Arbitrary Mode-n Convolution of Physical Tensors with Applications in Optics

by

Pandhittaya Noikorn

A Thesis submitted to the Faculty of Graduate Studies of

The University of Manitoba

in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

University of Manitoba

Winnipeg

Abstract

Multidimension functions frequently appear in *Fourier Optics*. For example, in a polychromatic optical system, one could have three spatial dimensions for space, one dimension for time, and one dimension for wavelength. Such systems would require multi-dimension mathematical models that could be efficiently analyzed or solved using Tensor Analysis.

In *Fourier Optics*, convolution describes light propagation in multidimensional linear shift-invariant media and image formation in multidimensional linear shift-invariant imaging systems. In Tensor Analysis, many tensor operations, including convolution, are well defined in the literature. However, in this literature, tensor convolution is defined under three limiting assumptions 1) tensors to be convolved must have the same size; 2) tensors are expected to be convolved along all its dimensions; 3) tensors to be convolved should represent the same physical variables on each of its dimensions. In practice, one could possibly seek convolution along a specific subset of physical variables, which would not be well defined by this standard definition of tensor convolution.

In this thesis, to overcome these limitations inherent in the definition of tensor convolution, we defined arbitrary mode-*n* convolution that allows convolution of different size tensors along a specific subset of their physical variables. We then applied our novel arbitrary mode-*n* convolution method to simulate three simple multidimensional *Fourier Optics* problems, i.e., free space propagation, diffraction by an aperture, and imaging using a thin lens. We simulated these problems using 1) full-sized tensors; 2) Tensor Tucker Decomposition; and 3) Tensor Train Decomposition. Our numerical results demonstrated that the Tensor Train approach is most efficient in terms of accuracy, storage requirement, and computation time.

Acknowledgments

I would like to express my sincere gratitude to my academic advisor Dr. Sherif Sherif for his continuous guidance in both my research and career. His encouragement, advice, feedback, and suggestions helped me overcome many difficulties in completing this thesis. I also want to thank my examining committee, Dr. Arkady Major and Dr. Christopher Bidinosti, for their time and insightful input. My sincere gratitude to my undergraduate advisor, Dr. Waleed Mohammed, and my internship supervisor Dr. Raymond C. Rumpf, for their continuous advice and encouragement.

Furthermore, I would like to thank the University of Manitoba for supporting my graduate studies via the *University of Manitoba Graduate Fellowship* (UMGF) and Dr. Gerry Price for his support via the *Women in Engineering Scholarship*.

I would like to sincerely thank my friend Pimrapat Thanusutiyabhorn and acknowledge my late friend Dr. Wattamon Srisakuldee for their help and encouragement. I am also very grateful to my homestay mother, Alice Reimer, and her family for providing me with a comfortable home where I spent most of my research time. Finally, I sincerely thank my beloved family for their continuous support and endless love. I dedicate this thesis to my parents, who unconditionally provide their best support.

Table of Contents

Abstract ii

Acknowledgments iii

1. Introduction 1

1.1 Introduction 1
1.2 Research Motivation and Objectives 2
1.3 Thesis Contributions and Publications 3
1.4 Thesis Outline 3
1.5 Notation 3

2. Tensor Operations and Decompositions 5

2.1 Review of Matrix Operations 5
2.1.1 Kronecker product of matrices 6
2.1.2 Khatri-Rao product of matrices 6
2.1.3 Vectorization of a matrix 6
2.2 Introduction to Multidimensional Arrays 7
2.3 Vectorization of a Tensor and Multi-indices 7
2.4 The Difference Between Matrices and Second-order Tensors 8
2.5 Tensor Operations 9
2.5.1 Kronecker Product 9
2.5.2 Khatri-Rao Product 9
2.5.3 Hadamard Product 10
2.5.4 Outer Product 11
2.5.5 Inner Product and Norm 11
2.5.6 Mode-*n* Matricization of Tensors 11
2.5.7 Mode-*n* Tensor Product 12
2.5.8 Tensor Contraction 12
2.5.9 *N*-D Convolution of Tensors 13
2.5.10 Partial Mode-*n* Convolution 14
2.6 Tucker Decomposition 14
2.6.1 Hadamard Product of Tensors Using Tucker Decomposition 16

2.6.2 *N*-D Convolution of Tensors Using Tucker Decomposition 16
2.7 Tensor Networks and Tensor Train Decomposition 17
2.7.1 Tensors Network and its Graphical Representation 17
2.7.2 Hierarchical Tucker Decomposition 17
2.7.3 Tensor Train Decomposition 18
2.8 Chapter Summary 22
3. Arbitrary Mode-n Convolution of Physical Tensors 23
3.1 Motivation for Arbitrary Mode-n Convolution of Physical Tensors 23
3.1.1 Arbitrary Mode-n Convolution with Mode Expansion on Tensor Edges 24
3.1.2 Arbitrary Mode-n Convolution with Mode Expansion Independent of Their Positions 28
3.3 General Formulation of Arbitrary Mode-n Convolution of Physical Tensors 31
3.3.1 Pre-processing Physical Tensors by Artificial Mode Expansion 31
3.3.2 Select the Arbitrary Mode-*n* to Apply Convolution 32
3.3.3 Arbitrary Mode-*n* Convolution 33
3.4 Chapter Summary 34
4. Application of Arbitrary Mode-*n* Convolution in Tensor-based Optical Problems 35
4.1 Free Space Optical Propagation and Diffraction under Fresnel Approximation 35
4.1.1 Free Space Optical Propagation 35
4.1.2 Fresnel Number 36
4.1.3 Light Propagation Through a Pupil function 37
4.1.4 Simple Diffraction Model with a Pupil Function 37
4.2 Tensor-Based Formulation of a Diffraction System 39
4.2.1 Tensor-Based Formulation of a Free Space Propagation 39
4.2.2 Tensor-Based Formulation of Diffraction by an Aperture 39
4.2.3 Formulation of Diffraction by an Aperture using Tucker Decomposition 41
4.2.4 Formulation of Diffraction by an Aperture using Tensor Train Decomposition 42
4.3 Numerical Simulation Results 42
4.3.1 Simulation of 3-D Free Space Light Propagation 42
4.3.2 Simulation of Optical Diffraction by an Aperture 46
4.3.3 Optical Imaging using a Thin Lens 49
4.4 Chapter Summary 51

5. Conclusions and Future Work 52
5.1 Conclusions 52
5.2 Future Work 52
References 54
Appendix A 56
A.1 Artificially expanded tensor order 56
A.1.1 Tucker Decomposition of a Tensor with an Artificially Expanded Order 56
A.1.2 Tensor Train Decomposition with an Artificially Expanded Order 56

List of Figures

Fig. 2.1. Mode fibers of a three-dimensional tensor..... 7

Fig. 2.2. Vectorization of a matrix (above) and a 2nd-order tensor (below) in lexicographic and reverse lexicographic ordering..... 8

Fig. 2.3. Kronecker product of 3rd-order tensors..... 9

Fig. 2.4. Khatri-Rao (mode-1) product of 3rd-order tensors..... 10

Fig. 2.5. Partial mode-1 convolution of 2nd-order tensors; the convolving tensors (top); convolution on every possible combination (middle); the convolution product (bottom)..... 14

Fig. 2.6. Tucker decomposition of a three-dimensional tensor..... 14

Fig. 2.7. Examples of tensor network diagrams..... 17

Fig. 2.8. Example of HT decomposition of a 6th-order tensor..... 18

Fig. 2.9. Tensor network diagrams of (top) 4th order TT/MPS (bottom) N^{th}order TT/MPS..... 19

Fig. 2.10. The decomposition of an 8th order tensor in (a) Tucker decomposition (b) Hierarchical Tucker decomposition (c) tensor train/MPS, blue node represents 3rd or higher order tensor, and green node defines 2nd order tensor..... 20

Fig. 3.1. Network diagram of convolution on *x* of $f(u,v,x,y)$ and $h(x,y,z)$ using Tensor Train Decomposition..... 26

Fig. 3.2. Slice $\mathcal{S}[:,:,1,1,1]$ of different tensor convolutions along *x* obtained using direct computation, TKD, and TT (first row), and their differences (second row)..... 26

Fig. 3.3. Slice $\mathcal{S}[1,1,:,:,1]$ of different tensor convolutions along *x* obtained using direct computation, TKD, and TT (first row), and their differences (second row)..... 27

Fig. 3.4. Slice $\mathcal{S}[1,1,:,1,:]$ of different tensor convolutions along *x* obtained using direct computation, TKD, and TT (first row), and their differences (second row)..... 27

Fig. 3.5. Tensor train based arbitrary mode-*x* convolution on a larger problem size..... 28

Fig. 3.6. Network diagram of $f(t,u,w,x,y)$ and $h(t,v,w,x,y,z)$ in Tensor Train Decomposition and their mode-(t,x) arbitrary convolution..... 29

Fig. 3.7. Slices of tensor convolutions along *u*-*v* (top row) and *t*-*x* (bottom row) obtained using direct computation (1st column), TKD (2nd column), and TT (3rd column)..... 30

Fig. 4.1. A 3-D object imaging system with a square aperture..... 37

Fig. 4.2. A system of free space propagation of a 3-D object onto a 2-D plane of observation... 43

Fig. 4.3. Our 3-D object and its field distribution along the *x*-*z* plane……..... 43

Fig. 4.4. Planes of field distributions in the object at different *z* distances..... 44

Fig. 4.5. Observed normalized intensities at different planes using direct (1st column) Tucker Decomposition (2nd column) Tensor Train (3rd column) approaches..... 45

Fig. 4.6. Observed normalized intensities from direct (left) Tucker Decomposition (middle) Tensor Train (right) approaches..... 45

Fig. 4.7. A 3-D imaging system with an aperture..... 46

Fig. 4.8. Observed normalized intensities from different *z* distances using direct (1st column) Tucker Decomposition (2nd column) Tensor Train (3rd column) approaches..... 48

Fig. 4.9. Normalized intensity along the *x*-*z* axis at the observation region using direct (top) Tucker Decomposition (middle) Tensor Train (bottom) approaches..... 48

Fig. 4.10. Observed normalized intensities from different *z* distances using direct (1st column) Tucker Decomposition (2nd column) Tensor Train (3rd column) approaches..... 50

Fig. 4.11. Normalized intensity along the *x*-*z* axis at the observation region using direct (top) Tucker Decomposition (middle) Tensor Train (bottom) approaches..... 50

Fig. A.1. Tensor network diagrams of artificially expanded 4th-order to 5th-order TT/MPS (top) original (middle) expanded on the first and last order (bottom) expanded in between the train…….. 57

List of Tables

Table 1.1. List of mathematical notation.……..……..... 3

Table 2.1. Common matrix products as tensor contractions ……...……..... 13

Table 2.2. Comparison of storage complexity and tensor format of a N^{th}-order tensor with I dimensions and R rank……. 20

Table 4.1. Comparison of free space propagation using full-tensor, Tucker Decomposition, and Tensor Train approaches..... 46

Table 4.2. Computation time and error from different approaches……..... 49

1. Introduction

1.1 Introduction

Multidimensional functions and signals arise naturally in both forward and inverse optics problems. The relationship between the information capacity of an optical system and its degrees of freedom, i.e., the number of its temporal, spectral, and spatial variables, is discussed in [1]. Computations in such high-dimensional problems could involve impractical computer storage requirements, as the number of variables typically scales exponentially with problem size. For instance, representing an N-dimensional problem with I samples in each dimension would result in an array having I^N elements. This issue is often referred to as the *curse of dimensionality* [2]. The use of projection operations and sparse representations could possibly reduce computer storage requirements, but they could be limited in their applications.

The study of multidimensional arrays known as tensors has been a topic of interest since the 19th century. Tensors and their different mathematical models could enable the practical analysis of N dimensional problems. Known applications of tensors are in data analysis [3-4, 15-16], psychometrics, quantum mechanics, and signal and image processing [5-6, 14,17]. An important term used in tensors is *order*; an N-dimensional array is referred to as an N^{th}-order tensor. Another important term is a *mode* which refers to a particular dimension of an N-dimensional array. Since tensors could be inherently very large arrays, many tensor decompositions have been developed. The most basic tensor decompositions are Canonical Polyadic (CP) and Tucker Decomposition [7-8]. The CP format represents an arbitrary tensor as a finite linear combination of outer products of rank-1 tensors (vectors).

The Tucker Decomposition format factorizes an N^{th} order tensor into an N^{th} order tensor core (possibly of smaller size) and N factor matrices. These factor matrices are essentially linear transformations applied to the tensor core. Therefore, Tucker Decomposition could be thought of as a multilinear transformation of the core tensor.

Another family of tensor decompositions, known as tensor networks, was developed to further improve Tucker Decomposition's ability to address the curse of dimensionality. These tensor networks factorize higher-order tensors into a sequence of lower-order tensors. The most

fundamental tensor network is the Hierarchical Tucker Decomposition, which recursively decomposes higher-order tensors in a binary tree of lower-order tensors [9].

A particularly simple case of the Hierarchical Tucker Decomposition, one that avoids the need for a recursive algorithm to obtain it, is the Tensor Train Decomposition that was introduced in [10]. This Tensor Train Decomposition decomposes a high-order tensor into a train-like structure (a simple binary tree) consisting of a sequence of 3^{rd}-order tensors. Due to its simplicity and efficiency, this Tensor Train Decomposition has been widely used recently.

1.2 Research Motivation and Objectives

In Physical Optics, the behavior of both electric and magnetic optical fields is described by Maxwell's equations. In *Fourier Optics*, the behavior of scalar electric fields is described using linear operators, mainly Hadamard (element-wise) products and convolutions. However, Fourier Optics formulations of optical problems could become impractical for large dimension and/or large size problems, as their multidimensional arrays could become very large (computational difficulties), and their required multidimensional convolutions could become quite cumbersome (analysis difficulties).

One possible way to overcome these practical difficulties is to formulate *Fourier Optics* in Terms of tensors. However, one important obstacle to developing such *Tensor Optics* is the definition of tensor convolution in the literature on tensors. In this literature, the convolution of any two tensors is well defined, but 1) tensors to be convolved are assumed to have the same order; 2) corresponding modes of tensors to be convolved are assumed to represent the same physical variables; and 3) convolution along every tensor mode (physical dimension) is assumed.

In this thesis, to overcome the above limitations of this definition of tensor convolution, as found in the literature on tensors, we generalize it to allow arbitrary mode-n convolution of physical tensors instead of abstract multidimensional arrays. We also develop this new arbitrary mode-n convolution further to incorporate tensors in both Tucker Decomposition and Tensor Train representations. Finally, to demonstrate the value of our arbitrary mode-n convolution, we apply it to simulate three simple *Fourier Optics* problems, i.e., free space propagation, diffraction by an aperture, and imaging using a thin lens as a first step in developing *Tensor Optics*.

1.3 Thesis Contributions and Publications

- Generalized current tensor convolution of abstract multidimensional arrays to allow arbitrary mode-n convolution of physical tensors.
- Developed arbitrary mode-n convolution of physical tensors further to include Tucker Decomposition and Tensor Train representations.
- Applied our novel arbitrary mode-n convolution to simulate three simple *Fourier Optics* problems as a first step in developing *Tensor Optics*
- A manuscript is being prepared for publication in the *J. of Optical Society of America – A*

1.4 Thesis Outline

This thesis is structured as follows:

- Chapter 2: Review of tensors concepts, operations, and decompositions
- Chapter 3: Presents our approach for arbitrary mode-*n* convolution of physical tensors, including numerical examples
- Chapter 4: Application of our arbitrary mode-*n* Convolution to simulate three simple *Fourier Optics* problems
- Chapter 5: Conclusions and suggested future work

1.5 Notation

The mathematical notation used in this thesis is listed in the table below.

Notation	Description	Representation
x, y, a, b	Italic	Scalar
$\mathbf{x}, \mathbf{y}, \mathbf{a}, \mathbf{b}$	Bold small letter	Vector
$\mathbf{X}, \mathbf{Y}, \mathbf{A}, \mathbf{B}$	Bold capital letter	Matrix
$\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}$	Script capital letter	Tensor
$\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}$	Bold script letter	Artificially expanded order tensor
$\mathcal{G}$		Tucker core tensor

Notation	Description	Representation
$\mathbf{X}_n$	Matrix with subscript	Mode-*n* matricization of $\mathcal{X}$
$\mathrm{vec}(\cdot)$	Function vec()	Vectorization
$I_1, \dots, I_N$ and $J_1, \dots, J_N$	I and J	Tensor physical dimensions
$R_1, \dots, R_N$ and $Q_1, \dots, Q_N$	R and Q	Tensor ranks
$\overline{\imath_1 \imath_2 \dots \imath_N}$	Indices with bar	Multi-index
$*$ and $*_n$	Asterisk	Full and mode-*n* convolution
$\circledast, \otimes$	Operators	Hadamard and Kronecker product
$\odot_n$		Mode-*n* Khatri-Rao product
$\boxdot_n$		Partial mode-*n* convolution

Table 1.1. List of mathematical notation

2. Tensor Operations and Decompositions

In this chapter, we review basic tensor definitions, operations and decompositions. It will serve as prerequisite information for the following chapters.

Given a linear equation

$$a_{i_1}x_1 + a_{i_2}x_2 + \cdots + a_{i_N}x_N = b_i \tag{2.1}$$

If we have multiple equations with the same set of variables x_i, we could represent them as a linear equation system using vectors and a matrix as

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{N,1} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}$$

$$\mathbf{Ax} = \mathbf{b}. \tag{2.2}$$

The above matrix equation only has **x** as an unknown vector. We will refer to equation 2.1 as a one-dimensional problem. If the unknown variable is a matrix **X** instead, we could represent Eq. 2.2 as

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & \ddots & \vdots \\ a_{N,1} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} x_{1,1} & \cdots & x_{1,N} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,N} \end{bmatrix} \begin{bmatrix} c_{1,1} & \cdots & c_{1,N} \\ \vdots & \ddots & \vdots \\ c_{N,1} & \cdots & c_{N,N} \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} b_{1,1} & \cdots & b_{1,N} \\ \vdots & \ddots & \vdots \\ b_{N,1} & \cdots & b_{N,N} \end{bmatrix},$$

$$\mathbf{AXC}^{\mathrm{T}} = \mathbf{B} \tag{2.3}$$

where every row of matrix **C** are weight elements for the corresponding row of the unknown matrix **X**. We note that extending the above linear equations, where the unknowns are either vector (one dimension) or a matrix (two dimensions), to three or higher dimensions would be challenging. Therefore, we seek a generalized mathematical model for *N*-dimensional linear problems that are represented with linear equations.

2.1 Review of Matrix Operations

In this section, we will review matrix operations that will be helpful in understanding the following sections.

2.1.1 Kronecker product of matrices

The Kronecker product is commonly used with matrices. Let $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$ and $\in \mathbb{R}^{J_1 \times J_2}$, their Kronecker product is defined as

$$\mathbf{Z} = \mathbf{X} \otimes \mathbf{Y} = \begin{bmatrix} x_{1,1}\mathbf{Y} & \cdots & x_{1,I_2}\mathbf{Y} \\ \vdots & \ddots & \vdots \\ x_{I_1,1}\mathbf{Y} & \cdots & x_{I_1,I_2}\mathbf{Y} \end{bmatrix} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2}. \tag{2.4}$$

2.1.2 Khatri-Rao product of matrices

The Khatri-Rao product of matrices could be viewed as a particular case of their Kronecker product, where the Kronecker product is only performed on a specific matrix partition. Typically, the Khatri-Rao product is performed on columns. Given $\mathbf{X} \in \mathbb{R}^{I_1 \times K}$ with columns $\mathbf{x}_i$ and $\mathbf{Y} \in \mathbb{R}^{J_1 \times K}$ with columns $\mathbf{y}_i$, their Khatri-Rao product is defined as

$$\mathbf{Z} = \mathbf{X} \odot \mathbf{Y} = [\mathbf{x}_1 \otimes \mathbf{y}_1 \quad \cdots \quad \mathbf{x}_K \otimes \mathbf{y}_K] \in \mathbb{R}^{I_1 J_1 \times K}. \tag{2.5}$$

2.1.3 Vectorization of a matrix

One way of solving Eq. 2.3 is to convert to a one-dimensional problem. By vectorizing **X** and **B**, Eq. 2.3 could be rewritten as

$$(\mathbf{C} \otimes \mathbf{A})\mathrm{vec}(\mathbf{X}) = \mathrm{vec}(\mathbf{B}) \tag{2.6}$$

where $\otimes$ is the Kronecker product of matrices. This matrix vectorization is performed by stacking its rows into a single vector. Therefore, the vectorization of **X** is given by

$$\mathrm{vec}(\mathbf{X}) = [x_{1,1} \quad \cdots \quad x_{1,N} \quad x_{2,1} \quad \cdots \quad x_{N,1} \quad \cdots \quad x_{N,N}]^{\mathrm{T}}. \tag{2.7}$$

We note that instead of using i_1, i_2 to refer to an element in vec(**X)**, we could use a single index $i = i_2 + (i_1 - 1)N$. Similarly, **B** could be vectorized in the same way.

When describing three dimensional (or higher-dimensional) linear equations with multidimensional arrays, one could still use similar vectorizations to transform the higher dimensional problem to a one-dimensional problem. However, this vectorization-based approach could result in extremely large one-dimensional problems.

2.2 Introduction to Multidimensional Arrays

A tensor is a multidimensional array. Tensors play significant roles in Signal Processing, Machine Learning, Computer Vision, and many more applications. An *N*-D tensor is an *N*-D array, where *N* is known as the *order* of the tensor. Therefore, a zero-order tensor is a scalar, a first-order tensor is a vector, and a second-order tensor is a matrix. Another important concept in tensor analysis is the concept of a tensor *mode*, i.e., a particular physical dimension of a tensor.

2.3 Vectorization of a Tensor and Multi-indices

To represent scalar tensor elements we use different indices to denote each mode. An *N*-order tensor would have *N* indices, $i_1, i_2 \cdots, i_N$. By fixing all indices except one, we could obtain different tensor fibers. For example, a matrix has two modes, rows (mode-2 fibers) and columns (mode-1 fibers), and an *N*-D array has *N* mode *fibers* where mode-1 fibers are columns, mode-2 fibers are rows, *etc*. Also, by specifying all modes of a tensor except two, we could obtain different tensor slices. Different tensor modes and fibers are shown in Fig. 2.1.

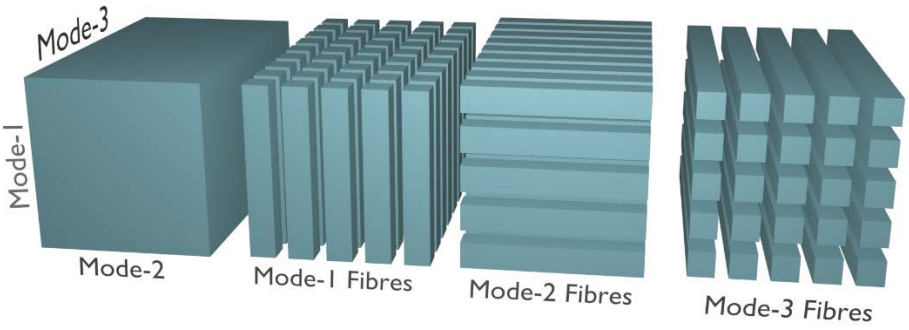

Fig. 2.1. Mode fibers of a three-dimensional tensor

Any *N*-D tensor can be reshaped into a vector. Such vectorization only requires a single index called a multi-index. This multi-index could be defined in two different ways [11-12], where little-endian (reverse lexicographic ordering) vectorization defines it as

$$\overline{\imath_1 \imath_2 \cdots \imath_{N-1} \imath_N} = i_1 + (i_2 - 1)I_1 + (i_3 - 1)I_1 I_2 + \cdots + (i_N - 1)I_1 \cdots I_{N-1}, \tag{2.8}$$

and big-endian (lexicographic ordering) defines it as

$$\overline{\imath_1 \imath_2 \cdots \imath_{N-1} \imath_N} = i_N + (i_{N-1} - 1)I_N + (i_{N-2} - 1)I_N I_{N-1} + \cdots + (i_1 - 1)I_2 \cdots I_N. \tag{2.9}$$

In this thesis, we will use big-endian notation in all mathematical expressions, and, since most computer languages store data in reverse lexicographic ordering, we will use little-endian notation in our computer code (see appendix A). We note that the multi-index used to vectorize Eq. 2.6 follows the big-endian notation.

Also, multi-indices are directly related to the Kronecker product. Using big-endian notation, one could verify that $(\mathbf{C}\otimes\mathbf{A})_{\overline{i_1 J_1},\overline{i_2 J_2}} = c_{i_1,i_2} a_{j_1,j_2}$ in Eq. 2.4. On the other hand, using little-endian notation, $(\mathbf{C}\otimes\mathbf{A})_{\overline{i_1 J_1},\overline{i_2 J_2}} = c_{i_1,i_2} a_{j_1,j_2} = (\mathbf{A}\otimes_{\mathrm{L}}\mathbf{C})$ where $\otimes_{\mathrm{L}}$ is the Left Kronecker product. For $\mathbf{X} \in \mathbb{R}^{I_1\times I_2}$ and $\mathbf{Y} \in \mathbb{R}^{J_1\times J_2}$ the Left Kronecker product is defined as

$$\mathbf{X}\otimes_{\mathrm{L}}\mathbf{Y} = \begin{bmatrix} \mathbf{X}y_{1,1} & \cdots & \mathbf{X}y_{1,I_2} \\ \vdots & \ddots & \vdots \\ \mathbf{X}y_{I_1,1} & \cdots & \mathbf{X}y_{I_1,I_2} \end{bmatrix} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2}. \tag{2.10}$$

2.4 The Difference Between Matrices and Second-order Tensors

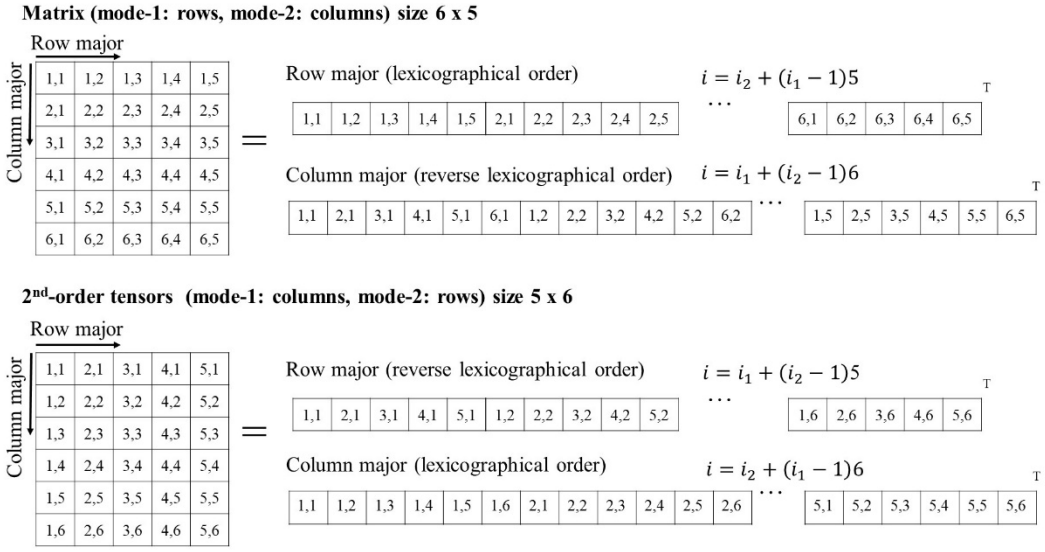

Fig. 2.2. Vectorization of a matrix (above) and a 2nd-order tensor (below) in lexicographic and reverse lexicographic ordering

Matrices and 2nd-order tensors are not always viewed as the same mathematical entities. Matrices always denote their rows as mode-1 fibers and their columns as mode-2 fibers (see Fig. 2.2). On the other hand, using big-endian notation, tensors' mode-1 fibers are their columns, and mode-2 fibers are their rows.

2.5 Tensor Operations

In this section, we describe common tensor operations that will be needed to understand the following sections. Further reading on other tensor operations could be found in [13].

2.5.1 Kronecker Product

As mentioned above, multi-indices (big-endian or little-endian) could be used to obtain the Kronecker product. Therefore, the Kronecker product of any two tensors having the same order $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ is

$$\mathcal{Z} = \mathcal{X} \otimes \mathcal{Y} \in \mathbb{R}^{I_1 J_1 \times I_2 J_2 \times \cdots \times I_N J_N} \tag{2.11}$$

where its scalar entries are given by

$$z_{\overline{i_1 j_1}, \cdots, \overline{i_N j_N}} = x_{i_1, \cdots, i_N} y_{j_1, \cdots, j_N}. \tag{2.12}$$

An example of Kronecker Product of 3rd-order tensors is shown in Fig. 2.3. Only a small cube of the first tensor represents is shown, while the second tensor is shown as a full tensor.

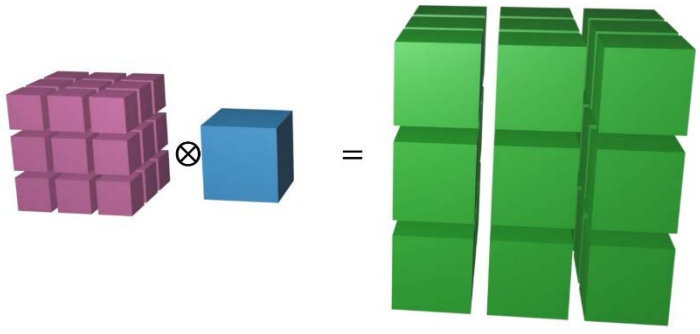

Fig. 2.3. Kronecker product of 3rd-order tensors

2.5.2 Khatri-Rao Product

The mode-*n* Khatri-Rao product of two tensors is defined as their Kronecker product involving all modes, except mode-*n*. This definition requires that mode-*n* fibers of both tensors should have the

same dimension. Therefore, the mode-*n* Khatri-Rao of two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times K \times I_{n+1} \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times \cdots \times J_{n-1} \times K \times J_{n+1} \times \cdots \times J_N}$ is defined as

$$\mathcal{Z} = \mathcal{X} \odot_n \mathcal{Y} \in \mathbb{R}^{I_1 J_1 \times \cdots \times I_{n-1} J_{n-1} \times K \times I_{n+1} J_{n+1} \times \cdots \times I_N J_N} \tag{2.13}$$

where

$$\mathcal{Z}(:, \cdots, :, k, :, \cdots, :) = \mathcal{X}(:, \cdots, :, k, :, \cdots, :) \otimes \mathcal{Y}(:, \cdots, :, k, :, \cdots, :). \tag{2.14}$$

An example of Khatri-Rao (mode-1) product of 3rd-order tensors is shown in Fig. 2.4. A small cube of the first tensor represents is shown, while the second tensor is shown as three slices along modes 2 and 3.

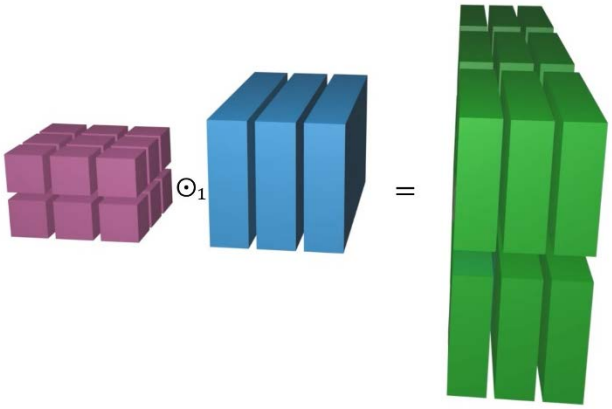

Fig. 2.4. Khatri-Rao (mode-1) product of 3rd-order tensors

2.5.3 Hadamard Product

The Hadamard product, i.e., elementwise product, requires that both tensors should be of equal order and equal dimension along each mode, i.e., $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. Their Hadamard product is defined as

$$\mathcal{Z} = \mathcal{X} \circledast \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \tag{2.15}$$

where its scalar entries are given by

$$z_{i_1, \cdots, i_N} = x_{i_1, \cdots, i_N} y_{i_1, \cdots, i_N}. \tag{2.16}$$

2.5.4 Outer Product

Given two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times \cdots \times J_M}$ their outer product is an $(N+M)$ order tensor,

$$\mathcal{Z} = \mathcal{X} \circ \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times J_1 \times \cdots \times J_M} \tag{2.17}$$

with entries

$$z_{i_1,\cdots,i_N,j_1,\cdots,j_M} = x_{i_1,\cdots,i_N} y_{j_1,\cdots,j_M}. \tag{2.18}$$

2.5.5 Inner Product and Norm

The inner product of two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ having the same order and dimensions is defined as

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \mathrm{vec}(\mathcal{X})^{\mathrm{T}} \mathrm{vec}(\mathcal{Y}) = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i=1}^{I_N} x_{i_1,\cdots,i_N} y_{i_1,\cdots,i_N} \quad . \tag{2.19}$$

The norm of a tensor $\mathcal{X}$ is defined as $\|\mathcal{X}\| = \langle \mathcal{X}, \mathcal{X} \rangle^{1/2}$.

2.5.6 Mode-*n* Matricization of Tensors

Sometimes we need to reshape a tensor into a matrix. This process is called matricization, unfolding, or flattening. The flattening of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ into a matrix $\mathbf{X}_{\langle n \rangle} \in \mathbb{R}^{I_1 \cdots I_n \times I_{n+1} \cdots I_N}$ is called its mode-n canonical matricization, where the rows and columns of this matrix involve tensor dimensions $I_1 \cdots I_n$ and $I_{n+1} \cdots I_N$, respectively. Formally, mode-n canonical matricization, $\mathbf{X}_{\langle n \rangle}$, is defined as

$$\left(\mathbf{X}_{\langle n \rangle}\right)_{\overline{i_1 \cdots i_n}, \overline{i_{n+1} \cdots i_N}} = x_{i_1,\cdots,i_N}. \tag{2.20}$$

A special case of this mode-*n* canonical matricization is the mode-*n* matricization, where a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is flattened to $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$ defined as

$$\left(\mathbf{X}_{(n)}\right)_{i_n, \overline{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N}} = x_{i_1,\cdots,i_N} \tag{2.21}$$

Let i_n be an index of I_n and j be the multi-index corresponding to $I_1 \cdots I_{n-1} I_{n+1} \cdots I_N$ then the column index j of the resulting matrix is given by (big-endian notation)

$$j = 1 + \sum_{p=N;p\neq n}^{1} (i_p - 1)\, I_N \cdots I_{n+1} I_{n-1} \cdots I_p \quad . \tag{2.22}$$

Similarly, using little-endian notation [14-15]

$$j = 1 + \sum_{p=1;p\neq n}^{N} (i_p - 1)\, I_1 \cdots I_{n-1} I_{n+1} \cdots I_p \quad . \tag{2.23}$$

2.5.7 Mode-*n* Tensor Product

The mode-*n* tensor product is a linear transformation, where any tensor mode-*n* fibers are linearly transformed (multiplied) by a matrix. Given tensor $\mathcal{X}$ and matrix $\mathbf{A}^{(n)}$, their mode-*n* tensor product could be obtained from

$$y_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1,\ldots,i_n,\ldots,i_N}\, a_{j\times i_n} \quad . \tag{2.24}$$

or by unfolding tensor $\mathcal{X}$ into a mode-n matrix, we could conveniently obtain the required mode-n tensor product as

$$\mathbf{Y}_{(n)} = \mathbf{A}^{(n)}\mathbf{X}_{(n)}. \tag{2.25}$$

Given a tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and matrices $\mathbf{A}^{(1)},\cdots,\mathbf{A}^{(N)}$ where $\mathbf{A}^{(n)} \in \mathbb{R}^{J^{(n)}\times I_n}$, a full multilinear tensor product $\mathcal{Y} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ could be defined as

$$\begin{aligned}\mathcal{Y} &= \mathcal{X} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} \\ &= [\![\mathcal{X}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N)}]\!].\end{aligned} \tag{2.26}$$

This full multilinear tensor product is closely related to the Tucker Decomposition, described in Section 2.6.

2.5.8 Tensor Contraction

Tensor contraction is a generalization of the mode-*n* tensor product. Instead of defining a product involving mode-*n* of one tensor, tensor contraction [11,13] allows us to define a product on two modes $\binom{m}{n}$ each belonging to one of the two tensors. Given tensors $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1\times J_2\times\cdots\times J_M}$ with modes n and m having the same dimensions $I_n = J_m$, the mode-$\binom{m}{n}$ product of these tensors is written as

$$\mathcal{Z} = \mathcal{X} \times_n^m \mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N \times J_1 \times \cdots \times J_{m-1} \times J_{m+1} \times \cdots \times J_M}, \quad (2.27)$$

in which $\mathcal{Z}$ is a tensor of order $(N + M - 2)$ with entries

$$z_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N,j_1,\ldots,j_{m-1},j_{m+1},\ldots,j_M} = \sum_{i_n=1}^{I_n} x_{i_1,\ldots,i_{n-1},i_n,i_{n+1},\ldots,i_N} \, y_{j_1,\ldots,j_{m-1},i_n,j_{m+1},\ldots,j_M}. \quad (2.28)$$

In other words, tensor mode-$\binom{m}{n}$ product could be viewed as an outer product with an inner product along modes m and n. Mode-n tensor product is a particular case of tensor contraction where $\times_n$ has the same meaning as $\times_n^1$. Therefore Eq. 2.26 can be written as

$$\mathcal{Y} = \mathcal{X} \times_1^1 \mathbf{A}^{(1)} \times_2^1 \mathbf{A}^{(2)} \cdots \times_N^1 \mathbf{A}^{(N)} \quad (2.29)$$

We could write common matrix products as mode-$\binom{m}{n}$ tensor products as shown in Table 2.1.

Matrix-Vector forms	Tensor Contraction forms
$\mathbf{XY}$	$\mathbf{X} \times_2^1 \mathbf{Y}$
$\mathbf{XY^T}$	$\mathbf{X} \times_2^2 \mathbf{Y}$
$\langle \mathbf{X}, \mathbf{Y} \rangle$	$\mathbf{X} \times_{1,2}^{1,2} \mathbf{Y}$
$\mathbf{Xy}$	$\mathbf{X} \times_2^1 \mathbf{Y}$

Table 2.1. Common matrix products as tensor contractions

2.5.9 *N*-D Convolution of Tensors

It is common to seek the convolution of two multidimension functions along all of their dimensions, for example, in three dimension

$$s(x,y,z) = f(x,y,z) *_{x,y,z} h(x,y,z)$$
$$= \sum_{x'} \sum_{y'} \sum_{z'} h(x',y',z') f(x-x', y-y', z-z'). \quad (2.30)$$

Writing the above equation in tensor form yields

$$\mathcal{S}(x,y,z) = \sum_{x'} \sum_{y'} \sum_{z'} \mathcal{H}(x',y',z') \mathcal{F}(x-x', y-y', z-z'). \quad (2.31)$$

2.5.10 Partial Mode-*n* Convolution

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ be two tensors with the same order *N*. If one is interested in obtaining convolutions between every combination of their mode-1 fibers, i.e., $\mathcal{Z} = \mathcal{X} \boxdot_1 \mathcal{Y} \in \mathbb{R}^{(I_1+J_1-1) \times I_2 J_2 \times \cdots \times I_N J_N}$, then this partial mode-1 convolution of the two tensors is given by $\mathcal{Z}(:, \overline{i_2 J_2}, \cdots, \overline{i_N J_N}) = \mathcal{X}(:, i_2, \cdots, i_N) * \mathcal{Y}(:, j_2, \cdots, j_N)$ where $\overline{i_n J_n}$ denotes the multi-index. This definition of partial convolution is useful when computing the full *N*-D tensor convolution using Tucker Decomposition and Tensor Train representations.

An example of partial mode-1 convolution of 2nd-order tensors is shown in Fig. 2.5 where tensors are represented as mode-1 fibres.

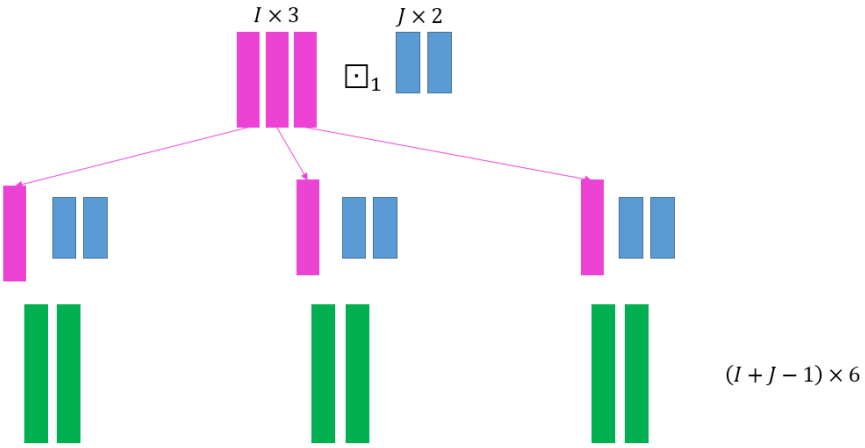

Fig. 2.5. Partial mode-1 convolution of 2nd-order tensors; the convolved tensors (top); convolution on every possible combination (middle); the convolution product (bottom)

2.6 Tucker Decomposition

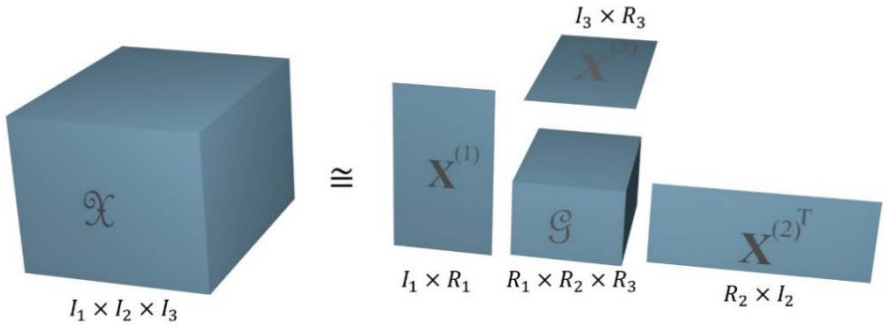

Fig. 2.6. Tucker decomposition of a three-dimensional tensor

Tensor Tucker Decomposition (TKD) [8-9] decomposes a tensor of order *N* into a core tensor (usually smaller than the original tensor) and *N* factor matrices. Fig. 2.6 shows a 3rd-order tensor in the Tucker Decomposition representation. TKD could also be viewed as a full multilinear transformation of the core tensor, as described by Eq. 2.26. Mathematically, an *N*-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ could be decomposed into a (possibly smaller) core tensor $\mathcal{G}_{\mathcal{X}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and factor matrices $\mathbf{X}^{(1)}, \cdots, \mathbf{X}^{(N)}$ where $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. This TKD of tensor $\mathcal{X}$ could be written as

$$\begin{aligned} \mathcal{X} &\cong \mathcal{G}_{\mathcal{X}} \times_1 \mathbf{X}^{(1)} \times_2 \mathbf{X}^{(2)} \cdots \times_N \mathbf{X}^{(\mathrm{N})} \\ &= [\![\mathcal{G}_{\mathcal{X}}; \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(N)}]\!]. \end{aligned} \tag{2.32}$$

As mentioned earlier, one could view TKD as a multilinear transform of the core tensor by the factor matrices. TKD could also be viewed as a method of tensor compression because when $R_n < I_n$ TKD would decompose the original tensor, in addition to the factor matrices, into a smaller tensor core. The values R_1, R_2, ... R_N are known as the original tensor's multilinear rank. The most popular way to obtain TKD is by using the Alternating Least Squares (ALS) [16] algorithm.

The TKD of a tensor $\mathcal{X}$ can also be represented as the sum of outer products of a set of vectors

$$\mathcal{X} \cong \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 \cdots r_N} \left(\mathbf{x}_{r_1}^{(1)} \circ \mathbf{x}_{r_2}^{(2)} \circ \cdots \circ \mathbf{x}_{r_N}^{(N)} \right) \quad . \tag{2.33}$$

where $\mathbf{x}_{r_n}^{(n)}$ are the columns of the factor matrix $\mathbf{X}^{(n)} = \left[\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, \ldots, \mathbf{x}_{R_n}^{(n)} \right]$. Other forms of TKD using mode-n matricized and vectorized tensor forms could be written as

$$\mathbf{X}_{(n)} = \mathbf{X}^{(n)} \mathbf{G}_{(n)} \left(\mathbf{X}^{(N)} \otimes \cdots \otimes \mathbf{X}^{(n-1)} \otimes \mathbf{X}^{(n+1)} \otimes \cdots \otimes \mathbf{X}^{(1)} \right)^{\mathrm{T}}, \tag{2.34}$$

$$\mathrm{vec}(\mathcal{X}) = (\mathbf{X}^{(N)} \otimes \cdots \otimes \mathbf{X}^{(1)}) \mathrm{vec}(\mathcal{G}). \tag{2.35}$$

These equations are the multidimensional form of Eq. 2.2 and Eq. 2.4. Typically, the storage requires to store an *N*-dimensional tensor is $\mathcal{O}(I^N)$ with TKD it reduces to $\mathcal{O}(NIR + R^N)$ [7] where I and R are tensor physical and rank dimensions.

2.6.1 Hadamard Product of Tensors Using Tucker Decomposition

The Hadamard, elementwise, product requires the two tensors to have the same order and dimensions along all of its modes. If we have two tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ that have Tucker decompositions, $\mathcal{X} = [\![\mathcal{G}_\mathcal{X}; \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(N)}]\!]$ and $\mathcal{Y} = [\![\mathcal{G}_\mathcal{Y}; \mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \cdots, \mathbf{Y}^{(N)}]\!]$. Their Hadamard product is obtained by

$$\mathcal{X} \circledast \mathcal{Y} = [\![\mathcal{G}_\mathcal{X} \otimes \mathcal{G}_\mathcal{Y}; \mathbf{X}^{(1)} \odot_1 \mathbf{Y}^{(1)}, \mathbf{X}^{(2)} \odot_1 \mathbf{Y}^{(2)}, \cdots, \mathbf{X}^{(N)} \odot_1 \mathbf{Y}^{(N)}]\!]. \tag{2.36}$$

The operator $\otimes$ is the Kronecker product and $\odot_1$ is the mode-1 Khatri-Rao product. This Khatri-Rao product requires that its two matrices have the same number of rows.

2.6.2 *N*-D Convolution of Tensors Using Tucker Decomposition

In this section, we use an example of convolving three-dimensional functions $f(x, y, z)$ and $h(x, y, z)$ to demonstrate *N*-D convolution of tensors using Tucker decomposition. Given the Tensor Tucker decompositions of these functions

$$\mathcal{F} = [\![\mathcal{G}_\mathcal{F}; \mathbf{X}_\mathcal{F}, \mathbf{Y}_\mathcal{F}, \mathbf{Z}_\mathcal{F}]\!] \text{ and } \mathcal{H} = [\![\mathcal{G}_\mathcal{H}; \mathbf{X}_\mathcal{H}, \mathbf{Y}_\mathcal{H}, \mathbf{Z}_\mathcal{H}]\!] \tag{2.37}$$

with $\mathcal{F} \in \mathbb{R}^{I_x \times I_y \times I_z}, \mathcal{G}_\mathcal{F} \in \mathbb{R}^{R_x \times R_y \times R_z}$ and $\mathcal{H} \in \mathbb{R}^{J_x \times J_y \times J_z}, \mathcal{G}_\mathcal{H} \in \mathbb{R}^{Q_x \times Q_y \times Q_z}$. The *N*-D Convolution of the two functions in TKD representation is given by

$$\mathcal{F} * \mathcal{H} = [\![\mathcal{G}_\mathcal{F} \otimes \mathcal{G}_\mathcal{H}; \mathbf{X}_\mathcal{F} \boxdot_1 \mathbf{X}_\mathcal{H}, \mathbf{Y}_\mathcal{F} \boxdot_1 \mathbf{Y}_\mathcal{H}, \mathbf{Z}_\mathcal{F} \boxdot_1 \mathbf{Z}_\mathcal{H}]\!]. \tag{2.38}$$

where $\boxdot_1$ is partial mode-1 convolution defined in Section 2.5.10.

In general, if we have two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ that are represented as TKD

$$\mathcal{X} = [\![\mathcal{G}_\mathcal{X}; \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(N)}]\!] \text{ and } \mathcal{Y} = [\![\mathcal{G}_\mathcal{Y}; \mathbf{Y}^{(1)}, \mathbf{Y}^{(2)}, \cdots, \mathbf{Y}^{(N)}]\!]. \tag{2.39}$$

A full *N*-D convolution of these two tensors is obtained by obtaining the Kronecker product of the cores, and the partial mode-1 convolution of the corresponding factor matrices. Therefore,

$$\mathcal{X} * \mathcal{Y} = [\![\mathcal{G}_\mathcal{X} \otimes \mathcal{G}_\mathcal{Y}; \mathbf{X}^{(1)} \boxdot_1 \mathbf{Y}^{(1)}, \mathbf{X}^{(2)} \boxdot_1 \mathbf{Y}^{(2)}, \cdots, \mathbf{X}^{(N)} \boxdot_1 \mathbf{Y}^{(N)}]\!]. \tag{2.40}$$

2.7 Tensor Networks and Tensor Train Decomposition

In this thesis, we will focus on two tensor decompositions: Tucker Decomposition and Tensor Train. The Tensor Train decomposition is the simplest case of Hierarchical Tucker (HT) decompositions, which in turn are a simpler case of tensor network decompositions.

2.7.1 Tensors Network and its Graphical Representation

Similar to Tucker Decomposition, the objective of a Tensor Network (TN) is to decompose a higher-order tensor into a network of lower-order tensors. The difference between TKD and TN is that TKD has one core tensor while TN has multiple lower-order core tensors.

Multidimensional arrays are hard to visualize; hence, we could use a *tensor network diagram* to visualize a tensor and its relationship with another tensor. An example of tensor network diagrams is shown in Fig. 2.7. A circle or node represents a tensor, and the number of branches is its order. Typically, each branch would have its dimension written along with it.

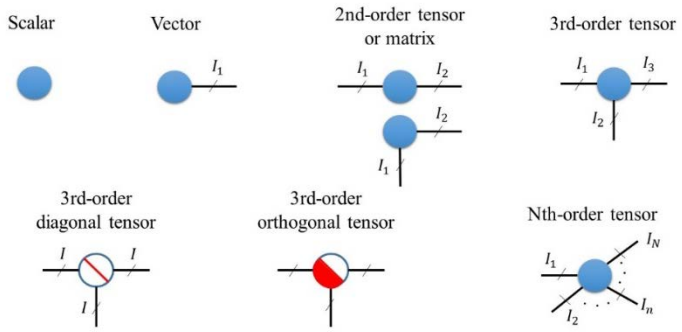

Fig. 2.7. Examples of tensor network diagrams

2.7.2 Hierarchical Tucker Decomposition

The Hierarchical Tucker (HT) decomposition introduced in [9] is a well-known tensor network framework. The HT decomposition uses a recursive algorithm to decompose a tensor into a binary tree of 2nd or 3rd- order core tensors. The example of HT decomposition of a 6th-order tensor is shown in Fig. 2.8. Since the highest order of its core tensors is three, the storage requirement for the HT decomposition is $\mathcal{O}(NIR + NR^3)$ [13]. This represents a significant storage reduction compared to TKD.

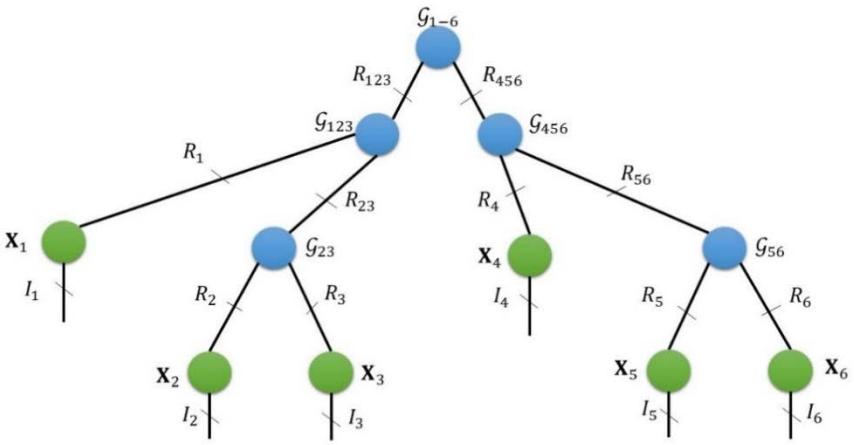

Fig. 2.8. Example of HT decomposition of a 6th-order tensor

The HT Decomposition of the tensor shown in Fig. 2.8 could be written as

$$\mathcal{X} \cong \sum_{r_{123}=1}^{R_{123}} \sum_{r_{456}=1}^{R_{456}} g_{r_{123},r_{456}}^{(1\cdots6)} \left(\mathcal{X}_{r_{123}}^{(123)} \circ \mathcal{X}_{r_{456}}^{(456)} \right) \quad , \tag{2.41}$$

$$\mathcal{X}_{123} \cong \sum_{r_1=1}^{R_1} \sum_{r_{23}=1}^{R_{23}} g_{r_1,r_{23},r_{123}}^{(123)} \left(\mathbf{x}_{r_1}^{(1)} \circ \mathbf{X}_{r_{23}}^{(23)} \right) \quad , \tag{2.42}$$

$$\mathcal{X}_{456} \cong \sum_{r_4=1}^{R_4} \sum_{r_{56}=1}^{R_{56}} g_{r_4,r_{56},r_{456}}^{(456)} \left(\mathbf{x}_{r_4}^{(4)} \circ \mathbf{X}_{r_{56}}^{(56)} \right) \quad , \tag{2.43}$$

$$\mathbf{X}_{23} \cong \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} g_{r_2,r_3,r_{23}}^{(23)} \left(\mathbf{x}_{r_2}^{(2)} \circ \mathbf{x}_{r_3}^{(3)} \right) \quad , \tag{2.44}$$

$$\mathbf{X}_{56} \cong \sum_{r_5=1}^{R_5} \sum_{r_6=1}^{R_6} g_{r_5,r_6,r_{56}}^{(56)} \left(\mathbf{x}_{r_5}^{(5)} \circ \mathbf{x}_{r_6}^{(6)} \right) \quad . \tag{2.45}$$

2.7.3 Tensor Train Decomposition

The Tensor Train (TT) decomposition, also known as the *linear tensor network*, was introduced by Oseledet in [10]. It is a particular case of Hierarchical Tucker decomposition, where all the tensor cores are connected as a train. Fig. 2.9 shows the tensor network diagram of a Tensor Train representation of a 4th-order tensor. The advantage of the TT decomposition over the HT decomposition is that the TT decomposition algorithm does not require recursive computations.

TT decomposition is purely based on QR and Singular Value Decomposition (SVD); therefore, we could easily impose low-rank approximations. A particular case of TT known as the Matrix Product State (MPS) decomposes a higher-order tensor into a set of 3rd-order tensor cores and two 2nd-order tensor cores. Given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ TT/MPS decomposition is given by

$$\begin{aligned} \mathcal{X} &\cong \mathbf{G}^{(1)} \times_2^1 \mathcal{G}^{(2)} \times_3^1 \mathcal{G}^{(3)} \times_3^1 \cdots \times_3^1 \mathcal{G}^{(N-1)} \times_3^1 \mathbf{G}^{(\mathrm{N})} \\ &= [\![\mathbf{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}, \cdots, \mathcal{G}^{(N-1)}, \mathbf{G}^{(\mathrm{N})}]\!] \end{aligned} \tag{2.46}$$

where $\mathcal{G}^{(\mathrm{n})} \in \mathbb{R}^{R_{n-1} \times I_n \times R_{n+1}}$ for $n = 2,3,\cdots,N-1$, $\mathbf{G}^{(1)} \in \mathbb{R}^{I_1 \times R_1}$ and $\mathbf{G}^{(\mathrm{N})} \in \mathbb{R}^{R_N \times I_N}$. The $R_1, R_2, \ldots, R_{N-1}$ are the ranks of these cores. Compared to TKD, the TT framework has a drawback as the permutation of the modes affects its core ranks. Another way to mathematically describe TT is by using all 3rd-order tensors where the $\mathcal{G}^{(1)} \in \mathbb{R}^{1 \times I_1 \times R_1}$ and $\mathcal{G}^{(\mathrm{N})} \in \mathbb{R}^{R_N \times I_N \times 1}$ we get

$$\begin{aligned} \mathcal{X} &\cong \mathcal{G}^{(1)} \times_3^1 \mathcal{G}^{(2)} \times_3^1 \mathcal{G}^{(3)} \times_3^1 \cdots \times_3^1 \mathcal{G}^{(N-1)} \times_3^1 \mathcal{G}^{(\mathrm{N})} \\ &= [\![\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}, \cdots, \mathcal{G}^{(N-1)}, \mathcal{G}^{(\mathrm{N})}]\!] \end{aligned} \tag{2.47}$$

The benefit of this TT approach is that all modes of the original tensor are represented by mode-2 of its cores. Therefore, one could define universal operations without concern for the position of the core in the train. In this thesis, we will use the convention presented in Eq. 2.47.

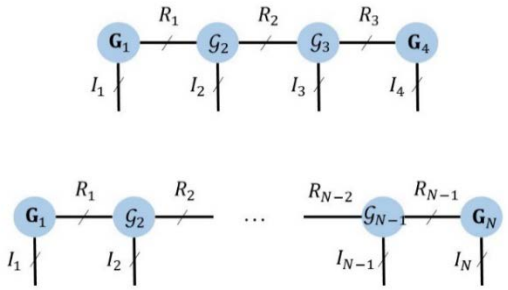

Fig. 2.9. Tensor network diagrams of (top) 4th order TT/MPS
(bottom) N^{th}order TT/MPS

Also, frequently, a tensor train core is represented as slice matrices given by $\mathbf{G}_{i_n}^{(\mathrm{n})} = \mathcal{G}^{(n)}(:, i_n, :) \in \mathbb{R}^{R_{n-1} \times 1 \times R_{n+1}}$ for $i_n = 1,2,\ldots,I_N$.

In Fig. 2.10, the 8th order tensors have been decomposed into three different structures. The Tucker Decomposition shown in (a) has one 8th-order tensor core, leading to massive storage and computation requirements. On the contrary, the Tensor Networks shown in (b, c) have several lower-order tensors that significantly reduce storage requirements. Unlike TKD and HT, the TT decomposition only requires the storage of $\mathcal{O}(NIR^2)$ [10].

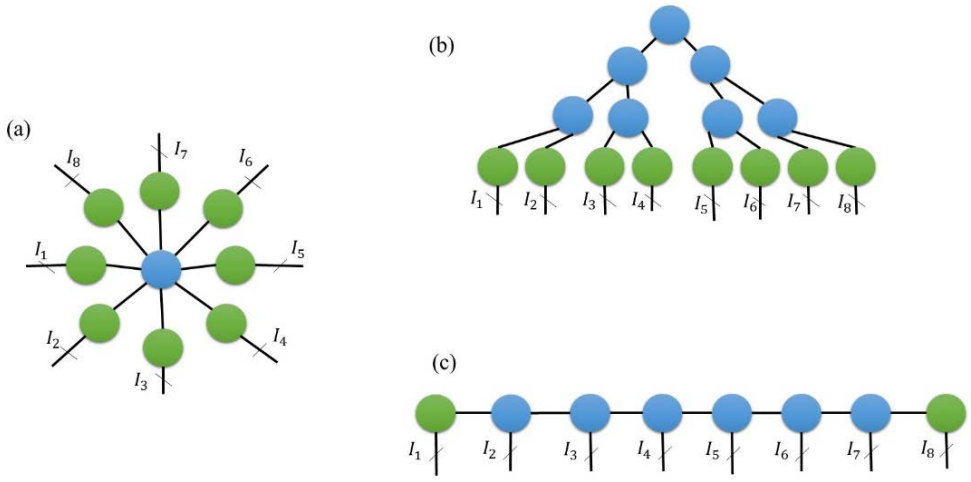

Fig. 2.10. The decomposition of an 8th order tensor in (a) Tucker decomposition (b) Hierarchical Tucker decomposition (c) tensor train/MPS, blue node represents 3rd or higher order tensor, and green node defines 2nd order tensor

Tensor Representation	Storage Complexity
Full Tensor	$\mathcal{O}(I^N)$
Tensor in Tucker Decomposition	$\mathcal{O}(NIR + R^N)$
Tensor in Hierarchical Tucker Decomposition	$\mathcal{O}(NIR + NR^3)$
Tensor in Tensor Train Decomposition (MPS)	$\mathcal{O}(NIR^2)$

Table 2.2. Comparison of storage complexities of an N^{th} order tensor with I dimensions and R ranks using different tensor representations

2.7.3.1 Hadamard Product of Tensor Train

Given two tensors with the same order N and same dimensions along all their modes, $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ that are decomposed as

$$\mathcal{X} = \left[\!\left[\mathcal{G}_X^{(1)}, \mathcal{G}_X^{(2)}, \mathcal{G}_X^{(3)}, \cdots, \mathcal{G}_X^{(N-1)}, \mathcal{G}_X^{(\mathrm{N})}\right]\!\right] \text{ and } \mathcal{Y} = \left[\!\left[\mathcal{G}_Y^{(1)}, \mathcal{G}_Y^{(2)}, \mathcal{G}_Y^{(3)}, \cdots, \mathcal{G}_Y^{(N-1)}, \mathcal{G}_Y^{(\mathrm{N})}\right]\!\right] \quad (2.48)$$

The Hadamard product of their two tensor trains [10,17] is given by

$$\mathcal{Z} = \mathcal{X} \circledast \mathcal{Y} = \left[\!\left[\mathcal{G}_{\mathcal{Z}}^{(1)}, \mathcal{G}_{\mathcal{Z}}^{(2)}, \mathcal{G}_{\mathcal{Z}}^{(3)}, \cdots, \mathcal{G}_{\mathcal{Z}}^{(\mathrm{N})} \right]\!\right] \tag{2.49}$$

where $\mathcal{G}_{\mathcal{Z}}^{(n)} = \mathcal{G}_{\mathcal{X}}^{(n)} \odot_2 \mathcal{G}_{\mathcal{Y}}^{(n)} \in \mathbb{R}^{R_{n-1}R_{n-1} \times I_n \times R_{n+1}R_{n+1}}$ with $R_0 = R_N = 1$ and $\odot_2$ is the mode-2 Khatri-Rao product. We note that mode-n Khatri-Rao products could be described using Kronecker products of slice cores

$$\mathbf{G}_{\mathcal{Z}_{i_n}}^{(n)} = \mathcal{G}_{\mathcal{Z}}^{(n)}(:, i_n, :) = \mathbf{G}_{\mathcal{X}_{i_n}}^{(n)} \otimes \mathbf{G}_{\mathcal{Y}_{i_n}}^{(n)} = \mathcal{G}_{\mathcal{X}}^{(n)}(:, i_n, :) \otimes \mathcal{G}_{\mathcal{Y}}^{(n)}(:, i_n, :) \tag{2.50}$$

for $i_n = 1,2, \ldots, I_N$.

2.7.3.2 *N*-D Convolution of Tensor Train

Suppose we have TT decompositions of functions $f(x, y, z)$ and $h(x, y, z)$ given as

$$\mathcal{F} = [\![\mathcal{X}_{\mathcal{F}}, \mathcal{Y}_{\mathcal{F}}, \mathcal{Z}_{\mathcal{F}}]\!] \text{ and } \mathcal{H} = [\![\mathcal{X}_{\mathcal{H}}, \mathcal{Y}_{\mathcal{H}}, \mathcal{Z}_{\mathcal{H}}]\!] \tag{2.51}$$

their dimensions are $\mathcal{F} \in \mathbb{R}^{I_x \times I_y \times I_z}, \mathcal{X}_{\mathcal{F}} \in \mathbb{R}^{1 \times I_x \times R_x}, \mathcal{Y}_{\mathcal{F}} \in \mathbb{R}^{R_x \times I_y \times R_y}, \mathcal{Z}_{\mathcal{F}} \in \mathbb{R}^{R_y \times I_z \times 1}$ and $\mathcal{H} \in \mathbb{R}^{J_x \times J_y \times J_z}, \mathcal{X}_{\mathcal{H}} \in \mathbb{R}^{1 \times J_x \times Q_x}, \mathcal{Y}_{\mathcal{H}} \in \mathbb{R}^{Q_x \times J_y \times Q_y}, \mathcal{Z}_{\mathcal{H}} \in \mathbb{R}^{Q_y \times J_z \times 1}$. Similar to *N*-D Convolution in Tucker Decomposition, we could apply partial convolutions to the physical modes of each pair of tensor cores to obtain the final convolution result. This partial convolution should be performed on mode-2 fibers, yielding

$$\mathcal{F} * \mathcal{H} = [\![\mathcal{X}_{\mathcal{F}} \boxdot_2 \mathcal{X}_{\mathcal{H}}, \mathcal{Y}_{\mathcal{F}} \boxdot_2 \mathcal{Y}_{\mathcal{H}}, \mathcal{Z}_{\mathcal{F}} \boxdot_2 \mathcal{Z}_{\mathcal{H}}]\!]. \tag{2.52}$$

where $\boxdot_2$ is the mode-2 partial convolution.

Generally, if we have two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ that are represented as Tensor Train representation

$$\mathcal{X} = \left[\!\left[\mathcal{X}^{(1)}, \mathcal{X}^{(2)}, \cdots, \mathcal{X}^{(N)} \right]\!\right] \text{ and } \mathcal{Y} = \left[\!\left[\mathcal{Y}^{(1)}, \mathcal{Y}^{(2)}, \cdots, \mathcal{Y}^{(N)} \right]\!\right]. \tag{2.53}$$

A full *N*-D convolution of these two tensors is obtaining by the partial mode-2 convolution of the corresponding tensor cores. Therefore,

$$\mathcal{X} * \mathcal{Y} = \left[\!\left[\mathcal{X}^{(1)} \boxdot_2 \mathcal{Y}^{(1)}, \mathcal{X}^{(2)} \boxdot_2 \mathcal{Y}^{(2)}, \cdots, \mathcal{X}^{(N)} \boxdot_2 \mathcal{Y}^{(N)} \right]\!\right]. \tag{2.54}$$

2.8 Chapter Summary

In this chapter, we reviewed tensors including their operations and decompositions. We described three tensor decompositions, i.e., Tucker Decompositions, Hierarchical Tucker Decomposition, and Tensor Train Decomposition. Tucker Decomposition requires the most storage of $\mathcal{O}(NIR + R^N)$ due to a high-order tensor core. The Hierarchical Tucker Decomposition requires $\mathcal{O}(NIR + NR^3)$ of storage and a recursive computation algorithm. In comparison, Tensor Train decomposition requires the least storage of $\mathcal{O}(NIR^2)$. Therefore, TT is usually most suitable for high-dimensional problems. We also described both tensor element-wise product and full N-D convolution and their implementation in these tensor decompositions. Practically, we would need to apply convolution to some tensor modes but not to others. In the next chapter, we will describe our novel approach to tensor convolution along arbitrary tensor modes.

3. Arbitrary Mode-n Convolution of Physical Tensors

3.1 Motivation for Arbitrary Mode-n Convolution of Physical Tensors

In the current literature on tensor convolution, tensors to be convolved are viewed as abstract mathematical entities. *N*-D tensor convolution has three assumptions imposed on the tensors to be convolved: 1) they have the same order; 2) they represent the same physical variables on their corresponding modes; 3) they are to be convolved on every mode.

Our ultimate long-term research objective is to represent Fourier Optics problems using tensors; hence we are interested in imposing physical interpretations on tensors. In this thesis, we refer to a tensor with a physical interpretation associated with its modes as a *physical tensor*.

In general, convolution of two functions implies convolution along the same physical variable, e.g., for $f(x) * h(x)$ x would have the same physical interpretation. That is why, given multidimensional functions $f(y,z,x)$ and $h(x,y,z)$ with tensor representations, $\mathcal{F}(y,z,x)$ and $\mathcal{H}(x,y,z)$, caution must be taken before applying *N*-D tensor convolution as their modes are ordered differently.

In the above example, there would be a need to reorder the modes of $\mathcal{F}(y,z,x)$ and $\mathcal{H}(x,y,z)$ before performing tensor convolution. Therefore, tensor mode matching via reordering could be occasionally essential for tensor convolution.

Another problem arises in *N*-D tensor convolution when we try to convolve, e.g., $f(u,v,x,y)$ with $h(x,y,z)$, where variables $u, v,$ and z are absent from the second and first function, respectively. To perform such convolution along a specific subset of modes, one would need to expand the number of modes of each tensor to include dummy variables instead of the missing variables in each tensor. Mode-matching could also be necessary afterward.

3.1.1 Arbitrary Mode-n Convolution with Mode Expansion on Tensor Edges

Often, we only seek to convolve along a specific subset of tensor modes. For example, given two functions $f(u,v,x,y)$ and $h(x,y,z)$. We would like to convolve along *x* only and obtain the result as a function $s(u,v,x,y,z)$. The two functions have the common variables x and y. From a physical aspect, y of both functions should have the same sampling rate and size. The direct approach for this convolution along *x* would be

$$\mathcal{S}(u,v,x,y,z) = \mathcal{F} *_x \mathcal{H} = \sum_{x'} \mathcal{H}(x',y,z)\mathcal{F}(u,v,x-x',y). \tag{3.1}$$

This approach would work well with small problem sizes. However, tensors could be very large; thus, arbitrary mode-n convolution using decomposed tensors would be needed. Given the tensor Tucker Decomposition of both functions as

$$\mathcal{F} = [\![\mathcal{G}_{\mathcal{F}}; \mathbf{U}_{\mathcal{F}}, \mathbf{V}_{\mathcal{F}}, \mathbf{X}_{\mathcal{F}}, \mathbf{Y}_{\mathcal{F}}]\!] \text{ and } \mathcal{H} = [\![\mathcal{G}_{\mathcal{H}}; \mathbf{X}_{\mathcal{H}}, \mathbf{Y}_{\mathcal{H}}, \mathbf{Z}_{\mathcal{H}}]\!]. \tag{3.2}$$

Their dimensions are $\mathcal{F} \in \mathbb{R}^{I_u \times I_v \times I_x \times I_y}, \mathcal{G}_{\mathcal{F}} \in \mathbb{R}^{R_u \times R_v \times R_x \times R_y}$ and $\mathcal{H} \in \mathbb{R}^{J_x \times J_y \times J_z}, \mathcal{G}_{\mathcal{H}} \in \mathbb{R}^{Q_x \times Q_y \times Q_z}$. Given their Tensor Train Decompositions as well

$$\mathcal{F} = [\![\mathcal{U}_{\mathcal{F}}, \mathcal{V}_{\mathcal{F}}, \mathcal{X}_{\mathcal{F}}, \mathcal{Y}_{\mathcal{F}}]\!] \text{ and } \mathcal{H} = [\![\mathcal{X}_{\mathcal{H}}, \mathcal{Y}_{\mathcal{H}}, \mathcal{Z}_{\mathcal{H}}]\!] \tag{3.3}$$

where $\mathcal{F} \in \mathbb{R}^{I_u \times I_v \times I_x \times I_y}, \mathcal{U}_{\mathcal{F}} \in \mathbb{R}^{1 \times I_u \times R_u}, \mathcal{V}_{\mathcal{F}} \in \mathbb{R}^{R_u \times I_v \times R_v}, \mathcal{X}_{\mathcal{F}} \in \mathbb{R}^{R_v \times I_x \times R_x}, \mathcal{Y}_{\mathcal{F}} \in \mathbb{R}^{R_x \times I_y \times 1}$ and $\mathcal{H} \in \mathbb{R}^{J_x \times J_y \times J_z}, \mathcal{X}_{\mathcal{H}} \in \mathbb{R}^{1 \times J_x \times Q_x}, \mathcal{Y}_{\mathcal{H}} \in \mathbb{R}^{Q_x \times J_y \times Q_y}, \mathcal{Z}_{\mathcal{H}} \in \mathbb{R}^{Q_y \times J_z \times 1}$. We note that the dimension of the common-mode *y* that does not get convolved should be the same, i.e., $I_y = J_y$.

In this example, one needs to consider three subsets of modes (variables)

1. The convolved mode, *x*
2. The common mode y where convolution is not applied
3. The uncommon modes, *u*, *v* and *z*

This thesis presents a tensor convolution method that considers all subsets of modes (variables) listed above. Our approach starts with pre-processing the input tensors to satisfy the following conditions

1. Both tensors $\mathcal{X}$ and $\mathcal{Y}$ must have the same order *L*, where *L* is the number of all distinct modes (variables) in both tensors.

2. Tensor cores and factor matrices having the same physical interpretation must correspond to the same mode in both to be convolved tensors.

We start this tensor pre-processing first by artificially expanding the order of both tensors (see appendix A) to have the same order

$$\boldsymbol{\mathcal{F}} = [\![\boldsymbol{\mathcal{G}}_{\mathcal{F}}; \mathbf{U}_{\mathcal{F}}, \mathbf{V}_{\mathcal{F}}, \mathbf{X}_{\mathcal{F}}, \mathbf{Y}_{\mathcal{F}}, 1]\!] \text{ and } \boldsymbol{\mathcal{H}} = [\![\boldsymbol{\mathcal{G}}_{\mathcal{H}}; 1,1, \mathbf{X}_{\mathcal{H}}, \mathbf{Y}_{\mathcal{H}}, \mathbf{Z}_{\mathcal{H}}]\!] \tag{3.4}$$

where the last artificially expanded mode of tensors $\boldsymbol{\mathcal{F}}$ and $\boldsymbol{\mathcal{G}}_{\mathcal{F}}$ have a dimension of one. Similarly, $\boldsymbol{\mathcal{H}}$ and $\boldsymbol{\mathcal{G}}_{\mathcal{H}}$ have artificially expanded first two modes of dimension one.

Similarly, artificially enlarging the modes of TT would result in

$$\boldsymbol{\mathcal{F}} = [\![\mathcal{U}_{\mathcal{F}}, \mathcal{V}_{\mathcal{F}}, \mathcal{X}_{\mathcal{F}}, \mathcal{Y}_{\mathcal{F}}, 1]\!] \text{ and } \boldsymbol{\mathcal{H}} = [\![1,1, \mathcal{X}_{\mathcal{H}}, \mathcal{Y}_{\mathcal{H}}, \mathcal{Z}_{\mathcal{H}}]\!] \tag{3.5}$$

These steps would conclude our required pre-processing steps.

Computing arbitrary mode-*n* convolution on physical tensors is not as direct as the *N*-D convolution mentioned in the previous chapter. First, we perform mode-*n* partial convolution on the to be convolved mode x. Secondly, for the common mode, *y*, no operations are needed for this mode. Therefore, we could write arbitrary mode-n convolution of physical tensors using TKD as

$$\begin{aligned} \mathcal{F} *_x \mathcal{H} &= \boldsymbol{\mathcal{F}} *_x \boldsymbol{\mathcal{H}} \\ &= [\![\boldsymbol{\mathcal{G}}_{\mathcal{F}} \otimes \boldsymbol{\mathcal{G}}_{\mathcal{H}}; \mathbf{U}_{\mathcal{F}}, \mathbf{V}_{\mathcal{F}}, \mathbf{X}_{\mathcal{F}} \boxdot_1 \mathbf{X}_{\mathcal{H}}, \mathbf{Y}_{\mathcal{F}} \odot_1 \mathbf{Y}_{\mathcal{H}}, \mathbf{Z}_{\mathcal{H}}]\!]. \end{aligned} \tag{3.6}$$

Similarly, we could write arbitrary mode-n convolution of physical tensors using TT as

$$\begin{aligned} \mathcal{F} *_x \mathcal{H} &= \boldsymbol{\mathcal{F}} *_x \boldsymbol{\mathcal{H}} \\ &= [\![\mathcal{U}_{\mathcal{F}}, \mathcal{V}_{\mathcal{F}}, \mathcal{X}_{\mathcal{F}} \boxdot_2 \mathcal{X}_{\mathcal{H}}, \mathcal{Y}_{\mathcal{F}} \odot_2 \mathcal{Y}_{\mathcal{H}}, \mathcal{Z}_{\mathcal{H}}]\!]. \end{aligned} \tag{3.7}$$

Diagram in Fig. 3.1 shows the Tensor Train Decomposition of the original tensors, artficially expanded order tensors, and their tensor convolution result. We have validated our results, given by Eq. 3.6 and Eq. 3.7 using Matlab tensor toolboxes provided by both Sandia National Laboratories and Oseledets. The tensors corresponding to these two functions $f(u, v, x, y)$ and $h(x, y, z)$ were generated using random entries. The dimensions of the generated tensors are $\mathcal{F} \in \mathbb{R}^{18\times16\times20\times13}$ corresponding to 74,880 entries, and $\mathcal{H} \in \mathbb{R}^{19\times13\times21}$ corresponding to 5,187 entries. The common mode *y* in both tensors had the same dimension equal to 13.

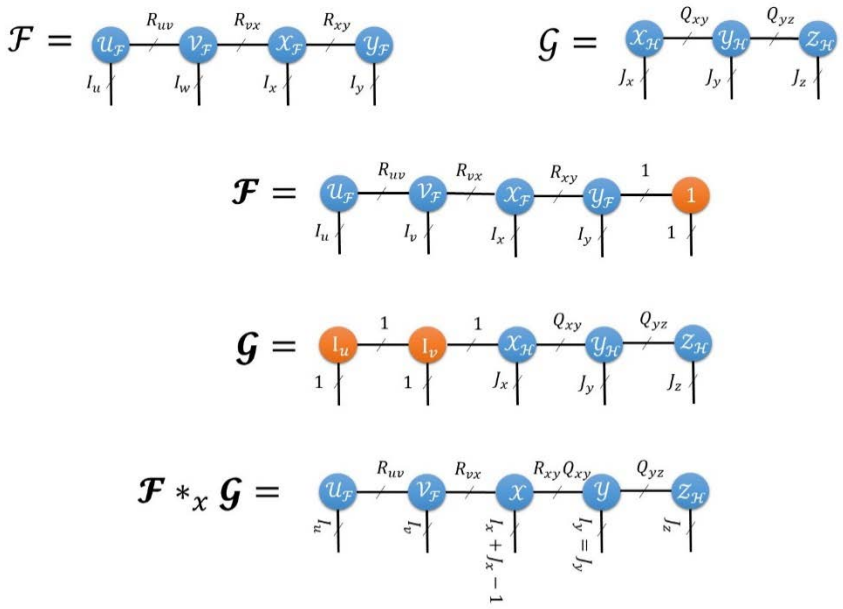

Fig. 3.1. Network diagram of convolution on x of $f(u,v,x,y)$ and $h(x,y,z)$ using Tensor Train Decomposition

We computed the tensor convolution along x using three approaches, Eq. 3.1, Eq. 3.6 and Eq. 3.7, where its direct computation is given by Eq. 3.1. Our results are shown in Fig. 3.2 to 3.4, where the first rows show different slices from the resulting tensor, $\mathcal{S}$, using direct computation (1st columns), TKD (2nd columns) and TT (3rd columns). The second rows of Fig. 3.2 to 3.4 show the corresponding slices from different tensors representing their differences.

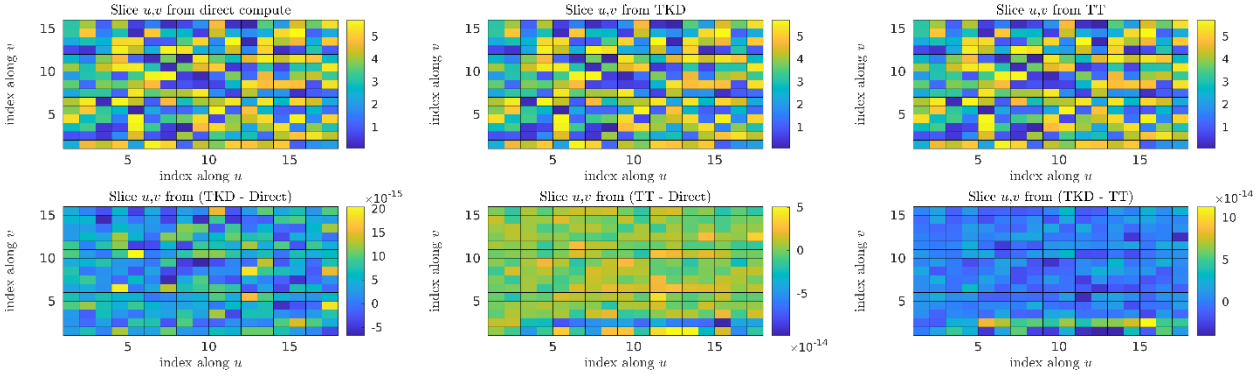

Fig. 3.2. Slice $\mathcal{S}[:,:,1,1,1]$ of different tensor convolutions along x obtained using direct computation, TKD, and TT (first row), and their differences (second row)

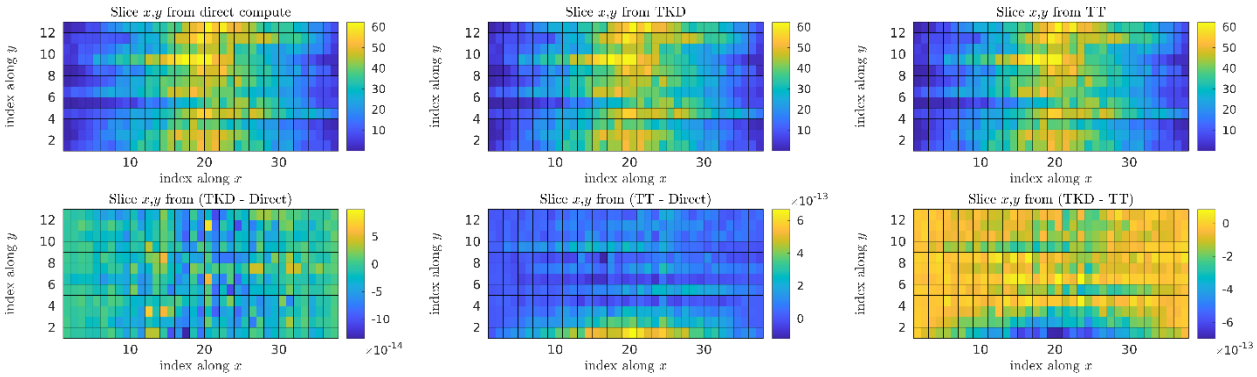

Fig. 3.3. Slice $\mathcal{S}[1,1,:,:,1]$ of different tensor convolutions along *x* obtained using direct computation, TKD, and TT (first row), and their differences (second row)

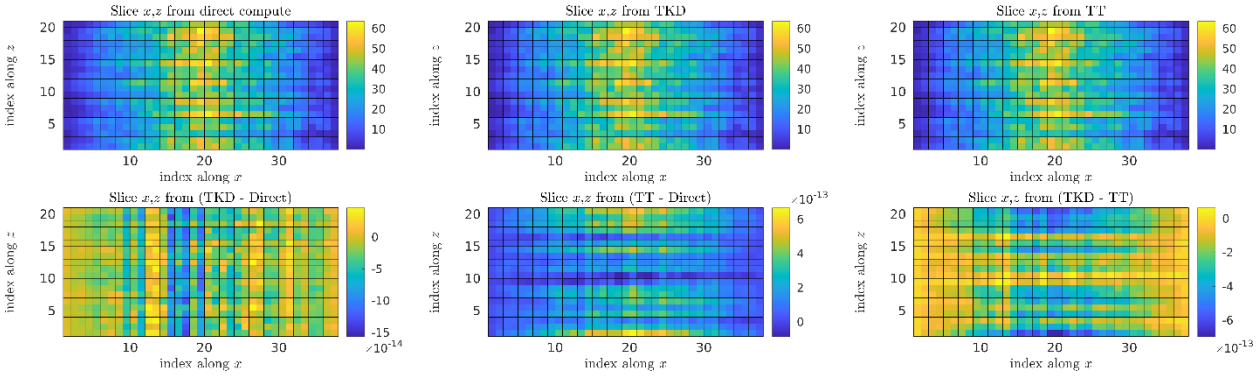

Fig. 3.4. Slice $\mathcal{S}[1,1,:,1,:]$ of different tensor convolutions along *x* obtained using direct computation, TKD, and TT (first row), and their differences (second row)

The norm of the tensor containing the differences (error) between the tensor results obtained by using direct computation and TKD (computation time 9 sec) is 6.3795×10^{-11}. Also, the Tensor Train-based tensor results (computation time 0.2674 sec) resulted in a tensor error norm of 2.0835×10^{-10}. Therefore, our results demonstrate that using tensor TKD-based arbitrary mode-*n* convolution reduces numerical errors, but it requires significantly longer computation time and storage (due to required Kronecker product between tensor cores).

To confirm these results, we obtained mode-*n* convolution of tensors of the same order but with larger size due to higher mode dimensions, i.e., $\mathcal{F} \in \mathbb{R}^{30\times62\times43\times56}$ and $\mathcal{H} \in \mathbb{R}^{51\times56\times41}$ corresponding to 3,908 GB entries.

The results shown in Fig. 3.5 were obtained with TT using computer storage of 3.76 GB in 27 seconds, thereby further confirming our earlier conclusions.

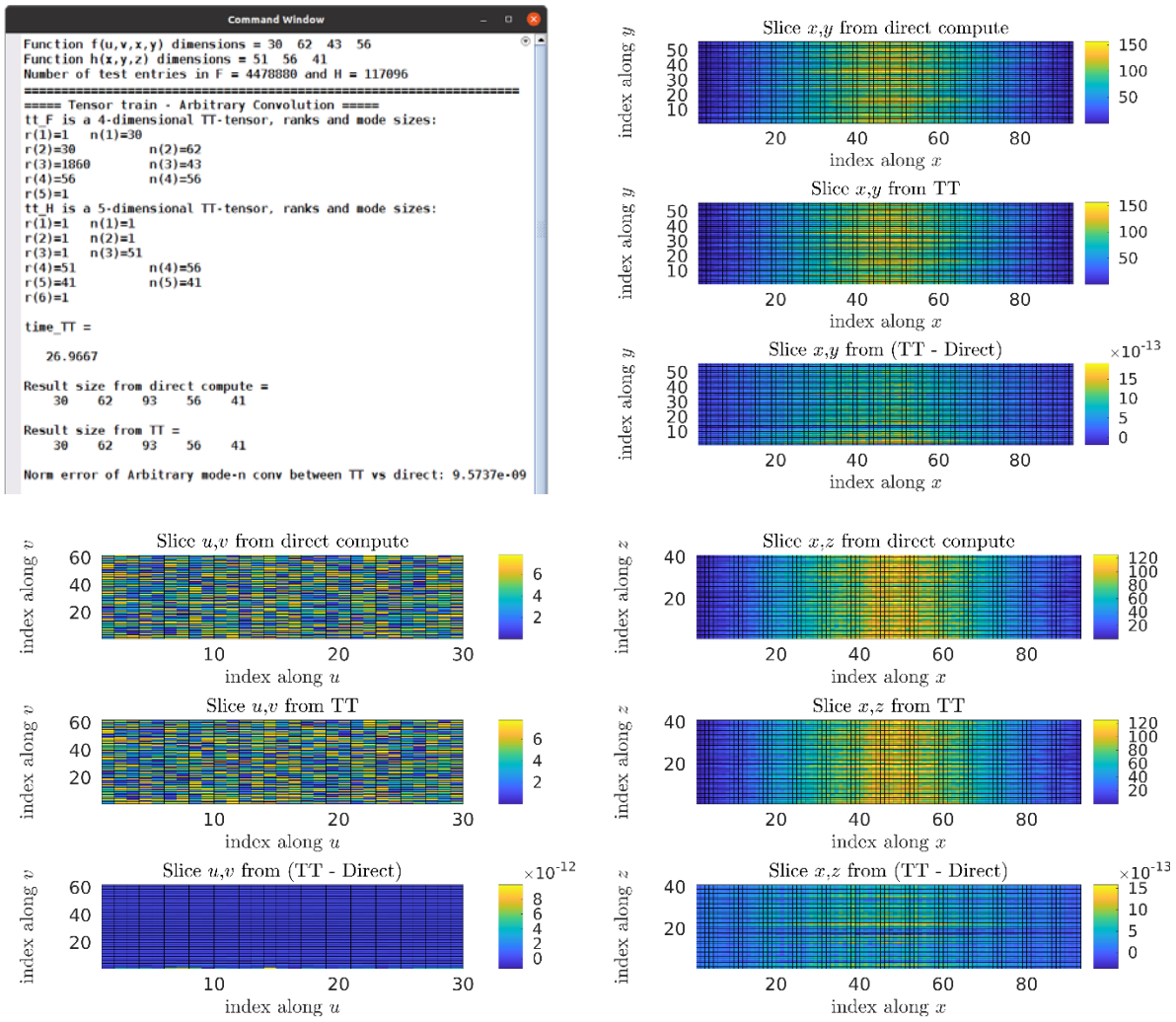

Fig. 3.5. Tensor train based arbitrary mode-x convolution on a larger problem size

3.1.2 Arbitrary Mode-n Convolution with Mode Expansion Independent of Their Positions

Given the functions $f(t,u,w,x,y)$ and $h(t,v,w,x,y,z)$. We would like to obtain their convolution along both variables t and x. Both functions have the same sampling rate and support along modes w and y. The direct approach to perform this convolution is to compute

$$\begin{aligned}\mathcal{S}(t,u,v,w,x,y,z) &= \mathcal{F} *_{t,x} \mathcal{H} \\ &= \textstyle\sum_{t'}\sum_{x'} \mathcal{H}(t-t',v,w,x-x',y,z)\mathcal{F}(t-t',u,w,x-x',y) \end{aligned} \quad (3.8)$$

to obtain the 7th order tensor.

We could also obtain the synthetically order expanded TKD of the two tensors (see appendix A) corresponding to these two multidimensional functions, as follows

$$\boldsymbol{\mathcal{F}} = [\![\boldsymbol{\mathcal{G}}_{\mathcal{F}}; \mathbf{T}_{\mathcal{F}}, \mathbf{U}_{\mathcal{F}}, 1, \mathbf{W}_{\mathcal{F}}, \mathbf{X}_{\mathcal{F}}, \mathbf{Y}_{\mathcal{F}}, 1]\!] \text{ and}$$

$$\boldsymbol{\mathcal{H}} = [\![\boldsymbol{\mathcal{G}}_{\mathcal{H}}; \mathbf{T}_{\mathcal{H}}, 1, \mathbf{V}_{\mathcal{H}}, \mathbf{W}_{\mathcal{H}}, \mathbf{X}_{\mathcal{H}}, \mathbf{Y}_{\mathcal{H}}, \mathbf{Z}_{\mathcal{H}}]\!] \tag{3.9}$$

Similarly, obtaining the synthetically order expanded TT of the two tensors

$$\boldsymbol{\mathcal{F}} = [\![\mathcal{T}_{\mathcal{F}}, \mathcal{U}_{\mathcal{F}}, \mathbf{I}_v, \mathcal{W}_{\mathcal{F}}, \mathcal{X}_{\mathcal{F}}, \mathcal{Y}_{\mathcal{F}}, 1]\!] \text{ and } \boldsymbol{\mathcal{H}} = [\![\mathcal{T}_{\mathcal{H}}, \mathbf{I}_u, \mathcal{V}_{\mathcal{H}}, \boldsymbol{\mathcal{W}}_{\mathcal{H}}, \mathcal{X}_{\mathcal{H}}, \mathcal{Y}_{\mathcal{H}}, \mathcal{Z}_{\mathcal{H}}]\!]. \tag{3.10}$$

Fig. 3.6 shows the original tensors, their Tensor Train Decomposition, and the expected tensor convolution result.

We note that we could not directly use the previously developed approach, Eq. 3.7, to obtain the required mode-n tensor convolution due to the rank mismatch between the train cores. To overcome this problem, a modification could easily be made by applying a Kronecker product between the expanded modes such that $\mathcal{U}_{\mathcal{F}} \otimes \mathbf{I}_u \in \mathbb{R}^{R_{tu}Q_{tv} \times I_u \times R_{uw}Q_{tv}}$ and $\mathbf{I}_v \otimes \mathcal{V}_{\mathcal{H}} \in \mathbb{R}^{R_{uw}Q_{tv} \times J_v \times R_{uw}Q_{vw}}$. Notice that at the first and last cores of the train, we could apply $1 \otimes \mathcal{Z}_{\mathcal{H}} = \mathcal{Z}_{\mathcal{H}}$ which is the same approach in Eq. 3.7.

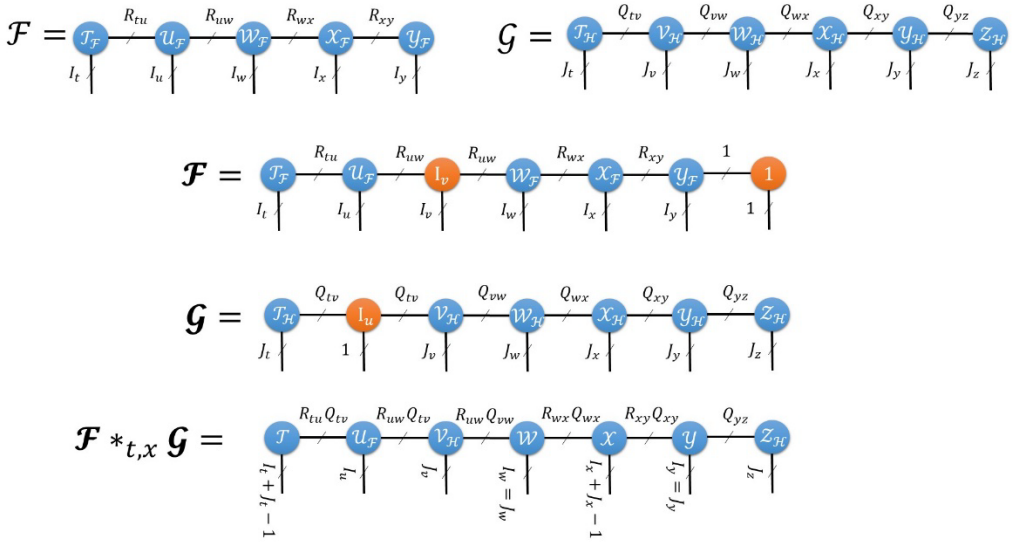

Fig. 3.6. Network diagram of $f(t,u,w,x,y)$ and $h(t,v,w,x,y,z)$ in Tensor Train Decomposition and their mode-(t,x) arbitrary convolution

Therefore, we could write the mode-*n* tensor convolution in TKD representation as

$$\mathcal{F} *_{t,x} \mathcal{H} = \boldsymbol{\mathcal{F}} *_{t,x} \boldsymbol{\mathcal{H}}$$

$$= [\![\boldsymbol{\mathcal{G}}_{\mathcal{F}} \otimes \boldsymbol{\mathcal{G}}_{\mathcal{H}}; \mathbf{T}_{\mathcal{F}} \boxdot_1 \mathbf{T}_{\mathcal{H}}, \mathbf{U}_{\mathcal{F}}, \mathbf{V}_{\mathcal{F}}, \mathbf{W}_{\mathcal{F}} \odot_1 \mathbf{W}_{\mathcal{H}}, \mathbf{X}_{\mathcal{F}} \boxdot_1 \mathbf{X}_{\mathcal{H}}, \mathbf{Y}_{\mathcal{F}} \odot_1 \mathbf{Y}_{\mathcal{H}}, \mathbf{Z}_{\mathcal{H}}]\!] \qquad (3.11)$$

and in TT representation as

$$\mathcal{F} *_{t,x} \mathcal{H} = \boldsymbol{\mathcal{F}} *_{t,x} \boldsymbol{\mathcal{H}}$$

$$= [\![\mathcal{T}_{\mathcal{F}} \boxdot_2 \mathcal{T}_{\mathcal{H}}, \mathcal{U}_{\mathcal{F}} \otimes \mathbf{I}_u, \mathbf{I}_v \otimes \mathcal{V}_{\mathcal{H}}, \mathcal{W}_{\mathcal{F}} \odot_2 \mathcal{W}_{\mathcal{H}}, \mathcal{X}_{\mathcal{F}} \boxdot_2 \mathcal{X}_{\mathcal{H}}, \mathcal{Y}_{\mathcal{F}} \odot_2 \mathcal{Y}_{\mathcal{H}}, \mathcal{Z}_{\mathcal{H}}]\!]. \qquad (3.12)$$

We verified these mode-*n* convolution formulations, Eq. 3.11 and Eq. 3.12, using relatively smaller dimension tensors, $\mathcal{F} \in \mathbb{R}^{6\times4\times8\times9\times7}$ and $\mathcal{H} \in \mathbb{R}^{5\times10\times8\times7\times7\times5}$, due to the storage limitations of computer we used. In Fig. 3.7, we show different slices from the resulting tensor, $\mathcal{S}$, using direct computation (1st column), TKD (2nd column) and TT (3rd column).

The norm of the tensor containing the differences (error) between the tensor results obtained by using direct computation and TKD (computation time 39.582 sec) is 7.7985×10^{-11}. Also, the Tensor Train-based tensor results (computation time 13.0741 sec) resulted in a tensor error norm of 3.2507×10^{-10}. These results further confirm that TT-based mode-*n* tensor convolution generally results in superior performance (lower computer storage and computation time requirements)

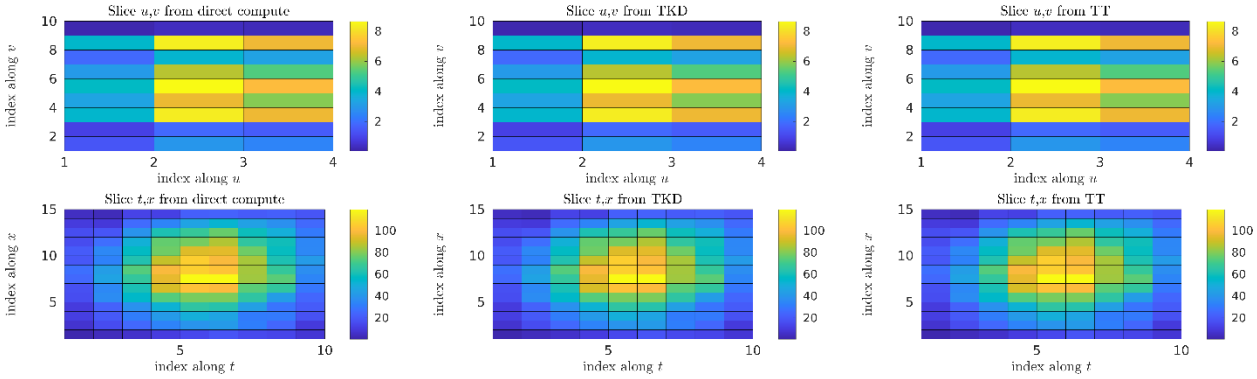

Fig. 3.7A. Slices of tensor convolutions along *u*-*v* (top row) and *t*-*x* (bottom row) obtained using direct computation (1st column), TKD (2nd column), and TT (3rd column)

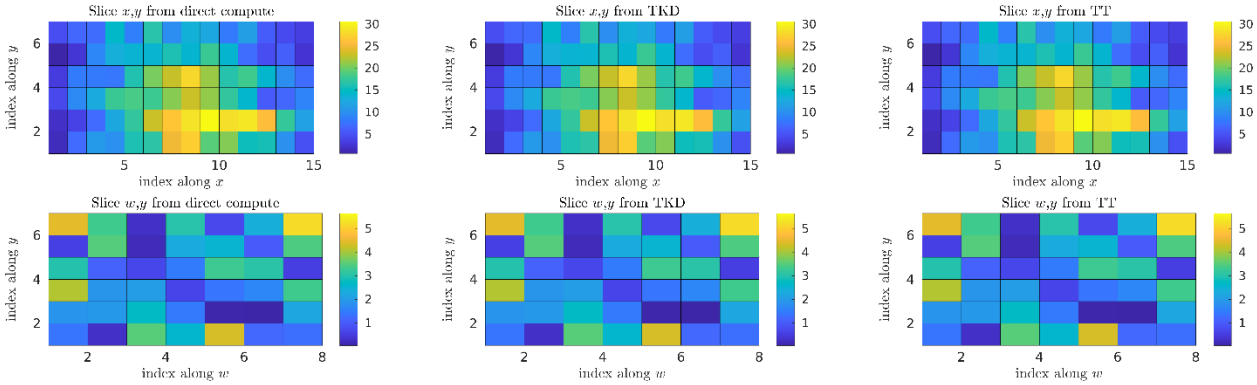

Fig. 3.7B. Slices of tensor convolutions along *x*-*y* (top row) and *w*-*y* (bottom row) obtained using direct computation (1st column), TKD (2nd column), and TT (3rd column)

3.3 General Formulation of Arbitrary Mode-n Convolution of Physical Tensors

Given two tensors $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\mathcal{B} \in \mathbb{R}^{I_1 \times \cdots \times I_k \times J_{k+1} \times \cdots \times J_M}$ representing physical variables, they could have K modes with the same physical interpretation where $K \leq N, M$,i.e., $\mathbf{A}^{(k)}$ and $\mathbf{B}^{(k)}$ represent the same physical variable in their corresponding functions. The common mode-n fibers that are not convolved should have the same sampling rate and support. Their Tucker and Tensor Train Decompositions are given by

$$\mathcal{A} = [\![\mathcal{G}_{\mathcal{A}}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N)}]\!] \text{ and } \mathcal{B} = [\![\mathcal{G}_{\mathcal{B}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \cdots, \mathbf{B}^{(M)}]\!], \tag{3.13}$$

and

$$\mathcal{A} = [\![\mathcal{G}_{\mathcal{A}}^{(1)}, \mathcal{G}_{\mathcal{A}}^{(2)}, \cdots, \mathcal{G}_{\mathcal{A}}^{(N-1)}, \mathcal{G}_{\mathcal{A}}^{(N)}]\!] \text{ and } \mathcal{B} = [\![\mathcal{G}_{\mathcal{B}}^{(1)}, \mathcal{G}_{\mathcal{B}}^{(2)}, \cdots, \mathcal{G}_{\mathcal{B}}^{(N-1)}, \mathcal{G}_{\mathcal{B}}^{(M)}]\!], \tag{3.14}$$

respectively.

3.3.1 Pre-processing Physical Tensors by Artificial Mode Expansion

First, we need to pre-process the given tensors so that both $\mathcal{A}$ and $\mathcal{B}$ have the same order, $L = M + N - K$, by artificially expanding the orders of these tensors. To assign mode-n fibers in these artificially expanded tensors, we start with the mode-n fibers that are represent the common-mode k first, followed by the modes that are unique to both $\mathcal{A}$ and $\mathcal{B}$. Denoting all K common physical mode-l fibers by using their corresponding physical variables $u, \cdots, z$, we could write their TKD representations

$$\boldsymbol{\mathcal{A}} = \left[\!\left[\boldsymbol{\mathcal{G}}_{\mathcal{A}}; \mathbf{U}_{\mathcal{A}}, \cdots, \mathbf{Z}_{\mathcal{A}}, \mathbf{A}^{(K+1)}, \cdots, \mathbf{A}^{(N)}, 1^{(N+1)}, \cdots, 1^{(L)} \right]\!\right]$$

$$\boldsymbol{\mathcal{B}} = \left[\!\left[\boldsymbol{\mathcal{G}}_{\mathcal{B}}; \mathbf{U}_{\mathcal{B}}, \cdots, \mathbf{Z}_{\mathcal{B}}, 1^{(K+1)}, \cdots, 1^{(N)}, \mathbf{B}^{(N+1)}, \cdots, \mathbf{B}^{(L)} \right]\!\right] \tag{3.15}$$

where $\mathbf{A}^{(K+1)}, \cdots, \mathbf{A}^{(N)}$ and $\mathbf{B}^{(N+1)}, \cdots, \mathbf{B}^{(L)}$ are the modes that are unique to both $\mathcal{A}$ and $\mathcal{B}$, respectively. The cores $\boldsymbol{\mathcal{G}}_{\mathcal{A}}$ and $\boldsymbol{\mathcal{G}}_{\mathcal{B}}$ have also been expanded where modes $N+1$ to L of $\boldsymbol{\mathcal{G}}_{\mathcal{A}}$ are fibers of length one or scalars as well as modes $K+1$ to N of $\boldsymbol{\mathcal{G}}_{\mathcal{B}}$.

For the Tensor Train representation, we have

$$\boldsymbol{\mathcal{A}} = \left[\!\left[\mathcal{U}_{\mathcal{A}}, \mathcal{V}_{\mathcal{A}} \cdots, \mathcal{Z}_{\mathcal{A}}, \mathcal{G}_{\mathcal{A}}^{(K+1)}, \cdots, \mathcal{G}_{\mathcal{A}}^{(N)}, 1^{(N+1)}, \cdots, 1^{(L)} \right]\!\right]$$

$$\boldsymbol{\mathcal{B}} = \left[\!\left[\mathcal{U}_{B}, \mathcal{V}_{\mathcal{B}} \cdots, \mathcal{Z}_{\mathcal{B}}, \mathbf{I}^{(K+1)}, \cdots, \mathbf{I}^{(N)}, \mathcal{G}_{\mathcal{B}}^{(N+1)}, \cdots, \mathcal{G}_{\mathcal{B}}^{(L)} \right]\!\right]. \tag{3.16}$$

The purpose of assigning the tensor modes this way is to facilitate understanding. However, one could constantly reorder the modes as desired. In the Tensor Train format, the core ranks depend on this ordering. The most efficient way to pre-process these tensors is to add the required expansion modes at the beginning or end of the Tensor Trains (see Eq. 3.21 and Eq. 3.22). In this case, all additional cores would be scalars.

3.3.2 Select the Arbitrary Mode-*n* to Apply Convolution

Since mode-*k* fibers of both tensors must have the same length except for the fibers mode-*n* that will be convolved. Therefore, we denote all the same length fibers with the same variables. The equation below assumes that the mode-*n* physical variable is *x*.

For the Tucker Decomposition representation, we have

$$\boldsymbol{\mathcal{A}} = \left[\!\left[\boldsymbol{\mathcal{G}}_{\mathcal{A}}; \mathbf{U}, \cdots, \mathbf{X}_{\mathcal{A}}, \cdots, \mathbf{Z}, \mathbf{A}^{(K+1)}, \cdots, \mathbf{A}^{(N)}, 1^{(N+1)}, \cdots, 1^{(L)} \right]\!\right]$$

$$\boldsymbol{\mathcal{B}} = \left[\!\left[\boldsymbol{\mathcal{G}}_{\mathcal{B}}; \mathbf{U}, \cdots, \mathbf{X}_{\mathcal{B}}, \cdots, \mathbf{Z}, 1^{(K+1)}, \cdots, 1^{(N)}, \mathbf{B}^{(N+1)}, \cdots, \mathbf{B}^{(L)} \right]\!\right] \tag{3.17}$$

For the Tensor Train representation, we have

$$\boldsymbol{\mathcal{A}} = \left[\!\left[\mathcal{U}, \mathcal{V}, \cdots, \mathcal{X}_{\mathcal{A}}, \cdots, \mathcal{Z}, \mathcal{G}_{\mathcal{A}}^{(K+1)}, \cdots, \mathcal{G}_{\mathcal{A}}^{(N)}, 1^{(N+1)}, \cdots, 1^{(L)} \right]\!\right]$$

$$\boldsymbol{\mathcal{B}} = \left[\!\left[\mathcal{U}, \mathcal{V}, \cdots, \mathcal{X}_{\mathcal{B}}, \cdots, \mathcal{Z}, \mathbf{I}^{(K+1)}, \cdots, \mathbf{I}^{(N)}, \mathcal{G}_{\mathcal{B}}^{(N+1)}, \cdots, \mathcal{G}_{\mathcal{B}}^{(L)} \right]\!\right] \tag{3.18}$$

3.3.3 Arbitrary Mode-*n* Convolution

After the performing the pre-processing steps above, the *arbitrary mode-n convolution* for two tensors, represented by their Tucker Decompositions, could be written as

$$\mathcal{A} *_n \mathcal{B} = [\![\boldsymbol{\mathcal{G}}_{\mathcal{A}} \otimes \boldsymbol{\mathcal{G}}_{\mathcal{B}}; \mathbf{U} \odot_1 \mathbf{U}, \cdots, \mathbf{X}_{\mathcal{A}} \boxdot_1 \mathbf{X}_{\mathcal{B}}, \cdots, \mathbf{Z} \odot_1 \mathbf{Z}, \mathbf{A}^{(K+1)}, \cdots, \mathbf{A}^{(N)}, \mathbf{B}^{(N+1)}, \cdots, \mathbf{B}^{(L)}]\!]. \tag{3.19}$$

Similarly, the *arbitrary mode-n Convolution* using Tensor Train representation could be written as

$$\begin{aligned} \mathcal{A} *_n \mathcal{B} = [\![& \mathcal{U} \odot_2 \mathcal{U}, \mathcal{V} \odot_2 \mathcal{V}, \cdots, \mathcal{X}_{\mathcal{A}} \boxdot_2 \mathcal{X}_{\mathcal{B}}, \cdots, \mathcal{Z} \odot_2 \mathcal{Z}, \\ & \mathcal{G}_{\mathcal{A}}^{(K+1)} \otimes \mathbf{I}^{(K+1)}, \cdots, \mathcal{G}_{\mathcal{A}}^{(N)} \otimes \mathbf{I}^{(N)}, \mathcal{G}_{\mathcal{B}}^{(N+1)}, \cdots, \mathcal{G}_{\mathcal{B}}^{(L)}]\!]. \end{aligned} \tag{3.20}$$

where $\otimes$ is Kronecker product, $\odot_n$ is the mode-*n* Khatri-Rao product and $\boxdot_n$ is the partial mode-*n* convolution.

To reduce computer storage and computation time, we could reorder the Tensor Train representation such that

$$\begin{aligned} \boldsymbol{\mathcal{A}} &= [\![\mathcal{G}_{\mathcal{A}}^{(1)}, \cdots, \mathcal{G}_{\mathcal{A}}^{(N-K)} \mathcal{U}, \mathcal{V}, \cdots, \mathcal{X}_{\mathcal{A}}, \cdots, \mathcal{Z}, 1^{(N+1)}, \cdots, 1^{(L)}]\!] \\ \boldsymbol{\mathcal{B}} &= [\![1^{(1)}, \cdots, 1^{(N-K)}, \mathcal{U}, \mathcal{V}, \cdots, \mathcal{X}_{\mathcal{B}}, \cdots, \mathcal{Z}, \mathcal{G}_{\mathcal{B}}^{(N+1)}, \cdots, \mathcal{G}_{\mathcal{B}}^{(L)}]\!] \end{aligned} \tag{3.21}$$

Which would result in arbitrary mode-*n* tensor convolution given by

$$\mathcal{A} *_n \mathcal{B} = [\![\mathcal{G}_{\mathcal{A}}^{(1)}, \cdots, \mathcal{G}_{\mathcal{A}}^{(N-K)}, \mathcal{U} \odot_2 \mathcal{U}, \mathcal{V} \odot_2 \mathcal{V}, \cdots, \mathcal{X}_{\mathcal{A}} \boxdot_2 \mathcal{X}_{\mathcal{B}}, \cdots, \mathcal{Z} \odot_2 \mathcal{Z}, \mathcal{G}_{\mathcal{B}}^{(N+1)}, \cdots, \mathcal{G}_{\mathcal{B}}^{(L)}]\!]. \tag{3.22}$$

The above expression is an arbitrary convolution only on mode-*n*. We could also trivially modify it to simultaneously convolve along multiple modes.

3.4 Chapter Summary

In this chapter, we developed new mathematical formulations for arbitrary mode-*n* tensor convolution that allows convolution of different size tensors along a specific subset of their physical variables. We formulated our arbitrary mode-*n* tensor convolution in both Tucker and Tensor Train decompositions. In the next chapter, we apply our novel arbitrary mode-*n* convolution method to simulate three simple multidimensional *Fourier Optics* problems, i.e., free space propagation, diffraction by an aperture, and imaging using a thin lens.

4. Application of Arbitrary Mode-*n* Convolution in Tensor-based Optical Problems

In this chapter, we simulated three simple Fourier Optics systems; free space propagation, diffraction through an aperture, and imaging using a thin lens. We formulated these systems and simulated them using our tensor-based formulations developed in the previous chapter. Our problem formulations and simulations require *N*-D tensor convolutions, artificially expanding tensor orders, Hadamard products and arbitrary mode-*n* convolutions that we either presented or developed in previous chapters.

4.1 Free Space Optical Propagation and Diffraction under Fresnel Approximation

4.1.1 Free Space Optical Propagation

Free space propagation in the near-field can be computed using the *Fresnel* approximation [19]. A spherical wave is generated at point $U(\xi,\eta,\psi)$ and observed at another point $U(x,y,z)$ as defined by

$$U(x,y,z) = \iiint_{-\infty}^{\infty} U(\xi,\eta,\psi)\frac{e^{jk(z-\psi)}}{j\lambda(z-\psi)}e^{j\frac{k}{2(z-\psi)}[(x-\xi)^2+(y-\eta)^2]}\,d\xi d\eta d\psi \quad . \tag{4.1}$$

The above equation can also be written in a convolution form as

$$\begin{aligned} U(x,y,z) &= \iiint_{-\infty}^{\infty} U(\xi,\eta,\psi)h(x-\xi,y-\eta,z-\psi)\,d\xi d\eta d\psi \\ &= U(x,y,z) *_{x,y,z} h(x,y,z) \end{aligned} \tag{4.2}$$

where the convolution kernel, impulse response of free space propagation, is

$$h(x,y,z) = \frac{e^{jkz}}{j\lambda z}e^{j\frac{k}{2z}(x^2+y^2)}. \tag{4.3}$$

The intensity of the wave $U(x,y,z)$ is obtained by

$$I(x,y,z) = |U(x,y,z)|^2. \tag{4.4}$$

In the particular case where z and ψ are fixed, the free space propagation becomes a two-dimensional convolution

$$\begin{aligned} U(x,y) &= \frac{e^{jk(z-\psi)}}{j\lambda(z-\psi)} \iint_{-\infty}^{\infty} U(\xi,\eta) e^{j\frac{k}{2(z-\psi)}[(x-\xi)^2+(y-\eta)^2]}\, d\xi d\eta \\ &= \iint_{-\infty}^{\infty} U(\xi,\eta) h(x-\xi, y-\eta)\, d\xi d\eta \\ &= U(x,y) *_{x,y} h(x,y). \end{aligned} \tag{4.5}$$

4.1.2 Fresnel Number

The diffraction described in this chapter uses the paraxial approximation which is the same approximation as the Fresnel approximation [20]. This approximation is valid when optical spherical waves could be approximated as paraboloidal waves. In this case, the free space propagation system is required to satisfy the following condition

$$(z-\psi)^3 = \frac{\pi}{4\lambda}[(x-\xi)^2 + (y-\eta)^2]^2_{max} \quad . \tag{4.6}$$

A typical way to determine whether the scalar optical field is a paraboloidal wave or could be further approximated as a plane wave is by using the Fresnel number. In an optical system with source wavelength λ, an aperture width $2a$, and observed at L distance away, the Fresnel number is given by

$$F = \frac{a^2}{L\lambda} \quad . \tag{4.7}$$

When $F \sim 1$ or $F > 1$, the optical field is propagating in the near-field and has a paraboloidal wavefront; hence, equations in the previous section can be used to approximate this system. On the other hand, when $F < 1$, the optical field is propagating in the far-field and has a planar wavefront. This propagation is a special case of Fresnel diffraction, where its quadratic phase factor is assumed to be unity. Therefore, Eq. 4.1 could be approximated using a Fourier Transform. This approximation is called *Fraunhofer* diffraction.

4.1.3 Light Propagation Through a Pupil function

The presence of a pupil function $P(x,y)$ located at a fixed distance z from a propagating light distribution would introduce a change in both its amplitude and phase. The relation between the optical field $U(x,y)$ right before the aperture, located at z, and the field $U(u,v,w)$ observed at distance w from the aperture is given by

$$U(u,v,w) = \frac{e^{jk(w-z)}}{j\lambda(w-z)} \iint_{-\infty}^{\infty} P(x,y)U(x,y)e^{j\frac{k}{2(w-z)}[(u-x)^2+(v-y)^2]}\,dxdy$$

$$= [U(x,y)P(x,y)].*_{u,v}\, h(u,v) \tag{4.8}$$

In other words, the observed field is obtained by convolving the impulse response of propagation with the product the input wave $U(x,y)$ and the pupil function $P(x,y)$. Similar to a clear aperture, a thin lens could be viewed as another form of a pupil phase function

$$P(x,y) = \exp\left[-j\frac{k}{2f}(x^2+y^2)\right] \tag{4.9}$$

where f is lens focal length.

4.1.4 Simple Diffraction Model with a Pupil Function

Let an optical system have a three-dimensional object $U(\xi,\eta,\psi)$, a two-dimensional pupil function $P(x,y)$ at fixed distance z, and an observation region $U(u,v,w)$. This system's diagram is shown in Fig. 4.1, with a square pupil function.

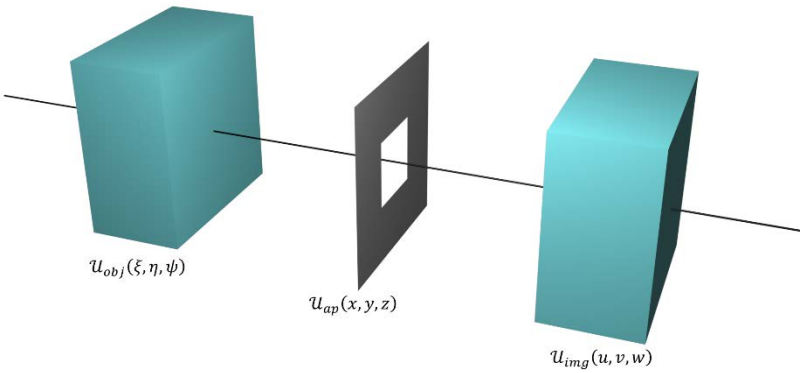

Fig. 4.1. A 3-D object imaging system with a square aperture

We could divide light propagation through this system into three stages: propagation from the object to the pupil, transmission through the aperture, and propagation from the aperture to the

observation plane. In the first stage, light propagates from all points (ξ, η, ψ) to points (x, y) at a fixed distance z right before the aperture:

$$U(x,y) = \iiint_{-\infty}^{\infty} U(\xi,\eta,\psi)\frac{e^{jk(z-\psi)}}{j\lambda(z-\psi)} e^{j\frac{k}{2(z-\psi)}\left[(x-\xi)^2+(y-\eta)^2\right]} d\xi d\eta d\psi \quad . \qquad (4.10)$$

In the second stage, light is transmitted through the aperture, giving

$$U'(x,y) = P(x,y)U(x,y). \qquad (4.11)$$

Since z is fixed, this equation will be effectively two-dimensional.

In the final stage, light propagates from the aperture to the observation plane yields

$$U(u,v,w) = \frac{e^{jk(w-z)}}{j\lambda(w-z)} \iint_{-\infty}^{\infty} U'(x,y)\, e^{j\frac{k}{2(w-z)}\left[(u-x)^2+(v-y)^2\right]} dxdy \quad . \qquad (4.12)$$

By combining the equations of all propagation stages, we obtain

$U(u,v,w) =$

$$= \frac{e^{jk(w-z)}}{j\lambda(w-z)} \iint_{-\infty}^{\infty} \left[\iiint_{-\infty}^{\infty} \frac{e^{jk(z-\psi)}}{j\lambda(z-\psi)} U(\xi,\eta,\psi) e^{j\frac{k}{2(z-\psi)}\left[(x-\xi)^2+(y-\eta)^2\right]} d\xi d\eta d\psi \right]$$

$$P(x,y) e^{j\frac{k}{2(w-z)}\left[(u-x)^2+(v-y)^2\right]} dxdy$$

$$= \frac{e^{jk(w-z)}}{j\lambda(w-z)} \iint \iiint_{-\infty}^{\infty} \frac{e^{jk(z-\psi)}}{j\lambda(z-\psi)} U(\xi,\eta,\psi) e^{j\frac{k}{2(z-\psi)}\left[(x-\xi)^2+(y-\eta)^2\right]}$$

$$P(x,y) e^{j\frac{k}{2(w-z)}\left[(u-x)^2+(v-y)^2\right]} d\xi d\eta d\psi dxdy. \qquad (4.13)$$

The above equation is complicated because of the multiple integrals. Alternatively, we could use convolution operations to simplify Eq. 4.13, yielding

$$U(u,v,w) = \left\{ [U(x,y,z)P(x,y)] *_{x,y,z} h(x,y,z) \right\} *_{x,y} h(u,v,w). \qquad (4.14)$$

Since z is fixed, the last convolution in Eq. 4.14, $*_{x,y}$, is two-dimensional using multiple kernels h of Eq. 4.3, where each kernel has a different propagation distance of $(w-z)$.

4.2 Tensor-Based Formulation of a Diffraction System

4.2.1 Tensor-Based Formulation of a Free Space Propagation

Let the object's optical field be $U(\xi,\eta,\psi)$ and the 3-D observed field be $U(u,v,w)$ be described by the tensors $\mathcal{U}_{obj},\mathcal{U}_{obs}$. From Eq. 4.2, $\mathcal{U}_{obs}$ can be obtained from convolution between $\mathcal{U}_{obj}$ and impulse response $\mathcal{H}_{obj\rightarrow obs}$.

We could rewrite Eq. 4.2 in tensor form as an *N*-D convolution, yielding

$$\mathcal{U}_{obs} = \mathcal{U}_{obj} *_{x,y,z} \mathcal{H}_{obj\rightarrow obs}. \tag{4.15}$$

Given the Tucker Decomposition of $\mathcal{U}_{obj}$ and $\mathcal{H}_{obj\rightarrow obs}$ as

$$\mathcal{U}_{obj} = \left[\!\left[\mathcal{G}_{\mathcal{U}_{obj}}; \boldsymbol{\xi}, \boldsymbol{\eta}, \boldsymbol{\psi} \right]\!\right] \text{ and } \mathcal{H}_{obj\rightarrow obs} = \left[\!\left[\mathcal{G}_{\mathcal{H}_{obj\rightarrow obs}}; \mathbf{U}, \mathbf{V}, \mathbf{W} \right]\!\right] \tag{4.16}$$

we could obtain the observed field by

$$\begin{aligned} \mathcal{U}_{obs} &= \mathcal{U}_{obj} *_{u,v,w} \mathcal{H}_{obj\rightarrow obs} \\ &= \left[\!\left[\mathcal{G}_{\mathcal{U}_{obj}} \otimes \mathcal{G}_{\mathcal{H}_{obj\rightarrow obs}}; \boldsymbol{\xi} \boxdot_1 \mathbf{U}, \boldsymbol{\eta} \boxdot_1 \mathbf{V}, \boldsymbol{\psi} \boxdot_1 \mathbf{W} \right]\!\right]. \end{aligned} \tag{4.17}$$

Similarly, let the Tensor Train decomposition of $\mathcal{U}_{obj}$ and $\mathcal{H}_{obj\rightarrow obs}$ be

$$\mathcal{U}_{obj} = \left[\!\left[\underline{\xi}, \underline{\eta}, \underline{\psi} \right]\!\right] \text{ and } \mathcal{H}_{obj\rightarrow obs} = [\![\mathcal{U}, \mathcal{V}, \mathcal{W}]\!], \tag{4.18}$$

the tensor observed light is then given as

$$\begin{aligned} \mathcal{U}_{obs} &= \mathcal{U}_{obj} *_{u,v,w} \mathcal{H}_{obj\rightarrow obs} \\ &= \left[\!\left[\underline{\xi} \boxdot_2 \mathcal{X}, \underline{\eta} \boxdot_2 \mathcal{Y}, \underline{\psi} \boxdot_2 \mathcal{Z} \right]\!\right]. \end{aligned} \tag{4.19}$$

4.2.2 Tensor-Based Formulation of Diffraction by an Aperture

There are three-stages of light propagation in this example. We start by denoting the object's optical field $U(\xi,\eta,\psi)$, optical field at the pupil $U(x,y,z)$ and output optical field $U(u,v,w)$ as $\mathcal{U}_{obj},\mathcal{U}_{pupil}$ and $\mathcal{U}_{obs}$. The first stage of propagation from the object to the pupil is obtained by an *N*-D convolution described in the previous section as

$$\mathcal{U}_{pupil} = \mathcal{U}_{obj} *_{x,y,z} \mathcal{H}_{obj\rightarrow pupil} \tag{4.20}$$

where

$$\mathcal{H}_{obj\rightarrow pupil} = \frac{e^{jkz}}{j\lambda \mathbf{z}} e^{j\frac{k}{2z}(\mathbf{x}^2+\mathbf{y}^2)} \tag{4.21}$$

is an impulse response tensor with different propagating distances $z - \psi$ in the third mode. Since wave $\mathcal{U}_{pupil}$ is at a fixed distance z, summing all the optical fields would result in the total field at the distance z. Therefore, we sum $\mathcal{U}_{pupil}$ over the third mode, yielding $\mathbf{U}_{pupil}$, a 2nd order tensor. The second propagation stage is when the field is transmitted through the aperture, which is described as a Hadamard product

$$\mathbf{U}'_{pupil} = \mathbf{P} \circledast \mathbf{U}_{pupil}. \tag{4.22}$$

Lastly, the third system stage is the propagation from the pupil to the observation region

$$\mathcal{U}_{obs} = \mathbf{U}'_{pupil} *_{u,v} \mathcal{H}_{pupil\rightarrow obs}. \tag{4.23}$$

This stage required an arbitrary mode-1,2 convolution since $\mathcal{U}_{obs}$ is a 3rd-order tensor with mode u, v, w; however, the convolution is along modes u, v only. In addition, $\mathbf{U}'_{pupil}$ is a 2nd-order tensor, so one would need to artificially expand its order (see Appendix A) before applying the convolution. Combining all three stages gives

$$\mathcal{U}_{obs} = \left\{\mathbf{P} \circledast \sum_z \left[\mathcal{U}_{obj} *_{x,y,z} \mathcal{H}_{obj\rightarrow pupil}\right]\right\} *_{u,v} \mathcal{H}_{pupil\rightarrow obs} \quad . \tag{4.24}$$

The intensity distribution is

$$\begin{aligned} \mathcal{I}_{obs} &= |\mathcal{U}_{obs}|^2 \\ &= \left\{\left[\mathbf{P} \circledast \sum_z \left[\mathcal{U}_{obj} *_{x,y,z} \mathcal{H}_{obj\rightarrow pupil}\right]\right] *_{u,v} \mathcal{H}_{pupil\rightarrow obs}\right\} \\ &\quad \left\{\left[\mathbf{P} \circledast \sum_z \left[\mathcal{U}_{obj} *_{x,y,z} \mathcal{H}_{obj\rightarrow pupil}\right]\right] *_{u,v} \mathcal{H}_{pupil\rightarrow obs}\right\}^{\mathrm{H}} \end{aligned} \tag{4.25}$$

where $(\cdot)^{\mathrm{H}}$ denoted conjugate transpose or Hermitian transpose.

4.2.3 Formulation of Diffraction by an Aperture using Tucker Decomposition

Given the Tucker Decomposition of $\mathcal{U}_{obj}$, $\mathcal{U}_{pupil}$ and $\mathcal{U}_{obs}$ as

$$\mathcal{U}_{obj} = \left[\!\left[\mathcal{G}_{\mathcal{U}_{obj}}; \boldsymbol{\xi}, \boldsymbol{\eta}, \boldsymbol{\psi} \right]\!\right], \mathcal{U}_{pupil} = \left[\!\left[\mathcal{G}_{\mathcal{U}_{pupil}}; \mathbf{X}, \mathbf{Y}, \mathbf{Z} \right]\!\right] \text{ and } \mathcal{U}_{obs} = \left[\!\left[\mathcal{G}_{\mathcal{U}_{obs}}; \mathbf{U}, \mathbf{V}, \mathbf{W} \right]\!\right]. \quad (4.26)$$

The convolution kernels are decomposed as

$$\mathcal{H}_{obj\rightarrow pupil} = \left[\!\left[\mathcal{G}_{\mathcal{H}_{obj\rightarrow pupil}}; \mathbf{X}, \mathbf{Y}, \mathbf{Z} \right]\!\right] \text{ and } \mathcal{H}_{pupil\rightarrow obs} = \left[\!\left[\mathcal{G}_{\mathcal{H}_{pupil\rightarrow obs}}; \mathbf{U}, \mathbf{V}, \mathbf{W} \right]\!\right]. \quad (4.27)$$

Lastly, the pupil function or aperture is

$$\mathbf{P} = \left[\!\left[\mathbf{G}_{pupil}; \mathbf{X}, \mathbf{Y} \right]\!\right]. \quad (4.28)$$

Then Eq. 4.20 can be written as

$$\mathcal{U}_{pupil} = \mathcal{U}_{obj} *_{x,y,z} \mathcal{H}_{obj\rightarrow pupil} = \left[\!\left[\mathcal{G}_{\mathcal{U}_{obj}} \otimes \mathcal{G}_{\mathcal{H}_{obj\rightarrow pupil}}; \boldsymbol{\xi} \boxdot_1 \mathbf{X}, \boldsymbol{\eta} \boxdot_1 \mathbf{Y}, \boldsymbol{\psi} \boxdot_1 \mathbf{Z} \right]\!\right]. \quad (4.29)$$

an *N*-D convolution. We then sum the tensor $\mathcal{U}_p$ over the third mode to obtain total waves at the aperture. The second stage, Eq. 4.22, is

$$\begin{aligned} \mathbf{U'}_{pupil} &= \mathbf{P} \circledast \mathbf{U}_{pupil} \\ &= \left[\!\left[\mathbf{G}_{pupil} \otimes \mathbf{G}_{\mathbf{U}_{pupil}}; \mathbf{X} \odot_1 \mathbf{X}, \mathbf{Y} \odot_1 \mathbf{Y} \right]\!\right]. \\ &= \left[\!\left[\mathbf{G}_{\mathbf{U'}_{pupil}}; \mathbf{X}, \mathbf{Y} \right]\!\right]. \end{aligned} \quad (4.30)$$

Lastly, Eq. 4.23 uses arbitrary mode-1,2 convolution and artificially expands tensor $\mathbf{U'}_p$ order gives

$$\begin{aligned} \mathcal{U}_{obj} &= \mathbf{U'}_{pupil} *_{u,v} \mathcal{H}_{pupil\rightarrow obs} \\ &= \left[\!\left[\boldsymbol{\mathcal{G}}_{\mathbf{U'}_{pupil}} \otimes \mathcal{G}_{\mathcal{H}_{pupil\rightarrow obs}}; \mathbf{X} \boxdot_1 \mathbf{U}, \mathbf{Y} \boxdot_1 \mathbf{V}, \mathbf{W}_{\mathcal{H}_{pupil\rightarrow obs}} \right]\!\right] \end{aligned} \quad (4.31)$$

where $\boldsymbol{\mathcal{G}}_{\mathbf{U'}_{pupil}} \in \mathbb{C}^{I_x \times I_y \times 1}$.

4.2.4 Formulation of Diffraction by an Aperture using Tensor Train Decomposition

Given Tensor Train decomposition of $\mathcal{U}_{obj}$, $\mathcal{U}_p$ and $\mathcal{U}_{obs}$ as

$$\mathcal{U}_{obj} = [\![\underline{\xi}, \underline{\eta}, \underline{\psi}]\!], \mathcal{U}_{pupil} = [\![\mathcal{X}, \mathcal{Y}, \mathcal{Z}]\!] \text{ and } \mathcal{U}_{obs} = [\![\mathcal{U}, \mathcal{V}, \mathcal{W}]\!]. \tag{4.32}$$

The convolution kernels are decomposed as

$$\mathcal{H}_{obj\to pupil} = [\![\mathcal{X}, \mathcal{Y}, \mathcal{Z}]\!] \text{ and } \mathcal{H}_{pupil\to obs} = [\![\mathcal{U}, \mathcal{V}, \mathcal{W}]\!]. \tag{4.33}$$

The pupil function or aperture is

$$\mathbf{P} = [\![\mathcal{X}, \mathcal{Y}]\!]. \tag{4.34}$$

Then the first stage is an *N*-D convolution in Eq. 4.20 can be written as

$$\mathcal{U}_{pupil} = \mathcal{U}_{obj} *_{x,y,z} \mathcal{H}_{obj\to pupil} = [\![\underline{\xi} \boxdot_2 \mathcal{X}, \underline{\eta} \boxdot_2 \mathcal{Y}, \underline{\psi} \boxdot_2 \mathcal{Z}]\!]. \tag{4.35}$$

Similar to the TKD approach, the $\mathcal{U}_{pupil}$ needs to sum over the third mode gives $\mathbf{U}_{pupil}$. The second stage, Eq. 4.22, is then described as

$$\mathbf{U}'_{pupil} = \mathbf{P} \circledast \mathbf{U}_{pupil} = [\![\mathcal{X} \odot_2 \mathcal{X}, \mathcal{Y} \odot_2 \mathcal{Y}]\!]. \tag{4.36}$$

The waves at the image region, using arbitrary mode-1,2 convolution and artificially expanding tensor order, Eq. 4.23 is described as

$$\mathcal{U}_{obs} = \mathbf{U}'_{pupil} *_{u,v} \mathcal{H}_{pupil\to obs} = [\![\mathcal{X} \boxdot_2 \mathcal{U}, \mathcal{Y} \boxdot_2 \mathcal{V}, \mathcal{W}_{\mathcal{H}_{pupil\to obs}}]\!]. \tag{4.37}$$

4.3 Numerical Simulation Results

4.3.1 Simulation of 3-D Free Space Light Propagation

In this example, we propagated a three-dimensional object $U(\xi, \eta, \psi)$ to a two-dimensional screen at $U(x, y, 0)$ where $\psi < 0$. Using Eq. 4.15, we obtain

$$\mathcal{U}_{obs} = \mathcal{U}_{obj} *_{x,y,z} \mathcal{H}_{obj\to obs} \tag{4.38}$$

where mode-3 of the convolution kernel contains a different propagation length of $-\psi$. This setup is shown in Fig 4.2. Three methods have been compared, the direct i.e., full-tensor, the Tucker Decomposition, and the Tensor Train approach. We have used tensor MATLAB toolboxes from Sandia National Laboratories and Oseledets. The object $\mathcal{U}_{obj} \in \mathbb{C}^{I_x \times I_y \times I_z}$ are complex electric fields pre-defined using the Finite-Difference Frequency-Domain method to compute the plane waves' behavior in a three-dimensional object.

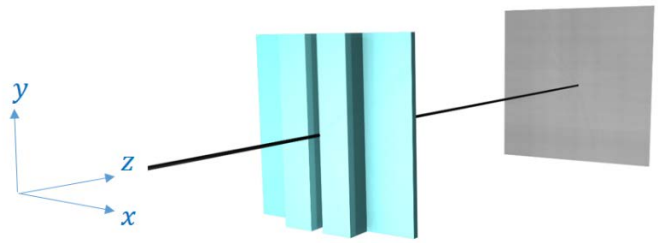

Fig. 4.2. A system of free space propagation of a 3-D object onto a 2-D plane of observation

The sample object has unity permeability and is homogenous along the *y*-axis. This object's permittivity and the field distribution along the *x*-*z* axes are shown in Fig. 4.3. This object is made of glass with a refractive index of 1.56, and it is placed 1.88 cm at $z = 0$, i.e., before the observation plane.

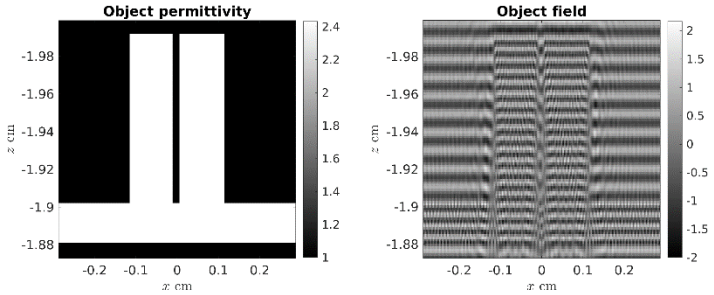

Fig. 4.3. Our 3-D object and its field distribution along the *x*-*z* plane

Furthermore, Fig. 4.4 shows planes inside the object along *x*-*y* axes at different *z* distances. The source wavelength is 100 μm, and the object's half width is 2.87 mm; therefore, the associated Fresnel number is 4.4.

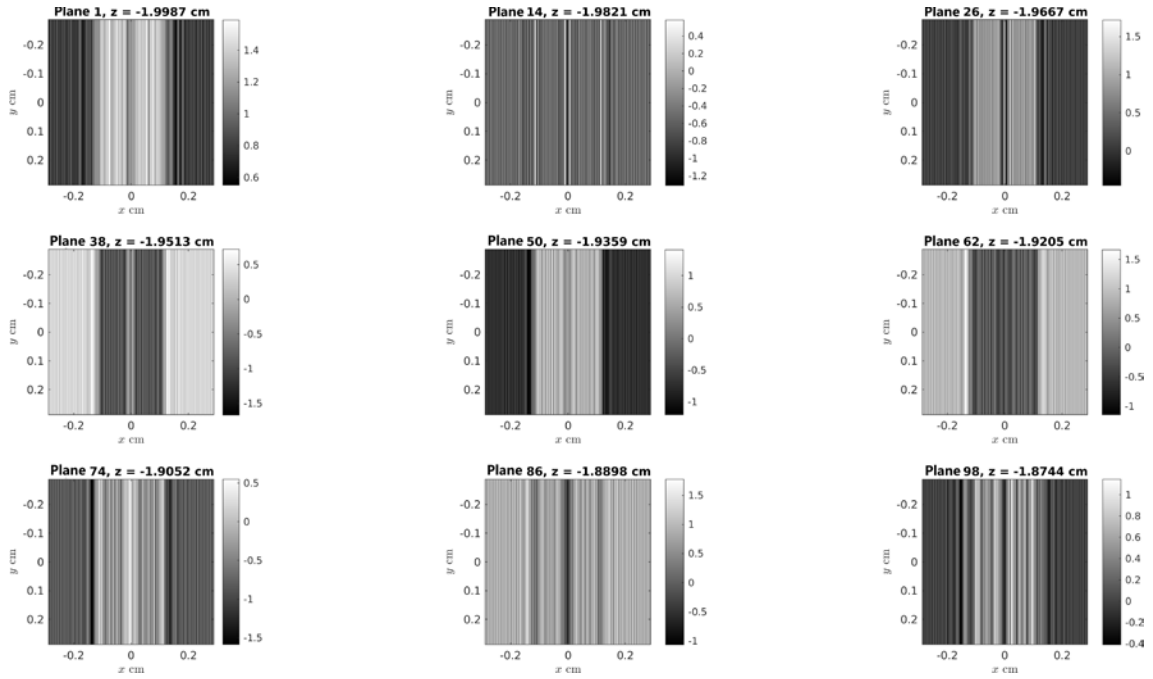

Fig. 4.4. Planes of field distributions in the object at different *z* distances

Since convolution of tensors in Tucker representation requires the Kronecker product of tensor cores, the computation becomes impractical without rank approximation. The test used rank approximations to decompose object tensor of 449x449x99 into smaller tensor cores of 15x15x20. The full tensor Kronecker product would require as high as 8.64 PB for complex entries; with the decomposition, the storage needed is only 308.99 MB. We had tested to our maximum storage capacity at 16 GB (40x40x20); however, the error due to representing the original tensor by a Tucker Decomposition is hardly improving. We used these smaller dimensions 15x15x20 for all the tensor cores in Tucker Decomposition format.

After the convolution, the product tensor dimension is 897x897x197, representing a larger space in the space domain, as shown in Fig. 4.5 and Fig. 4.6. The $\mathcal{U}_{obj}(:,:,1)$ or the object field's first plane will travel the longest distance hence the intensity will significantly drop according to the inverse-square law while $\mathcal{U}_{obj}(:,:,197)$ has the highest intensity. Fig. 4.5 shows the intensity contribution from different object planes along the propagation direction, and Fig. 4.6 shows the observation plane's total intensity at distance $z = 0$. Our results show that all three methods give very similar intensity distributions.

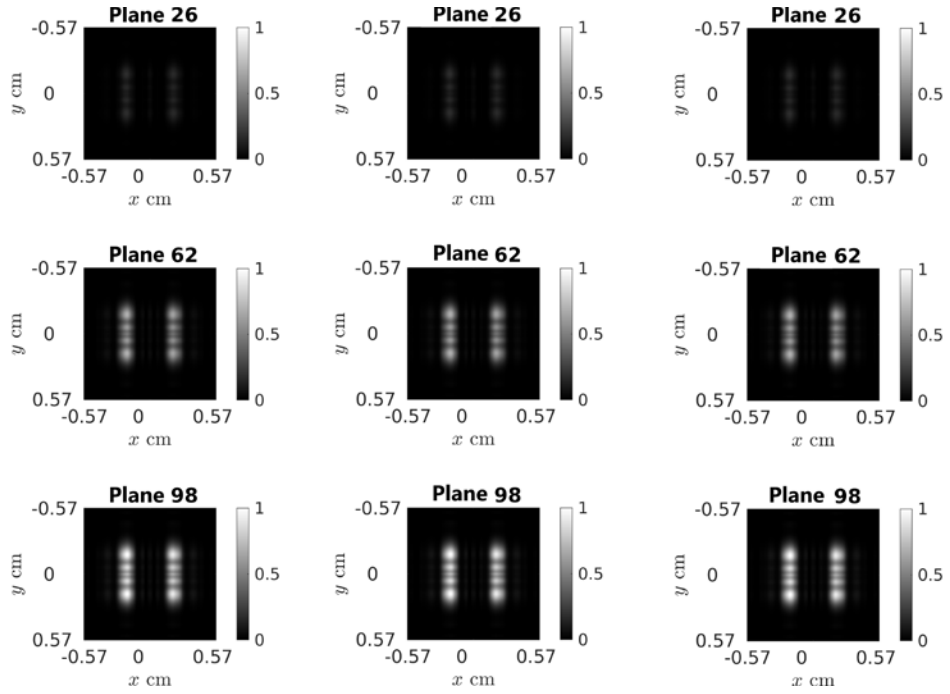

Fig. 4.5. Observed normalized intensities contributions at different planes using direct (1st column) Tucker Decomposition (2nd column) Tensor Train (3rd column) approaches

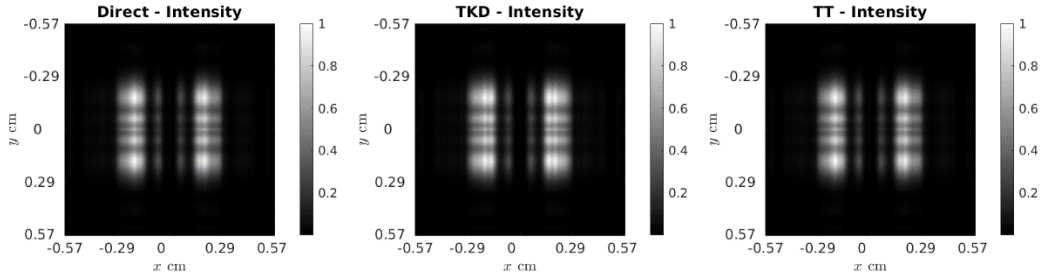

Fig. 4.6. Observed normalized intensities from direct (left) Tucker Decomposition (middle) Tensor Train (right) approaches

The error is computed by obtaining the norm of the absolute differences between corresponding entries of each decomposition method (Tucker Decomposition and Tensor Train) and the full-tensor. The observed complex field norm error of Tucker format is 4.7619 in contrast with the Tensor Train 7.9477×10^{-11}.

The computational time of directly computing full tensors is 5 hours, tensor in the Tucker representation is 2 minutes, and tensor in the Tensor Train representation is 1 minute. The decomposed and reconstructed time is included in the recorded times. The storage needs to store output from direct computation is 1.19 GB due to convolution increasing the output size. Tucker representation storage is highly dependent on rank approximation; our choice needs 316.17 MB to store product core and factor matrices. In the case of 40x40x20, 15.28 GB is required. Lastly, Tensor Train needs only 83.36 MB. Comparisons of different approaches using an object having 449x449x99 samples are shown in Table 4.1.

Methods	**Compute Dimensions**	**Complex Field Norm Error**	**Computational Time**
Direct	Full	-	5 hours
Tucker Decomposition	15x15x20	4.76	2 minutes
Tensor Train	Full	7.947×10^{-11}	1 minute

Table 4.1. Comparison of free space propagation using full-tensor, Tucker Decomposition, and Tensor Train approaches

4.3.2 Simulation of Optical Diffraction by an Aperture

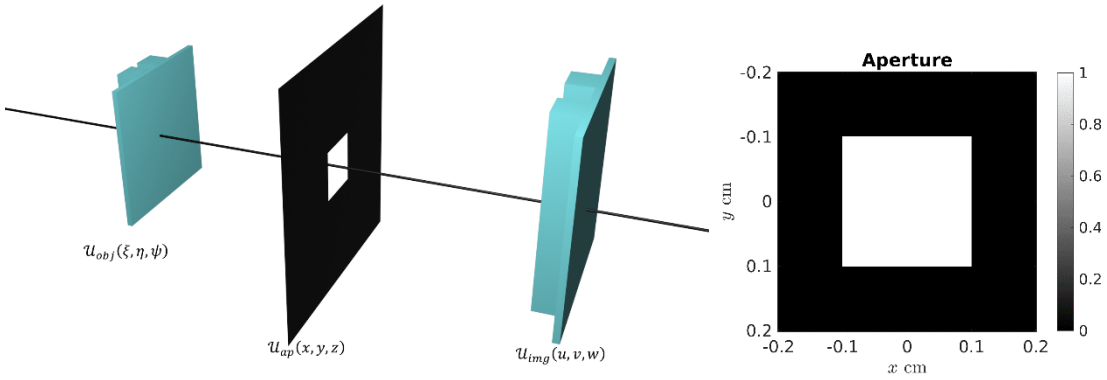

Fig. 4.7. A 3-D imaging system with an aperture

We have tested the tensor-based imaging system described in Sections 4.2.2 - 4.2.4, shown on the left of Fig. 4.7, which is an extension of the previous numerical example. The observation region width is 4 mm along *x*-*y* axes or 70% of the object width. A square aperture **P** has a 2 mm opening gap placed at $z = 0$, as shown on the right of Fig. 4.7. The observation region is 1.5 – 3.5 cm after the aperture.

Multiple convolutions are computational expensive since the dimensions increase with every operation. Suppose that $\mathcal{H}_{obj\rightarrow pupil}\mathbb{C}^{J_x\times J_y\times J_z}$; in the first step, the computation requires 3-D convolution of tensors. Thus, the product will have dimensions of $I_x + J_x - 1 \times I_y + J_y - 1 \times I_z + J_z - 1$. The second step involved the sum along mode-3 and Hadamard product; the tensor order is reduced by one, and the dimensions become $I_x + J_x - 1 \times I_y + J_y - 1$. The last step is another convolution; suppose that $\mathcal{H}_{pupil\rightarrow obs}\mathbb{C}^{K_x\times K_y\times K_z}$ then the final product dimensions are $I_x + J_x + K_x - 2 \times I_y + J_y + K_y - 2 \times K_z$. Due to storage limitations, we reduced the observation region by cropping the convolution product along the *x*-*y* axis to a smaller dimension $S_x \times S_y \times K_z$ where $S_x = 0.7 * I_x, S_y = 0.7 * I_y$.

The first propagation from the object to the aperture is shown in Fig. 4.6. Then the field is multiplied (Hadamard product) by the aperture, resulting in only the field distributions in the opening gap being passed. The last step is two-dimensional field propagation after the aperture to different distances *z* at the observation regions using arbitrary mode-1,2 convolution described in Eq. 4.23 in full-tensor, Eq. 4.31 in Tucker Decomposition, in Eq. 4.37 for Tensor Train representations.

The normalized intensities along the *x*-*y* axes in the observation region with different distances *z* are shown in Fig. 4.8. Similarly, the normalized intensities along the *x*-*z* axes in the observation region using different approaches are shown in Fig. 4.9. The normalized intensity distributions obtained using the direct and Tensor Train approaches give similar results. However, the Tucker Decomposition approach has significant error due to computational limitations that resulted in a reduction in the core tensor size through rank approximation. The error has accumulated from the decomposition of each entry since the first propagation and results in a high field error at the observation region.

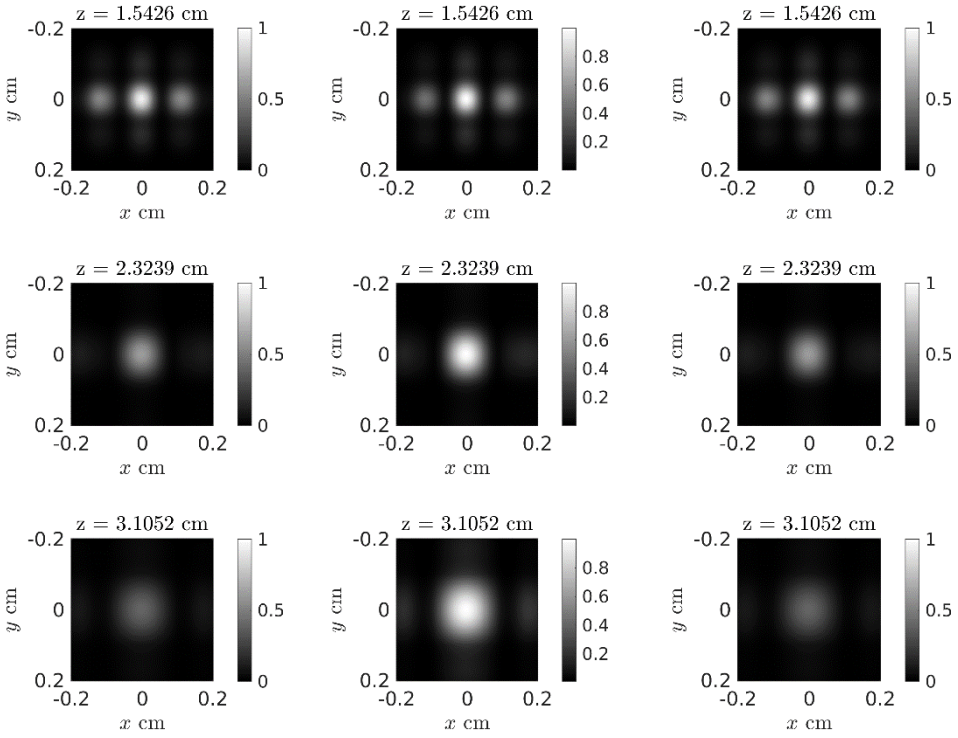

Fig. 4.8. Observed normalized intensities from different *z* distances using direct (1st column) Tucker Decomposition (2nd column) Tensor Train (3rd column) approaches

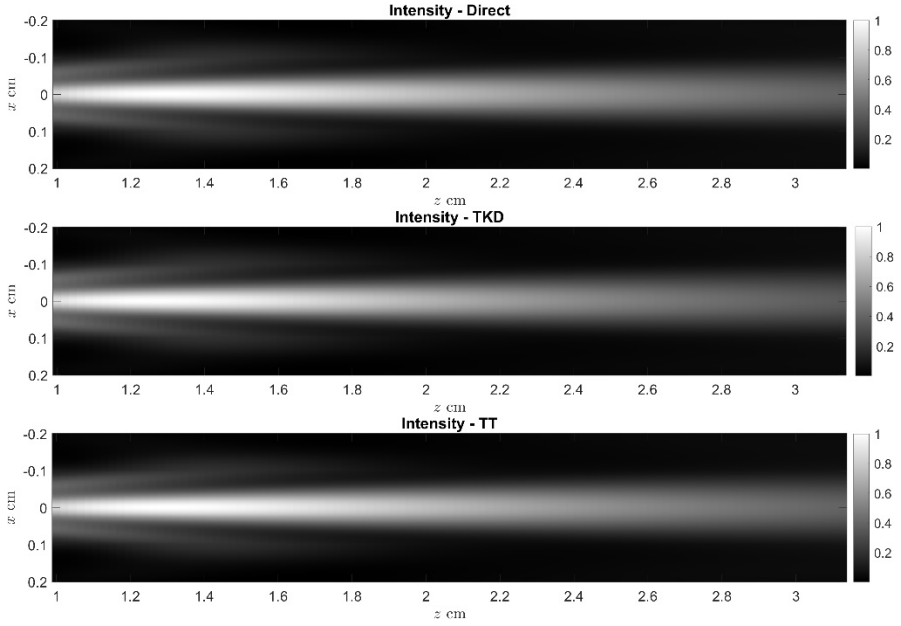

Fig. 4.9. Normalized intensity along the *x*-*z* axis at the observation region using direct (top) Tucker Decomposition (middle) Tensor Train (bottom) approaches

λ_{src} (μm)	Object Dimension	Method	Time (H:MM: SS)	Complex Field Norm Error
150	$299 \times 299 \times 70$	Direct	0:35:38	-
		Tucker Decomposition	0:00:23	1.9165×10^{-4}
		Tensor Train	0:00:04	4.2106×10^{-15}
120	$375 \times 375 \times 85$	Direct	3:25:34	-
		Tucker Decomposition	0:01:20	3.6169×10^{-3}
		Tensor Train	0:00:10	1.4068×10^{-14}
100	$449 \times 449 \times 99$	Direct	5:04:15	-
		Tucker Decomposition	0:02:26	9.3051×10^{-3}
		Tensor Train	0:01:44	1.1237×10^{-13}

Table 4.2. Computation time and error from different approaches

The norm error (norm of pointwise differences) of complex field distribution and computation time are presented in Table 4.2. The tests are conducted in the same machine with various problem sizes, where the times are included from the first propagation in the previous example. The second propagation used significantly less time than the *N*-D Convolution in the first propagation. For the object dimension $449 \times 449 \times 99$, direct computation spent 32.4 seconds, and both Decomposition approaches consume 3 seconds.

4.3.3 Optical Imaging using a Thin Lens

In the previous section, the observed image is out of focus; we fix it by adding a lens with a focal length of 2 cm instead of an aperture. The first propagation is the same in Section 4.3.1; after that, the wave passes the lens and propagates to the observation region at 1.5 – 2.65 cm from the lens. The observed intensity is shown in Fig. 4.10 and Fig. 4.11. Similar to the previous example, images observed at a further distance have lesser intensity than those closer to the lens. The image observed at 2 cm is in focus and noticeably resembles the object.

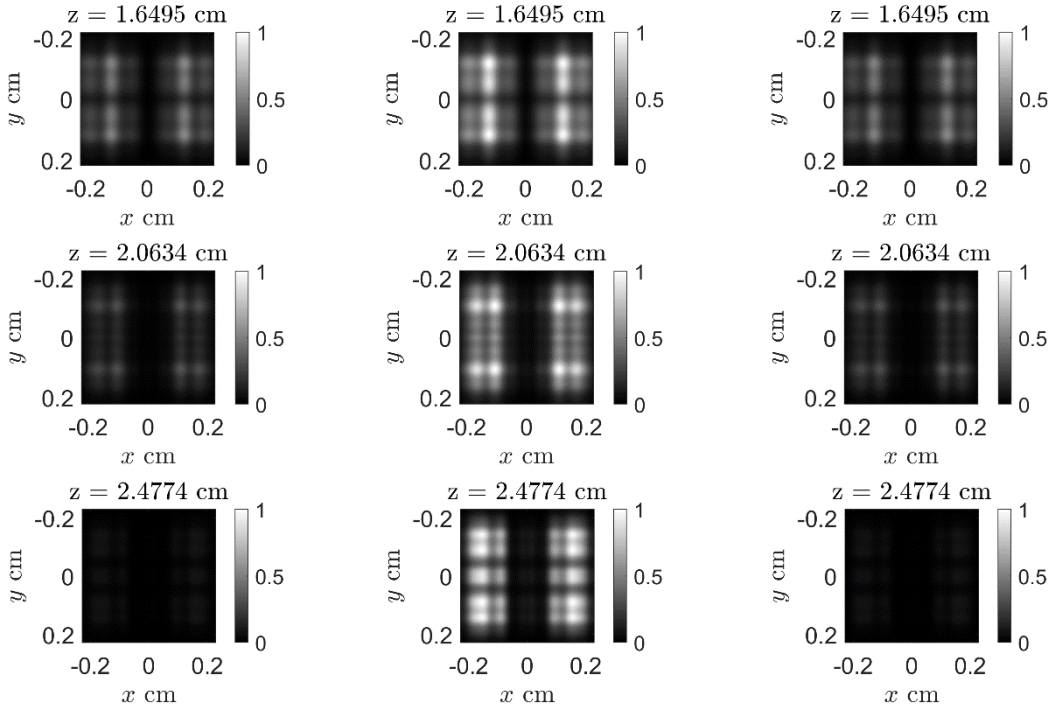

Fig. 4.10. Observed normalized intensities from different *z* distances using direct (1st column) Tucker Decomposition (2nd column) Tensor Train (3rd column) approaches

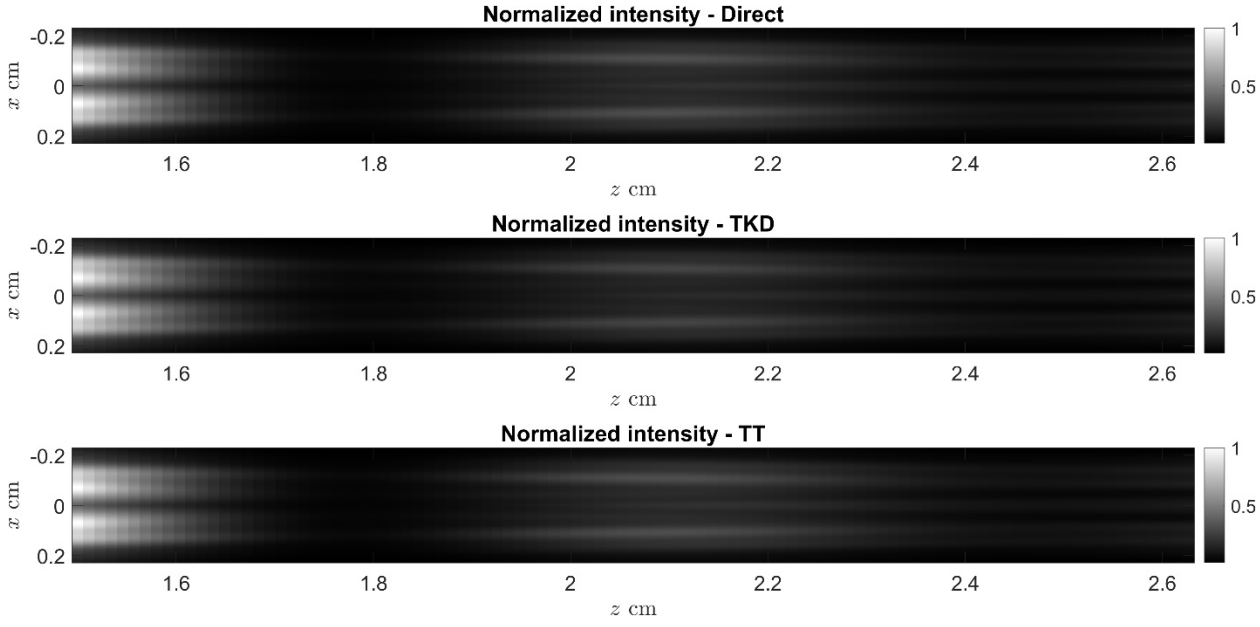

Fig. 4.11. Normalized intensity along the *x*-*z* axis at the observation region using direct (top) Tucker Decomposition (middle) Tensor Train (bottom) approaches

Comparing images from three different approaches, the direct and the Tensor Train approach results are indistinguishable. However, the Tucker Decomposition approach shows a noticeable error. This is because our computationally limited Tucker Decomposition approach resulted in a low rank core tensor.

The pupil function changes from binaries (aperture) into complex numbers (phase), resulting in longer computation time. Full-tensor computation, Tucker and Tensor Train approaches consume 58, 5, and 3.4 seconds, respectively, for the second propagation. The norm error (norm of pointwise differences) of complex field at the observed region between full-tensor and Tucker Decomposition computation is 0.014719. Conversely, the norm error between full-tensor and Tensor Train computation is 4.7574×10^{-13}.

In the previous chapter, we showed that Tucker Decomposition suffers less from round-off error without rank approximation. However, in practical with limited storage, the rank approximation is unavoidable. For Optical Problems, accuracy is essential; thus, with storage limitations, the Tucker Decomposition approach is unsuitable.

4.4 Chapter Summary

In this chapter, we simulated three simple Fourier Optics systems; free space propagation, diffraction through an aperture, and imaging using a thin lens. We formulated these systems and simulated them using our tensor-based formulations developed in the previous chapter. Our problem formulations and simulations require *N*-D tensor convolutions, artificially expanding tensor orders, Hadamard products and arbitrary mode-*n* convolutions that we either presented or developed in previous chapters. Our results showed that Tensor Train required the least computational time and computer storage while obtaining results with an excellent norm of error.

5. Conclusions and Future Work

5.1 Conclusions

Fourier Optics generally involve multidimensional functions that could require prohibitive computational resources to either simulate or manipulate. In this thesis, to address this computational complexity, we introduced tensors analysis as an efficient way to linearly process multidimensional functions. Typical operations in Fourier Optics are convolution and Hadamard products. In this thesis, we focused on tensor convolutions. This operation has been well defined in the tensor literature; however, three assumptions are made on the tensors to be convolved; 1) both tensors must have the same order; 2) corresponding modes of tensors represent the same physical variables; 3) the convolution is applied along every mode. These assumptions would make it difficult to use in physical Fourier Optics problems. We overcome these by generalizing this *N*-D tensor convolution to allow arbitrary mode-*n* convolutions.

We started by imposing a physical interpretation to tensors, and then formulated a novel approach for arbitrary mode-*n* convolution of physical tensors using three different tensor representations: full-tensor, Tucker Decomposition, and Tensor Train Decomposition. Lastly, we simulated three simple Fourier Optics systems; free space propagation, diffraction through an aperture, and imaging using a thin lens. We formulated these systems and simulated them using our tensor-based formulations developed in the previous chapter. Our problem formulations and simulations require *N*-D tensor convolutions, artificially expanding tensor orders, Hadamard products and arbitrary mode-*n* convolutions that we either presented or developed in previous chapters. Our results showed that Tensor Train, compared to using full-tensor representation or Tucker Decomposition, required the least computational time and computer storage while obtaining results with an excellent norm of error.

5.2 Future Work

This thesis addressed the possibilities and advantages of solving *Fourier Optics* problems using tensor analysis. We intend to extend our research to develop *Tensor Optics* in the Tensor Train representation in order to reduce computation complexities of both forward and inverse problems. *Tensor Optics* would possibly allow us to analyze full-wave optical propagation in large

inhomogeneous media, e.g., tissue. One possible way to further overcome the curse of dimensionality inherent in solving large size optical problems is to use sparse signal (tensor) representations.

References

[1] I. J. Cox and C. J. R. Sheppard. (1986), "Information capacity and resolution in an optical system." J. Opt. Soc. Am. A 3, 1152-1158

[2] R.E. Bellman. (1961), "Adaptive Control Processes." Princeton University Press, Princeton, NJ.

[3] Yipeng Liu, (2022), Tensors for Data Processing, Academic Press, ISBN 9780128244470. https://doi.org/10.1016/B978-0-12-824447-0.00003-0.

[4] Papalexakis E. E., Faloutsos C., and Sidiropoulos D. N., (2016), Tensors for Data Mining and Data Fusion: Models, Applications, and Scalable Algorithms. ACM Trans. Intell. Syst. Technol. 8, 2, Article 16 (March 2017), 44 pages. https://doi.org/10.1145/2915921

[5] Wickramasingha I., Sobhy M., Elrewainy A., and Sherif S. S., "Tensor least angle regression for sparse representations of multidimensional signals," Neural Comput., vol. 32, no. 9, pp. 1697–1732, Sep. 2020,https://doi.org/10.1162/neco_a_01304.

[6] Wickramasingha I., Sobhy M., and Sherif S. S., "Sparsity in Bayesian Signal Estimation," in Bayesian Inference, vol. 37, no. 2, J. P. Tejedor, Ed. InTech, 2017, https://doi.org/10.5772/intechopen.70529.

[7] L. R. Tucker. (1966). "Some mathematical notes on three-mode factor analysis," Psychometrika, vol. 31, no. 3, pp. 279–311.

[8] L. De Lathauwer, B. De Moor, and J. Vandewalle. (2000). "A multilinear singular value decomposition." SIAM Journal on Matrix Analysis and Applications, 21(4):1253–1278.

[9] Hackbusch, Wolfgang & Kühn, S. (2009). "A New Scheme for the Tensor Representation." Journal of Fourier Analysis and Applications. 15. 706-722. 10.1007/s00041-009-9094-9.

[10] Oseledets, Ivan. (2011). "Tensor-Train Decomposition." SIAM J. Scientific Computing. 33. 2295-2317. 10.1137/090752286.

[11] Cichocki, Andrzej. (2013). "Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions."

[12] Dolgov, SV & Savostyanov, DV. (2014). "Alternating minimal energy methods for linear systems in higher dimensions." SIAM Journal on Scientific Computing, vol. 36, no. 5, pp. A2248-A2271. https://doi.org/10.1137/140953289

[13] Cichocki, Andrzej & Lee, Namgil & Oseledets, Ivan & Phan, Anh-Huy & Zhao, Qibin & Mandic, Danilo. (2016). "Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions". Foundations and Trends® in Machine Learning. 9. 249-429. 10.1561/2200000059.

[14] I. Wickramasingha. (2021). "Computationally Efficient Methods for Sparse Tensor Signal Processing"

[15] Kolda, TG, & Bader, B.W. (2009). "Tensor Decompositions and Applications." *SIAM Rev., 51,* 455-500.

[16] Rabanser, Stephan & Shchur, Oleksandr & Günnemann, Stephan. (2017). "Introduction to Tensor Decompositions and their Applications in Machine Learning."

[17] Lee, Namgil & Cichocki, Andrzej. (2014). "Fundamental Tensor Operations for Large-Scale Data Analysis in Tensor Train Formats."

[18] Cichocki, Andrzej & Mandic, Danilo & Phan, Anh-Huy & Caiafa, Cesar & Zhou, Guoxu & Zhao, Qibin & Lathauwer, Lieven. (2014). "Tensor Decompositions for Signal Processing Applications From Two-way to Multiway Component Analysis." Signal Processing Magazine, IEEE. 32. 10.1109/MSP.2013.2297439.

[19] Saleh, Bahaa E A and Teich, Malvin Carl, (1991). Fundamentals of photonics; 1st ed., Wiley, New York.

[20] J. W. Goodman, “Introduction to Fourier Optics,” 3rd Edition, Roberts & Co., Greenwood Village, 2005

Appendix A

A.1 Artificially expanded tensor order

A.1.1 Tucker Decomposition of a Tensor with an Artificially Expanded Order

If we wish to broaden the tensor $\mathcal{X}$ to a new tensor with $(N+1)$-order, we could add another order with dimension 1 to both the tensor $\mathcal{X}$ and its core which gives $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times I_{N+1}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times 1}$ and $\mathcal{G}_{\mathcal{X}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N \times R_{N+1}} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N \times 1}$.

$$\mathcal{X} = [\![\mathcal{G}_{\mathcal{X}}; \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(N)}, \mathbf{X}^{(N+1)}]\!]. \tag{A.1}$$

The factor matrices of order 1 to N are the same; however, the $(N+1)$-order factor matrix is $\mathbf{X}^{(N+1)} \in \mathbb{R}^{I_{N+1} \times R_{N+1}} \in \mathbb{R}^{1 \times 1}$ a scalar. This scalar will be one in case of no scaling in the $(N+1)$ mode. Therefore, we can rewrite the above equation as

$$\boldsymbol{\mathcal{X}} = [\![\boldsymbol{\mathcal{G}}_{\mathcal{X}}; \mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(N)}, 1]\!]. \tag{A.2}$$

We will use a bolded tensor to denote an artificially expanded tensor. The artificially expanding could also occur in any arbitrary mode-n.

A.1.2 Tensor Train Decomposition with an Artificially Expanded Order

Similar to the approach discussed above for TKD. We could artificially expand the order of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ in tensor train format to have $(N+1)$-order as

$$\begin{aligned} \mathcal{X} &\cong \mathcal{G}^{(1)} \times_3^1 \mathcal{G}^{(2)} \times_3^1 \mathcal{G}^{(3)} \times_3^1 \cdots \times_3^1 \mathcal{G}^{(N)} \times_3^1 \mathcal{G}^{(\mathrm{N+1})} \\ &= [\![\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \mathcal{G}^{(3)}, \cdots, \mathcal{G}^{(N)}, \mathcal{G}^{(\mathrm{N+1})}]\!] \end{aligned} \tag{A.3}$$

However, unlike TKD, TT is a train. Thus, the order of the core $\mathcal{G}^{(n)}$ varies depending on its position on the train. There are two cases to be considered

1. Expanding the first or last of the tensor train cores
2. Expanding between the tensor train cores

The first case is when we add a dimension before $\mathcal{G}^{(1)} \in \mathbb{R}^{1\times I_1\times R_1}$ or after matrix $\mathcal{G}^{(N)} \in \mathbb{R}^{R_{N-1}\times I_N\times 1}$. We could add ranks R_0 in $\mathcal{G}^{(1)}$ and R_N in $\mathcal{G}^{(N)}$ instead of dimension one, then add a tensor $\mathcal{G}^{(0)}$ or $\mathcal{G}^{(N+1)} \in \mathbb{R}^{1\times 1}$ at the edge of the train, this will always make the new order just a scalar. The expansion is straightforward and will not change the number of elements in the train.

$$\begin{aligned}\boldsymbol{\mathcal{X}} &\cong 1 \times_3^1 \mathcal{G}^{(1)} \times_3^1 \mathcal{G}^{(2)} \times_3^1 \mathcal{G}^{(3)} \times_3^1 \cdots \times_3^1 \mathcal{G}^{(N)} \times_3^1 1 \\ &= [\![1, \mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(N)}, 1]\!] \end{aligned} \tag{A.4}$$

The second case is a little more complicated. We would like to add a dimension at mode-n. The new order has to be a 3rd-order tensor $\mathcal{G}^{(n)} \in \mathbb{R}^{R_{n-1}\times 1\times R_n}$ and the dimension R has to match with its neighbors. We could obtain R_{n-1} easily from the previous core $\mathcal{G}^{(n-1)}$ and we know that $\mathcal{G}^{(n+1)} \in \mathbb{R}^{R_n\times I_{n+1}\times R_{n+1}}$. Consider the original tensor $\mathcal{X}$, then R_n has to be equal to R_{n-1} in order to preserve the information. In other words, $\mathcal{G}^{(n)}$ is a square matrix.

$$\boldsymbol{\mathcal{X}} \cong \mathcal{G}^{(1)} \times_2^1 \mathcal{G}^{(2)} \times_3^1 \cdots \times_3^1 \mathcal{G}^{(n-1)} \times_3^1 \mathcal{G}^{(n)} \times_3^1 \mathcal{G}^{(n+1)} \cdots \times_3^1 \mathcal{G}^{(\mathrm{N})} \tag{A.5}$$

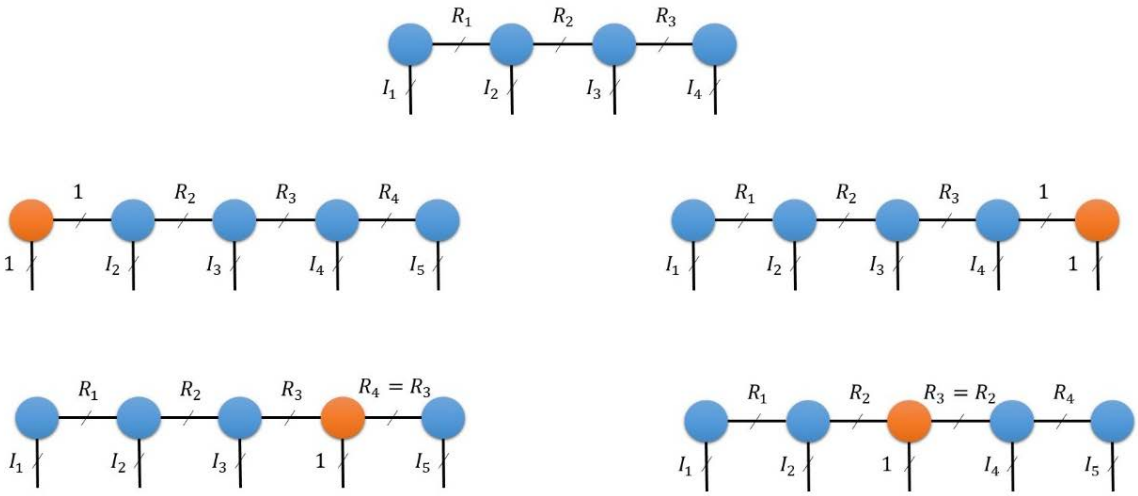

Fig. A.1. Tensor network diagrams of artificially expanded 4th-order to 5th-order TT/MPS
(top) original (middle) expanded on the first and last order
(bottom) expanded in between the train

An identity matrix is used so that the mode product would not change the neighbor tensor core elements hence $\mathcal{G}^{(n)} = \mathbf{I}^{(n)} \in \mathbb{R}^{R_{n-1}\times 1\times R_{n-1}}$. We get

$$\boldsymbol{\mathcal{X}} \cong \mathcal{G}^{(1)} \times_2^1 \mathcal{G}^{(2)} \times_3^1 \cdots \times_3^1 \mathcal{G}^{(n-1)} \times_3^1 \mathbf{I}^{(n)} \times_3^1 \mathcal{G}^{(n+1)} \cdots \times_3^1 \mathcal{G}^{(\mathrm{N})}$$

$$= [\![\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, \cdots, \mathcal{G}^{(n-1)}, \mathbf{I}^{(n)}, \mathcal{G}^{(n+1)} \cdots, \mathcal{G}^{(N)}, \mathcal{G}^{(\mathrm{N})}]\!] \quad \text{(A.6)}$$

Fig. A.1 shows tensor network diagram of the original 4$^{\text{th}}$-order tensor in Tensor Train representation, expansion an extra order on the edges of the train, and expansion in between the train.