remarkably when temperature increases. The increasing voltage causes a higher power output at higher temperatures as $P_{array} = I_{array} * V_{array}$. [2]

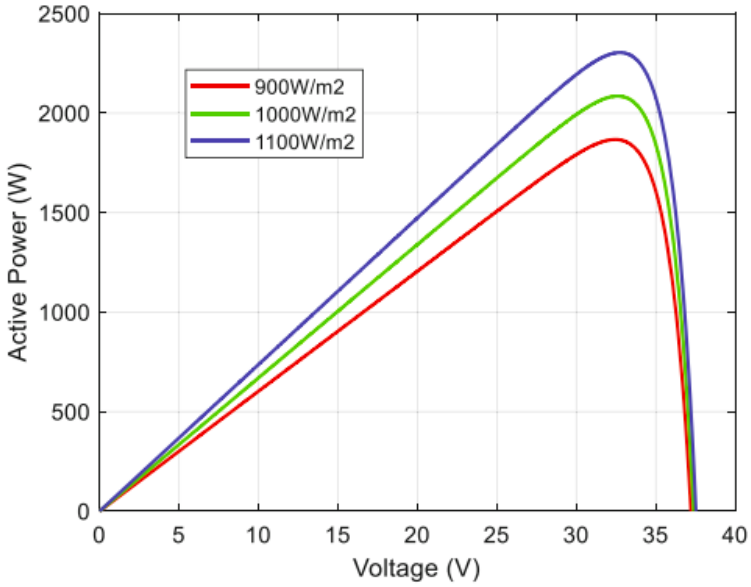

Figure 3.1: Power-Voltage relationship at different temperatures [2]

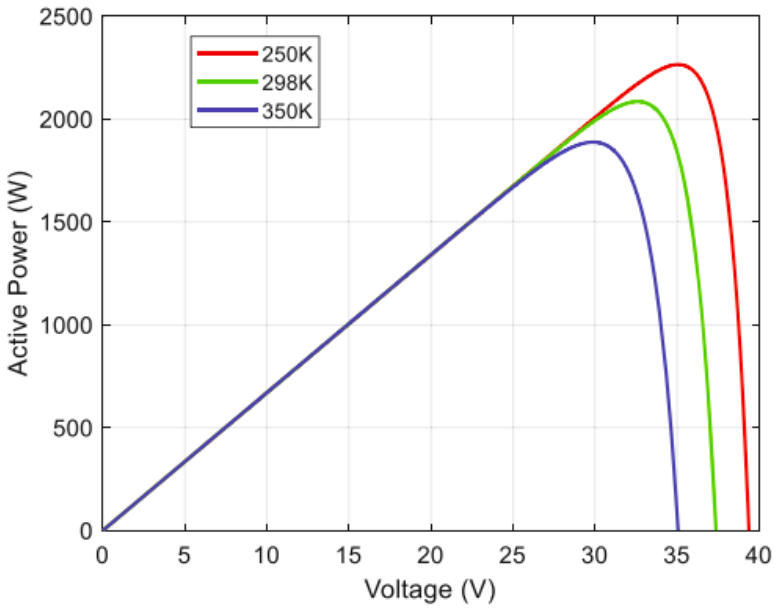

Figure 3.2: Power-Voltage relationship at different solar irradiances [2]

According to the figures above, the active power increases as the voltage increases. However, at one point, the active power suddenly drops if the voltage keeps increasing. The voltage that corresponds to the maximum active power is called the ”Maximum Power Point (MPP)”. Our goal is to operate solar arrays at this maximum power point in order to gain the highest power possible. MPPT (Maximum Power Point Tracking) is the algorithm used to draw the most power possible from PV modules under specific solar irradiance and temperature conditions. [17]

3.3 PV integration

Commercial and residential sectors can easily install solar arrays to generate photovoltaic (PV) power; however, the amount of power generated is frequently unstable. PV power output depends heavily on solar irradiance and temperature, causing solar arrays to produce little or even no power at times—during the night or when it is cloudy. Consequently, relying entirely on PV power is impractical. One of the solutions to this problem is to connect PV panels to the utility grid. When solar panels are unable to produce sufficient energy, the utility grid can provide energy to meet consumer needs. However, since solar arrays generate DC power, the output needs to be converted to AC before connecting to the grid. A network that enables PV power to enter the national electricity grid is known as PV integration [18]. Depending on the needs and PV output voltage, either single-stage integration or two-stage integration can be implemented [2].

3.3.1 Single-Stage Integration

An inverter is used in the single-stage integration to convert DC voltage to AC voltage. The inverter controller includes MPPT functionality [2]. However, since only inverters are utilized, its

controller is complex, making it challenging to organize the control parameters. [2]

In addition to inverters, the single-stage integration also needs a phase lock loop (PLL), an outer controller, an inner controller, and a PWM generator. A more detailed description of these components will be provided in a later section.

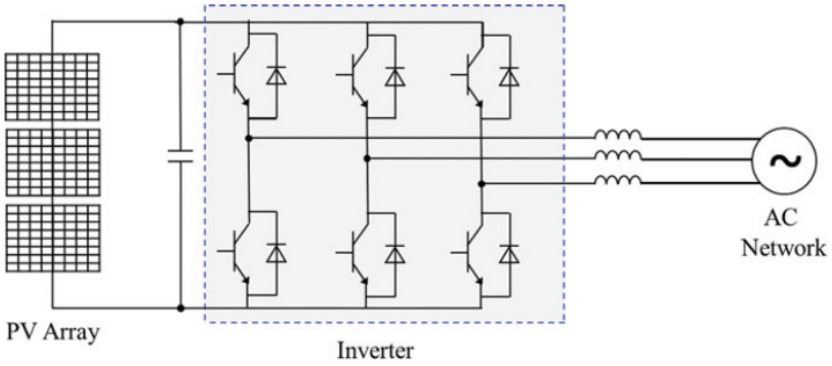

Figure 3.3: Topology of a single-stage PV integration [2]

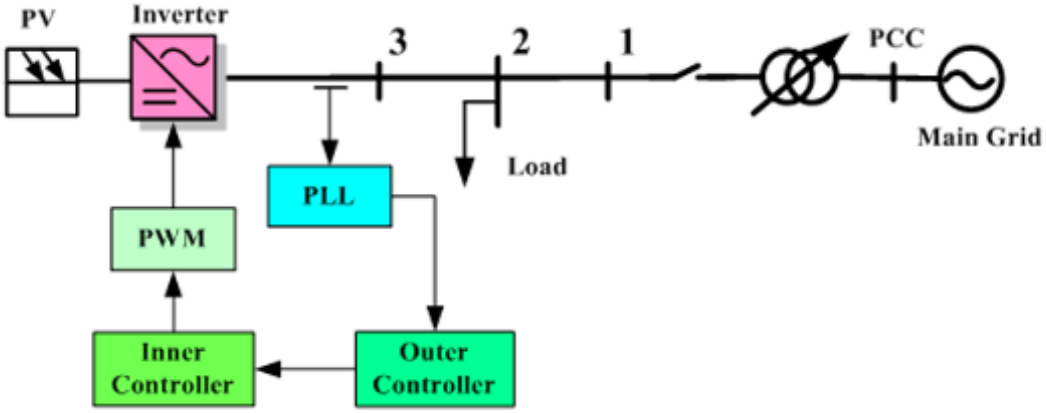

Figure 3.4: Representation of a single-stage PV integration [2]

3.3.2 Two-Stage Integration

In addition to what is in the single-stage integration, the two-stage integration also has a chopper with its controller. A chopper—a boost converter—is used to increase the DC output voltage of the PV array before letting the inverter convert the DC voltage to AC. In two-stage, the MPPT function is implemented in the chopper's controller rather than the inverter's controller. The job of the inverter's controller is to keep the capacitor's voltage constant.

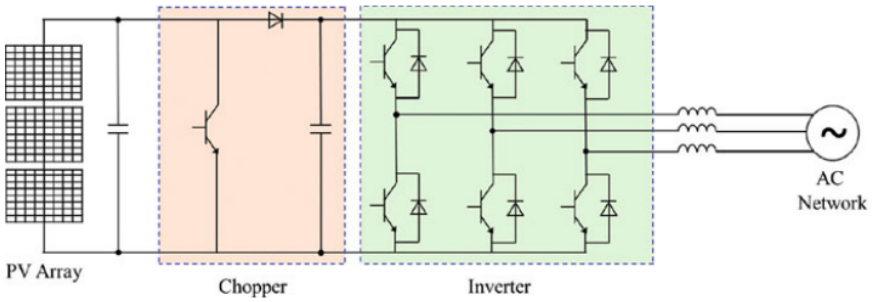

Figure 3.5: Topology of a two-stage PV integration [2]

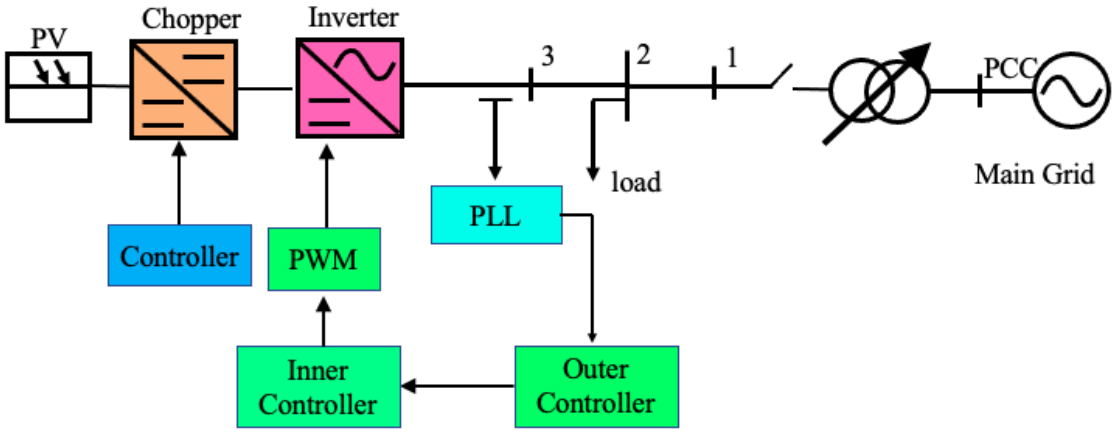

Figure 3.6: Representation of a two-stage PV integration [2]

3.4 Components in PV integration

3.4.1 Inverter

Inverter is a power electronic device that converts DC voltage to AC voltage. To produce AC voltage, we open and close the Insulated Gate Bipolar Transistor (IGBT) in pairs, and one of the common ways to regulate IGBT is through pulse width modulation (PWM) [2]. The PWM compares the carrier signal and modulation signal to determine the output voltage [2]. Assuming the DC voltage entered is U_{dc}, the output voltage will be U_{dc} when the modulation signal's value is greater than the value of the carrier signal. On the other hand, the output voltage will be $-U_{dc}$ when the modulation signal's value is less than the value of the carrier signal.

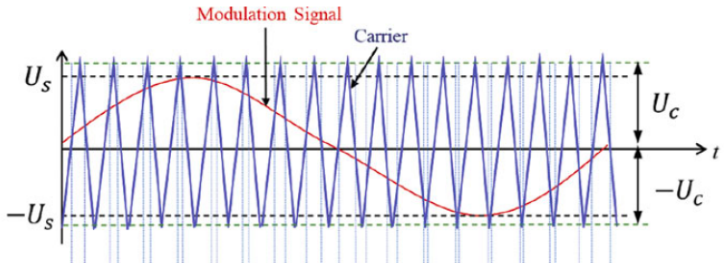

Figure 3.7: Carrier signal and modulation signal [2]

To generate the three-phase AC voltage, a three-phase inverter must be implemented. The line-to-line output voltage of an inverter can be expressed as follows [2].

$$U_{L1} = \frac{\sqrt{3}MU_{dc}}{2\sqrt{2}}$$

when U_{L1} = RMS value of the line-to-line output voltage
M = modulation index = ratio of reference signal amplitude to carrier signal amplitude [19]
U_{dc} = DC voltage

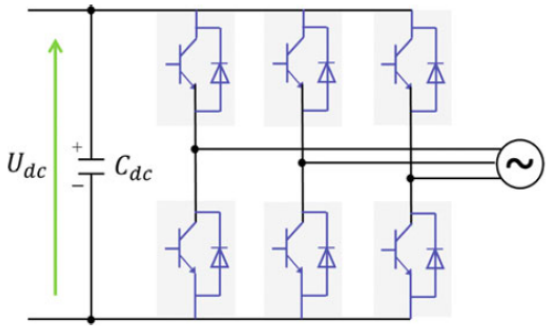

Figure 3.8: Three-phase inverter [2]

3.4.2 Inverter Controller

The purpose of the controller is to adjust the modulation index that relates to the modulation signal.

DQ0 Transformation

The direct-quadrature-zero (dq0) transformation is implemented to convert the AC signal to a DC signal in order to simplify the control of the signal [2]. Assuming v_a, v_b, v_c are the instantaneous voltages of phase A, B, and C, respectively, the dq0 signal can be obtained using the following formula [2].

$$\begin{bmatrix} v_d \\ v_q \\ v_0 \end{bmatrix} = k_1 \begin{bmatrix} cos\theta & cos(\theta - \frac{2\pi}{3}) & cos(\theta + \frac{2\pi}{3}) \\ -sin\theta & -sin(\theta - \frac{2\pi}{3}) & -sin(\theta + \frac{2\pi}{3}) \\ k_2 & k_2 & k_2 \end{bmatrix} \begin{bmatrix} v_a \\ v_b \\ v_c \end{bmatrix}$$

when θ = angle between the d-axis and a-axis
$k_1 = \sqrt{2/3}$ and $k_2 = \sqrt{1/2}$ for a constant power transformation
$k_1 = 2/3$ and $k_2 = 1/2$ for a constant amplitude transformation

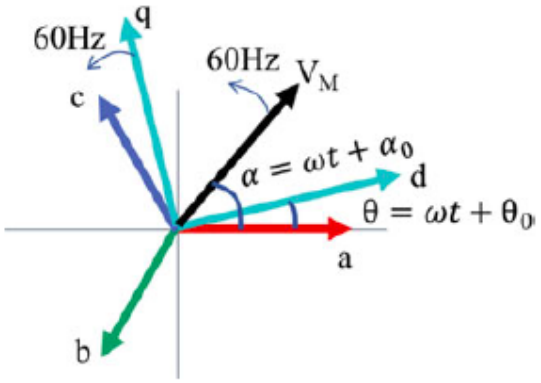

Figure 3.9: DQ0 transformation [2]

Phase Lock Loop (PLL)

Phase Lock Loop (PLL) makes the dq0 transformation even simpler. PLL aims to have v_q = 0 so that $v_d = V_M$ and the d-axis signal is in phase with the voltage phasor. As a result, V_M may be easily altered by manipulating merely the signal on the d-axis [2].

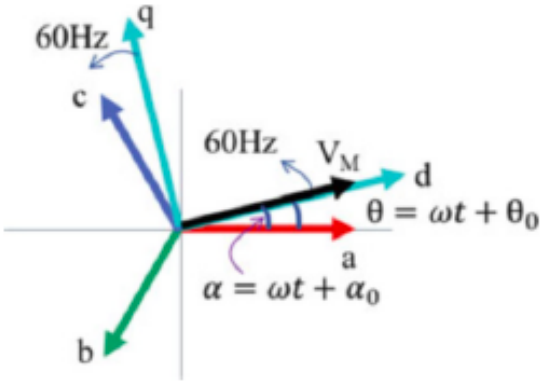

Figure 3.10: DQ0 transformation when $v_d = V_M$ [2]

Double Loop Controller

A double-loop controller consists of an inner controller and an outer controller. The inner controller regulates the current $[i_{Id}, i_{Iq}]$ in order to modify the output voltage $[v_{Id}, v_{Iq}]$ [2]. The outside controller, on the other hand, is employed to monitor voltage, active power, or reactive power [2].

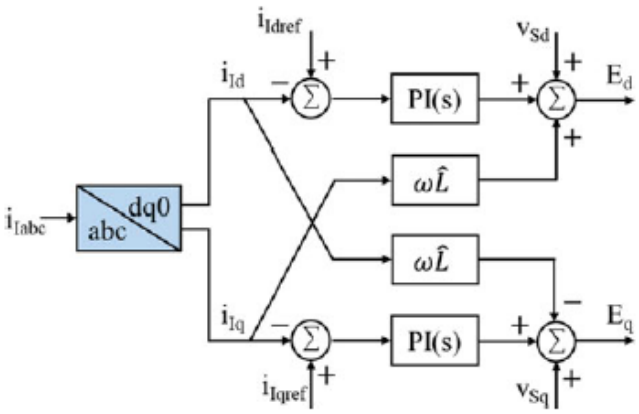

Figure 3.11: Inner controller [2]

PWM Generator

Using the PWM generator, switching signals for controlling IGBT are produced by comparing the carrier signal with the modulation signal. Additionally, PWM generator is used to convert the dc control signal to the abc signal via an inverse dq0 transformation [2].

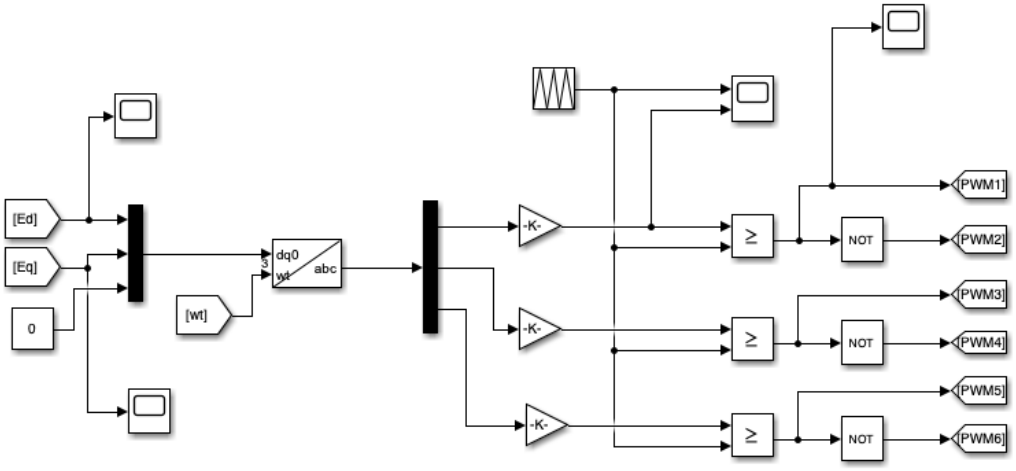

Figure 3.12: PWM generator

3.4.3 Chopper

A chopper, also referred to as a DC-to-DC converter, is a power electronic device. Boost choppers, buck choppers, and buck-boost choppers are different types of choppers. The boost chopper is a step-up converter, which means the output voltage of the chopper will be higher than the input voltage. The buck chopper is a step-down converter, so the output voltage will be lower than the input voltage. The buck-boost chopper can step up or step down voltage depending on the operation conditions.

Prior to letting an inverter convert DC voltage to AC in the two-stage integration, a boost

chopper is typically employed to raise the DC output voltage of the PV array. The boost chopper's controller typically implements the MPPT function.

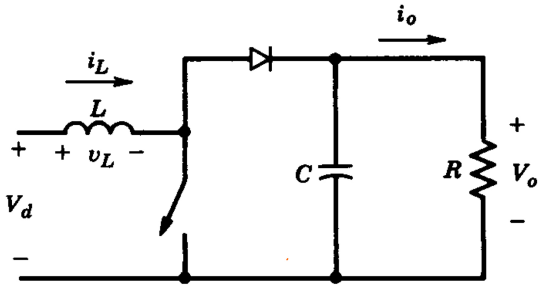

Figure 3.13: Boost converter [3]

Chapter 4

Tucker Decomposition Results and its Applications in PV Integration Simulation

4.1 Tucker Decomposition Results

As discussed in Chapter 2, we will compute Tucker Decomposition when the core tensor is in the size of $n*2*3$ when n is an integer between 2 and 11. The smaller n is preferred. The function **M = ttm(tensorG,A,B,C);** will be implemented to determine if the core tensor and factor matrices properly represent all data in the original tensor. The temperature and GHI values obtained from the NREL database will be compared with the data obtained from the decomposition for each location in the graph format. It is expected that the tensor **M** got from the operations between the core tensor **tensorG** and factor matrices **A, B, and C** will effectively estimate temperature and GHI data.

4.1.1 Results when core tensor is in the size of 2*2*3

In this case, the number of rows of the core tensor is 2, which is the least number possible. The numbers of columns and slices are 2 and 3, respectively, as discussed in Chapter 2. Graphs between the actual data and data got from the operation of a core tensor and factor matrices will be plotted. If the data got from the decomposition results effectively represents the original data, we can stop there. If it does not, we need to increase the number of rows of the core tensor specified. The drawback is that the core tensor will be more complicated, making it more difficult to understand how data related to each other. The curves for temperature are as follows.

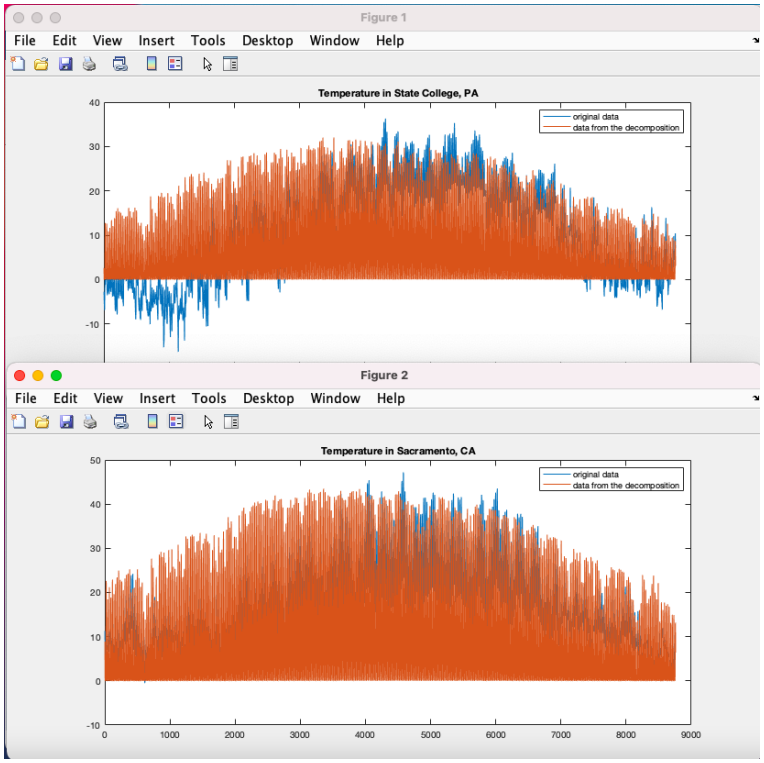

Figure 4.1: Actual temperature and temperature got from the decomposition at State College and Sacramento

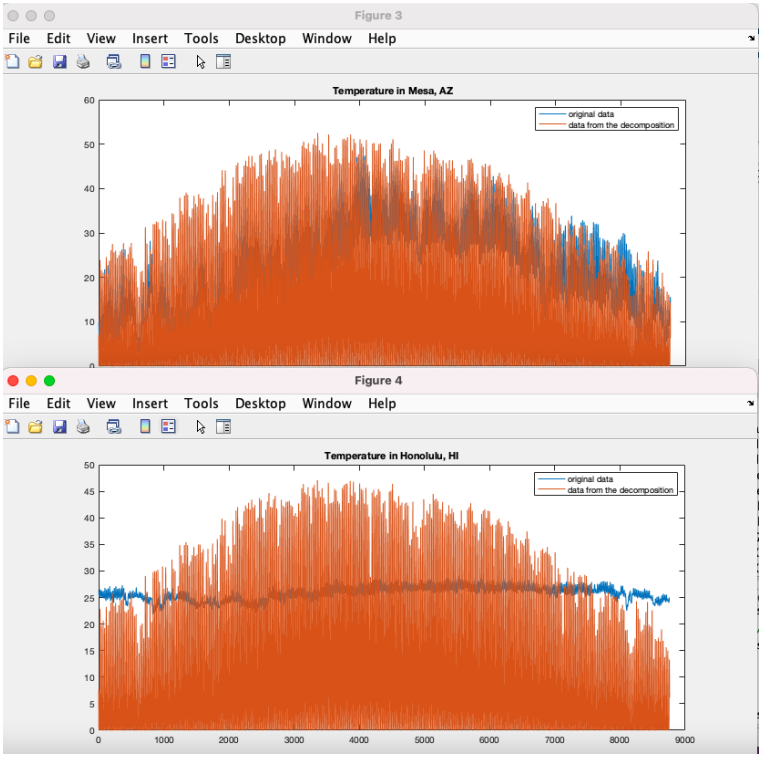

Figure 4.2: Actual temperature and temperature got from the decomposition at Mesa and Honolulu

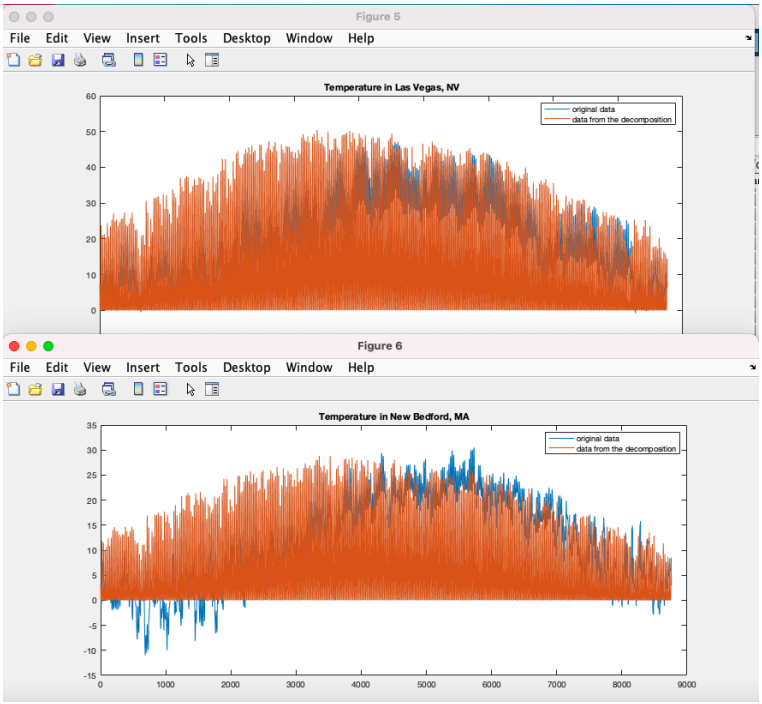

Figure 4.3: Actual temperature and temperature got from the decomposition at Las Vegas and New Bedford

Determining the actual temperature (blue curve) and the temperature got from the decomposition (orange curve), the curves do not match. We can conclude that the core tensor in 2*2*3 size does not adequately represents the actual data.

In terms of solar irradiance, the GHI data got from the decomposition follow the trend of the actual irradiance. However, the difference between them is still to some noticeable extent. As a result, the core tensor should be sized up so that it can provide more accurate temperature and GHI prediction.

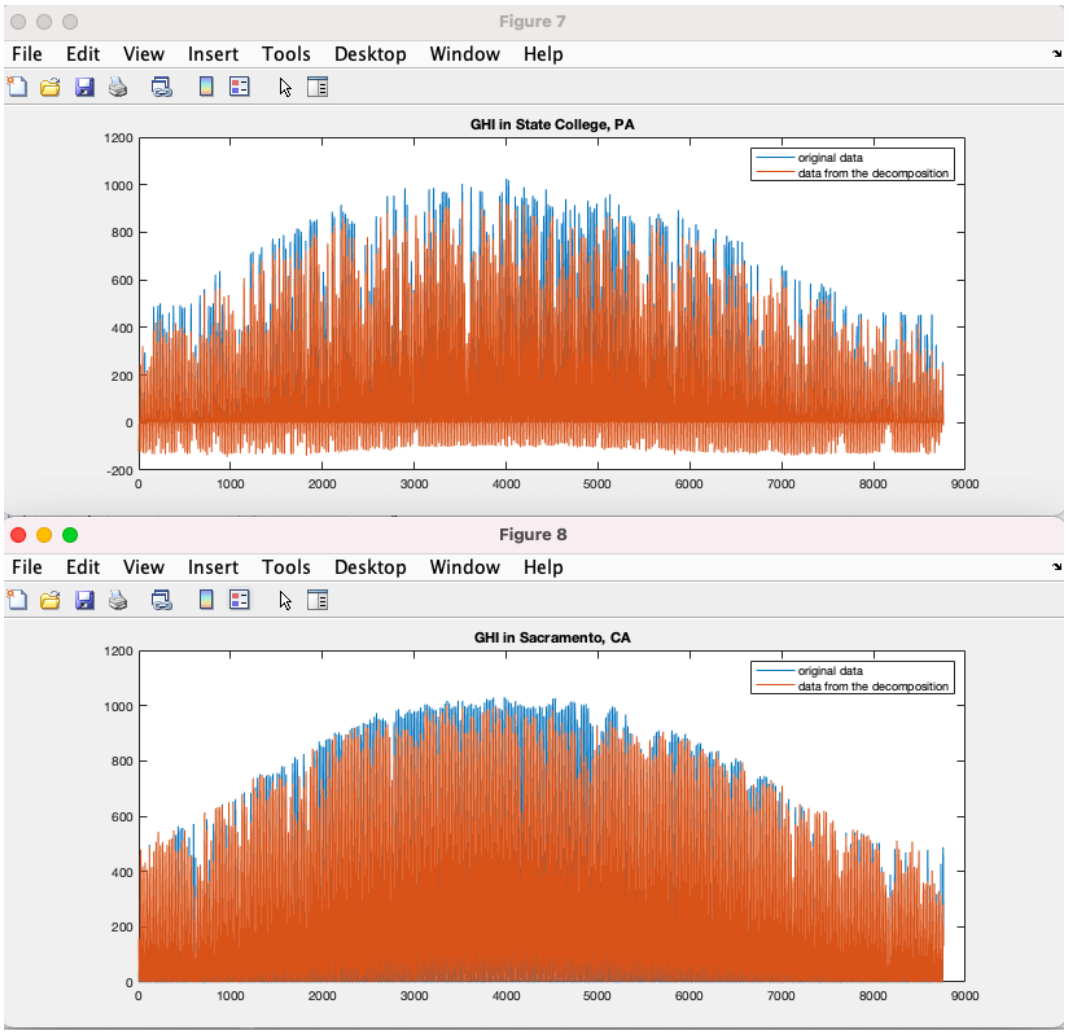

Figure 4.4: Actual GHI and GHI got from the decomposition at State College and Sacramento

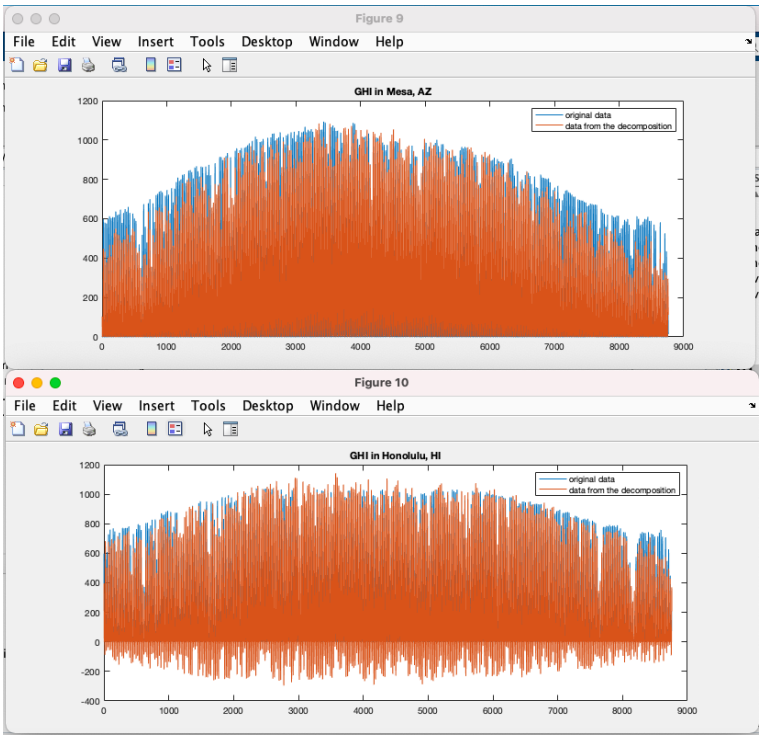

Figure 4.5: Actual GHI and GHI got from the decomposition at Mesa and Honolulu

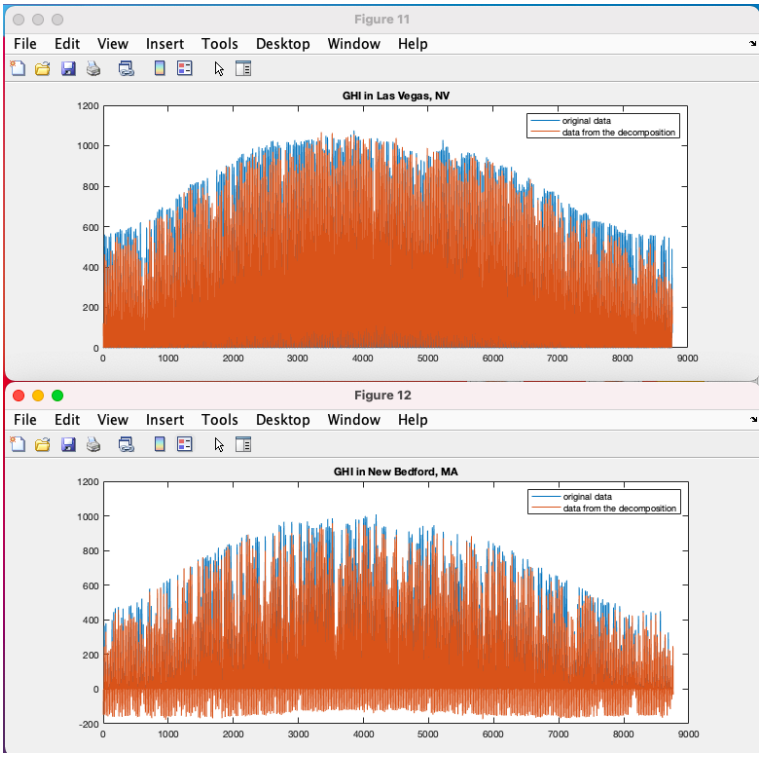

Figure 4.6: Actual GHI and GHI got from the decomposition at Las Vegas and New Bedford

4.1.2 Results when core tensor is in the size of 5*2*3

The number of rows of the core tensor has been increasing from 2 to 5. Even though the greatest number of rows possible is 11 as discussed previously, we do not go beyond 5. The 5-row core tensor provides a pretty accurate representation of the original tensor, and more rows mean a more complicated tensor representation, which is undesirable.

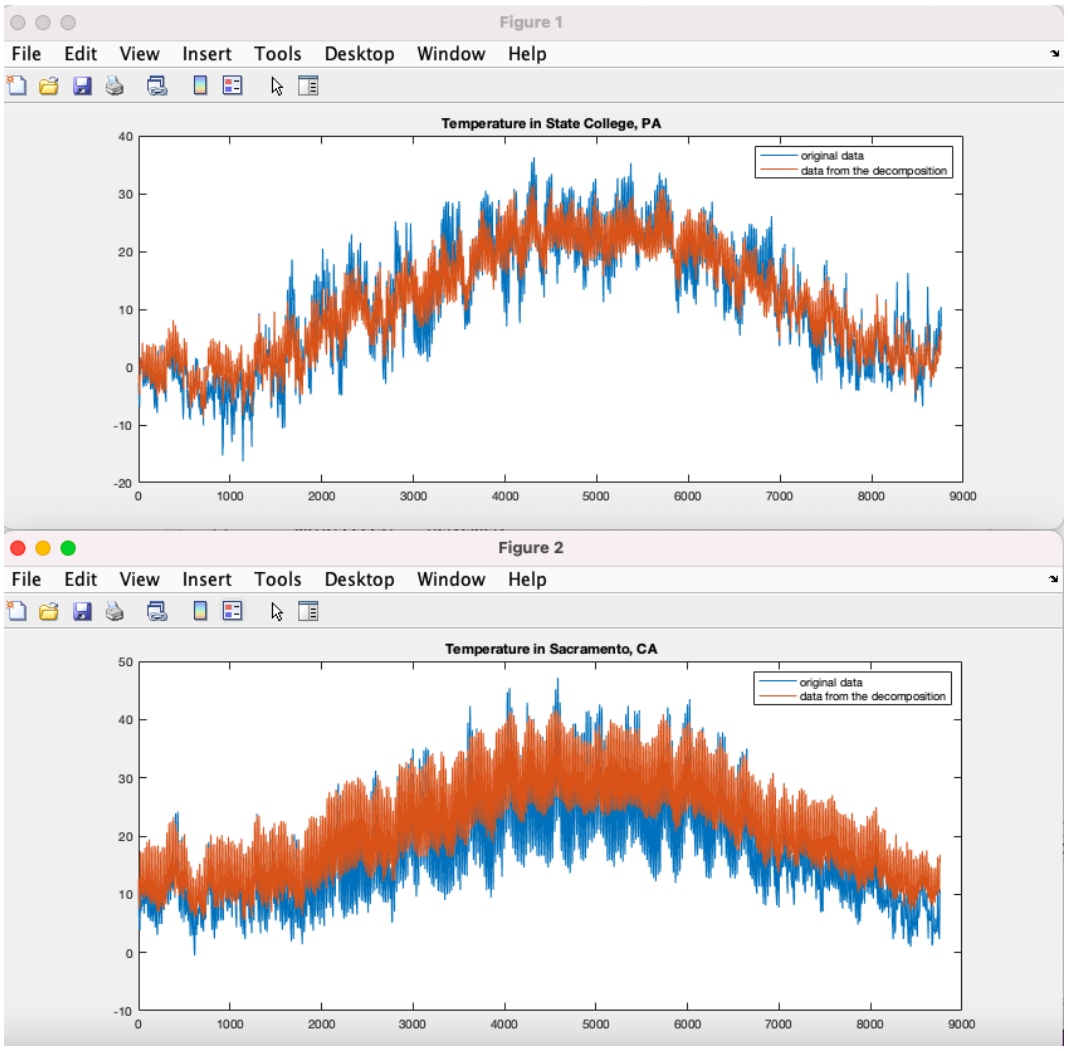

Figure 4.7: Actual temperature and temperature got from the decomposition at State College and Sacramento

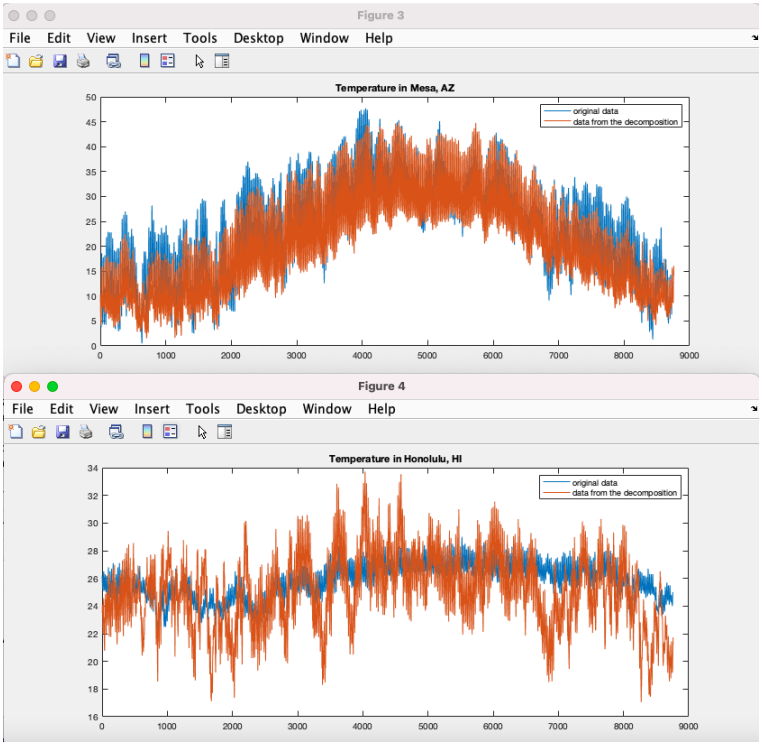

Figure 4.8: Actual temperature and temperature got from the decomposition at Mesa and Honolulu

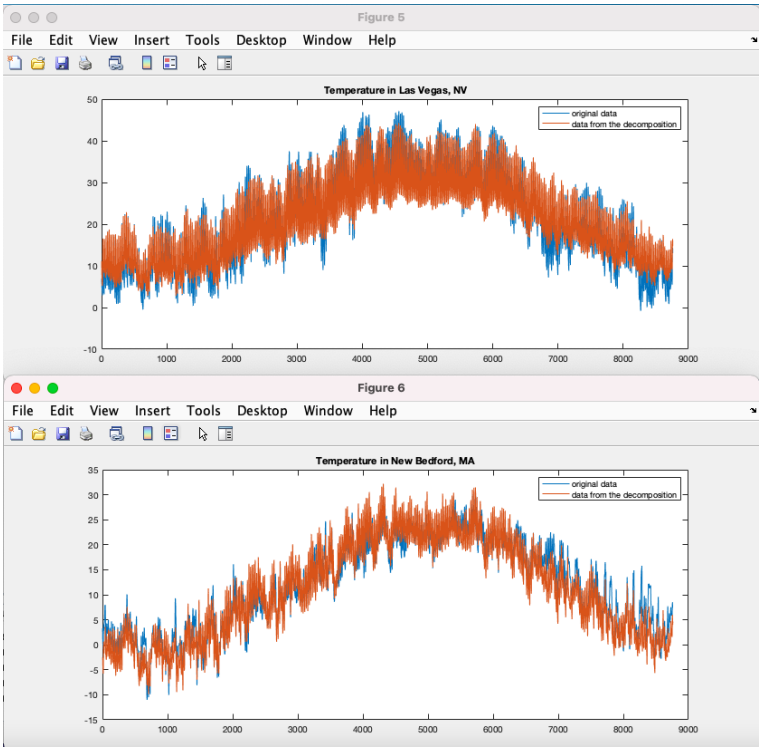

Figure 4.9: Actual temperature and temperature got from the decomposition at Las Vegas and New Bedford

Compared to Figures 4.1, 4.2, and 4.3, the temperature estimation using a 5-row core tensor shows a significant improvement. The estimation using the core tensor and factor matrices accurately follows the trend of the actual temperature data for most places. Unfortunately, the temperature estimation at Honolulu, Hawaii, is incorrect, as seen in Figure 4.8. Among all 6 locations of interest, Honolulu has the least temperature variation throughout the year. In conclusion, the Tucker decomposition for the particular tensor we developed is not appropriate to represent the temperature at sites with minimal temperature swings. Alternatively, the original tensor must include data at more locations, particularly those with weather patterns similar to Honolulu.

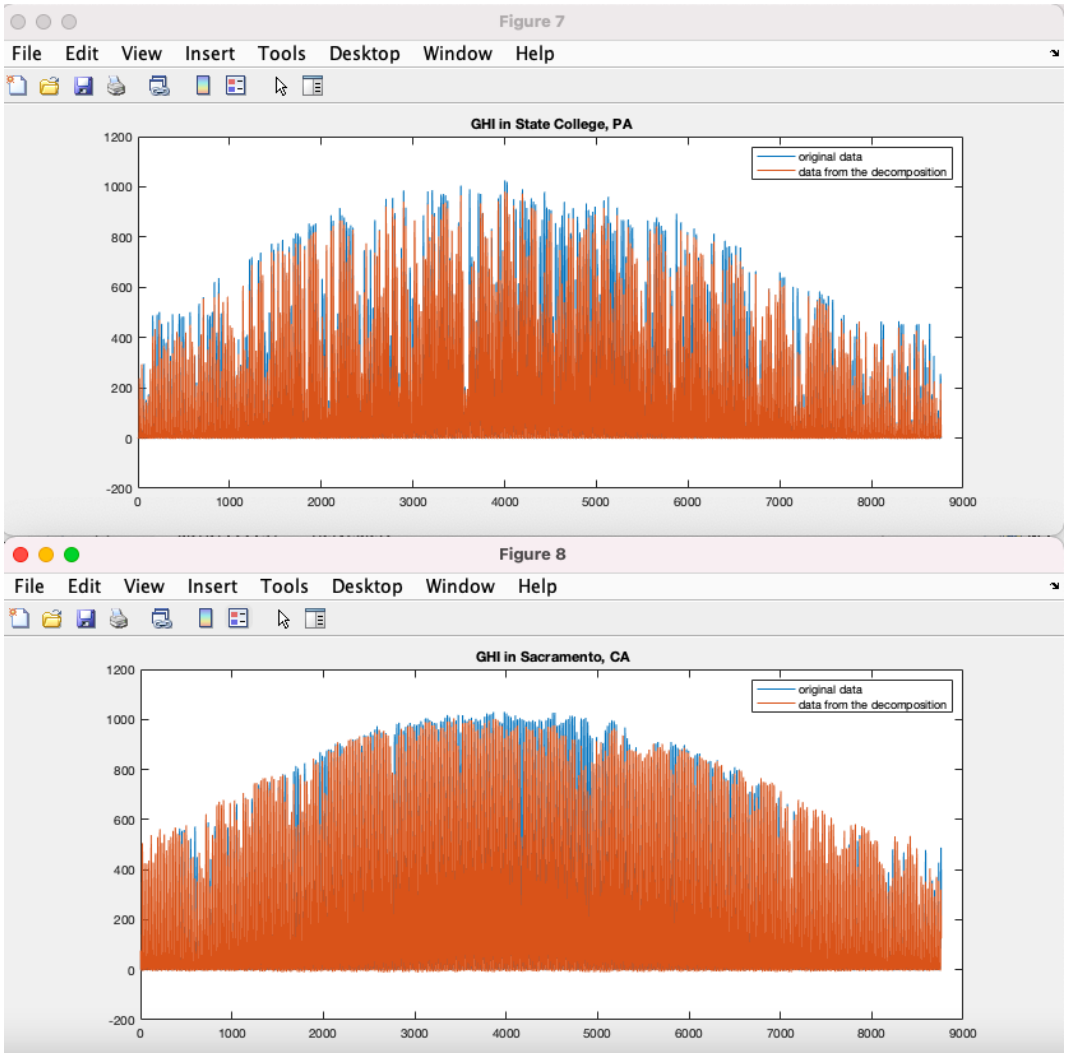

Figure 4.10: Actual GHI and GHI got from the decomposition at State College and Sacramento

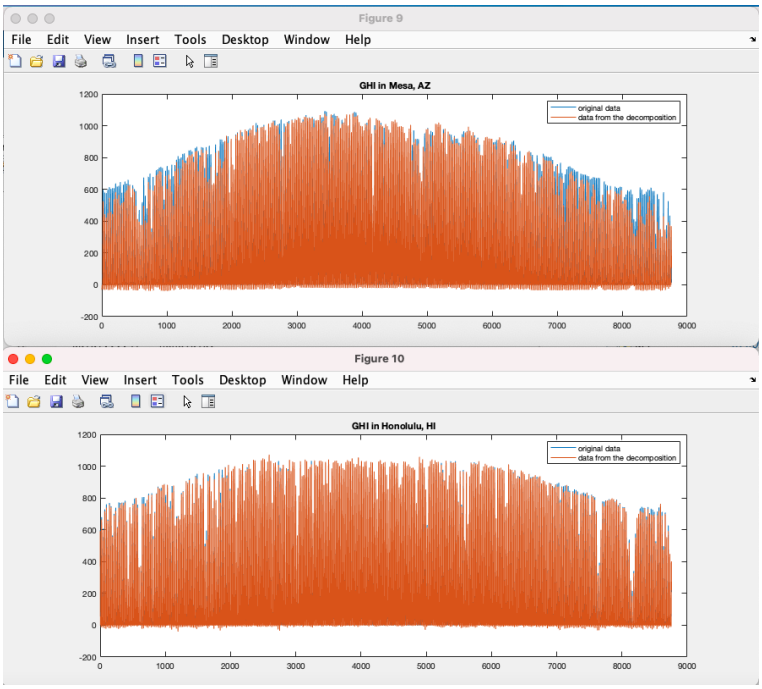

Figure 4.11: Actual GHI and GHI got from the decomposition at Mesa and Honolulu

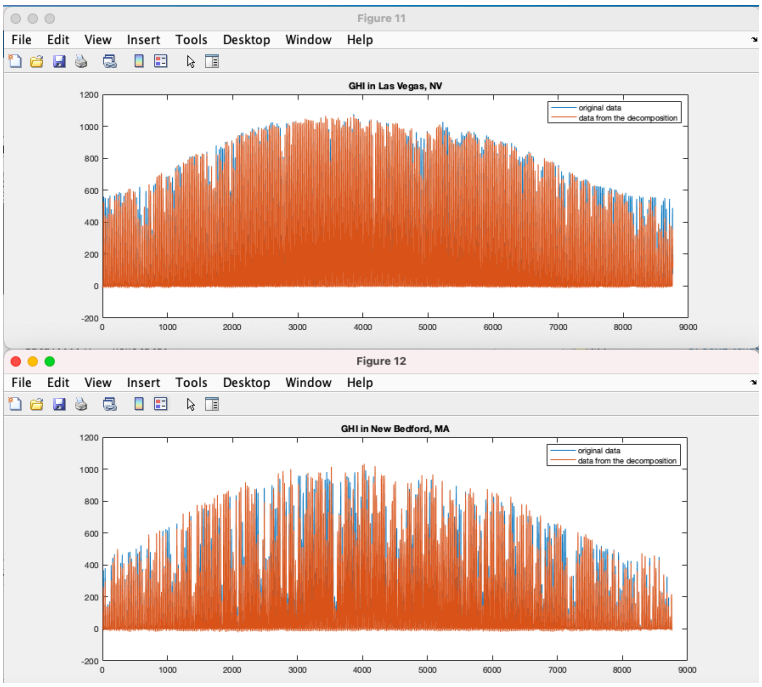

Figure 4.12: Actual GHI and GHI got from the decomposition at Las Vegas and New Bedford

For solar irradiance, the GHI estimation using a 5-row core tensor is more accurate than the estimation using a 2-row core tensor. The GHI estimates are good for all locations. The trends are correct, and the GHI values are pretty close to the actual data. Impressively, the estimations for

Honolulu and Las Vegas are almost 100% accurate, showing a significant improvement from the 2-row core tensor estimation.

Determining the accuracy in temperature and GHI estimation using a 5-row core tensor, it is appropriate to use a core tensor in the size of 5*2*3 to represent all temperature and GHI data at the six specified locations.

4.1.3 Verifying the Results

Factor Matrix A

As explained in Chapter 2 **Tucker Decomposition**, factor matrix **A** will categorize the time throughout the year into groups. In this case, we categorize the time into 5 groups. Outputing the factor matrix **A**, the numbers in all five columns—which correspond to five groups—repeat its pattern every 24 hours. For instance, numbers in the first row are similar to numbers in the 25^{th} row. This is what we expect to see since the temperature and GHI are similar when measured at the same time of the day.

A

8760x5 double

	1	2	3	4	5
1	0.0036	-0.0131	-0.0218	-0.0071	0.0136
2	0.0025	-0.0100	-0.0174	-0.0038	0.0146
3	0.0014	-0.0057	-0.0099	0.0014	0.0162
4	3.2991e-...	-0.0013	-0.0023	0.0066	0.0178
5	1.2484e-...	-6.1256e...	-1.2651e...	0.0078	0.0187
6	1.1845e-...	-5.9785e...	-1.2125e...	0.0075	0.0192
7	1.1377e-...	-5.8564e...	-1.1749e...	0.0072	0.0196
8	1.0866e-...	-5.6921e...	-1.1342e...	0.0069	0.0199
9	1.0378e-...	-5.5787e...	-1.0954e...	0.0067	0.0204
10	9.9245e-...	-5.4846e...	-1.0578e...	0.0064	0.0208
11	9.5703e-...	-5.3624e...	-1.0295e...	0.0062	0.0210
12	9.3171e-...	-5.0781e...	-1.0098e...	0.0061	0.0206
13	1.2091e-...	5.1821e-...	-0.0010	0.0057	0.0196
14	7.4030e-...	0.0033	-0.0057	0.0028	0.0183
15	0.0013	0.0059	-0.0100	7.0304e-...	0.0168
16	0.0033	0.0083	-0.0079	5.6440e-...	0.0162
17	0.0057	0.0067	0.0048	0.0051	0.0165
18	0.0081	0.0047	0.0148	0.0080	0.0170
19	0.0095	0.0021	0.0177	0.0083	0.0173
20	0.0123	-0.0058	0.0153	0.0032	0.0169
21	0.0130	-0.0109	0.0072	-0.0020	0.0160
22	0.0120	-0.0143	0.0021	-0.0037	0.0152
23	0.0089	-0.0140	-0.0066	-0.0042	0.0139
24	0.0064	-0.0138	-0.0145	-0.0051	0.0129

Figure 4.13: First 24 rows of the factor matrix A

A

8760x5 double

	1	2	3	4	5
25	0.0036	-0.0130	-0.0215	-0.0067	0.0124
26	0.0026	-0.0102	-0.0177	-0.0036	0.0127
27	0.0015	-0.0058	-0.0102	0.0019	0.0134
28	3.8637e-...	-0.0015	-0.0027	0.0074	0.0142
29	1.5150e-...	-5.8044e...	-1.4885e...	0.0093	0.0144
30	1.5001e-...	-5.7345e...	-1.4751e...	0.0093	0.0143
31	1.4706e-...	-5.4595e...	-1.4530e...	0.0091	0.0141
32	1.4080e-...	-4.9258e...	-1.4071e...	0.0088	0.0139
33	1.3563e-...	-4.4713e...	-1.3682e...	0.0085	0.0137
34	1.3511e-...	-4.3165e...	-1.3658e...	0.0085	0.0135
35	1.3504e-...	-4.2049e...	-1.3655e...	0.0085	0.0132
36	1.3446e-...	-4.1295e...	-1.3606e...	0.0085	0.0130
37	1.1051e-...	4.4968e-...	-9.1164e...	0.0080	0.0127
38	9.3023e-...	0.0041	-0.0069	0.0040	0.0127
39	6.3721e-...	0.0026	-0.0040	0.0060	0.0122
40	0.0019	0.0048	-0.0044	0.0051	0.0120
41	0.0044	0.0069	1.0782e-...	0.0053	0.0123
42	0.0084	0.0095	0.0048	0.0036	0.0134
43	0.0108	0.0043	0.0049	7.9544e-...	0.0130
44	0.0119	-5.9849e...	0.0020	-0.0022	0.0127
45	0.0107	-0.0079	-0.0028	-0.0035	0.0121
46	0.0084	-0.0087	0.0031	0.0023	0.0135
47	0.0071	-0.0108	-0.0045	-5.7036e...	0.0135
48	0.0052	-0.0121	-0.0140	-0.0033	0.0126

Figure 4.14: 25^{th} to 48^{th} row of the factor matrix A

Factor Matrix B

Similarly, factor matrix **B** will divide data types (temperature and GHI) into two groups. The first row of **B** represents the temperature data type. It is ranked low for group 1 (the first column) and ranked high for group 2 (the second column). On the contrary, the second row, representing GHI, is ranked high for group 1 and ranked low for group 2.

B

2x2 double

	1	2
1	0.0412	0.9992
2	0.9992	-0.0412

Figure 4.15: Factor matrix B

Factor Matrix C

Factor matrix **C** categorizes six locations into three groups. Among all locations, there are only two places that have snow. Similar numbers appear in all three columns in the first row, which corresponds to State College, PA, and the last column, which corresponds to New Bedford, MA. This indicates that one of the classification methods Tensor Toolbox uses to classify places is snow-no snow. In addition, the quantities in the third column of **C** are inversely proportional to the highest temperature in each location, e.g., the greater the highest temperature at a location is, the smaller number in the third column for the corresponding row will be.

C

6x3 double

	1	2	3
1	0.2804	-0.4807	0.3570
2	0.4634	0.0787	-0.2381
3	0.4822	-0.0639	-0.3785
4	0.4121	0.6867	0.5848
5	0.4805	-0.0185	-0.3552
6	0.2708	-0.5354	0.4521

Figure 4.16: Factor matrix C

4.2 PV Integration Simulation

After retrieving the temperature and GHI data at a location using the core tensor and factor matrices, we can predict the solar power output at that location over time. The irradiance data from **MATLAB** can be exported to **Simulink** using the **1-D Lookup Table** block.

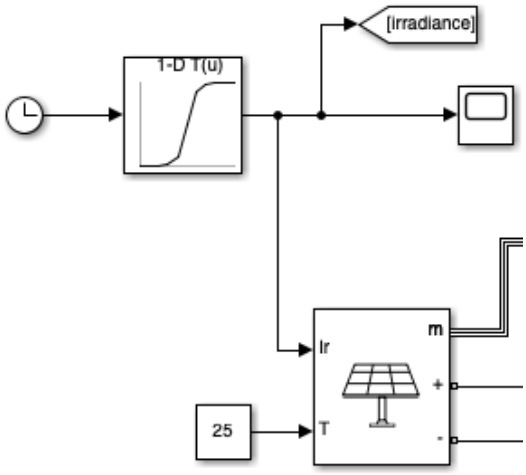

Figure 4.17: A part of PV integration where the irradiance data are exported to the modeling

To select which data to be used as input for the PV array, the parameters of the **1-D Lookup Table** need to be adjusted. In this case, we want to use the irradiance data at Honolulu. The table data's value is set to **honoluluirradiance**, which means the data in the matrix named **honoluluirradiance** in MATLAB will be brought to Simulink. The breakpoint 1's value is adjust to **1:8760** since **honoluluirradiance** contains 8760 entries (irradiance data for 365 days * 24 hours/day).

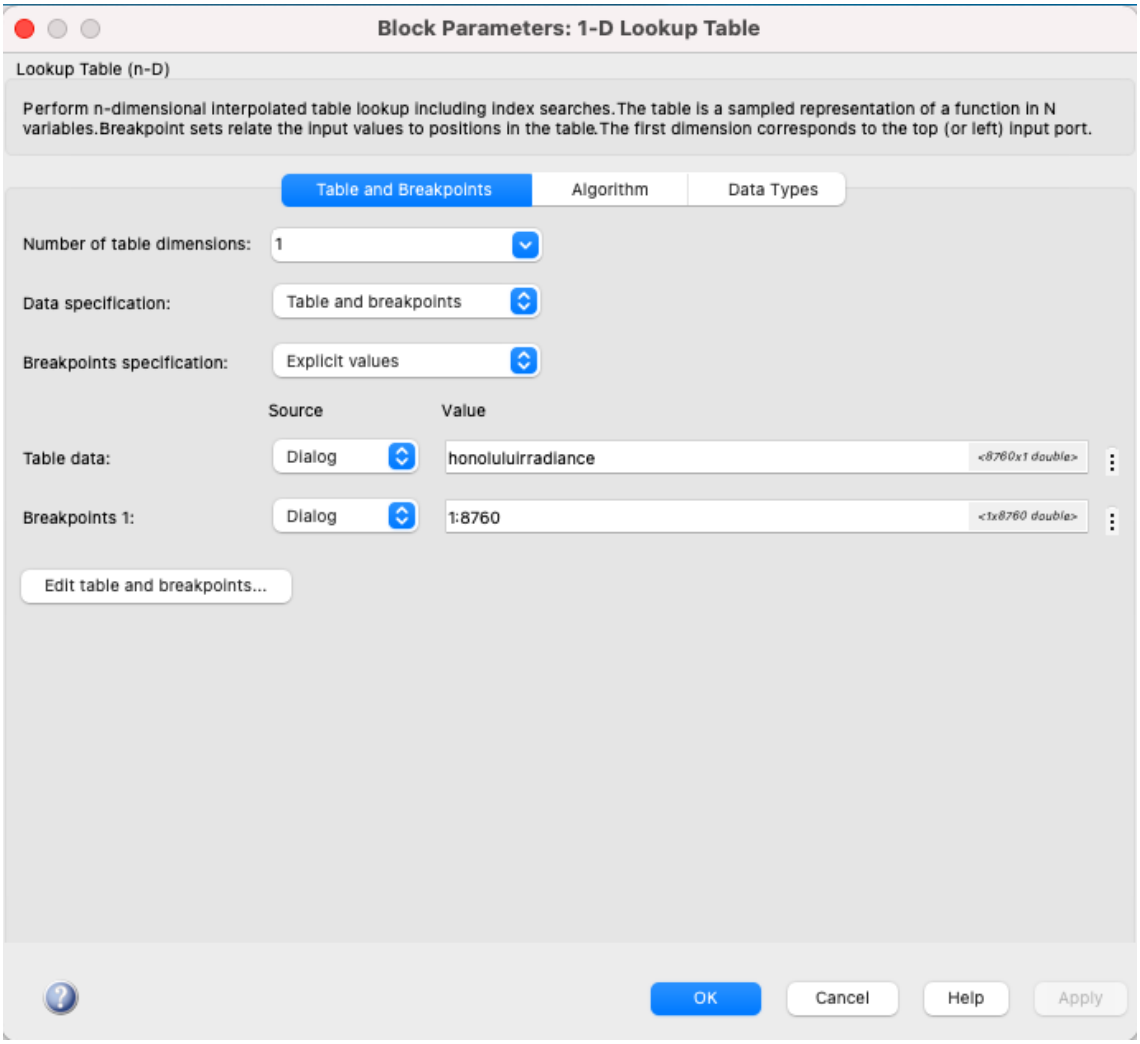

Figure 4.18: 1-D Lookup Table's parameters

Modeling the two-stage PV integration, the solar power output can be plotted over time. Since the GHI data of Honolulu are exported, the upper graph represents the solar irradiance at Honolulu, and the bottom graph is the power output corresponding to the irradiance. Both are over the 240-hour timespan.

The graphing can be modified to plot the power output at a desired timespan. Additionally, the irradiance data got from Tucker decomposition at other places can be implemented to determine the power output there. Similarly, the temperature data got from the decomposition can be applied to the PV integration.

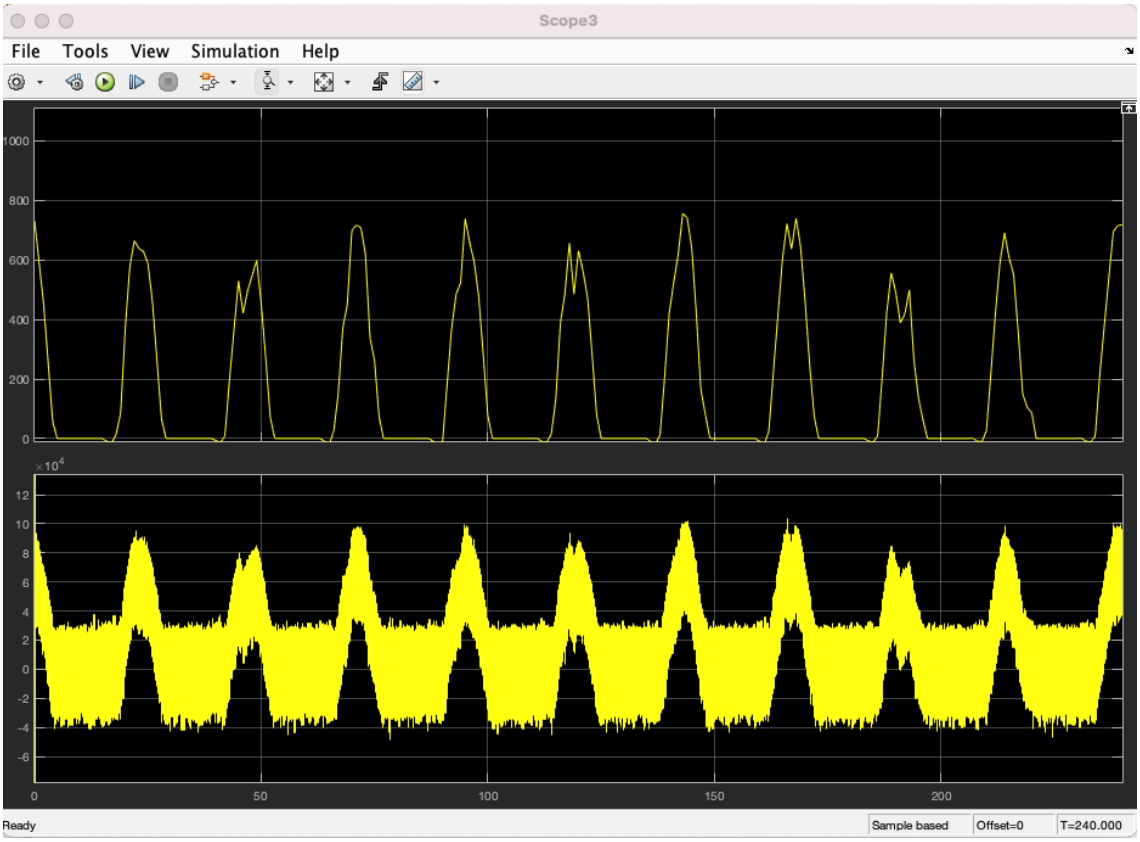

Figure 4.19: GHI and power output estimations at Honolulu over 240-hour timespan

Chapter 5

Conclusions

5.1 Conclusions

Tucker decomposition is beneficial in power system applications. Through the decomposition, we can obtain temperature and solar irradiance at different locations over time without the need to store all the original data. Tucker decomposition helps us minimize the data storage size. As discussed previously, our original tensor contains 105,120 entries. On the contrary, the core tensor only stores $5 * 2 * 3 = 30\, entries$. Factor matrix A contains $8760 * 5 = 43,800\, entries$, factor matrix B contains $2 * 2 = 4\, entries$, and factor matrix C contains $6 * 3 = 18\, entries$. The core tensor and factor matrices contain only 43,852 entries in total, and we can derive all the data from that. Moreover, the decomposition can predict missing temperatures and solar irradiance. It can also be used to forecast future temperatures and irradiance to determine the reliability of future solar power output.

Tensor Toolbox is a useful tool available in MATLAB when working with Tucker decomposition. The commands, such as, **tensor**, **tucker_als**, and **ttm** are easy to follow and provide great advantages in applications.

With the **1-D Lookup Table** block, any temperature or GHI estimations got from the core tensor operation can be used as temperature and GHI inputs for the PV array in **Simulink**. The power output over time can then be predicted corresponding to the varying temperature and irradiance.

5.2 Future Work

According to Chapter 4, the core tensor and factor matrices we obtained still have a fault in estimating the temperature at the locations with small temperature swings. Gathering data at more locations, especially those with different weather patterns, might help solve the issue.

Appendix

MATLAB file

```
data(:,:,1) = statecollege;
data(:,:,2) = sacramento;
data(:,:,3) = mesa;
data(:,:,4) = honolulu;
data(:,:,5) = lasvegas;
data(:,:,6) = newbedford;
tensorX = tensor(data);

tuckerX = tucker_als(tensorX, [5 2 3]);     %% tensor decomposition
tensorG = tuckerX.core;     %% core tensor
A = tuckerX.U{1};           %% factor matrix
B = tuckerX.U{2};           %% factor matrix
C = tuckerX.U{3};           %% factor matrix

%predict original tensor
M = ttm(tensorG,{A,B,C});
M1 = M(:,:,1);
M2 = M(:,:,2);
M3 = M(:,:,3);
M4 = M(:,:,4);
M5 = M(:,:,5);
M6 = M(:,:,6);
```

```
%convert back to matrix
statecollegeback = M1.data;
sacramentoback = M2.data;
mesaback = M3.data;
honolulubback = M4.data;
lasvegasback = M5.data;
newbedfordback = M6.data;

%%plot
%%time = linspace(0, 1, 8760);
figure(1)
plot(statecollege(:,1));
hold on
plot(statecollegeback(:,1));
title("Temperature in State College, PA");
legend("original data", "data from the decomposition");
hold off

figure(2)
plot(sacramento(:,1));
hold on
plot(sacramentoback(:,1));
title("Temperature in Sacramento, CA");
legend("original data", "data from the decomposition");
hold off
```

```
figure(3)
plot(mesa(:,1));
hold on
plot(mesaback(:,1));
title("Temperature in Mesa, AZ");
legend("original data", "data from the decomposition");
hold off

figure(4)
plot(honolulu(:,1));
hold on
plot(honolululuback(:,1));
title("Temperature in Honolulu, HI");
legend("original data", "data from the decomposition");
hold off

figure(5)
plot(lasvegas(:,1));
hold on
plot(lasvegasback(:,1));
title("Temperature in Las Vegas, NV");
legend("original data", "data from the decomposition");
hold off

figure(6)
plot(newbedford(:,1));
hold on
plot(newbedfordback(:,1));
title("Temperature in New Bedford, MA");
legend("original data", "data from the decomposition");
hold off
```

```
figure(7)
plot(statecollege(:,2));
hold on
plot(statecollegeback(:,2));
title("GHI in State College, PA");
legend("original data", "data from the decomposition");
hold off

figure(8)
plot(sacramento(:,2));
hold on
plot(sacramentoback(:,2));
title("GHI in Sacramento, CA");
legend("original data", "data from the decomposition");
hold off
```

```
figure(9)
plot(mesa(:,2));
hold on
plot(mesaback(:,2));
title("GHI in Mesa, AZ");
legend("original data", "data from the decomposition");
hold off

figure(10)
plot(honolulu(:,2));
hold on
plot(honolululback(:,2));
title("GHI in Honolulu, HI");
legend("original data", "data from the decomposition");
hold off

figure(11)
plot(lasvegas(:,2));
hold on
plot(lasvegasback(:,2));
title("GHI in Las Vegas, NV");
legend("original data", "data from the decomposition");
hold off

figure(12)
plot(newbedford(:,2));
hold on
plot(newbedfordback(:,2));
title("GHI in New Bedford, MA");
legend("original data", "data from the decomposition");
hold off
```

```
%%output to simulink @ honolulu
honoluluirradiance = honolulubback(:, 2);
```

Bibliography

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

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

[3] Yunting Liu. *EE413 Power Electronics: DC-DC Converters*. 2022.

[4] Todd Rowland and Eric W. Weisstein. Tensor. https://mathworld.wolfram.com/Tensor.html#:~:text=Tensors%20provide%20a%20natural%20and,fluid%20mechanics%2C%20and%20general%20relativity. Accessed: (01.30.2023).

[5] Denis Osipov and Joe H Chow. Pmu missing data recovery using tensor decomposition. *IEEE Transactions on Power Systems*, 35(6):4554–4563, 2020.

[6] José J Nuño-Ayón, Eduardo S Bañuelos-Cabral, Julián Sotelo-Castañón, Jorge L García-Sánchez, and José A Gutiérrez-Robles. A tensor decomposition-based approach for the analysis and visualization of ambient power system oscillations. *International Transactions on Electrical Energy Systems*, 31(12):e13177, 2021.

[7] Hongshan Zhao, Libo Ma, Xihui Yan, and Yang Zhao. Historical multi-station scada data compression of distribution management system based on tensor tucker decomposition. *IEEE Access*, 7:124390–124396, 2019.

[8] Brett W. Bader, Tamara G. Kolda, et al. Tensor toolbox for matlab, version 3.5. https://www.tensortoolbox.org/.

[9] Nsrdb: National solar radiation database. https://nsrdb.nrel.gov/data-viewer, February 2022.

[10] Sanjay Vashishtha. Differentiate between the dni, dhi and ghi? https://firstgreenconsulting.wordpress.com/2012/04/26/differentiate-between-the-dni-dhi-and-ghi/, April 2012. Accessed: (02.20.2023).

[11] M. Sengupta, Y. Xie, A. Lopez, A. Habte, Maclaurin G., and J. Shelby. These us cities are producing the most solar energy. https://www.nrel.gov/gis/solar-resource-maps.html, February 2018. Accessed: (01.25.2023).

[12] Stefan Ellerbeck. These us cities are producing the most solar energy. https://www.weforum.org/agenda/2022/05/us-solar-energy-cities-most/, May 2022. Accessed: (03.13.2023).

[13] Solar Nation. Why should you install a solar pv system in new bedford, ma? https://www.solar-nation.org/new-bedford-ma-solar-panel-installation. Accessed: (03.13.2023).

[14] Energy Systems Integration Group (ESIG). Transmission planning for 100% clean electricity. https://www.esig.energy/wp-content/uploads/2021/02/Transmission-Planning-White-Paper.pdf, 2021. Accessed: (01.14.2023).

[15] Marta Victoria, Nancy Haegel, Ian Marius Peters, Ron Sinton, Arnulf Jäger-Waldau, Carlos del Cañizo, Christian Breyer, Matthew Stocks, Andrew Blakers, Izumi Kaizuka, Keiichi Komoto, and Arno Smets. Solar photovoltaics is ready to power a sustainable future. *Joule*, 5(5):1041–1056, 2021.

[16] Cristina L. Archer and Mark Z. Jacobson. Evaluation of global wind power. *Journal of Geophysical Research: Atmospheres*, 110(D12), 2005.

[17] LEONICS. Basics of mppt solar charge controller. https://www.leonics.com/support/article2_14j/articles2_14j_en.php. Accessed: (03.14.2023).

[18] K.N. Nwaigwe, P. Mutabilwa, and E. Dintwa. An overview of solar power (pv systems) integration into electricity grids. *Materials Science for Energy Technologies*, 2(3):629–633, 2019.

[19] Modulation index and switching frequency effect on symmetric regular sampled spwm. *European Journal of Technic*, 7(2):103, 2017.