NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES
«DATA SCIENCE & INFORMATION TECHNOLOGIES»

SPECIALIZATION
«ARTIFICIAL INTELLIGENCE & BIG DATA»

Masters Thesis

Tensor Methods in Time Series Analysis

Dimitrios M. Aronis

ATHENS

February 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΕΠΙΣΤΗΜΗ ΔΕΔΟΜΕΝΩΝ & ΤΕΧΝΟΛΟΓΙΕΣ ΠΛΗΡΟΦΟΡΙΑΣ»

ΕΞΕΙΔΙΚΕΥΣΗ
«ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ & ΜΕΓΑΛΑ ΔΕΔΟΜΕΝΑ»

Διπλωματική Εργασία

Μέθοδοι Τανυστών στην Ανάλυση Χρονοσειρών

Δημήτριος Μ. Αρώνης

ΑΘΗΝΑ

Φεβρουάριος 2021

Masters Thesis

Tensor Methods in Time Series Analysis

Dimitrios M. Aronis
A.M.: DS1180002

SUPERVISOR: Eleftherios Kofidis, Associate Professor, University of Piraeus - Department of Statistics and Insurance Science

CO-SUPERVISOR: Yannis Panagakis, Associate Professor, National and Kapodistrian University of Athens - Department of Informatics and Telecommunications

EXAMINATION COMMITTEE:

Eleftherios Kofidis, Associate Professor, University of Piraeus - Department of Statistics and Insurance Science

Yannis Panagakis, Associate Professor, National and Kapodistrian University of Athens - Department of Informatics and Telecommunications

George Alexandropoulos, Assistant Professor, National and Kapodistrian University of Athens - Department of Informatics and Telecommunications

February 2021

Διπλωματική Εργασία

Μέθοδοι Τανυστών στην Ανάλυση Χρονοσειρών

Δημήτριος Μ. Αρώνης
A.M.: DS1180002

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ελευθέριος Κοφίδης, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Πειραιώς - Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

ΣΥΝΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Ιωάννης Παναγάκης, Αναπληρωτής Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών - Τμήμα Πληροφορικής και Τηλεπικοινωνιών

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Ελευθέριος Κοφίδης, Αναπληρωτής Καθηγητής, Πανεπιστήμιο Πειραιώς - Τμήμα Στατιστικής και Ασφαλιστικής Επιστήμης

Ιωάννης Παναγάκης, Αναπληρωτής Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών - Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Γεώργιος Αλεξανδρόπουλος, Επίκουρος Καθηγητής, Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών - Department of Informatics and Telecommunications

Φεβρουάριος 2021

ABSTRACT

Time Series have been studied for decades, resulting in the creation of various models with applications in many sciences. However, as we traverse the Big Data era, new challenges arise every day and existing models face certain difficulties. In many real world applications, data appear in the form of matrices or tensors resulting in datasets with high dimensionality structures. Taking into consideration not only the temporal but also any spatial information that is present in these higher order structures, new algorithms and methods were developed. In addition, in many cases we have limited access to a sufficient amount of data because of the problem's nature e.g predicting the market price of a technology item with a relatively short lifespan. In such cases the time series are referred to as Short Time Series. Modeling Short Time Series is challenging since traditional time series models generally perform better when provided with relatively large training datasets. Additionally, in the Big Data setting, the dimensionality increases the complexity even more. To address these problems we need to create new sophisticated models that deal with the increased dimensionality in an efficient way and maximally utilize any spatiotemporal correlations.

In recent years, different tensor decomposition methods were revisited in the context of Tensor Time Series. Tucker Decomposition is one of the most commonly used methods. It decomposes a tensor as a product of a core tensor with lower dimensions and a set of factor matrices. The core tensor ultimately is a summary that captures the intrinsic correlations of the data. Additionally, Tensorization, the process of embedding data into higher order tensors, found its way into the Tensor Time Series setting. Hankelization is a Tensorization method that uses data duplication combined with a folding step in order to create higher order tensors. A Hankel Tensor contains different sub-windows of the original data arranged in a symmetric way. This structure reveals local correlations that were not easily accessible in the original form of the data. In addition, Hankelization has been studied and utilized in various works, showing good results. In cases where low-rank pre-exists in the data, Hankelization reveals it in a more clear way. In the aforementioned works it has been shown experimentally, that such data, in their Hankel tensor form, can be represented by low-rank or a smooth manifold in the embedded space. Therefore, Tucker Decomposition can be very effective when combined with Hankelization and its assumed low-rank property. That is because computing a tensor's rank is an NP-hard task and thus, decomposing a tensor in its original space can be computationally exhausting when compared to decomposing its Hankelized form. In other words, Hankelization enables us to obtain relatively low dimensional core tensors that capture the important information of the data.

In this work, firstly we review some preliminary topics like Matrix and Tensor Algebra, Tensor Decompositions and Tensorization methods in order to familiarize the reader with the necessary concepts and operations and provide a general overview which is needed in order to dive into the Tensor Time Series setting. Early approaches vectorized the observations in order to make use of already existing models and methods. However, this reforming resulted in increased time complexity and high memory demands.

Realizing the need for different approaches various works emerged that utilize tensor decompositions and/or tensorization methods. Here, we focus on the Block Hankel Tensor Autoregression algorithm which combines Hankelization with Tucker Decomposition.

Firstly, the original time series is transformed into a higher-order Block Hankel Tensor. Then Tucker decomposition is applied on all modes except for the temporal, in order to obtain the core tensors and the jointly estimated factor matrices. This way we preserve the temporal continuity of the core tensors in order to better capture their intrinsic temporal correlations. In parallel, we use the obtained core tensors to train an autoregressive process with scalar coefficients. Finally, we use the trained model to forecast the next core tensor which is converted back in the original space via inverse Tucker Decomposition and de-Hankelization.

In summary, the main idea behind Block Hankel Tensor Autoregression, is to extract the most important information of the time series through low-rank Tucker decomposition. Since the approximation of a tensor's rank is a computationally exhausting problem, we project the data in a higher-order embedded space and solve the low-rank minimization problem in that space. Finally, we use the lower-rank core tensors to forecast the following core tensor and then we convert the data back to their original framework. My contribution is a first step towards the generalization of the algorithm. The scalar coefficients capture the spatiotemporal correlations in a restricted way in cases where the data are described by the more general matrix-coefficient model. Therefore, in an effort to make a first step towards the generalized model they are replaced with matrices which can capture efficiently the spatiotemporal correlations of the data, even in the more general case.

Finally, we evaluate Block Hankel Tensor Autoregression with scalar coefficients and its proposed generalization. We compare them to other traditional models and Facebook's Prophet, in terms of prediction error vs training data volume, forecasting horizon and time efficiency, measured as the total runtime of the training process.

SUBJECT AREA: Tensor Time Series

KEYWORDS: Tucker Decomposition, Hankelization, Autoregression

ΠΕΡΙΛΗΨΗ

Οι χρονοσειρές μελετώνται εδώ και δεκαετίες, με αποτέλεσμα τη δημιουργία πολλών μοντέλων με εφαρμογές σε πολλές επιστήμες. Καθώς διανύουμε την εποχή τον μεγάλων δεδομένων, νέες προκλήσεις εμφανίζονται καθημερινά και τα μοντέλα που υπάρχουν αντιμετωπίζουν διάφορες δυσκολίες. Σε πολλές εφαρμογές τα δεδομένα έχουν τη δομή πινάκων ή τανυστών με αποτέλεσμα να καλούμαστε να διαχειριστούμε σύνολα δεδομένων υψηλής διαστασιμότητας. Λαμβάνοντας υπ'όψην όχι μόνο την χρονική αλλά και την χωρική εξάρτηση που πιθανόν να είναι διαθέσιμη σε αυτή τη σύνθετη μορφή δεδομένων αναπτύχθηκαν νέοι αλγόριθμοι και μέθοδοι. Επιπλέον, σε πολλές εφαρμογές η συλλογή πολλών χρονικών δεδομένων είναι δύσκολη λόγω της φύσης του προβλήματος, όπως για παράδειγμα η πρόβλεψη της τιμής ενός προϊόντος τεχνολογίας με σχετικά μικρό διάστημα κυκλοφορίας. Τέτοιες περιπτώσεις χαρακτιρίζονται ως σύντομες χρονοσειρές. Η μοντελοποίηση τους αποτελεί ένα δύσκολο εγχείρημα καθώς η απόδοση των παραδοσιακών μοντέλων σχετίζεται στενά με τον όγκο των δεδομένων στα οποία έχουν εκπαιδευτεί. Επιπλέον, στην εποχή των Μεγάλων Δεδομένων η διαστασιμότητα των δεδομένων καθιστά ακόμα πιο περίπλοκη τη διαδικασία επιλογής και εκπαίδευσης ενός αποτελεσματικού μοντέλου. Συνεπώς, προβάλει επιτακτική η ανάγκη για δημιουργία νέων και εκλεπτυσμένων μοντέλων και αλγόριθμων που θα διαχειρίζονται την αυξημένη διαστασιμότητα και θα αξιοποιούν τις διαθέσιμες χωρικές αλληλεξαρτήσεις σε συνδυασμό με τις περιορισμένες χρονικές για τη δημιουργία αποτελεσματικών μοντέλων.

Τα τελευταία χρόνια, πολλές γνωστές μέθοδοι αποσύνθεσης τανυστών επανεξετάστηκαν στο πλαίσιο των Χρονοσειρών Τανυστών. Η Αποσύνθεση Tucker αποτελεί μια από τις πιο δημοφιλείς μεθόδους αποσύνθεσης, κατά την οποία ένας Τανυστής αποσυνθέτεται ως ένα γινόμενο ενός τανυστή-πυρήνα με ένα σύνολο από πίνακες-παράγοντες. Ο τανυστής-πυρήνας καταγράφει τις εγγενείς συσχετίσεις των δεδομένων. Επιπλέον διάφορες μέθοδοι τάνυσης βρήκαν θέση σε πολλές εφαρμογές χρονοσειρών τανυστών. Η μέθοδος Hankel είναι μια μέθοδος τάνυσης η οποία συνδυάζει ένα βήμα επαύξησης των δεδομένων με αντίγραφα τους και ένα βήμα μετατροπής αυτής της επαυξημένης δομής σε έναν τανυστή μεγαλύτερης τάξης. Ένας τανυστής Hankel αποτελείται από συλλογές των αρχικών δεδομένων οργανωμένα σε μια συμμετρική δομή. Αυτή η ιδιαίτερη δομή υποβοηθα στον εντοπισμό εξαρτήσεων που δεν ήταν εύκολα προσβάσιμες στην αρχική μορφή των δεδομένων. Επιπλέον, η μέθοδος αυτή έχει μελετηθεί σε διάφορες εργασίες στην πρόσφατη βιβλιογραφία με καλά αποτελέσματα. Όταν στα δεδομένα προϋπάρχει μια δομή μικρού βαθμού η τάνυση Hankel την φέρνει στην επιφάνεια. Έχει αποδειχθεί πειραματικά ότι για δεδομένα χαμηλού βαθμού ο αντίστοιχος τανυστής Hankel μπορεί να αναπαρασταθεί με σχετικά μικρό βαθμό ή μια ομαλή πολλαπλότητα στον χώρο προβολής. Συνεπώς, η αποσύνθεση Tucker μπορεί να συνδυαστεί με τη μέθοδο τάνυσης Hankel πολύ αποτελεσματικά. Γνωρίζουμε ότι ο υπολογισμός του βαθμού ενός τανυστή αποτελεί ένα πρόβλημα κλάσης NP και συνεπώς η αποσύνθεση του τανυστή στον αρχικό χώρο μπορεί να αποτελέσει ένα υπολογιστικά κοστοβόρο πρόβλημα συγκριτικά με την αποσύνθεση ενός τανυστή Hankel. Με άλλα λόγια η τάνυση Hankel μας επιτρέπει να αποκτήσουμε τανυστές-πυρήνες, με σχετικά μικρές διαστάσεις, οι οποίοι περιλαμβάνουν το πιο σημαντικό κομμάτι της πληροφορίας των δεδομένων.

Η εργασία αυτή ξεκινάει με μια ανασκόπηση επιλεγμένων προκαταρκτικών θεμάτων, όπως η άλγεβρα πινάκων και τανυστών, οι αποσυνθέσεις τανυστών καθώς και οι διαδικασίες τάνυσης. Στόχος αυτής της ανασκόπησης είναι να παρέχουμε στον αναγνώστη

τις απαραίτητες πληροφορίες που χρειάζεται για να αποκτήσει μια σφαιρική εικόνα του πεδίου. Πρώιμες προσεγγίσεις στον χώρο των χρονοσειρών τανυστών μετέτρεπαν τους τανυστές σε διανύσματα ώστε να χρησιμοποιηθούν τα υπάρχοντα μοντέλα. Αυτό οδήγησε σε αύξηση της απαιτούμενης μνήμης και των χρονικών απαιτήσεων.

Αντιλαμβανόμενοι την ανάγκη για διαφορετικές προσεγγίσεις, προέκυψαν διάφορες εργασίες που χρησιμοποιούν τις αποσυνθέσεις τανυστών και/ή τις μεθόδους τάνυσης. Στην παρούσα εργασία θα εξετάσουμε τον αλγόριθμο Block Hankel Tensor Autoregression ο οποίος συνδυάζει την αποσύνθεση Tucker και τη μέθοδο τάνυσης Hankel. Αρχικά, η χρονοσειρά μετατρέπεται σε έναν τανυστή Hankel ανώτερου βαθμού. Στη συνέχεια εφαρμόζουμε την αποσύνθεση Tucker σε όλες τις διαστάσεις εκτός της χρονικής για να εκτιμήσουμε τους τανυστές-πυρήνες και τους πίνακες-παράγοντες. Με αυτό τον τρόπο διατηρείται η χρονική συνέχεια μεταξύ των τανυστών-πυρήνων που έχει ως στόχο την αποτελεσματική αποτύπωση των χρονικών εξαρτήσεων. Παράλληλα, χρησιμοποιούμε τους τανυστές-πυρήνες για την εκπαίδευση ενός μοντέλου αυτοπαλινδρόμισης με ακέραιους συντελεστές. Τέλος, χρησιμοποιούμε το μοντέλο για να προβλέψουμε τον επόμενο τανυστή-πυρήνα ο οποίος στη συνέχεια προβάλεται στον αρχικό χώρο του προβλήματος μέσω των αντίστροφων διαδικασιών αποσύνθεσης Tucker και τάνυσης Hankel.

Συνοψίζοντας, ο αλγόριθμος που θα εξετάσουμε εξάγει την σημαντική εσωτερική πληροφορία των δεδομένων μέσω της αποσύνθεσης Tucker. Επειδή η εκτίμηση του βαθμού ενός τανυστή είναι μια υπολογιστικά κοστοβόρα διαδικασία, προβάλλουμε τα δεδομένα σε έναν χώρο μεγαλύτερου βαθμου μέσω τάνυσης Hankel. Εκμεταλλευόμενοι τις ιδιότητες ενός τανυστή Hankel λύνουμε το πρόβλημα ελαχιστοποίησης σε αυτό τον χώρο. Παράλληλα χρησιμοποιούμε τους τανυστές-πυρήνες για την πρόβλεψη των επόμενων τανυστών-πυρήνων και τέλος, χρησιμοποιούμε τις αντίστροφες διαδικασίες αποσύνθεσης και τάνυσης για να επαναφέρουμε τις προβλέψεις στον αρχικό χώρο. Η συνεισφορά μου είναι ένα πρώτο βήμα για τη γενίκευση του παραπάνω αλγόριθμου. Οι πραγματικοί συντελεστές αποτυπώνουν ελλειπώς τις χωρικές και χρονικές εξαρτήσεις των δεδομένων όταν αυτά περιγράφονται από ένα μοντέλο με συντελεστές πίνακες. Έτσι, σε μια προσπάθεια να γενικευτεί το παραπάνω μοντέλο αντικαθιστούμε τους πραγματικούς συντελεστές με πίνακες ώστε να μπορούμε να αποτυπώσουμε χωροχρονικές εξαρτήσεις δεδομένων που περιγράφονται από το γενικότερο μοντέλο.

Τέλος, αξιολογούμε τον αλγόριθμο και την προτεινόμενη γενίκευση του. Τους συγκρίνουμε με άλλα κλασικά μοντέλα χρονοσειρών και τον Prophet του Facebook ως προς το σφάλμα πρόβλεψης σε συνδυασμό με τον όγκο των δεδομένων που χρησιμοποιούνται για την εκπαίδευση του μοντέλου, τον ορίζοντα πρόβλεψης καθώς και την χρονική αποδοτικότητα η οποία αντιστοιχεί στη χρονική διάρκεια εκπαίδευσης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Χρονοσειρές Τανυστών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Αποσύνθεση Tucker, Τάνυση κατά Hankel, Αυτοπαλινδρόμηση

ACKNOWLEDGEMENTS

Since this thesis marks the end of my journey as a postgraduate student in National and Kapodistrian Universiy of Athens, Department of Informatics and Telecommunications I would like to thank the people that assisted me along the way.

Firstly, I would like to thank and express my sincere gratitude to my advisors, Dr.Kofidis and Dr.Panagakis for their support, guidance, motivation and encouragement throughout the whole process of writing this thesis.

I would also like to thank Dr.Alexandropoulos for being the third member of my committee and for his constructive comments and suggestions for further improvement of this thesis.

Furthermore, I am very grateful to the faculty members of the Department of Informatics and Telecommunications, National and Kapodistrian Universiy of Athens for their instruction and care over the past two years.

I would like to give special thanks my fellow graduate students who along the way became very good friends. These last two years turned out to be full of productivity, fun and knowledge acquisition, which would not have been possible to that great extent without you.

Last but not least, I want to express my deepest love and gratitude to my family and friends for their love, support and most importantly their understanding, throughout all of my endeavors over the past years.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Motivation

Time series have been in the center of research for many decades. With new challenges arising every day in the Big Data setting, it still remains a very popular research field. Their applications can be found in various scientific fields. Supply chain optimization is a typical example (e.g Forecasting supply and demand for inventory management and vehicle/-transportation scheduling, topology planning etc). Other scientific fields with a plethora of applications include Signal Processing (Audio, Speech, Radar, Biomedical etc), Operational Research, Finance and Economics, Meteorology, Genomics, Psychometrics, Chemometrics etc. Even though, forecasting is the first thing that comes to mind when time series are mentioned, this is not the only task for which they can be utilized. Examples of other tasks include Signal Detection and Estimation, Clustering, Classification and Anomaly Detection.

Traditional time series have been studied for many decades, resulting in the development of models and methods that address various problems. However, as we traverse the Big Data era these models face various difficulties. The volume and dimensionality of the data are some of the characteristics that in many cases make existing models and methods ineffective and/or time-consuming.

In various cases, the observations of a time series are not presented as scalars or vectors but in the form of matrices or tensors. Tensors are in essence arrays of order greater than 2 and therefore constitute the generalization of matrices. For example, in atmospheric temperature forecasting, at a given time point, the data are presented as a third-order tensor whose dimensions correspond to latitude, longitude and elevation, giving rise to a tensor-valued time series.

To address various tasks in which data are presented in this high-order structure, early approaches reshaped the data as vectors in order to use existing models and methods. However, this approach did not produce great results. This reforming resulted in high dimensional vectors which lead to increased time complexity, high memory demands and loss of any intrinsic structure information that was present in the original tensor. For example, adjacent cells of a matrix or a tensor may express spatial correlation (e.g adjacent cells of a tensor that contains atmospheric temperature data) and thus this information could be utilized for the benefit of a task.

Another approach was to maintain the tensor structure of the data and design new algorithms that use this form to overcome the aforementioned problems and exploit any additional information that is hidden in this structure. This idea presents us with a new field of research, that of the Tensor Time Series Analysis. It is essential that one should get familiar with Tensor Algebra and its notations as a first step towards understanding the proposed methods. Tensor Algebra has many similarities with Matrix Algebra as well as notable differences.

This thesis explores tensor methods in time series analysis. However, this area contains a plethora of tasks and approaches which cannot be covered here in detail. In recent years various works have risen regarding different tasks like forecasting, tensor completion and clustering. Most of these works utilize a tensor decomposition model and its respective benefits. Tensor Decompositions are in essence embeddings of high dimensional data in a lower dimensional framework while maintaining the important intrinsic information that

is present and removing problem-specific noise. Additionally, different methods can be divided in two major groups with respect to the form of the data they process. In the first group the data are presented naturally as tensors while in the second group they are tensorized i.e they are embedded in a higher-order vector space. Different methods that use Hankelization are studied and utilized to an increasing extent. Hankelization is a tensorization technique that transforms lower-order data arrays into a higher-order Hankel tensor via data duplication and a reshaping step. A Hankel tensor is symmetric and contains different sub-windows of the original data. This enables the exploration of different areas of data in order to discover local correlations that were not easily accessible originally. Furthermore, it has been shown experimentally that for data with a hidden underlying low-rank structure their produced Hankel tensor can be represented by low-rank or a smooth manifold in the embedded space. In the time series setting, the correlations between the data are strong. Therefore, this hidden low-rank structure is usually present in the data and thus, Hankelization will reveal this low-rank structure in a more clear way.

In that context, we choose to focus on the task of multivariate short time series forecasting via the Block Hankel Tensor Autoregression algorithm. In many real-world applications it is not possible to have access to a lot of observations of a time series due to case specific constraints. This may result in ineffective models since in the traditional time series setting the accuracy of the model is closely correlated with the amount of the provided training data. The main idea of the discussed method is the transformation of the data into a higher order tensor via Hankelization. As mentioned previously, such a Hankel tensor can be represented by low-rank or a smooth manifold in the embedded space and thus its structure is ideal to be combined with low-rank tensor decomposition. For this task we utilize Tucker decomposition. Tucker decomposition is applied in an iterative process that estimates the core tensors and a joint factor matrix for every mode but the temporal. By doing so, we preserve the temporal continuity of the core tensors in order to better capture their intrinsic temporal correlations. In parallel an autoregressive process is trained on the obtained core tensors. After obtaining the predicted core tensors we utilize the inverse process of Tucker Decomposition and Hankelization to transform the prediction back into the original framework.

Finally, my contribution is a first step towards the generalization of the algorithm. If a given time series is described by a general autoregressive process with matrix coefficients then an autoregressive process with scalar coefficients captures the spatiotemporal correlations of the core tensors in a restricted way. Thus, by substituting the scalar coefficients with matrices we can also capture the spatiotemporal correlations of the more general case. The experimental results seem promising, providing lower forecasting error on 1 synthetic and 7 publicly available datasets.

1.2 Thesis Outline

The remaining chapters of this work are organized as follows:

Chapter 2 covers various preliminary concepts, that the reader must be familiar with, such as notations, matrix and tensor algebra, tensor decompositions and tensorization procedures.

Chapter 3 provides an in-depth view of the tensorization method that will be applied in our setting. It contains proofs and derivations leading to the equations that will be used as the structural blocks of the algorithm and the proposed generalization.

Chapter 4 outlines the experiments that were conducted in order to evaluate the algorithm. We focus on short time series and experiment on short and long forecasting horizons.

Chapter 5 contains a brief review of two additional works in order to provide a wider overview of the field for the reader. Since the research field of tensor time series analysis is vast we chose two works that are conceptually close to the rest of this thesis and thus they can be easily grasped by the reader. Furthermore, they certainly pose a lot of interest. In the first work the proposed method decomposes a tensor valued time series via CP Decomposition and uses a 2-dimensional ARMA model for long-term prediction. Furthermore, they proposed another method that used tensor clustering and treated short-term forecasting as tensor completion. Finally, the second work combines Hankelization and Tensor Train decomposition for the task of time series reconstruction.

Finally, in Chapter 6 we conclude with a summary and review various ideas for the extension of this work in the future.

2. PRELIMINARIES

In this chapter we are providing the notations, definitions, operations and basic properties of matrices and tensors. Having a good understanding of this chapter's material before studying the rest of this thesis is vital. In the first section, basic notations and definitions are introduced. The second section is dedicated to Matrix Algebra which includes several topics such as the matrix rank, trace, various products and their properties. The third section contains an overview of some basic Tensor Algebra operations and Tensor decompositions. Finally, the last section focuses on tensorization, the mapping or transformation of lower-order data to higher-order representations.

2.1 Notations

Definition 2.1

In computer science an *array* is a structure or a collection of data that can be accessed with a set of pointers.

Definition 2.2

Order of an array is the number of its dimensions. The dimensions can also be referred to as *ways* or *modes*. [1]

In essence, *vectors* are one dimensional arrays, *matrices* are second order arrays and *tensors* are arrays with three or more modes. Throughout this thesis, the following notation will be used to refer to vectors, matrices and tensors. Vectors will be denoted as bold low-case letters, Matrices will be referred to with bold capital letters and Tensors will be denoted with calligraphic capital letters. The letter N will be used to refer to the order of a tensor and I_n will denote the dimension of the n-th mode where $n = 1, 2, ..., N$. To refer to specific elements of vectors, matrices and tensors, italic low-case letters will be used followed by the specified indices. The following table provides a visual representation of the different notations.

Table 1: Notations

	Array Notation	**Element Notation**
Vector	$\boldsymbol{x} \in \mathbb{R}^{I_1}$	x_i
Matrix	$\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2}$	x_{ij}
Tensor	$\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$	$x_{i_1 i_2 ... i_N}$

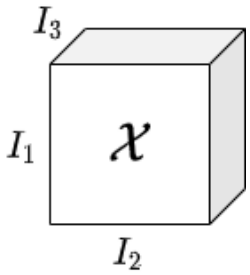

Figure 1: Visualization of a third-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

Definition 2.3

The i_n-th mode-n slice of a 3rd order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is defined as the second order tensor (matrix) obtained by fixing the n-th mode index of $\mathcal{X}$ to i_n.

The previous definition can be generalized to a N-th order tensor as follows.

Definition 2.4

The i_n-th mode-n slab of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is defined as an $(N-1)$-th order tensor obtained by fixing the n-th mode index of $\mathcal{X}$ to i_n.

We will use the symbol $\bullet$ to denote the non-fixed modes. For example, given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, a first mode slice is denoted as $\mathcal{X}_{i_1 \bullet\bullet}$. Given a time series in the form of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ we assume, without loss of generality, that I_N is the temporal mode. For simplicity, the time series at a given time point t will be denoted as $\mathcal{X}_t$ instead of $\mathcal{X}_{\bullet \ldots \bullet t}$.

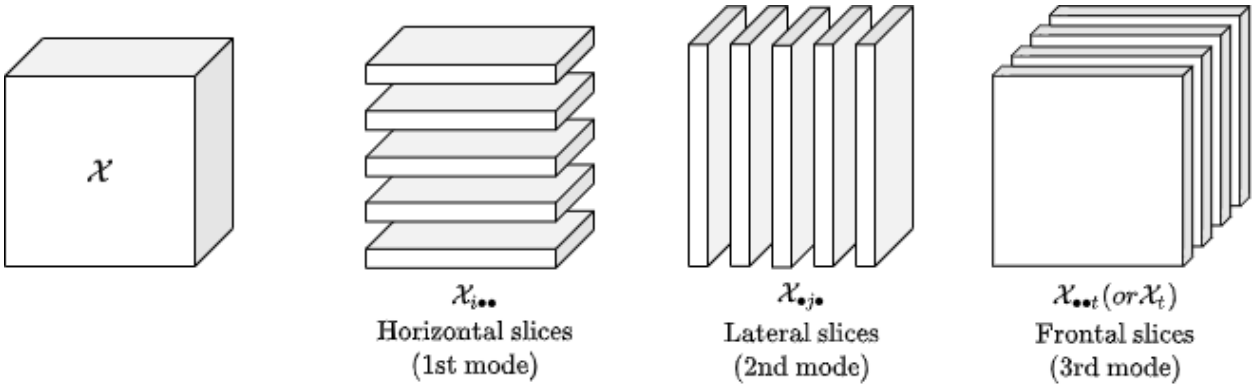

Figure 2: Visualization of different slices of a third order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

2.2 Vectors

Inner Product

The *Inner Product* of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$ is a scalar defined as:

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{N} x_i y_i \tag{2.1}$$

Outer Product

The *Outer Product* of two vectors $\boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{y} \in \mathbb{R}^M$ is a matrix $\boldsymbol{Z} \in \mathbb{R}^{N \times M}$ that contains the product of each pair of elements of those vectors. It is denoted as:

$$\boldsymbol{Z} := \boldsymbol{x} \circ \boldsymbol{y} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \dots & x_1 y_M \\ x_2 y_1 & x_2 y_2 & \dots & x_2 y_M \\ \vdots & \vdots & \dots & \\ x_N y_1 & x_N y_2 & \dots & x_N y_M \end{bmatrix} \tag{2.2}$$

2.3 Matrices

2.3.1 Trace

The trace of a square matrix $\boldsymbol{X} \in \mathbb{R}^{n \times n}$ is defined as the sum of elements on the main diagonal of $\boldsymbol{X}$. It is denoted and formulated as $tr(\boldsymbol{X}) = \sum_{i=1}^{n} x_{ii}$.

Useful Properties

- $tr(\boldsymbol{X} + \boldsymbol{Y}) = tr(\boldsymbol{X}) + tr(\boldsymbol{Y})$ for $\boldsymbol{X}$,$\boldsymbol{Y} \in \mathbb{R}^{n \times n}$
- $tr(c\boldsymbol{X}) = c\, tr(\boldsymbol{X})$
- $tr(\boldsymbol{X}) = tr(\boldsymbol{X}^\top)$
- $tr(\boldsymbol{X}^\top \boldsymbol{Y}) = tr(\boldsymbol{X}\boldsymbol{Y}^\top) = tr(\boldsymbol{Y}\boldsymbol{X}^\top) = tr(\boldsymbol{Y}^\top \boldsymbol{X})$ for $\boldsymbol{X}$,$\boldsymbol{Y} \in \mathbb{R}^{n \times m}$
- $tr(\boldsymbol{X}\boldsymbol{Y}) \neq tr(\boldsymbol{X})tr(\boldsymbol{Y})$ in general.
- $tr(\boldsymbol{X}\boldsymbol{Y}\boldsymbol{Z}) = tr(\boldsymbol{Y}\boldsymbol{Z}\boldsymbol{X}) = tr(\boldsymbol{Z}\boldsymbol{X}\boldsymbol{Y})$ (Cyclic Property)
- $tr(\boldsymbol{X} \otimes \boldsymbol{Y}) = tr(\boldsymbol{X})tr(\boldsymbol{Y})$

2.3.2 Matrix Rank

The *row rank* of a matrix is defined as the dimension of the vector space that is spanned by its rows (respectively *column rank* is the dimension of the vector space spanned by its columns), which is equivalent to the maximum number of linearly independent rows (respectively columns). In linear algebra, a fundamental result is that the row and column ranks of a matrix are always equal and thus we will refer to them simply as rank. The rank of a matrix $\boldsymbol{A}$ is denoted as rank($\boldsymbol{A}$). A matrix is full-rank if its rank equals the minimum of its numbers of rows and columns. Finally, an important corollary of the above is that for a given matrix $\boldsymbol{A}$, rank($\boldsymbol{A}$) = rank($\boldsymbol{A}^\top$).

For example, the matrix $\boldsymbol{A} = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 1 \\ 1 & -1 & 0 \end{bmatrix}$ has rank($\boldsymbol{A}$) $= 2$ since the 1st and 3rd columns are linearly independent and the second column is a linear combination $(c_2 = c_3 - c_1)$ of them.

2.3.3 Matrix Products and their Properties

Hadamard Product

Given two matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathbb{R}^{I \times J}$ their *Hadamard* product is defined [2] as their element-wise multiplication and is denoted as:

$$\boldsymbol{X} * \boldsymbol{Y} := \begin{bmatrix} x_{11}y_{11} & x_{12}y_{12} & \dots & x_{1J}y_{1J} \\ x_{21}y_{21} & x_{22}y_{22} & \dots & x_{2J}y_{2J} \\ \vdots & \vdots & \dots & \vdots \\ x_{I1}y_{I1} & x_{I2}y_{I2} & \dots & x_{IJ}y_{IJ} \end{bmatrix}$$

Kronecker Product

Given two matrices $\boldsymbol{X} \in \mathbb{R}^{I \times J}$ and $\boldsymbol{Y} \in \mathbb{R}^{K \times L}$, their *Kronecker* product is a $IK \times JL$ matrix, defined [3] as:

$$\boldsymbol{X} \otimes \boldsymbol{Y} := \begin{bmatrix} x_{11}\boldsymbol{Y} & x_{12}\boldsymbol{Y} & \dots & x_{1J}\boldsymbol{Y} \\ x_{21}\boldsymbol{Y} & x_{22}\boldsymbol{Y} & \dots & x_{2J}\boldsymbol{Y} \\ \vdots & \vdots & \dots & \vdots \\ x_{I1}\boldsymbol{Y} & x_{I2}\boldsymbol{Y} & \dots & x_{IJ}\boldsymbol{Y} \end{bmatrix}$$

For example, given the matrices $\boldsymbol{X} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} e & f & g \\ h & i & j \end{bmatrix}$ their Kronecker product is:

$$\boldsymbol{X} \otimes \boldsymbol{Y} = \begin{bmatrix} a\begin{bmatrix} e & f & g \\ h & i & j \end{bmatrix} & b\begin{bmatrix} e & f & g \\ h & i & j \end{bmatrix} \\ c\begin{bmatrix} e & f & g \\ h & i & j \end{bmatrix} & d\begin{bmatrix} e & f & g \\ h & i & j \end{bmatrix} \end{bmatrix} = \begin{bmatrix} ae & af & ag & be & bf & bg \\ ah & ai & aj & bh & bi & bj \\ ce & cf & cg & de & df & dg \\ ch & ci & cj & dh & di & dj \end{bmatrix}$$

Khatri-Rao Product

Given two matrices $\boldsymbol{X} \in \mathbb{R}^{I \times K}$ and $\boldsymbol{Y} \in \mathbb{R}^{J \times K}$, the *Khatri-Rao* product is a $IJ \times K$ matrix, defined [3] as:

$$\boldsymbol{X} \odot \boldsymbol{Y} := \begin{bmatrix} \boldsymbol{X}_{\bullet 1} \otimes \boldsymbol{Y}_{\bullet 1} & \boldsymbol{X}_{\bullet 2} \otimes \boldsymbol{Y}_{\bullet 2} & \dots & \boldsymbol{X}_{\bullet K} \otimes \boldsymbol{Y}_{\bullet K} \end{bmatrix}$$

For example, given the matrices $\boldsymbol{X} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ and $\boldsymbol{Y} = \begin{bmatrix} e & f \\ g & h \\ i & j \end{bmatrix}$ their Khatri-Rao product is:

$$\boldsymbol{X} \odot \boldsymbol{Y} = \begin{bmatrix} \begin{bmatrix} a \\ c \end{bmatrix} \otimes \begin{bmatrix} e \\ g \\ i \end{bmatrix} & \begin{bmatrix} b \\ d \end{bmatrix} \otimes \begin{bmatrix} f \\ h \\ j \end{bmatrix} \end{bmatrix} = \begin{bmatrix} a\begin{bmatrix} e \\ g \\ i \end{bmatrix} & b\begin{bmatrix} f \\ h \\ j \end{bmatrix} \\ c\begin{bmatrix} e \\ g \\ i \end{bmatrix} & d\begin{bmatrix} f \\ h \\ j \end{bmatrix} \end{bmatrix} = \begin{bmatrix} ae & bf \\ ag & bh \\ ai & bj \\ ce & df \\ cg & dh \\ ci & dj \end{bmatrix}$$

Useful Properties

- $\boldsymbol{X} \otimes (\boldsymbol{Y} + \boldsymbol{Z}) = \boldsymbol{X} \otimes \boldsymbol{Y} + \boldsymbol{X} \otimes \boldsymbol{Z}$
- $(\boldsymbol{Y} + \boldsymbol{Z}) \otimes \boldsymbol{X} = \boldsymbol{Y} \otimes \boldsymbol{X} + \boldsymbol{Z} \otimes \boldsymbol{X}$
- $(k\boldsymbol{X}) \otimes \boldsymbol{Y} = \boldsymbol{X} \otimes (k\boldsymbol{Y}) = k(\boldsymbol{X} \otimes \boldsymbol{Y})$
- $(\boldsymbol{X} \otimes \boldsymbol{Y}) \otimes \boldsymbol{Z} = \boldsymbol{X} \otimes (\boldsymbol{Y} \otimes \boldsymbol{Z})$
- $\boldsymbol{X} \otimes \boldsymbol{0} = \boldsymbol{0} \otimes \boldsymbol{X} = \boldsymbol{0}$
- In general, $\boldsymbol{X} \otimes \boldsymbol{Y} \neq \boldsymbol{Y} \otimes \boldsymbol{X}$ but they are permutation equivalent.
- $(\boldsymbol{X} \otimes \boldsymbol{Y})(\boldsymbol{Z} \otimes \boldsymbol{P}) = \boldsymbol{XZ} \otimes \boldsymbol{YP}$
- $(\boldsymbol{X} \otimes \boldsymbol{Y})^{-1} = \boldsymbol{X}^{-1} \otimes \boldsymbol{Y}^{-1}$ for invertible $\boldsymbol{X}, \boldsymbol{Y}$ (likewise for pseudo-inverse).
- $(\boldsymbol{X} \otimes \boldsymbol{Y})^\top = \boldsymbol{X}^\top \otimes \boldsymbol{Y}^\top$
- For $\boldsymbol{X} \in \mathbb{R}^{n \times n}, \boldsymbol{Y} \in \mathbb{R}^{m \times m}$: $det(\boldsymbol{X} \otimes \boldsymbol{Y}) = det(\boldsymbol{X})^m det(\boldsymbol{Y})^n$
- $tr(\boldsymbol{X} \otimes \boldsymbol{Y}) = tr(\boldsymbol{X}) tr(\boldsymbol{Y})$, for square matrices $\boldsymbol{X}, \boldsymbol{Y}$
- $rank(\boldsymbol{X} \otimes \boldsymbol{Y}) = rank(\boldsymbol{X}) rank(\boldsymbol{Y})$
- $(\boldsymbol{X} \odot \boldsymbol{Y}) \odot \boldsymbol{Z} = \boldsymbol{X} \odot (\boldsymbol{Y} \odot \boldsymbol{Z})$
- In general, $\boldsymbol{X} \odot \boldsymbol{Y} \neq \boldsymbol{Y} \odot \boldsymbol{X}$
- $(\boldsymbol{X} \odot \boldsymbol{Y})^\top (\boldsymbol{X} \odot \boldsymbol{Y}) = \boldsymbol{X}^\top \boldsymbol{X} * \boldsymbol{Y}^\top \boldsymbol{Y}$
- $(\boldsymbol{X} \otimes \boldsymbol{Y})(\boldsymbol{Z} \odot \boldsymbol{P}) = \boldsymbol{XZ} \odot \boldsymbol{YP}$
- $tr(\boldsymbol{XYZ}) = tr(\boldsymbol{ZXY}) = tr(\boldsymbol{YZX})$
- $vec(\boldsymbol{X})^\top vec(\boldsymbol{Y}) = tr(\boldsymbol{X}^\top \boldsymbol{Y})$
- $vec(\boldsymbol{XYZ}) = (\boldsymbol{Z}^\top \otimes \boldsymbol{X}) vec(\boldsymbol{Y})$

2.4 Tensors

2.4.1 Tensor Rank

A tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is called rank-one if it can be expressed as the outer product of N vectors. Each vector corresponds to a different mode.

$$\mathcal{X} = \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \ldots \circ \boldsymbol{x}^{(N)}$$

Given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, its rank is denoted as $rank(\mathcal{X})$ and defined [1] as the minimum number of rank-one tensors that sum up to the given tensor $\mathcal{X}$.

$$rank(\mathcal{X}) := argmin\{R \in \mathbb{N} : \mathcal{X} = \sum_{i=1}^{R} \boldsymbol{x}_i^{(1)} \circ \boldsymbol{x}_i^{(2)} \circ \ldots \circ \boldsymbol{x}_i^{(N)}\}$$

Determining the rank of a tensor is an NP-complete problem over any finite field and NP-hard over rational numbers. As a result, various works studied the upper bounds of a tensor's rank. For example, for a general tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ its upper bound is given by $max\{I_1, I_2, ..., I_N\}^{N-1}$ while a 3rd order general tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is upper-bounded by $min\{I_1 I_2, I_2 I_3, I_1 I_3\}$.

2.4.2 Tensor Vectorization

Vectorization [1] is the process of converting a higher order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ into a vector $\boldsymbol{x} \in \mathbb{R}^{I_1 I_2 ... I_N}$. The vector $\boldsymbol{x}$ is obtained by mapping each element $x_{i_1 i_2 ... i_N}$ of the tensor to the vector's element with index:

$$index = 1 + \sum_{n=1}^{N} (i_n - 1) \prod_{m=1}^{n-1} I_m$$

For example, given a tensor $\mathcal{X}$:

$$\mathcal{X}_{\bullet\bullet 1} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}, \mathcal{X}_{\bullet\bullet 2} = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$$

The vectorization operator produces the following result.

$$vec(\mathcal{X}) = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}^\top$$

2.4.3 Tensor Matricization/Unfolding

Matricization or *Unfolding* is the process of converting a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ with $N > 2$ into a matrix. The general operators that are used to denote this mapping are $mat(\mathcal{X})$ or more commonly $unfold(\mathcal{X})$. This mapping can be achieved in N different ways, each one producing a mode-n unfolding, denoted as $\boldsymbol{X}^{(n)}$. A mode-n unfolding is defined as a function

$$f : \mathbb{R}^{I_1 \times I_2 \times ... \times I_N} \rightarrow \mathbb{R}^{I_n \times \prod_{k \neq n} I_k}$$

that maps (as defined by Kolda & Bader [4]) each element $x_{i_1 ... i_n ... i_N}$ of the tensor to the index (i_n, j) of the unfolded matrix, where:

$$j = 1 + \sum_{k \neq n} (i_k - 1) \prod_{m \neq n}^{m<k} I_m$$

For example, given a tensor $\mathcal{X} \in \mathbb{R}^{3 \times 2 \times 3}$ with frontal slices:

$$\mathcal{X}_{\bullet\bullet 1} = \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}, \mathcal{X}_{\bullet\bullet 2} = \begin{bmatrix} g & j \\ h & k \\ i & l \end{bmatrix}, \mathcal{X}_{\bullet\bullet 3} = \begin{bmatrix} m & p \\ n & q \\ o & r \end{bmatrix}$$

The mode-1 (row) unfolding produces the matrix:

$$\boldsymbol{X}^{(1)} = \begin{bmatrix} a & d & g & j & m & p \\ b & e & h & k & n & q \\ c & f & i & l & o & r \end{bmatrix}$$

The mode-2 (column) unfolding produces the matrix:

$$\boldsymbol{X}^{(2)} = \begin{bmatrix} a & b & c & g & h & i & m & n & o \\ d & e & f & j & k & l & p & q & r \end{bmatrix}$$

The mode-3 (depth) unfolding produces the matrix:

$$\boldsymbol{X}^{(3)} = \begin{bmatrix} a & b & c & d & e & f \\ g & h & i & j & k & l \\ m & n & o & p & q & r \end{bmatrix}$$

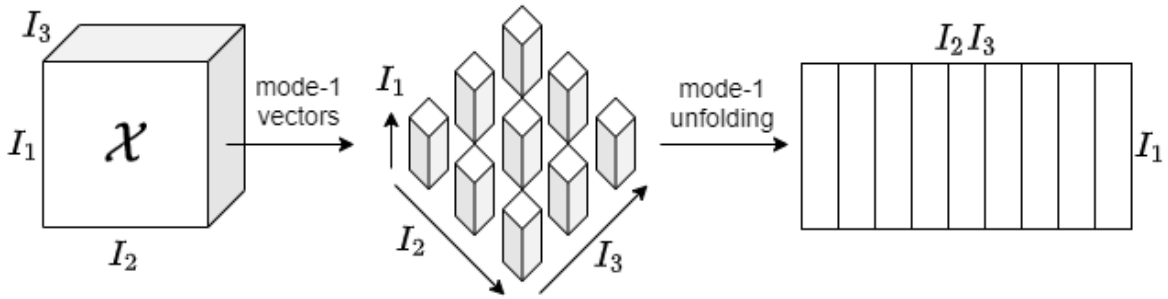

Figure 3: Visualization of mode-1 unfolding of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

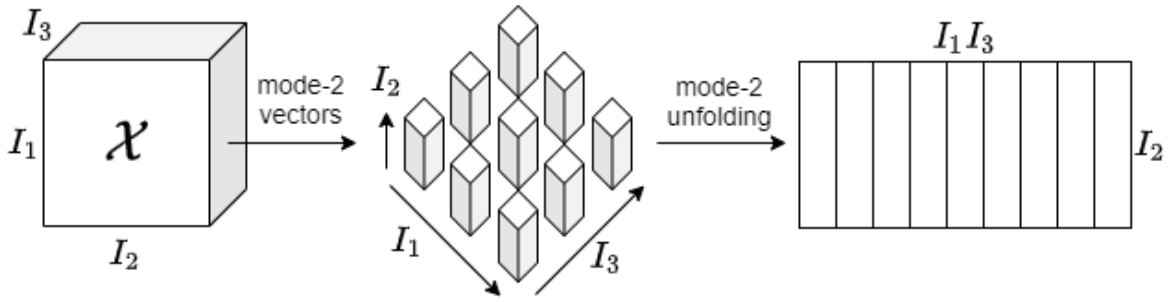

Figure 4: Visualization of mode-2 unfolding of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

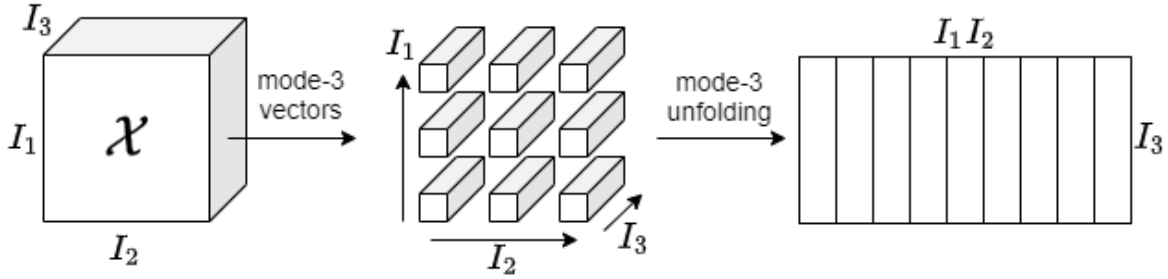

Figure 5: Visualization of mode-3 unfolding of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

2.4.4 Tensor Multiplication

In this section we explore various tensor multiplication methods.

Inner Product

The inner product $z \in \mathbb{R}$ of two tensors of the same order and size, $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is defined as the inner product of their vectorizations and it is denoted as:

$$z = \langle \mathcal{X}, \mathcal{Y} \rangle := \langle vec(\mathcal{X}), vec(\mathcal{Y}) \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N} y_{i_1 i_2 \ldots i_N}$$

Outer Product

The outer product of two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ and $\mathcal{Y} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_K}$ is defined as a function f that produces a new tensor $\mathcal{Z} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times J_1 \times J_2 \times \ldots \times J_K}$. It is denoted as $\mathcal{Z} = \mathcal{X} \circ \mathcal{Y}$ and each element of $\mathcal{Z}$ is calculated as: $z_{i_1 \ldots i_N j_1 \ldots j_K} = x_{i_1 \ldots i_N} y_{j_1 \ldots j_K}$

Tensor-Matrix Product

The Tensor-Matrix product is a process in which, given a matrix $\boldsymbol{Y} \in \mathbb{R}^{K \times I_n}$ and a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_n \times \ldots \times I_N}$, they are multiplied producing the tensor $\mathcal{Z} \in \mathbb{R}^{I_1 \times \ldots \times K \times \ldots \times I_N}$. The process requires the *mode-n unfolding* of $\mathcal{X}$, which produces the matrix:

$$\boldsymbol{X}^{(n)} \in \mathbb{R}^{I_n \times \prod_{q=1, q \neq n}^{N} I_q}$$

Then, $\boldsymbol{Y}$ and $\boldsymbol{X}^{(n)}$ are multiplied, producing the matrix:

$$\boldsymbol{Z}^{(n)} \in \mathbb{R}^{K \times \prod_{q=1, q \neq n}^{N} I_q}$$

Finally, $\boldsymbol{Z}^{(n)}$ is converted into a tensor by folding (the reverse process of unfolding mentioned in 2.4.3).

$$\mathcal{Z} = fold(\boldsymbol{Z}^{(n)}) \in \mathbb{R}^{I_1 \times \ldots \times K \times \ldots \times I_N}$$

The process described above is called the mode-n product and it is denoted as:

$$\mathcal{Z} = \mathcal{X} \times_n \boldsymbol{Y} := fold(\boldsymbol{Y} \boldsymbol{X}^{(n)}) \tag{2.3}$$

Each element of the tensor $\mathcal{Z}$ can be computed as:

$$z_{i_1 \ldots i_{n-1}\, k\, i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_n \ldots i_N} y_{k\, i_n} \tag{2.4}$$

Tensor Contraction

Given two tensors $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_1 \times \ldots \times J_K}$ and $\mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots \times I_N \times J_{K+1} \times \ldots \times J_M}$, their contraction over N modes of the same size is a tensor $\mathcal{Z} \in \mathbb{R}^{J_1 \times \ldots \times J_M}$. The contraction is denoted as $\mathcal{X} \circledast \mathcal{Y}$ and its order is given by $order(\mathcal{Z}) = order(\mathcal{X}) + order(\mathcal{Y}) - 2N$. The elements of tensor $\mathcal{Z}$ are calculated as

$$z_{j_1 \ldots j_K j_{K+1} \ldots j_M} = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \ldots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \ldots i_N j_1 j_2 \ldots j_K} y_{i_1 \ldots i_N j_{K+1} \ldots j_M} \tag{2.5}$$

2.4.5 Frobenius Norm

The *Frobenius* norm of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$, is equal to the squared root of the sum of the squared elements of the tensor and it is analogous to the Frobenius Norm of a matrix.

$$\|\mathcal{X}\|_F := \sqrt{<\mathcal{X}, \mathcal{X}>} = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} ... \sum_{i_N=1}^{I_N} x_{i_1 i_2 ... i_N}^2}$$

2.4.6 Tensor Decompositions

In this section we briefly discuss Tensor Decomposition (or Factorization), a process that decomposes a tensor in lower-rank tensors. Two majorly used decompositions will be presented, the Canonical Polyadic Decomposition and the Tucker Decomposition.

2.4.6.1 Canonical Polyadic Decomposition

The *Canonical Polyadic Decomposition* is a model in which, a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ is decomposed as a weighted sum of R rank-one tensors:

$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \boldsymbol{x}_r^{(1)} \circ \boldsymbol{x}_r^{(2)} \circ ... \circ \boldsymbol{x}_r^{(N)} \tag{2.6}$$

where R is the rank of the tensor, N is the number of modes, $\circ$ is the outer product, λ_r is the scalar weight of the r-th factor and $\boldsymbol{x}_r^{(n)} \in \mathbb{R}^{I_n}$ for $n = 1, 2, ..., N$ are unit norm vectors. The CP Decomposition is essentially a generalization of *Singular Value Decomposition* to tensors. A visualization of the Canonical Polyadic Decomposition is presented below in Figure 6.

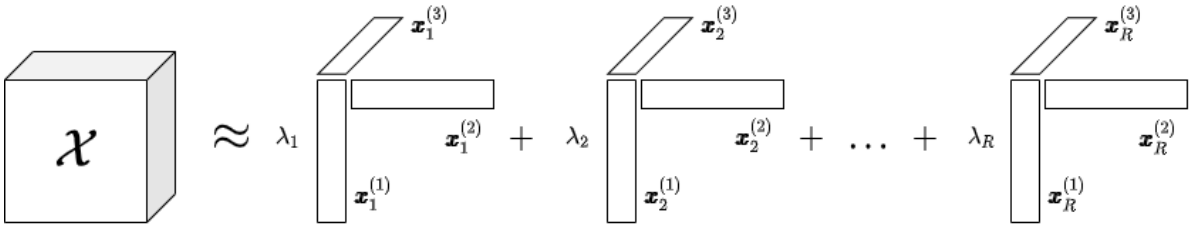

Figure 6: Canonical Polyadic decomposition of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

2.4.6.2 Tucker Decomposition

Tucker Decomposition [1] [4] is a model in which a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ is decomposed as a series of mode-n products between a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times ... \times R_N}$ and a set of factor matrices $\boldsymbol{U}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. The vector $\begin{bmatrix} R_1 & R_2 & ... & R_N \end{bmatrix}$ is called the multi-linear rank or the *Tucker Rank* of $\mathcal{X}$. The above definition is summarized asg:

$$\mathcal{X} \approx \mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \times_3 ... \times_N \boldsymbol{U}^{(N)} \tag{2.7}$$

Each element of $\mathcal{X}$ is given by:

$$x_{i_1 i_2 ... i_N} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} ... \sum_{r_N=1}^{R_N} g_{i_1 i_2 ... i_N} \boldsymbol{u}^{(1)}_{i_1 r_1} \boldsymbol{u}^{(2)}_{i_2 r_2} ... \boldsymbol{u}^{(N)}_{i_N r_N} \tag{2.8}$$

where $\boldsymbol{u}^{(k)}_{ij}$ is the element located at the position (i, j) in the factor matrix $\boldsymbol{U}^{(k)}$.

Tucker Decomposition is visualized in Figure 7 below.

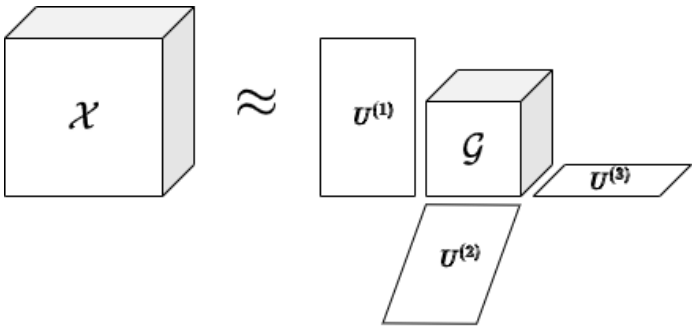

Figure 7: Visualization of Tucker Decomposition of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$

2.5 Tensorization

Tensorization is the process of mapping data presented as arrays of lower-order into a higher-order structure. For example, a vector can be transformed into a matrix or tensor, a matrix into a tensor and a tensor into a tensor of higher order.

2.5.1 Hankelization

While there are various tensorization procedures, *Hankelization* (named after Herman Hankel) has been widely used in a variety of applications. Applying second order Hankelization on a vector results in a *Hankel* matrix i.e. all elements on each secondary diagonal (or skew diagonal) are the same. When applied on a matrix or a tensor the result is a higher-order tensor that contains slabs which are Hankel matrices. The Hankelization operator is denoted as $\mathcal{H}(\cdot)$. *De-Hankelization*, the inverse process that transforms a Hankelized array into its original form, is denoted as $\mathcal{H}^{-1}(\cdot)$. In general, applying K-th order Hankelization on the n-th mode of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times ... \times I_n \times ... \times I_N}$ results in tensor $\mathcal{V} \in \mathbb{R}^{I_1 \times ... \times I_{n-1} \times J_1 \times ... \times J_K \times I_{n+1} \times ... \times I_N}$ where $I_n = \sum_{k=1}^{K} J_k - K + 1$. The element at position $(i_1, ..., i_{n-1}, j_1, ..., j_K, i_{n+1}, ..., i_N)$ corresponds to the element of $\mathcal{X}$ at the position $(i_1, ..., l, ..., i_N)$ where $l = \sum_{k=1}^{K} j_k - K + 1$. Below we present some examples.

Vector to Matrix

When second-order ($K = 2$) Hankelization is applied on a vector $\boldsymbol{v} \in \mathbb{R}^N$, it is transformed into a matrix $\boldsymbol{V} \in \mathbb{R}^{I \times J}$ where $N = I + J - K + 1 = I + J - 1$.

For example, given a vector $\boldsymbol{v} \in \mathbb{R}^7$:

$$\boldsymbol{v} = [1, 2, 3, 4, 5, 6, 7]^\top$$

applying second-order Hankelization with $I = J = 4$ produces the following matrix:

$$\mathcal{H}(\boldsymbol{v}) = \boldsymbol{V} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

while applying second-order Hankelization with $I = 3$ and $J = 5$ would produce the following matrix:

$$\mathcal{H}(\boldsymbol{v}) = \boldsymbol{V} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 6 & 7 \end{bmatrix}$$

Vector to Tensor

When K-th order Hankelization is applied on a vector $\boldsymbol{v} \in \mathbb{R}^N$, it is transformed into a tensor $\mathcal{V} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_K}$ where $N = \sum_{k=1}^{K} I_k - K + 1$. For example, applying third-order Hankelization on $\boldsymbol{v} = [1, 2, 3, 4, 5, 6, 7]^\top$ with $I_1 = I_2 = I_3 = 3$ results in the tensor:

$$\mathcal{V}_{\bullet\bullet 1} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix} \mathcal{V}_{\bullet\bullet 2} = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{bmatrix} \mathcal{V}_{\bullet\bullet 3} = \begin{bmatrix} 3 & 4 & 5 \\ 4 & 5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

Matrix to Tensor

When applying Hankelization on a matrix, we need to specify the mode (or modes) on which it will be applied. A K-th order Hankelization applied on the first mode would transform a matrix $\boldsymbol{V} \in \mathbb{R}^{I_{start} \times J}$ into a tensor $\mathcal{V} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_K \times J}$. We remind that when applying K-th order Hankelization on a mode the rule $I_{start} = \sum_{k=1}^{K} I_k - K + 1$ must be satisfied. Note that when applying Hankelization on both modes, the order K can be different for each mode.

For example given the matrix $\boldsymbol{V} \in \mathbb{R}^{4 \times 3}$

$$\boldsymbol{V} = \begin{bmatrix} 1 & 5 & 9 \\ 2 & 6 & 10 \\ 3 & 7 & 11 \\ 4 & 8 & 12 \end{bmatrix}$$

applying second order ($K = 2$) Hankelization on the first mode we have:

$$I = \sum_{k=1}^{K} I_k - K + 1 \implies$$

$$4 = \sum_{k=1}^{2} I_k - 2 + 1 \implies$$
$$I_1 + I_2 = 5$$

For $I_1 = 3$ and $I_2 = 2$, Hankelization results in the following tensor $\mathcal{V} \in \mathbb{R}^{3\times2\times3}$

$$\mathcal{V}_{\bullet\bullet1} = \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{bmatrix} \mathcal{V}_{\bullet\bullet2} = \begin{bmatrix} 5 & 6 \\ 6 & 7 \\ 7 & 8 \end{bmatrix} \mathcal{V}_{\bullet\bullet3} = \begin{bmatrix} 9 & 10 \\ 10 & 11 \\ 11 & 12 \end{bmatrix}$$

2.5.2 Delay Embedding Transform

In this section we review the Delay Embedding Transform as it was defined by Yokota et al [5] [6]. The Standard or Multi-way Delay Embedding Transform are specific cases of the aforementioned Hankelization methods but it is important to lay out their definitions since they will be utilized in the following algorithms.

2.5.2.1 Standard Delay Embedding Transform

The *Standard Delay Embedding Transform* is essentially a second-order Hankelization on a single mode. As mentioned in 2.5.1, when applying K-th order Hankelization, the values of I_k for $k = 1, 2, ..., K$ must satisfy $\sum_{k=1}^{K} I_k = I_{start} + K - 1$. Thus, for a second order Hankelization the equation becomes $I_1 + I_2 = I_{start} + 1$. The Standard Delay Embedding Transform sets $I_1 = r$, where r is a user-defined value and as a result I_2 is calculated by the formula $I_2 = I_{start} - r + 1$. The Standard Delay Embedding Transform will be denoted as $\mathcal{H}_r(\cdot)$ and the reverse process will be denoted as $\mathcal{H}_r^{-1}(\cdot)$. Applying Standard Delay Embedding Transform on a vector $\boldsymbol{v} \in \mathbb{R}^N$ results in $\mathcal{H}_r(\boldsymbol{v}) = \boldsymbol{V}_H \in \mathbb{R}^{r\times(N-r+1)}$. The following equivalence holds:

$$vec(\mathcal{H}_r(\boldsymbol{v})) = \boldsymbol{S}\boldsymbol{v} \iff \mathcal{H}_r(\boldsymbol{v}) = fold_{(N,r)}(\boldsymbol{S}\boldsymbol{v})$$

where $fold_{(N,r)} : \mathbb{R}^{r(N-r+1)} \to \mathbb{R}^{r\times(N-r+1)}$ is a function that transforms a vector into a matrix. One can equivalently describe this transformation as multiplying vector $\boldsymbol{v}$ with the duplication matrix $\boldsymbol{S} \in \mathbb{R}^{r(N-r+1)\times N}$ followed by the folding operation. The duplication matrix $\boldsymbol{S}$ consists of r identity matrices of size $r \times r$ stacked diagonally as shown In the following example. Given a vector $\boldsymbol{v} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}^\top$, we have $N = 5$. Setting $r = 3$ results in the following duplication matrix $\boldsymbol{S} \in \mathbb{R}^{9\times5}$:

$$\boldsymbol{S} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix}$$

The product of the duplication matrix and the vector is:

$$\boldsymbol{Su} = \begin{bmatrix} \mathbf{1} & 0 & 0 & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{1} & 0 \\ 0 & 0 & 0 & 0 & \mathbf{1} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 2 \\ 3 \\ 4 \\ 3 \\ 4 \\ 5 \end{bmatrix}$$

Finally, by applying the folding operation we obtain the Hankel matrix:

$$\mathcal{H}_r(\boldsymbol{u}) = \boldsymbol{V}_H = fold_{(5,3)}(\boldsymbol{Su}) = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

De-Hankelization is essentially a series of inverse operations that result in the original vector. Since the last step of Hankelization was the folding operation, the first step of De-Hankelization is the vectorization of the matrix, which results in $\boldsymbol{Su}$. Finally, we left-multiply with the Moore-Penrose pseudo-inverse of $\boldsymbol{S}$. This process is summarized as:

$$\mathcal{H}_r^{-1}(\boldsymbol{V}_H) = \boldsymbol{S}^{\dagger} vec(\boldsymbol{V}_H)$$

The Standard Delay Embedding Transform can be also applied on matrices and tensors. Applying Standard Delay Embedding Transform on the first mode of a matrix $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2}$ results in the tensor $\mathcal{H}_r(\boldsymbol{X}) = \mathcal{X}_H \in \mathbb{R}^{r \times (I_1 - r + 1) \times I_2}$. In general, applying Standard Delay Embedding Transform on the n-th mode of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ produces the tensor $\mathcal{H}_r(\mathcal{X}) = \mathcal{X}_H \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_{n-1} \times r \times (I_n - r + 1) \times I_{n+1} \times \ldots \times I_N}$
The operation that produces the tensors in these two cases is denoted as:

$$\mathcal{H}_r(\mathcal{X}) = \mathcal{X}_H = fold_{(I_n,r)}(\mathcal{X} \times_n \boldsymbol{S})$$

while the inverse process can be written as:

$$\mathcal{H}_r^{-1}(\mathcal{X}_H) = unfold_{(I_n,r)}(\mathcal{X}_H) \times_n \boldsymbol{S}^{\dagger}$$

where $unfold_{(I_n,r)}(\cdot)$ is the inverse process of $fold_{(I_n,r)}(\cdot)$. The following Figure visualizes the procedure of the Standard Delay Embedding Transform. Firstly, matrix $\boldsymbol{X}$ is multiplied with the duplication matrix $\boldsymbol{S}$. Then, the folding step is applied on their product. A third mode thus appears, increasing the order from 2 to 3.

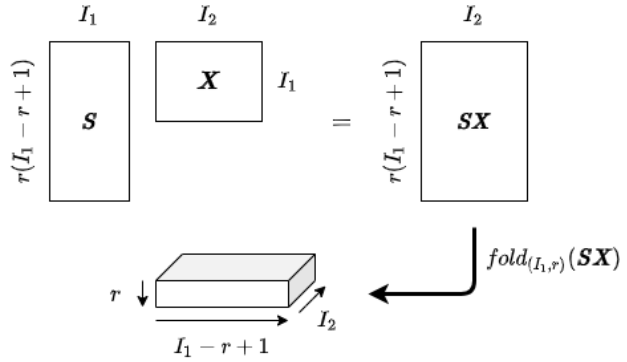

Figure 8: Standard Delay Embedding Transform on the 1st mode of a matrix $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2}$

2.5.2.2 Multi-way Delay Embedding Transform

The *Multi-way Delay Embedding Transform* (MDT) constitutes the generalization of the Standard Delay Embedding Transform. Applying MDT on every mode of a N-th order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ requires N Hankelization parameters $r = \begin{pmatrix} r_1, & r_2, & ..., & r_N \end{pmatrix}$ and N duplication matrices $S_1, S_2, ..., S_N$. Applying MDT on specific modes is achieved by setting their corresponding r_n to the desired value and $r_n = 1$ for the rest. By setting $r_n = 1$, each duplication matrix $\boldsymbol{S}_d \in \mathbb{R}^{r_n(I_n - r_n + 1) \times I_n}$ becomes $\boldsymbol{S}_n \in \mathbb{R}^{I_n \times I_n}$ i.e an identity matrix of size I_n. The procedure of MDT can be summarized as:

$$\mathcal{H}_r(\mathcal{X}) = \mathcal{X}_H = fold_{(I,r)}(\mathcal{X} \times_1 \boldsymbol{S}_1 \times_2 ... \times_N \boldsymbol{S}_N) \tag{2.9}$$

where $I = I_1 \times I_2 \times ... \times I_N$ and $fold$ is defined as a function:

$$fold_{(I,r)}(\cdot) : \mathbb{R}^{r_1(I_1 - r_1 + 1) \times ... \times r_N(I_N - r_N + 1)} \rightarrow \mathbb{R}^{r_1 \times (I_1 - r_1 + 1) \times ... \times r_N \times (I_N - r_N + 1)}$$

The inverse procedure is defined as:

$$\mathcal{H}_r^{-1}(\mathcal{X}_H) = unfold_{(I,r)}(\mathcal{X}_H) \times_1 \boldsymbol{S}_1^{\dagger} \times_2 ... \times_N \boldsymbol{S}_N^{\dagger} \tag{2.10}$$

where $unfold$ is the inverse process of the previously defined function $fold$. The following Figure visualizes the procedure of the Multi-way Delay Embedding Transform on a matrix. The matrix $\boldsymbol{X}$ is multiplied with the duplication matrices $\boldsymbol{S}_1$ and $\boldsymbol{S}_2$ followed by a folding step. A third and fourth mode thus appear, increasing the order from 2 to 4.

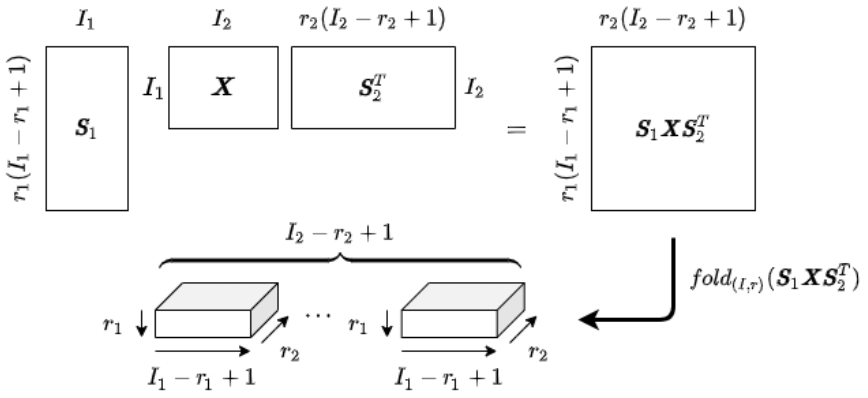

Figure 9: Multi-way Delay Embedding Transform on both modes of a matrix $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2}$

3. BLOCK HANKEL TENSOR AUTOREGRESSION

In this chapter, we focus on multivariate short time series forecasting. As mentioned in Chapter 1 there is a need to overcome various disadvantages of early approaches (e.g vectorization of data in the form of tensors). In addition, given the short aspect of the time series, more problems arise. It is expected that traditional models will not perform so well due to the dimensionality of the data and the fact that their accuracy is closely correlated with the amount of data provided during the training process. On the other hand, neural networks and their great expressive capabilities could be utilized in the tensor time series setting. However, they usually require an even larger amount of data in the training process which is not available in the short time series setting. Finally, applications of tensorization and tensor decompositions in various studies and their promising results paved the road for further research on such applications.

In this context we focus on the Block Hankel Tensor Autoregression algorithm. It based on the Multilinear Orthogonal Autoregressive model proposed by Jing et al [7] and the work of Shi et al [8], the Block Hankel Tensor ARIMA process. In their work, Jing et al. starting with a tensor-valued time series, exploited the advantages of Tucker Decomposition in an iterative process that estimated the semi-orthogonal projection matrices $\widehat{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. Using the estimated factor matrices they obtained the core tensors $\widehat{\mathcal{G}}_t$. Finally, instead of using the original tensors $\mathcal{X}_t$, they used the obtained core tensors to estimate the coefficients of an Autoregressive process. Shi et al. expanded this method utilizing the power of Hankelization and transforming lower-order data into a higher order Block Hankel Tensor (i.e. a block tensor whose entries are Hankel tensors). Each observation of the Block Hankel tensor is essentially a sub-window on the original data that contains various timestamps. This structure could help capturing the temporal correlations of the data in a more effective way. Furthermore, exploiting the Hankel tensor's low-rank property, they applied low-rank Tucker decomposition on the Hankelized tensor and estimated the joint projection matrices $\widehat{\boldsymbol{U}}^{(n)}$ and the core tensors $\widehat{\mathcal{G}}_t$. The obtained core tensors were used to estimate the coefficients of an ARIMA process. Finally, after forecasting on the core tensor level the result is transformed back into the original framework. In the following paragraphs we review each of the algorithm's components in detail and provide the necessary proofs.

3.1 Multi-way Delay Embedding Transform

In the first part of the algorithm we utilize MDT to transform the data into a higher-order tensor. For a given tensor $\mathcal{X}_{start} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_M \times T_{start}}$, MDT is applied on the temporal mode, resulting in the tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times T}$ where $N = M + 1$, $J_1 = I_1$, ..., $J_M = I_{N-1}$, $I_N = r$ and $T = T_{start} - r + 1$. In the following sections we denote this Hankelization as $\mathcal{H}_r(\cdot)$. This is achieved by a mode-n product between the tensor and the duplication matrix $\boldsymbol{S} \in \mathbb{R}^{r(T-r+1) \times T}$. The main benefit of applying MDT on time series data (where the correlations are strong) is that the produced Block Hankel Tensor can be represented by low-rank or a smooth manifold in the embedded space [5] [6]. Therefore, this property allows us to exploit low-rank Tucker decomposition effectively in order to extract the intrinsic local correlations of the data while "compressing" the tensor. Finally, each temporal slab of a Hankel tensor, is a sub-window that contains consecutive observations of the original time series. This mapping could also boost the ability of the forecasting model to capture

spatiotemporal correlations, especially in the short time series context.

3.2 Block Hankel Tensor Autoregression with Scalar Coefficients

Starting with the tensorized time series $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N \times T}$, where the last mode corresponds to the temporal direction, applying Tucker Decomposition (2.7) on the first N modes results in:

$$\mathcal{X} \approx \widehat{\mathcal{G}} \times_1 \widehat{\boldsymbol{U}}^{(1)} \times_2 \widehat{\boldsymbol{U}}^{(2)} \times_3 ... \times_N \widehat{\boldsymbol{U}}^{(N)}$$

Then each temporal slab of the core tensor can be estimated as:

$$\widehat{\mathcal{G}}_t \approx \mathcal{X}_t \times_1 \widehat{\boldsymbol{U}}^{(1)^\dagger} \times_2 \widehat{\boldsymbol{U}}^{(2)^\dagger} \times_3 ... \times_N \widehat{\boldsymbol{U}}^{(N)^\dagger} \tag{3.1}$$

where $\widehat{\mathcal{G}}_t$ are the estimated core tensor's temporal slabs and $\widehat{\boldsymbol{U}}^{(n)^\dagger}$ denotes the Moore-Penrose inverse of the estimated factor matrix $\widehat{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ with $I_n > R_n$ for $n = 1, 2, ..., N$. Additionally, the constraint of semi-orthogonality is applied on the projection matrices. Thus, the following property must hold:

$$\widehat{\boldsymbol{U}}^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)} = \boldsymbol{I}_{R_n} \tag{3.2}$$

Combining (3.2) and (3.1) we get:

$$\widehat{\mathcal{G}}_t \approx \mathcal{X}_t \times_1 \widehat{\boldsymbol{U}}^{(1)^\top} \times_2 \widehat{\boldsymbol{U}}^{(2)^\top} \times_3 ... \times_N \widehat{\boldsymbol{U}}^{(N)^\top} \tag{3.3}$$

A core tensor contains a compressed representation of the intrinsic interactions of multiple factors that are present in the original tensor. Furthermore, the factor matrices are estimated jointly on all modes but the temporal in order to maximally preserve the temporal continuity. This way the temporal correlations between the core tensors are maximally preserved while we extract only the important information that is present in the rest of the modes and remove case specific noise. Furthermore, the volume of the data is decreased as we obtain the compressed core tensors. Thus, training a model on the temporal slabs of the core tensors could be more promising in the scope of capturing the latent temporal correlations and intrinsic interactions among multiple time series when compared to a model trained on the temporal slabs of the original tensor.

We remind that in the first step of the algorithm we utilized MDT to transform the original data into a Block Hankel Tensor. Therefore, the autoregressive process that will be utilized from this point forward is the following:

$$\mathcal{X}_t = \sum_{i=1}^{p} a_i \mathcal{X}_{t-i} + \tilde{\mathcal{E}}_t \tag{3.4}$$

where $\alpha_i \in \mathbb{R}$ are the scalar coefficients, $\mathcal{X}_t, \mathcal{X}_{t-i} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ are two observations of the time series with lag i and $\tilde{\mathcal{E}}_t \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N}$ corresponds to white noise. For simplicity we assume that $E[\mathcal{X}_t] = 0$. If $E[\mathcal{X}_t] \neq 0$ then $\mathcal{X}_t$ is replaced by $\mathcal{Y}_t = \mathcal{X}_t - E[\mathcal{X}_t]$.

Left-multiplying (3.4) with $\prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top}$ we get:

$$\mathcal{X}_t \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top} = \left(\sum_{i=1}^{p} a_i \mathcal{X}_{t-i} \right) \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top} + \tilde{\mathcal{E}}_t \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top} \implies$$

$$\mathcal{X}_t \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top} = \left(\sum_{i=1}^{p} a_i \mathcal{X}_{t-i} \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top} \right) + \tilde{\mathcal{E}}_t \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top} \implies$$

$$\widehat{\mathcal{G}}_t = \sum_{i=1}^{p} a_i \widehat{\mathcal{G}}_{t-i} + \mathcal{E}_t \tag{3.5}$$

Our goal is to minimize the norm of the prediction error $\|\mathcal{E}_t\|_F$ which corresponds to:

$$\|\mathcal{E}_t\|_F = \|\widehat{\mathcal{G}}_t - \sum_{i=1}^{p} a_i \widehat{\mathcal{G}}_{t-i}\|_F \tag{3.6}$$

Furthermore, in order to minimize the noise produced by the inverse decomposition procedure we add the following term that corresponds to the inverse decomposition error.

$$\|\widehat{\mathcal{G}}_t - \mathcal{X}_t \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top} \|_F \tag{3.7}$$

$$\text{subject to } \widehat{\boldsymbol{U}}^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)} = \boldsymbol{I}_{R_n} \text{ for } n = 1, 2, ..., N.$$

Finally, by combining both terms (3.6 & 3.7) the optimization problem is written as:

$$\underset{\{\widehat{\mathcal{G}}_t, \widehat{\boldsymbol{U}}^{(n)}, a_i\}}{\text{argmin}} \sum_{t=p+1}^{T} \left(\frac{1}{2} \|\widehat{\mathcal{G}}_t - \sum_{i=1}^{p} a_i \widehat{\mathcal{G}}_{t-i}\|_F^2 + \frac{1}{2} \|\widehat{\mathcal{G}}_t - \mathcal{X}_t \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top} \|_F^2 \right)$$

$$\text{subject to } \widehat{\boldsymbol{U}}^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)} = \boldsymbol{I}_{R_n} \text{ for } n = 1, 2, ..., N.$$

To facilitate the differentiation of the above optimization problem, it is written as a sum of N terms, each one corresponding to a mode-n unfolding.

$$\underset{\{\widehat{\boldsymbol{G}}_t^{(n)}, \widehat{\boldsymbol{U}}^{(n)}, a_i\}}{\text{argmin}} \sum_{t=p+1}^{T} \sum_{n=1}^{N} \left(\frac{1}{2} \|(\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})\|_F^2 + \frac{1}{2} \|\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \|_F^2 \right) \tag{3.8}$$

$$\text{subject to } \widehat{\boldsymbol{U}}^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)} = \boldsymbol{I}_{R_n} \text{ for } n = 1, 2, ..., N$$

where $\widehat{\boldsymbol{U}}^{(-n)} = \widehat{\boldsymbol{U}}^{(N)^\top} \otimes ... \otimes \widehat{\boldsymbol{U}}^{(n+1)^\top} \otimes \widehat{\boldsymbol{U}}^{(n-1)^\top} \otimes ... \otimes \widehat{\boldsymbol{U}}^{(1)^\top} \in \mathbb{R}^{\prod_{j \neq n} R_j \times \prod_{j \neq n} I_j}$

3.2.1 Updating the core tensors

To compute the partial derivative of (3.8) both of the above norms are expressed as traces. The first norm is written equivalently as:

$$\|\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}\|_F^2 =$$

$$tr \left((\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})^\top (\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}) \right) =$$

$$tr\left((\widehat{\boldsymbol{G}}_t^{(n)^\top} - (\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})^\top)(\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})\right) =$$

$$tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)} - (\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})^\top \widehat{\boldsymbol{G}}_t^{(n)} + (\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})^\top (\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})\right) =$$

$$tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)} - (\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})^\top + (\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})^\top (\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})\right) \overset{(*)}{=}$$

$$tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\right) - 2\,tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}\right) + tr\,(\boldsymbol{B})$$

where $\boldsymbol{B} = (\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})^\top \sum_{j=1}^{p} a_j \widehat{\boldsymbol{G}}_{t-j}^{(n)}$

Note that in the last step we used the following property of traces: $tr(\boldsymbol{A}) = tr(\boldsymbol{A}^\top)$

The partial derivatives of the above traces are:

- $\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\right) = 2\widehat{\boldsymbol{G}}_t^{(n)}$
- $\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}\right) = \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}$
- $\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\,(\boldsymbol{B}) = \frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left((\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)})^\top \sum_{j=1}^{p} a_j \widehat{\boldsymbol{G}}_{t-j}^{(n)}\right) = 0$

The partial derivative of the first term with respect to $\widehat{\boldsymbol{G}}_t^{(n)}$ is:

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}}\left(\frac{1}{2}\|\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}\|_F^2\right) =$$

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}}\left(\frac{1}{2}\left(tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\right) - 2\,tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}\right) + tr\,(\boldsymbol{B})\right)\right) =$$

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}}\left(\frac{1}{2}tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\right) - tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}\right) + \frac{1}{2}tr\,(\boldsymbol{B})\right) =$$

$$\frac{1}{2}\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\right) - \frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}\right) + \frac{1}{2}\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\,(\boldsymbol{B}) =$$

$$\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}$$

Summarizing the above we have:

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}}\left(\frac{1}{2}\|\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)}\|_F^2\right) = \widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)} \tag{3.9}$$

The second norm is written equivalently as:

$$\|\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \|_F^2 =$$

$$tr\left((\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top})^\top (\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top}) \right) =$$

$$tr\left((\widehat{\boldsymbol{G}}_t^{(n)^\top} - \widehat{\boldsymbol{U}}^{(-n)} \boldsymbol{X}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)}) (\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top}) \right) =$$

$$tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} - \widehat{\boldsymbol{U}}^{(-n)} \boldsymbol{X}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)} \widehat{\boldsymbol{G}}_t^{(n)} + \boldsymbol{C} \right) =$$

$$tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{G}}_t^{(n)} \right) - tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \right) - tr\left(\widehat{\boldsymbol{U}}^{(-n)} \boldsymbol{X}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)} \widehat{\boldsymbol{G}}_t^{(n)} \right) + tr\left(\boldsymbol{C} \right) =$$

$$tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{G}}_t^{(n)} \right) - tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \right) - tr\left((\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top})^\top \right) + tr\left(\boldsymbol{C} \right) =$$

$$tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{G}}_t^{(n)} \right) - 2\, tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \right) + tr\left(\boldsymbol{C} \right)$$

where $\boldsymbol{C} = \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \widehat{\boldsymbol{U}}^{(-n)} \boldsymbol{X}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)}$

The partial derivatives of the above traces are:

- $\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{G}}_t^{(n)} \right) = 2\widehat{\boldsymbol{G}}_t^{(n)}$
- $\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \right) = \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top}$
- $\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\boldsymbol{C} \right) = \frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \widehat{\boldsymbol{U}}^{(-n)} \boldsymbol{X}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)} \right) = 0$

The partial derivative of the second term with respect to $\widehat{\boldsymbol{G}}_t^{(n)}$ is

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} \left(\frac{1}{2} \|\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \|_F^2 \right) =$$

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} \left(\frac{1}{2} \left(tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{G}}_t^{(n)} \right) - 2\, tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \right) + tr\left(\boldsymbol{C} \right) \right) \right) =$$

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} \left(\frac{1}{2} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{G}}_t^{(n)} \right) - tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \right) + \frac{1}{2} tr\left(\boldsymbol{C} \right) \right) =$$

$$\frac{1}{2} \frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{G}}_t^{(n)} \right) - \frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top} \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \right) + \frac{1}{2} \frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} tr\left(\boldsymbol{C} \right) =$$

$$\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top}$$

Summarizing the above we have:

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}} \left(\frac{1}{2} \|\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \|_F^2 \right) = \widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \tag{3.10}$$

Using (3.9) & (3.10) we compute the partial derivative of (3.8) with respect to $\widehat{\boldsymbol{G}}_t^{(n)}$. By setting it to zero we get:

$$\sum_{t=p+1}^{T}\left(\widehat{\boldsymbol{G}}_t^{(n)}-\sum_{i=1}^{p}a_i\widehat{\boldsymbol{G}}_{t-i}^{(n)}+\widehat{\boldsymbol{G}}_t^{(n)}-\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\right)=0$$

$$\sum_{t=p+1}^{T}\left(2\widehat{\boldsymbol{G}}_t^{(n)}-\sum_{i=1}^{p}a_i\widehat{\boldsymbol{G}}_{t-i}^{(n)}-\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\right)=0$$

So, each $\widehat{\boldsymbol{G}}_t^{(n)}$, $n=1,2,...,N$ is updated by:

$$\widehat{\boldsymbol{G}}_t^{(n)}=\frac{1}{2}\left(\sum_{i=1}^{p}a_i\widehat{\boldsymbol{G}}_{t-i}^{(n)}+\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\right) \tag{3.11}$$

After the unfolded matrices are updated, they are converted back to their tensor form by

$$\widehat{\mathcal{G}}_t=fold(\widehat{\boldsymbol{G}}_t^{(n)})$$

3.2.2 Updating the factor matrices

(3.8) with respect to $\widehat{\boldsymbol{U}}^{(n)}$ is:

$$\underset{\{\widehat{\boldsymbol{U}}^{(n)}\}}{\text{argmin}}\sum_{t=p+1}^{T}\sum_{n=1}^{N}\left(\frac{1}{2}\|\widehat{\boldsymbol{G}}_t^{(n)}-\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\|_F^2\right)$$

$$\text{subject to }\ \widehat{\boldsymbol{U}}^{(n)^\top}\widehat{\boldsymbol{U}}^{(n)}=\boldsymbol{I}_{R_n}\text{ for }n=1,2,...,N$$

Shang, Lie and Cheng in 2014 [9] have shown that the minimization of the quantity above with respect to the factor matrices $\widehat{\boldsymbol{U}}^{(n)}$ implies:

$$\widehat{\boldsymbol{U}}^{(n)}=\underset{\{\widehat{\boldsymbol{U}}^{(n)}\}}{\text{argmax}}\sum_{t=p+1}^{T}tr\left(\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\widehat{\boldsymbol{G}}_t^{(n)^\top}\right) \tag{3.12}$$

(3.12) corresponds to the orthogonality Procrustes problem. The global optimal solution for this problem is given by the SVD of:

$$\sum_{t=p+1}^{T}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\widehat{\boldsymbol{G}}_t^{(n)^\top}\in\mathbb{R}^{I_n\times R_n} \tag{3.13}$$

We conduct Singular Value Decomposition on (3.13) and keep only the non-zero singular values. We obtain the matrices $\boldsymbol{L}\in\mathbb{R}^{I_n\times R_n}$ and $\boldsymbol{R}\in\mathbb{R}^{R_n\times R_n}$ whose columns are the left and right singular vectors respectively. The following holds $\boldsymbol{L}^\top\boldsymbol{L}=\boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I}_{R_n}$. However, since $\boldsymbol{R}$ is a square matrix with orthonormal columns it is orthogonal. Therefore, we have $\boldsymbol{R}\boldsymbol{R}^\top=\boldsymbol{I}_{R_n}$.

The factor matrices $\widehat{\boldsymbol{U}}^{(n)}\in\mathbb{R}^{I_n\times R_n}$ are updated by:

$$\widehat{\boldsymbol{U}}^{(n)}=\boldsymbol{L}\boldsymbol{R}^\top \tag{3.14}$$

We can easily deduct that the estimation of $\widehat{\boldsymbol{U}}^{(n)}$, obtained by (3.13), is a semi-orthogonal matrix with orthonormal columns since:

$$\widehat{\boldsymbol{U}}^{(n)^\top}\widehat{\boldsymbol{U}}^{(n)}=(\boldsymbol{L}\boldsymbol{R}^\top)^\top(\boldsymbol{L}\boldsymbol{R}^\top)=\boldsymbol{R}\boldsymbol{L}^\top\boldsymbol{L}\boldsymbol{R}^\top=\boldsymbol{R}\boldsymbol{I}_{R_n}\boldsymbol{R}^\top=\boldsymbol{R}\boldsymbol{R}^\top=\boldsymbol{I}_{R_n}$$

3.2.3 Estimating the scalar coefficients

The coefficients of the AR model are estimated on the core tensors, generalizing a Least Squares modified *Yule-Walker* method that supports data in the form of tensors. We calculate auto-correlation as

$$r_p = \frac{\sum_{t=1}^{T-p} \langle vec(\mathcal{X}_t), vec(\mathcal{X}_{t+l}) \rangle}{\sum_{t=1}^{T} \langle vec(\mathcal{X}_t), vec(\mathcal{X}_t) \rangle}$$

We estimate the coefficients by solving the following linear system, known as the Yule-Walker equations.

$$\begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{p-1} \\ r_p \end{bmatrix} = \begin{bmatrix} r_0 & r_1 & r_2 & \dots & r_{p-2} & r_{p-1} \\ r_1 & r_0 & r_1 & \dots & r_{p-3} & r_{p-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{p-2} & r_{p-3} & r_{p-4} & \dots & r_0 & r_1 \\ r_{p-1} & r_{p-2} & r_{p-3} & \dots & r_1 & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{p-1} \\ a_p \end{bmatrix} \iff \boldsymbol{r} = \boldsymbol{R}\boldsymbol{A}$$

where $\boldsymbol{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{p-1} \\ r_p \end{bmatrix}$, $\boldsymbol{R} = \begin{bmatrix} r_0 & r_1 & r_2 & \dots & r_{p-2} & r_{p-1} \\ r_1 & r_0 & r_1 & \dots & r_{p-3} & r_{p-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ r_{p-2} & r_{p-3} & r_{p-4} & \dots & r_0 & r_1 \\ r_{p-1} & r_{p-2} & r_{p-3} & \dots & r_1 & r_0 \end{bmatrix}$ and $\boldsymbol{A} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{p-1} \\ a_p \end{bmatrix}$

The above system is well-posed since $\boldsymbol{R}$ is a symmetric, positive semi-definite matrix. Its solution is given by $\widehat{\boldsymbol{A}} = \boldsymbol{R}^{-1}\boldsymbol{r}$.

3.2.4 Forecasting

Finally, after the coefficient estimation process is finished, the next core tensor is forecasted by:

$$\widehat{\mathcal{G}}_{T+1} = \sum_{i=1}^{p} a_i \widehat{\mathcal{G}}_{T-i+1} \in \mathbb{R}^{R_1 \times R_2 \times \dots \times R_N}$$

The forecast is concatenated with the rest of the observations along the temporal mode, resulting in the tensor $\widehat{\mathcal{G}}_{new} \in \mathbb{R}^{R_1 \times R_2 \times \dots \times R_N \times (T+1)}$. After $\widehat{\mathcal{G}}_{new}$ is obtained, the inverse Tucker Decomposition is applied using the estimated factor matrices $\widehat{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, to re-map the data into the original space. Finally, the inverse MDT process is applied on the Hankelized tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_N \times (T+1)}$ to obtain the lower-order tensor $\mathcal{X}_{start} \in \mathbb{R}^{J_1 \times J_2 \times \dots \times J_M \times (T_{start}+1)}$ where $M = N - 1$.

3.2.5 The Algorithm

The algorithm requires as input the time series $\mathcal{X}_{start} \in \mathbb{R}^{J_1 \times \dots \times J_M \times T_{start}}$, the AR order *p*, the maximum number of iterations max_iter, the MDT order r and the tolerance for the stop criterion tol. Finally, to incorporate non-stationary time series we conduct d-th order differencing.

BHT_AR_SC

1. Apply MDT to get $\widetilde{\mathcal{X}} = \mathcal{H}_r(\mathcal{X}_{start}) \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N \times T}$, where $T = T_{start} - r + 1$
2. Conduct d-order differencing on $\widetilde{\mathcal{X}}$ and get $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N \times (T-d)}$
3. Set the Tucker Decomposition ranks, R_n, $n = 1, 2, ..., N$
4. Set $iter = 0$ and initialize $\widehat{\boldsymbol{U}}^{(n)}$, $n = 1, 2, ..., N$ randomly
5. While $iter \leq max\_iter$ and $convergence\_value > tol$
   (a) Estimate $\widehat{\mathcal{G}}_t = \mathcal{X}_t \times_1 \widehat{\boldsymbol{U}}^{(1)^\top} \times_2 \widehat{\boldsymbol{U}}^{(2)^\top} \times_3 ... \times_N \widehat{\boldsymbol{U}}^{(n)^\top}$, for $t = 1, 2, ..., T - d$
   (b) Estimate the scalar coefficients a_i for $i = 1, 2, ..., p$ by solving the Yule-Walker equations on the core tensors $\widehat{\mathcal{G}}_t$, for $t = 1, 2, ..., T - d$.
   (c) For $n = 1, 2, ..., N$
      i. By (3.11) set $\widehat{\boldsymbol{G}}_t^{(n)} = \frac{1}{2}\left(\sum_{i=1}^{p} a_i \widehat{\boldsymbol{G}}_{t-i}^{(n)} + \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top}\right)$ for $t = p+1, ..., T-d$
      ii. Restore tensor form $\widehat{\mathcal{G}}_t = Fold(\widehat{\boldsymbol{G}}_t^{(n)})$
      iii. $old\_\widehat{\boldsymbol{U}}^{(n)} = \widehat{\boldsymbol{U}}^{(n)}$
      iv. By (3.14) set $\widehat{\boldsymbol{U}}^{(n)} = \boldsymbol{L}\boldsymbol{R}^\top$
   (d) $iter = iter + 1$
   (e) $convergence\_value = \frac{\sum_{n=1}^{N} \|\widehat{\boldsymbol{U}}^{(n)} - old\_\widehat{\boldsymbol{U}}^{(n)}\|_F^2}{\sum_{n=1}^{N} \|\widehat{\boldsymbol{U}}^{(n)}\|_F^2}$
6. Estimate $\widehat{\mathcal{G}}_{T-d+1} = \sum_{i=1}^{p} a_i \widehat{\mathcal{G}}_{T-d-i}$
7. Compute $\mathcal{X}_{T-d+1} = \widehat{\mathcal{G}}_{T-d+1} \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)}$
8. Conduct inverse differencing to obtain $\widetilde{\mathcal{X}}^{new} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N \times (T+1)}$
9. Apply inverse MDT: $\mathcal{H}_r^{-1}(\widetilde{\mathcal{X}}^{new}) = \mathcal{X}^{new} \in \mathbb{R}^{J_1 \times J_2 \times ... \times J_M \times (T_{start}+1)}$
10. Return $\mathcal{X}_{T+1}^{new}$, the factor matrices $\widehat{\boldsymbol{U}}^{(n)}$ and the scalar coefficients a_i

3.3 Block Hankel Tensor Autoregression with Matrix Coefficients

The method that was discussed so far shows promising results that will be presented in Chapter 4. This serves as a trigger for further experimentation on the algorithm, in order to provide its generalization. As we have already mentioned, the algorithm's main components are Hankelization, Tucker Decomposition and the solution of an Autoregressive model with scalar coefficients. Thus, in an effort to make the first step towards the generalization of the algorithm, we substitute the scalar coefficients with matrices to extend its applicability to AR series with matrix-valued coefficients.

In general, autoregression with scalar coefficients is a special case of the more general model whose coefficients are matrix-valued. In the autoregression model that was considered so far, each coefficient can be interpreted as a weight that describes a previous tensor's contribution in the forecasted value. However, different elements of a tensor could weigh in differently. Furthermore, in this model each element is expressed as a weighted

summation of only the respective elements of the previous tensors. Thus, spatial correlations are not fully taken into consideration. On the other hand, its generalization, i.e Autoregression with matrix coefficients, is not bound by the aforementioned restrictions.

In this section, the goal is to derive a variation of the algorithm that overcomes the restrictions introduced by the scalar coefficient autoregressive process. We review autoregression with matrix coefficients and argue that this model overcomes these restrictions. Finally, we present proof that, autoregression with matrix coefficients can substitute the existing autoregressive model in the algorithm.

We have already mentioned that early approaches reshaped data of higher order structures into vectors. Such methods pose various disadvantages like increased time complexity, high memory demands and obscuring of any intrinsic structure information that was present in the data (in cases of matrices or tensors). However, in our case it is possible to overcome most of these disadvantages because Tucker Decomposition is utilized. The matrix coefficients are estimated over the vectorized form of the core matrices and as a result their size is greatly decreased compared to what it would have been if they were estimated on the vectorized form of the original tensors. In addition, this leads to a reduction of the number of estimated parameters. Thus, memory demands and time complexity remain relatively low. Finally, any intrinsic structure that was originally present in the data has already been modified in the Tucker decomposition step.

3.3.1 Vector Autoregression with Matrix Coefficients

This subsection is dedicated to Autoregression with matrix coefficients in order to provide a better understanding of the differences between those two autoregressive processes. A p-th order autoregressive process with scalar coefficients, for a vector valued time series is defined as:

$$\boldsymbol{x}_t = a_1\boldsymbol{x}_{t-1} + a_2\boldsymbol{x}_{t-2} + ... + a_p\boldsymbol{x}_{t-p} + \boldsymbol{e}_t$$

where $\boldsymbol{e}_t$ is a vector that corresponds to white noise and a_i are scalars. Therefore, given a vector $\boldsymbol{x}$, each element is written as a linear combination of the elements at the same index i, on the previous p vectors, plus a white noise quantity. For a given vector $\boldsymbol{x}_t = (x_t^1, ..., x_t^i, ..., x_t^n)$ we have:

$$x_t^i = a_1 x_{t-1}^i + a_2 x_{t-2}^i + ... + a_p x_{t-p}^i + e_t^i$$

This linear combination of elements at the same index does not take into consideration spatial correlations between different indices that could be utilized to produce a more representative model.

For simplicity, we focus on a second-order autoregression model and a vector-valued time series with $\boldsymbol{x}_t \in \mathbb{R}^2$. For an autoregressive process with scalar coefficients we would write:

$$\begin{bmatrix} x_t^1 \\ x_t^2 \end{bmatrix} = \begin{bmatrix} a_1 x_{t-1}^1 + a_2 x_{t-2}^1 \\ a_1 x_{t-1}^2 + a_2 x_{t-2}^2 \end{bmatrix} + \boldsymbol{e}_t$$

The above equation shows that the value x_t^1 is written as a summation of $a_1 x_{t-1}^1$ and $a_2 x_{t-2}^1$ and so, any useful correlation between x_t^1 and x_{t-1}^2, x_{t-2}^2 is not taken into consideration. This is the essence of what the matrix coefficient autoregression introduces into the model,

by substituting the scalar coefficients with matrices. Autoregression with matrix coefficients for a vector-valued time series is defined [10] as:

$$\boldsymbol{x}_t = \boldsymbol{c} + \boldsymbol{A}_1\boldsymbol{x}_{t-1} + \boldsymbol{A}_2\boldsymbol{x}_{t-2} + ... + \boldsymbol{A}_p\boldsymbol{x}_{t-p} + \boldsymbol{e}_t$$

where $\boldsymbol{x}_t \in \mathbb{R}^n$ are the time series observations, $\boldsymbol{c} = (\boldsymbol{I} - \sum_{i=1}^{p} \boldsymbol{A}_i)E[\boldsymbol{x}_t] \in \mathbb{R}^n$ serves as the intercept, $\boldsymbol{A}_i \in \mathbb{R}^{n\times n}$ are the matrix coefficients and $\boldsymbol{e}_t \in \mathbb{R}^n$ corresponds to white noise. Therefore, in the same setting as in the previous example, with $p = 2$, $\boldsymbol{x}_t \in \mathbb{R}^2$ and a^i_{jk} the element of the matrix $\boldsymbol{A}_i$ located at (j, k) we have:

$$\begin{bmatrix} x^1_t \\ x^2_t \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} a^1_{11} & a^1_{12} \\ a^1_{21} & a^1_{22} \end{bmatrix} \begin{bmatrix} x^1_{t-1} \\ x^2_{t-1} \end{bmatrix} + \begin{bmatrix} a^2_{11} & a^2_{12} \\ a^2_{21} & a^2_{22} \end{bmatrix} \begin{bmatrix} x^1_{t-2} \\ x^2_{t-2} \end{bmatrix} + \boldsymbol{e}_t \Rightarrow$$

$$\begin{bmatrix} x^1_t \\ x^2_t \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} a^1_{11}x^1_{t-1} + a^1_{12}x^2_{t-1} \\ a^1_{21}x^1_{t-1} + a^1_{22}x^2_{t-1} \end{bmatrix} + \begin{bmatrix} a^2_{11}x^1_{t-2} + a^2_{12}x^2_{t-2} \\ a^2_{21}x^1_{t-2} + a^2_{22}x^2_{t-2} \end{bmatrix} + \boldsymbol{e}_t \Rightarrow$$

$$\begin{bmatrix} x^1_t \\ x^2_t \end{bmatrix} = \begin{bmatrix} c_1 + a^1_{11}x^1_{t-1} + a^1_{12}x^2_{t-1} + a^2_{11}x^1_{t-2} + a^2_{12}x^2_{t-2} \\ c_2 + a^1_{21}x^1_{t-1} + a^1_{22}x^2_{t-1} + a^2_{21}x^1_{t-2} + a^2_{22}x^2_{t-2} \end{bmatrix} + \boldsymbol{e}_t$$

As we can see, autoregression with matrix coefficients expresses each element of a vector-valued time series at a given time point, as a combination of every element of the previous p observations, regardless of index, plus the intercept and white noise. This enables the model to capture correlations that otherwise would stay hidden. Note that the matrix coefficient model is essentially a generalization of the scalar coefficient model. Having zero-valued non diagonal elements and same-valued diagonal elements on a matrix $\boldsymbol{A}$ leads to a scalar matrix which essentially is the scalar coefficient autoregressive model.

3.3.2 Model Transformation

In this subsection it is shown that the Block Hankel Tensor Autoregression model with scalar coefficients can be generalized to a model with matrix coefficients. MDT is applied on the original time series to obtain its Hankelized form. Then, it is shown that an AR process of a tensor-valued time series is equivalent with a process that uses its vectorized form and therefore it can be generalized to an AR process that uses matrix coefficients. Finally, it is shown that this model can be further transformed into one that uses the core tensors obtained through Tucker Decomposition.

$$\mathcal{X}_t = \sum_{i=1}^{p} a_i\mathcal{X}_{t-i} + \mathcal{E}_t \iff$$

$$vec(\mathcal{X}_t) = \sum_{i=1}^{p} a_i vec(\mathcal{X}_{t-i}) + vec(\mathcal{E}_t) \iff$$

$$vec(\mathcal{X}_t) = \sum_{i=1}^{p} a_i \boldsymbol{I} vec(\mathcal{X}_{t-i}) + vec(\mathcal{E}_t) \iff$$

$$vec(\mathcal{X}_t) = \sum_{i=1}^{p} \boldsymbol{\Phi}_i vec(\mathcal{X}_{t-i}) + vec(\mathcal{E}_t)$$

where $\mathbf{\Phi}_i$ is a scalar matrix. Therefore, by replacing the scalar matrices $\mathbf{\Phi}_i$ we get the Autoregressive model with matrix coefficients. The next step is to find the equivalent model that utilizes the core tensors. For simplicity, we start with a third order tensor-valued time series and a first-order autoregressive process before generalizing to a p-th order autoregressive model for a N-th order tensor time series.

In the following proof each tensor is decomposed over a single mode. By repeating the process for each mode we obtain the final core tensors. The tensors that are obtained in each step can be interpreted as intermediate tensors between the original and the final core tensors. The n-th intermediate tensor is denoted as $\mathcal{M}_t^n \in \mathbb{R}^{R_1 \times \ldots \times R_n \times I_{n+1} \times \ldots \times I_N}$. It is straightforward that $\mathcal{M}_t^N = \widehat{\mathcal{G}}_t$.

For simplicity, we start by providing the proof for a third order tensor-valued time series and a first order autoregressive process. Then we generalize to an N-th order tensor-valued time series and a p-th order autoregressive model. Let $\mathcal{X}_t \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ a tensor-valued time series, $vec(\mathcal{X}_t) \in \mathbb{R}^{I_1 I_2 I_3}$ its vectorization, $\mathbf{\Phi} \in \mathbb{R}^{I_1 I_2 I_3 \times I_1 I_2 I_3}$ the coefficient matrix and $\widehat{\boldsymbol{U}}^{(1)} \in \mathbb{R}^{I_1 \times R_1}$, $\widehat{\boldsymbol{U}}^{(2)} \in \mathbb{R}^{I_2 \times R_2}$, $\widehat{\boldsymbol{U}}^{(3)} \in \mathbb{R}^{I_3 \times R_3}$ the factor matrices.

In the following proof we will use the property $vec(\boldsymbol{AB}) = (\boldsymbol{I} \otimes \boldsymbol{A}) vec(\boldsymbol{B})$

Starting with the vectorized model we have:

$$vec(\mathcal{X}_t) = \mathbf{\Phi} vec(\mathcal{X}_{t-1}) + vec(\mathcal{E}_t)$$

The vectorization of a tensor is a equivalent to the vectorization of its 1-st mode unfolding. Therefore, the above is equivalently written as:

$$vec(\boldsymbol{X}_t^{(1)}) = \mathbf{\Phi} vec(\boldsymbol{X}_{t-1}^{(1)}) + vec(\boldsymbol{E}_t^{(1)}) \iff$$

$$(\boldsymbol{I}_{I_2 I_3} \otimes \widehat{\boldsymbol{U}}^{(1)^\top}) vec(\boldsymbol{X}_t^{(1)}) = (\boldsymbol{I}_{I_2 I_3} \otimes \widehat{\boldsymbol{U}}^{(1)^\top}) \mathbf{\Phi} vec(\boldsymbol{X}_{t-1}^{(1)}) + (\boldsymbol{I}_{I_2 I_3} \otimes \widehat{\boldsymbol{U}}^{(1)^\top}) vec(\boldsymbol{E}_t^{(1)}) \iff$$

$$vec(\widehat{\boldsymbol{U}}^{(1)^\top} \boldsymbol{X}_t^{(1)}) = (\boldsymbol{I}_{I_2 I_3} \otimes \widehat{\boldsymbol{U}}^{(1)^\top}) \mathbf{\Phi} vec(\widehat{\boldsymbol{U}}^{(1)} \boldsymbol{M}_{t-1}^{1(1)}) + vec(\widehat{\boldsymbol{U}}^{(1)^\top} \boldsymbol{E}_t^{(1)}) \iff$$

$$vec(\boldsymbol{M}_t^{1(1)}) = (\boldsymbol{I}_{I_2 I_3} \otimes \widehat{\boldsymbol{U}}^{(1)^\top}) \mathbf{\Phi} (\boldsymbol{I}_{I_2 I_3} \otimes \widehat{\boldsymbol{U}}^{(1)}) vec(\boldsymbol{M}_{t-1}^{1(1)}) + vec(\boldsymbol{E}_t^{1(1)}) \iff$$

$$vec(\boldsymbol{M}_t^{1(1)}) = \mathbf{\Phi}^1 vec(\boldsymbol{M}_{t-1}^{1(1)}) + vec(\boldsymbol{E}_t^{1(1)}) \tag{3.15}$$

where $\boldsymbol{M}_t^{1(1)}, \boldsymbol{M}_{t-1}^{1(1)} \in \mathbb{R}^{R_1 \times I_2 I_3}$, $\mathbf{\Phi}^1 = (\boldsymbol{I}_{I_2 I_3} \otimes \widehat{\boldsymbol{U}}^{(1)^\top}) \mathbf{\Phi} (\boldsymbol{I}_{I_2 I_3} \otimes \widehat{\boldsymbol{U}}^{(1)}) \in \mathbb{R}^{R_1 I_2 I_3 \times R_1 I_2 I_3}$ and $\boldsymbol{E}_t^{1(1)} = \widehat{\boldsymbol{U}}^{(1)^\top} \boldsymbol{E}_t^{(1)} \in \mathbb{R}^{R_1 \times I_2 I_3}$.

At this point we decompose over the second mode. Since vectorizations of different unfoldings are not always equal we have to rearrange the elements of the first mode's unfolding to obtain the vectorized form of the second mode's unfolding.

For a given pair of vectors $\boldsymbol{u}_1, \boldsymbol{u}_2$ where $\boldsymbol{u}_2$ is a permutation of $\boldsymbol{u}_1$ there exists an invertible transition matrix $\boldsymbol{T}^{12}$ such as $\boldsymbol{T}^{12}\, \boldsymbol{u}_1 = \boldsymbol{u}_2 \iff \boldsymbol{u}_1 = \boldsymbol{T}^{21}\, \boldsymbol{u}_2$ where $\boldsymbol{T}^{21} = \boldsymbol{T}^{12^{-1}}$.

In our context the transition matrix permutes the vectorization of $\boldsymbol{M}_t^{1(1)}$ in a way equivalent to the following steps:

- Fold $\boldsymbol{M}_t^{1(1)}$ to obtain $\mathcal{M}_t^1$
- Unfold over the second mode to obtain $\boldsymbol{M}_t^{1(2)}$
- Vectorize the unfolded matrix $\boldsymbol{M}_t^{1(2)}$

The above can be summarized as:

$\boldsymbol{T}^{12}\ vec(\boldsymbol{M}^{1^{(1)}}) = vec(\boldsymbol{M}^{1^{(2)}}) \iff vec(\boldsymbol{M}^{1^{(1)}}) = \boldsymbol{T}^{21}\ vec(\boldsymbol{M}^{1^{(2)}})$.

Left-Multiplying (3.15) with $\boldsymbol{T}^{12}$ we get:

$$\boldsymbol{T}^{12}vec(\boldsymbol{M}_t^{1^{(1)}}) = \boldsymbol{T}^{12}\boldsymbol{\Phi}^1 vec(\boldsymbol{M}_{t-1}^{1^{(1)}}) + \boldsymbol{T}^{12}vec(\boldsymbol{E}_t^{1^{(1)}})$$

$$vec(\boldsymbol{M}_t^{1^{(2)}}) = \boldsymbol{T}^{12}\boldsymbol{\Phi}^1\ \boldsymbol{T}^{21}vec(\boldsymbol{M}_{t-1}^{1^{(2)}}) + vec(\boldsymbol{E}_t^{1^{(2)}})$$

$$(\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)^\top})vec(\boldsymbol{M}_t^{1^{(2)}}) = (\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)^\top})\boldsymbol{T}^{12}\boldsymbol{\Phi}^1\ \boldsymbol{T}^{21}vec(\widehat{\boldsymbol{U}}^{(2)}\boldsymbol{M}_{t-1}^{2^{(2)}}) + (\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)^\top})vec(\boldsymbol{E}_t^{1^{(2)}})$$

$$vec(\widehat{\boldsymbol{U}}^{(2)^\top}\boldsymbol{M}_t^{1^{(2)}}) = (\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)^\top})\boldsymbol{T}^{12}\boldsymbol{\Phi}^1\ \boldsymbol{T}^{21}(\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)})vec(\boldsymbol{M}_{t-1}^{2^{(2)}}) + vec(\widehat{\boldsymbol{U}}^{(2)^\top}\boldsymbol{E}_t^{1^{(2)}})$$

$$vec(\boldsymbol{M}_t^{2^{(2)}}) = \boldsymbol{\Phi}^2 vec(\boldsymbol{M}_{t-1}^{2^{(2)}}) + vec(\boldsymbol{E}_t^{2^{(2)}})$$

where $\boldsymbol{\Phi}^2 = (\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)^\top})\boldsymbol{T}^{12}\boldsymbol{\Phi}^1\ \boldsymbol{T}^{21}(\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)}) \in \mathbb{R}^{R_1R_2I_3\times R_1R_2I_3}$ and $\boldsymbol{E}_t^{2^{(2)}}, \boldsymbol{M}_t^{2^{(2)}}, \boldsymbol{M}_{t-1}^{2^{(2)}} \in \mathbb{R}^{R_2\times R_1I_3}$.

Finally, left-multiplying with $\boldsymbol{T}^{23}$ we get:

$$\boldsymbol{T}^{23}vec(\boldsymbol{M}_t^{2^{(2)}}) = \boldsymbol{T}^{23}\boldsymbol{\Phi}^2 vec(\boldsymbol{M}_{t-1}^{2^{(2)}}) + \boldsymbol{T}^{23}vec(\boldsymbol{E}_t^{2^{(2)}})$$

$$vec(\boldsymbol{M}_t^{2^{(3)}}) = \boldsymbol{T}^{23}\boldsymbol{\Phi}^2\ \boldsymbol{T}^{32}vec(\boldsymbol{M}_{t-1}^{2^{(3)}}) + vec(\boldsymbol{E}_t^{2^{(3)}})$$

$$(\boldsymbol{I}_{R_1R_2}\otimes\widehat{\boldsymbol{U}}^{(3)^\top})vec(\boldsymbol{M}_t^{2^{(3)}}) = (\boldsymbol{I}_{R_1R_2}\otimes\widehat{\boldsymbol{U}}^{(3)^\top})\boldsymbol{T}^{23}\boldsymbol{\Phi}^2\ \boldsymbol{T}^{32}vec(\widehat{\boldsymbol{U}}^{(3)}\boldsymbol{M}_{t-1}^{3^{(3)}}) + (\boldsymbol{I}_{R_1R_2}\otimes\widehat{\boldsymbol{U}}^{(3)^\top})vec(\boldsymbol{E}_t^{2^{(3)}})$$

$$vec(\widehat{\boldsymbol{U}}^{(3)^\top}\boldsymbol{M}_t^{2^{(3)}}) = (\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(3)^\top})\boldsymbol{T}^{23}\boldsymbol{\Phi}^2\ \boldsymbol{T}^{32}(\boldsymbol{I}_{R_1R_2}\otimes\widehat{\boldsymbol{U}}^{(3)})vec(\boldsymbol{M}_{t-1}^{3^{(3)}}) + vec(\widehat{\boldsymbol{U}}^{(3)^\top}\boldsymbol{E}_t^{2^{(3)}})$$

$$vec(\boldsymbol{M}_t^{3^{(3)}}) = \boldsymbol{\Phi}^3 vec(\boldsymbol{M}_{t-1}^{3^{(3)}}) + vec(\boldsymbol{E}_t^{3^{(3)}})$$

$$\boldsymbol{T}^{31}vec(\boldsymbol{M}_t^{3^{(3)}}) = \boldsymbol{T}^{31}\boldsymbol{\Phi}^3\boldsymbol{T}^{13}vec(\boldsymbol{M}_{t-1}^{3^{(1)}}) + \boldsymbol{T}^{31}vec(\boldsymbol{E}_t^{3^{(3)}})$$

$$vec(\boldsymbol{M}_t^{3^{(1)}}) = \boldsymbol{A}vec(\boldsymbol{M}_{t-1}^{3^{(1)}}) + vec(\boldsymbol{E}_t^{3^{(1)}})$$

$$vec(\widehat{\boldsymbol{G}}_t^{(1)}) = \boldsymbol{A}vec(\widehat{\boldsymbol{G}}_{t-1}^{(1)}) + vec(\boldsymbol{E}_t^{3^{(1)}})$$

$$vec(\widehat{\mathcal{G}}_t) = \boldsymbol{A}vec(\widehat{\mathcal{G}}_{t-1}) + vec(\mathcal{E}_t)$$

This process can be summarized as a left-multiplication of (3.15) with:

$$\boldsymbol{T}^{31}(\boldsymbol{I}_{R_1R_2}\otimes\widehat{\boldsymbol{U}}^{(3)^T})\boldsymbol{T}^{23}(\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)^T})\boldsymbol{T}^{12}(\boldsymbol{I}_{I_2I_3}\otimes\widehat{\boldsymbol{U}}^{(1)^T}) = \prod_{n=3}^{1}\boldsymbol{T}^{n[(n+1)mod3]}(\boldsymbol{I}_{C_n}\otimes\widehat{\boldsymbol{U}}^{(n)^\top})$$

and writing $vec(\boldsymbol{X}_{t-1}^{(1)})$ as:

$$\begin{aligned} vec(\boldsymbol{X}_{t-1}^{(1)}) &= (\boldsymbol{I}_{I_2I_3}\otimes\widehat{\boldsymbol{U}}^{(1)})\boldsymbol{T}^{21}(\boldsymbol{I}_{R_1I_3}\otimes\widehat{\boldsymbol{U}}^{(2)})\boldsymbol{T}^{32}(\boldsymbol{I}_{R_1R_2}\otimes\widehat{\boldsymbol{U}}^{(3)})\boldsymbol{T}^{13}vec(\widehat{\boldsymbol{G}}_{t-1}^{(1)}) \\ &= \prod_{k=1}^{3}(\boldsymbol{I}_{C_k}\otimes\widehat{\boldsymbol{U}}^{(k)})\boldsymbol{T}^{[(k+1)mod3]k}vec(\widehat{\boldsymbol{G}}_{t-1}^{(1)}) \end{aligned}$$

where $C_n = \prod_{i=1}^{n-1} R_i \prod_{j=n+1}^{N} I_j$

The above can be generalized to an N-th order tensor-valued time series and a p-th order autoregressive process as a left-multiplication of (3.15) with:

$$\boldsymbol{F_1} = \prod_{n=N}^{1} \boldsymbol{T}^{n[(n+1)modN]}(\boldsymbol{I}_{C_n} \otimes \widehat{\boldsymbol{U}}^{(n)^\top})$$

and expressing $vec(\boldsymbol{X}_{t-i}^{(1)})$ as: $\prod_{n=1}^{N}(\boldsymbol{I}_{C_n} \otimes \widehat{\boldsymbol{U}}^{(n)})\boldsymbol{T}^{[(n+1)modN]n} vec(\widehat{\boldsymbol{G}}_{t-i}^{(1)}) = \boldsymbol{F_2} vec(\widehat{\boldsymbol{G}}_{t-i}^{(1)})$

where $\boldsymbol{F}_2 = \prod_{n=1}^{N}(\boldsymbol{I}_{C_n} \otimes \widehat{\boldsymbol{U}}^{(n)})\boldsymbol{T}^{[(n+1)modN]n}$.

We have:

$$vec(\mathcal{X}_t) = \sum_{i=1}^{p} \boldsymbol{\Phi}_i vec(\mathcal{X}_{t-i}) + vec(\mathcal{E}_t) \iff$$

$$vec(\boldsymbol{X}_t^{(1)}) = \sum_{i=1}^{p} \boldsymbol{\Phi}_i vec(\boldsymbol{X}_{t-i}^{(1)}) + vec(\boldsymbol{E}_t^{(1)}) \iff$$

$$\boldsymbol{F}_1 vec(\boldsymbol{X}_t^{(1)}) = \boldsymbol{F}_1 \sum_{i=1}^{p} \boldsymbol{\Phi}_i \boldsymbol{F_2} vec(\widehat{\boldsymbol{G}}_{t-i}^{(1)}) + \boldsymbol{F}_1 vec(\boldsymbol{E}_t^{(1)}) \iff$$

$$vec(\widehat{\boldsymbol{G}}_t^{(1)}) = \sum_{i=1}^{p} \boldsymbol{F}_1 \boldsymbol{\Phi}_i \boldsymbol{F_2} vec(\widehat{\boldsymbol{G}}_{t-i}^{(1)}) + vec(\boldsymbol{E'}_t^{(1)}) \iff$$

$$vec(\widehat{\boldsymbol{G}}_t^{(1)}) = \sum_{i=1}^{p} \boldsymbol{A}_i vec(\widehat{\boldsymbol{G}}_{t-i}^{(1)}) + vec(\boldsymbol{E'}_t^{(1)}) \iff$$

$$vec(\widehat{\mathcal{G}}_t) = \sum_{i=1}^{p} \boldsymbol{A}_i vec(\widehat{\mathcal{G}}_{t-i}) + vec(\mathcal{E'}_t) \iff$$

$$\widehat{\mathcal{G}}_t = fold(\sum_{i=1}^{p} \boldsymbol{A}_i vec(\widehat{\mathcal{G}}_{t-i})) + \mathcal{E'}_t \iff$$

$$\widehat{\mathcal{G}}_t = \sum_{i=1}^{p} fold(\boldsymbol{A}_i vec(\widehat{\mathcal{G}}_{t-i})) + \mathcal{E'}_t$$

Therefore, the optimization problem of (3.8) can be written as:

$$\sum_{t=p+1}^{T} \sum_{n=1}^{N} \left(\frac{1}{2} \| \widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} fold(\boldsymbol{A}_i \, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)} \|_F^2 + \frac{1}{2} \| \widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \|_F^2 \right) \quad (3.16)$$

3.3.3 Updating the core tensors

To compute the partial derivative of (3.16) with respect to $\widehat{\boldsymbol{G}}_t^{(n)}$ we write both norms as traces and follow the same procedure as in 3.2.1. The first norm is written equivalently as

$$\| \widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} fold(\boldsymbol{A}_i \, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)} \|_F^2 =$$

$$tr\Big((\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} fold(\boldsymbol{A}_i \, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)})^\top (\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{j=1}^{p} fold(\boldsymbol{A}_j \, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}) \Big) =$$

$$tr\Big((\widehat{\boldsymbol{G}}_t^{(n)^\top} - (\sum_{i=1}^{p} fold(\boldsymbol{A}_i\, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)})^\top)(\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)})\Big) =$$

$$tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)} - (\sum_{i=1}^{p} fold(\boldsymbol{A}_i\, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)})^\top\widehat{\boldsymbol{G}}_t^{(n)}$$
$$+ (\sum_{i=1}^{p} fold(\boldsymbol{A}_i\, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)})^\top(\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)})\Big) =$$

$$tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)} - \widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)} - (\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{i=1}^{p} fold(\boldsymbol{A}_i\, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)})^\top + \boldsymbol{B}\Big) \overset{(*)}{=}$$

$$tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\Big) - 2\,tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}\Big) + tr(\boldsymbol{B})$$

where $\boldsymbol{B} = (\sum_{i=1}^{p} fold(\boldsymbol{A}_i\, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)})^\top(\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)})$

Note that in the last step we used the following property of traces: $tr(\boldsymbol{A}) = tr(\boldsymbol{A}^\top)$

The partial derivatives of the traces above are:

- $\frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\,tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\right) = 2\widehat{\boldsymbol{G}}_t^{(n)}$
- $\frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\,tr\left(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}\right) = \sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}$
- $\frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\,tr\,(\boldsymbol{B}) = \frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\,tr\left((\sum_{i=1}^{p} fold(\boldsymbol{A}_i\, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)})^\top(\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)})\right) = 0$

The partial derivative of the first term with respect to $\widehat{\boldsymbol{G}}_t^{(n)}$ is:

$$\frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\left(\frac{1}{2}\,\|\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{i=1}^{p} fold(\boldsymbol{A}_i\, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)}\|_F^2\right) =$$

$$\frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\left(\frac{1}{2}\left(tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\Big) - 2\,tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}\Big) + tr(\boldsymbol{B})\right)\right) =$$

$$\frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\Big(\frac{1}{2}\,tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\Big) - tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}\Big) + \frac{1}{2}\,tr(\boldsymbol{B})\Big) =$$

$$\frac{1}{2}\,\frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\,tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\widehat{\boldsymbol{G}}_t^{(n)}\Big) - \frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\,tr\Big(\widehat{\boldsymbol{G}}_t^{(n)^\top}\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}\Big) + \frac{1}{2}\,\frac{\partial}{\partial\widehat{\boldsymbol{G}}_t^{(n)}}\,tr\,(\boldsymbol{B}) =$$

$$\widehat{\boldsymbol{G}}_t^{(n)} - \sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}$$

Summarizing the above we have:

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}}\left(\frac{1}{2}\|\widehat{\boldsymbol{G}}_t^{(n)}-\sum_{i=1}^{p} fold(\boldsymbol{A}_i\, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)}\|_F^2\right)=\widehat{\boldsymbol{G}}_t^{(n)}\ -\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)} \quad (3.17)$$

The partial derivative of the second norm as calculated in (3.2.1) is:

$$\frac{\partial}{\partial \widehat{\boldsymbol{G}}_t^{(n)}}\left(\frac{1}{2}\|\widehat{\boldsymbol{G}}_t^{(n)}-\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\|_F^2\right)=\widehat{\boldsymbol{G}}_t^{(n)}-\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top} \quad (3.18)$$

Using (3.17) & (3.18) we compute the partial derivative of (3.16) with respect to $\widehat{\boldsymbol{G}}_t^{(n)}$. By setting it to zero we get:

$$\sum_{t=p+1}^{T}\left(\widehat{\boldsymbol{G}}_t^{(n)}-\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}+\widehat{\boldsymbol{G}}_t^{(n)}-\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\right)=0 \iff$$

$$\sum_{t=p+1}^{T}\left(2\widehat{\boldsymbol{G}}_t^{(n)}-\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}-\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\right)=0 \iff$$

$$\widehat{\boldsymbol{G}}_t^{(n)}=\frac{1}{2}\left(\sum_{j=1}^{p} fold(\boldsymbol{A}_j\, vec(\widehat{\mathcal{G}}_{t-j}))^{(n)}+\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\right) \quad (3.19)$$

3.3.4 Updating the factor matrices

(3.16) with respect to $\widehat{\boldsymbol{U}}^{(n)}$ is:

$$\underset{\{\widehat{\boldsymbol{U}}^{(n)}\}}{\text{argmin}}\sum_{t=p+1}^{T}\sum_{n=1}^{N}\left(\frac{1}{2}\|\widehat{\boldsymbol{G}}_t^{(n)}-\widehat{\boldsymbol{U}}^{(n)^\top}\boldsymbol{X}_t^{(n)}\widehat{\boldsymbol{U}}^{(-n)^\top}\|_F^2\right)$$

$$\text{subject to}\ \ \widehat{\boldsymbol{U}}^{(n)^\top}\widehat{\boldsymbol{U}}^{(n)}=\boldsymbol{I}_{R_n}\ \text{for}\ n=1,2,...,N$$

which is minimized by (3.14) as shown in section 3.2.2

3.3.5 Estimating the Matrix Coefficients

A p-th order autoregressive process with n variables and $T+1$ observations can be written in a matrix concise form as $\boldsymbol{X}=\boldsymbol{A}\boldsymbol{Z}+\boldsymbol{E}$

where $\boldsymbol{X}=[\boldsymbol{x}_p,\boldsymbol{x}_{p+1},...,\boldsymbol{x}_T]\in\mathbb{R}^{n\times(T-p+1)}$, $\boldsymbol{A}=[c,\boldsymbol{A}_1,...,\boldsymbol{A}_p]\in\mathbb{R}^{n\times(np+1)}$,

$$\boldsymbol{Z}=\begin{bmatrix}1 & 1 & ... & 1\\ \boldsymbol{x}_{p-1} & \boldsymbol{x}_p & ... & \boldsymbol{x}_{T-1}\\ \boldsymbol{x}_{p-2} & \boldsymbol{x}_{p-1} & ... & \boldsymbol{x}_{T-2}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{x}_0 & \boldsymbol{x}_1 & ... & \boldsymbol{x}_{T-p}\end{bmatrix}\in\mathbb{R}^{(np+1)\times(T-p+1)}\ \text{and}\ \boldsymbol{E}=[\boldsymbol{e}_p,\boldsymbol{e}_{p+1}....,\boldsymbol{e}_T]\in\mathbb{R}^{n\times(T-p+1)}$$

Using a *Multivariate Least Squares* approach we estimate $\boldsymbol{A}$ by minimizing the following:

$$\begin{aligned}
\|\boldsymbol{E}\|_F^2 &= \|\boldsymbol{X} - \boldsymbol{AZ}\|_F^2 \\
&= tr\Big((\boldsymbol{X} - \boldsymbol{AZ})^\top(\boldsymbol{X} - \boldsymbol{AZ})\Big) \\
&= tr\Big((\boldsymbol{X}^\top - (\boldsymbol{AZ})^\top)(\boldsymbol{X} - \boldsymbol{AZ})\Big) \\
&= tr\Big(\boldsymbol{X}^\top\boldsymbol{X} - \boldsymbol{X}^\top\boldsymbol{AZ} - (\boldsymbol{AZ})^\top\boldsymbol{X} + (\boldsymbol{AZ})^\top\boldsymbol{AZ}\Big) \\
&= tr\Big(\boldsymbol{X}^\top\boldsymbol{X}\Big) - tr\Big(\boldsymbol{X}^\top\boldsymbol{AZ}\Big) - tr\Big((\boldsymbol{AZ})^\top\boldsymbol{X}\Big) + tr\Big((\boldsymbol{AZ})^\top\boldsymbol{AZ}\Big) \\
&= tr\Big(\boldsymbol{X}^\top\boldsymbol{X}\Big) - tr\Big(\boldsymbol{X}^\top\boldsymbol{AZ}\Big) - tr\Big((\boldsymbol{X}^\top\boldsymbol{AZ})^\top\Big) + tr\Big(\boldsymbol{Z}^\top\boldsymbol{A}^\top\boldsymbol{AZ}\Big) \\
&= tr\Big(\boldsymbol{X}^\top\boldsymbol{X}\Big) - 2tr\Big(\boldsymbol{X}^\top\boldsymbol{AZ}\Big) + tr\Big(\boldsymbol{A}^\top\boldsymbol{AZZ}^\top\Big)
\end{aligned}$$

Computing the partial derivative with respect to $\boldsymbol{A}$ we have:

- $\dfrac{\partial}{\partial \boldsymbol{A}} tr\left(\boldsymbol{X}^\top\boldsymbol{X}\right) = 0$
- $\dfrac{\partial}{\partial \boldsymbol{A}} tr\left(\boldsymbol{X}^\top\boldsymbol{AZ}\right) = (\boldsymbol{X}^\top)^\top\boldsymbol{Z}^\top = \boldsymbol{XZ}^\top$
- $\dfrac{\partial}{\partial \boldsymbol{A}} tr\left(\boldsymbol{A}^\top\boldsymbol{AZZ}^\top\right) = \boldsymbol{A}(\boldsymbol{ZZ}^\top)^\top + \boldsymbol{AZZ}^\top = \boldsymbol{A}(\boldsymbol{Z}^\top)^\top\boldsymbol{Z}^\top + \boldsymbol{AZZ}^\top = 2\boldsymbol{AZZ}^\top$

Thus, we have:

$$\frac{\partial}{\partial \boldsymbol{A}} \|\boldsymbol{X} - \boldsymbol{AZ}\|_F^2 = -2\boldsymbol{XZ}^\top + 2\boldsymbol{AZZ}^\top$$

By setting it to zero we get:

$$-2\boldsymbol{XZ}^\top + 2\boldsymbol{AZZ}^\top = 0$$

$$\boldsymbol{AZZ}^\top = \boldsymbol{XZ}^\top \tag{3.20}$$

Provided that $\boldsymbol{ZZ}^\top$ is invertible (i.e the rows of $\boldsymbol{Z}$ are linearly independent) we estimate $\boldsymbol{A}$ by:

$$\widehat{\boldsymbol{A}} = \boldsymbol{XZ}^\top(\boldsymbol{ZZ}^\top)^{-1} \tag{3.21}$$

If $ZZ^\top$ is not invertible the optimal solution of (3.20) in a least squares manner is given by:

$$\widehat{\boldsymbol{A}} = \boldsymbol{XZ}^\top(\boldsymbol{ZZ}^\top)^\dagger \tag{3.22}$$

3.3.6 Forecasting

Finally, after the coefficient estimation process is finished, the next core tensor is forecasted as:

$$\widehat{\mathcal{G}}_{T+1} = \sum_{i=1}^{p} fold(\boldsymbol{A}_i vec(\widehat{\mathcal{G}}_{T-i+1})) \in \mathbb{R}^{R_1 \times R_2 \times \ldots \times R_N}$$

The forecast is concatenated with the rest of the core tensors along the temporal mode, resulting in the tensor $\widehat{\mathcal{G}}_{new} \in \mathbb{R}^{R_1 \times R_2 \times \ldots \times R_N \times (T+1)}$. After $\widehat{\mathcal{G}}_{new}$ is obtained, it is transformed back into the tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times (T+1)}$ using the estimated factor matrices $\widehat{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, $n = 1, 2, ..., N$. Finally, the inverse process of MDT is applied on the Hankelized tensor $\mathcal{X}$ to obtain the original time series $\mathcal{X}_{start} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_M \times (T_{start}+1)}$.

3.3.7 The Algorithm

The algorithm requires as input the time series $\mathcal{X}_{start} \in \mathbb{R}^{J_1 \times ... \times J_M \times T_{start}}$, the AR order *p*, the maximum number of iterations max_iter, the MDT order r and the tolerance for the stop criterion tol. Finally, to incorporate non-stationary time series we conduct d-th order differencing.

BHT_AR_MC

1. Apply MDT to get $\widetilde{\mathcal{X}} = \mathcal{H}_r(\mathcal{X}_{start}) \in \mathbb{R}^{I_1 \times I_2 \times ... I_N \times T}$, where $T = T_{start} - r + 1$
2. Conduct d-order differencing on $\widetilde{\mathcal{X}}$ and get $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N \times (T-d)}$
3. Set the Tucker Decomposition Ranks, R_n, $n = 1, 2, ..., N$
4. Set $iter = 0$ and initialize $\widehat{\boldsymbol{U}}^{(n)}$, $n = 1, 2, ..., N$ randomly
5. While $iter \leq max\_iter$ and $convergence\_value > tol$
   (a) Estimate $\widehat{\mathcal{G}}_t = \mathcal{X}_t \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)^\top}$, for $t = 1, 2, ..., T - d$
   (b) Estimate the matrix coefficients $\boldsymbol{A}_i$ by (3.22) using $vec(\widehat{\mathcal{G}}_t)$, for $t = 1, 2, ..., T - d$
   (c) For $n = 1, 2, ..., N$
      i. By (3.19) update $\widehat{\boldsymbol{G}}_t^{(n)} = \frac{1}{2}\left(\sum_{i=1}^{p} fold(\boldsymbol{A}_i \, vec(\widehat{\mathcal{G}}_{t-i}))^{(n)} + + \widehat{\boldsymbol{U}}^{(n)^\top} \boldsymbol{X}_t^{(n)} \widehat{\boldsymbol{U}}^{(-n)^\top} \right)$ for $t = p + 1, ..., T - d$
      ii. $old\_\widehat{\boldsymbol{U}}^{(n)} = \widehat{\boldsymbol{U}}^{(n)}$
      iii. By (3.14) update $\widehat{\boldsymbol{U}}^{(n)} = \boldsymbol{L}\boldsymbol{R}^\top$
   (d) $iter + +$
   (e) $convergence\_value = \frac{\sum_{n=1}^{N} \| \widehat{\boldsymbol{U}}^{(n)} - old\_\widehat{\boldsymbol{U}}^{(n)} \|_F^2}{\sum_{n=1}^{N} \| \widehat{\boldsymbol{U}}^{(n)} \|_F^2}$
6. Estimate $\widehat{\mathcal{G}}_{T-d+1} = \sum_{i=1}^{p} fold(\boldsymbol{A}_i vec(\widehat{\mathcal{G}}_{T-d+1-i}))$
7. Compute $\mathcal{X}_{T-d+1} = \widehat{\mathcal{G}}_{T-d+1} \prod_{n=1}^{N} \times_n \widehat{\boldsymbol{U}}^{(n)}$
8. Conduct inverse differencing to obtain $\widetilde{\mathcal{X}}^{new} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_N \times (T+1)}$
9. Apply inverse MDT: $\mathcal{H}_r^{-1}(\widetilde{\mathcal{X}}^{new}) = \mathcal{X}^{new} \in \mathbb{R}^{J_1 \times J_2 \times ... \times J_M \times (T_{start}+1)}$
10. Return $\mathcal{X}_{T+1}^{new}$, the factor matrices $\widehat{\boldsymbol{U}}^{(n)}$ and the matrix coefficients $\boldsymbol{A}_i$

4. EXPERIMENTAL SETUP

In this section, we present and discuss the results of the experiments that were conducted for the evaluation of the Block Hankel Tensor Autoregression algorithm (with scalar coefficients) and its matrix coefficient generalization.

4.1 Datasets

In the evaluation process, a series of experiments are conducted on 1 synthetic and 7 publicly available datasets. Each dataset is further described in the following paragraphs.

Synthetic Data

As a starting point and in order to conduct experiments with some control over the data, we generate a first order, vector-valued autoregressive process with matrix-valued coefficients utilizing the following model:

$$\boldsymbol{x}_t = \boldsymbol{c} + \boldsymbol{A}\boldsymbol{x}_{t-1} + \boldsymbol{e}_t \tag{4.1}$$

where $\boldsymbol{x}_t, \boldsymbol{x}_{t-1} \in \mathbb{R}^3$ are two consecutive observations of the time series, $\boldsymbol{c} \in \mathbb{R}^3$ is the intercept, $\boldsymbol{A} \in \mathbb{R}^{3\times 3}$ is the coefficient matrix and $\boldsymbol{e}_t \in \mathbb{R}^3$ corresponds to white noise.

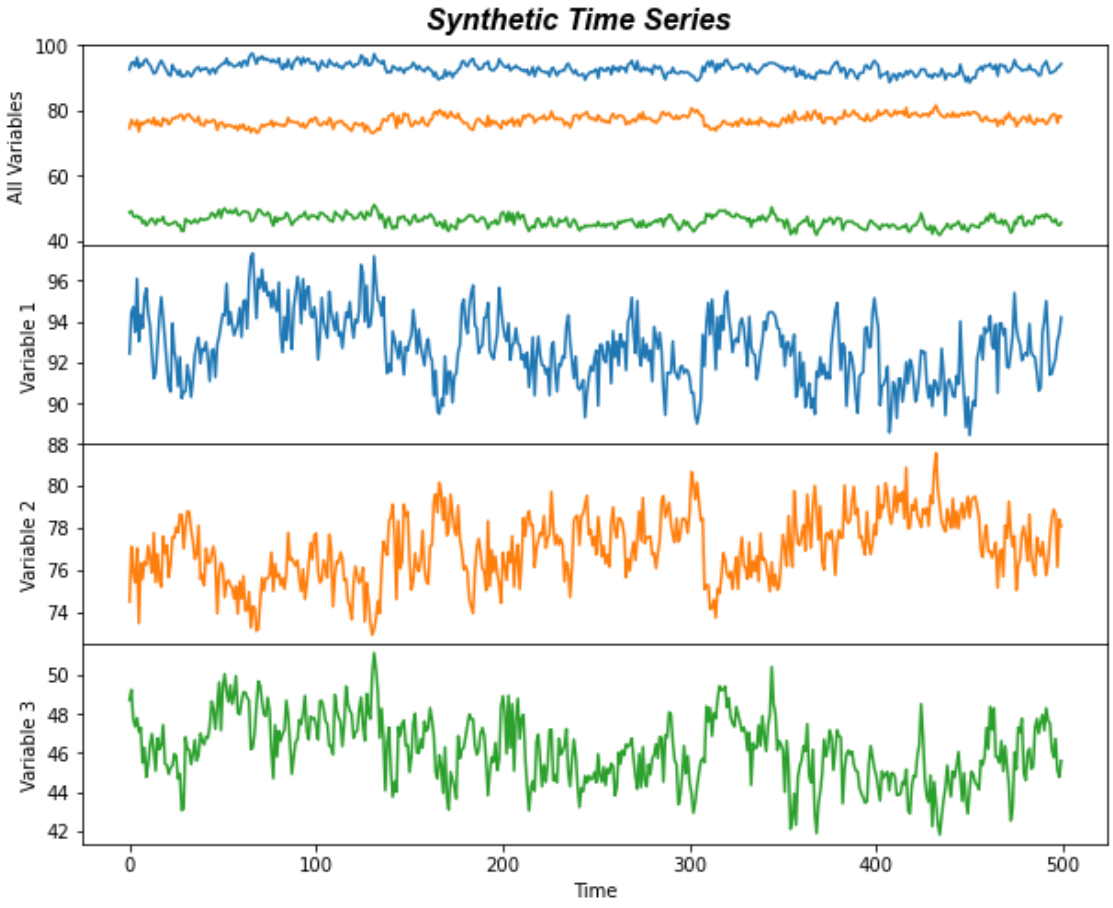

Figure 10: Sample of 500 observations for the generated time series

To generate a stable first-order Autoregressive process with matrix coefficients [11] we set

$$\boldsymbol{c} = \begin{bmatrix} 73.23 \\ 67.59 \\ 67.46 \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} .46 & -.36 & .10 \\ -.24 & .49 & -.13 \\ -.12 & -.48 & .58 \end{bmatrix}$$

on (4.1) and generate $\boldsymbol{e}_t$ as a sample of white noise. Finally, to obtain a sample of N observations, we generate 30% more and keep only the last N observations. By doing so, we discard the initialization part of the generated data, which will not be representative of the time series.

United States Macroeconomic Dataset

The dataset contains 203 observations of a vector-valued time series with the following 12 variables.

- **realgdp:** Real gross domestic product
- **realcons:** Real personal consumption expenditures
- **realinv:** Real gross private domestic investment
- **realgovt:** Real federal consumption expenditures & gross investment
- **realdpi:** Real private disposable income
- **cpi:** End of the quarter consumer price index for all urban consumers
- **m1:** End of the quarter M1 nominal money stock (Seasonally adjusted)
- **tbilrate:** Quarterly monthly average of the monthly 3-month treasury bill
- **unemp:** Seasonally adjusted unemployment rate %
- **pop:** End of the quarter total population
- **infl:** Inflation rate
- **realint:** Real interest rate

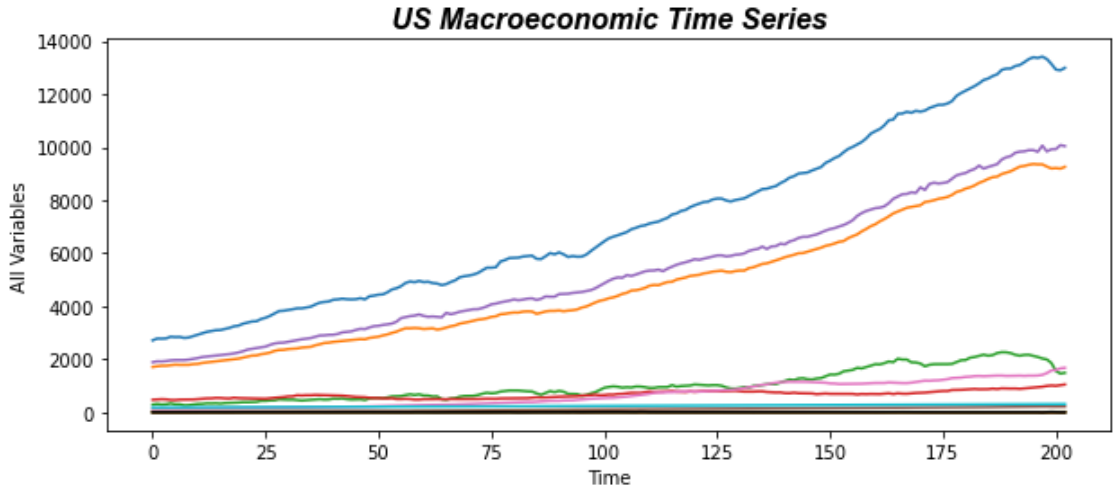

Figure 11: The US Macroeconomic Dataset

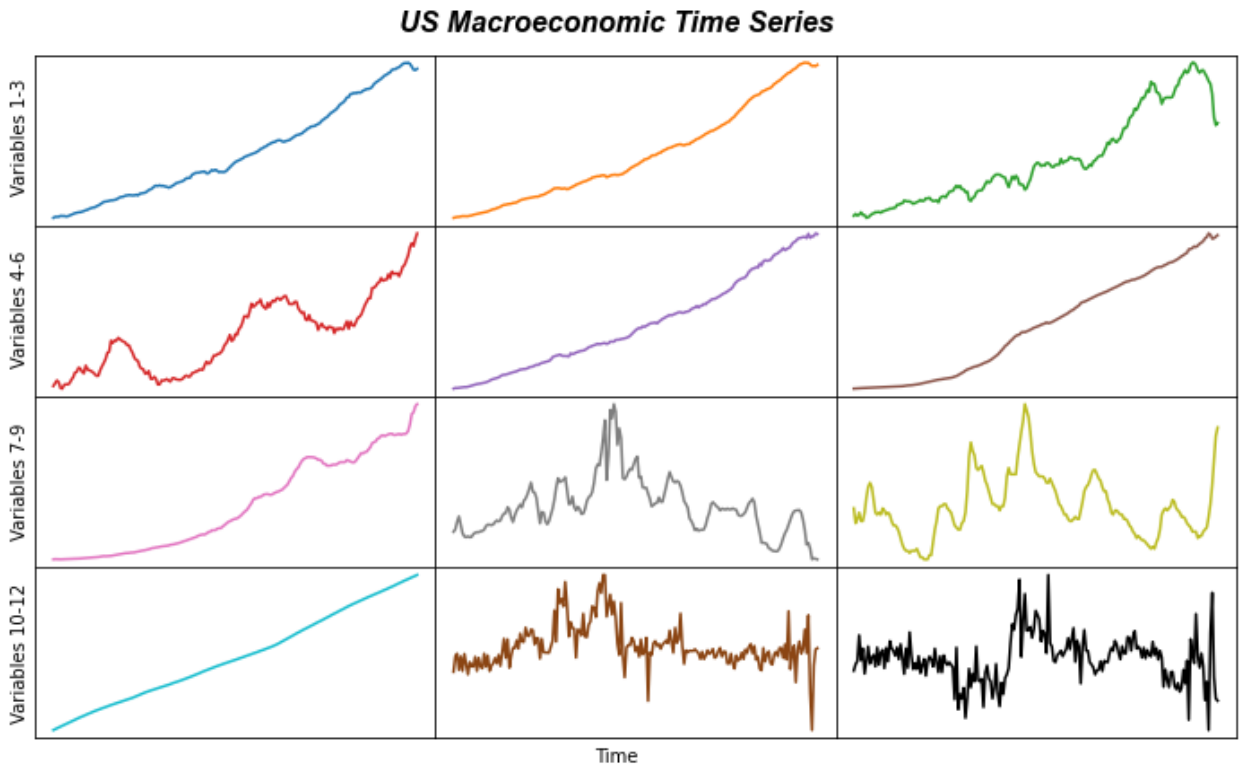

Figure 12: US Macroeconomic Individual Time Series

El Nino - Sea Surface Temperatures Dataset

The dataset contains 61 observations of a vector-valued time series with 12 variables. Each variable corresponds to a month of the year. Thus, forecasting 1 step into the future equals to forecasting the temperature of the sea surface for each month of the next year.

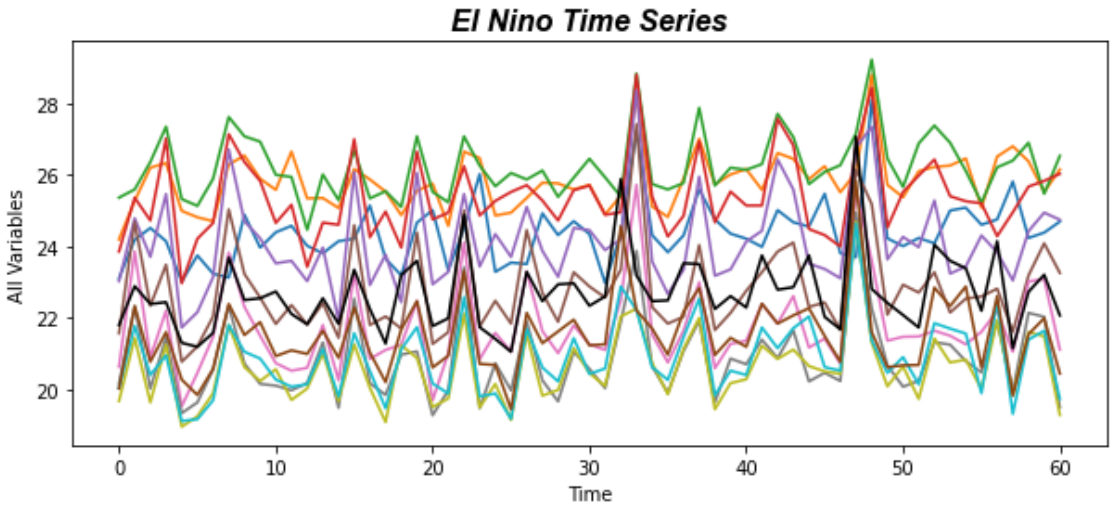

Figure 13: The El-Nino dataset

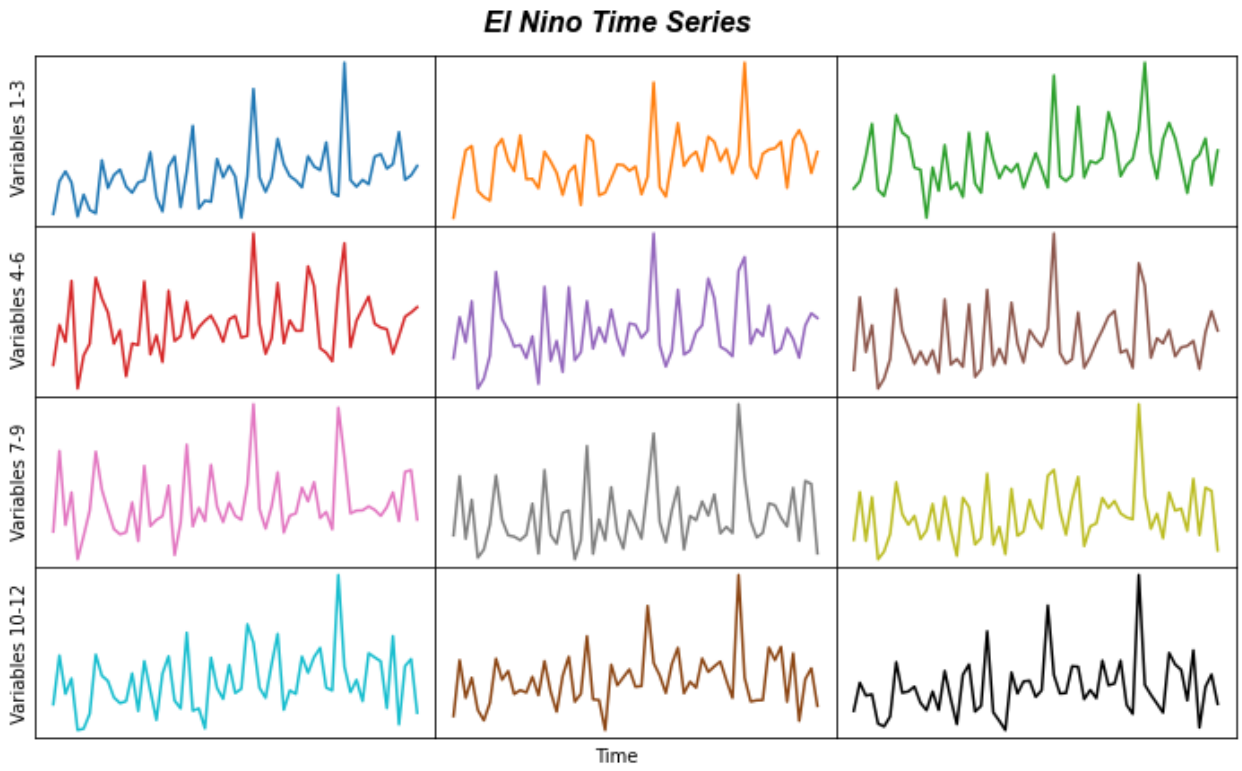

Figure 14: El-Nino Individual Time Series

Stack Loss Dataset

The dataset contains 21 measurements from a plant's oxidation of ammonia to nitric acid. The observations form a vector-valued time series with the following 4 variables.

- **stackloss:** 10 times the percentage of ammonia going into the plant that escapes from the absorption column
- **airflow:** Rate of operation of the plant
- **watertemp:** Cooling water temperature in the absorption tower
- **acidconc:** Acid concentration of circulating acid minus 50 times 10

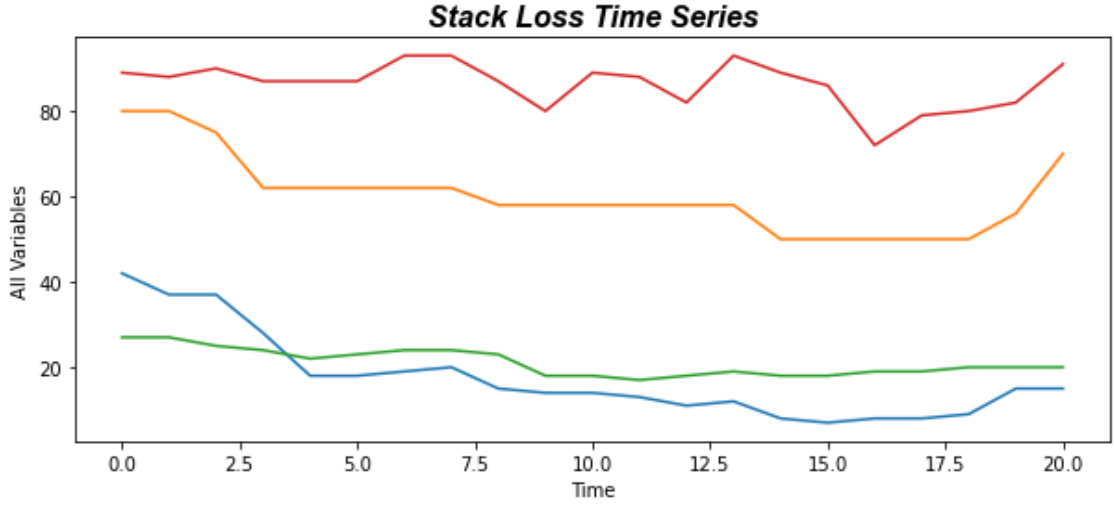

Figure 15: The Stack Loss Dataset

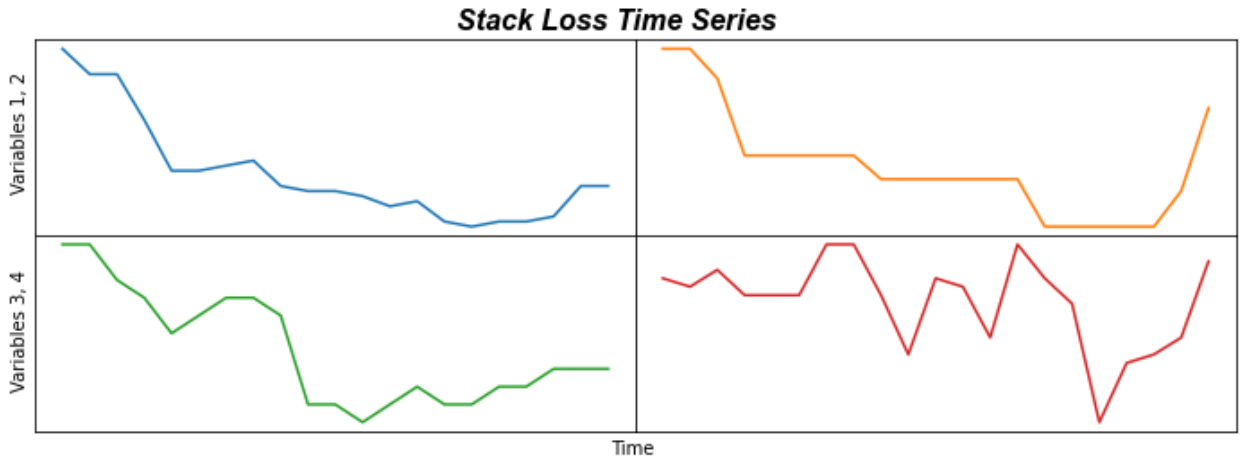

Figure 16: Stack Loss Individual Time Series

Ozone Dataset

The dataset contains 203 observations of a vector-valued time series with the following 8 variables.

- ozone reading: The Ozone level value that was observed

- wind speed: The speed in which wind travels

- humidity: The levels of humidity

- temperature Sandburg: The temperature that was observed in Sandburg, California

- temperature El Monte: The temperature that was observed in El Monte, California

- pressure gradient: The observed pressure gradient value

- inversion temperature: The observed inversion temperature value

- visibility: The observed visibility value

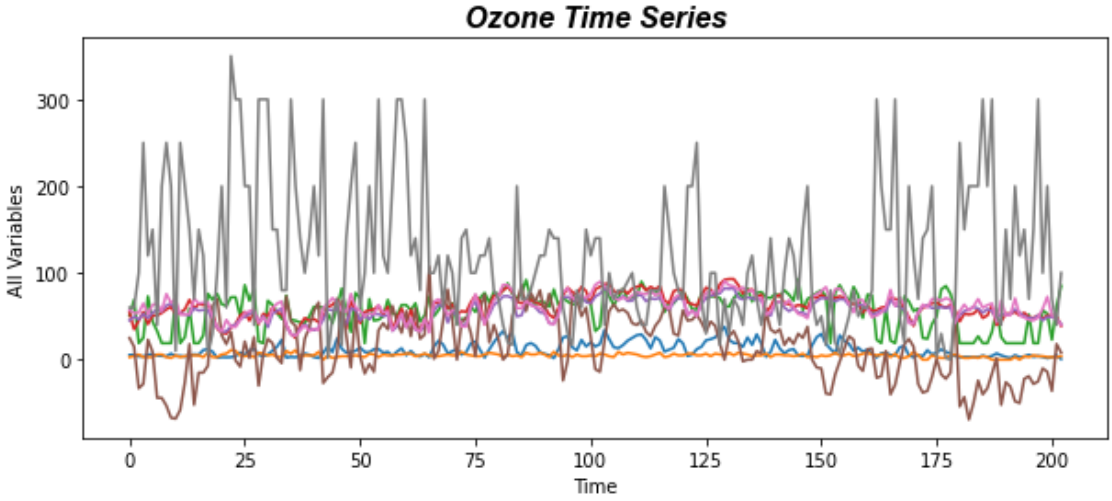

Figure 17: The Ozone Dataset

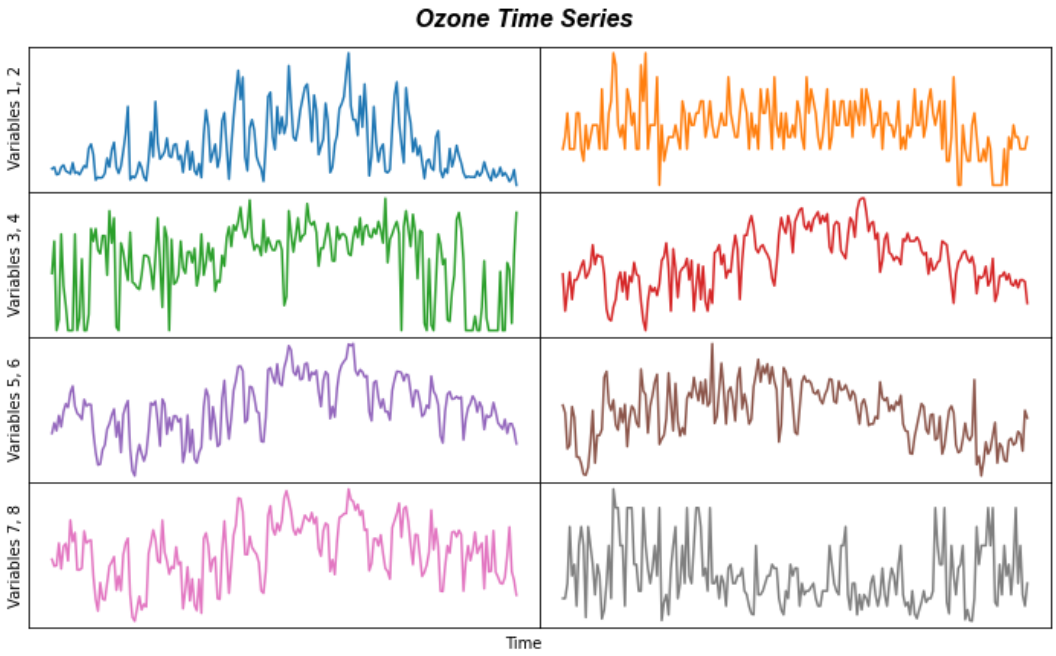

Figure 18: Individual Time Series of the Ozone Dataset

Night Visitors Dataset

The dataset contains 56 observations of a vector-valued time series representing the number of overnight tourism visitors in various regions of Australia from 2006/07 to 2014/15. It contains the following 8 variables.

- **Sydney:** The number of passengers that visited Sydney
- **NSW:** The number of passengers that visited the New South Wales state
- **Melbourne:** The number of passengers that visited Melbourne
- **VIC:** The number of passengers that visited the Victoria State
- **BrisbaneGC:** The number of passengers that visited Gold Coast south of Brisbane
- **QLD:** The number of passengers that visited the state of Queensland
- **Capitals:** The number of passengers that visited a capital city
- **Other:** The number of passengers that visited other cities

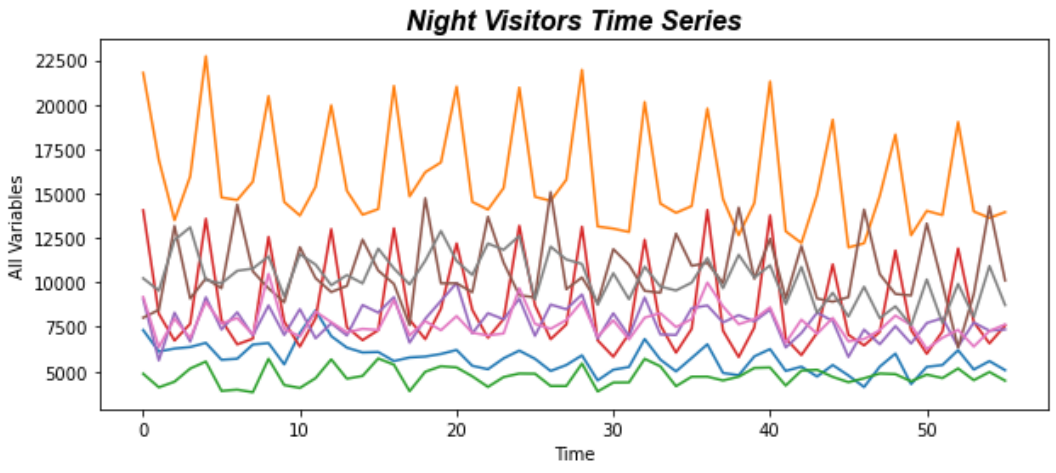

Figure 19: The Night Visitors dataset

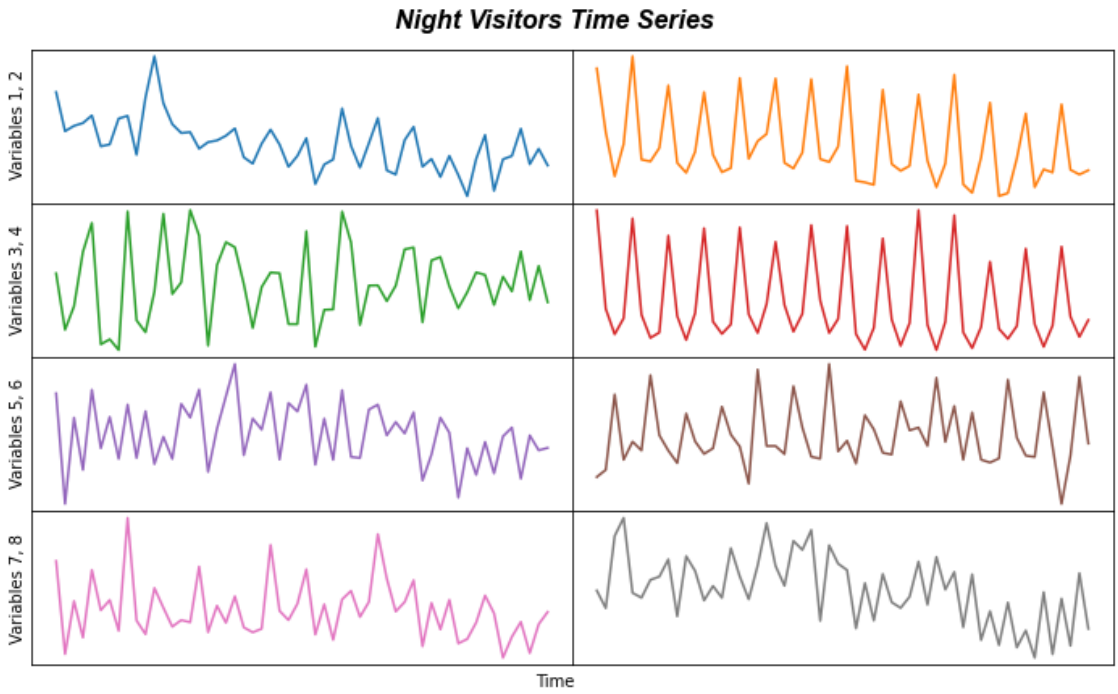

Figure 20: Individual Time Series of the Night Visitors Dataset

NASDAQ Dataset

The NASDAQ dataset contains data for 82 of the largest domestic and international non-financial securities based on market capitalization listed on the stock market [7]. The dataset contains 40560 observations for each entity. The following Figures visualize the first 100 observations of the dataset.

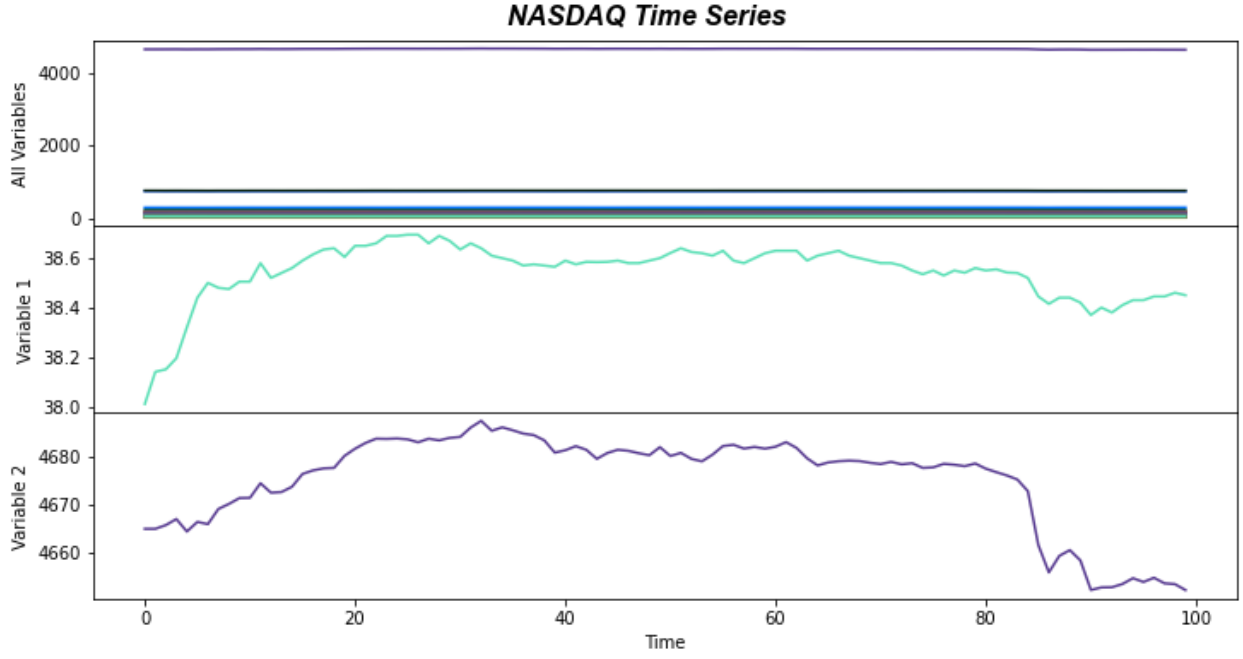

Figure 21: The NASDAQ dataset - Part 1

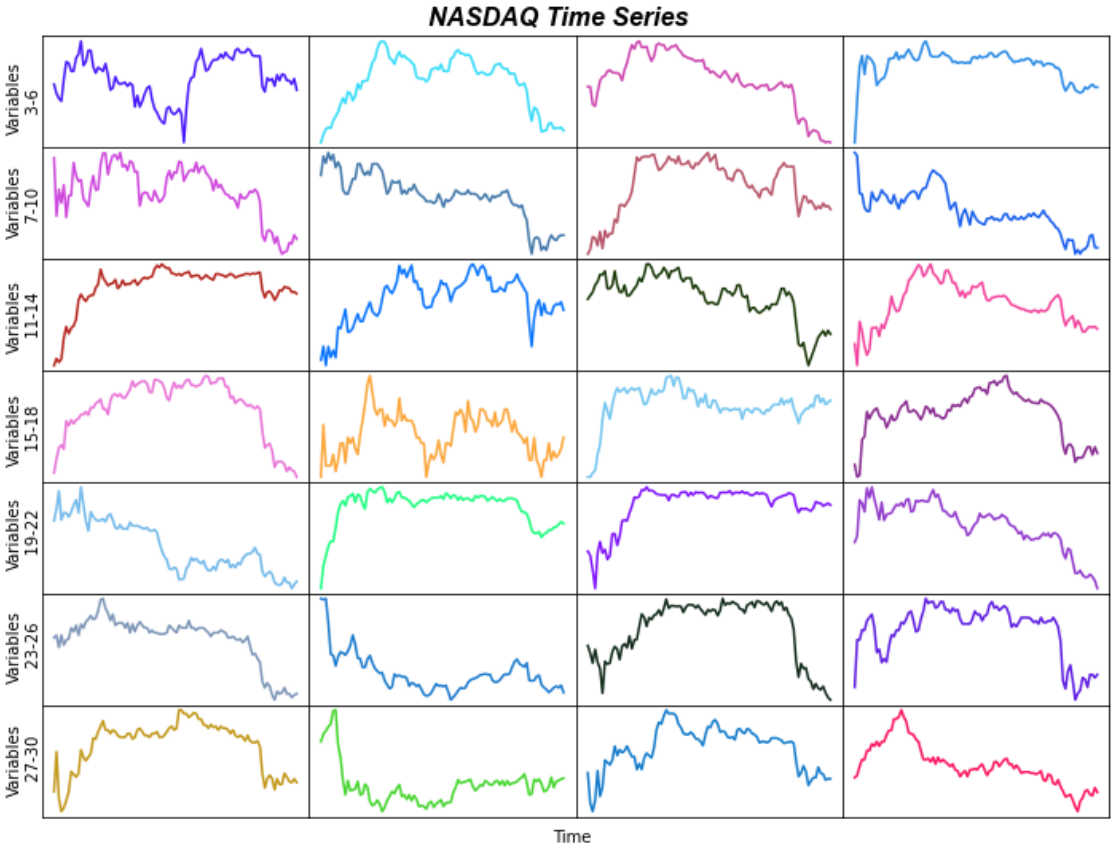

Figure 22: The NASDAQ dataset - Part 2

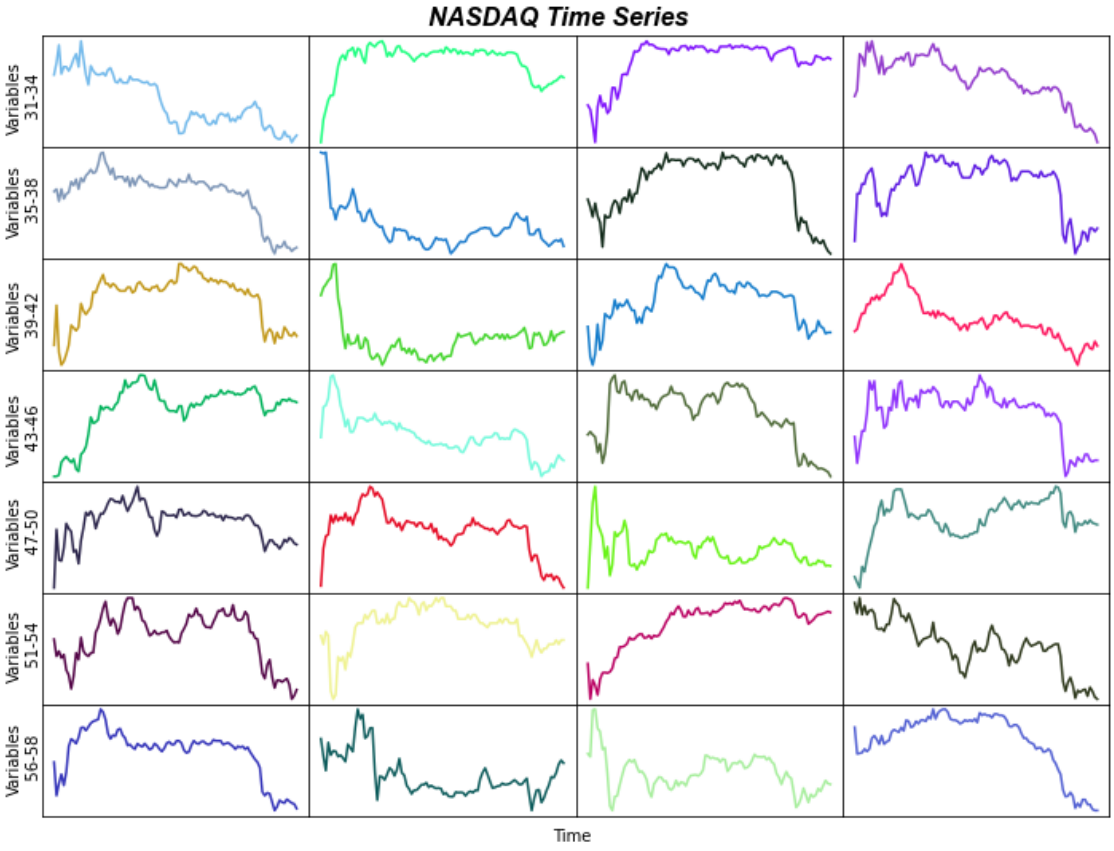

Figure 23: The NASDAQ dataset - Part 3

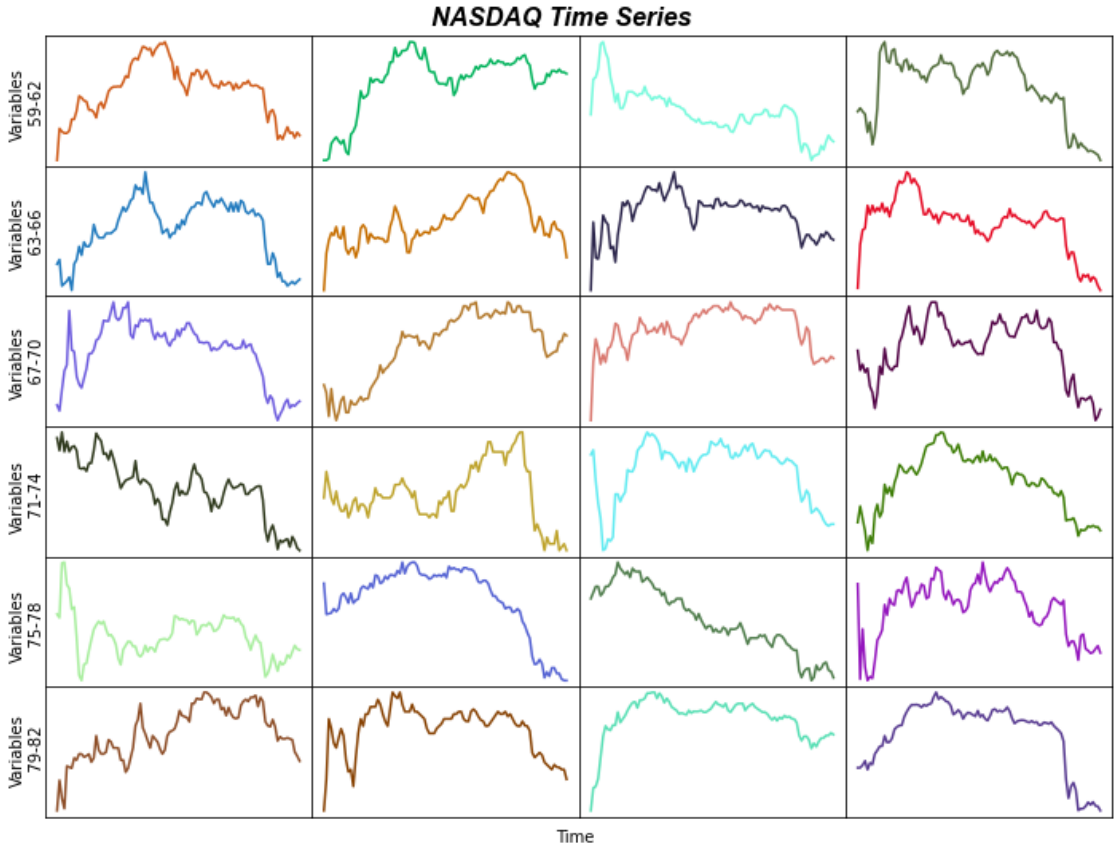

Figure 24: The NASDAQ dataset - Part 4

Yahoo Dataset

The Yahoo dataset contains 2469 observations of the Yahoo daily stock providing information over the following variables. The first 100 observations are presented in the Figures below.

- **VIX.Open:** The opening value of the stock on the Volatile Index
- **VIX.High:** The highest value of the stock on the Volatile Index
- **VIX.Low:** The lowest value of the stock on the Volatile Index
- **VIX.Close:** The closing value of the stock on the Volatile Index

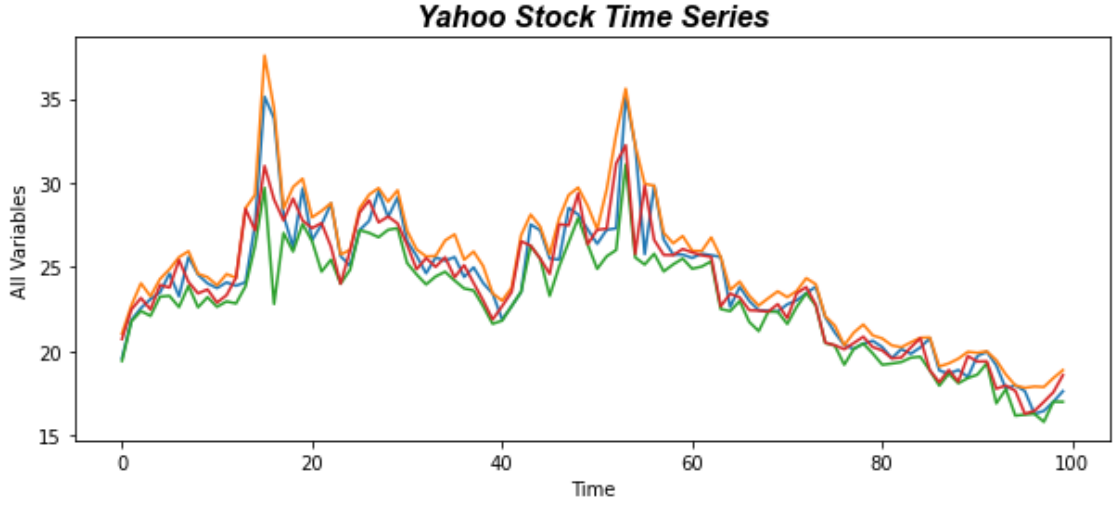

Figure 25: The Yahoo Stock dataset

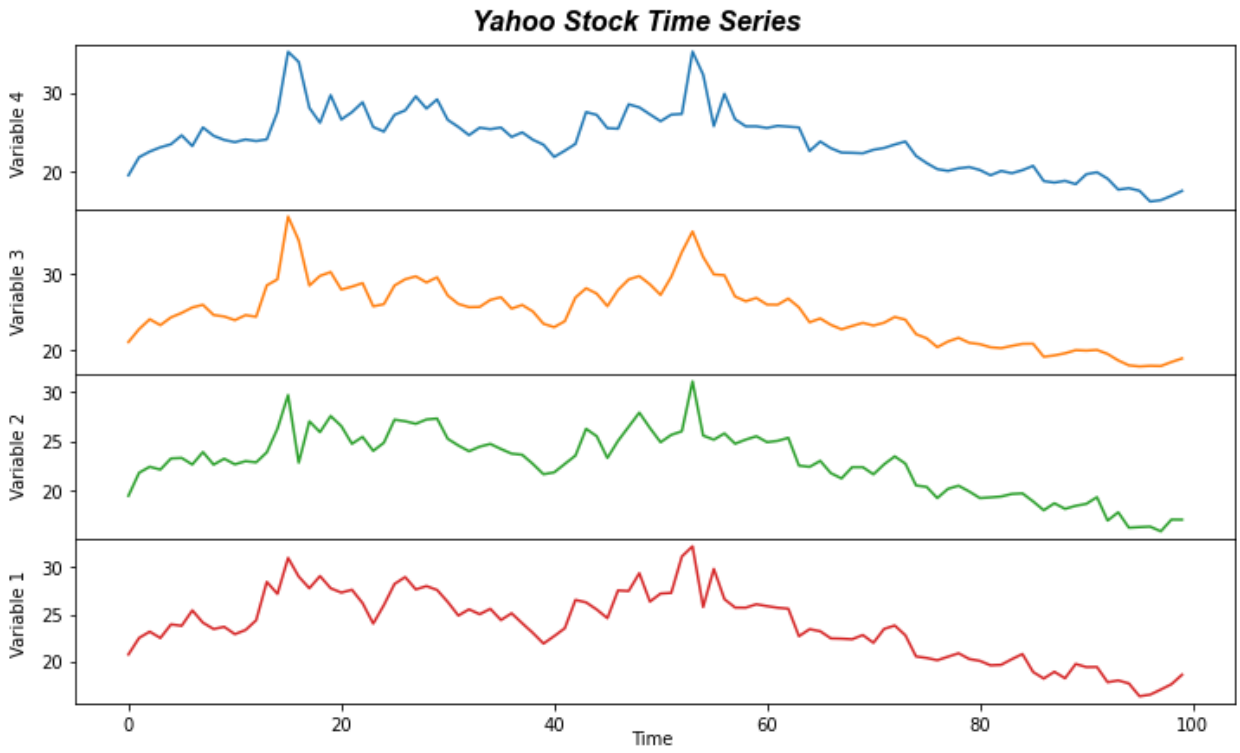

Figure 26: The Yahoo Stock Individual Time Series

Stationarity

For each dataset we ensure stationarity by utilizing the **Augmented Dickey-Fuller** test on a significance level of 5% . Note that in the case of the Block Hankel Tensor Autoregression variations we are interested in the stationarity of the Hankelized form of the data. The Augmented Dickey-Fuller test shows that there is not enough statistical evidence to ensure stationarity for the 7 public datasets in their original forms. Thus, we conduct differencing to obtain a stationary form of each time series. The differencing parameter d will be included in the grid search that will be conducted for each algorithm.

4.2 Compared Algorithms

To provide an accurate evaluation, we compare Block Hankel Tensor Autoregression with scalar and matrix coefficients with other important and widely used models like **Vector Autoregression** with scalar and matrix coefficients and **Facebook's Prophet**. Note that the last three algorithms do not receive as input the Block Hankel Tensor form of the data. For each algorithm, a grid search is conducted to determine the optimal set of hyperparameters.

4.3 Experiments

In the following experiments, the algorithms are evaluated over several important factors. Firstly, we focus on time series with very few observations and experiment with short and long forecasting horizons. Additionally, we evaluate the algorithms over different volumes of training data to determine the effectiveness on longer time series as well. Finally, a

runtime comparison between the algorithms will be conducted since time efficiency is a usual and important restriction in real world applications.

The evaluation procedure in the following experiments is conducted as follows. First, each dataset is split in three time-consecutive sets for the purposes of training, validating and testing each algorithm. The training and validation sets are utilized to estimate the optimal set of hyperparameters and the autoregression coefficients. Finally, we measure the forecasting error on the test set using the chosen set of hyperparameters and the estimated coefficients.

The metric that will be used, to measure the forecasting error in the following experiments, is the *Normalized Root Mean Squared Error*. While the *RMSE* is a reliable metric, it lacks the ability to facilitate comparison between models of different scales. This is what NRMSE achieves by applying a normalization technique. While there are various ways to normalize RMSE, in the context of this thesis NRMSE is defined as:

$$NRMSE = \frac{RMSE}{\overline{\mathcal{X}}}$$

where $RMSE = \sqrt{\frac{\|\mathcal{X} - \widehat{\mathcal{X}}\|_F^2}{\prod_{n=1}^{N} I_n}}$ and $\overline{\mathcal{X}} = \frac{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\dots\sum_{i_N=1}^{I_N}|x_{i_1 i_2 \dots i_N}|}{\prod_{n=1}^{N} I_n}$

4.3.1 Parameter & Convergence Analysis

In this section, we study the sensitivity of Block Hankel Tensor Autoregression towards the MDT order r and the Tucker Decomposition Ranks R_1, R_2. Furthermore, we study the convergence of the algorithms in terms of the quantity:

$$\frac{\sum_{n=1}^{N}\|\boldsymbol{U}^{(n)} - old\_\boldsymbol{U}^{(n)}\|_F^2}{\sum_{n=1}^{N}\|\boldsymbol{U}^{(n)}\|_F^2}$$

We start by conducting a grid search over the aforementioned parameters on the synthetic dataset. More specifically, we generate 3 samples of 21, 25 and 30 observations. In each case the first 20 observations are used as the training set and the remaining observations serve as the validation set. The results of the grid search process are presented in the tables 2, 3, 4, 5, 6 and 7. Each table contains the sets of parameters that resulted in the lowest NRMSE value for each validation. The row in bold contains the set of parameters that produces the lowest NRMSE on the validation set. As we can see in each validation, the obtained parameters and the NRMSE values vary. This shows a high-sensitivity towards the training data and their volume since even an addition of 4 or 9 observations affects the results to an extent. On the other hand, in each validation we can see that the obtained NRMSEs are close-valued and thus there is no need to carefully tune the hyper-parameter set.

Table 2: Sample of Grid Search Results for BHT_AR_SC on 21 observations

Block Hankel Tensor AR - Scalar Coefficients			
r	R_1	R_2	NRMSE
8	**3**	**4**	**0.013154**
10	2	3	0.013682
7	3	3	0.013999
7	2	2	0.014009
9	3	4	0.014126

Table 3: Sample of Grid Search Results for BHT_AR_MC on 21 observations

Block Hankel Tensor AR - Matrix Coefficients			
r	R_1	R_2	NRMSE
8	**2**	**4**	**0.003173**
8	3	8	0.004098
7	2	7	0.005350
7	3	8	0.005651

Table 4: Sample of Grid Search Results for BHT_AR_SC on 25 observations

Block Hankel Tensor AR - Scalar Coefficients			
r	R_1	R_2	NRMSE
9	**3**	**5**	**0.021681**
9	3	8	0.022208
2	3	2	0.022208
9	3	9	0.022097
9	3	7	0.022343

Table 5: Sample of Grid Search Results for BHT_AR_MC on 25 observations

Block Hankel Tensor AR - Matrix Coefficients			
r	R_1	R_2	NRMSE
7	**3**	**5**	**0.020156**
2	3	2	0.020586
10	2	8	0.021617
10	2	10	0.021785
1	3	6	0.021898

Table 6: Sample of Grid Search Results for BHT_AR_SC on 30 observations

Block Hankel Tensor AR - Scalar Coefficients			
r	R_1	R_2	**NRMSE**
3	**2**	**3**	**0.020653**
3	2	2	0.020884
10	3	2	0.022179
10	2	2	0.022158
10	3	3	0.022363

Table 7: Sample of Grid Search Results for BHT_AR_MC on 30 observations

Block Hankel Tensor AR - Matrix Coefficients			
r	R_1	R_2	**NRMSE**
5	**2**	**5**	**0.019599**
5	3	5	0.019987
9	2	8	0.020337
9	3	8	0.020675
8	2	7	0.020756

To analyse the convergence of the algorithms, we generate 9 samples. The first 5 samples contain training sets of 20 observations and validation sets of 1, 3, 5, 7 and 9 observations respectively. The remaining 4 samples contain training sets of 150 observations and validation sets of 1, 5, 10 and 15 observations respectively. As we can see in the Figures below (27, 28, 29, 30, 31, 32, 33, 34, 35), both algorithms converge really fast. In only a few iterations, the convergence quantity reaches and maintains a small value indicating that the changes of the factor matrices U are very small for the remaining iterations. It should be noted that in all cases the convergence value of the BHT_AR_MC stabilizes at a value close to zero and as a result the changes of the factor matrices U are imperceptible. Furthermore, we can see that in most of the cases, the NRMSE of the BHT_AR_MC decreases smoothly and stabilizes near a very small value. On the other hand the NRMSE of BHT_AR_SC shows a relatively less stable behaviour.

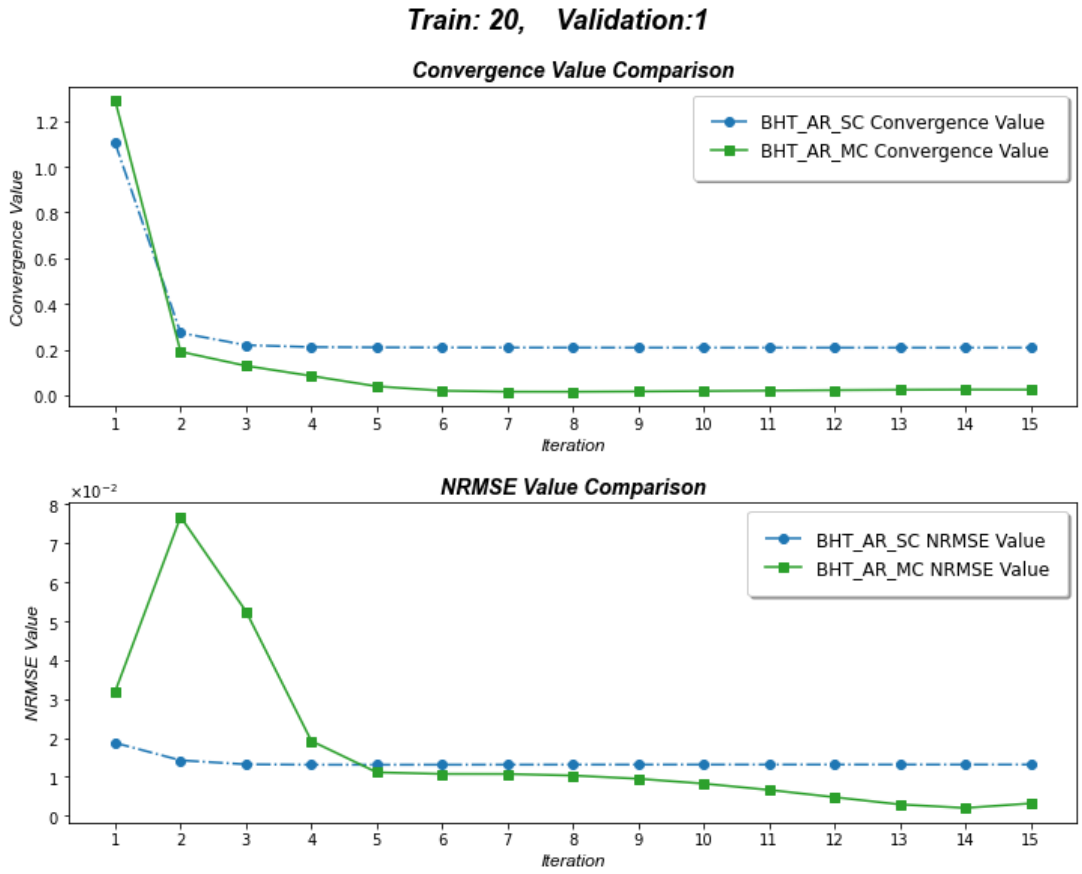

Figure 27: Convergence & NRMSE Comparison on a 1-point Validation set

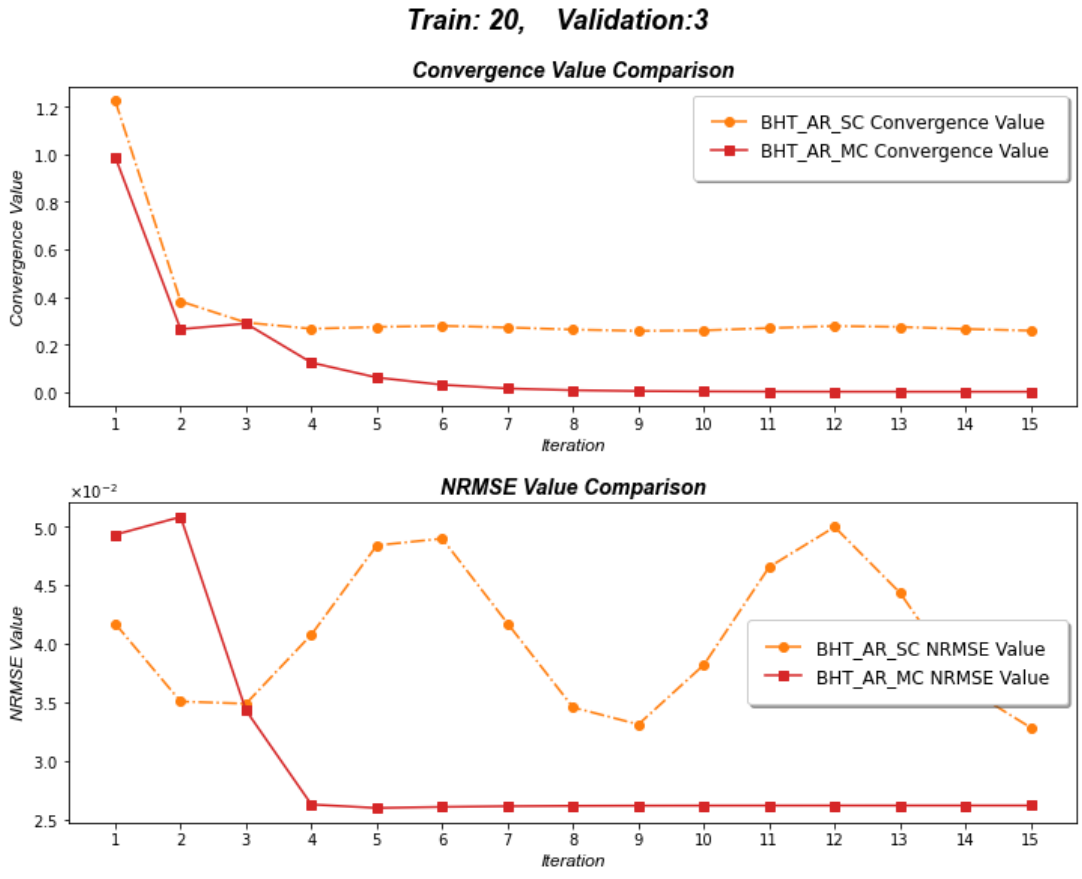

Figure 28: Convergence & NRMSE Comparison on a 3-point Validation set

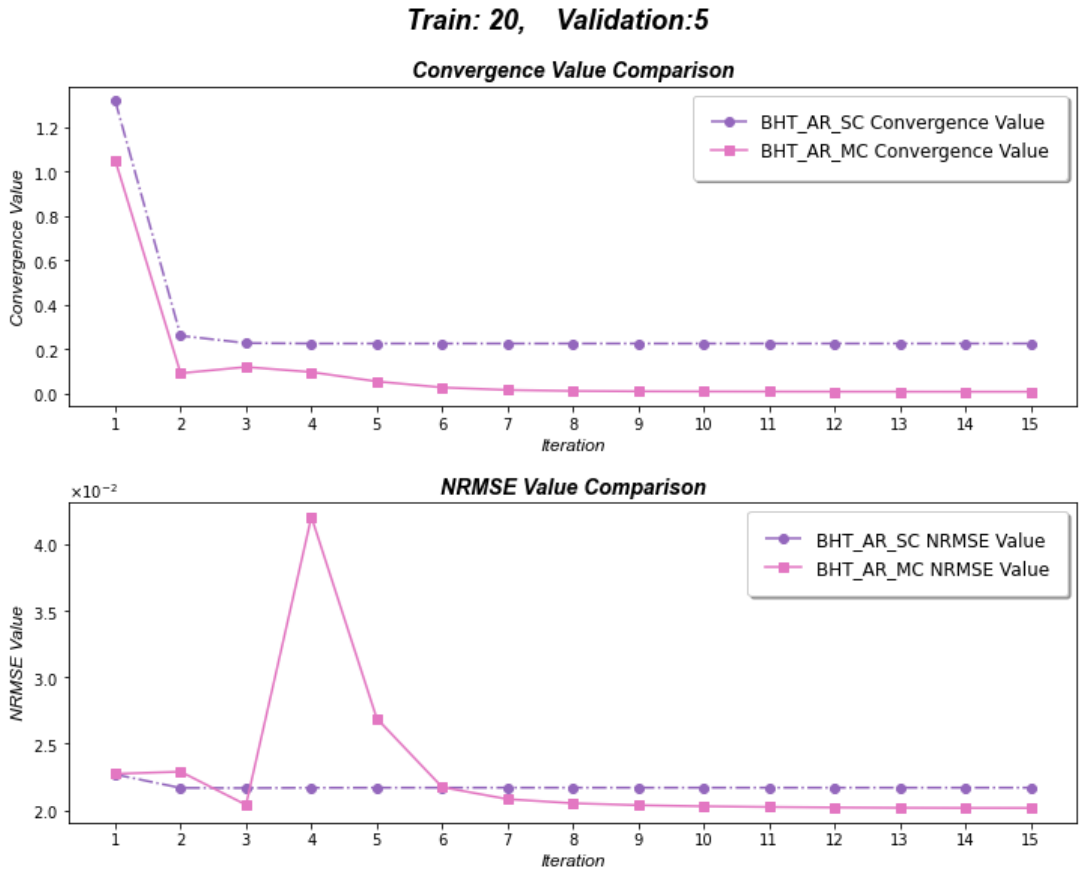

Figure 29: Convergence & NRMSE Comparison on a 5-point Validation set

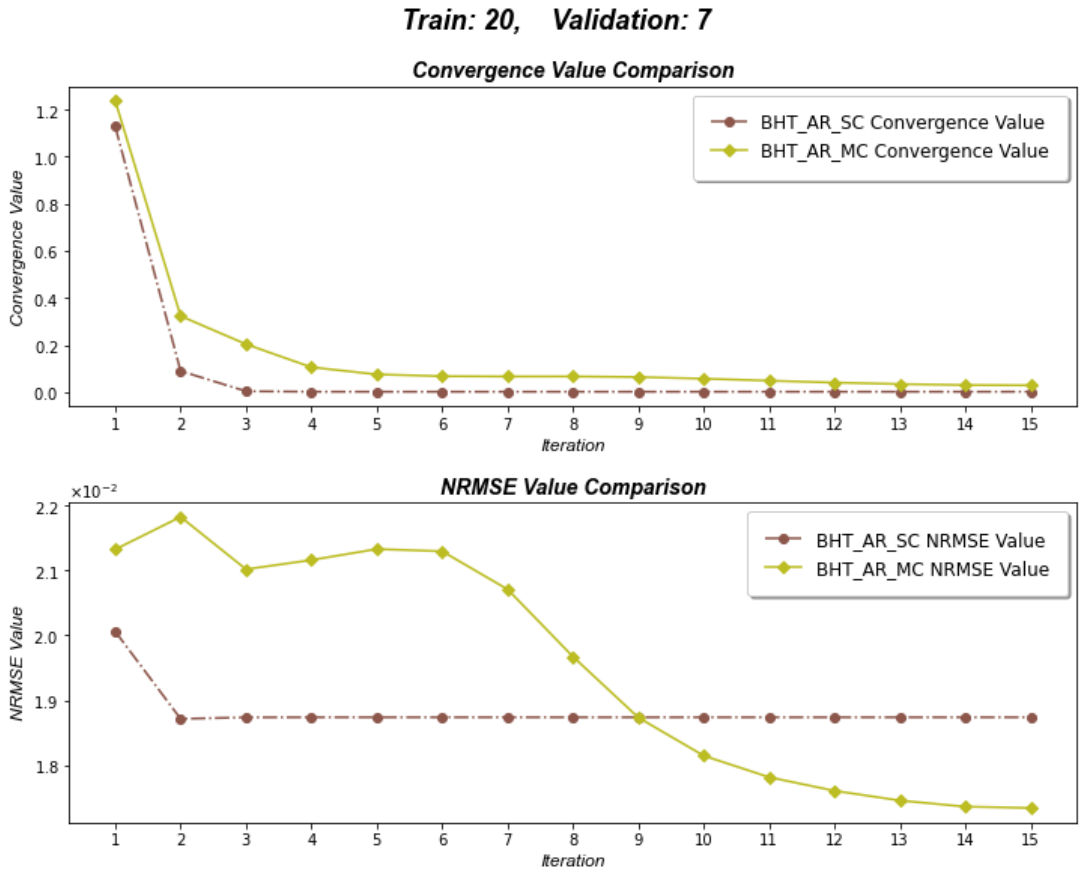

Figure 30: Convergence & NRMSE Comparison on a 7-point Validation set

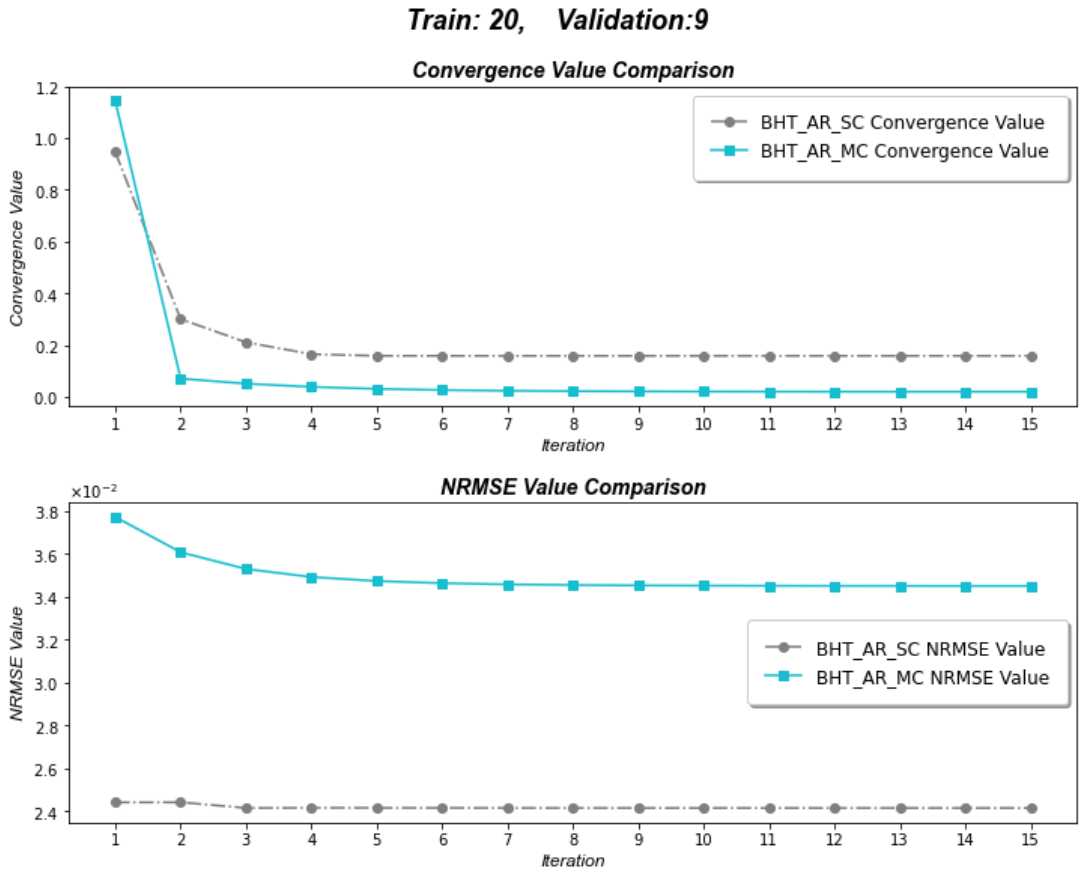

Figure 31: Convergence & NRMSE Comparison on a 9-point Validation set

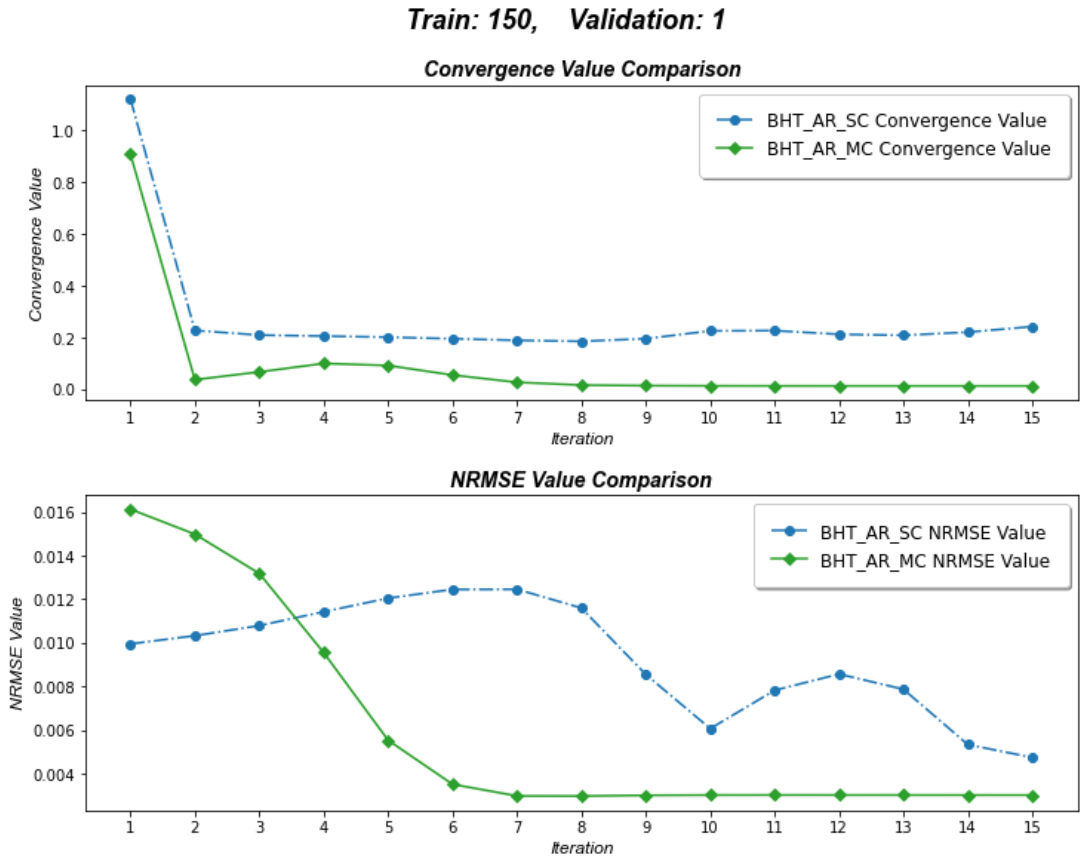

Figure 32: Convergence & NRMSE Comparison on an 1-point Validation set

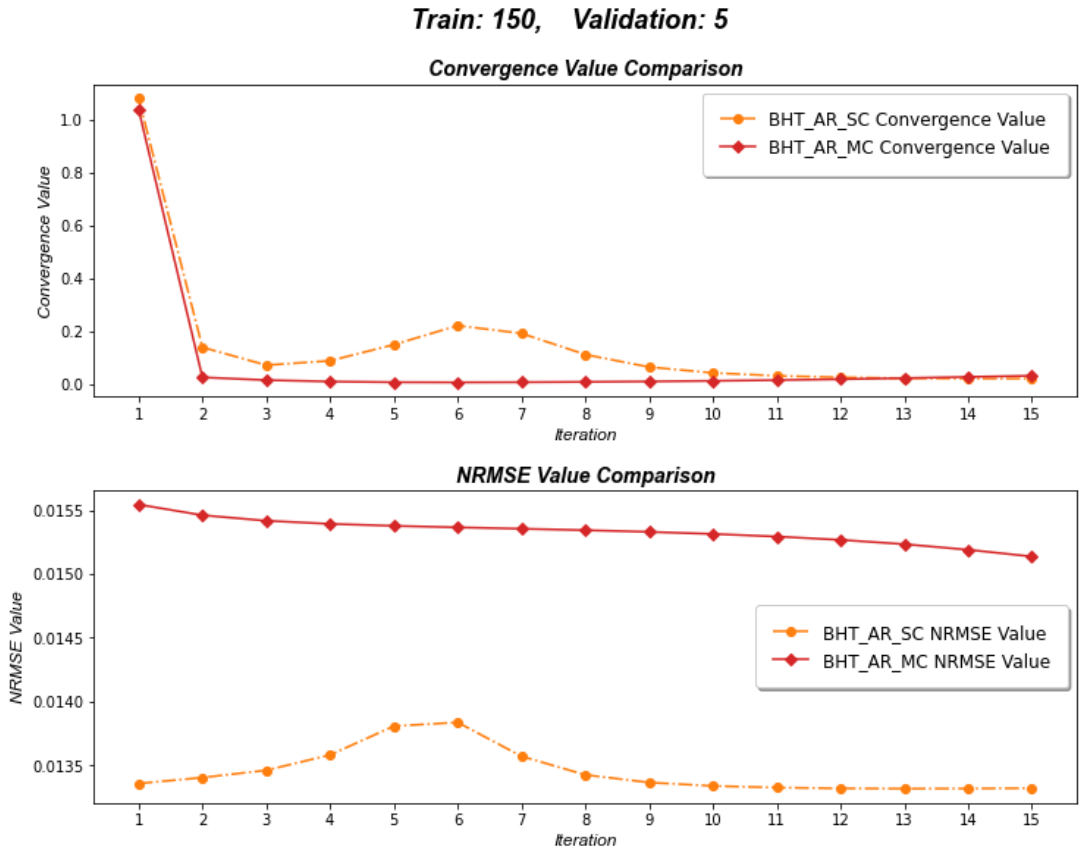

Figure 33: Convergence & NRMSE Comparison on a 5-point Validation set

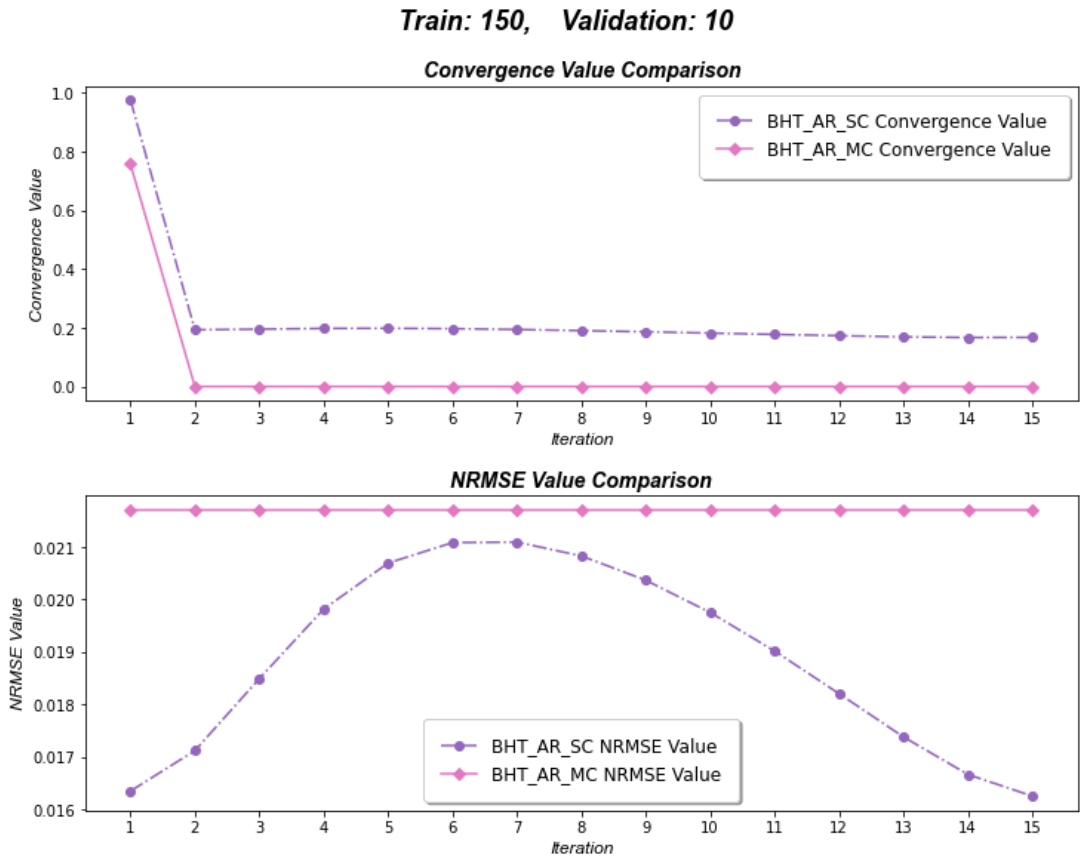

Figure 34: Convergence & NRMSE Comparison on a 10-point Validation set

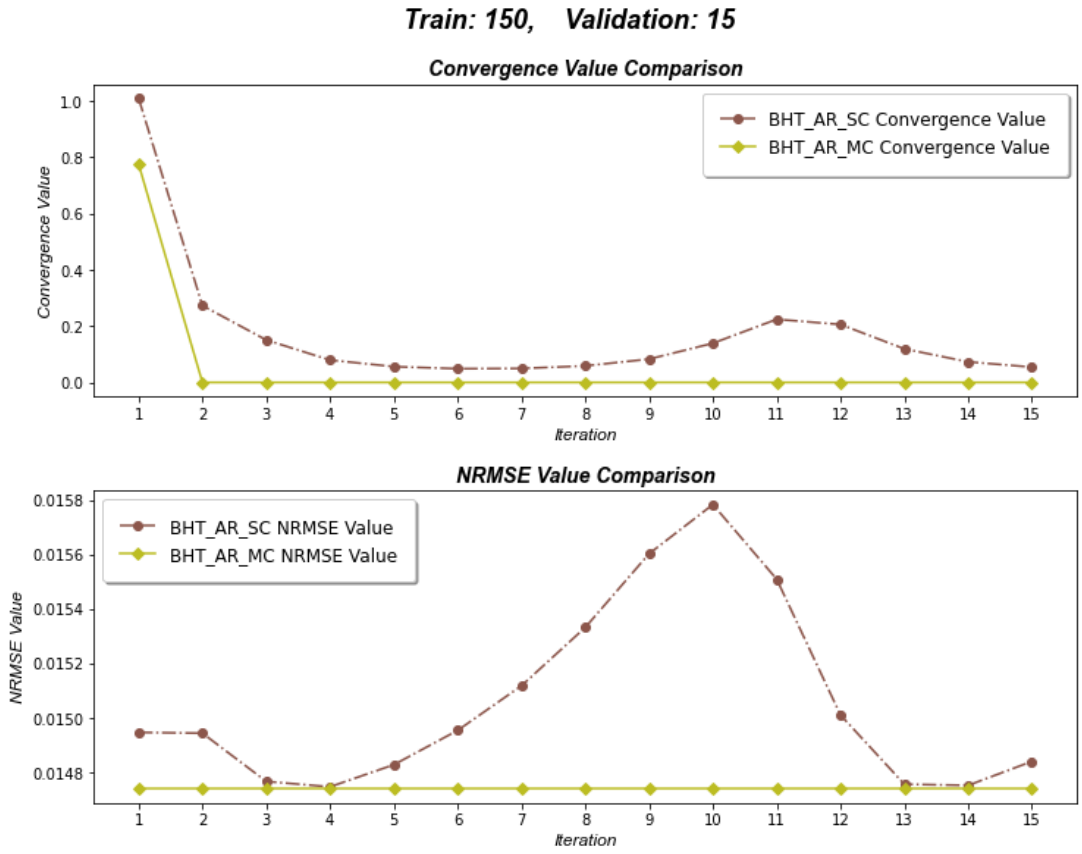

Figure 35: Convergence & NRMSE Comparison on a 15-point Validation set

4.3.2 Experiments on Forecasting Horizon

In this section we evaluate the algorithms over their ability to maintain their performance while the forecasting horizon increases. Following the same procedure as in the previous experiment we generate 10 samples of synthetic data. Each training set contains 20 observations while the validation and test sets have equal volumes varying from 1 to 10 observations. In addition, we generate 4 training samples with 150 observations and validation/test sets of 1, 5, 10 and 15 observations. In each case, the algorithms are trained using the respective training set. We select the set of hyperparameters that results in the lowest NRMSE on the validation set while monitoring the convergence and total runtime of the training process. Finally, we compute the NRMSE on the test set in order to obtain an unbiased assessment of the performance of the algorithms. The results are shown in the Figures (36, 37) and tables (8, 9) below. The results of autoregression with scalar coefficients are truncated in the Figures.

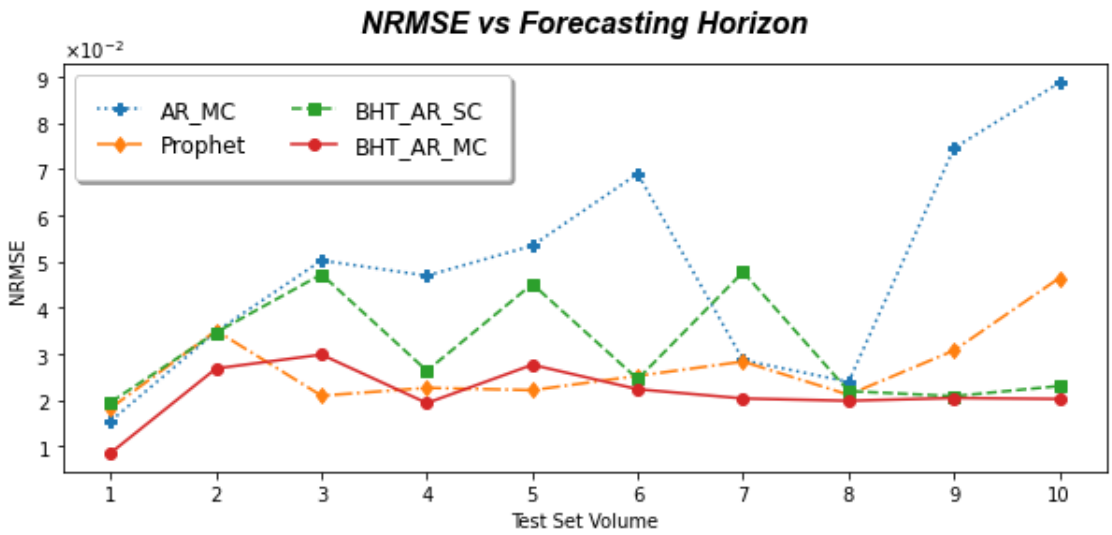

Figure 36: NRMSE vs forecasting horizon on a 20-point training set

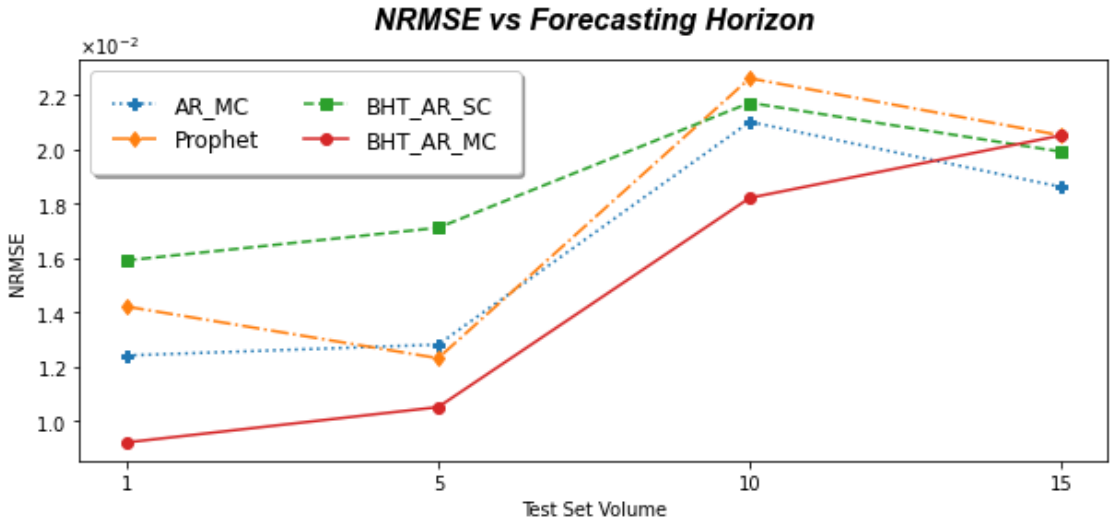

Figure 37: NRMSE vs forecasting horizon on a 150-point training set

Table 8: Experimental Forecasting Results on Synthetic Data

Short Time Series Forecasting NRMSE					
Partitioning	**AR_SC**	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
20 - 1 - 1	0.1304	0.0154	0.0182	0.0195	**0.0085**
20 - 2 - 2	0.1342	0.0348	0.0350	0.0346	**0.0268**
20 - 3 - 3	0.1393	0.0502	**0.0209**	0.0472	0.0298
20 - 4 - 4	0.1379	0.0469	0.0226	0.0263	**0.0193**
20 - 5 - 5	0.1397	0.0534	**0.0221**	0.0451	0.0276
20 - 6 - 6	0.1480	0.0690	0.0252	0.0245	**0.0223**
20 - 7 - 7	0.1284	0.0286	0.0283	0.0477	**0.0203**
20 - 8 - 8	0.1281	0.0239	0.0211	0.0219	**0.0198**
20 - 9 - 9	0.1478	0.0745	0.0307	0.0208	0.**0204**
20 - 10 - 10	0.1498	0.0887	0.0464	0.0230	**0.0201**
Long Time Series Forecasting NRMSE					
150 - 1 - 1	0.4390	0.0124	0.0142	0.0159	**0.0092**
150 - 5 - 5	0.4348	0.0128	0.0123	0.0171	**0.0105**
150 - 10 - 10	0.5346	0.0210	0.0226	0.0217	**0.0182**
150 - 15 - 15	0.4370	**0.0186**	0.0205	0.0199	0.0205

Table 9: Synthetic Data - Training Runtime in seconds

Training Runtime					
Partitioning	**AR_SC**	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
20 - 1 - 1	0.0006	0.0007	6.0825	0.1110	0.0581
20 - 2 - 2	0.0003	0.0003	6.2098	0.0559	0.0766
20 - 3 - 3	0.0004	0.0004	6.3582	0.0387	0.0659
20 - 4 - 4	0.0007	0.0002	6.2495	0.0430	0.0732
20 - 5 - 5	0.0006	0.0002	7.0670	0.0686	0.1354
20 - 6 - 6	0.0008	0.0001	6.6102	0.0610	0.0803
20 - 7 - 7	0.0007	0.00007	6.3104	0.0406	0.0665
20 - 8 - 8	0.0009	0.0002	6.5736	0.0430	0.0667
20 - 9 - 9	0.0006	0.0003	6.7177	0.0702	0.0740
20 - 10 - 10	0.0007	0.0004	7.3967	0.1087	0.2341
150 - 1 - 1	0.0042	0.0002	6.4954	0.7224	0.8137
150 - 5 - 5	0.0042	0.0003	6.4101	0.3994	0.3810
150 - 10 - 10	0.0040	0.0002	7.7392	0.3333	0.4016
150 - 15 - 15	0.0042	0.0002	6.5440	0.4171	0.3469

As we can see, in most cases, Block Hankel Tensor Autoregression with matrix coefficients outperforms the rest of the algorithms. Furthermore, in the short time series framework it manages to maintain a stable performance regardless the forecasting horizon. Finally, while the training time for the Block Hankel Tensor Autoregression algorithms is higher compared with the traditional autoregression algorithms, it is lower than the training runtime of Prophet. Thus, the proposed algorithms manage to achieve great results while maintaining low runtime levels which makes them an efficient and competitive choice in the context of short time series forecasting.

4.3.3 Short-term Forecasting Experiments

In this section we focus on a short forecasting horizon of 1 observation and study the performance of the algorithms on short time series. In addition, we study the effects of data reduction during the training process. For this purpose we use the Stack Loss, US macroeconomic, El-Nino, Ozone and Night Visitors datasets. We start by training and evaluating each algorithm on the datasets using every observation. For each dataset with size N we use $N-2$ observations as the training set and the last 2 observations are the validation and test sets. Since the Stack Loss dataset contains 21 observations it is excluded from the rest of the experiment. For the remaining 4 datasets each algorithm is re-evaluated on 75%, 50% and 25% of the original volume of each dataset. The results are grouped by dataset and summarized in the following tables (10, 11, 12, 13 and 14) and Figures (38, 39, 40 and 41)

Table 10: Experimental Short Forecasting Results - Stack Loss Dataset

Stack Loss - Short Forecasting NRMSE				
AR_SC	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
0.1585	0.1515	0.2110	0.1237	**0.0867**

Table 11: Experimental Short Forecasting Results - US Macroeconomic Dataset

US Macroeconomic - Short Forecasting NRMSE					
Dataset Size	**AR_SC**	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
100%	0.0150	0.0066	0.0179	0.0078	**0.0057**
75%	0.0056	0.0045	0.0095	0.0046	**0.0025**
50%	0.0087	0.0085	0.0086	0.0052	**0.0032**
25%	0.0053	0.0079	0.0070	0.0080	**0.0037**

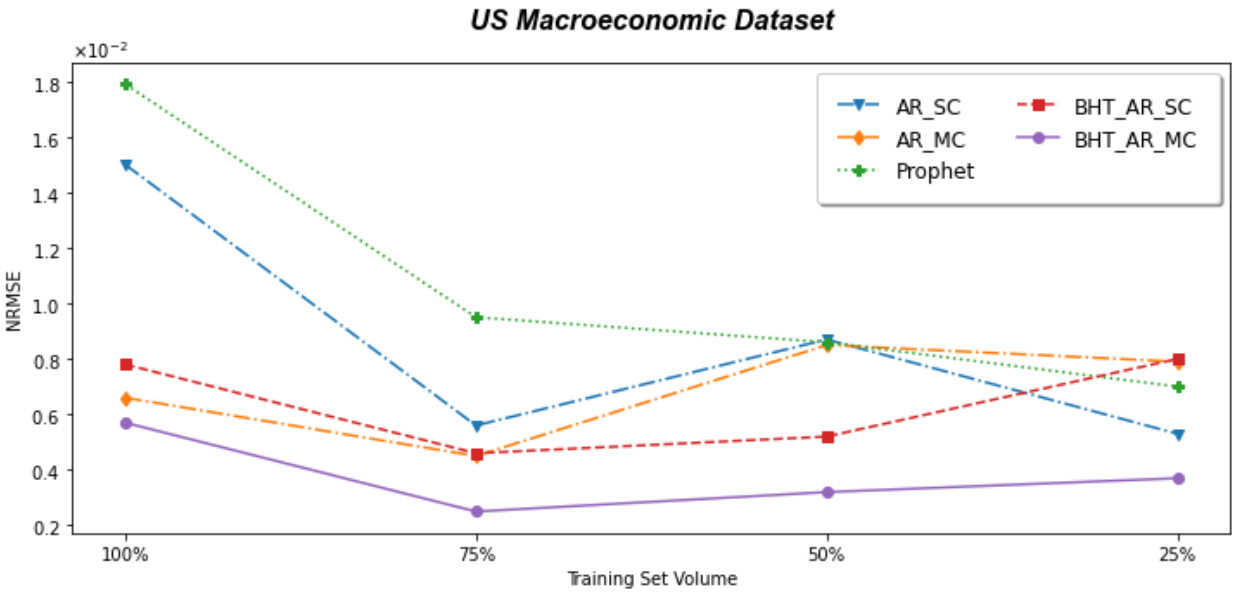

Figure 38: NRMSE vs Training Volume on the US Macroeconomic Dataset

Table 12: Experimental Short Forecasting Results - El-Nino Dataset

El-Nino - Short Forecasting NRMSE					
Dataset Size	**AR_SC**	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
100%	0.0259	0.0334	0.0218	**0.0160**	0.0164
75%	0.0236	0.0326	0.1515	0.0245	**0.0103**
50%	0.0196	0.0332	0.0454	0.0104	**0.0093**
25%	0.0198	0.0392	0.0131	0.0114	**0.0107**

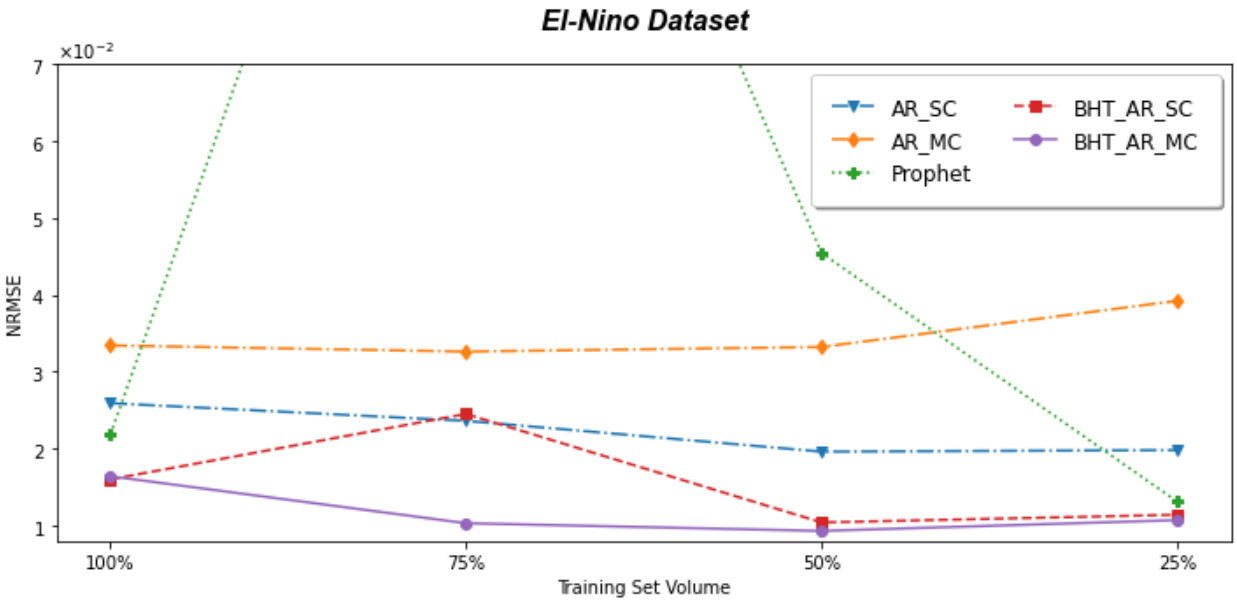

Figure 39: NRMSE vs Training Volume on the El-Nino Dataset

Table 13: Experimental Short Forecasting Results - Ozone Dataset

Ozone - Short Forecasting NRMSE					
Dataset Size	**AR_SC**	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
100%	0.6900	0.2908	0.3077	0.3052	**0.1707**
75%	0.0734	0.1062	0.4138	0.0587	**0.0416**
50%	0.2089	0.2354	0.2734	**0.1038**	0.1127
25%	0.4789	0.4007	0.3045	0.2941	**0.1947**

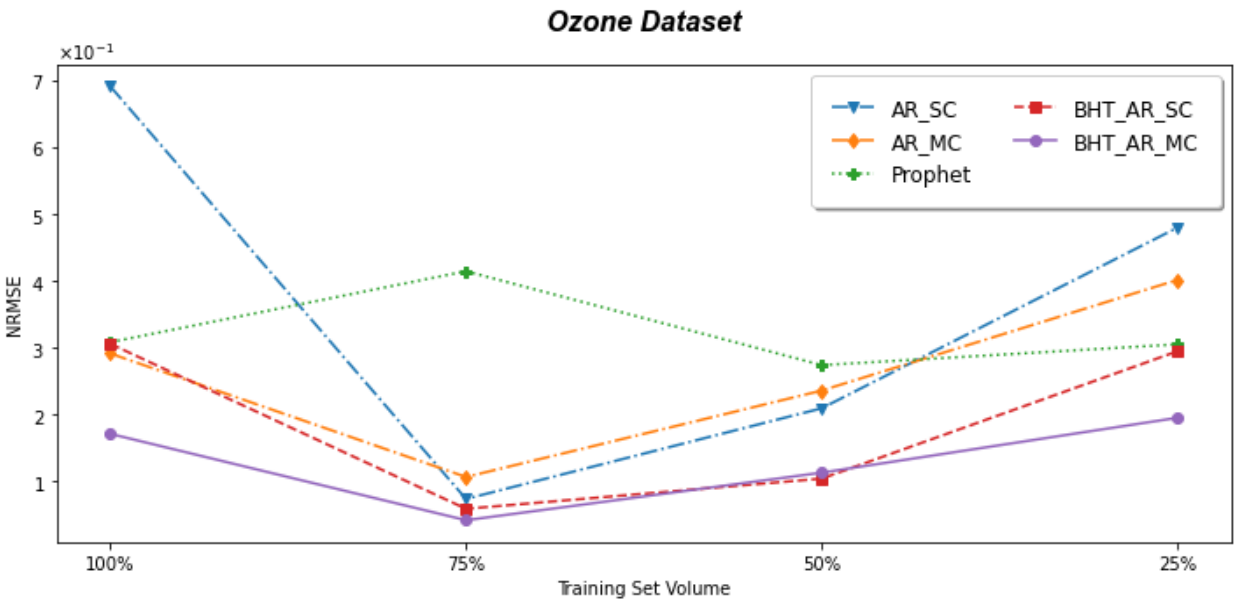

Figure 40: NRMSE vs Training Volume on the Ozone Dataset

Table 14: Experimental Short Forecasting Results - Night Visitors Dataset

Night Visitors - Short Forecasting NRMSE					
Dataset Size	**AR_SC**	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
100%	0.0680	0.1310	0.1865	**0.0453**	0.0826
75%	0.0880	0.0813	0.2802	0.0950	**0.0705**
50%	0.0830	0.0720	0.2028	0.1074	**0.0546**
25%	0.2657	0.4195	0.2226	0.1192	**0.0718**

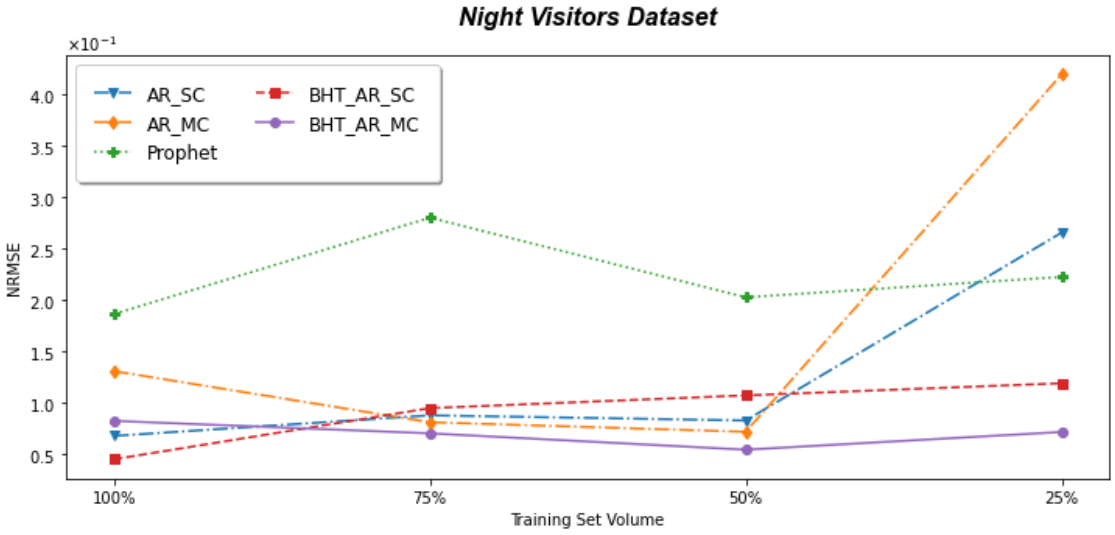

Figure 41: NRMSE vs Training Volume on the Night Visitors Dataset

As we can see in the Figures, both BHT_AR_SC and BHT_AR_MC show stability with regards to NRMSE as the volume of training data is reduced. BHT_AR_MC manages to maintain a more stable performance, with smaller increases in forecasting error as the

volume of the training set is reduced. Furthermore, it achieves the lowest NRMSE value in most of the experiments.

4.3.4 Long-term Forecasting experiments

Finally, we evaluate BHT_AR_SC and BHT_AR_MC over relatively longer forecasting horizons. We experiment on Yahoo Stocks and NASDAQ datasets by taking a sample of 50 observations as training set and 4 validation/test sets with varying volumes of 1, 5, 10 and 15 observations. Following the same procedure as in the previous experiments, we estimate the optimal set of hyperparameters on the validation sets and then measure the forecasting error on the test sets.

As we can see in table 15 and Figure 42 as the forecasting horizon increases all algorithms' forecasts result in higher NRMSEs on the Yahoo Stock dataset. Both BHT_AR_SC and BHT_AR_MC perform relatively well. BHT_AR_SC shows a more stable behaviour as the forecasting horizon increases. On the other hand, on the NASDAQ dataset (table 16 and Figure 43) BHT_AR_MC performs the best overall with low NRMSE values and a stable behaviour as the forecasting horizon increases.

Table 15: Experimental Forecasting Results - Yahoo Stock Dataset

Yahoo Stocks - Forecasting NRMSE					
Forecasting Horizon	**AR_SC**	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
1	0.0510	0.0314	0.0432	0.0551	**0.0276**
5	0.1223	0.1244	0.0986	0.1286	**0.0878**
10	0.1383	0.1470	**0.1254**	0.1320	0.1530
15	0.1626	0.1818	0.1345	**0.1294**	0.1415

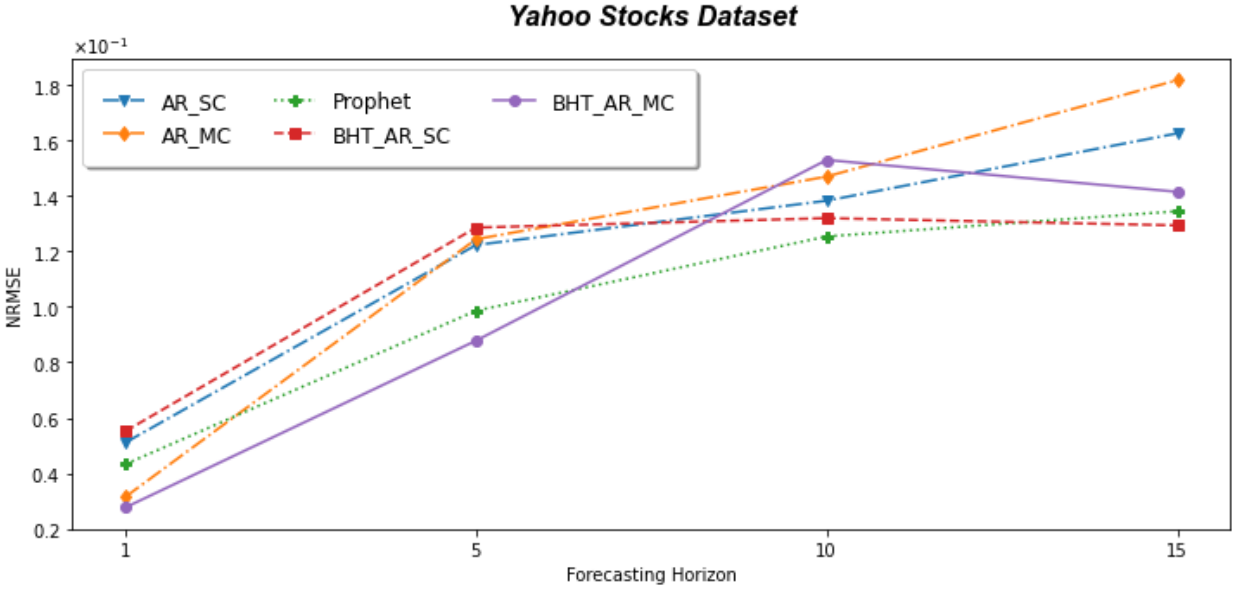

Figure 42: NRMSE vs Forecasting Horizon on the Yahoo Stocks Dataset

Table 16: Experimental Forecasting Results - NASDAQ Dataset

NASDAQ - Forecasting NRMSE					
Forecasting Horizon	**AR_SC**	**AR_MC**	**Prophet**	**BHT_AR_SC**	**BHT_AR_MC**
1	0.2046	0.2191	0.1998	0.2141	**0.1938**
5	0.2205	0.2373	**0.2014**	0.2143	0.2046
10	2.3643	1.6568	0.2112	0.2321	**0.2022**
15	3.8674	2.1464	0.2353	0.2237	**0.2031**

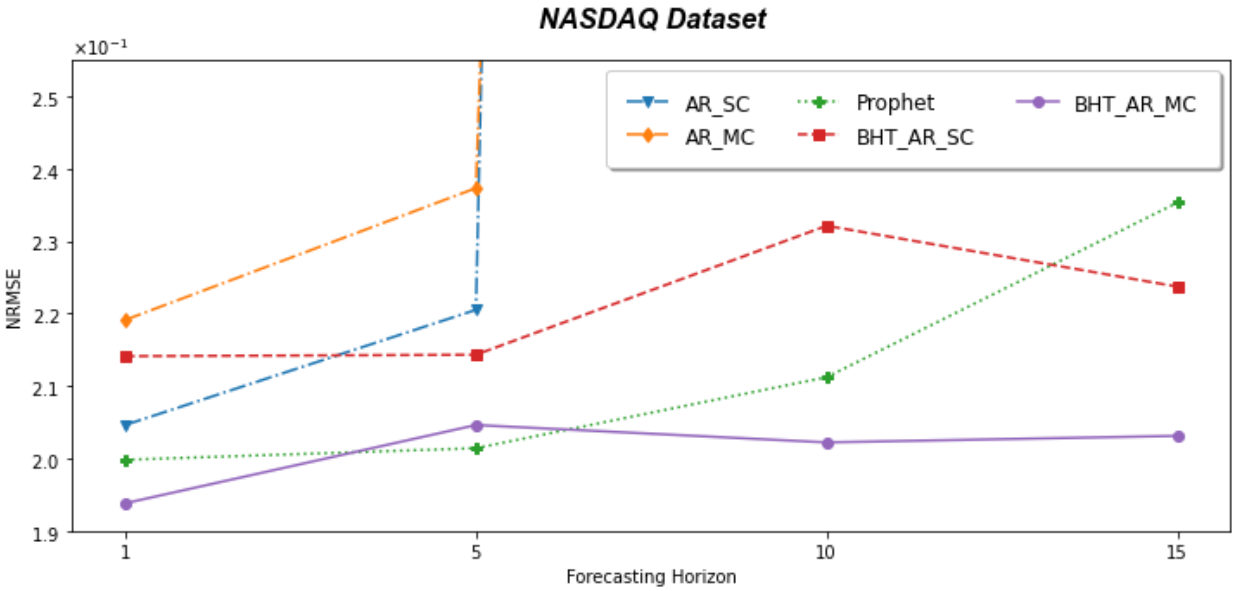

Figure 43: NRMSE vs Forecasting Horizon on the NASDAQ Dataset

5. RELATED WORK

In this chapter we provide a brief review of two additional works in order to provide a wider overview of the vast research field that is tensor time series analysis. We chose two works that pose a lot of interest and are conceptually similar with the rest of this thesis and thus they can be easily grasped by the reader.

Li et al [12] worked on tensor-based prediction with applications on manufacturing and transportation systems, where the spatiotemporal correlations between the data are strong. They focused on short and long term prediction of passenger flow for an Urban Rapid Transit system. For the short-term prediction task they proposed a tensor completion method based on tensor clustering while for the long-term prediction task they proposed a method that combines tensor decomposition and the solution of a 2-dimensional Autoregressive Moving Average model. The spatial dependencies between stations are studied over two aspects. The first aspect is the geographical position of each station since a station's passenger flow is usually affected by its neighbours. The second aspect corresponds to the contextual similarities (e.g school, industrial area etc). The temporal correlations are studied over two scales, weekly and daily. On the weekly scale an observation is correlated with observations of the same day on previous weeks while on the daily scale it is correlated with observations of the previous days.

The two approaches will be outlined in the following paragraphs. The data are represented as a tensor $\mathcal{X} \in \mathbb{R}^{L \times T \times P}$ where L corresponds to the location of 120 stations, T is the temporal mode which corresponds to 59 days and P corresponds to 247 observations recorded over a day with a 5-minute sampling period. Thus, a forecasting horizon of t days would result in the tensor $\mathcal{X} \in \mathbb{R}^{L \times (T+t) \times P}$.

2-step Tensor Long-Term Prediction

This method combines CP Decomposition with the solution of a 2D-ARMA model. The first step is to decompose $\mathcal{X} \in \mathbb{R}^{L \times T \times P}$ as a summation of R rank-one tensors:

$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r \boldsymbol{u}_r^{(1)} \circ \boldsymbol{u}_r^{(2)} \circ \boldsymbol{u}_r^{(3)}$$

where $\boldsymbol{u}_r^{(k)} \in \mathbb{R}^{I_k}$ is a unit vector. Additionally, $\boldsymbol{U}^{(k)} \in \mathbb{R}^{I_k \times R}$ is defined as the matrix obtained by stacking the R column-vectors that correspond to the k-th dimension. For simplicity we will refer to them as $\boldsymbol{U}_L, \boldsymbol{U}_T$ and $\boldsymbol{U}_P$. At this point we utilize the 2D-ARMA model to forecast t observations. The 2D-ARMA process is applied on $\boldsymbol{U}_T \in \mathbb{R}^{T \times R}$ as follows:

For each $\boldsymbol{u}_r \in \mathbb{R}^T$, $r = 1, 2, ..., R$, corresponding to the r-th column of $\boldsymbol{U}_T$ for $r = 1, 2, ..., R$, do the following:

• Reshape $\boldsymbol{u}_r$ as a 2D matrix $\boldsymbol{V}_r$, where each row represents a day of the week and each column the seven days of the week. Note that $\boldsymbol{V}_r \in \mathbb{R}^{D \times W}$ with $DW \geq R$

• Apply 2D-ARMA iteratively to forecast t values into the future. In each iteration $\boldsymbol{V}_r$ is updated to include the new predicted value.

• When every $\boldsymbol{V}_r$ has been updated to include the t predicted values, they are vectorized into $\boldsymbol{u}_r^{'} \in \mathbb{R}^{T+t}$. These column-vectors are stacked to generate $\boldsymbol{U}_{T+t} \in \mathbb{R}^{(T+t) \times R}$

• Finally, use $\boldsymbol{U}_L, \boldsymbol{U}_{T+t}$ and $\boldsymbol{U}_P$ to reconstruct the tensor $\mathcal{X} \in \mathbb{R}^{L \times (T+t) \times P}$

2D-ARMA: Each $\boldsymbol{V}_r$ can be seen as a random field $v[d, w]$. The 2D-ARMA model uses the values located inside a rectangle positioned such as to contain the to-be forecasted value at the lower right corner. More formally, a 2D-ARMA(p_1, p_2, q_1, q_2) model uses the elements of the random field whose indices satisfy $0 \leq d \leq D-1$ and $0 \leq w \leq W-1$. It is described by the following equation:

$$\sum_{i=0}^{p_1}\sum_{j=0}^{p_2} a_{ij} v[d-i, w-j] = \sum_{i=0}^{q_1}\sum_{j=0}^{q_2} b_{ij} e[d-i, w-j]$$

where $a_{00} = 1$, $e[d, w]$ is a stationary white noise random field, a_{ij}, b_{ij} the coefficients of the model, p_1, p_2 are the time lags for the random field v and q_1, q_2 are the time lags for the random field e. The coefficient estimation process can be found in the work of Bouaynaya et al [13].

Another challenge that was addressed in the long-term prediction setting, was the efficient updating of the prediction as new data arrived throughout the day. Suppose that the predictions for the following day have already been made, resulting in a tensor $\mathcal{X} \in \mathbb{R}^{L \times (T+1) \times P}$. The procedure of updating efficiently the forecast is the following:

• Apply CP Decomposition to $\mathcal{X}$ and obtain $\boldsymbol{U}_L, \boldsymbol{U}_T$ and $\boldsymbol{U}_P$. The updates take place on dimension L.

• Update tensor $\mathcal{X}$ with the new data and keep the rest of the predicted values. Use the following formula to update $\boldsymbol{U}_L$.

$$\boldsymbol{U'}_L = \boldsymbol{X}^{(0)}(\boldsymbol{U}_T \odot \boldsymbol{U}_P)(\boldsymbol{U}_T^T \boldsymbol{U}_T * \boldsymbol{U}_P^T \boldsymbol{U}_P)^{\dagger}$$

Where $\boldsymbol{X}^{(0)}$ is the unfolding of tensor $\mathcal{X}$ over dimension L, $\odot$ is the Khatri-Rao product, $*$ is the Hadamard product and $\dagger$ denotes the Moore-Penrose inverse of a matrix.

• Finally, reconstruct $\mathcal{X}$ using $\boldsymbol{U'}_L$, $\boldsymbol{U}_T$ and $\boldsymbol{U}_P$.

1-step Tensor Short-Term Prediction

For the short-term prediction task they proposed a method based on Tensor Completion. Tensor Completion was originally used as a way to estimate and fill a tensor's missing data. This problem is formulated as the minimization of the quantity $\|\mathcal{X}_\Omega - \mathcal{Y}_\Omega\| + \alpha\|\mathcal{Y}\|_*$ over $\mathcal{Y}$, where $\mathcal{X}$ is the incomplete tensor and $\mathcal{Y}$ is the completed output. To ensure low-rank nuclear norm is used ($\|.\|_*$). Finally Ω is a mask that "hides" the missing elements of tensor $\mathcal{X}$. This is achieved by enabling only the indices of $\mathcal{X}$ that contained an element.

They treated the sequence of data as observed values and the horizon to be predicted as missing. They utilized the *Bayesian Low-Rank Tensor Completion* framework, originally proposed by Zhao et al [14]. Utilizing the factor matrices $\boldsymbol{U}$, the log-joint distribution and the posterior distribution this framework estimates the missing values by: $P(\mathcal{Y}_{\Omega^C}|\mathcal{Y}_\Omega) = \int P(\mathcal{Y}_{\Omega^C}|\Theta)\, P(\Theta|\mathcal{Y}_\Omega)$. However, this method is suited for low-rank tensors. This prerequisite is assumed to be violated in the case of a URT since there is an observed diversity in the data over different stations. To overcome this obstacle they used tensor clustering [15] [16] to group similar elements of $\boldsymbol{U}_L$ in clusters and then apply Bayesian LRTC in each cluster separately.

The whole process can be summarized by the following steps:

• Apply CP decomposition to obtain $\boldsymbol{U}_L$.

• Conduct Principal Components Analysis to reduce the dimension R.

• Cluster the data by using k-means, Hierarchical clustering or other clustering algorithms.

• Since each row of $\boldsymbol{U}_T$ is assigned to a cluster, the Bayesian LRTC can be conducted by modifying mask Ω to include only the elements that are grouped together in the same cluster.

Sedighin et al [17] addressed the problem of time series reconstruction in a multi-way delay embedded space through Tensor Train decomposition. In many problems data may be incomplete or corrupted by outliers or noise. Completion is the problem where an algorithm has to estimate these values using only the available or uncorrupted values. Hankelization has been utilized in various settings, since it is expected to result in a higher order tensor with relatively low rank. In this thesis, we have already seen Hankelization based on individual elements. However, an alternative Hankelization procedure was utilized in this paper that uses blocks of elements. This Hankelization variation allows us to exploit local correlations between neighbouring sub-windows of time series in addition to the local correlations of elements. For the tensor completion task, Tensor Train Decomposition was utilized since it usually provides better representations for higher order tensors compared to Tucker or CP Decomposition. Furthermore, Tensor Train decomposition is more stable and does not suffer from the curse of dimensionality. In Tensor Train a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times ... \times I_k}$ is decomposed as a set of N interconnected third order core tensors $\mathcal{G}^{(n)} \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$ where $R_1 = R_{N+1} = 1$.

The Hankelization Technique

As mentioned, the Hankelization technique used in this work transforms the data into a block Hankel tensor using blocks of elements. Suppose that we are given a scalar time series. If the temporal mode is taken into account the dataset takes the form of a vector $\boldsymbol{v}$. The steps that lead to that tensorization are summarized below:

• Transform $\boldsymbol{v}$ into a Hankel matrix $\boldsymbol{V}$.

• Multiply $\boldsymbol{V}$ by two block duplication matrices. Each duplication matrix is similar with the one described in 3.1. It contains smaller identity matrices of size $PT_k \times PT_k$, where T_k is the window size and P is the block size. Another difference with the procedure described in 3.1 is that two consecutive identity matrices are shifted by P columns in the duplication matrix $\boldsymbol{S}$. The Block Hankel matrix $\boldsymbol{V}_H \in \mathbb{R}^{PT_1(I_1/P - T_1 + 1) \times PT_2(I_2/p - T_2 + 1)}$ is calculated as: $\boldsymbol{V}_H = \boldsymbol{S}_1 \boldsymbol{V} \boldsymbol{S}_2^T$ where $\boldsymbol{S}_k \in \mathbb{R}^{PT_k(I_k/P - T_k + 1) \times I_k}$. Note that when I_k/P is not an integer, then zero-padding is applied on $\boldsymbol{V}$ so that I_k/P becomes an integer.

• Fold $\boldsymbol{V}_H$ to obtain the 6-th order tensor $\mathcal{V}_H \in \mathbb{R}^{P \times P \times T_1 \times (I_1/P - T_1 + 1) \times T_2 \times (I_2/P - T_2 + 1)}$.

The Algorithm

The objective is to complete tensor $\mathcal{V}_H$ using Tensor Train decomposition. This is formulated as the minimization of $J(\theta) = \|\Omega * (\mathcal{V}_H - \hat{\mathcal{V}}_H(\theta))\|_F^2$ where Ω is a tensor-mask that has 1 in the positions where a value is observed in $\mathcal{V}_H$ and 0 otherwise. $\hat{\mathcal{V}}_H(\theta)$ is the estimation of the tensor through the core tensors $\theta = (\mathcal{G}^{(1)}, \mathcal{G}^{(2)}, ..., \mathcal{G}^{(N)})$. Using a Majorization Minimaization approach, the objective is reformulated as the minimization of the auxiliary function $J(\theta|\theta^k) = \|\Omega * (\mathcal{V}_H - \hat{\mathcal{V}}_H(\theta))\|_F^2 + \|(\underline{\mathbf{1}} - \Omega) * (\hat{\mathcal{V}}_H(\theta^k) - \hat{\mathcal{V}}_H(\theta))\|_F^2$ where $\underline{\mathbf{1}}$ is a 6-th order tensor whose elements are all equal to 1. Since the two masked tensors inside these two Frobenius norms do not have elements in overlapping positions, the function can be rewritten as:

$$J(\theta|\theta^k) = \|\Omega * (\mathcal{V}_H - \hat{\mathcal{V}}_H(\theta))\|_F^2 + \|(\underline{\mathbf{1}} - \Omega) * (\hat{\mathcal{V}}_H(\theta^k) - \hat{\mathcal{V}}_H(\theta))\|_F^2 \Rightarrow$$
$$J(\theta|\theta^k) = \|\Omega * (\mathcal{V}_H - \hat{\mathcal{V}}_H(\theta)) + (\underline{\mathbf{1}} - \Omega) * (\hat{\mathcal{V}}_H(\theta^k) - \hat{\mathcal{V}}_H(\theta))\|_F^2 \Rightarrow$$

$$J(\theta|\theta^k) = \|\Omega * (\mathcal{V}_H - \hat{\mathcal{V}}_H(\theta)) + \underline{\mathbf{1}} * (\hat{\mathcal{V}}_H(\theta^k) - \hat{\mathcal{V}}_H(\theta)) - \Omega * (\hat{\mathcal{V}}_H(\theta^k) - \hat{\mathcal{V}}_H(\theta))\|_F^2 \Rightarrow$$
$$J(\theta|\theta^k) = \|\Omega * (\mathcal{V}_H - \hat{\mathcal{V}}_H(\theta) - \hat{\mathcal{V}}_H(\theta^k) + \hat{\mathcal{V}}_H(\theta)) + \underline{\mathbf{1}} * (\hat{\mathcal{V}}_H(\theta^k) - \hat{\mathcal{V}}_H(\theta))\|_F^2 \Rightarrow$$
$$J(\theta|\theta^k) = \|\Omega * \mathcal{V}_H - \Omega * \hat{\mathcal{V}}_H(\theta^k)) + \underline{\mathbf{1}} * \hat{\mathcal{V}}_H(\theta^k) - \underline{\mathbf{1}} * \hat{\mathcal{V}}_H(\theta)\|_F^2 \Rightarrow$$
$$J(\theta|\theta^k) = \|\Omega * \mathcal{V}_H + (\underline{\mathbf{1}} - \Omega) * \hat{\mathcal{V}}_H(\theta^k)) - \hat{\mathcal{V}}_H(\theta)\|_F^2$$

The above cost function is minimized using an Alternating Least Squares approach. Initially the core tensors are initialized randomly. In each iteration a new estimation of the core tensors is produced which then serves as their initialization on the next iteration. For the core tensors a rank incremental strategy is followed in which the elements of the cores whose rank will be increased take new random values.

It is important to determine the ranks of the core tensors $\mathcal{G}^{(n)}$. The rank incremental strategy that was proposed began with low ranks on the core tensors and gradually increased the rank of the core tensor that produced the largest approximation error.

The matricization of the estimated tensor $\hat{\boldsymbol{V}}_H(\theta) := \hat{\boldsymbol{V}}_H$ over the n-th mode can be written as $\hat{\mathcal{V}}_{H(n)} = \boldsymbol{G}_{(2)}^{(n)} \left(\boldsymbol{G}_{(1)}^{>n} \otimes \boldsymbol{G}_{(n)}^{<n} \right)$ [18] where:

- $\boldsymbol{G}_{(1)}^{>n} \in \mathbb{R}^{R_{n+1} \times I_{n+1} \ldots I_N}$ is the 1-st mode unfolding of the tensor
  $\boldsymbol{G}^{>n} = \ll \boldsymbol{G}^{(n+1)}, \boldsymbol{G}^{(n+2)}, ..., \boldsymbol{G}^{(N)} \gg \in \mathbb{R}^{R_{n+1} \times I_{n+1} \times \ldots \times I_N}$
- $\boldsymbol{G}_{(n)}^{<n} \in \mathbb{R}^{R_n \times I_1 \ldots I_{n-1}}$ is the n-th mode unfolding of the tensor
  $\boldsymbol{G}^{<n} = \ll \boldsymbol{G}^{(1)}, \boldsymbol{G}^{(2)}, ..., \boldsymbol{G}^{(n-1)} \gg \in \mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times R_n}$
- $\boldsymbol{G}_{(2)}^{(n)} \in \mathbb{R}^{I_n \times R_n R_{n+1}}$ is the 2-nd mode unfolding of the tensor
  $\boldsymbol{G}^{(n)} \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$
- $\boldsymbol{G}^{>N} = \boldsymbol{G}^{<1} = 1$

The approximation error of the n-th core can be estimated by the following formula.

$$e(n) = \|\boldsymbol{G}_{(2)}^{(n)\dagger} (\boldsymbol{\Omega}_{(n)} * (\boldsymbol{V}_{H(n)} - \hat{\boldsymbol{V}}_{H(n)}))\|_F^2$$

In each iteration we estimate the approximation error for each core tensor and update the one with the highest approximation error. The ranks are updated using the following formula.

$$R_n^{new} = min(R_n + step, R_{n-1} I_{n-1}, I_n R_{n+1})$$

Note that when a tensor $\mathcal{G}^{(n)} \in \mathbb{R}^{R_n \times I_n \times R_{n+1}}$ is chosen to be updated, the updates take place on both ranks R_n and R_{n+1}. In addition, its neighbouring core tensors $\mathcal{G}^{(n-1)} \in \mathbb{R}^{R_{n-1} \times I_{n-1} \times R_n}$ and $\mathcal{G}^{(n+1)} \in \mathbb{R}^{R_{n+1} \times I_{n+1} \times R_{n+2}}$ ranks R_n and R_{n+1} should be updated as well. Furthermore, this update mechanism can be used simultaneously on various non-consecutive core tensors, whose approximation errors are the highest.

After the completion of the tensor is finished, a de-Hankelization technique must be applied to transform the data back into their original framework. In this work, the tensor is de-Hankelized using an averaging step on the blocks that correspond to a specific block in the original tensor. This process results in a Hankel matrix which is then used to reconstruct the time series.

6. CONCLUSION & FUTURE WORK

We conclude this thesis by providing a summary of this work, presenting our results and highlighting various paths that we would like to explore in the future in order to improve it.

We focus on the Block Hankel Tensor Autoregression algorithm which is essentially a combination of 3 major parts. The first part of the algorithm is Hankelization, a tensorization method that maps the data as a tensor of a symmetric structure that constitutes the generalization of the Hankel matrix. In our case, Hankelization is applied on the temporal mode. Various works have studied Hankelization and it has been shown experimentally, that for data with an underlying hidden low-rank structure, their produced Hankel tensor can be represented by low-rank or a smooth manifold in the embedded space. The second vital part of the algorithm is low-rank Tucker Decomposition, a process in which a tensor is decomposed as a core tensor with lower dimensions that captures the intrinsic correlations of the data and a set of jointly used factor matrices. We apply Tucker Decomposition on every mode but the temporal and use the obtained core tensors to train an autoregressive forecasting model which constitutes the third part of the algorithm. Finally, in an effort to provide the first step towards the generalization of the algorithm we substitute the scalar coefficients of the forecasting model with matrices.

To evaluate the performance of Block Hankel Tensor Autoregression and its generalization, we conducted experiments on 1 synthetically generated and 7 publicly available datasets. To provide a reliable evaluation, they were compared with other widely-used models like autoregression with scalar and matrix coefficients and Facebook's Prophet. The chosen metric that measured the performance of each algorithm was the NRMSE. We conducted an analysis on the convergence of the algorithms and their sensitivity towards the user defined hyperparameters. Additionally, we experimented on short time series. The algorithms were evaluated with regards to their time efficiency and forecasting error on various cases of short and long term forecasting. Furthermore, we conducted an experiment where we measured the NRMSE while training each algorithm on subsets of different volumes. This experiment's goal is to evaluate and provide a better understanding about the effects of training each algorithm on fewer observations.

Our findings suggest that both BHT_AR_SC & BHT_AR_MC converge really fast, under only a few iterations. They show high sensitivity towards the parameters' choice even in evaluations with slightly different data. At the same time they show small need for careful parameter picking, as different sets of parameters provide similar results within the same evaluation. In the short time series setting they show great forecasting results. However the matrix coefficient variation shows a more stable behaviour both for short and long term forecasting. Furthermore, the performance of BHT_AR_MC does not seem to be affected when the training set volume is greatly reduced which indicates a great performance even on very short time series. Additionally, when dealing with longer forecasting horizons both variations perform well but they do not show the previously described stability. However, this is something that was expected since they were designed for time series with very few observations and shorter forecasting horizons. Finally, both BHT_AR_SC & BHT_AR_MC are time efficient which means that they can be applied in various real world applications.

While we would like to conduct more experiments and test various different ideas we are limited by the constraint of time. In the following paragraphs we mention some of these ideas and encourage the readers to implement and evaluate, in addition to their own ideas, those that seem more interesting in order to contribute in the advancement of the tensor time series analysis research field.

As mentioned Block Hankel Tensor Autoregression is constructed using 3 key parts. As a first step towards experimentation we could try different choices and combinations for these parts. For example, we can try to substitute the existing Hankelization method. As mentioned in the related works section, *Sedighin et al* [17] utilized a variation of Hankelization which could prove to be useful in our setting. Furthermore, in our work we utilized Hankelization on the temporal mode only. In future works we would like to experiment by applying Hankelization on other modes as well. However, applying Hankelization on every mode could result in tensors of very high orders which could affect the performance of the algorithm. Therefore, we should proceed by monitoring the trade-off between the benefits of Hankelization and the performance of the algorithm.

Another path that should be explored is that of the forecasting model. In this thesis we substitute the scalar coefficients with matrices. The first path that we would like to explore in the future is the effect of Hankelization on the model. It is possible that Hankelization affects the coefficient matrices, forcing a specific structure on them. Therefore, it is important to explore this possibility and adjust the algorithm accordingly. In addition, we would like to extend the algorithm by substituting the current forecasting process with a VARIMA model. Recently *Chen et al* [19] proposed a bilinear model for matrix valued time series which could be utilized in our setting as well.

Apparently, the path that seems to show the least potential for improvement, is that of the decomposition process. While there are various tensor decomposition methods, it is not straightforward how they could be utilized in our setting. In a variation of the algorithm we could utilize Tensor Train Decomposition to obtain a set of interconnected core tensors. Using these core tensors we can train in parallel different forecasting processes, one for each tensor respectively. Finally, the forecasted core tensors would be used in the inverse process to reconstruct the predicted tensor in the original space. Note that tensor train decomposition can be combined harmonically with the matrix coefficient autoregression model since these core tensors would have smaller sizes compared to the core tensor obtained by Tucker Decomposition.

Finally, an important step towards establishing these algorithms as an industrial standard, is their ability to update the forecasting process in real time which is not a straightforward task but it is surely a path that is worth experimenting on.

ABBREVIATIONS - ACRONYMS

SVD	Singular Value Decomposition
PCA	Principal Component Analysis
CPD	Canonical Polyadic Decomposition
MDT	Multi-way Delay Transform
AR	Autoregression
VAR	Vector Autoregression
BHT	Block Hankel Tensor
BHT_AR_SC	Block Hankel Tensor Autoregression with Scalar Coefficients
BHT_AR_MC	Block Hankel Tensor Autoregression with Matrix Coefficients

BIBLIOGRAPHY

[1] H. M. W. B. Herath, “TENSOR REGRESSION AND TENSOR TIME SERIES ANALYSES FOR HIGH DIMENSIONAL DATA,” Master’s thesis, University of Peradeniya, Jun. 2019.

[2] S. Rabanser, O. Shchur, and S. Günnemann, “Introduction to Tensor Decompositions and their Applications in Machine Learning,” *ArXiv*, vol. abs/1711.10781, 2017.

[3] N. D. Sidiropoulos, L. D. Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, “Tensor Decomposition for Signal Processing and Machine Learning,” *IEEE Transactions on Signal Processing*, vol. 65(13), pp. 3551 – 3582, Jul. 2017.

[4] T. G. Kolda and B. W. Bader, “Tensor Decompositions and Applications,” *SIAM Review*, vol. 51(3), Aug. 2009.

[5] T. Yokota, B. Erem, S. Guler, S. K. Warfield, and H. Hontani, “Missing Slice Recovery for Tensors Using a Low-rank Model in Embedded Space,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8251–8259, IEEE, Jun. 2018.

[6] T. Yokota and H. Hontani, “Tensor completion with shift-invariant cosine bases,” in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 1325–1333, 2018.

[7] P. Jing, Y. Su, X. Jin, and C. Zhang, “High-Order Temporal Correlation Model Learning for Time-Series Prediction,” *IEEE Transactions on Cybernetics*, vol. 49(6), pp. 2385–2397, Jun. 2019.

[8] Q. Shi, J. Yin, J. Cai, A. Cichocki, T. Yokota, L. Chen, M. Yuan, and J. Zeng, “Block Hankel Tensor ARIMA for Multiple Short Time Series Forecasting,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34(4), pp. 5758–5766, Apr. 2020.

[9] F. Shang, Y. Liu, and J. Cheng, “Generalized Higher-Order Tensor Decomposition via Parallel ADMM,” *Proceedings of the National Conference on Artificial Intelligence*, vol. 28(1), Jul. 2014.

[10] C. W. Anderson, E. A. Stolz, and S. Shamsunder, “Multivariate Autoregressive Models for Classification of Spontaneous Electroencephalogram During Mental Tasks,” *IEEE Transactions on Biomedical Engineering*, Mar. 1998.

[11] R. H. Shumway and D. S. Stoffer, *Time Series Analysis and Its Applications*, ch. 5, pp. 271–282. Springer International Publishing, 4th ed., 2017.

[12] Z. Li, H. Yan, C. Zhang, and F. Tsung, “Long-Short Term Spatiotemporal Tensor Prediction for Passenger Flow Profile,” *IEEE Robotics and Automation Letters*, vol. 5(4), pp. 5010–5017, Oct. 2020.

[13] N. Bouaynaya, J. Zielinski, and D. Schonfeld, “Two-dimensional ARMA modeling for breast cancer detection and classifications,” in *2010 International Conference on Signal Processing and Communications (SPCOM)*, pp. 1–4, Jul. 2010.

[14] Q. Zhao, L. Zhang, and A. Cichocki, “Bayesian CP Factorization of Incomplete Tensors with Automatic Rank Determination,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37(9), p. 1751–1763, Sep. 2015.

[15] K. Yu, L. He, P. S. Yu, W. Zhang, and Y. Liu, “Coupled Tensor Decomposition for User Clustering in Mobile Internet Traffic Interaction Pattern,” *IEEE Access*, vol. 7, pp. 18113–18124, Jan. 2019.

[16] W. W. Sun and L. Li, “Dynamic Tensor Clustering,” *Journal of the American Statistical Association*, vol. 114(528), pp. 1894–1907, Oct. 2019.

[17] F. Sedighin, A. Cichocki, T. Yokota, and Q. Shi, “Matrix and Tensor Completion in Multiway Delay Embedded Space Using Tensor Train, with Application to Signal Reconstruction,” *IEEE Signal Processing Letters*, vol. 27, pp. 810–814, Apr. 2020.

[18] L. Yuan, Q. Zhao, L. Gui, and J. Cao, “High-order Tensor Completion via Gradient-based Optimization Under Tensor Train Format,” *Signal Processing: Image Communication*, vol. 73, pp. 53–61, Apr. 2019.

[19] R. Chen, H. Xiao, and D. Yang, “Autoregressive Models for Matrix-Valued Time Series,” *Journal of Econometrics*, Aug. 2020.