Compression Methods for Structured Floating-Point Data and their Application in Climate Research

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

von der KIT-Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Uğur Çayoğlu

aus Karlsruhe

Tag der mündlichen Prüfung: 18. Dezember 2019
Erster Gutachter: Prof. Dr. Achim Streit
Zweiter Gutachter: Prof. Dr. Peter Braesicke

Erklärung zur selbständigen Anfertigung der Dissertationsschrift
Hiermit erkläre ich, dass ich die Dissertationsschrift mit dem Titel

Compression Methods for Structured Floating-Point Data and their Application in Climate Research

selbständig angefertigt und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und die Regeln zur Sicherung guter wissenschaftlicher Praxis am Karlsruher Institut für Technologie (KIT) beachtet habe.

Ort, Datum Uğur Çayoğlu

Anne, baba ve ablamlara adanmıştır

Acknowledgements

This work would have not been possible without the help of several important people. I want to use this opportunity to express my gratitude and appreciation to everyone who has helped make this possible.

First of all, I would like to thank Prof. Dr. Achim Streit for supervising and to Prof. Dr. Peter Braesicke for co-supervising this thesis. Thank you, Prof. Dr. Achim Streit, for giving me the opportunity to pursue the doctoral degree under your guidance. Throughout the four and a half years at the institute, you helped me with invaluable advice, crucial for the quality of my research which made it possible to publish my work at top venues. I would also like to express my deep appreciation to Prof. Dr. Peter Braesicke. Your continuous support of my doctoral studies and tireless editing efforts have improved my work tremendously.

Besides, I would like to thank you, Jörg Meyer, for always having an open door for discussions, your contagious love for science, your passionate (and misguided) love for dying programming languages, and for being the most passionate physicist I know. Tobias Kerzenmacher, thank you for proofreading this thesis and the several papers we have written together, for having always an open ear for questions, and your support during difficult times. Thank you, Frank Tristram, for our innumerable discussions about bits and bytes as well as the German party system. Your thirst for knowledge, your head on approach to science, and your ability to think outside the box I will never forget. You might want to work on your returns on squash, though.

Further, I would like to thank my colleagues and friends: Darko Dubravica for being bold and accepting to be the first to proofread this thesis, for being a genuinely good person, and for the time spend on and off the football pitch especially the nutmegging challenges*; Arsen Hayrapetyan for being

*I won every single time.

a true friend, having an open heart for everyone, and sharing the passion to take the side of the belittled on a debate just to drive people mad; Andreas Schoknecht for being an exemplary scientist, for being always clear and logical in expression and thought; Daniel Hofmann for proofreading this thesis, for your welcoming heart, your never-ending knowledge about data structures and algorithms, for your contagious passion for analog photography and memes†; Amelie Ninja Röhling for having an open ear for everything, introducing me to wakeboarding and support; Marleen Braun for being one of the greatest persons I got to know and for sharing the same ~~weird~~ great sense of humour and appetite; Diana Gudu and Qiansi Tu for sharing your offices with me; Benjamin Ertl for being the most leaned back person I know; Matthias Frey for always directly communicating your thoughts; Barış Özçelik for helping me relax and getting my mind off work if necessary and being always there for me; Roland Ruhnke for your incredible ability to pinpoint problems and calling a spade a spade; Peter Krauß for always helping out and sharing a passion for keyboards, fonts, kerning and proper aᴉgnmeᵰt; Elnaz Azmi for our discussion about Matlab and parallelisation; Christian Scharun for your wonderful (and sometimes dark) sense of humour; Marcus Strobl for being always positive and adding value to our discussions during the tea breaks; Carlos Alberti for our time on the football pitch; and to all the others who joined me through this endeavour: Christopher Jung, Eileen Kühn, Florian Haenel, Isabella Bierenbaum, Jennifer Schröter, Jörg Kramer, Jos van Wezel, Lukas Muser, Manuel Giffels, Marcus Hardt, Max Fischer, Michael Weimer, Parinaz Ameri, Pavel Efros, Samuel Ambroj Perez, Sören Johansson, Sabine Barthlott, and Stella Möhrle.

† ¯_(ツ)_/¯

Abstract

The use of new technologies, such as GPU boosters, have led to a dramatic increase in the computing power of High-Performance Computing (HPC) centres. This development, coupled with new climate models that can better utilise this computing power thanks to software development and internal design, led to the bottleneck moving from solving the differential equations describing Earth's atmospheric interactions to actually storing the variables. The current approach to solving the storage problem is inadequate: either the number of variables to be stored is limited or the temporal resolution of the output is reduced. If it is subsequently determined that another variable is required which has not been saved, the simulation must run again. This thesis deals with the development of novel compression algorithms for structured floating-point data such as climate data so that they can be stored in full resolution.

Compression is performed by decorrelation and subsequent coding of the data. The decorrelation step eliminates redundant information in the data. During coding, the actual compression takes place and the data is written to disk. A lossy compression algorithm additionally has an approximation step to unify the data for better coding. The approximation step reduces the complexity of the data for the subsequent coding, e.g. by using quantification. This work makes a new scientific contribution to each of the three steps described above.

This thesis presents a novel lossy compression method for time-series data using an Auto Regressive Integrated Moving Average (ARIMA) model to decorrelate the data. In addition, the concept of information spaces and contexts is presented to use information across dimensions for decorrelation. Furthermore, a new coding scheme is described which reduces the weaknesses of the eXclusive-OR (XOR) difference calculation and achieves

a better compression factor than current lossless compression methods for floating-point numbers. Finally, a modular framework is introduced that allows the creation of user-defined compression algorithms.

The experiments presented in this thesis show that it is possible to increase the information content of lossily compressed time-series data by applying an adaptive compression technique which preserves selected data with higher precision. An analysis for lossless compression of these time-series has shown no success. However, the lossy ARIMA compression model proposed here is able to capture all relevant information. The reconstructed data can reproduce the time-series to such an extent that statistically relevant information for the description of climate dynamics is preserved.

Experiments indicate that there is a significant dependence of the compression factor on the selected traversal sequence and the underlying data model. The influence of these structural dependencies on prediction-based compression methods is investigated in this thesis. For this purpose, the concept of Information Spaces (IS) is introduced. IS contributes to improving the predictions of the individual predictors by nearly 10% on average. Perhaps more importantly, the standard deviation of compression results is on average 20% lower. Using IS provides better predictions and consistent compression results.

Furthermore, it is shown that shifting the prediction and true value leads to a better compression factor with minimal additional computational costs. This allows the use of more resource-efficient prediction algorithms to achieve the same or better compression factor or higher throughput during compression or decompression. The coding scheme proposed here achieves a better compression factor than current state-of-the-art methods.

Finally, this paper presents a modular framework for the development of compression algorithms. The framework supports the creation of user-defined predictors and offers functionalities such as the execution of benchmarks, the random subdivision of n-dimensional data, the quality evaluation of predictors, the creation of ensemble predictors and the execution of validity tests for sequential and parallel compression algorithms.

This research was initiated because of the needs of climate science, but the application of its contributions is not limited to it. The results of this thesis are of major benefit to develop and improve any compression algorithm for structured floating-point data.

Zusammenfassung

Der Einsatz von neuen Technologien, wie GPU-Boostern, haben zu einem dramatischen Anstieg der Rechenleistung von HPC-Zentren geführt. Diese Entwicklung gekoppelt mit neuen Klimamodellen, welche diese Rechenleistung dank Softwareentwicklungen und internem Aufbau besser auslasten können, führte dazu, dass sich der Engpass weg von der Lösung der Differentialgleichungen hin zur eigentlichen Speicherung der Variablen verschob. Der aktuelle Ansatz zur Lösung des Speicherproblems ist unzureichend: Entweder wird die Anzahl der zu speichernden Variablen begrenzt oder die zeitliche Auflösung der Ausgabe reduziert. Sollte im Nachhinein festgestellt werden, dass eine weitere Variable notwendig ist, welche nicht gespeichert wurde, muss die Simulation von neuem laufen. Diese Arbeit beschäftigt sich mit der Entwicklung neuartiger Kompressionsalgorithmen für strukturierte Gleitkommazahlen wie Klimadaten, damit diese in voller Auflösung gespeichert werden können.

Komprimierung erfolgt durch Dekorrelation und anschließende Kodierung der Daten. Der Dekorrelationsschritt eliminiert redundante Informationen in den Daten. Bei der Kodierung findet die eigentliche Kompression statt und die Daten werden auf die Festplatte geschrieben. Ein verlustbehafteter Kompressionsalgorithmus hat zusätzlich einen Annäherungsschritt, um die Daten für die Kodierung zu vereinheitlichen. Der Annäherungsschritt reduziert die Komplexität der Daten, indem z.B. Methoden der Quantifizierung verwendet werden. Diese Arbeit leistet zu jedem der drei oben beschriebenen Schritte einen neuen wissenschaftlichen Beitrag.

In dieser Dissertation wird ein neuartiges verlustbehaftetes Kompressionsverfahren für Zeitreihendaten vorgestellt, welches ein Auto Regressive Integrated Moving Average (ARIMA) Modell zur Dekorrelation der

Daten verwendet. Darüber hinaus wird das Konzept der Informationsräume und -kontexte vorgestellt, um Informationen über Dimensionen hinweg besser für die Dekorrelation zu nutzen. Weiterhin wird ein neues Kodierungsschema beschrieben, welches die Schwächen der eXclusive-OR (XOR)-Differenzberechnung reduziert und einen besseren Kompressionsfaktor erreicht als aktuelle verlustfreie Kompressionsverfahren für Gleitkommazahlen. Schließlich wird ein modulares Framework eingeführt, das die Erstellung von benutzerdefinierten Kompressionsalgorithmen ermöglicht.

Die in dieser Dissertation aufgeführten Experimente zeigen, dass es möglich ist, den Informationsgehalt von verlustbehaftet komprimierten Zeitreihendaten durch die Anwendung eines adaptiven Kompressionsverfahrens zu erhöhen. Eine Analyse für verlustfreie Kompression dieser Zeitreihen hat keinen Erfolg gezeigt. Das hier vorgeschlagene verlustbehaftete ARIMA-Kompressionsmodell ist jedoch in der Lage alle relevanten Informationen zu erfassen. Die Rekonstruktion der Daten kann die Zeitreihen so weit reproduzieren, dass statistisch relevante Informationen zur Beschreibung der Klimadynamik erhalten bleiben.

Diese Arbeit zeigt, dass eine signifikante Abhängigkeit des Kompressionsfaktors von der gewählten Traversierung und dem zugrunde liegenden Datenmodell besteht. Der Einfluss dieser strukturellen Abhängigkeiten auf Vorhersage-basierende Kompressionsverfahren wird in dieser Arbeit untersucht. Es werden Möglichkeiten vorgestellt diese Abhängigkeiten zu entdecken und den Kompressionsfaktor zu verbessern. Hierfür wird das Konzept der Information Spaces (IS) eingeführt, welches dazu beiträgt, die Vorhersagen der einzelnen Prädiktoren um durchschnittlich fast 10% zu verbessern. Vielleicht noch wichtiger jedoch ist, dass die Standardabweichung der Kompressionsergebnisse um durchschnittlich über 20% verringert wird. Die Verwendung des IS-Ansatzes bietet bessere Vorhersagen und konsistente Kompressionsergebnisse.

Weiterhin wird gezeigt, dass ein besserer Kompressionsfaktor mit minimalen Rechenkosten erreicht wird, wenn vor der Differenzberechnung die Vorhersage und der wahre Wert verschoben werden. Das ermöglicht die Verwendung von ressourcenschonenderen Vorhersagealgorithmen, um den gleichen oder besseren Kompressionsfaktor oder einen höheren Durch-

satz während der Kompression bzw. Dekompression zu erreichen. Darüber hinaus erreicht das hier vorgeschlagene Kodierungsschema einen besseren Kompressionsfaktor als der aktuelle Stand der Technik.

Schließlich wird in dieser Arbeit ein modulares Framework zum Aufbau eigener Kompressionsalgorithmen zur Verfügung gestellt. Dieses Framework unterstützt die Erstellung von benutzerdefinierten Prädiktoren und weitere Funktionen wie die Ausführung von Benchmarks, die zufällige Unterteilung von n-dimensionalen Daten, die Qualitätsbewertung von Prädiktoren, die Erstellung von Ensemble-Prädiktoren und die Ausführung von Gültigkeitstests für sequentielle und parallele Kompressionsalgorithmen.

Diese Forschung wurde durch Bedürfnisse der Klimawissenschaften begründet. Letztendlich sind die Ergebnisse aber nicht auf die Klimawissenschaften beschränkt. Die Ergebnisse dieser Arbeit sind zur Entwicklung und Verbesserung eines beliebigen Kompressionsalgorithmus für strukturierte Gleitkommazahlen von großem Nutzen.

Contents

1 Introduction 1

1.1 Contributions to the Research Field 3

1.2 Outline of the Thesis 7

2 Background 11

2.1 Data Compression 11

2.1.1 Classification of Compression Algorithms 13

2.1.2 Data Compression and Information Theory 14

2.1.3 Application of Custom Compression Algorithms . . 16

2.1.4 Prediction-Based Compression 17

2.1.5 Metrics 21

2.2 Climate Data 22

2.2.1 Weather and Climate Models 22

2.2.2 Data Structure of Climate Model Output 24

3 Related Work 27

3.1 Compression of Climate Data 27

3.2 Compression of Data Similar to Climate Data 29

3.2.1 Image Compression 30

3.2.2 Time-Series Compression 33

3.2.3 Floating-Point Compression 34

3.3 Compression of Other Data 39

4 Data Analysis and Identification of Redundant Information 43

4.1 Motivation 43

4.2 Proposed Methods 44

4.3 Experimental Setup 45

4.3.1 Data . . . 45
4.3.2 Experiments . . . 46
4.4 Evaluation . . . 47
4.4.1 Long-Term Variance Analysis . . . 47
4.4.2 Short-Term Variance Analysis . . . 49
4.4.3 Entropy Analysis . . . 53
4.4.4 Mutual Information Analysis . . . 54
4.5 Summary . . . 58
4.6 Code and Data Availability . . . 59

5 Data Decorrelation using Information Spaces and Contexts 61
5.1 Motivation . . . 62
5.2 Proposed Method . . . 62
5.2.1 Consolidation of Predictions . . . 64
5.2.2 Traversal Methods . . . 65
5.2.3 Predictors . . . 66
5.3 Experimental Setup . . . 66
5.3.1 Data . . . 66
5.3.2 Metrics . . . 67
5.3.3 Experiments . . . 68
5.4 Evaluation . . . 69
5.4.1 Expt 1: Influence of Starting Point . . . 69
5.4.2 Expt 2: Traversal Order of Dimensions . . . 71
5.4.3 Expt 3: New Traversal without the use of Information Spaces . . . 71
5.4.4 Expt 4: New Traversal with the use of Information Spaces . . . 74
5.5 Summary . . . 78
5.6 Code and Data Availability . . . 80

6 Data Approximation using ARIMA Models 81
6.1 Motivation . . . 81
6.2 Proposed Method . . . 82
6.2.1 Model . . . 83
6.2.2 Compression . . . 84
6.2.3 Replacement Methods . . . 84

6.3 Experimental Setup . . . 86
6.3.1 Data . . . 86
6.3.2 Metrics . . . 89
6.3.3 Experiments . . . 89
6.4 Evaluation . . . 90
6.4.1 Model . . . 90
6.4.2 Compression . . . 91
6.4.3 Replacement Methods . . . 95
6.5 Summary . . . 101
6.6 Code and Data Availability . . . 102

7 Data Coding and Residual Calculation 103
7.1 Motivation . . . 104
7.2 Proposed Method . . . 106
7.2.1 Shifted XOR . . . 107
7.2.2 Splitting of the Residual . . . 108
7.2.3 Coding of LZC/FOC . . . 109
7.3 Experimental Setup . . . 109
7.3.1 Data . . . 109
7.3.2 Metrics . . . 110
7.3.3 Experiments . . . 110
7.4 Evaluation . . . 111
7.4.1 State-Of-The-Art Compression Algorithms . . . 111
7.4.2 Shifted XOR . . . 112
7.4.3 Splitting of the Residual . . . 115
7.4.4 Performance of Coding Methods . . . 115
7.4.5 Comparison of `pzip` and `fpzip` . . . 118
7.5 Summary . . . 120
7.6 Code and Data Availability . . . 121

8 Compression Framework 123
8.1 Motivation . . . 123
8.2 Proposed Method . . . 123
8.3 Implementation . . . 127
8.4 Summary . . . 129
8.5 Code and Data Availability . . . 129

9 Conclusions and Outlook **131**
9.1 Conclusions 131
9.2 Outlook 132

A Variance Analysis **137**

B Pascal Predictor **145**

Bibliography **149**

List of Figures

1.1 Outline of the Thesis 8

2.1 Compression in Three Steps 14
2.2 Example for Row-Major Traversal Sequence 18
2.3 Sources for Climate Data 23

3.1 Neighbourhoods Used in Lossless Image Compression 32
3.2 Memory Representation of Floating-Point Values 35

4.1 Zonal Band 46
4.2 Long-Term Variance Analysis on a Global Scale 48
4.3 Short-Term Variance Analysis for Temperature Across Time 50
4.4 Short-Term Variance Analysis for Temperature Across Longitude 51
4.5 Entropy Analysis 55
4.6 NMI Between Data Variables for Each Dataset 56

5.1 Adjacent Data Points for a d-Dimensional Object 62
5.2 IS and IC of Example 64
5.3 Four Different Traversal Path 65
5.4 Difference Plot of LZC for Two Different Starting Points . . . 70
5.5 Difference Plot of LZC for Traversal Orders 72
5.6 The Maximum Reached LZC for Each Variable 74
5.7 LZC and SD of Predictors with and without IS 76
5.8 LZC per Variable Using the Best Traversal and Prediction Method 79

6.1 Flowchart of Analysis 82
6.2 Histogram of Each Weather Index 88

6.3 NAO and QBO Indices and Their Reconstruction 92
6.4 Pearson Correlation with 10% Replacement 97
6.5 Pearson Correlation r_1, t for NAO and QBO30 98

7.1 Flowchart of the Residual Calculation and Coding Phase . . 106
7.2 An Example for the XOR Residual Calculation Method 107
7.3 Average CF and Throughput 112
7.4 LZC for Datasets with Gaussian Distribution 114
7.5 Distribution of set/unset Bits 116
7.6 Time Spend in Each Step of the Compression Algorithm . . . 118

8.1 State Diagram of a Prediction-Based Compression Algorithm 124
8.2 UML Class Diagram for Object Components 128
8.3 A None Exhaustive List of Modifiers 129

A.1 Short-Term Variance Analysis for Specific Humidity Across Time 138
A.2 Short-Term Variance Analysis for Specific Humidity Across Longitude 139
A.3 Short-Term Variance Analysis for Meridional Wind Across Latitude 140
A.4 Short-Term Variance Analysis for Meridional Wind Across Altitude 141
A.5 Short-Term Variance Analysis for Zonal Wind Across Time . 142
A.6 Short-Term Variance Analysis for Zonal Wind Across Longitude 143

List of Tables

3.1 JPEG Predictors 31

5.1 Variables Used for Testing of IS 67
5.2 LZC and SD Across Predictors with Varying Starting Points . 69
5.3 LZC and SD Across Predictors Using Linear Traversal 73
5.4 Changes on Average Due to the Application of IS 75
5.5 Ranking of Individual Predictors 75
5.6 LZC Comparison of Consolidation Methods 77
5.7 SD Comparison of Consolidation Methods 78
5.8 Highest Achieved CR per Variable 79

6.1 Spatial Borders and Variables 87
6.2 Information About the Monthly Indices 89
6.3 Results of (Seasonal) ARIMA Model 90
6.4 Results of DFT 91
6.5 BPF of Lossless Compression of Daily and Monthly Data . . 91
6.6 CR for Lossy Compression of Daily and Monthly Data 93
6.7 Pearson Correlation by Replacing 5 and 10% of the Monthly Indices 96
6.8 CR for NAO and QBO30 100
6.9 Correlation Coefficient for `zfp06+03` for Daily and Monthly Data 100

7.1 Effects of Using a Shifted XOR with Lorenz and Last Value Predictor 113
7.2 File Size of Temperature Data After Transformation and Coding Schemes. 117
7.3 CF and Throughput [MiB/s] of Climate Simulation Data . . . 119

B.1 Coefficients for Pascal **K** Predictor 145

List of Algorithms

3.1 Prediction Step of JPEG-LS Image Compression Algorithm . 31
8.1 Compression Function of Proposed Framework 125
8.2 An Ensemble Compression Algorithm 126

List of Abbreviations

AC auto-correlation.

ACF auto-correlation function.

AIC Akaike's Information Criterion.

APAX APplication AXceleration.

AR auto-regressive.

ARIMA Auto Regressive Integrated Moving Average.

ASCII American Standard Code for Information Interchange.

BPF bits per float.

BWT Burrow-Wheeler-Transform.

CDF cumulative distribution function.

CE computational efficiency.

CF compression factor.

CPU Central Processing Unit.

CR compression ratio.

DChT discrete Chebyshev transform.

DCT discrete cosine transform.

DF-test Dickey-Fuller-Test.

DLT discrete Legendre transform.

EEG Electroencephalography.

EMAC ECMWF Hamburg/Modular Earth Submodel System Atmospheric Chemistry.

ENSO El Niño Southern Oscillation.

ENSO34 El Niño Southern Oscillation 3.4.

FIR finite impulse response filter.

FLAC free lossless audio compression.

FOC following one count.

FTIR Fourier-Transform Infrared Spectrometers.

GPU Graphical Processing Unit.

GRIB General Regularly-distributed Information in Binary form.

HDF Hierarchical Data Format.

HPC High Performance Computing.

IC information context.

ICON ICOsahedral Nonhydrostatic.

IID Independent and Identically Distributed.

IS information space.

ISABELA Sort-And-B-spline Error-bounded Lossy Abatement.

JPEG Joint Photographic Experts Group.

LPC linear predictive coding.

LSTM long short-term memory.

LZ Lempel Ziv.

LZC leading zero count.

LZMA Lempel–Ziv–Markov chain Algorithm.

MA moving-average.

NAO North Atlantic Oscillation.

NetCDF Network Common Data Format.

PMF probability mass function.

POLSTRACC POLar STRAtosphere in a Changing Climate.

PSNR peak signal to noise ratio.

QBO Quasi-Biennial Oscillation.

RE range encoding.

RGB red, green, and blue.

RLE run-length encoding.

RMSD Root Mean Square Deviation.

SD standard deviation.

SSD Solid State Disk.

SVD singular value decomposition.

SZ Squeeze.

XOR eXclusive OR.

ZFP zip floating-point.

CHAPTER 1

Introduction

The development of GPU clusters, high bandwidths within and between compute nodes, faster CPUs using multi-core architecture and the introduction of SSDs have drastically reduced the computation time for simulations on HPC clusters. In addition to these developments in raw computing power, a better utilisation of hardware nodes is achieved through optimised software libraries for parallel and distributed computing. While the computing power used to be the bottleneck for climate simulations, today it is typically the memory bandwidth and the storage space required for simulation output. The climate sciences are severely affected by this bottleneck.

New climate models allow model integration at unprecedented resolution, simulating decades and centuries of climate change, including complex interactions in the Earth system under different scenarios. The number of variables stored for analysis is minimised to keep the simulation output small and manageable. Further, if the model assessment requires more information, often simulations have to be rerun to calculate the requested variables. Another applied solution is to reduce resolution. The available storage space often forces scientists to reduce the temporal resolution of their output and to use interpolation for missing time steps. The generated output then becomes an inferior representation of the actual model used for simulation. One method to tackle this problem is to apply compression.

Although compression algorithms can differ in their specifics, the basic principle is always the same: Data compression is achieved by removing redundant information in the data. Removing redundancy allows a smaller representation of the data without information loss. Both of the applied solutions would not be necessary any more.

First, applying compression enables the researcher to increase the temporal resolution of the simulation output, since more information can be saved on the same storage space as before. Therefore, no interpolation meth-

ods are for missing time steps necessary. Even if the desired temporal resolution can not be achieved and interpolation methods still need to be used, the results of the interpolation methods will be better than before since more data points can be used for interpolation.

Second, compression enables saving more variables per simulation run. This prevents possible simulation reruns, since more variables can be saved with the initial run of the simulation. This decreases the CO_2 footprint of the HPC cluster since unnecessary simulation runs are avoided. However, there are challenges involved in developing novel compression algorithms.

Challenge 1

Understanding the structure and intrinsics of the data.

The first step is to analyse the structure and intrinsics of the data. Knowledge must be gained about what is represented in the data. Possible data sources must be identified and analysed for differences and information content. The most common data variables, their data types, and their value ranges must be identified. This information will help relate one data point to any other data point. In order to do this it will help identify related research fields and recommended solutions from the literature.

Challenge 2

Analysing available compression techniques for strengths and weaknesses.

The next step is to find out which compression techniques are most promising for the data. For this purpose, an extensive and in-depth analysis of the available literature must be carried out. With an ever deeper knowledge, compression techniques for related data can be analysed, that are similar but not identical to the available data.

Challenge 3

Integrating existing knowledge about the interactions of variables.

If knowledge about the data already exists, it can be used to identify redundant information. To do this, the first step is to find out which knowledge about the variables makes sense to integrate. Then it has to be decided in which phase of the compression process this knowledge can be integrated. Finally, a suitable form of representation must be chosen with which this knowledge can be integrated into the algorithm.

Challenge 4

Identifying new patterns and relationships within and between variables for the current data.

Previous knowledge can only help to a certain extent. The compression algorithm must adapt to the data to be compressed. This process can be accelerated by analysing in advance which types of relationships occur most frequently and contain the most information within and between the data variables. Finally, a method must be developed to quickly identify these types of relationships in the current data.

Challenge 5

Building a framework to perform rapid testing of new compression algorithms.

Developing a compression algorithm is an iterative process. There is a lot of fine-tuning involved. If parts of the compression algorithm can be replaced quickly, iteration steps can be accelerated. For this purpose, the common components of a compression algorithm must first be identified. Then a concept for a modular structure has to be worked out. Following this, the interfaces can be developed by analysing which of the algorithm steps offer the greatest opportunity for improvement. Finally, concepts to accelerate testing must be developed. These can be parallel processing or the (random) selection of data areas for tests. Such an environment can reduce the development time of an algorithm.

1.1 Contributions to the Research Field

Overall, the contributions of this thesis focus on developing novel compression techniques for structured floating-point data. This research was initiated because of the needs of climate science, but the application of its contributions is not limited to them. The results of this thesis can be used to develop and improve any compression algorithm for structured floating-point data. Each contribution is discussed in more detail in the following.

Contribution 1

Analysis methods for identification of redundant information in data

The first contribution is about analysis methods for the discovery of redundant information in data. In the following, it is shown how analysis methods from statistics, entropy analysis and information theory can be used for the recognition of redundant information between and within variables. The presented analyses can be applied to one-dimensional and multidimensional data. This contribution is associated with Challenge 1.

Contribution 2

Comparison of lossless compression algorithms for structured floating-point data

The next contribution is an in-depth study of lossless compression algorithms for floating-point data. The compression factor and throughput of these algorithms is investigated. For this purpose, general-purpose algorithms as well as custom compression algorithms for floating-point data are used. This contribution makes it possible to test the compression algorithms developed in this thesis against state-of-the-art algorithms. This contribution has been published in Cayoglu et al. (2019b) and is associated with Challenge 2.

Contribution 3

Novel data coding scheme for prediction-based lossless compression methods

Third, a novel data coding scheme is developed. During the analysis of compression algorithms, a weakness in the coding of the data is identified. Extensive analysis of this weakness shows that by applying a shift operation to move the data into a more suitable value range it is possible to improve the compression factor. In addition, this method allows the use of computationally less intensive algorithms to increase throughput without affecting the achieved compression factor. This contribution has been published in Cayoglu et al. (2019b) and is associated with Challenge 2.

Contribution 4

Development of a novel lossy compression algorithm for time-series data

There are various established climate indices, which can predict the development of data variables. These indices are time-series data and can help identify redundant information in temperature or precipitation. However, these time-series data must be available to the encoder. The fourth contribution of this thesis is a novel lossy compression algorithm for time-series data. By applying statistical models and improving the precision of individual data points, the quality of the reconstructed data is increased without significantly increasing the required storage space. This contribution has been published in Cayoglu et al. (2017) and is associated with Challenge 3.

Contribution 5

Introduction of Information Spaces to use information across all dimensions for data prediction

Analyses of climate data suggest that information from all dimensions are important for predicting variables. However, the importance of each dimension depends on the time and location of the data point on Earth. Therefore, it is necessary to have an algorithm that adapts the context on which a prediction is based on to previous successful or erroneous predictions. The concept of information spaces (IS) and contexts presented in this thesis serves this purpose. Based on the quality of earlier predictions, the information used for the next prediction is adjusted. Further, IS can use and adapt prior knowledge about the data variables to achieve a better prediction and compression factor. Using this concept it is possible to increase the compression factor and, more importantly, to reduce dependence of the compression factor on the structure of the data. This contribution has been published in Cayoglu et al. (2019a, 2018c,a) and is associated with Challenge 1 and 4.

Contribution 6

A modular framework for testing and quality assessment of compression algorithms

The final contribution is a modular framework for the development of custom compression algorithms. This framework allows the creation of custom prediction-based compression algorithms with support for ensemble predictors, quality assessment, parallel execution with random subsetting of

multi-dimensional data. Further, it defines an interface for each module to help the user extent the framework. This contribution has been published in Cayoglu et al. (2018b,c) and is associated with Challenge 5.

List of Publications. Most of the contributions discussed above have been published in peer-reviewed conferences and workshops. In the following the corresponding publications are listed:

- Cayoglu, U., Braesicke, P., Kerzenmacher, T., Meyer, J., and Streit, A. (2017). Adaptive Lossy Compression of Complex Environmental Indices Using Seasonal Auto-Regressive Integrated Moving Average Models. In **2017 IEEE 13th International Conference on e-Science (e-Science)**, pages 315–324. DOI: 10.1109/eScience.2017.45. [best paper award]
- Cayoglu, U., Schröter, J., Meyer, J., Streit, A., and Braesicke, P. (2018b). A Modular Software Framework for Compression of Structured Climate Data. In **Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems**, SIGSPATIAL '18, pages 556–559, New York, NY, USA. ACM, ISBN: 978-1-4503-5889-7, DOI: 10.1145/3274895.3274897
- Cayoglu, U., Tristram, F., Meyer, J., Kerzenmacher, T., Braesicke, P., and Streit, A. (2018c). Concept and Analysis of Information Spaces to improve Prediction-Based Compression. In **2018 IEEE International Conference on Big Data (Big Data)**, pages 3392–3401. DOI: 10.1109/BigData.2018.8622313
- Cayoglu, U., Tristram, F., Meyer, J., Kerzenmacher, T., Braesicke, P., and Streit, A. (2019a). On Advancement of Information Spaces to Improve Prediction-Based Compression. In David, K., Geihs, K., Lange, M., and Stumme, G., editors, **INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik – Informatik für Gesellschaft**, pages 271–272, Bonn. Gesellschaft für Informatik e.V., ISBN: 978-3-88579-688-6, ISSN: 1617-5468, DOI: 10.18420/inf2019_39
- Cayoglu, U., Tristram, F., Meyer, J., Schröter, J., Kerzenmacher, T., Braesicke, P., and Streit, A. (2019b). Data Encoding in Lossless Prediction-Based Compression Algorithms. In **IEEE 15th International Conference on e-Science (e-Science)**. ISBN: 978-1-7281-2451-3, DOI: 10.1109/eScience.2019.00032

- Cayoglu, U., Braesicke, P., Kerzenmacher, T., Meyer, J., and Streit, A. (2018a). Towards an optimised environmental data compression method for structured model output. In **EGU General Assembly Conference Abstracts**, volume 20, page 8609. https://meetingorganizer.copernicus.org/EGU2018/EGU2018-8609.pdf
- Kerzenmacher, T., Cayoglu, U., Kellmann, S., Kirner, O., Versick, S., Wang, S., and Braesicke, P. (2018). QBO influence on the ozone distribution in the extra-tropical stratosphere. In **EGU General Assembly Conference Abstracts**, volume 20, page 16565. https://meetingorganizer.copernicus.org/EGU2018/EGU2018-16565.pdf

Code and Data Availability. Data and corresponding implementations of all methods introduced in this thesis and previously published articles are available under GNU GPLv3 license at the following addresses:

- https://github.com/ucyo/adaptive-lossy-compression (Cayoglu et al., 2017)
- https://github.com/ucyo/informationspaces (Cayoglu et al., 2019a, 2018c)
- https://github.com/ucyo/cframework (Cayoglu et al., 2018b,c)
- https://github.com/ucyo/xor-and-residual-calculation (Cayoglu et al., 2019b)
- https://github.com/ucyo/climate-data-analysis (Cayoglu, 2019a)

1.2 Outline of the Thesis

The remainder of this thesis is structured as follows (see Fig. 1.1):

Chapter 2 provides the necessary background for the thesis. The basic principles and structure of compression algorithms are introduced, methods for classification presented, and the relationship between compression and information theory explained. Further, prediction-based compression is explained in detail and several metrics for the evaluation of compression methods are introduced. The chapter concludes with an introduction to climate data and their different sources with a deep dive in simulation output.

Chapter 3 goes on to discuss related work. The chapter is divided into three parts. First, related work regarding the compression of climate data is introduced. Next, related work in image, time-series and floating-point

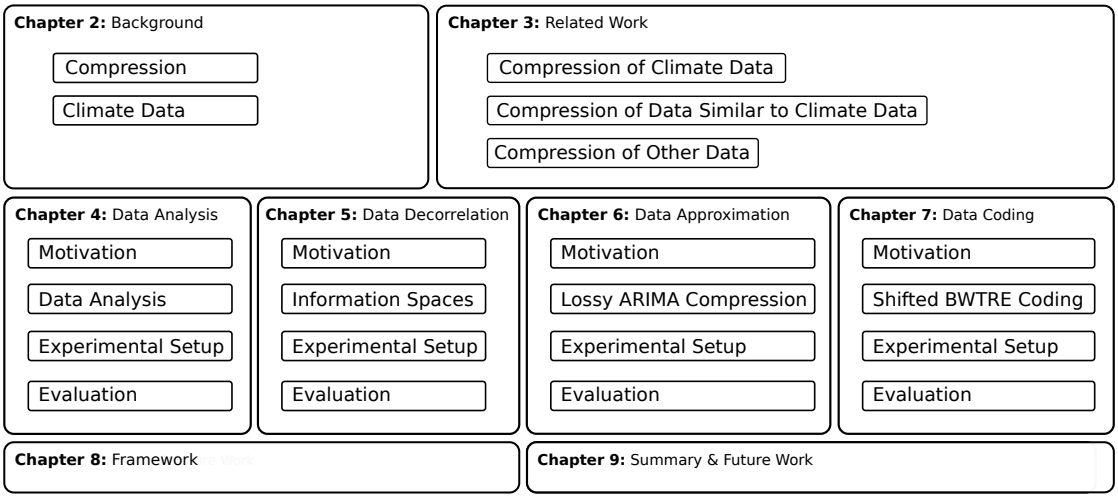


FIGURE 1.1 Outline of the thesis

compression is described, since these compression algorithms deal with data similar to climate data. Finally, a short summary of the remaining field of compression is given.

Chapter 4 provides a critical analysis of climate data to identify redundant information. Several analyses are conducted: Long-term and short-term variance analysis, Shannon and Sample entropy analysis as well as Mutual Information analysis.

Chapter 5 builds upon the results of the previous chapter and introduces the concept of information spaces and contexts. Since several contexts are used, methods for the consolidation of predictions are investigated and alternative traversal methods are recommended and evaluated to optimise the information context.

Chapter 6 introduces the aforementioned lossy compression method for time-series data. The chapter starts with an introduction of the Auto Regressive Integrated Moving Average (ARIMA) model used to identify redundant information. After an introduction of the environmental indices, the replacement methods are explained, since the algorithm must decide autonomously which data points to save with higher precision. The chapter concludes with an evaluation of the proposed method.

Chapter 7 introduces the novel coding approach discussed in the third contribution. After an introduction to established residual calculation algorithms, the weakness of the XOR residual calculation is explained. Next, the shift calculation is presented to overcome the weakness. This is followed

by an explanation of the transformation and coding steps applied to the residual. Finally putting it all together, state-of-the-art compression algorithms are compared regarding compression factor and throughput, the coding performance evaluated and the proposed compression algorithm with state-of-the-art algorithms compared.

In Chapter 8 the proposed compression framework from the sixth contribution is presented. The internal structure and the implemented modules are described. Afterwards, more details about the libraries used and possibilities for future extensions are explained.

In Chapter 9 a summary of the thesis is given as well as a an overview of open questions and the possibilities for future work are presented.

CHAPTER 2

Background

This chapter gives an overview of the basic components of data compression and its history in information theory. The goal of this chapter is to build a common understanding of data compression and climate data. The chapter starts with an introduction to common terminology and notation. Afterwards, compression algorithms are classified based on their characteristics and application field. Following this overview of different compression algorithms, one specific compression algorithm, prediction-based compression, is introduced in more detail. The introduction to data compression concludes with an overview of metrics that can be used to assess the quality of a compression algorithm. Since the compression algorithms introduced and developed in this thesis are applied to climate data, an overview about the structure and properties of climate data are presented in the concluding section.

2.1 Data Compression

Data compression was originally developed to reduce transportation time of data from a source location S to a predefined destination D. This is the reason why data compression is also referred to as **source coding**. The reduction in transportation time was mostly achieved by reducing the size of the data. The following definitions are based on Salomon and Motta (2010).

DEFINITION 2.1 (Source) A source S of data items can be a file stored on a disk, a file that is input from outside the computer, text input from a keyboard, or a program that generates data symbols to be compressed or processed in some way. A source is always ordered and might therefore be

called a **sequence**.

$$\begin{aligned} S &= s_1 s_2 s_3 \dots s_{n-1} s_n \\ &= (s_i)_{i=1}^{n} \text{ with } i \in \mathbb{N}^* \\ |S| &= n \end{aligned} \tag{2.1}$$

with $|S|$ representing the size of S. A subsequence of S can be expressed using S_m^k notation

$$\begin{aligned} S_m^k &= s_m s_{m+1} s_{m+2} \dots s_{k-1} s_k \\ &= (s_i)_{i=m}^{k} \text{ with } k \geq m \end{aligned} \tag{2.2}$$

In a **memoryless source**, the probability of occurrence of a data symbol does not depend on its context. Such a source is termed Independent and Identically Distributed (IID). The source consists of a set of data symbols which is called its Alphabet X.

DEFINITION 2.2 (Alphabet) An alphabet defines the set of symbols from which a source S can draw its symbols from.

$$\begin{aligned} X &= \{x_1, x_2, x_3, \dots, x_m\} \text{ with } m \in \mathbb{N}^* \\ |X| &= m \end{aligned} \tag{2.3}$$

with $|X|$ representing the size of X. An alphabet may consist of 128 ASCII codes (text data), two bits, or any other set of symbols.

A compression algorithm consists of two transformation processes. The first process transforms the data from source S to destination D. This process is called **encoding** or **compression** and is performed by the **encoder**. The data being encoded is called **raw**, **original**, or **unencoded** data, while the data after the encoding process is called **encoded** or **compressed** data. The second process reverses the first transformation. This process transforms data from D back to S. This process is called **decoding** or **decompression** and is performed by the **decoder**. The data after the decoding process is called **reconstructed** data.

While compression algorithms can differ in their specifics, the basic principle is always the same: Data compression is achieved by removing **redundancy** also called **structure**, or **correlation** in the original data. Any

non-random data has some structure, and this structure can be exploited to achieve a smaller representation of the data, a representation where no structure is discernible (Salomon and Motta, 2010).

2.1.1 Classification of Compression Algorithms

There are two main types of compression algorithms: lossless and lossy. **Lossless** compression algorithms encode the original data in such a way, that the reconstructed data is identical to the original data. Lossless compression is performed by decorrelation and subsequent coding of the data. The decorrelation step eliminates redundant information in the data. During coding, the actual compression takes place and the data is written to disk. **Lossy** compression algorithms aim to achieve smaller file sizes for encoded data by removing or approximating unimportant information. This process is called **approximation.** The approximation step reduces the complexity of the data for the subsequent coding, e.g. by using quantification. Approximation and is not applied in lossless compression. An overview of the steps is depicted in Figure 2.1. Some additional classifications for compression algorithms are the following (Salomon and Motta, 2010):

- **Non-adaptive vs adaptive**
  A compression algorithm which modifies its operations and parameters based on the original data is called an **adaptive** compression algorithm. An algorithm which does not do this is called a **non-adaptive** compression algorithm.
- **Symmetric vs asymmetric**
  The data processing of the encoder and decoder might be different depending on the compression algorithm. In a **symmetric** compression algorithm the encoder and decoder apply the same algorithm in different direction. In an **asymmetric** compression algorithm one of the two parts has a heavier workload.
- **Single-pass vs multi-pass**
  A **single-pass** algorithm traverses the data only once, while a **multi-pass** or n**-pass** algorithm traverses the data several (n) times. Statistical compression algorithms are often multi-pass algorithms, since the first pass is used to gather statistical information about the data like distribution and skewness.

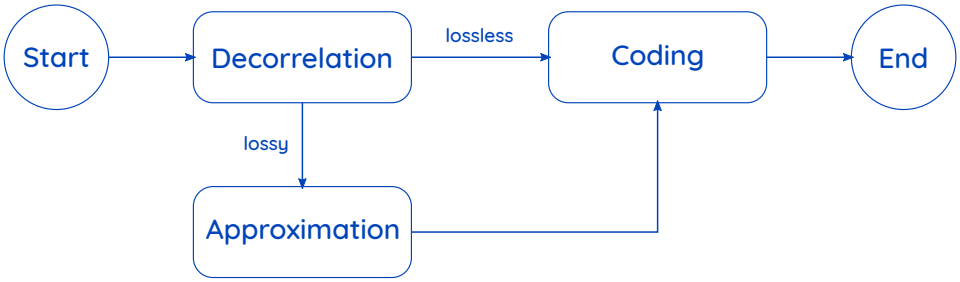


FIGURE 2.1 A compression algorithm is defined by three steps: decorrelation, approximation and coding. A lossless compression algorithm uses decorrelation and coding. A lossy compression algorithm has an approximation step in between. This figure is based on Cappello and Lindstrom (2018).

- **Streaming vs block mode**
  A compression algorithm working in **streaming mode** reads a symbol from the source, encodes it and moves on to the next symbol until the source is exhausted. A **block mode** compression algorithm reads the source in several blocks of n symbols and encodes each block separately.
- **Universal vs custom**
  A **universal** compression algorithm has no statistical information about the original data. A **custom**, **specific** or **non-universal** compression algorithm has a predefined knowledge about the data to be compressed. A universal method is **optimal** if the compressor can produce compression ratios (see Section 2.1.5) which asymptotically approaches the entropy of the input stream for long inputs.

EXAMPLE: SYMMETRIC VS. ASYMMETRIC

Archiving is a good example for an asymmetric compression algorithm. The encoder has often the heavier workload, so that it takes significantly more time to encode the data. The reason for this imbalance is the assumption, that the files will be read more often than written in an archive. Therefore, the decompression should be fast.

2.1.2 Data Compression and Information Theory

Claude Shannon single-handedly created the research field of information theory in the late 1940s. He was interested in creating a reliable and fast communication channel from a source to a receiver. Back in the days the

communication channels were often affected by faulty hardware (e.g. bad wiring, isolation problems) or in case of audio communication actual noise in the background. Therefore, he worked on special codes (e.g. error-control codes) to reliably communicate over a noisy channel. Further, he tried to reduce the amount of data to be transferred over the communication channel. To achieve this, Shannon (1948) created a metric to measure the information content of each symbol of the message to be transmitted called the Shannon Entropy $\mathrm{H}(X)$,

$$\mathrm{H}(X) = -\sum_{x_i \in X} P(x_i) \cdot \log_b(P(x_i)) \tag{2.4}$$

where $X = \{x_1, x_2, \ldots, x_n\}$ is the alphabet of the source, $P(\cdot)$ the probability mass function (PMF), and b the base of the logarithm. Common values for b are 2, Euler e, and 10. The unit of $H(X)$ depends on b. For those mentioned above the corresponding units are **bits** for $b = 2$, **nats** for $b = e$, and **bans** for $b = 10$. Unless otherwise stated, $b = 2$ is assumed in the remainder of the thesis, since the Shannon Entropy with $b = 2$ is considered the lower limit for bits per float (BPF) in lossless compression algorithms for floating-point numbers.

One advantage (and disadvantage) of the Shannon Entropy is that no further information other than the PMF will be considered for calculating the information content of a symbol. This may lead to information about fluctuation or the regularity of a dataset (e.g. in time-series data) to be disregarded. In cases where more information about the dataset is available (e.g. temporal interdependency) other entropy metrics such as the Sample Entropy might be more accurate as to the information content of each symbol.

Sample Entropy (Richman and Moorman, 2000) is an approximate entropy established in the field of neurology to analyse brain wave data. Given a source $S = s_1 s_2 \ldots s_{n-1} s_n$ and a subset $S_m^{m+l} = s_m s_{m+1} \ldots s_{m+l-1}, s_{m+l}$, the Sample Entropy is defined as follows:

$$\begin{aligned} \mathrm{SampEn}(X, l, \tau) &= -\log_2 \frac{Z(X, l+1, \tau)}{Z(X, l, \tau)} \\ Z(X, l, \tau) &= \sum_i \sum_j [d(S_i^{i+l}, S_j^{j+l}) < \tau] \quad \forall i \neq j \end{aligned} \tag{2.5}$$

with $d(\cdot)$ being a distance function of two datasets (i.e. vectors) like the Euclidean or Chebyshev distance (Chebyshev, 1853), τ a threshold value for the distance, and $[\,\cdot\,]$ defined as follows:

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{else} \end{cases}$$

In literature the values $l = 2$ and $\tau = 0.2 \cdot \sigma(X)$ with $\sigma(\cdot)$ representing the standard deviation, have established themselves.

EXAMPLE: SHANNON ENTROPY VS. SAMPLE ENTROPY

Given a source S_1 with alphabet X_1 and a source S_2 with alphabet X_2

$$S_1 = (10, 20, 10, 20, 10, 20, 10, 20, 10, 20, 10, 20) \text{ with } X_1 = \{10, 20\}$$
$$S_2 = (10, 10, 20, 10, 10, 20, 20, 20, 10, 10, 20, 20) \text{ with } X_2 = \{10, 20\}$$

the resulting entropies based on Eq. 2.4 and 2.5 are

$$\mathrm{H}(X_1) = 1.0 \qquad \mathrm{SampEn}(S_1, 2, \sigma(X_1)) = 0.223$$
$$\mathrm{H}(X_2) = 1.0 \qquad \mathrm{SampEn}(S_2, 2, \sigma(X_2)) = 0.693$$

Since the fluctuation in S_1 is rather regular than pure noise the $\mathrm{SampEn}(X)$ returns a lower entropy for S_1 than for S_2. $\mathrm{H}(X)$ returns the same entropy for both sources since no information other than the PMF is considered.

2.1.3 Application of Custom Compression Algorithms

One of the classification criteria for compression algorithms explained in the previous section is the type of customisation: **universal** and **custom**. For subsequent chapters it is helpful to understand in which aspects algorithms are customised. **Custom** compression algorithms usually take into account structural dependencies in the data. This might be previous or following frames in a video format or neighbouring pixels in an image file. These dependencies are represented in the structure of the file and can be taken advantage of to discover redundancies in the data. There are several custom compression algorithms for video, audio (e.g. wavelet, transform-

based), image (e.g. partial pattern matching), text (e.g. dictionary-based), and numerical data (e.g. prediction-based). Most of these algorithms use **probability models** to figure out what the most probable symbol might be next in the data. This is the reason why most of the research in the field of compression can be split into two groups. The first group works on improving the probability model for a given application field (e.g. video, numerical data, or audio). The second group works on improving the coding process where the actual compression of the data happens. Novel approaches in both of these fields are discussed in Chapter 6, 5 and 7.

2.1.4 Prediction-Based Compression

One often customised algorithm for floating-point data is the prediction-based compression algorithm. This section describes the steps involved in a prediction-based compression algorithm. A prediction-based compression algorithm involves five steps (Cayoglu et al., 2018c):

1) Mapping the floating-point data to integer values
2) Choosing a traversal sequence
3) Giving a prediction for each value
4) Calculating the residual between prediction and true value
5) Coding the residuals and writing them on disk

The five steps are now discussed in more detail:

1. Mapping the Floating-Point Data to Integer Values. The mapping of floating-point data to positive integer values is necessary to use integer operations. Using integer operations guarantees reproducibility across computer architectures, since floating-point operations are vulnerable to rounding errors which may lead to numerical imprecision.

$$m : \mathbb{R} \to \mathbb{N} \tag{2.6}$$

2. Choosing a Traversal Sequence. In prediction-based compression each data point is first given a prediction and later compared with the actual value. This is an iterative process and each prediction is based on data points seen in the past. The traversal function t is a permutation of the original source

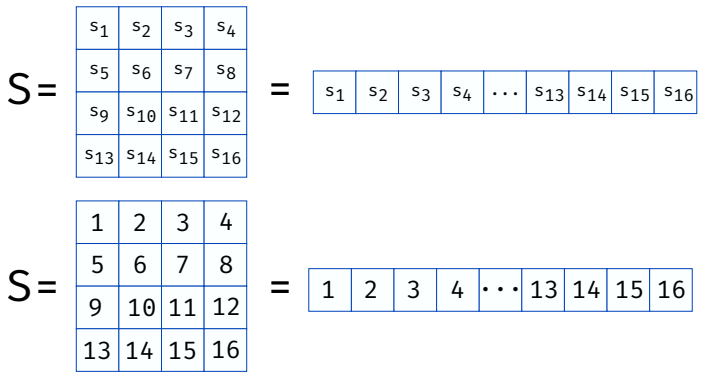


FIGURE 2.2 Example for row-major traversal sequence.

and defines the sequence in which each data point is visited.

$$t : \mathbb{N} \to \mathbb{N} \tag{2.7}$$

The resulting traversal sequence S' is defined as follows:

$$S' = (s_{t(i)})_{i=1}^{|S|} \tag{2.8}$$

A traversal function can be considered as a space-filling curve introduced by Peano (1890). The traversal sequence is important for designing a good prediction-based compression algorithm since it defines the available information for making a prediction. Non-universal compression algorithms can adapt their traversal sequence to fully exploit the structure of the data to decorrelate it better than universal compression algorithms.

EXAMPLE: FROM ROW-MAJOR TO COLUMN-MAJOR TRAVERSAL

Given a source $S = (s_1 s_2 \dots s_{15} s_{16})$ with $s_i = i$ representing a 4×4 grid (see Figure 2.2) and the traversal function

$$t(x) = 4 \cdot (x \bmod 4) + \left\lfloor \frac{x}{4} \right\rfloor$$

the resulting sequence is $S' = (s_1 s_5 s_9 s_{13} s_2 s_6 s_{10} s_{14} s_3 s_7 s_{11} s_{15} s_4 s_8 s_{12} s_{16})$. This traversal function transformed a row-major dataset into a column-major dataset. This traversal might be more successful in decorrelating S if there is a higher correlation between the columns of the dataset rather than the rows.

3. Giving a Prediction for Each Value. In the next step each data point is visited according to the traversal sequence. At each visit a **prediction method** or **predictor** p_k gives a prediction $\hat{s}_i \in \hat{S}$ for each symbol $s \in S$ using the data points of $S' \to \hat{S}$ with $p_k(S') = (\hat{s}_i)_{i=1}^{|S'|}$.

$$p_k(S') = \begin{cases} \sum_j^k s'_{i-j} \cdot c_j & \forall i \geq k \wedge s'_i \in S' \\ pb_k(S') & \text{else} \end{cases} \tag{2.9}$$

where $p_k(\cdot)$ is the prediction function using k elements of the sequence S', pb_k a prediction function for the border cases where $|S'| < k$, c_j the weight coefficient of element s'_{i-j}.

The above notation focuses on the traversal sequence and can become confusing very quickly:

$$S \xrightarrow{t(i)} S' \xrightarrow{p_k(s'_i)} \hat{S} \tag{2.10}$$

Since climate data is represented using a four dimensional cube i.e. tesseract, any prediction function $p_k(\cdot)$ can be expressed using the relative position of each data point. In the following the notation for a $M \times N$ matrix is described, but it can be extended to any other multi-dimensional data like a cube (three dimensions) or tesseract (four dimensions).

Given a matrix A of size $M \times N$, each element of A can be identified using $A(i, j)$ with $0 \leq i < M$ and $0 \leq j < N$. The prediction method $p : \{A, \vec{c}\} \mapsto \hat{A}$ can then be defined as follows:

$$p(A, \vec{c}) = \begin{cases} \sum\limits_{c_{i,j} \in \vec{c}} A(x - i, y - j) \cdot c_{i,j} & x \in M \wedge \geq i, y \in N \wedge \geq j \\ p_b(A, \vec{c}) & \text{else} \end{cases} \tag{2.11}$$

with $p(\cdot)$ representing the prediction function, $c_{i,j} \in \vec{c}$ the weight coefficient of element $A(x - i, y - j)$ and $pb(\cdot)$ the prediction function for the border cases if $A(x-i, y-j)$ is not available. The notation can be further shortened by expressing the relative position of cells using $\rho_{i,j}$ as follows:

$$\rho_{i,j} := A(x - i, y - j) \quad \forall x \in M \wedge \forall y \in N \tag{2.12}$$

$$\begin{aligned} p &= \vec{\rho} \cdot \vec{c} \\ &= \sum_{i,j} \rho_{i,j} \cdot c_{i,j} \end{aligned} \tag{2.13}$$

where $\rho_{i,j}$ defines the relative position of the data point $A(x-i, y-j)$ and $c_{i,j}$ the weight coefficient for that particular data point.

4. Calculating the Residual Between Prediction and True Value. The **difference method** $\text{diff}(\cdot)$ defines how the difference between the prediction $\hat{s}_i$ and the true value s_i is calculated. The difference itself is called the **residual**. In most cases this will be either the **absolute difference** using diff_{abs} or **eXclusive OR (XOR)** difference using diff_{xor} defined as follows:

$$\text{diff}_{xor} = \hat{s}_i \oplus s_i \tag{2.14}$$

$$\text{diff}_{abs} = |\, \hat{s}_i - s_i \,| \tag{2.15}$$

If the prediction is close to the true value, the leading zero count (LZC) of the residual will be high. LZC represents the number of most significant bits of a floating-point value that are zero i.e. unset. These bits do not need to be saved on disk, since they can be reproduced by the prediction algorithm. The remaining bits of the residual are then processed in the next step.

5. Coding the Residuals and Writing Them on Disk. In the last step the remainder of the residual is coded and written on disk. There are several options to choose from: Huffman coding (Huffman, 1952), Arithmetic coding (Howard and Vitter, 1994), Range coding (Martin, 1979), Golomb coding (Golomb, 1966), Asymmetric Numeral System coding (Duda, 2013) or any combination of these. More details regarding step four and five are given in Chapter 7.

In prediction-based compression there is a clear distinction between the decorrelation and coding phase depicted in Fig. 2.1. The goal of the decorrelation phase (steps one to three) is to use the available information in the best possible way to decorrelate the data and reduce the size of the residual. The task of the coding phase (steps four and five) is to write these residuals on disk in the most space-efficient way. If necessary, an approximation step can be added before calculating the residuals in step four or by a post-processing step using quantisation in step three.

2.1.5 Metrics

To assess the quality of a compression algorithm several metrics will be described in this section. The following metrics are commonly used: compression factor (CF), compression ratio (CR), throughput, and memory usage. The first two metrics are used if the size of the encoded data is important. They set the size of the raw and compressed data in relation.

$$\text{compression factor (CF)} = \frac{\text{Size of raw data}}{\text{Size of compressed data}} \tag{2.16}$$

A CF greater than 1 means the compression algorithm was successful in reducing the size of the original data. A value smaller than 1 means that the encoded data is actually larger than the raw data. The greater the compression factor, the better the compression algorithm.

$$\begin{aligned}\text{compression ratio (CR)} &= \frac{\text{Size of compressed data}}{\text{Size of raw data}} \\ &= \text{CF}^{-1}\end{aligned} \tag{2.17}$$

The CR is the inverse of the CF. The smaller the CR, the better the compression algorithm. A value close to 0 is optimal, while a value greater than 1 suggests that the compression algorithm failed and the size of the encoded data is actually larger than the raw data. It gives information about how a single bit in the encoded data is related to a single bit in the raw data.

The following two metrics are about compression speed and memory usage. The **throughput** indicates the amount of data processed per unit of time. In most cases this will be either Bytes/s or MiB/s.

$$\text{throughput} = \frac{\text{Filesize [Bytes or MiB]}}{\text{Total Processing Time [sec]}} \tag{2.18}$$

The higher the throughput, the faster the algorithm. The throughput is calculated separately for the encoder and decoder since the compression algorithm might be asymmetric. The **memory usage** is the maximum amount of memory used by the encoder (decoder) at any given time during the compression (decompression) process.

Most custom compression algorithms developed for a specific type of data also adjust their metrics to that data type. One such a metric is bits per float (BPF), which is derived from CR.

$$\text{BPF} = \frac{\text{Size of compressed data [bits]}}{\text{Number of floating-point values in data} \cdot bits} \tag{2.19}$$

with $bits = 32$ for single precision and $bits = 64$ for double precision floating-point data. BPF describes how many bits a single floating-point value occupies in a dataset after compression.

2.2 Climate Data

There are many different sources for climate data. However, these sources can be divided into two main groups: **observational data** and **simulation data**. The former can be split again into **ground-based remote sensing data** using measurement devices on the ground e.g. Fourier-Transform Infrared Spectrometers (FTIR), **remote sensing data** from devices attached to aircraft, satellites or balloons and **in-situ measurements** by direct contact (see Fig. 2.3). Observational data is often constrained in its spatial as well as temporal resolution due to hardware or software specifications. Further, there are more missing values in observational data compared to simulation data, since there are conditions where the sensors are not able to work properly (e.g. due to clouds or heavy rain). **Simulation data** are climate or weather data calculated by simulation models like ICOsahedral Nonhydrostatic (ICON) (Zängl et al., 2015; Schröter et al., 2018) or ECMWF Hamburg/Modular Earth Submodel System Atmospheric Chemistry (EMAC) (Jöckel et al., 2006). The rest of the thesis will deal exclusively with simulation data. It therefore makes sense to take a closer look at the weather and climate simulation models that generate these data.

2.2.1 Weather and Climate Models

Weather and climate models simulate the interactions in the Earth's atmosphere, ocean, cryosphere, and land surface. These models solve coupled differential equations describing central physical and chemical interactions on Earth, i.e. radiation (solar and terrestrial), advection, emissions, convec-

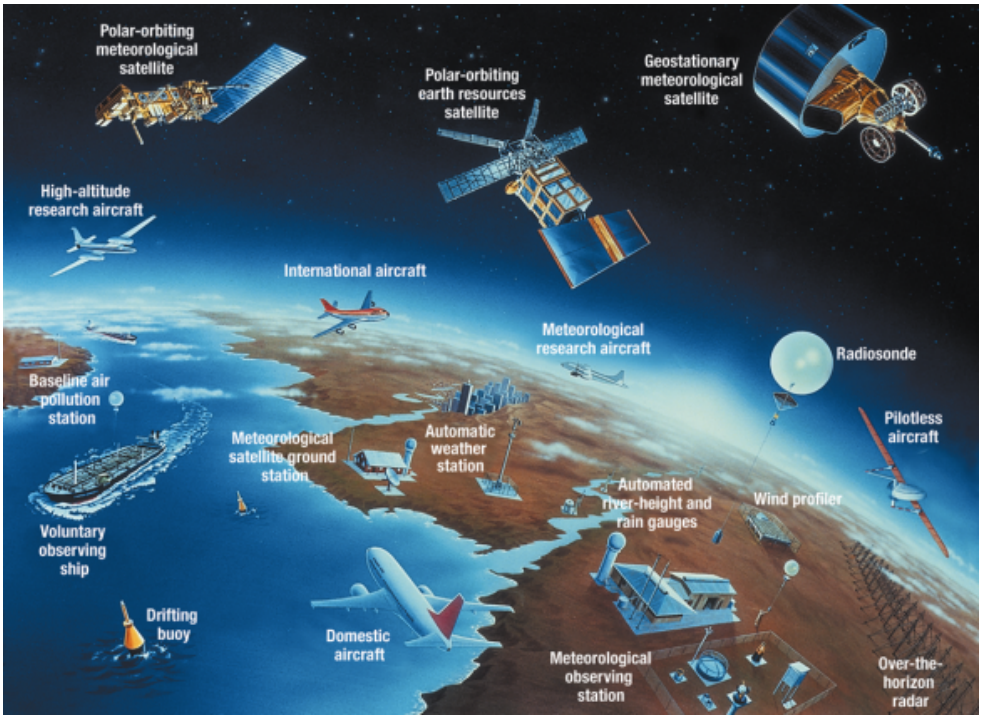


FIGURE 2.3 There are many different sources for climate data. However, these sources can be divided into two main groups: *observational data* and *simulation data*. This figure is taken from WMO (2019).

tion and condensation. Models simulate different **prognostic** and **diagnostic variables.** The variables described directly by the differential equations are called **prognostic variables.** They are integrated and calculated at each time step. Prognostic variables are e.g. temperature, wind, moisture, surface pressure, salinity (of the ocean), and water vapour. The **diagnostic variables** are evaluated from prognostic variables at specific time steps (set by the model parameters). One example for a diagnostic variable is humidity, which can be derived from temperature and water vapour.

These variables are calculated by dividing the Earth into a three-dimensional grid and solving the underlying differential equations and their interactions along the fourth dimension, time. While the differential equations solved are the same in weather and climate models, often the horizontal resolution, simulated area, and time scale is different. Weather models have often a higher horizontal resolution, but simulate at short time scales (from hours to couple weeks) and might be limited to local regions (e.g. Germany, or Europe). Climate models often simulate several decades at a global scale. These models are often run on a coarser horizontal resolution

than weather models due to the high compute power needed to solve the differential equations on a global scale for many years. Simulation models can further be distinguished based on their approach to solve the differential equations (i.e. spectral models using Fourier-based transformations or finite-difference models), their vertical consistency (i.e. hydrostatic or nonhydrostatic) as well as their grid model (i.e. structured models with a constant angular grid or unstructured models with a non-rectangular grid).

2.2.2 Data Structure of Climate Model Output

Due to historical reasons the most common data structure in climate data is the **hypercube** or **tesseract**. Although next generation climate models switched to unstructured grids to better use the compute power provided by High Performance Computing (HPC) systems the output of these models will be often interpolated to a structured grid format for ease of use. The four dimensions of the tesseract represent the three spatial dimensions (longitude, latitude, and altitude) and time. These dimensions are called **coordinates** or **coordinate variables**. The calculated prognostic and diagnostic variables are referred to as **data variables**. While the horizontal dimensions with longitude and latitude are usually represented by [°E] (longitude) or [°N] (latitude), there is no agreement about the altitude unit. The altitude might be represented in (hybrid) pressure levels (e.g. `[Pa]`, `[hPa]`) or metre (e.g. `[m]`). The temporal resolution can be anything between seconds `[s]`, days `[d]`, months `[months]` or years `[years]`. Most of the climate data is stored using hierarchical data formats like the Network Common Data Format (NetCDF) (Rew and Davis, 1990), General Regularly-distributed Information in Binary form (GRIB) (WMO, 2013), or Hierarchical Data Format (HDF) (Folk et al., 1999). But it is not uncommon for remote sensing data to be stored as ASCII data. The data and coordinate variables are single or double precision floating-point values.

In this chapter, terms and notations were presented that are used throughout the work and a general overview of the basic principles of data compression was given. The reader should now be able to classify different compression algorithms based on their properties and evaluate their perfor-

mance using the metrics presented in this chapter. In the following, related work in the field of compression of climate data as well as structurally similar data is introduced.

CHAPTER 3

Related Work

The biggest challenge with the compression of climate data is data volume. The processing of high volumes of data is a well-known problem in computer science in the era of Big Data. This chapter features an overview of similar challenges in related scientific fields as well as their proposed solutions. This overview is by no means exhaustive. There are many developments in compression, such as succinct data structures, which are skipped for reasons of brevity. The focus of this chapter is on the algorithms used in compression and not on the data structures.

First, related work and solutions directly connected to the topic of climate data compression is introduced. Afterwards, an outline of the state-of-the-art compression methods for image, time-series and floating-point compression is given, since these data share most similarities with climate data. Finally, compression techniques from video, audio and text compression are described.

3.1 Compression of Climate Data

A thorough search of the relevant literature yielded one other group working on lossless compression of climate data. In their work Huang et al. (2016) and Liu et al. (2014) introduce `czip` a lossless compression algorithm for climate data. Huang et al. (2016) use a prediction-based compression algorithm with adaptive prediction, XOR difference and 'multi-way' compression. First the authors identify the most powerful prediction method from a predefined set of predictors by analysing the first 100 time steps of the data. Afterwards, each dataset is split into static and dynamic regions. Static regions are defined as two or more consecutive data points with the same value. Regions not satisfying this criteria are declared as dynamic. The residual is

calculated using XOR difference as described in Section 2.1.4. The residual of dynamic regions is then 'multi-way' compressed: Each single or double precision floating-point data is split into four or eight byte sections respectively, and is compressed using `zlib` (Deutsch and Gailly, 1996) for the first byte sections and `lz4` (Collet, 2011) for the last section. The experimental results in Huang et al. (2016) and Liu et al. (2014) suggest that `czip` achieves a very good CR as well as throughput. Unfortunately no public version of the algorithm was available at the time of writing, so it could not be used for the experiments described in this thesis. Further, the authors assume that the time coordinate variable is large, which might not be the case. A large part of the climate simulation output analysed in this thesis shows that the spatial resolution in a single data set is finer than that of the temporal dimension. The reason for this is that the output of simulations takes place at fixed time steps which are significantly shorter compared to the simulated time scale. As a result, individual files have a few time steps but high spatial resolution.

While there is little related work regarding lossless compression of climate data, there is much more literature about lossy compression of climate data. Baker et al. (2016) and Hübbe et al. (2013) compare the effects of lossy compression on climate data using different error thresholds and quality metrics.

Baker et al. (2016) conclude that 'lossy compression can effectively reduce climate simulation data volumes without negatively impacting scientific conclusions' if certain precautions have been taken. First, the compression must be done on a variable-by-variable basis. This would allow applying different compression algorithms per variable and defining adaptive thresholds. The authors also suggest considering derived variables from post-processing workflows when choosing the above mentioned thresholds, since these variables are most severely affected by error propagation from lossily compressed data. Further, Baker et al. (2016) argue that compression algorithms should consider a special treatment for missing or fill values, because these can appear very often and intermittently in climate data.

Hübbe et al. (2013) compare three lossy compression methods (GRIB2 coding, GRIB2 with JPEG2000 and LZMA, and proprietary APAX) regarding data quality, compression ratio and processing time. The experimental results show that APAX outperforms GRIB2 coding regarding data qual-

ity especially in climate variables with often repeating values. The authors recommend using one of the other two lossy compression algorithms compared if not execution speed but high precision is the main concern in the reconstructed data, e.g. for archiving purposes. Hübbe et al. (2013) suggest 'due to the statistical nature of climate research' that the input and output datasets could probably be lossy compressed, but suggest lossless compression for intermediate results (e.g. checkpoint and restart files during simulation) to 'carefully monitor the numerical stability and consistency of results'.

Sasaki et al. (2015) analyse lossy compression of climate data specifically for checkpoints and restarts during simulation runs. The authors propose a lossy compression technique based on Haar wavelet transformation (Haar, 1910) for checkpoints. The wall clock time of a checkpoint is reduced by 81%, while the average relative error introduced by the compression method is around 1.2%. The reduction in checkpoint time is achieved due to the strong decrease in I/O compared to additional computation time. The authors expect derivations to be in the order of $\sqrt{n}$ with n representing the calculation steps in the simulation. Sasaki et al. (2015) suggest this 'may be acceptable compared to inherent errors to scientific simulations'. But this derivations are additional to the error computed by the scientific models. Further, an error in the per cent range is high for certain environmental variables especially if derived variables are considered. Unfortunately, the authors do only a single restart of the simulations during their experiments. It would be interesting to know how the error propagates with several restarts.

OBSERVATION 3.1 Lossless compression of climate data is not very well researched.

OBSERVATION 3.2 Lossy compression is a viable option if certain criteria are met: variable-based thresholds, consideration of post-processing steps, and special treatment of missing values.

3.2 Compression of Data Similar to Climate Data

While there is only a handful of research papers regarding the compression of climate data, there are a lot of research papers about compression of data that share structural similarities with climate data. Image data are two dimensional numerical data sharing similarities with the horizontal

resolution of climate data. Time-series data share similarities with climate data along the temporal dimension. Further, since climate simulation output is mostly floating-point data, climate research has the same challenges as other scientific fields dealing with high volume floating-point data.

3.2.1 Image Compression

A digital image is a rectangular array of **dots**, **cells** or **pixels**, arranged in a grid. In case of grayscale images the pixels are shades of grey with integer values in the range of 0–255. Colour images are often represented as a triple of 0–255 for each primary colour or a quad including the alpha level. Image compression algorithms are often designed specifically for various kinds of images (Salomon and Motta, 2010): bi-level images (i.e. images with only two colours), grayscale images, continuous tone images, or discrete-tone images (i.e. images without noise nor blur like text images, cartoons or charts). The assumption for image compression is that each random pixel is strongly correlated with its neighbouring pixels. The neighbouring values of a pixel are called its **context**. The reason for the similarities between compression techniques for image and climate data lies in the spatial structure of the climate data, which is also structured like a grid. As with pictures, it can be assumed that the meteorological conditions at two locations are rather similar at short distances.

Many image compression algorithms use lossy compression since the human eye has inefficiencies in the perception of certain shades or gradients. Compression algorithms exploiting these inefficiencies are called **perceptive compression algorithms**. One of these perceptive image compression algorithms is JPEG (ITU-T.81, 1992). JPEG can be used in lossy and lossless mode. In lossy mode the algorithm first maps the red, green, and blue (RGB) values to luminance and chrominance representation. Since the human eye is more sensitive to changes in luminance rather than chrominance, the chrominance values get downsampled to artificially increase correlation in the dataset. Afterwards the data is decorrelated using discrete cosine transform (DCT) and encoded using run-length encoding (RLE) coupled with Huffman Coding (Huffman, 1952). In lossless mode the algorithm does not use downsampling and switches the decorrelation step from a DCT algorithm to a prediction-based algorithm using the predictors given in Ta-

TABLE 3.1 Predictors used by JPEG in lossless mode.

Type	Predictor			
One-dimensional	$\rho_{1,0}$	$\rho_{0,1}$	$\rho_{1,1}$	
Two-dimensional	$\rho_{1,0} + \rho_{0,1} - \rho_{1,1}$	$\rho_{1,0} + \frac{\rho_{0,1}-\rho_{1,1}}{2}$	$\rho_{0,1} + \frac{\rho_{1,0}-\rho_{1,1}}{2}$	$\frac{\rho_{1,0}+\rho_{0,1}}{2}$

ble 3.1. The user has to choose manually which predictor to use. Since the lossless compression results of JPEG yield low CF, this mode is often not implemented by JPEG library providers (Salomon and Motta, 2010). Due to the problems of JPEG in lossless mode the standard JPEG-LS (ITU-T.87, 1998) was developed. Algorithm 3.1 depicts the prediction step of JPEG-LS. This

```
1:  procedure PREDICTION(ρ_{1,0},ρ_{0,1},ρ_{1,1})
2:      r ← ρ_{1,0} + ρ_{0,1} − ρ_{1,1}
3:      if max{ρ_{1,0}, ρ_{0,1}, ρ_{1,1}} = ρ_{1,1} then
4:          r ← min{ρ_{1,0}, ρ_{0,1}, ρ_{1,1}}
5:      else if min{ρ_{1,0}, ρ_{0,1}, ρ_{1,1}} = ρ_{1,1} then
6:          r ← max{ρ_{1,0}, ρ_{0,1}, ρ_{1,1}}
7:      end if
8:      ctx ←CONTEXT(ρ_{1,0},ρ_{0,1},ρ_{1,1},ρ_{−1,1})
9:      s ←SIGN(ρ_{1,0},ρ_{0,1},ρ_{1,1},ρ_{−1,1})
10:     r ← r + C[ctx] · s
11:     r ← min{r, maxval}
12:     return max{r, 0}
13: end procedure
```

ALGORITHM 3.1 Prediction step of JPEG-LS image compression algorithm.

update of the algorithm is a good example on how compression algorithms develop. The decorrelation is done with a more sophisticated prediction algorithm: the prediction-model uses an edge-detection mechanism coupled with a correction algorithm. The correction algorithm calculates the context ctx based on the surrounding gradients $\rho_{-1,1} - \rho_{0,1}$, $\rho_{0,1} - \rho_{1,1}$, and $\rho_{1,1} - \rho_{1,0}$ and applies the appropriate correction using a predefined correction table C. Finally the data is coded using Golomb coding (Golomb, 1966).

OBSERVATION 3.3 The development of JPEG from the use of a rather simple decorrelation method to a more sophisticated one is common in the development of compression algorithms. Where previously the neighbourhood was sufficient, now edge probabilities are calculated and corrections are made.

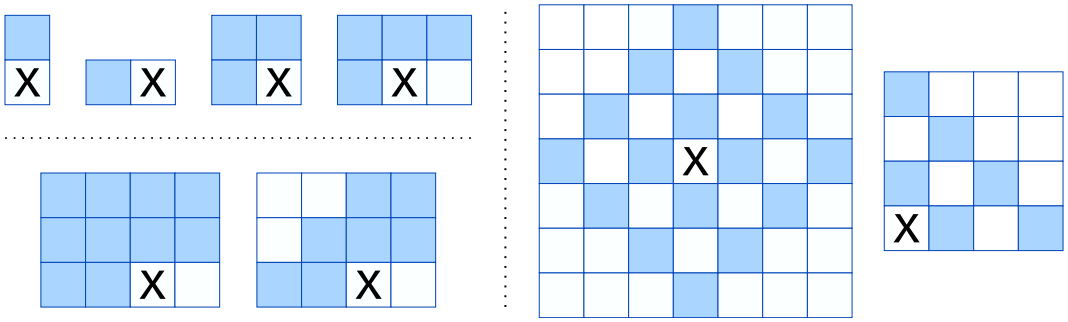


FIGURE 3.1 Neighbourhoods used from lossless image compression methods. The left top row represents those used by JPEG (ITU-T.81, 1992) and JPEG-LS (ITU-T.87, 1998). The left bottom row is used by Motta et al. (2000), Wu and Memon (1997), Moffat (1991), and Langdon and Rissanen (1981). The right column represents those used by Howard and Vitter (1993). The X depicts the pixel to be predicted.

The available information for making a prediction and decorrelating the data is the same, since the sequence in which the data points are traversed is the same. But the further the techniques advance the more sophisticated the methods are to extract the right information from the neighbourhood.

Further developments in the area of lossless image compression improved upon the context calculation and patterns of neighbourhoods to be considered. If lossless compression is expected, then most of the algorithms switch to a prediction-based algorithm for decorrelation of the data (Li and Orchard, 2001; Aiazzi et al., 1999; Weinberger et al., 2000; Neves and Pinho, 2009; Taquet and Labit, 2012). Some of these neighbourhoods are represented in Figure 3.1. Due to brevity, only a few of these patterns are listed. Please refer to Salomon and Motta (2010) for an overview of the topic. Lossy image compression algorithms often change to a more time-efficient method than prediction-based compression, like the above mentioned JPEG (ITU-T.81, 1992). The decorrelation methods applied in lossy compression are diverse: neural networks (Toderici et al., 2017), centroids (Karadimitriou and Tyler, 1998), wavelets (Bruylants et al., 2015), Haar transforms (Haar, 1910), discrete cosine transform (Ahmed et al., 1974), as well as perceptual methods (Alakuijala et al., 2017).

The keen reader might observe similarities with multi-dimensional interpolation methods. Both problem fields are related. In mathematics the problem described above is known as **Lagrange**, **Hermite**, or **Birkhoff interpolation problem**. The **Lagrange interpolation problem** defines the

challenges of interpolation on a multi-dimensional grid if only the actual values are considered. The **Hermite interpolation problem** considers additionally the derivative of the multi-dimensional problem. Finally, the **Birkhoff interpolation problem**, considers the interpolation of the actual value, the derivative and possible missing values. For more details, please refer to Burden and Faires (1997); Sauer and Xu (2006); Allasia et al. (2018); Krislock and Wolkowicz (2012); Schultz (1970) and Olver (2006).

3.2.2 Time-Series Compression

Since one of the coordinate variables of climate data is representing temporal information, there are strong similarities between climate and time-series data. There are only a few scientific communities which need lossless compression of time-series data. One of these communities is medicine, specifically neurology (Sriraam, 2012; Sriraam and Eswaran, 2008; Srinivasan and Reddy, 2010).

Sriraam (2012) and Sriraam and Eswaran (2008) developed a lossless compression algorithm for Electroencephalography (EEG) data using a prediction-based compression algorithm. The authors train three neural networks (e.g. single-layer perceptron) and two linear prediction models i.e. auto-regressive (AR) and finite impulse response filter (FIR) for decorrelation. This prediction step is then followed by a statistical error modelling scheme (Sriraam and Eswaran, 2008). Since recording EEG is time-critical the authors define **computational efficiency (CE)** as their quality metric:

$$\mathrm{CE} = \frac{\mathrm{CR}}{\text{processing time}} \tag{3.1}$$

The results indicate that there is almost no performance difference between neural networks and linear predictions. Among the different predictors used, it is found that the single-layer perceptron yields the best compression results closely followed by the AR model.

Pelkonen et al. (2015) introduce a lossless time-series compression algorithm for high velocity data for in-memory time-series databases. The algorithm uses a prediction-based method in which the delta-of-delta is computed. Delta compression refers to techniques when, instead of storing the absolute values of the time-series data, the differences between the in-

dividual data points are stored. This difference calculation can be applied recursively, which results in the delta-of-delta definition. The difference is calculated using XOR (see Eq. 2.14). Because of the high data velocity, the XOR operation results in a high leading zero count (LZC). These residuals are saved and finally coded using variable length-coding. The results indicate high throughput possibilities '[to] efficiently store monitoring data comprising of over 700 million points per minute' (Pelkonen et al., 2015). An additional analysis of less frequent time-series data would also be interesting as the correlations between the time steps would likely be lower than those obtained with high frequency time-series data.

OBSERVATION 3.4 Lossless compression methods for time-series data use mostly prediction-based methods.

Lossy time-series compression algorithms are applied in a lot of communities: databases (Deri et al., 2012), haptic telepresence systems (Kuschel et al., 2006), artificial intelligence (Fink and Gandhi, 2011; Del Testa and Rossi, 2015), sensor networks (Huamin Chen et al., 2005), and smart grids (Eichinger et al., 2015). As in the previous section about lossy image compression, many of these algorithms use different transformation techniques like wavelets and DCT to decorrelate the data.

There are other interesting time-series models for compression of time-series data. These models are so-called forecasting models. One of these models is the Auto Regressive Integrated Moving Average (ARIMA) model first introduced by Box and Jenkins (1976). The ARIMA model helps identify interdependencies in time-series data and is applied by different communities e.g. economics (French et al., 1987; Pai and Lin, 2005), telecommunications and multimedia (Al Tamimi et al., 2008; De la Cruz et al., 1998; Garrett and Willinger, 1994) and social studies (Chen et al., 2008).

3.2.3 Floating-Point Compression

In the last couple of years the development of compression algorithms for floating-point data experienced a renaissance (Cayoglu et al., 2019b). The reason for the increasing interest in floating-point data compression is the amount of data from the next generation of high-resolution models that can simulate increasingly complex systems with fast HPC systems. The first

		byte number	1	2	3	4	5	6	7	8
a_1	2.3667**1**76745585676	Int(a_1)	40	02	ef	**09**	**ad**	**18**	**c0**	**f6**
a_2	2.3667**2**76745585676	Int(a_2)	40	02	ef	**0e**	**eb**	**46**	**23**	**2f**
a_3	2.3667**3**76745585676	Int(a_3)	40	02	ef	**14**	**29**	**73**	**85**	**6a**

FIGURE 3.2 Memory representation of double precision floating-point values. While the decimal representation differs in one position, the last five bytes of the memory representation are different. The memory representation is given in hexadecimal. The differences in each representation are emphasised. This figure is taken from Engelson et al. (2002).

articles introducing novel lossless floating-point compression algorithms for scientific data were Engelson et al. (2002); Isenburg et al. (2005); Ratanaworabhan et al. (2006) and Lindstrom and Isenburg (2006).

Engelson et al. (2002) develop a lossless compression method for floating-point time-series data using delta compression. In the next step Lagrange extrapolation (Waring, 1779) is used for prediction. The authors also explain the problem with memory representation of floating-point data (see Fig. 3.2). Even if the decimal representation of two numbers can be very similar, this does not need to be the case for the representation in memory. This problem is also addressed by Isenburg et al. (2005) and later extended in Lindstrom and Isenburg (2006).

Isenburg et al. (2005) use the Lorenzo predictor (Ibarria et al., 2003) for decorrelation of the data and use range coding (Martin, 1979) in the final step. To prevent the pitfalls of binary memory representation the authors split the residual in different blocks and code them separately. More details of this method will follow in Chapter 7.

Ratanaworabhan et al. (2006) follow a different approach in residual calculation. Like Engelson et al. (2002), the authors try to address the problem of memory representation by delta compression in conjunction with an additional correction step. The correction is calculated using previous errors saved in a hash table. The key of each correction is calculated using the current context. The context is defined by the last three deltas in the data. Afterwards, the residual is calculated using XOR difference and saved on disk using Range Coding (Martin, 1979).

Fout and Ma (2012) focus on high throughput compression. They propose an asymmetric compression method where the context is calculated based on the last four values in the source. Each of the 16 predictors cal-

culate a prediction for the context and the best predictor is chosen. This information is saved for future use. If the same context reoccurs, a recalculation is not necessary. The residual is being calculated using XOR and coded using Golomb Coding (Golomb, 1966). The information about the chosen predictor is saved in a nibble and passed on to the decoder.

Gomez and Cappello (2013) describe a masking technique to decrease the entropy of the data. The data is split into several blocks. For each block a **signature value** is calculated 'based on the most frequent value for the given bit column' (Gomez and Cappello, 2013). The appropriate masks are then calculated and coded. The masks are then saved using the `zlib` compression library (Deutsch and Gailly, 1996). Experimental results suggest that in some cases a reordering of the masks results in high CF. The reorganisation is done by splitting and rearranging each byte block in the data.

Ainsworth et al. (2017) analyse the effects of decimation and define a formula to calculate **a priori** the lower bound and the expected CR using decimation depending on the stride size. The authors propose a compression algorithm based on their experimental results. Although the proposed compression algorithm performs worse than those it is compared with, the uniqueness of their work lies in the analysis and calculation of the expected CR.

Due to the challenges mentioned above and low compression factors achieved using lossless compression, several working groups turned their focus on lossy compression of floating-point data. Currently there are two dominating lossy compression algorithms for floating-point data: Squeeze (SZ) and zip floating-point (ZFP). Both algorithms apply different decorrelation algorithms for the data. SZ uses a prediction-based algorithm, while ZFP applies a transformation-based algorithm.

SZ is introduced in Di and Cappello (2016) and has since been updated several times: In Tao et al. (2017b) the authors improve the applied predictors and achieve better compression results for absolute as well as relative error bound lossy compression. Tao et al. (2018) improve the algorithm upon the fixed peak signal to noise ratio (PSNR) error bound, Liang et al. (2018a) regarding the point-wise relative error bound and Liang et al. (2018b) improve the prediction model of the algorithm. The current model of SZ (Liang et al., 2018b) works with three predictors: the Lorenzo predictor (Ibarria et al., 2003), the mean-integrated Lorenzo predictor, and a

linear-regression based predictor. The authors sample the dataset to identify which prediction model to use for the data. Some prediction models are 'uniformly skewed from the original values if the error bound is relatively large'. Therefore, the authors introduce the mean-integrated Lorenzo predictor for datasets with low variance coupled with a relatively large error bound. In these cases the Lorenzo predictor performs badly, which results in artefacts in the reconstructed data. SZ first chooses the best Lorenzo predictor based on data sampling. Afterwards, the regression coefficients of the linear-regression based predictor is calculated. In the last step, each data block is compressed using either the regression-based predictor or the best Lorenzo predictor. This choice is based on the estimated prediction errors of each predictor for the current data block. Finally, the errors pass through a quantisation step and are Huffman coded (Huffman, 1952). Further, domain specific versions of SZ are available for cosmology simulations (Tao et al., 2017a) and quantum chemistry (Gok et al., 2018).

ZFP is introduced in Lindstrom (2014) and works with an orthogonal block transformation for decorrelation of the data. The decorrelation step is as follows (Lindstrom, 2014): the data is split into $4 \times 4 \times 4$ blocks, aligned by common exponent, converted to a fixed-point representation and transformed using an orthogonal block transform. Afterwards, in the coding step, the transform coefficients are ordered by the expected magnitude, sorted by bit plane and coded using embedded coding (Hong and Ladner, 2002). There are domain specific versions of ZFP: Lindstrom et al. (2016) customise the ZFP algorithm for seismic waveform tomography data. There are several other lossy compression algorithms for floating-point data.

Due to the different nature of their decorrelation steps SZ and ZFP perform better for different data. Tao et al. (2019) introduces an automatic online selection process between these two methods, to choose the optimal lossy compression algorithm for any given data.

Marin et al. (2016) focus on high throughput lossy compression and rely on fast integral transformations, such as the discrete Chebyshev transform (DChT) (Chebyshev, 1853) and the discrete Legendre transform (DLT) (Legendre, 1790) for the decorrelation of the data. The data is mapped to the spectral space, truncated using a user-defined error threshold and Huffman coded (Huffman, 1952) using binary symbols.

Lakshminarasimhan et al. (2011) develop the lossy compression algorithm Sort-And-B-spline Error-bounded Lossy Abatement (ISABELA) for spatio-temporal data. The authors focus on in-situ compression, which processes the data 'in-tandem with the simulation by utilising either the same compute nodes or the staging nodes'. For spatial decorrelation the data is first split into windows of a certain size, the values inside each window is sorted and then decorrelated using cubic B-splines (Bartels et al., 1987). The original index position of each data point within the corresponding window and the coefficients of the splines are later saved on disk. The temporal dependencies are decorrelated by comparing each window with the previous one and using delta coding for the result.

Liu et al. (2017) develop a lossy compression algorithm by truncating the last s bits of a double precision floating-point value, in such a way, that the remaining 64 minus s bits are within the user specified relative point-wise error threshold. The authors suggest saving the truncated bits in remote storage and applying these on the data only if full precision is desired.

Lu et al. (2018) perform different experiments with lossy compression algorithms to identify good **compressibility indicators** to predict which lossy compression algorithm will perform best for any given dataset. Following data features are identified as good compressibility indicators: cumulative distribution function (CDF), **byte entropy**, **coreset size** (indicating the number of unique symbols composing > 90% of the data), and auto-correlation which measures the extent to which each symbol depends upon the previous symbol. The experimental results show that in general SZ exceeds the performance of ZFP, which in turn exceeds that of ISABELA. Since ISABELA and SZ are asymmetric compression methods, the decoding throughput is higher than the coding throughput. Further, the authors suggest different subsampling methods for SZ and ZFP. A random block-based sampling is suggested for ZFP and a Gaussian model is suggested for SZ.

Velegar et al. (2019) suggest using singular value decomposition (SVD) for the lossy compression of data. Originally proposed for scalable diagnostic analysis, the authors suggest using SVD for low rank representations of the data to reach compression factors of about 1000.

Dunton et al. (2019) proposes using matrix decompositions for the decorrelation of data and introduce a novel single-pass algorithm for high throughput computing of interpolative decompositions. Lossy compres-

sion factors 'exceeding 400 are achieved while maintaining accuracy with respect to first- and second-order flow statistics' (Dunton et al., 2019). The difficulty in using SVD or other matrix decompositions are error thresholds. None of these methods support relative or absolute thresholds for controlled information loss during lossy compression. For this reason, these methods are not yet widely used.

OBSERVATION 3.5 If lossless compression is the goal, almost all proposed algorithms use prediction-based compression algorithms for decorrelation of the data. Lossy compression promises very high compression factors compared to lossless compression. The authors of lossy compression methods often choose transformation-based compression algorithms, with the exception of SZ.

3.3 Compression of Other Data

This section introduces compression techniques used in video, audio, and text compression. For reasons of brevity, only the most common methods are described in this section. Please see Salomon and Motta (2010) for a more comprehensive study.

Dictionary-based compression methods are very successful in compressing textual data. One of the most used (and extended) dictionary-based method is Lempel Ziv (LZ) introduced in Ziv and Lempel (1977). The algorithm decorrelates the data using references to previous occurrences of the words in the text. The data is traversed using two buffers called **search space** and **look-ahead buffer**. The text is traversed from the beginning to the end. During the traversal the search space consists of the last s characters of the text and the look-ahead buffer consists of the current and u upcoming characters of the text with $s \gg u$. If the current character is found in the search space the algorithm tries to match every following character from the look-ahead buffer. The offset and match length is then saved on disk. The DEFLATE algorithm introduced in Deutsch (1996) and implemented in `zlib` (Deutsch and Gailly, 1996) is based on LZ coupled with Huffman coding (Huffman, 1952) as is the Lempel–Ziv–Markov chain Algorithm (LZMA) algorithm implemented in `7z` (Pavlov, 2019).

Video compression algorithms are usually lossy (Salomon and Motta, 2010) and use prediction-based compression for decorrelation. A video signal consists of several **frames** or **pictures**. For compression the frames are split into **Intra (I)**, **Bidirectional (B)**, and **Predictive (P)** frames. An **I** frame is encoded independently, a **P** frame is encoded using preceding **I** and **P** frames, and a **B** frame is encoded using preceding and following **I** and **P** frames. The number of encoded frames is usually $B > P > I$. Some decorrelation techniques applied to these frames are according to Salomon and Motta (2010): subsampling (only any other frame is encoded), differentiation (comparing each pixel with the pixel at the same position on the previous frame and delta coding if the difference is above a certain threshold), block comparison (differentiating by comparing pixel blocks), motion compensation (similar to LZ compression by referring to blocks from the previous frame), frame segmentation, and distortion measures.

While it is possible to compress digital audio data using lossless compression methods, this might not be beneficial if the piece does not have strong reoccurring characteristics. Similar to human sight, human perception of sound has inefficiencies which can be utilised more effectively with lossy compression algorithms. Two of these techniques are (Salomon and Motta, 2010): **silence compression** and **companding**. **Silence compression** is a technique in which very short samples (up to three samples) are treated as if they were silent. **Companding** (a word composed from **compression** and **expanding**) refers to a technique which makes use of human perception of sound. Sounds with low amplitudes need generally more precise samples than sounds with higher amplitudes. Therefore, the sampling rate for higher amplitude sounds can be reduced without effecting the human perception.

Robinson (1994) introduces a lossless audio compression algorithm with Lagrange interpolation for decorrelation and RICE codes (Rice and Plaunt, 1971) for coding of the data. Josh Coalson (2019) extends upon these ideas and introduces the FLAC compression algorithm. Next to the fixed linear predictors employed by Robinson (1994) Coalson suggests using linear predictive coding (LPC) first introduced in Rabiner and Schafer (1978). These predictors analyse the auto-correlation of the data. FLAC applies the Levinson-Durbin recursion (Levinson, 1946) to solve the system of linear

equations and calculate the optimal predictor coefficients minimising residuals. Finally, the residuals are coded using Rice coding (Rice and Plaunt, 1971).

This chapter gave a brief overview of related work. First, research papers directly related to lossy and lossless compression of climate data were discussed. A thorough search of the relevant literature yielded two studies on lossless compression of climate data. This lack of in-depth analyses is the raison d'être of this thesis.

Next, related work of data compression that shares similarities with climate data were presented. Our findings suggest that most of the lossless compression algorithms apply a prediction-based compression algorithm, while lossy compression algorithms tend to use transformation-based compression methods. Further, the algorithms differ mostly in their way of using the available information, while the information sources being used are often the same. The results suggest that a good compression algorithm has to have the ability to adjust to the information available and assess the quality of the available information.

CHAPTER 4

Data Analysis and Identification of Redundant Information

Decorrelation is a key step in the design of a good compression algorithm. For successful decorrelation, it is important to understand the intrinsics of the data to be compressed. This understanding is the reason why custom compression techniques like JPEG are more successful in compressing images i.e. domain-specific data. Knowledge about the internal structure helps to identify redundant information which can then be removed to achieve better compression factors. This chapter describes different analysis techniques to achieve this understanding for climate data. The analyses described here help to overcome the first challenge described in Chapter 1.

Section 4.1 gives a short motivation about the different analysis techniques. These techniques are then described in detail in Section 4.2. This is then followed by a description of the experimental setup in Section 4.3 and the evaluation of the experimental results in Section 4.4. Finally, Section 4.5 concludes the chapter with a summary and possible future work.

4.1 Motivation

The Earth's atmosphere is a chaotic system. While mankind now has a better understanding of the Earth's atmosphere than a couple of decades ago, there are still atmospheric interactions that are not fully understood or not known to mankind.

The goal of this chapter is to use data analysis to gain insights into the interactions of variables. The current temperature is more comparable to a measurement from an hour ago, than to a measurement taken last week.

But does the relationship change for daily or monthly data? Does the interaction between temperature and humidity at the Equator differ from those at the polar regions? Finding out about these relationships will help to identify types of relationships that can exist between variables. Once these are recognised, methods can be developed that automatically recognise these relationships.

The goal is to develop a method that can identify these types of relationships during the traversal of the data. That method then can assess the most significant information in the immediate context to calculate a good prediction for any data point in the data.

4.2 Proposed Methods

For the analysis of climate variables and their interrelations three different analysis are carried out. Each of these techniques focuses on a different relationship aspect.

First, a variance analysis is conducted. The goal of the variance analysis is to identify variations in distribution for each variable. The analysis helps to identify which dimension is the most important information source. The less variance a variable has across a dimension, the easier it is to predict along that dimension. Each data point is analysed for irregularities along each of the dimensions. Doing this for each data point helps to identify context-based differences e.g. changes depending on location on the globe, time of the year or elevation level. Since datasets with several temporal resolutions are available, these relationships are calculated for long and short-term data. It is important to identify relationships at several time scales, since the temporal resolution of climate data may range from hourly data to monthly or yearly data.

Next, analyses from information theory are carried out. Foremost the Shannon Entropy is calculated. The Shannon Entropy can be used as a threshold for the maximum achievable bits per float for each variable. But the Shannon Entropy does not consider interdependencies within the data. Temporal relationships can therefore not be analysed using Shannon Entropy. In order to discover these kind of relationships the Sample Entropy is calculated. Analysing the Shannon and Sample Entropy of the data will help to set a lower limit for the achievable CR. It is important to note that

the gained information from either entropy analyses is bound to each variable and does not entail information across variables. A third experiment is conducted to gain that information: mutual information.

Mutual information expresses the interdependency between two variables. It expresses the information gained about one variable by observing the other. This information helps to identify how close the relationship between two variables is.

4.3 Experimental Setup

4.3.1 Data

Three datasets with different temporal resolutions are used for the analyses. These datasets originate from a climate simulation run using the ECMWF Hamburg/Modular Earth Submodel System Atmospheric Chemistry (EMAC) climate model (Jöckel et al., 2006). They have a 128×64 horizontal grid (longitude $\times$ latitude) and 47 vertical levels. Three different temporal resolutions are used:

- One month (January, 2013) with 74 time steps i.e. every 10 hours (**hourly data**)
- One year (2013) with 365 time steps i.e. every 24 hours (**daily data**)
- Fourteen years (2000-2013) with 168 time steps i.e. every month (**monthly data**)

The **hourly data** with 10 hour temporal resolution are the original simulation output. The **daily** and **monthly data** are generated by taking daily and monthly means from the original simulation output.

The variables used for the experiments are temperature, zonal and meridional wind as well as specific humidity. The zonal winds flow along the latitudes, crossing each longitudinal line with a positive direction implying wind flow from west to east. The meridional winds travel along the longitudes and cross each latitude. Figure 4.1 depicts an exemplary zonal band (NASA, 2019) along which the zonal wind is calculated. All variables are available as single precision floating-point values.

4.3.2 Experiments

FIGURE 4.1 Zonal band

Five experiments, the results of which are presented in the next section, were conducted to explore the data.

Long-Term Variance Analysis. The first experiment analyses the long-term relationships of all four variables. The monthly dataset is being used for this analysis. It provides the longest temporal information. Therefore it is the most suitable dataset to identify long-term relationships. The variance across each dimension for each of the data points are calculated. In case of climate data these are: time, latitude, longitude and altitude. Less variability along a dimension means that it is a good candidate for decorrelation.

Short-Term Variance Analysis. The steps conducted for the short-term variance analysis are the same as the ones for the long-term analysis. The most significant change is the data being used. For the second experiment, the daily data with the highest temporal resolution is being used.

Shannon Entropy Analysis. The third experiment is conducted with all three datasets. Since the PMF of the datasets is not known a discretization technique must be applied to identify it. For the analysis of the Shannon Entropy and the experiments described in the following sections, a binning strategy is applied for discretization. Therefore, a slightly different definition of the Shannon Entropy is being used than the one defined in Eq. 2.4:

$$\mathrm{H}(X) = \lim_{b \to \infty} \left[- \sum_{x_i \in X} P_b(x_i) \cdot \log_2(P_b(x_i)) \right] \tag{4.1}$$

where $X = \{x_1, x_2, \ldots, x_n\}$ is the alphabet of the source, b defining the bin size and $P_b(\cdot)$ the appropriate PMF given a discretization of X with bin size b. Here, the Shannon Entropy is being calculated with ever increasing binning sizes until the entropy plateaus at a certain value.

Sample Entropy Analysis. The fourth experiment is conducted to calculate the Sample Entropy (see Eq. 2.5). Since the Sample Entropy builds upon the Shannon Entropy a discretization of the data is necessary. Based on the results of the previous experiment a discretization with a bin size of 50 is used. The Sample Entropy expresses the correlation within a variable across a predefined dimension. For the analysis of the Sample Entropy the

altitude dimension is chosen. This decision has been made based on the results of the long-term and short-term variance analyses. The experiment is conducted with all three datasets.

Mutual Information. Finally, the mutual information of all pairs of variables are calculated. For the analysis of the data described in Section 4.3.1 the normalised mutual information NMI is used:

$$\text{NMI}(X,Y) = \frac{\text{MI}(X,Y)}{\text{mean}(H(X),H(Y))} \tag{4.2}$$

$$\text{MI}(X,Y) = \sum_{x_i \in X} \sum_{y_j \in Y} P(x_i,y_j) \log \left(\frac{P(x_i,y_j)}{P(x_i) \cdot P(y_j)} \right) \tag{4.3}$$

where X and Y are the alphabet of each dataset, $P(\cdot)$ the PMF, $\text{MI}(\cdot)$ the mutual information and $\text{H}(X)$ the Shannon Entropy of X. The same binning strategy for discretization as in the previous experiments with $b = 50$ is chosen. The $\text{NMI}(\cdot)$ scales the results of the mutual information $MI(\cdot)$ between 0 (i.e. no information) and 1 (i.e. high correlation).

4.4 Evaluation

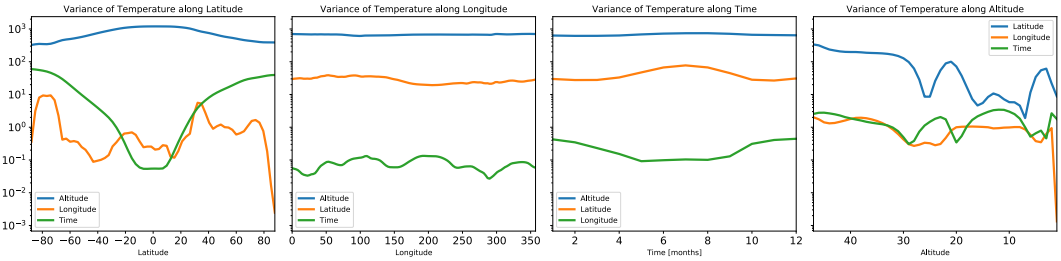
The results of the previously defined experiments are presented in this section.

4.4.1 Long-Term Variance Analysis

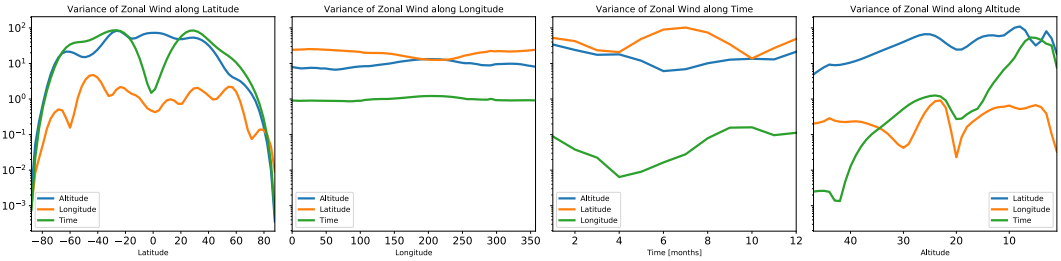
In the following the results of the long-term variance analysis for the monthly dataset are presented. The results are depicted in Fig. 4.2.

Temperature. The results suggest that for temperature temporal and longitudinal information are the most stable (see Fig. 4.2a). If the data point is around the Equator $[-20\,°\text{N}; +20\,°\text{N}]$ the most stable information source is the temporal dimension. Any data point x outside the latitude range of $x < -20\,°\text{N}$ and $x > 20\,°\text{N}$ is more similar to its neighbouring data point on the longitudinal dimension. A possible explanation for this finding is that the temperature distribution is more uniform over time at the Equator than at the poles. Another interesting finding is depicted in the last panel. For the first 17 altitude levels ($[1000\,\text{hPa}; 340\,\text{hPa}]$) the longitudinal and temporal

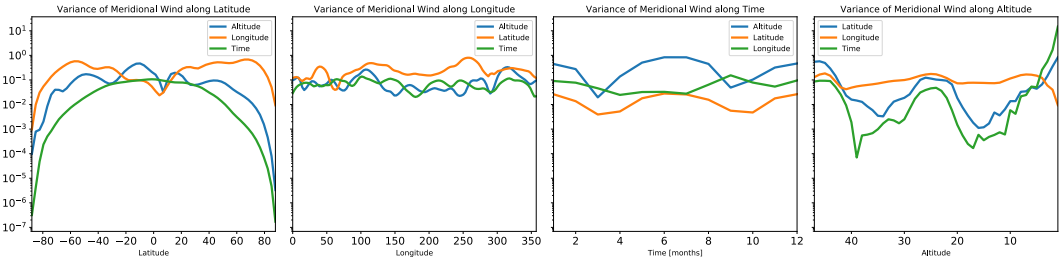
(a) Temperature



(b) Zonal Wind



(c) Meridional Wind



(d) Specific Humidity

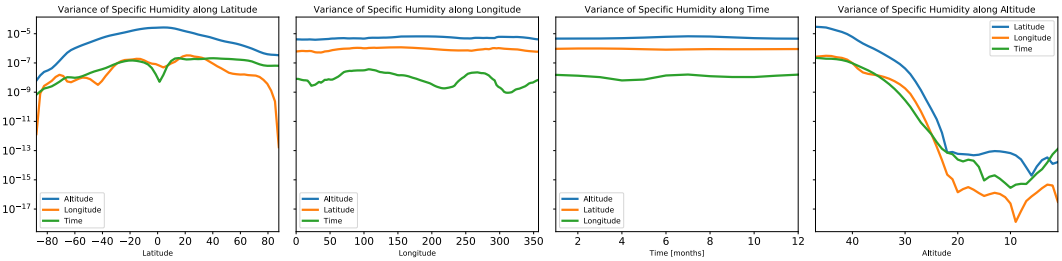


FIGURE 4.2 Long-term variance analysis on a global scale for each variable. Since the altitude is flipped in the data, the abscissa of the last column is descending. Low altitudes are on the far left and high altitudes on the far right.

information have the same variance. At higher altitudes the longitudinal information is more stable than the temporal. The drop around level 21 (100 hPa) is likely due to the tropopause which is roughly at this level.

Zonal Wind. The panels suggest that if longitudinal information is available, it should be used almost all the time for decorrelation of zonal wind (see Fig. 4.2b). This is in agreement with the zonal wind direction depicted in Fig. 4.1. The only exception is in the last panel, for data points at low altitudes ($x < 540$ hPa; level: $35 < x < 47$). Here the data suggests using temporal information for decorrelation might reduce prediction errors.

Meridional Wind. The meridional winds are most similar across the temporal dimension (see Fig. 4.2c). In almost all cases the temporal dimension has the least variability. Should the temporal dimension be missing, the results suggest the use of the neighbouring points on the same latitudes (third panel from left). This is again in agreement with Fig. 4.1 for meridional wind along the longitudes and crossing the latitudes.

Specific Humidity. The last variable analysed is the specific humidity (see Fig. 4.2d). It is a diagnostic variable and can be derived from temperature and water vapour. Therefore it is no surprise, that it shows similar properties as temperature. Around the Equator the smallest variance is across the temporal dimension. Should the temporal dimension be missing it is recommended to use longitudinal information for decorrelation (and vice versa). Should the data point be at pressure levels $47 - 24$ (i.e. $x > 155$ hPa) the preferred dimension should be time, otherwise the longitude is the ideal candidate for decorrelation.

OBSERVATION 4.1 Most often the data points along the temporal and longitudinal dimension are the best sources of information for decorrelation of long-term climate simulation output. Having information about the represented information (e.g. wind direction) or calculation method for diagnostic variables (specific humidity in relation to temperature) will help in the decorrelation step.

4.4.2 Short-Term Variance Analysis

Next, the short-term variances are analysed. Due to brevity, only the plots for temperature are depicted in Fig. 4.3 and 4.4. The remainder of the plots can be found in Appendix A.

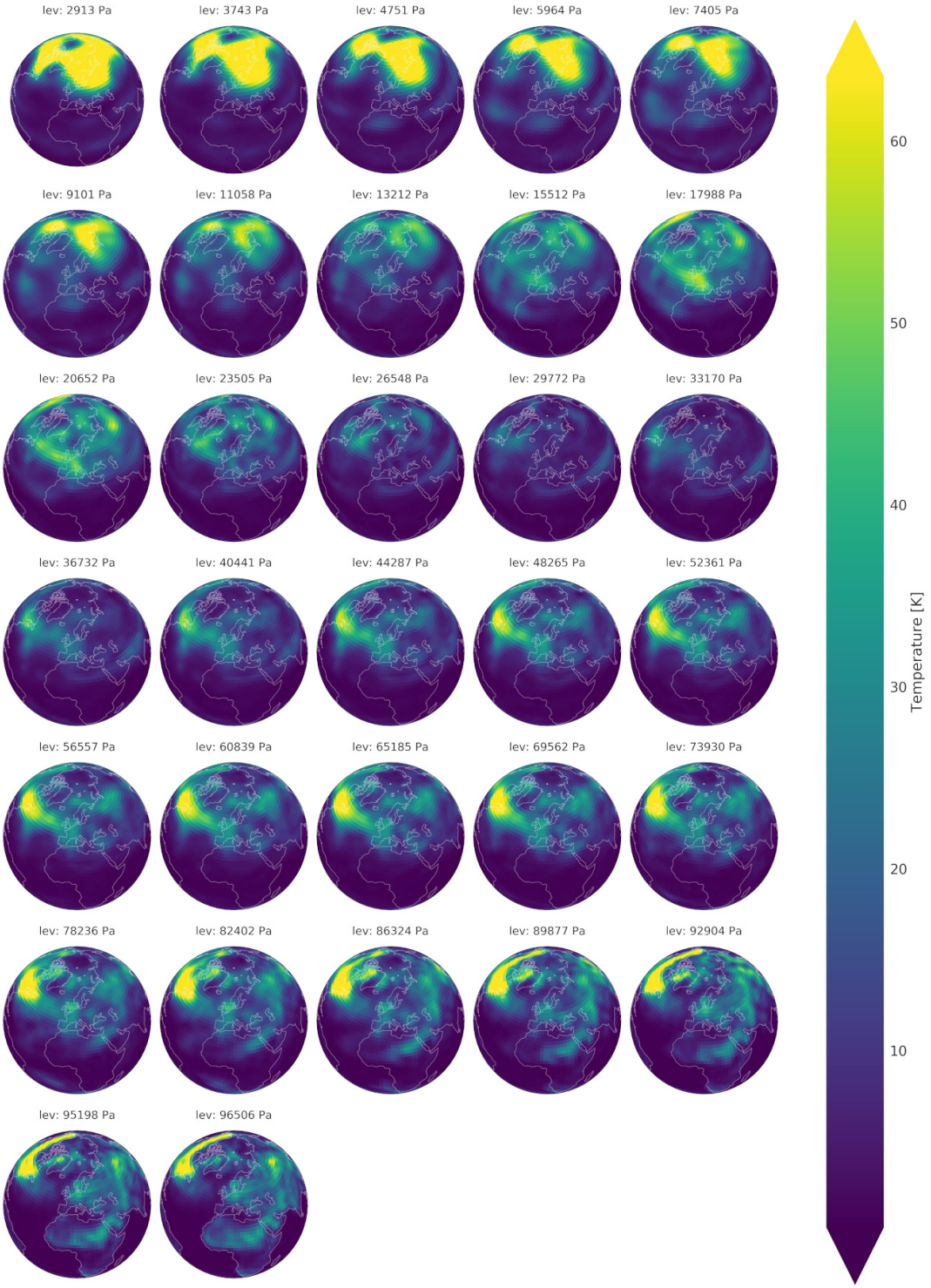


FIGURE 4.3 Short-term variance analysis for temperature across time for the northern hemisphere. For reasons of brevity, only the lowest altitudes are displayed.

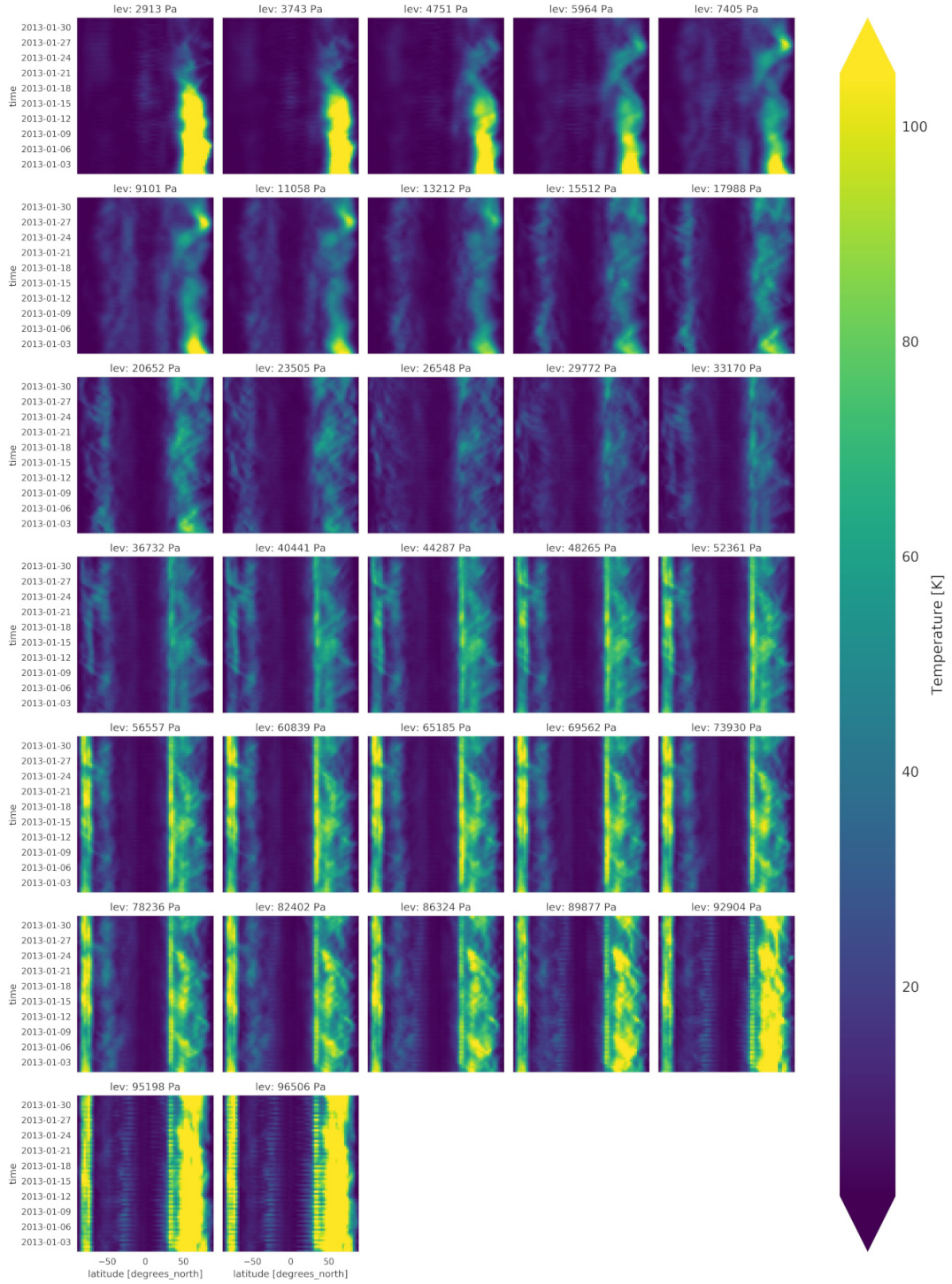


FIGURE 4.4 Short-term variance analysis for temperature across longitude along the latitudes. For reasons of brevity, only the lowest altitudes are displayed.

Temperature. Overall the variance across the temporal dimension (Fig. 4.3) is lower with variances of up to 70 Kelvin than along the longitudinal dimension with up to 110 Kelvin (Fig. 4.4). There is high temporal variance at low and high altitudes of the northern hemisphere. The high variance in temporal dimension is not observed in the southern hemisphere (not depicted). This difference in the northern and southern hemisphere is likely due to the respective season for each hemisphere since January data were used. It is common that temperatures fluctuate more during the winter months (northern hemisphere) than during the summer months (southern hemisphere).

The temporal stability at the Equator and the instability at the poles is nicely depicted in Figure 4.4. There are high dynamics at the poles (latitudes $> 35\,^{\circ}$N and latitudes $< -35\,^{\circ}$N), but the Equator is stable.

Fig. 4.3 also suggests that longitudinal information is more important than latitudinal. The differences in variance seem to follow along a meridional line which suggests that cells at the same latitude but neighbouring longitude might be a good candidate for prediction. This suggests that the results from the long-term experiment can also be seen in the short-term data for temperature.

Zonal Wind. What is true for temperature also applies to zonal wind: The temporal variance is half than the longitudinal variance. The results indicate that temporal variance is mostly stable except for the region just below the tropopause between 500 hPa $- 120$ hPa. The zonal wind is most unstable around the Equator and moves along the latitudes.

Meridional Wind. Contrary to temperature and zonal wind, the variance of the meridional wind is lowest across altitudes (max. $\sim 80\ m/s$) and latitudes (max. $\sim 120\ m/s$). Therefore, the data point at the next altitude/latitude is the most likely candidate for a good prediction. Spot-like patterns are seen in variance across time at different altitudes. These patterns are similar in northern and southern hemisphere. The variance across the latitudes is highest just below the tropopause between 111 hPa $- 520$ hPa. The chaotic pattern continues through here.

Specific Humidity. The variance of specific humidity is low and only appears at the lowest altitudes. While there are higher variances around the Equator due to their low magnitude (max. $1.2e-5$) it can be assumed that specific humidity is mostly stable.

OBSERVATION 4.2 Results from the long-term analysis are reappearing in the short-term analysis. Temporal and longitudinal data are the most stable sources of information for any given data point for temperature, zonal wind and specific humidity.

OBSERVATION 4.3 Latitudinal data is more stable for meridional wind. Therefore, using data points at the same latitude for the prediction of meridional wind is more likely to be a successful candidate for decorrelation of meridional wind.

OBSERVATION 4.4 The respective season has an influence on the temperature variable depending on the location on the globe. Although the temporal dimension is often the most stable, during the winter months there is an increase in variance compared to the summer months.

4.4.3 Entropy Analysis

The results of the Shannon and Sample Entropy analysis are depicted in Figure 4.5. The number of bins used for discretization is depicted on the abscissa in Figure 4.5a. With the bin size, the calculated entropy in the data also increased. For all data variables, a plateau was reached with a bin size of ~ 30. Although the rise of the Shannon Entropy of each dataset is slightly different, the final plateau value is the same. The entropy for all four variables is between 12.3 and 12.5 bits. With this information the lower limit for the CR can be calculated. For single precision floating-point numbers a CR of $12.3/32 = .384$ (i.e. CF of $32/12.3 = 2.60$) might is achievable.

Sample Entropy incorporates auto-correlation for calculating the entropy. The results shown in Figure 4.5b suggest that the monthly data has a higher auto-correlation than the daily or hourly data for all of the variables. The only exception to this is the meridional wind between level 30-25. Here the monthly data have a higher Sample Entropy than the other data, indicating that there are more irregularities at these altitudes.

There is almost no difference in Sample Entropy of the daily and hourly data for temperature and zonal wind. Both variables follow closely the same curvature and differentiate only little (< 0.04) from each other. The same can not be said for meridional wind and specific humidity, the differences between the Sample Entropy of the daily and hourly data exceeds more often and more significantly (~ 0.1) than with the other two variables.

While there are two peaks at the 25 and 3 level mark (i.e. 540 hPa and 11 Pa) in the hourly and daily data for temperature, those do not appear as significant in the monthly data. It is possible that the peak at level 3 is caused by the polar vortex seen at high altitudes in the short-term data (see Fig. 4.3).

OBSERVATION 4.5 The drop of Sample Entropy in the monthly data, coupled with the information of a rather stable Shannon Entropy across the three datasets, suggests that the variations in the monthly data are locally constrained. These regions seem to span a greater area for temperature and zonal wind, since the drop in Sample Entropy is more severe.

OBSERVATION 4.6 Temperature and zonal wind do not fluctuate often over time, as the curvature of the Sample Entropy for the daily and hourly data follow each other closely.

4.4.4 Mutual Information Analysis

Unlike the previous analyses, MI analysis does not search for relationships within a variable, but between variables. The results are illustrated in Figure 4.6.

Temperature. Temperature has the highest NMI with specific humidity and zonal wind. The NMI gets as high as 0.6 at certain altitudes. The interdependence between temperature and specific humidity is expected, since specific humidity is derived from temperature. The high correlation with zonal wind is unexpected. Since temperature is correlated with both variables, it is no surprise that the experiments also show a high correlation between zonal wind and specific humidity. The NMI of temperature and zonal wind follow the same curvature as the NMI of temperature and specific humidity. For low altitudes (i.e. levels > 33) the NMI between temperature and specific humidity is high. At altitudes above the tropopause (levels < 25) there is a sudden increase in NMI between zonal wind and specific humidity.

Figure 4.6a depicts the time average of NMI. There is a clear seasonal dependency between temperature, zonal wind and specific humidity. During the spring/summer months of the northern hemisphere (from March till August), there is a valley in the NMI between temperature and specific humidity, while the NMI of temperature and zonal wind peaks at that time period. The seasonality of this relationship can be observed in the monthly

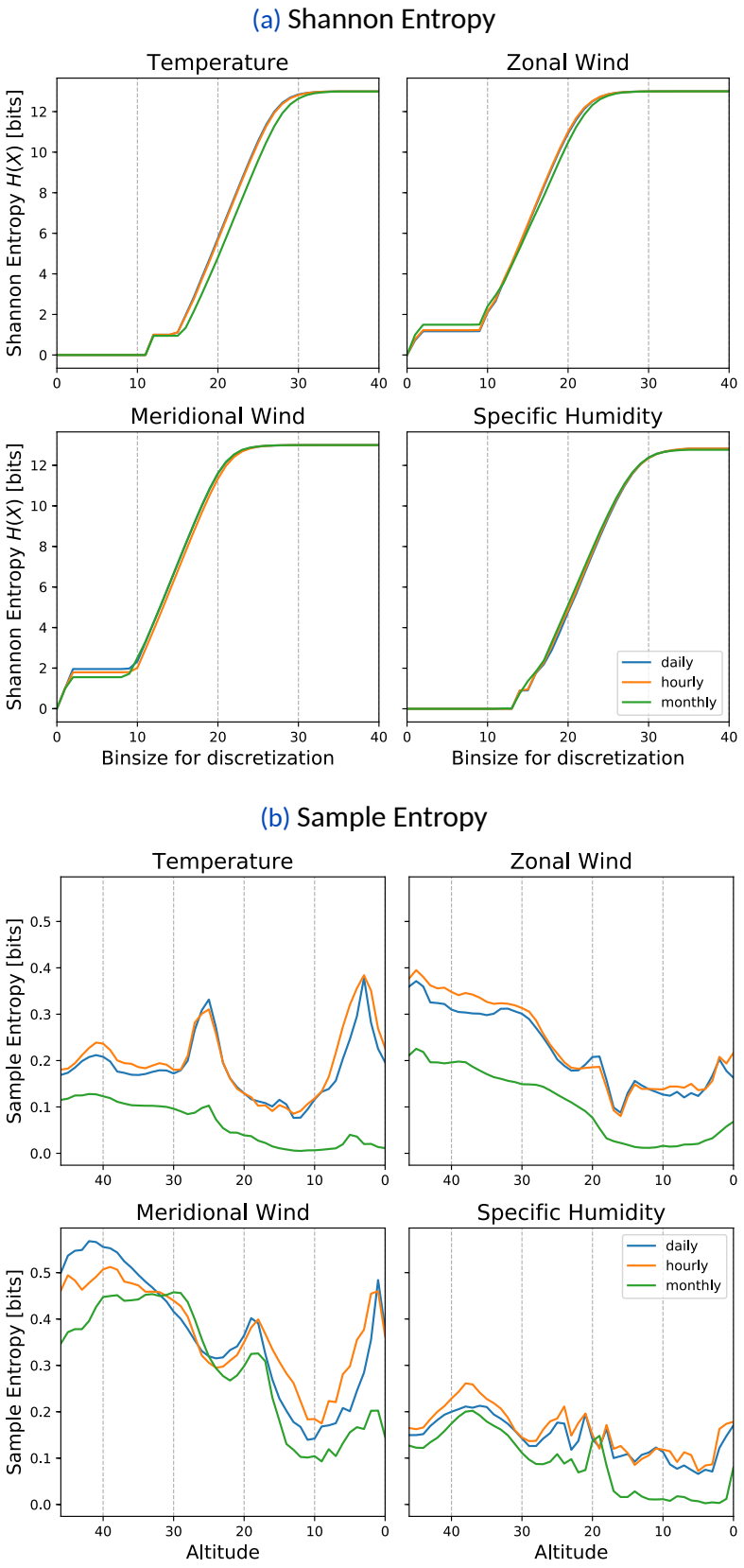
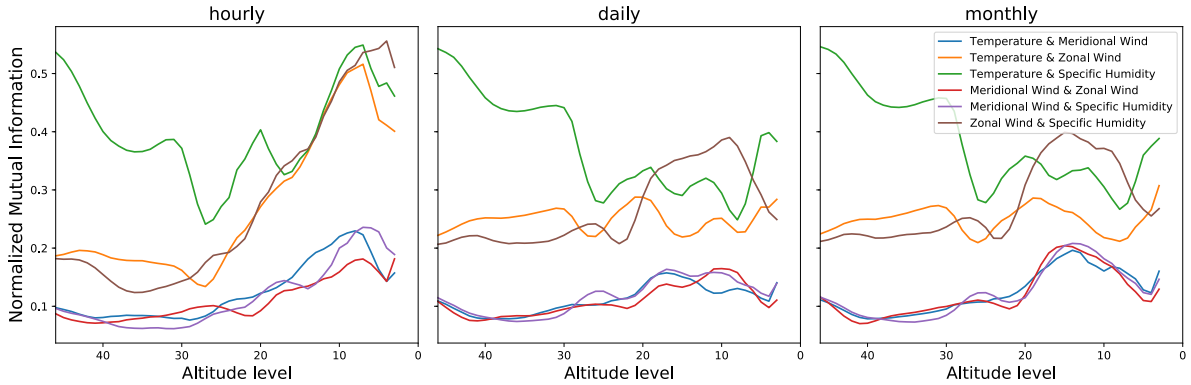


FIGURE 4.5 Entropy Analysis: (a) Shannon Entropy for different bin sizes and (b) Sample Entropy along altitude levels. Since the altitude is flipped in the data, the abscissa is descending. Low altitudes are on the far left and high altitudes on the far right.

(a) NMI across time for each altitude



(b) NMI across altitudes for each time step

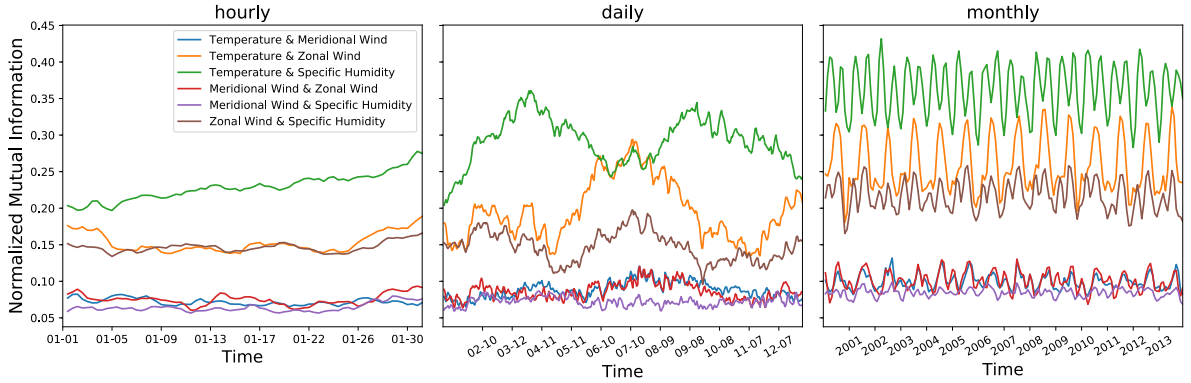


FIGURE 4.6 NMI between data variables for each dataset (a) shows the NMI across time at each altitude and (b) shows the NMI across altitudes at each time step.

data depicted in Figure 4.6a. There is a yearly reoccurring pattern. The NMI between temperature and meridional wind is non-existent with values ranging between $0.05 - 0.1$ across the temporal dimension and between $0.05 - 0.2$ across the altitudes. Although the NMI with meridional wind is low, there is a seasonal pattern observable. Two peaks are reoccurring yearly at the summer months and at the beginning of the year.

Zonal Wind. The zonal wind has a seasonal correlation with temperature and specific humidity along the temporal dimension. The pattern visible in the daily data is yearly reoccurring. The peak is reached in the summer months of the northern hemisphere. The two local minima are often reached at the beginning of spring and mid-autumn of the northern hemisphere. Generally, the correlation of zonal wind with temperature and with specific humidity is not as strong as the correlation between temperature and specific humidity. An exception to this occurs at high altitudes (levels < 20). At these altitudes the correlation between zonal wind and specific humidity peaks with ~ 0.4 for the daily and monthly data and ~ 0.53 for the hourly data. The correlation between zonal wind and specific humidity is the third strongest across the temporal dimension.

Meridional Wind. The meridional wind has the lowest correlation with any of the other variables. There is no difference for the temporal or the vertical dimension. At correlations across time the high altitudes seem to be higher correlated than the lower altitudes, but still below all other correlations with a NMI < 0.2.

Specific Humidity. Since specific humidity is a diagnostic variable and can be derived from temperature it shares a lot of its similarities with temperature regarding NMI. It is correlated with zonal wind, which increases with the altitude. At high altitude levels (i.e. levels 23-7 [130 hPa $- 1$ hPa]) the correlation between specific humidity and zonal wind is higher than that of specific humidity and temperature. However, this connection only occurs when considering the NMI across time (Fig. 4.6a). For the NMI across altitudes (Fig. 4.6b), the NMI to temperature dominates the NMI with the zonal wind.

OBSERVATION 4.7 There is a clear seasonal correlation of temperature with zonal wind and with specific humidity. This relationship is reoccurring over several years based on the results of the monthly data. Additionally, the

monthly data shows the highest NMI compared to the other datasets. The development of the temperature variable seems to be a good indicator for the prediction of zonal wind and specific humidity.

OBSERVATION 4.8 At high altitudes there is a significant correlation between zonal wind and specific humidity. A seasonal pattern similar to the one of temperature is appearing in the zonal wind data.

4.5 Summary

Different data analysis techniques are used in this chapter to identify relationships between and within variables. Five experiments are conducted to learn about the interaction of climate variables.

First, long-term correlations are studied. To this end, a decade of climate simulation is analysed. The experiments show that temporal and longitudinal dimensions are the best sources of information. It is indicated that the data point along these dimensions fluctuate less and are therefore more suitable for context-based predictions.

Second, a short-term variance analysis is conducted. The results of the long-term analysis are confirmed here. Data points along the temporal dimension are the most stable information source for predictions, while those at the horizontal grid depend on the variable.

The third and fourth experiments are carried out to understand the entropy of the data. The result of the Shannon Entropy shows that the maximum achievable compression ratio (CR) is 0.3875 (compression factor (CF) = 2.58). The Sample Entropy implies that there is high auto-correlation in the data (especially for the monthly data). These results support that a better CR than 0.3875 are achievable if auto-correlation effects are considered during the decorrelation.

Finally, the mutual information between the variables is examined. There is a seasonal correlation of temperature with zonal wind and with specific humidity. The correlation also extends onto zonal wind and specific humidity, specifically at high altitudes.

Overall, the results show that there are preferred dimensions which are stable for information gathering. These dimensions should be preferred when predicting data points. However, the experiments also show that the position of the data point on Earth plays a strong role in determining the

context which to be used for the prediction. It is therefore recommended not to use a fixed "template", but to build the context iteratively and change it on-the-fly. This ensures that information from all dimensions is used.

4.6 Code and Data Availability

The data of the environmental indices and an implementation of the analysis methods described above are available under GNU GPLv3 license at `https://github.com/ucyo/climate-data-analysis` (Cayoglu, 2019a).

CHAPTER 5

Data Decorrelation using Information Spaces and Contexts

This chapter addresses challenges one and four, defined in Chapter 1. The concept of Information Spaces (IS) and its components Information Contexts (IC) are presented. IS and IC help to adapt the compression algorithm to the data to be compressed by identifying new patterns and relationships during the compression process.

In prediction-based compression all data points are processed in a predefined sequence. Often, the data is traversed according to the layout in which it is mapped on the disk. This means that already at the start of the compression procedure, information that can be used for each prediction is fixed. In addition, a reorganisation of the data, e.g. by transposition, can lead to a significantly different compression factor than with the original data.

Especially, since the results in the previous chapter suggest, that a more successful decorrelation is possible if the context used for the prediction is adjusted for each data point. The results presented in this chapter have been published in parts in Cayoglu et al. (2018c).

After a short motivation in Section 5.1 the proposed method is introduced in Section 5.2, the experimental setup is described in Section 5.3, and the experiments are evaluated in Section 5.4.

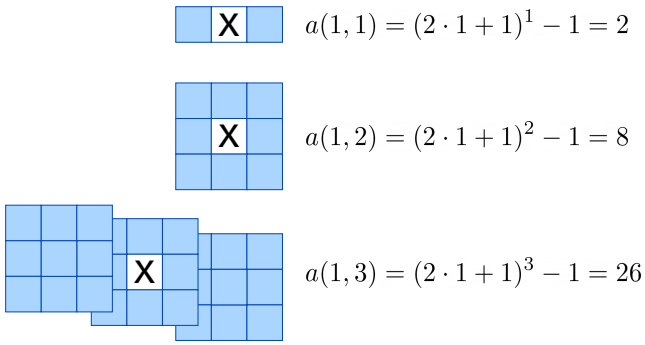


FIGURE 5.1 Adjacent data points for a d-dimensional object according to Eq. 5.1.

5.1 Motivation

A good decorrelation method is crucial for a good compression algorithm. But how to decide which neighbouring cells to use for prediction? The number of data points at distance n in a d-dimensional object is given by the following formula:

$$a(n, d) = (2n + 1)^d - 1 \tag{5.1}$$

where the first term $(2n+1)^d$ describes the n neighbouring data points before and after the data point to be predicted along each dimension (see Fig. 5.1). Since climate data is structured as a tesseract the equation above results in $a(1, 4) = 3^4 - 1 = 80$ adjacent data points and potential information sources. If not only adjacent data points, but data points at intermediate distances are considered e.g. adjacent cells of adjacent cells $a(2, 4) = 624$, the number of potential data points grows very fast. Therefore, an adaptive method for the selection of good neighbouring cells is needed. The method proposed in this chapter provides this selection.

5.2 Proposed Method

The proposed method adapts for each data point the context on which its prediction is based. IS is defined as follows:

DEFINITION 5.1 (information space) The IS of a data point s_i to be predicted is the set of already traversed data points $s_k \in S_0^i$ within a certain range r of s_i.

$$IS(s_i) = \{s_k \mid \forall s_k \in S_0^i : a_j^i - r \leq a_j^k \leq a_j^i + r\} \tag{5.2}$$

where a_m^n defines the coordinate position at dimension m of element n in sequence S. The restriction r is necessary to locally constrain the available information for the prediction of s_i.

The IS can be divided into several components to isolate the information contained along the various dimensions. These components are called information context (IC):

DEFINITION 5.2 (information context) The ICs split the IS into different subsets based on their information level for each dimension and if applicable to each combination of dimensions.

$$IC_l^p(s_i) = \{s_k : \sum_{j=0}^{d} [a_j^i = a_j^k] = l\}_p \tag{5.3}$$

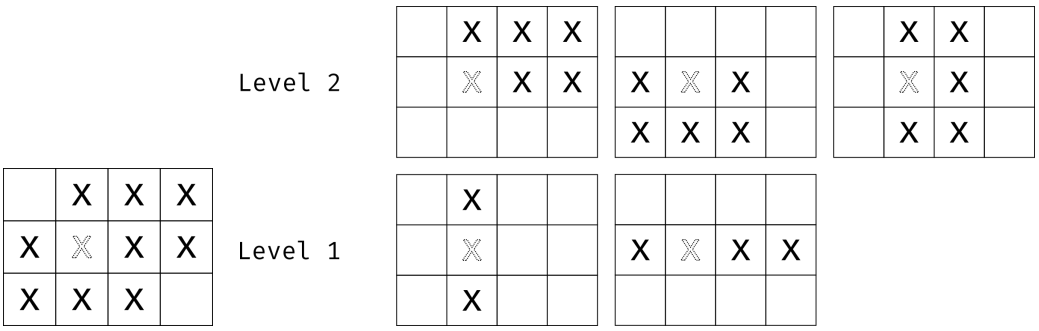
with $0 \leq p \leq \binom{d}{l}$ being the index position of IC_l at level l and $[\,\cdot\,]$ defined as follows:

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{else} \end{cases}$$

Each IC contains information along one or more dimensions. Each IC within a level can contain overlapping data points with other ICs, but none is a subset of the other. This distribution of data allows predictions to be made on the basis of information from different dimensions and later to merge them into a single consolidated prediction.

EXAMPLE: INFORMATION SPACE AND CONTEXT

Given a grid of size 3×4 and the following sequence S defined by its coordinate positions on a grid: $S = \big((0,1),(0,2),(0,3),(1,0),(1,2),(1,3),(2,0),(2,1),(2,2),(1,1),(0,0),(2,3)\big)$. For the prediction of s_{10} at position $(1,1)$ the resulting IS is depicted in Figure 5.2a. This IS consists of the five ICs depicted in Figure 5.2b. These ICs can then be used to improve the prediction of the value at s_{10} by using one of the consolidation methods.



(a) Information Space (b) Information Context

FIGURE 5.2 The IS for the example above is depicted in (a). The value to be predicted is depicted as a dotted x and the values known at the time of prediction are marked with a filled x. The IS can be split into five ICs on two levels. (b) The ICs can then be used to predict the value. This figure is adapted from Cayoglu et al. (2018c).

5.2.1 Consolidation of Predictions

Each IC calculates a prediction. These predictions then are consolidated to a single prediction. Therefore, appropriate consolidation methods are necessary. Five different techniques are implemented and tested:

- **Average (AV)**
  Takes the average of the IC predictions.
- **Minimum (Min)**
  Takes the minimum of the IC predictions.
- **Maximum (Max)**
  Takes the maximum of the IC predictions.
- **LastBest (LB)**
  Tracks which IC was best for s_{i-1} and uses its prediction.
- **Reforced (R)**
  ICs are sorted by the number of data points used from each dimension and given a preference list of dimensions, the IC with the most data points from the most preferred dimension is used.

The motivation behind using Minimum and Maximum for consolidation is to find out if the predictor has a bias towards one or the other. With the introduction of IS and IC, as well as methods for consolidation, the traversal methods are now described in more detail.

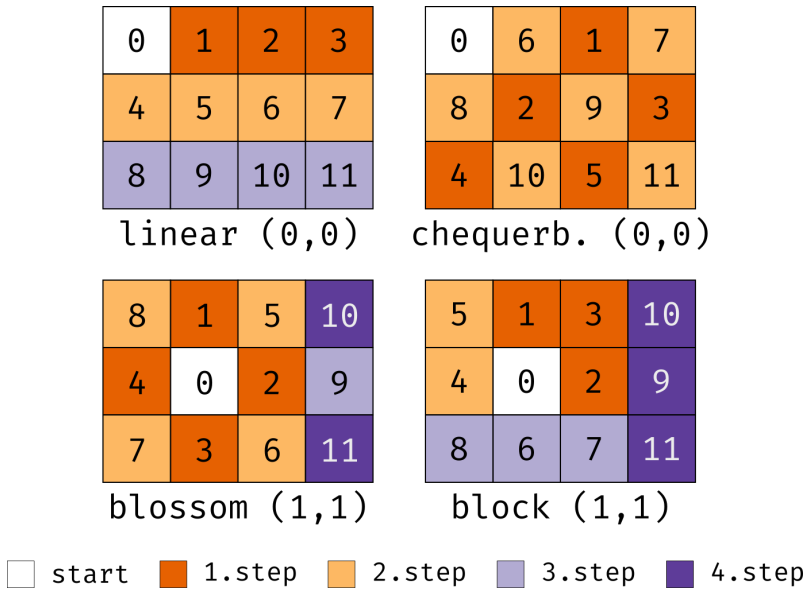


FIGURE 5.3 Four different traversal path tested during the experiments. This figure is taken from Cayoglu et al. (2018c).

5.2.2 Traversal Methods

In order to ensure an ideal use of the IS a closer look at the traversal methods is necessary. Complementary traversal methods could improve predictions if more similar data points are traversed back to back. For this purpose three alternative traversal methods, besides the usual linear traversal, are suggested in this section (an example for each traversal method is given in Fig. 5.3):

Linear Traversal. The linear traversal processes the data along a predefined order of dimensions. For the given two dimensional example (Fig. 5.3) the order is to first traverse the x-axis and then the y-axis.

Chequerboard Traversal. The sequence based on the chequerboard traversal is structured like a mosaic. First every other value along the preferred dimension is traversed. Afterwards the remaining data points are traversed. As in the linear traversal an order must first be determined for the dimensions. In the example the order of dimensions is first x-axis and then y-axis.

Blossom Traversal. This traversal is structured like a blossoming rose which spreads around the starting point. Here, too, an order must be determined in which the dimensions will be processed. In the example the traversal follows a clockwise rotation starting at 12 o'clock.

Block Traversal. The block traversal follows a sequence around the starting point with the aim of building fully connected blocks. In the two dimensional case this may look like a spiral around the starting point. Again, an order for the dimension is considered.

5.2.3 Predictors

Each IC calculates a prediction for the data point to be predicted. There are several predictors that can be used for this. The following 12 predictors are implemented and evaluated in the experiments:

- `Akumuli` from Eugene (2017)
- `LastValue` from Goeman et al. (2002)
- `Stride`, `TwoStride`, and `Stride Confidence` from Goeman et al. (2002)
- `Ratana 3` and `Ratana 5` from Ratanaworabhan et al. (2006)
- `Pascal 1`, `Pascal 2`, ... , and `Pascal 5` (see Appendix B)

The details for the individual predictors can be found in the respective articles. Pascal x are predictor based on the Lagrange polynomials detailed in Appendix B. The different variations of the predictors Ratana **x** and Pascal **x** define the number of elements used for the prediction. In case of Ratana 3 this would be: $S_{i-3}^{i-1} = s_{i-3}s_{i-2}s_{i-1}$.

5.3 Experimental Setup

This section describes the simulation data used in the experiments and moves on to the metrics for evaluation and concludes with a description of the experiments.

5.3.1 Data

The data used for the compression experiments in this chapter is obtained from a climate simulation created by the ECMWF Hamburg/Modular Earth Submodel System Atmospheric Chemistry (EMAC) model (Jöckel et al., 2006). It consists of a 128×64 grid (longitude $\times$ latitude) with 47 vertical levels. The following temporal resolutions are used:

- One month (January, 2013) with 74 time steps (every 10 hours)

TABLE 5.1 Variables available in each dataset being used in the experiments.

Variable	Abbreviation
Specific Humidity	Spec. Hum.
Relative Humidity	Rel. Hum.
Pressure	Press.
Dry Air Temperature	Temp.
Zonal Wind (W-E)	Wind (W-E)
Meridional Wind (S-N)	Wind (S-N)

- One year (2013) with 365 time steps (every 24 hours)
- Fourteen years (2000-2013) with 168 time steps (every month)

The model output is given as single precision floating-point values. The selection of variables used are shown in Table 5.1.

5.3.2 Metrics

Metrics are necessary for evaluating the experimental results. Three metrics are used for the evaluation of the experiments: leading zero count (LZC), compression ratio (CR), and standard deviation (SD). LZC is a measure for the quality of the prediction. It represents the number of bits not needed to be saved on disk:

$$\mathrm{LZC}(r) = \#\text{Significant Zeros of } r + 1 \tag{5.4}$$

with r representing a residual according to Equation 2.14. Another metric used in the evaluation of the experimental results is the CR of the files* (see Eq. 2.17).

Further, the standard deviation (SD) of the LZC is calculated. The SD gives an indication for the vulnerability of the prediction to the structure of the data. The larger the SD, the more vulnerable the predictor is.

*Please note that this is not the final compression ratio, since the coding process creates additional overhead.

5.3.3 Experiments

Several experiments are conducted to investigate each step of the proposed algorithm.

Expt 1: Influence of Starting Point. First, the influence of different starting points on the compression rate are analysed. For this purpose, random blocks[†] with 1024 data points are build from each data set and for each variable. Then, ten starting points are chosen randomly and the blocks are compressed. This setup provides unbiased information on whether and how susceptible each predictor (and therefore the compression algorithm) is for changing starting positions of the applied traversal.

Expt 2: Traversal Order of Dimensions. Since most prediction methods use the linear traversal method (see Chapter 3), the second experiment analyses how the order of dimensions influences the CR. First, the data is split randomly into blocks with 1024 data points. Then, a list of every possible dimension ordering is generated. Afterwards, the data is traversed using linear traversal according to each of the permutations from the former list. This experiment provides information if the predictors need to adapt to the data.

Expt 3: New Traversal without the use of Information Spaces. In the third experiment the newly proposed traversal methods are tested. The predictors are not adjusted to the new traversal methods. Since most of the predictors consider the traversal sequence as a data stream, this change should cause changes in CR.

Expt 4: New Traversal with the use of Information Spaces. Finally, experiments with fully adjusted predictors to information space and the various consolidation methods are conducted. The Stride predictor is used as a fallback predictor, if no IS can be constructed (e.g. at the start or in case of an empty IC).

[†]Most compression algorithm split the data into several blocks and compress each separately to save time during decompression by decompressing only the requested blocks of data.

TABLE 5.2 [Expt 1] LZC±SD across predictors with varying starting points for daily, monthly and 10h dataset. The three highest SD for each dataset are highlighted.

	Daily	Monthly	10h
Akumuli	11.550 ± 0.041	12.890 ± 0.036	**12.730 ± 0.029**
Last Value	13.040 ± 0.004	14.200 ± 0.003	14.310 ± 0.003
Stride	13.300 ± 0.007	14.950 ± 0.006	14.240 ± 0.005
Stride Conf	**10.770 ± 0.056**	**11.190 ± 0.124**	**13.640 ± 0.053**
Stride 2	12.360 ± 0.011	13.590 ± 0.010	13.800 ± 0.006
Ratana 3	**12.760 ± 0.048**	**14.240 ± 0.038**	**13.840 ± 0.024**
Ratana 5	**12.760 ± 0.048**	**14.240 ± 0.039**	13.840 ± 0.023
Pascal 1	13.040 ± 0.004	14.190 ± 0.003	14.310 ± 0.003
Pascal 2	11.420 ± 0.005	12.570 ± 0.005	12.970 ± 0.004
Pascal 3	13.150 ± 0.009	14.780 ± 0.008	13.880 ± 0.006
Pascal 4	12.510 ± 0.011	14.200 ± 0.010	13.070 ± 0.008
Pascal 5	12.050 ± 0.014	13.450 ± 0.014	12.380 ± 0.010

5.4 Evaluation

In this section the results of the experiments are presented and evaluated.

5.4.1 Expt 1: Influence of Starting Point

The influence of different starting points on the LZC is depicted in Table 5.2. The achieved LZC seems to be independent of the initial value for most predictors since each SD is very low. The Stride Conf (SC) predictor has the highest SD in the monthly data record. Here, the LZC is 11.19 with a SD of 0.124 which is about 1.1%. Overall, the SD seems very low for any of the predictors, but the SC and Ratana **x** predictors seem to be the most prone. The SD of the remaining predictors are usually around 3‰. This is a magnitude lower than the ones of SC and Ratana.

This sensitivity can also be observed in the difference plots depicted in Figure 5.4. At Akumuli and Pascal the two starting points of $(11, 7)$ and $(5, 3)$ the different sequences are clearly visible. After a short time, the predictors calculate the same predictions as if the starting point had not changed. This is not the case with Stride Conf and Ratana. The differently calculated predictions are much more scattered and do not show a uniform pattern.

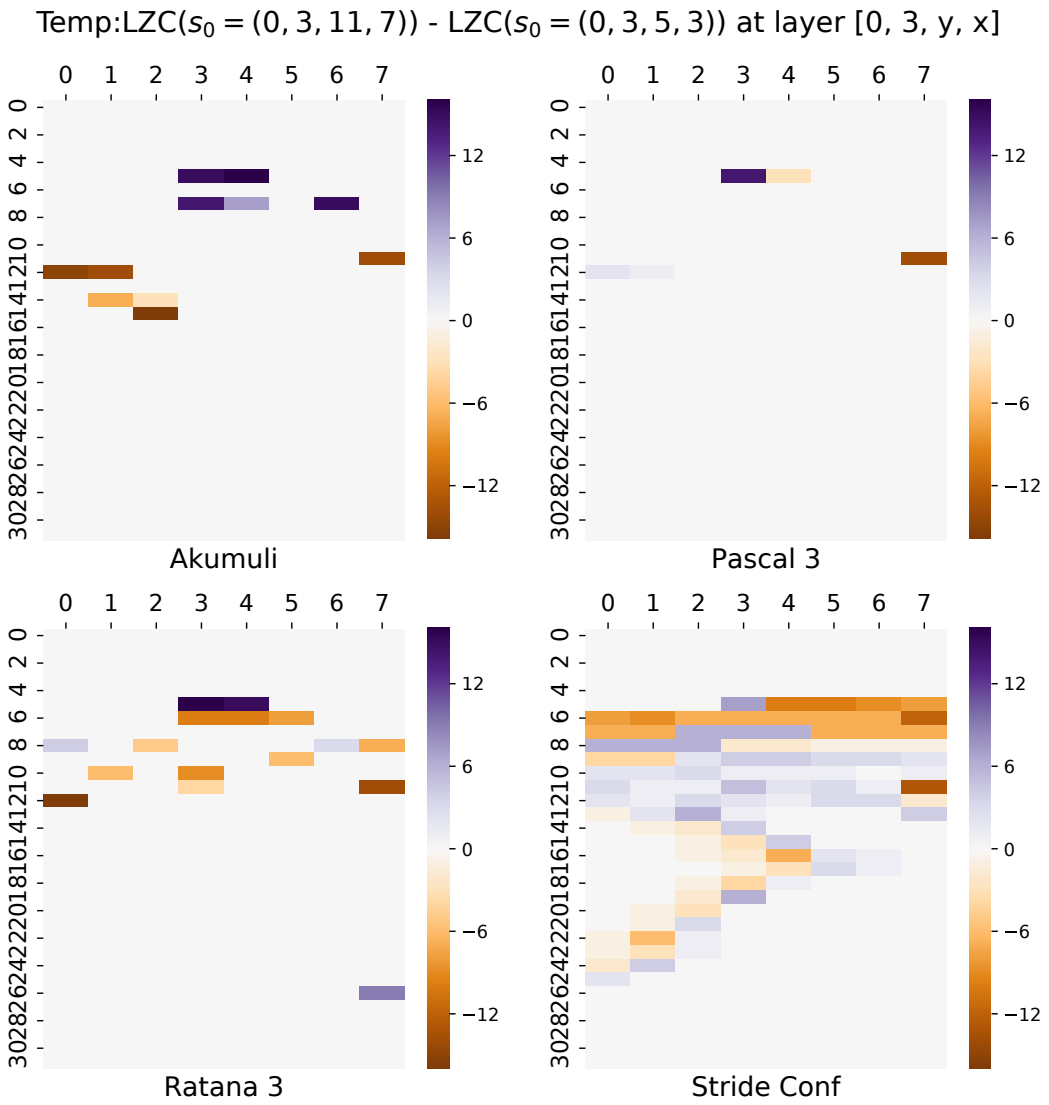


FIGURE 5.4 [Expt 1] Difference plot of LZC for two different starting points. The starting points are at $(11, 7)$ and $(5, 3)$ and the temperature variables is used. Difference plots are unitless. This figure is taken from Cayoglu et al. (2018c).

OBSERVATION 5.1 There is a steady increase of SD of Pascal **x** with increasing x. The more values are used for the prediction, the higher the fluctuation. This is valid across all datasets. The reason for this is that high order polynomials such as Pascal 4 and Pascal 5 lead to large local fluctuations and therefore worse extrapolation.

OBSERVATION 5.2 The starting point has little to no effect on the compression factor or rather the LZC. SC and the Ratana predictors seem to be the most prone to starting point changes. Overall, all predictors show good stability for starting point changes.

5.4.2 Expt 2: Traversal Order of Dimensions

In the second experiment the effects of the traversal order are analysed (see Table 5.3). The standard deviation increased several magnitudes, which confirms that a simple linear traversal through the data in an arbitrary order does not lead to success. In comparison to Expt 1 the LZC decreased by $10 - 40\%$ and SD increased significantly. SD reaches rates higher than 21% for Pascal 1. These fluctuations are illustrated in Figure 5.5. The figure suggests that the traversal order of the dimensions greatly influences the compression rate. This explains the high variance in the LZC depicted in Table 5.3. The LZC of the predictions are wildly disrupted for Pascal 3 and Ratana 3. The traversal order (0, 1, 2) seems to be almost consistently better than the order of (1, 2, 0) for Akumuli.

OBSERVATION 5.3 The predictors are more prone to changes regarding the traversal direction. The SD is several magnitudes stronger than with changes of the starting point. This vulnerability is shown by all the predictors. This supports the premise, that the context from which the predictions are made, must adapt to the data.

5.4.3 Expt 3: New Traversal without the use of Information Spaces

The effects of the various traversal methods on the LZC are analysed in the third experiment. In most cases the predictors are performing best using linear traversal. Only in two cases one of the new traversal methods performs better: The Akumuli predictor reaches a LZC of 5.360 using block

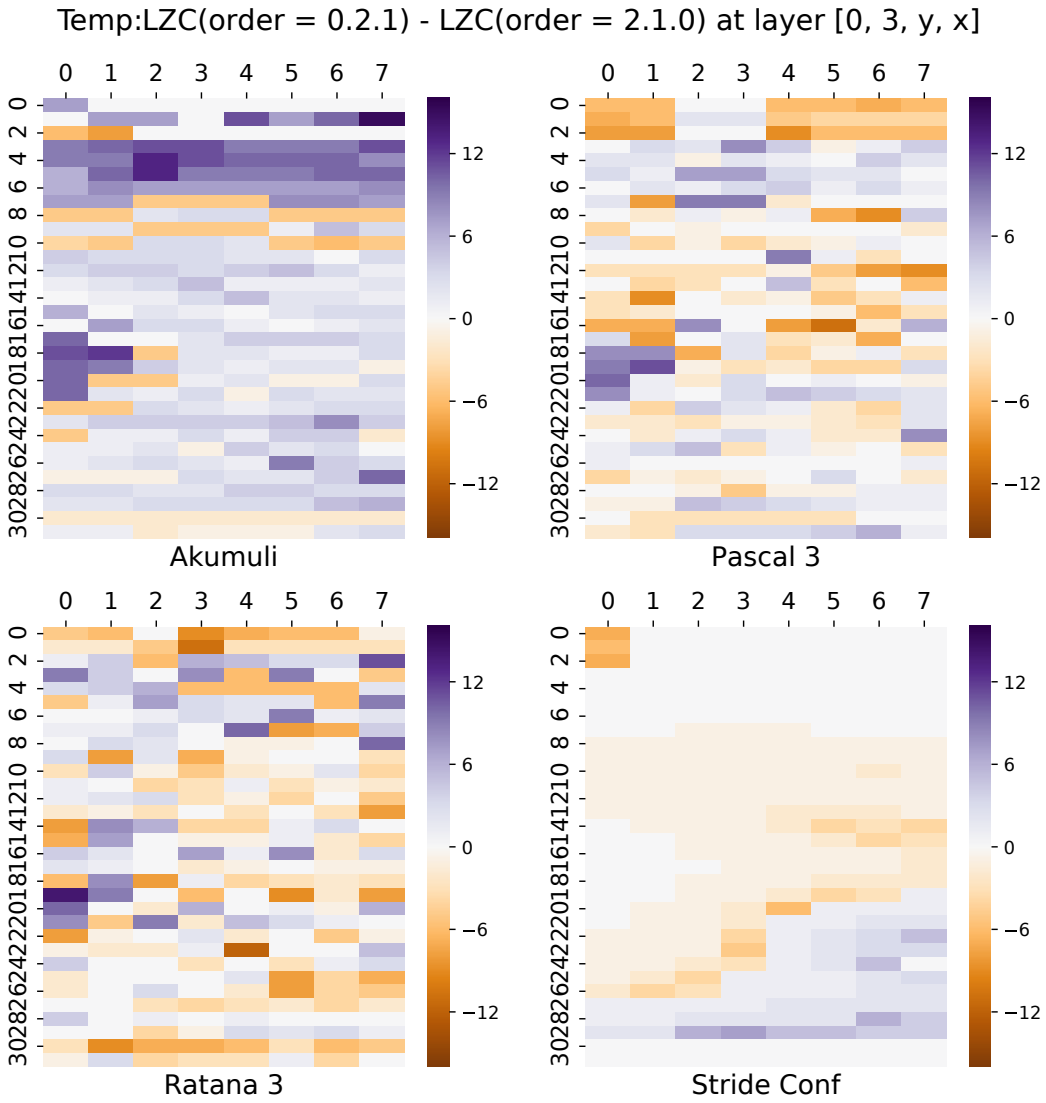


FIGURE 5.5 [Expt 2] Difference plot of LZC for traversal orders (0, 1, 2) and (1,2,0). This figure is taken from Cayoglu et al. (2018c).

TABLE 5.3 [Expt 2] LZC±SD across predictors for all possible dimension orders using linear traversal for daily, monthly and 10h dataset. The three highest SD for each dataset are highlighted.

	Daily	Monthly	10h
Akumuli	9.90 ± 0.98	9.81 ± 1.09	10.70 ± 0.57
Last Value	**10.82 ± 1.59**	**10.82 ± 1.55**	**10.87 ± 1.90**
Stride	10.95 ± 1.43	11.26 ± 1.50	10.71 ± 1.82
Stride Conf	9.53 ± 0.68	9.50 ± 0.66	9.88 ± 1.00
Stride 2	**10.27 ± 1.48**	**10.13 ± 1.56**	10.24 ± 1.88
Ratana 3	10.77 ± 1.19	10.79 ± 1.38	10.85 ± 1.60
Ratana 5	10.77 ± 1.19	10.79 ± 1.38	10.84 ± 1.60
Pascal 1	**10.82 ± 1.59**	**10.82 ± 1.55**	**10.87 ± 1.90**
Pascal 2	9.30 ± 1.47	**9.17 ± 1.58**	9.37 ± 1.83
Pascal 3	10.56 ± 1.22	10.86 ± 1.40	**10.07 ± 2.03**
Pascal 4	9.69 ± 1.27	9.87 ± 1.45	9.25 ± 2.00
Pascal 5	8.85 ± 1.27	9.13 ± 1.40	**8.34 ± 2.17**

traversal compared to 5.213 using linear traversal. Stride Conf reaches a LZC of 11.990 using block traversal compared to 11.843 using linear traversal. This suggests that the linear traversal (given the correct ordering) is a safe choice.

Figure 5.6 depicts the maximum reached LZC for each variable across predictors for the monthly dataset. While the linear traversal is several bits better than the other traversal methods, the results suggest an order for the other traversal methods: Block traversal is better than Blossom traversal, and Blossom traversal is better than Chequerboard traversal. There are several reasons for this:

Chequerboard. Due to the usage of every other data point in the first half of the algorithm (see Section 5.2.2) the data locality of the points in the sequence is scattered. This has more significance at the borders of the data cube since a jump might occur very often depending on the size of each dimension. The value differences caused by a jump from a long dimension to a smaller dimension can be greater than those between dimensions of the same size, since the travelled distance is greater.

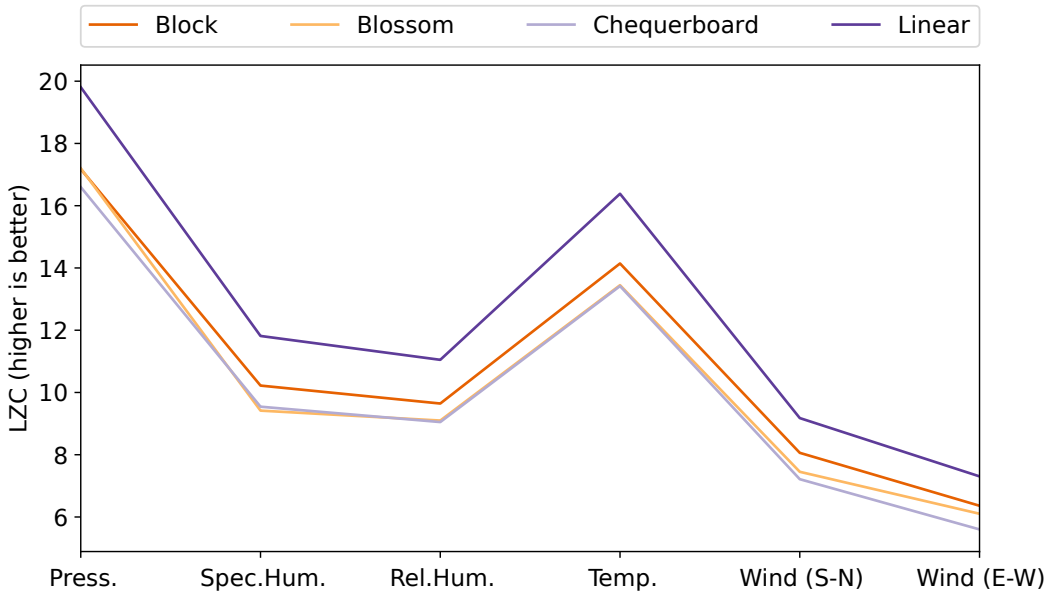


FIGURE 5.6 [Expt 3] Depicted is the maximum reached LZC for each variable in the daily dataset without the use of IS. This figure is taken from Cayoglu et al. (2018c).

Blossom vs Blocks. While the difference between both traversal algorithms is small, the data suggests that block traversal is reaching higher LZCs on average for the tested predictors. The reason for this is the building and prediction structure used by the block traversal. The number of interpolations compared to extrapolations is higher in the block traversal than in the blossom traversal.

5.4.4 Expt 4: New Traversal with the use of Information Spaces

Finally, the proposed method is evaluated in combination with the new traversal methods. An overview of the overall results is given in Table 5.4. The LZC increases by $9.6 \pm 0.4\,\%$ and the SD decreases by $23.5 \pm 0.9\,\%$ on average. For Pascal 5 the LZC climbs from 9.4 to 12.4 bits ($+31.4\%$, 10h dataset, LB consolidation) while the effective SD declines by 5.1%.

Performance of Individual Predictors. Next, all predictors are ranked by their LZC performance on each dataset. Each predictor compressed the data twice. Once with and once without the use of IS and the respective consolidation methods (see Section 5.2.1). The results are given in Table 5.5.

TABLE 5.4 [Expt 4] Changes on average due to the application of IS for each dataset with linear traversal

	10h	daily	monthly
ΔLZC	12.26%	9.03%	7.60%
ΔSD	−12.91%	−29.17%	−34.72%

TABLE 5.5 [Expt 4] Ranking of individual predictors for each dataset. Due to reasons of space the Pascal *x* predictor is abbreviated with P*x*. The consolidation method is given in round brackets, if the predictor used the proposed method.

10h		daily		monthly	
Predictor	LZC	Predictor	LZC	Predictor	LZC
1. P2 (LB)	13.29	1. P3 (LB)	13.33	1. P3 (LB)	15.88
2. P3 (LB)	13.13	2. P2 (LB)	13.26	2. P3 (R)	15.76
3. P1 (LB)	13.00	3. P3 (R)	13.11	3. P4 (LB)	15.75
⋮	⋮	⋮	⋮	⋮	⋮
17. P1	12.30	16. P2	12.48	12. P2	14.86
17. Last Value	12.30	18. Stride	12.47	13. Stride	14.84
23. P2	12.11	19. P1	12.40	14. P3	14.62

The highest LZCs are achieved by predictors using IS with 15.88 (monthly dataset), 13.33 (daily dataset) and 13.29 LZCs for the 10h dataset. These results are achieved with the Last Best (LB) consolidation method by Pascal 3 (for the monthly and daily dataset) and Pascal 2 (10h dataset).

The best predictors not using IS are ranked 12th (14.86 LZC, monthly), 16th (12.48 LZC, daily) and 17th (12.30, 10h) overall. While the best predictor for the monthly and daily dataset is Pascal 2, the best performance for the 10h dataset is achieved by Pascal 1.

Using IS helps Pascal 3 to improve the LZC from 11.40 to 13.13 LZC for the 10h dataset. It jumped from 50th to the 2nd place in the ranking. This is a huge gain considering the data are single precision values and the goal is lossless compression.

The results for each consolidation and traversal method for the daily dataset are presented in Figure 5.7. Table 5.8 lists the CR for each variable. For reasons of brevity, only the results of the daily dataset are displayed, since the results of the monthly and 10h datasets are similar.

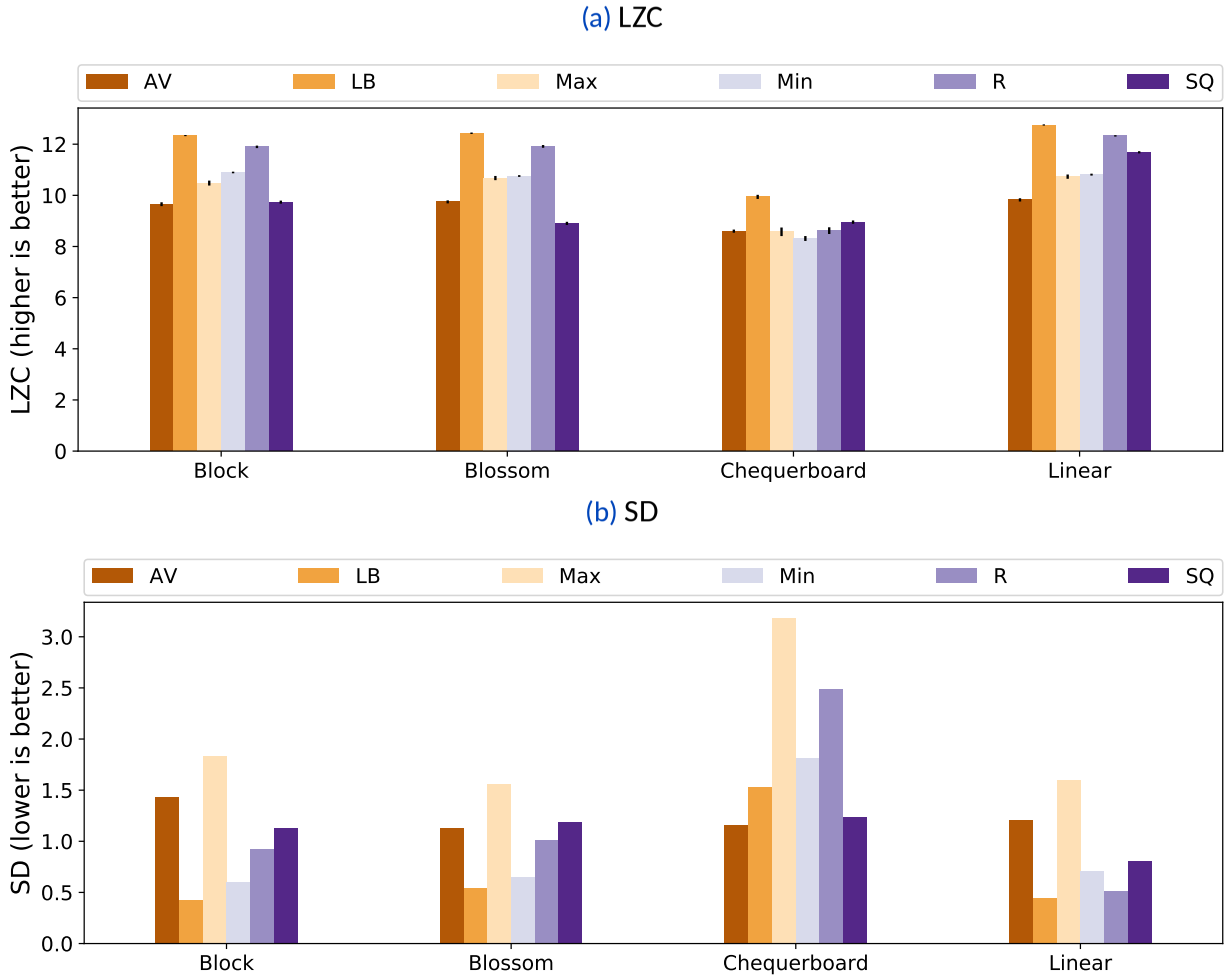


FIGURE 5.7 [Expt 4] LZC and SD of predictors using IS and not IS. Depicted are the mean leading zero counts/standard deviations across all variables for the daily dataset. Each bar represents a consolidation method described in Section 5.2.1 with *SQ* being the common method without the use of IS. This figure is taken from Cayoglu et al. (2018c).

TABLE 5.6 [Expt 4] LZC comparison (higher is better) of proposed consolidation methods and common method (SQ) with linear traversal. These results are obtained using the daily dataset.

	Block	Blossom	Cheq.	Linear
AV	70.69%	76.63%	67.04%	77.26%
LB	90.85%	105.82%	80.48%	107.55%
Max	73.08%	83.40%	65.86%	83.82%
Min	83.30%	89.74%	66.41%	90.16%
R	88.60%	100.77%	69.73%	105.07%
SQ	64.83%	69.70%	66.32%	100.00%

Comparing of Traversal Methods. In the following, the compression method using linear traversal and no IS is defined as the **common method**. The relative differences for each traversal method coupled with each consolidation method using IS compared with the common method is given in Table 5.6 for the LZC and in Table 5.7 for the SD. The result of the **common method** is depicted in the lower right corner.

The linear traversal delivers better LZC results than all the other traversal methods. The LB consolidation is the best performing consolidation method no matter which traversal method is used. Therefore, it is no surprise that using LB for consolidation and linear traversal end with the highest increase in LZC on average across all predictors with 7.55%. The LB consolidation also has less fluctuation in its results than most other traversal methods. Table 5.7 shows a reduction in SD of almost 50% for LB using linear and block traversal. The results are interesting since the order of performance suggested in Section 5.4.3 of Block > Blossom > Chequerboard does not seem to be valid any more. Chequerboard still performs worst, but the Blossom method outperforms Block in every case regarding the LZC.

Maximum and Minimum Consolidation. The Maximum and Minimum consolidation methods are introduced to gain information about possible biases of the predictions. Though the LZC of both methods is similar, the SD of Maximum is many times worse than that of the Minimum. Using Block traversal the SD of the Maximum method increases by a factor of three compared to that of the Minimum. This indicates that the predictors are somewhat biased against the minima.

TABLE 5.7 [Expt 4] SD comparison (lower is better) of proposed consolidation methods and common method (SQ) with linear traversal. These results are obtained using the daily dataset.

	Block	Blossom	Cheq.	Linear
AV	214.50%	167.36%	195.66%	178.82%
LB	50.47%	63.67%	223.30%	50.66%
Max	253.35%	212.16%	537.39%	215.39%
Min	80.04%	87.71%	316.41%	95.48%
R	112.90%	122.96%	418.52%	60.37%
SQ	168.09%	194.27%	201.06%	100.00%

Performance per Variable. In this section the performance of IS with respect to each individual variable is discussed, since the performance might depend on the variable under consideration. The results are represented in Table 5.8. A comparison of the best performing IS and the common method is depicted in Figure 5.8. In this comparison, there is not a single dominating consolidation method. The Reforced (R) and LB consolidation methods achieve the best CR for different variables. LB performs best regarding wind fields and the pressure variable. R performs best regarding humidity and temperature. There is no clear winner, but they always share the first two places among themselves. It should also be noted, that if LB is not performing best, it is always within $< .005$ bits. This is not the case for R. Here, the difference is on average ~ 0.113 bits.

OBSERVATION 5.4 Given the results of the previous experiments and the fact that LB is always close to R in cases where it is only the second best consolidation method, the experiments suggest using LB as the standard consolidation method and the linear traversal method as the standard traversal method in the future.

5.5 Summary

This chapter analyses the performance of different prediction-based compression algorithms on climate data.

TABLE 5.8 [Expt 4] Highest achieved compression rate using the best traversal and prediction method per variable for the daily dataset.

	Press.	Spec. Hum.	Rel. Hum.	Temp.	Wind (N-S)	Wind (E-W)
AV	0.376	0.659	0.673	0.520	0.736	0.795
LB	**0.337**	0.623	0.644	0.464	**0.681**	**0.740**
Max	0.350	0.657	0.671	0.512	0.729	0.775
Min	0.375	0.654	0.673	0.508	0.725	0.786
R	0.360	**0.619**	**0.641**	**0.459**	0.692	0.750
SQ	0.381	0.631	0.655	0.488	0.713	0.772

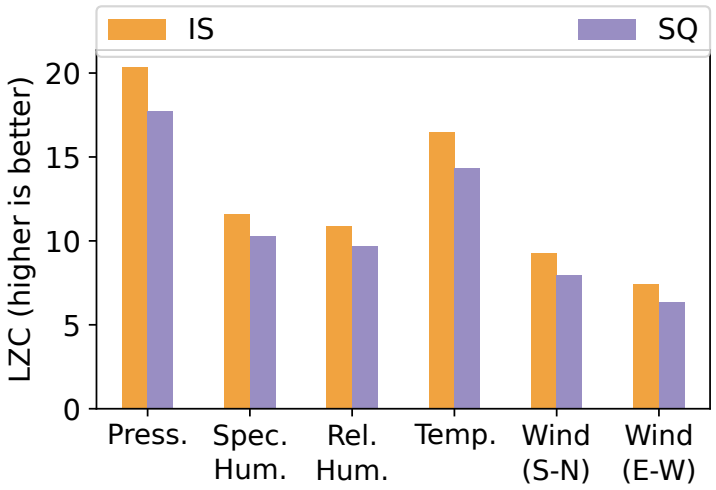


FIGURE 5.8 [Expt 4] LZC per variable using the best traversal and prediction method per variable for the daily dataset. This figure is taken from Cayoglu et al. (2018c).

The results show that changing the starting point of the compression algorithm has only negligible effects on the compression rate, while changing the traversal direction can influence the compression rate significantly. Information Spaces (IS) introduced in this chapter suggest that with the help of IS it is possible to improve the predictions of each predictor. More importantly, the stability of the predictions are increased. The Information Contexts which define the Information Space help consolidate information from several dimensions. This results in higher quality forecasts with less fluctuation than that of the common method.

The advantages of the proposed method are higher stability and better compression ratios. The use of IS increases the complexity of the process. The calculation of IS in each step is memory intensive and creates an overhead. However, the potential advantages of this new model have not yet been exhausted.

There are still different optimisation possibilities. For example, possible weights can be considered which can be used within the Information Contexts for the prediction. The individual Information Contexts can be evaluated by calculating a grading factor (such as information density) which allows deciding which IC to use or to avoid. The different layers of ICs could also be considered separately during the grading process. However, the current configuration achieves already a 10% improvement on LZC and decreases the standard deviation of the compression results by over 20% on average. The use of Information Spaces offers new possibilities and levers to further increase compression rates and gain independence from the internal structure of the data.

5.6 Code and Data Availability

The data and an implementation of the concepts described in this chapter are available under GNU GPLv3 license at `https://github.com/ucyo/informationspaces` (Cayoglu, 2018a).

CHAPTER 6

Data Approximation using ARIMA Models

The concept of IS introduced in the previous chapter helps adapt the compression algorithm to the data being compressed. In this chapter a method for lossy compression of time-series data is presented. This makes it possible to store existing knowledge about the interactions of climate variables in the encoder using established climate indices. Therefore the encoder can use this information to decorrelate the data. The contribution of this chapter helps tackle the third challenge described in Chapter 1. The results presented in this chapter have been published in parts in Cayoglu et al. (2017).

6.1 Motivation

Here, options for compressing environmental indices by using a statistical method based on the Auto Regressive Integrated Moving Average (ARIMA) model introduced in Box and Jenkins (1976) are investigated. The ARIMA model helps identify interdependencies in the dataset. The introduced method shows that it is possible to improve the accuracy of lossily compressed data by applying an adaptive compression method which preserves selected data with higher precision. It takes advantage of the interdependencies of the model and improves the correlation between the original and reconstructed data while using negligible more storage. Established environmental indices are saved using a lossy compression method for later use by the encoder. The indices are being used for seasonal forecasting of rainfall, temperature and monsoon precipitation. These indices help improve the decorrelation performed by the encoder.

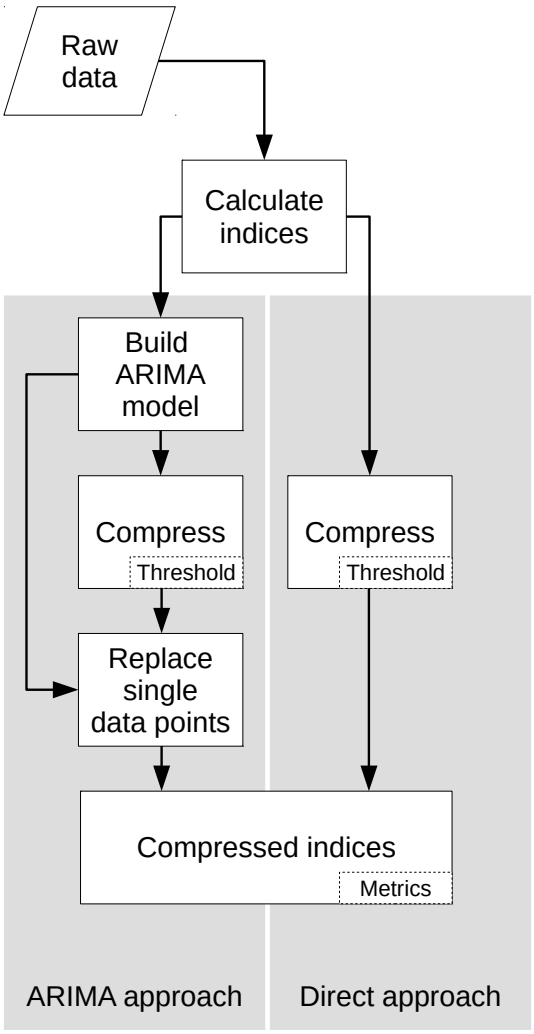


FIGURE 6.1 Flowchart of data compression using the proposed method (ARIMA approach) compared to the direct approach. This figure is taken from (Cayoglu et al., 2017).

6.2 Proposed Method

Two different approaches are used to obtain compressed indices. Figure 6.1 illustrates both approaches. The proposed method using an ARIMA model is depicted as 'ARIMA approach'. The second approach illustrates the usual process by applying compression directly on the indices and is described as 'Direct approach'.

After calculating the indices an ARIMA model is build for each index. The results from the ARIMA model are then compressed. After this step several data points are selected by replacement methods. These data points

are then replaced by ones with higher precision. The following sections describe the ARIMA model, the applied compression method, and the available replacement methods.

6.2.1 Model

The ARIMA model tries to find interdependencies in the dataset and was first introduced by Box and Jenkins (1976). It assumes that each datum in a time-series is dependent on its previous values and can be expressed by a function of its previous values. Because of the seasonal dependency in weather dynamics the proposed method uses a seasonal ARIMA model (Chattopadhyay and Chattopadhyay, 2014) for monthly and the original ARIMA model (Box and Jenkins, 1976) for daily data. The seasonal ARIMA model is being described by the following notation:

$$ARIMA(p, d, q)(P, D, Q)_s$$

with (p, d, q) representing the non-seasonal auto-regressive (p), difference (d) and moving-average (q) order and (P, D, Q) the equivalent seasonal order with period length s.

The general equation for seasonal ARIMA is as follows:

$$\Phi(B^s)\phi(B)(x_t - \mu) = \Theta(B^s)\theta(B)\varepsilon_t \tag{6.1}$$

with x_t representing the target value at time t, μ the expected mean value of the data, ε_t the error term of the model, and B^k the backpropagation with $B^k x_t = x_{t-k}$ and following components:

$$\begin{aligned}
\text{Seasonal AR} : \Phi(B^s) &= 1 - \Phi_1 B^s - \cdots - \Phi_P B^{P \cdot s} \\
\text{AR} : \phi(B) &= 1 - \phi_1 B - \cdots - \phi_p B^p \\
\text{Seasonal MA} : \Theta(B^s) &= 1 + \Theta_1 B^s + \cdots + \Theta_Q B^{Q \cdot s} \\
\text{MA} : \theta(B) &= 1 + \theta_1 B + \cdots + \theta_q B^q
\end{aligned}$$

with i representing the time step before the target value, $\Phi(B^s)$ the seasonal auto-regressive (AR) parameter, $\phi(B)$ the AR parameter, Φ_i the seasonal AR coefficients, ϕ_i the AR coefficients, $\Theta(B^s)$ the seasonal moving-average (MA)

parameter, $\theta(B)$ the MA parameter, Θ_i the seasonal MA coefficients and θ_i the MA coefficients of the model. The (partial) auto-correlation function (ACF) is used for analysing the data and the selection of the range of AR and MA parameters. ACF is defined as follows:

$$\text{ACF} = \frac{\sum_{t=k+1}^{n}(Y_t - \overline{Y})(Y_{t-k} - \overline{Y})}{\sum_{t=1}^{n}(Y_t - \overline{Y})^2} \tag{6.2}$$

with $k \in \mathbb{N}$ representing the temporal lag, Y_t time-series with start at time t and $\overline{Y}$ the mean value of the time-series. The coefficients Φ_i, ϕ_i, Θ_i and θ_i are optimised using the Akaike's Information Criterion (AIC) introduced in Akaike (1974).

6.2.2 Compression

The zfp compression method introduced in Lindstrom (2014) is being used for the compression of the indices. It has already been applied successfully on climate data (Baker et al., 2016) and supports lossy as well as lossless data compression. The following notation is being used throughout the thesis: `zfpPR`, where `PR` denotes the precision of the applied compression. In case of single precision floating-point numbers, a lossless compression would be denoted as `zfp32`.

6.2.3 Replacement Methods

The proposed ARIMA method improves compression by replacing several data points by ones with higher precision. Those points are chosen by the replacement methods described in this section.

Let $x^b = x_1^b x_2^b \dots x_n^b$ be a lossily compressed time-series with b representing the bits preserved from the original time-series. A lossless compression for single precision floating-point numbers would be depicted as x^{32} while the most lossy compression would be x^1. Further, let $k \in \mathbb{N}$ be the number of data points to be replaced, let $l \in \mathbb{N}$ be the number of additional precision bits to be saved and block size $bs = \max\{p, q\}$ represent either the auto-regressive or moving-average order of the ARIMA model. The parameter bs helps identify data contributing to the calculation of a datum x_i^b. The updated time-series is represented by $\hat{x} = \hat{x}_1 \hat{x}_2 \dots \hat{x}_n$. Further on let $sort(X)$

be the sorted set of X, $argsort(X)$ the arguments of the sorted set of X and $S(t, X)$ the t previous values of each element of X:

$$sort(X) = \left\{ x_i \mid x_i \leq x_{i+1} \wedge x_i \in X \right\} \tag{6.3}$$

$$argsort(X) = \left\{ \arg x_i \mid x_i \leq x_{i+1} \wedge x_i \in X \right\} \tag{6.4}$$

$$S(t, X) = \left\{ x - j \mid j \leq t \wedge j \in \mathbb{N} \wedge x \in X \right\} \tag{6.5}$$

The algorithm differentiates between the following methods to choose the data points being replaced.

First. The first k values are replaced.

$$\hat{x}_i = \begin{cases} x_i^{b+l} & \text{if } i \leq k \\ x_i^b & \text{else} \end{cases} \tag{6.6}$$

Even. The k values being replaced are evenly distributed over the whole time-series. The time-series is split in $bl = \left\lfloor \frac{k}{bs} \right\rfloor + 1$ evenly distributed blocks with size $\left\lfloor \frac{n}{bl} \right\rfloor$ and M midpoints.

$$\begin{aligned} M &= \left\{ j \cdot \left\lfloor \frac{n}{bl} \right\rfloor \mid j \in \mathbb{N} \wedge j \leq bs \right\} \\ \hat{x}_i &= \begin{cases} x_i^{b+l} & \text{if } i \in [m - \left\lfloor \frac{bs}{2} \right\rfloor, \ldots, m + \left\lfloor \frac{bs}{2} \right\rfloor) \text{ with } m \in M \\ x_i^b & \text{else} \end{cases} \end{aligned} \tag{6.7}$$

Special. The cumulative correlation (Eq. 6.8) of the time-series is calculated, the results sorted and those data points replaced, which contribute to the data with the lowest correlation.

$$C = \left\{ r_{1,j} \mid j \in \mathbb{N} \wedge j \leq n \right\} \tag{6.8}$$

$$\begin{aligned} C' &= argsort(C) \\ \hat{x}_i &= \begin{cases} x_i^{b+l} & \text{if } \arg i \leq k \text{ with } i \in S(bs, C') \\ x_i^b & \text{else} \end{cases} \end{aligned} \tag{6.9}$$

Rolling. The rolling correlation (Eq. 6.10) with window size bs is calculated, the results sorted and those data points replaced which contribute to the data with the lowest correlation.

$$R = \left\{ r_{j-bs,j} \mid j \in \mathbb{N} \wedge bs < j \leq n \right\} \tag{6.10}$$

$$R' = argsort(R)$$

$$\hat{x}_i = \begin{cases} x_i^{b+l} & \text{if } \arg i \leq k \text{ with } i \in S(bs, R') \\ x_i^b & \text{else} \end{cases} \tag{6.11}$$

Cumcorr. The cumulative correlation of the time-series is calculated (Eq. 6.8) and the datum identified which is followed by the biggest consecutive drop in correlation. The data responsible for this datum is then replaced. Afterwards the process will be repeated until k data points are replaced.

$$C = \left\{ r_{1,j} \mid j \in \mathbb{N} \wedge j \leq n \right\} \tag{6.12}$$

$$C' = \begin{cases} c_i & \text{if } c_{i+1} \geq 0 \\ \sum_{j=0}^{b} c_{i+j} & \text{else with } c_{i+j} < 0 \wedge b \in \mathbb{N} \end{cases} \tag{6.13}$$

$$C'' = \text{argsort}(C')$$

$$\hat{x}_i = \begin{cases} x_i^{b+l} & \text{if } \arg i \leq k \text{ with } i \in S(bs, C'') \\ x_i^b & \text{else} \end{cases} \tag{6.14}$$

6.3 Experimental Setup

The following sections describe the steps to create the climate indices, the metrics used and the conducted experiments.

6.3.1 Data

The data used in this chapter is obtained from a simulation with the ECMWF Hamburg/Modular Earth Submodel System Atmospheric Chemistry (EMAC) (Jöckel et al., 2006) model. It consists of a 128×64 (longitude, latitude) horizontal grid with six vertical levels (from 1000 hPa to 10 hPa) and spans a time period from the beginning of 1979 till the end of 2013 with

TABLE 6.1 Spatial borders and variables used for calculating indices with temperature (T), pressure (p) and westerly wind (u).

Index	Variable	Lat [°N]	Lon [°E]	Lev [hPa]
ENSO34	T	−5 to 5	190 to 240	1000
QBO**x**	u	−5 to 5	0 to 360	**indicated by x**
NAO	p	Lisbon and Reykjavík		1000

10h time steps. The following variables are available as single precision floating-point values: ozone, pressure, dry air temperature and westerly wind.

The following climate indices are created for investigation: El Niño Southern Oscillation 3.4 (ENSO34), North Atlantic Oscillation (NAO), Quasi-Biennial Oscillation (QBO) at 30 (QBO30) and 50 hPa (QBO50). These indices show high significance in climate research (de Guenni et al., 2017; Nowack et al., 2017; Hurrell and Van Loon, 1997; Hurwitz et al., 2011) and help in numerical weather predictions and seasonal forecasting. ENSO34 is being used in forecasting rainfall, NAO in forecasting seasonal temperature for Europe and QBO is being used for predicting monsoon precipitation. Each index is created with two temporal resolutions: monthly and daily. For the calculation of ENSO34 and QBO**x** a spatial subset of the data according to Table 6.1 is selected. Next the zonal and meridional means are calculated. The NAO index is calculated using the difference in surface pressure between Lisbon and Reykjavík. Afterwards yearly monthly mean and multi-year monthly mean for all indices are calculated. The multi-year monthly mean is then subtracted from the corresponding yearly monthly mean and divided by the multi-year standard deviation for each month. This concludes the process for the monthly indices. For the daily indices these steps are repeated with respective daily resolution. A histogram of each index is depicted in Figure 6.2 and a summary of their characteristics are given in Table 6.2.

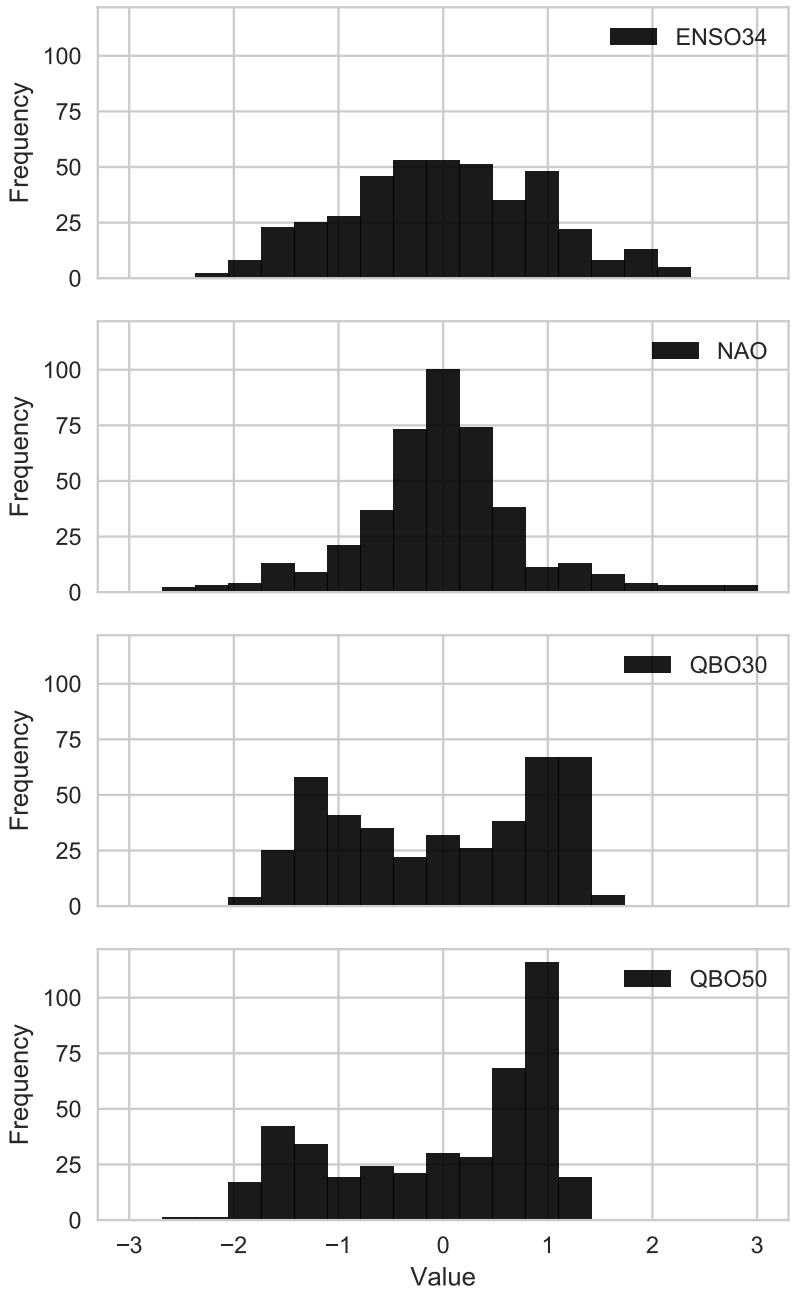


FIGURE 6.2 Histogram of each weather index in monthly resolution. The indices are dimensionless. This figure is taken from Cayoglu et al. (2017).

TABLE 6.2 Information about the monthly indices.

	ENSO34	NAO	QBO30	QBO50
mean	0.000	0.000	0.000	0.000
std	0.936	0.822	0.997	0.995
min	−2.208	−2.626	−1.876	−2.464
25%	−0.656	−0.421	−1.011	−0.934
50%	0.001	−0.016	0.097	0.364
75%	0.693	0.397	0.999	0.869
max	2.273	3.097	1.472	1.383
skewness	0.028	0.393	−0.161	−0.565
kurtosis	−0.481	2.055	−1.474	−1.149

6.3.2 Metrics

For evaluating the forecasting model the Root Mean Square Deviation (RMSD) is used. The reconstructed index from the lossy compression is evaluated using the Pearson correlation coefficient $r_{s,e}$ (Pearson, 1896):

$$r_{s,e} = \frac{\sum_{i=s}^{e}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=s}^{e}(x_i - \bar{x})^2}\sqrt{\sum_{i=s}^{e}(y_i - \bar{y})^2}} \tag{6.15}$$

with s and e representing the starting and respectively ending indices of the time-series, x_i representing the original value, $\overline{x}$ the original mean value, y_i the reconstructed value, and $\overline{y}$ the reconstructed mean value. The reason for choosing the Pearson correlation coefficient as a metric is that most of the time the correlation between the index and other weather phenomena is being analysed. Therefore it is of utmost importance to reconstruct an index correlated to the original index. The compression quality is measured using the **bits per float** (bpf) metric (see Eq. 2.19) and the CR (see Eq. 2.17).

6.3.3 Experiments

Several experiments are carried out to investigate possible compression methods. The first experiment focuses on lossless compression of the indices. Since the datasets are single precision floating-point data `zfp32` is used for compression.

TABLE 6.3 Results of (seasonal) ARIMA model run for monthly and daily data.

Index	Model	RMSD
Monthly data		
ENSO34	$ARIMA(3,0,2)(1,0,0)_{12}$	5.067e08
NAO	$ARIMA(1,0,0)(1,0,0)_{12}$	8.195e09
QBO30	$ARIMA(2,0,3)(1,0,0)_{12}$	1.088e07
QBO50	$ARIMA(1,1,1)(1,0,1)_{12}$	2.909e06
Daily data		
ENSO34	$ARIMA(5,2,4)(0,0,0)_{0}$	4.686e04
NAO	$ARIMA(2,0,2)(0,0,0)_{0}$	1.440e07
QBO30	$ARIMA(5,0,4)(0,0,0)_{0}$	1.084e07
QBO50	$ARIMA(5,0,4)(0,0,0)_{0}$	4.488e08

Furthermore, a lossy compression is analysed with the aim of achieving the smallest possible deviation for a given error limit. For this experiment an error bound of $\tau = 1\text{e}{-5}$ is set so that $r_{1,n} \geq 1.0 - \tau$ is always satisfied.

Finally, a third experiment is conducted to analyse what effect a gradual decline in precision from `zfp32` to `zfp01` has on the correlation coefficient. Further, it is analysed if replacing several data points with a higher precision improves the correlation coefficient. These indices with updated data are described by the following notation: `zfpPR+l` with l representing the number of additional precision bits. The notation `zfp06+02` depicts a lossy compression method with six precision bits where several data points have two additional bits of precision. For the following experiments five and ten percent of the data with $l \in \{1, 2, 3\}$ are replaced.

6.4 Evaluation

6.4.1 Model

Since ARIMA can only be applied to stationary data, the Dickey-Fuller-Test (DF-test) introduced by Dickey and Fuller (1979) is conducted to analyse the indices for stationariness. All indices are stationary with a confidence level of 99%. The results of the DF-test are represented in Table 6.4.

TABLE 6.4 Results of DF-test.

	ENSO34	NAO	QBO30	QBO50
DFT Test Statistic	−5.341	−17.571	−7.447	−9.257
Critical Value (1%)	−3.446	−3.446	−3.447	−3.447
Critical Value (5%)	−2.869	−2.868	−2.869	−2.869
Critical Value (10%)	−2.571	−2.570	−2.571	−2.571

TABLE 6.5 BPF of lossless compression of daily and monthly data for the residuals of the ARIMA model and direct approach. CR > 32 bits are highlighted red, since this means the compressed file is actually bigger than the original data. Header files are excluded.

	Monthly data		Daily data	
Index	ARIMA	Direct	ARIMA	Direct
ENSO34	**33.371**	**32.762**	**33.072**	**32.300**
NAO	**33.371**	**33.067**	**33.071**	**32.821**
QBO30	**33.219**	**32.152**	**33.031**	**30.753**
QBO50	**33.451**	**32.457**	**33.051**	**31.009**

The (seasonal) ARIMA model can reconstruct all indices with good accuracy. The RMSD of the reconstructed indices for monthly data is better than the one for the daily data. ARIMA models with differentiation steps, QBO50 for monthly data and ENSO34 for daily data, perform worst in their respective group. Detailed results are described in Table 6.3. The Pearson correlation coefficient $r_{1,n}$ for all indices is $1.0 \pm 2\text{e}{-12}$. Figure 6.3 illustrates the ARIMA model for NAO and QBO30. It can be seen, that the reconstructed index defined by the ARIMA model represents the original index very well.

6.4.2 Compression

First, the ARIMA approach without replacements is being compared with the direct approach. In Section 6.4.3 the results of the proposed algorithm including the replacement methods is compared with the direct approach.

Lossless. The experimental results show that lossless compression of the ARIMA output is resulting in bigger files than without compression. A lossless compression applied directly on the indices returns similar results. The only exceptions are the QBO30 and QBO50 indices at daily resolution. The

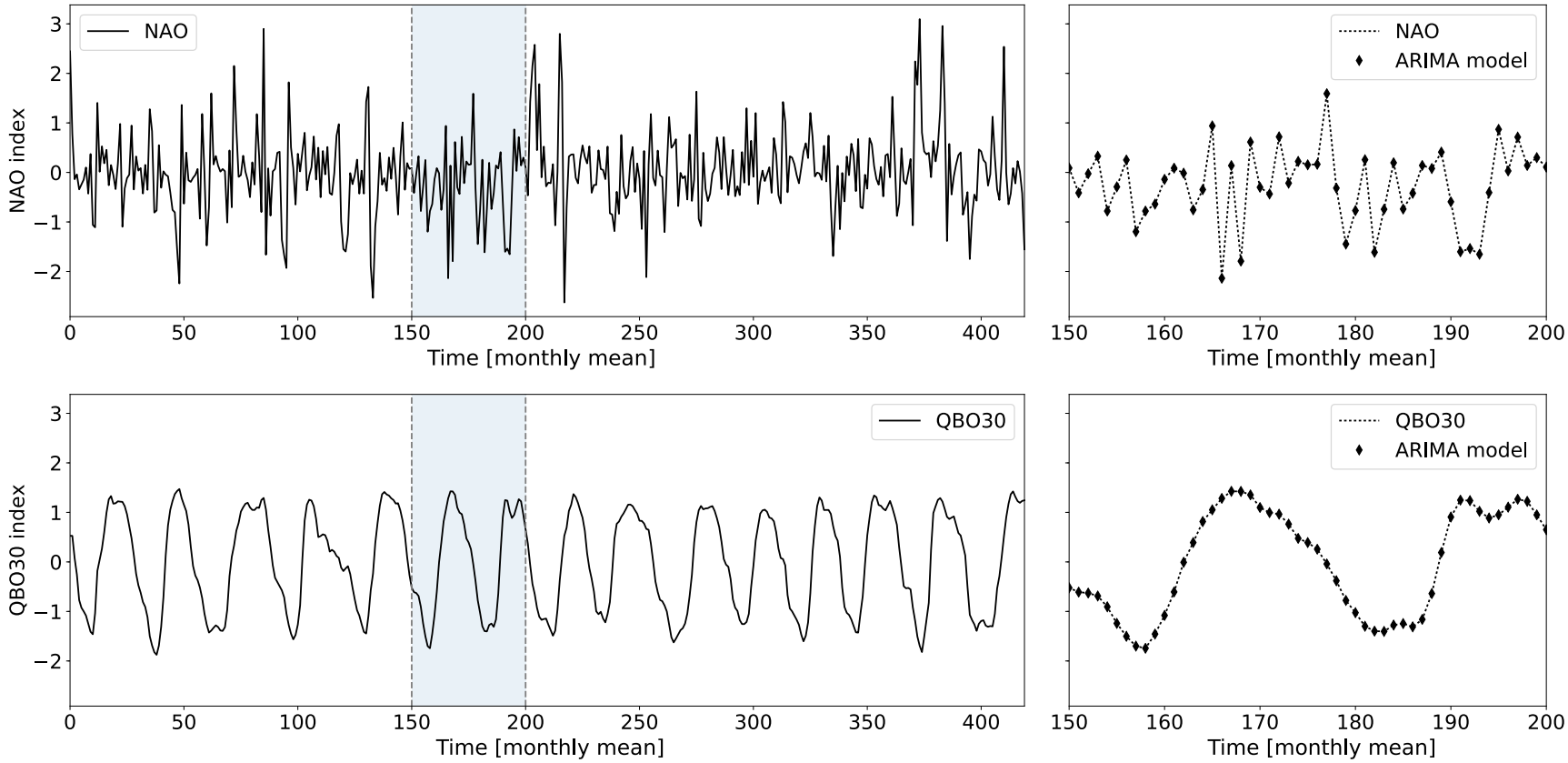


FIGURE 6.3 NAO (top) and QBO (bottom) indices and their reconstruction via the ARIMA model. Error bars have been omitted for plotting purposes. This figure is taken from Cayoglu et al. (2017).

TABLE 6.6 CR for lossy compression of daily and monthly data with $\tau = 1e{-}5$ as error threshold. Header files are excluded.

	Monthly data		Daily data	
Index	ARIMA	Direct	ARIMA	Direct
ENSO34	0.386	0.371	0.658	0.322
NAO	0.386	0.386	0.377	0.370
QBO30	0.381	0.357	0.376	0.273
QBO50	0.668	0.362	0.377	0.281

file size of the QBO30 and QBO50 daily data is decreasing by four percent for QBO30 and three percent for QBO50. Detailed results are presented in Table 6.5.

Strict Lossy Compression. Lossy compression with $\tau = 1e{-}5$ achieves in most cases a CR of ~ 0.4. The ARIMA approach achieves a CR of 0.381 on average for monthly and daily data. The only exceptions are QBO50 (monthly) and ENSO34 (daily) which reach a CR of ~ 0.663. Detailed results are presented in Table 6.6.

This deviation is due to the differentiation step during model building. This additional calculation step increases error propagation which results in additional precision bits needed to meet the threshold τ.

It is no surprise that the direct method achieves smaller CRs for these two indices. The compression ratio of QBO50 (monthly) is down from 0.668 to 0.362 and ENSO34 (daily) from 0.658 to 0.322. While applying lossy compression directly on the monthly indices does not improve the NAO index, the CR for QBO30 and ENSO34 are slightly improved by three and one percent, respectively.

The daily data show better results on average (from 0.37 to 0.30). Only the NAO (monthly) index does not show any decrease or increase regarding CR by the direct approach.

Lossy Compression with Gradual Decline. The third and last experiment is conducted to analyse the effects of a more and more aggressive lossy compression method. The precision level of the lossy compression algorithm is gradually reduced from `zfp32` to `zfp01`.

Applying lossy compression on daily and monthly data directly shows that the NAO index is performing worst regarding CR. The results of the ARIMA approach are similar to the strict lossy compression. It looks like the difference step of the ARIMA model (see Table 6.3) has a negative effect on the correlation.

OBSERVATION 6.1 The results until now suggest that the additional calculation steps needed for generating the ARIMA model have a negative effect on the compressed indices. This effect is expected due to the interdependence in the ARIMA model and thus the error propagation. The accuracy of each datum depends on all calculations done up to that point. The later the datum in the time-series, the greater is the effect of calculation errors. This effect is even more significant if an ARIMA model with a differentiation step is used. This can be seen in the CR of QBO50 in the monthly dataset and ENSO34 in the daily dataset (see Table 6.6). Overall, the ARIMA approach (without replacement) is causing a $1-3\%$ loss in storage space for the monthly indices and 10% for the daily indices.

OBSERVATION 6.2 Most interesting are the results for the NAO index. While the other indices show similar behaviour in gain and loss of CR with both approaches, the NAO index does not. The direct and ARIMA approach have no effect on the CR of the monthly indices and only negligible effect on the daily indices with 0.007 difference in CR. A closer look at the index (Table 6.2 and Figure 6.2) reveals properties unique for NAO which can provide an explanation for its behaviour. The standard deviation of the NAO index is the lowest with 0.822 and the first and third quartile are the closest to the mean with -0.421 and 0.397. Additionally, the NAO index has several outliers. The minimum and maximum have the highest absolute distance to the mean regarding of all indices. In addition to these properties, the NAO index shows an unbiased skewness and kurtosis (see Table 6.2). The NAO index is heavy tailed with a slight asymmetry on the right tail.

This unique position of NAO can also be observed in conjunction with the replacement methods. While the ENSO34, QBO50 and QBO30 indices behave similar to each other, NAO does not. Because of these similarities only the replacement results for NAO and QBO30 are presented in the next section.

6.4.3 Replacement Methods

The results of the former section suggest that CR and $r_{1,n}$ for a lossy compression algorithm using the ARIMA approach are worse than for the direct approach. This is due to the interdependency of the data. If one or several of these data points deviate too far from its original datum, then it will negatively effect the calculation of the following data. But there is a possibility to use this interdependency for the benefit of the method. If those data points which have a negative impact on the reconstruction of each index can be identified, they can then be replaced by ones with higher precision. In the following, first the indices reconstructed by the different replacement methods are compared with the original ARIMA output. Afterwards, they are compared with the directly compressed indices.

Replacement of 5% and 10% of Data. Several tests are carried out to analyse how many data points need to be replaced to see an effect on the correlation coefficient $r_{1,n}$. Table 6.7 illustrates the effects for the monthly indices. Most of the time the gain in correlation by replacing ten instead of five percent of the data is negligible. There are two exceptions to this: The increase in correlation from 0.468 to 0.624 with `zfp02+01` on the NAO index and the increase from 0.691 to 0.935 with `zfp04+01` on the QBO30 index. It should be pointed out that the correlation value of 0.935 with `zfp04+01` is almost as good as using `zfp05` for the whole index which has a correlation coefficient of 0.972.

A more striking and disappointing finding is that replacing data with higher precision did not always increase the correlation coefficient. While the NAO index showed no decline, the correlation coefficient of QBO30 dropped in several cases. Most of the time the drops where < 0.01, but the most significant drop was for QBO30 from 0.139 to 0.027 with `zfp02+01` which is a drop of $\sim 80\%$. Further research is needed to analyse why these drops occur in the lowest precision level. Figure 6.4 illustrates the correlation coefficient of each replacement method from `zfp02+01` to `zfp06+03`.

Replacement Methods. Figure 6.5 illustrates the correlation coefficient $r_{1,t}$ at month t for `zfp06+03` with ten percent replacement. The NAO index is represented best by the special method. The reason for this seems twofold:

TABLE 6.7 Pearson correlation by replacing five and ten percent of the monthly indices. Replacements causing a worsening of the Pearson correlation are highlighted red and those resulting in an improvement are highlighted green. The replacement method being used is 'Special'.

		zfp02	zfp03	zfp04	zfp05	zfp06
NAO (5%)	$l = 0$	0.354	0.725	0.924	0.979	0.994
	$l = 1$	**0.468**	**0.825**	**0.950**	**0.983**	**0.996**
	$l = 2$	**0.506**	**0.826**	**0.952**	**0.984**	**0.996**
	$l = 3$	**0.519**	**0.831**	**0.953**	**0.984**	**0.996**
NAO (10%)	$l = 0$	0.354	0.725	0.924	0.979	0.994
	$l = 1$	**0.624**	**0.864**	**0.959**	**0.987**	**0.997**
	$l = 2$	**0.692**	**0.870**	**0.964**	**0.989**	**0.997**
	$l = 3$	**0.705**	**0.878**	**0.965**	**0.989**	**0.997**
QBO (5%)	$l = 0$	0.139	0.482	0.635	0.972	0.986
	$l = 1$	**0.027**	**0.566**	**0.691**	**0.967**	**0.996**
	$l = 2$	**0.039**	**0.591**	**0.692**	**0.973**	**0.993**
	$l = 3$	**0.042**	**0.596**	**0.677**	**0.985**	**0.995**
QBO (10%)	$l = 0$	0.139	0.482	0.635	0.972	0.986
	$l = 1$	**0.050**	**0.575**	**0.935**	**0.968**	**0.996**
	$l = 2$	**0.082**	**0.615**	**0.940**	**0.973**	**0.993**
	$l = 3$	**0.084**	**0.607**	**0.944**	**0.987**	**0.996**

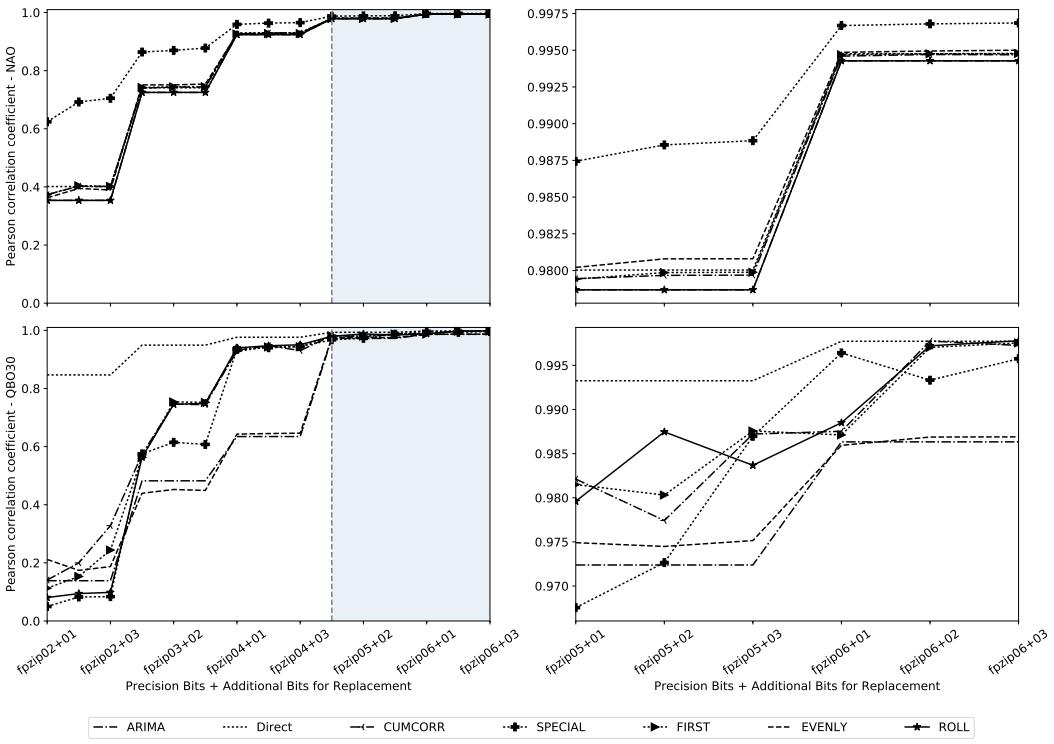


FIGURE 6.4 Pearson correlation with ten percent replacement. Replacement methods from `zfp02+01` to `zfp06+03` (left). A zoom on the replacement methods with $r_{1,n} > 0.97$ (right). This figure is taken from Cayoglu et al. (2017).

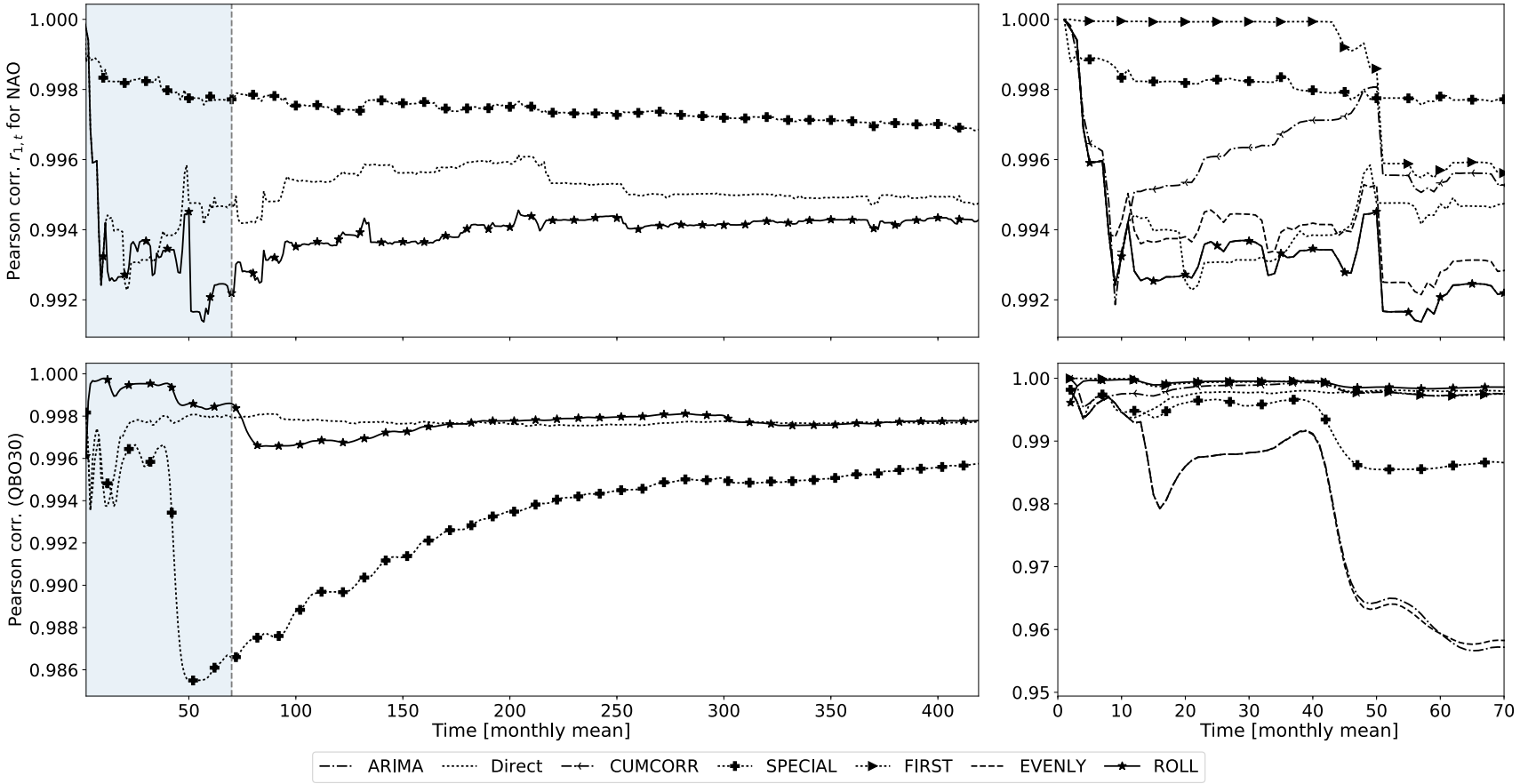


FIGURE 6.5 Pearson correlation $r_{1,t}$ for NAO and QBO30 with `zfp06+03` lossy compression on ten percent of the data. The left part depicts the whole index, while the right part only the first 70 months. For illustration purposes the left figures only show the direct approach and ARIMA approach improved by the rolling and special replacement methods. The right figure illustrates all methods. This figure is taken from Cayoglu et al. (2017).

First, the special method decides which datum to replace depending on the lowest correlation coefficient. The correlation coefficient at each time step is sorted and those data replaced, which contribute to the lowest correlation. With this property the special method can compensate best for sudden changes in the index. Especially, the first drop at the beginning of the NAO index and the one at $t = 50$ are not having as big of an impact on the correlation coefficient with the special replacement method compared to the others. This is illustrated on the right of Figure 6.5.

Second, the model being used for the NAO index is $ARIMA(1,0,0)(1,0,0)_{12}$. Every single datum is only depending on its immediate predecessor and the one at the same time the previous year. A single datum is only depending on two previous values. This small dependence helps in (1) correcting more data points and (2) limiting error propagation.

The ARIMA approach improves the reconstruction of the NAO index significantly. The reconstruction has a better correlation coefficient on each time step t than the direct approach with only using negligible more storage space (see Table 6.8). For the QBO30 index the rolling method has the highest correlation coefficient (see Figure 6.5). The rolling method calculates the rolling correlation coefficient with window size $bs = max\{p, q\}$ where p describes the auto-regressive and q the moving-average of the ARIMA model. The coefficients are then sorted and those data replaced which contribute to the data with the lowest correlation.

Unfortunately, in the case of the QBO30 index the ARIMA approach is not consistently better. In the beginning of the time-series with $t < 50$ it performs significantly better: The direct approach drops to 0.991 while the ARIMA approach stays constantly above 0.997. Afterwards the direct approach performs better until $t = 175$ where the ARIMA approach starts to outperform the direct approach again.

For the daily indices the results are different. Because of the increased number of calculation steps, the error propagation has a more severe impact. While the correlation coefficient for the direct approach is at 0.999 for QBO30 and 0.997 for NAO, the best ARIMA approach can only achieve 0.994 for QBO30 and 0.996 for NAO. Table 6.9 shows the results for `zfp06+03` on daily and monthly data.

TABLE 6.8 CR for NAO and QBO30 index after invocation of the ARIMA approach and replacing ten percent of the data. Header files are excluded.

		zfp02	zfp03	zfp04	zfp05	zfp06
NAO (5%)	$l = 0$	0.094	0.120	0.150	0.182	0.213
	$l = 1$	0.097	0.123	0.153	0.185	0.217
	$l = 2$	0.100	0.126	0.156	0.188	0.220
	$l = 3$	0.104	0.129	0.159	0.191	0.223
NAO (10%)	$l = 0$	0.100	0.133	0.167	0.200	0.229
	$l = 1$	0.103	0.137	0.170	0.203	0.232
	$l = 2$	0.106	0.140	0.173	0.206	0.235
	$l = 3$	0.110	0.143	0.176	0.209	0.238
QBO (5%)	$l = 0$	0.099	0.117	0.134	0.151	0.169
	$l = 1$	0.103	0.121	0.137	0.154	0.172
	$l = 2$	0.106	0.124	0.140	0.157	0.176
	$l = 3$	0.109	0.127	0.143	0.160	0.179
QBO (10%)	$l = 0$	0.105	0.124	0.148	0.171	0.200
	$l = 1$	0.108	0.127	0.151	0.174	0.203
	$l = 2$	0.111	0.130	0.154	0.178	0.206
	$l = 3$	0.114	0.133	0.157	0.180	0.210

TABLE 6.9 Correlation coefficient for zfp06+03 for daily and monthly data.

	Monthly data		Daily data	
	NAO	QBO30	NAO	QBO30
First	0.994 78	0.997 55	0.996 00	0.994 04
Even	0.995 00	0.986 90	**0.996 11**	0.988 14
Special	**0.996 86**	0.995 75	0.995 98	0.988 99
Rolling	0.994 28	**0.997 79**	0.996 08	**0.994 09**
Cumcorr	0.994 69	0.997 26	0.995 98	0.984 00
Direct	0.994 76	0.997 74	0.996 69	0.999 38

Effects on Storage Space. Until now only the impact on the Pearson correlation coefficient is analysed. But the additional precision bits used by the replacement methods have a negative impact on the CR. The effects on the CR are depicted in Table 6.8 where l is the number of precision bits added.

The introduced replacement methods are conceptualised to use only a certain amount of additional storage space. They were designed to use only l additional precision bits for k data points of the indices (see Section 6.2.3).

This design decision allows to limit exactly how much additional storage space is being used by each method. This precaution is reflected in Table 6.8. In the worst case one percent more storage space is needed. This occurred when using `zfp06+03` and replacing ten percent of the data. The compression ratio increased from 0.200 to 0.210.

6.5 Summary

In this chapter the efficiency of compression algorithms for environmental indices are investigated. A lossy compression method for climate indices is developed based on an established statistical method known as the Auto Regressive Integrated Moving Average (ARIMA) model. The indices examined are derived from the El Niño Southern Oscillation (ENSO), the North Atlantic Oscillation (NAO) and the Quasi-Biennial Oscillation (QBO) indices. Each index describes a different aspect of large-scale atmospheric dynamics.

An adaptive compression algorithm is introduced to improve the lossily compressed indices. The experimental results show that it is possible to improve the accuracy of the reconstructed data by replacing several data points with slightly higher precision. The improved reconstruction can reproduce the chosen indices to such a high degree that statistically relevant information needed for describing climate dynamics is preserved. The compressed indices have the same diagnostic performance than the original indices.

This study shows that ARIMA models using a differentiation step have difficulties and perform worse than models without differentiation steps. The experimental findings indicate that time-series data which can be expressed with small auto-regressive and moving-average order can be im-

proved significantly. Further analysis should focus on the aspect why certain time-series data like the QBO30 do not show the same improvement in reconstruction as the NAO index.

6.6 Code and Data Availability

The data of the environmental indices and an implementation of the replacement methods described above are available under GNU GPLv3 license at `https://github.com/ucyo/adaptive-lossy-compression` (Cayoglu, 2017).

CHAPTER 7

Data Coding and Residual Calculation

So far, the methods presented have concentrated on either the decorrelation or the approximation of data. Information spaces help to adapt the compression algorithm to the data to be compressed. The ARIMA model defines a lossy compression method for time series data that adds prior knowledge to the encoder. This chapter contributes to the final step of a compression algorithm by introducing a novel coding algorithm.

Coding is the final step of a compression algorithm. The actual compression of the data is happening in this step. In prediction-based compression this step starts with the calculation of the residual between prediction and true value. Currently there are two established forms of residual calculation: Exclusive-or and numerical difference. This chapter summarises both techniques and describes their strengths and weaknesses. Further, it is shown that shifting values improves upon some of the weaknesses. Shifting the prediction and true value to a binary number with certain properties results in a better compression factor with minimal computational costs. This gain in compression factor enables the usage of a less sophisticated prediction algorithm to achieve higher throughput during compression and decompression. In addition, a new coding scheme is introduced which helps to achieve a 10% increase on average in compression factor compared to the current state-of-the-art methods. The results presented in this chapter have been published in parts in Cayoglu et al. (2019b).

In the following section, the two methods for the residual calculation are described. Section 7.2 introduces the proposed method by first describing the value shift, then the preparation of the residual followed by the actual

coding of the residual. Section 7.3 explains the experimental setup and is followed by the evaluation of the experiments. Finally, an overview of the results and possible steps for further work is given.

7.1 Motivation

The biggest challenge with compression of floating-point data is the potentially infinite candidate space for making a prediction. While compression of textual data uses a 26 letter alphabet, the number of words of a certain length occurring in an English text is rather limited. This is not the case for numerical data like floating-points.

There are an infinite number of possible real numbers between arbitrary two numbers. While the precision of a single or double precision floating-point value is limiting these numbers, the number of possible values is still large compared to the English alphabet. A single precision floating-point value x using the IEEE754 standard (754-2008, 2008) has a 32 bits precision. One bit is being used for the sign, eight for the exponent and 23 for the mantissa. The exponent represents the largest power of two, that is still smaller than x. The mantissa represents the difference between x and the exponent. With 23 bits for the mantissa, the IEEE754 standard allows $2^{23} = 8388608$ values between each power of two. For values between 2 and 4 this results in a resolution of $2.4e-7$. This is the candidate space for the prediction of x if it is known beforehand that the searched value lies within this range. This is often not the case.

As Section 2.1.4 explains, a prediction-based compression algorithm makes a prediction for each data point. Afterwards, the difference between the true value and the prediction is being calculated. If the prediction is good, this residual will have a large LZC. These bits are disregarded and the remaining residuals are saved on disk. The original value can be reconstructed lossless with the same predictor and the remaining residual. The calculation method for the residual determines the size of the residual and the necessary steps to reverse the operation during decompression. Therefore, the residual calculation is of great importance.

Again, the two established methods for residual calculations are (see Eq. 2.14 and 2.15):

$$\text{diff}_{xor}(\hat{s}_i, s_i) = \hat{s}_i \oplus s_i \tag{7.1}$$

$$\text{diff}_{abs}(\hat{s}_i, s_i) = |\, \hat{s}_i - s_i \,| \tag{7.2}$$

with s_i being a data point in a source S and $\hat{s}_i$ a prediction for s_i based on a subset of S.

Both methods have their strengths and weaknesses. The first approach (Eq. 7.1) uses an eXclusive OR (XOR) operation for residual calculation. The advantage of this approach is that it is a very fast operation on modern hardware. Another advantage is that the reverse operation applied during decompression is literally the same operation with $s_i = \hat{s}_i \oplus \text{diff}_{xor}(\hat{s}_i, s_i)$. No information is needed to be transferred between encoder and decoder. The disadvantage of XOR is that two numbers representing very close values can still produce a very large residual. This is due to the two's-complement binary representation of floating-point numbers defined by the IEEE754 floating-point standard 754-2008 (2008) first introduced in 1985. This pitfall can be observed if $\hat{s}_i$ and s_i are close, but on opposite sides of a power of two e.g. $\hat{s}_i = 256.321$ and $s_i = 255.931$. While the absolute difference is 0.39, the residual calculated using Eq. 7.1 is $\text{diff}_{xor} = 16762689$ and using Eq. 7.2 is $\text{diff}_{abs} = 15041$. These residuals need to be saved on disk and the former residual needs 24 bits while the latter only 14 bits.

The second approach (Eq. 7.2) uses the absolute difference of the two numbers. In other words, it represents the amount of binary numbers between the two values using two's-complement binary representation. Since the accuracy of two's-complement binary representation of a value is limited, the value range of the compared values plays a central role in residual calculation using this approach. Given $\hat{s}_i = 847,390.837$ and $s_i = 847,794.417$ the difference is 403.58, but $\text{diff}_{xor} = 6458$. At first glance the prediction of $\hat{s}_i = 847,390.837$ for $s_i = 847,794.417$ seems to be worse than $\hat{s}_i = 256.321$ for $s_i = 255.931$. But using Eq. 7.1 it is still considered a better prediction due to the two's-complement binary representation of floating-point values.

The advantage of diff_{abs} is that the resulting residual is always smaller or at worst the same size as the residual calculated by diff_{xor}. The disadvantage is that additional information needs to be stored about the prediction $\hat{s}_i$. Without the information of $\hat{s}_i$ being above or below s_i a successful decompression can not happen. Another disadvantage is that to avoid an overflow or underflow one cannot calculate diff_{abs} in a single computation contrary

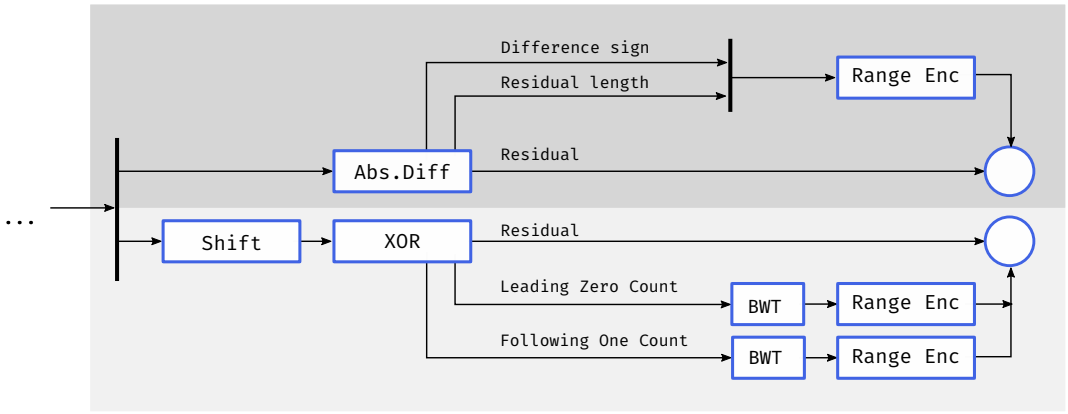


FIGURE 7.1 Flowchart of the residual calculation and coding phase of the current state of the art lossless compression algorithm for floating-point data (top) and the proposed method (bottom). This figure is taken from Cayoglu et al. (2019b).

to diff_{xor}. The calculation must be split into two steps by first calculating $\max(\hat{s}_i, s_i)$ and then subtracting the smaller one from the bigger. This is especially important if the values are close to the smallest or biggest representable number for single and double precision floating-point numbers, because of a possible overflow respectively underflow. In the following section a novel algorithm for calculating and coding the residual using XOR residual calculation is introduced.

7.2 Proposed Method

Chapter 3 introduced several state-of-the-art lossless compression algorithms for floating-point values. The current state-of-the-art lossless compression algorithm for floating-point data is `fpzip` which was introduced in Lindstrom and Isenburg (2006). In the following, the coding scheme of `fpzip` is compared to the coding scheme introduced in this chapter. Both methods are depicted in Figure 7.1. The coding scheme of `fpzip` is on top and highlighted with a dark grey background. The proposed coding scheme is on the bottom and highlighted with a light grey background and further referred as `pzip`.

The first step of the proposed algorithm is to shift $\hat{s}_i$ and s_i to a value range more suitable for difference calculation. Afterwards the diff_{xor} difference calculation method is applied to the data. The residual is then split into three streams: leading zero count (LZC), following one count (FOC),

and residual. The first two streams are transformed using Burrow-Wheeler-Transform (BWT) and written on disk using range encoding (RE). The third and final stream is saved verbatim on disk. In the following section each of these steps is described in more detail.

7.2.1 Shifted XOR

At the beginning of this chapter it is mentioned that the disadvantage of using XOR for residual calculation is that small differences between $\hat{s}_i$ and s_i might result in large residuals if both values are on opposite sides of a power of two. A closer look at the XOR operation helps to understand this challenge. The XOR is defined as follows:

$$\begin{aligned} \hat{s}_i \oplus s_i &= (\hat{s}_i \vee s_i) \wedge \neg(\hat{s}_i \wedge s_i) \\ &= (\hat{s}_i \wedge \neg s_i) \vee (\neg\hat{s}_i \wedge s_i) \end{aligned} \tag{7.3}$$

The bit of $\hat{s}_i \oplus s_i$ at index i is set, if the bit of $\hat{s}_i$ is different than the bit of s_i at index i. The bit is unset if they are the same. In the example given in Figure 7.2 the XOR calculation has a small LZC of eight and a rather large following one count (FOC) of ten. The FOC is defined as the number of set bits following the most significant unset bits in a binary representation. Although the example is deliberately chosen so that FOC is large, these in-

```
bin(p=256.321) = 01000011100000000010100100010111
bin(t=255.931) = 01000011011111111110111001010110
         p ⊕ t = 00000000111111111100011101000001
               index 1  3  5  7  9  11 13 15 17 19 21 23 25 27 29 31
```

FIGURE 7.2 An example for the XOR residual calculation method. This figure is taken from Cayoglu et al. (2019b).

stances occur very frequently when XOR is used for difference calculation. An advantage, however, is that extreme cases with very large FOC are well predictable. Due to the high number of zeros at positions 10-18 it can be estimated that a possible bit flip is imminent and therefore can act accordingly. The proposed shift to overcome this weakness of XOR can be calculated in two steps: The first step is to calculate a shift value s to be added to the prediction $\hat{s}_i$ so that $s = g(\hat{s}_i) - \hat{s}_i$ is satisfied, where $g(\hat{s}_i)$ is defined as follows

for single precision floating-point values:

$$g(\hat{s}_i) = \begin{cases} \sum_{k=1}^{15} 2^{2k-1} & \text{if } \hat{s}_i < 2^{30} \\ \sum_{k=0}^{14} 2^{2k} & \text{if } 2^{30} \leq \hat{s}_i < 2^{31} \\ \sum_{k=1}^{16} 2^{2k-1} & \text{if } 2^{31} \leq \hat{s}_i < 2^{32} \\ \sum_{k=0}^{15} 2^{2k} & \text{if } 2^{32} \leq \hat{s}_i \end{cases} \tag{7.4}$$

The binary representation of each of the goals is a fluctuation of set and unset bits. An example for the goal is the following:

$$g_1 = \sum_{k=1}^{15} 2^{2k-1} \text{ and } g_2 = \sum_{k=0}^{15} 2^{2k}$$
$$\text{bin}(g_1) = 00101010101010101010101010101010$$
$$\text{bin}(g_2) = 01010101010101010101010101010101$$

Finally, the shifted prediction $\hat{s}_i + s$ and shifted true value $s_i + s$ are calculated. Afterwards, the residual calculation proceeds as usual with applying the XOR operation to the shifted values calculating the residual:

$$\text{diff}^s_{xor}(\hat{s}_i, s_i, s) = (\hat{s}_i + s) \oplus (s_i + s) \tag{7.5}$$

The shift value can be recalculated without any information transfer between the encoder and decoder, since the decoder can recalculate the shift s with the information it has.

7.2.2 Splitting of the Residual

In the next step, each residual is split into three components: LZC, FOC and the remainder of the residual. This split is performed for each data point. The respective component of each residual is then grouped and coded together. As mentioned before, the LZC specifies how many of the Most Significant Bits (MSB) are unset. Due to the LZC definition, the block of unset bits is always followed by a block of set bits of size ≥ 1. The FOC determines the size of this block. The sum of LZC and FOC for a residual

is restricted to 32 (64) for single (double) precision floating-point values. The left-over bits are captured as the third component in the remainder. Since the bit following FOC will be unset, this bit will not be included in the remainder.

Given the previous example with $\hat{s}_i = 256.321$ and $s_i = 255.931$ the following components are obtained for the residual of $\hat{s}_i \oplus s_i$:

$$\begin{aligned} \hat{s}_i \oplus s_i &= \texttt{00000000011111111110001110100000 1} \\ \text{LZC}(\hat{s}_i \oplus s_i) &= 8 \\ \text{FOC}(\hat{s}_i \oplus s_i) &= 10 \\ \text{RES}(\hat{s}_i \oplus s_i) &= \texttt{0011101000001} \end{aligned}$$

where RES represents the remaining residual in binary representation.

7.2.3 Coding of LZC/FOC

In the next step, the LZC and FOC are coded. First, the LZCs and FOCs are reordered using the Burrow-Wheeler-Transform (BWT) introduced in Burrows and Wheeler (1994). The BWT algorithm rearranges a given set of values in such a way, that same values are more likely to appear one after another compared to the original data. Finally, the newly transformed LZC and FOC are coded using Range Encoding (Martin, 1979). Since LZC and FOC are independent from each other the BWT and RE can be performed concurrently.

7.3 Experimental Setup

7.3.1 Data

Two different datasets are used for the experiments conducted in this chapter. The first dataset is a synthetic dataset generated using a Gaussian distribution with different mean and standard deviations. These data cover a wide range of possible datasets that can be compressed with both compression algorithms. The synthetic data is used for the analysis of the XOR residual calculation when the data is close to a power of two.

The second dataset is obtained from a climate simulation using the ICON model (Schröter et al., 2018). The simulation is performed for an analysis of the POLar STRAtosphere in a Changing Climate (POLSTRACC) (POLSTRACC, 2019; Oelhaf et al., 2019) campaign, which performed flight measurements between December 2015 and March 2016 near the northern polar region. This dataset consists of two different vertical resolutions. The datasets consist of a 901×351 structured horizontal grid (longitude$\times$latitude) with 47 (respectively 90) vertical levels and four time steps with six hour resolution.

The following single precision floating-point variables are available: geopotential, vertical velocity, potential vorticity, cloud water, cloud ice content, specific humidity, temperature, virtual temperature, zonal wind, vorticity and meridional wind.

7.3.2 Metrics

Two metrics are used for evaluating the coding algorithms: compression factor (CF) and throughput. CF puts the file size before and after compression into relation (see Eq. 2.16). The higher the CF, the better the coding algorithm. The second quality measure for the coding algorithms is the throughput, which indicates the amount of data processed per unit of time (see Eq.2.18). The higher the throughput, the faster the algorithm.

7.3.3 Experiments

Several experiments are carried out to test currently available coding schemes as well as the proposed method.

State-Of-The-Art Compression Algorithm. The first experiment is conducted to identify the best currently available lossless compression algorithm. For this, several general-purpose and custom floating-point compression algorithms are run. This experiment is run using the climate data described in the previous section.

Shifted XOR. In the next experiment the behaviour of XOR residual calculation is analysed. These results help identify weaknesses of the method. The main focus during this experiment are the difficult cases where the val-

ues are close to powers of two. This experiment is run using the synthetic data described in the previous section. Further, possibilities to improve the throughput of the compression algorithm are analysed.

Splitting of the Residual. The third experiment is conducted to analyse the residuals. The distribution of set and unset bits in the residual are analysed. This distribution gives clues if the residual still contains information or whether it is white noise. Like in the previous experiment the synthetic dataset is being used for this experiment.

Performance of Coding Methods. The fourth experiment helps to assess available transformation and coding methods. For data transformation the BWT (Burrows and Wheeler, 1994) and Move-to-front (Ryabko, 1980) algorithms are analysed. Analysed data coding methods are range encoding (RE), Huffman Coding and run-length encoding (RLE). These transformations and coding schemes are applied on the original data as well as its delta. This experiment is run using the climate data.

Comparison of `pzip` and `fpzip`. Finally, the proposed method is compared with the state-of-the-art lossless compression algorithm `fpzip` regarding compression factor, throughput as well as the theoretical complexity of the compression algorithm.

All experiments are conducted on an Intel i5-7200U with 2.5 GHz running GNU/Linux 4.19.28 Debian with 16 GiB RAM. A native C implementation of `fpzip` is used. The proposed algorithm is implemented in Rust 1.33.0-nightly.

7.4 Evaluation

7.4.1 State-Of-The-Art Compression Algorithms

Various state-of-the-art compression algorithms are run using the climate simulation output to determine the currently best compression algorithm for floating-point data. The applied compression applications are: `blosc` (Alted, 2010), `fpzip` (Lindstrom and Isenburg, 2006), `xz`, `bzip2`, `zip`, `brotli` (Alakuijala and Szabadka, 2016), `spdp` (Claggett et al., 2018), and `fpc` (Burtscher and Ratanaworabhan, 2008). All algorithms were set to maximise the compression factor. The results are illustrated in Figure 7.3.

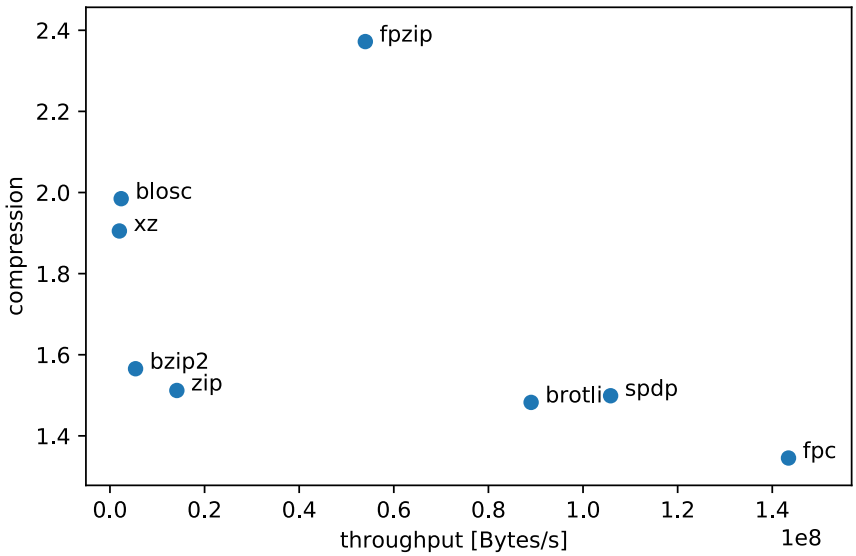


FIGURE 7.3 Average CF and throughput of the climate simulation dataset with currently available lossless compression algorithms. This figure is adapted from Cayoglu et al. (2019b).

The graph indicates that `fpzip` performs best with a compression factor ~ 2.4. The runner-up is `blosc` with a compression factor of ~ 2.0. The throughput of `fpzip` is not as good as those of `brotli`, `spdp` or `fpc`, but taking into account the mediocre performance of these algorithms regarding compression factor, it can be argued that `fpzip` is currently the best performing algorithm regarding lossless compression of floating-point data.

OBSERVATION 7.1 While there might be usage scenarios where `fpzip` is not the most successful compression algorithm, the results indicate that `fpzip` is on average the best performing algorithm for lossless compression of floating-data regarding compression factor. Every future algorithm should measure itself against these results.

7.4.2 Shifted XOR

In the following, the average LZC of a residual is analysed. The residuals are calculated using XOR with different Gaussian distributions using the synthetic dataset.

The results are illustrated in Figure 7.4. LZC falls most severely when the value is a power of two (marked by dotted lines). The intensity of these LZC dips are greater the smaller the standard deviation is in the data. The closer

TABLE 7.1 Effects of using a shifted XOR with a more sophisticated prediction algorithm (Lorenz) and inferior one (Last Value). Depicted are the Throughput and average LZC with and without (round brackets) shift operation.

Climate variable	Prediction	Throughput [MiB/s]	Avg. LZC
Temperature	Lorenz	19.86 (22.32)	24.01 (23.75)
	Last Value	31.33 (36.15)	22.89 (22.65)
Zonal wind	Lorenz	18.23 (18.68)	19.88 (19.61)
	Last Value	30.27 (37.49)	18.12 (17.83)
Geopotential	Lorenz	18.86 (19.92)	29.05 (28.85)
	Last Value	32.10 (37.91)	28.92 (28.83)

the two compared values are to each other at this range, the more important it is to move out of this range and perform the residual calculation in a different value range. The shift operation introduced in the previous section is depicted by the solid vertical lines. It achieves a higher LZC and thus reduces the final compression factor of the data.

One more advantage of shifting values to a more welcoming value range is the possibility to increase throughput. Shifting the values enables throughput increase by allowing the usage of a simpler prediction model for compression. The shift operation compensates the weaknesses of a simpler and faster predictor. Table 7.1 shows that using a simple prediction model (Last Value) a 50% higher throughput can be achieved, while the LZC is still close to that of a more sophisticated predictor. For details on the Lorenz and Last Value prediction methods, see Lindstrom and Isenburg (2006) and Cayoglu et al. (2018c).

OBSERVATION 7.2 The compression performance is dependent on the value ranges covered by the data as well as its distribution. A shifted residual calculation can improve the LZC and help to reduce the final compression factor of the data.

OBSERVATION 7.3 The shifted XOR calculation improves the average LZC and therefore the compression factor. It allows the use of simpler prediction models for higher throughput with comparable average LZCs.

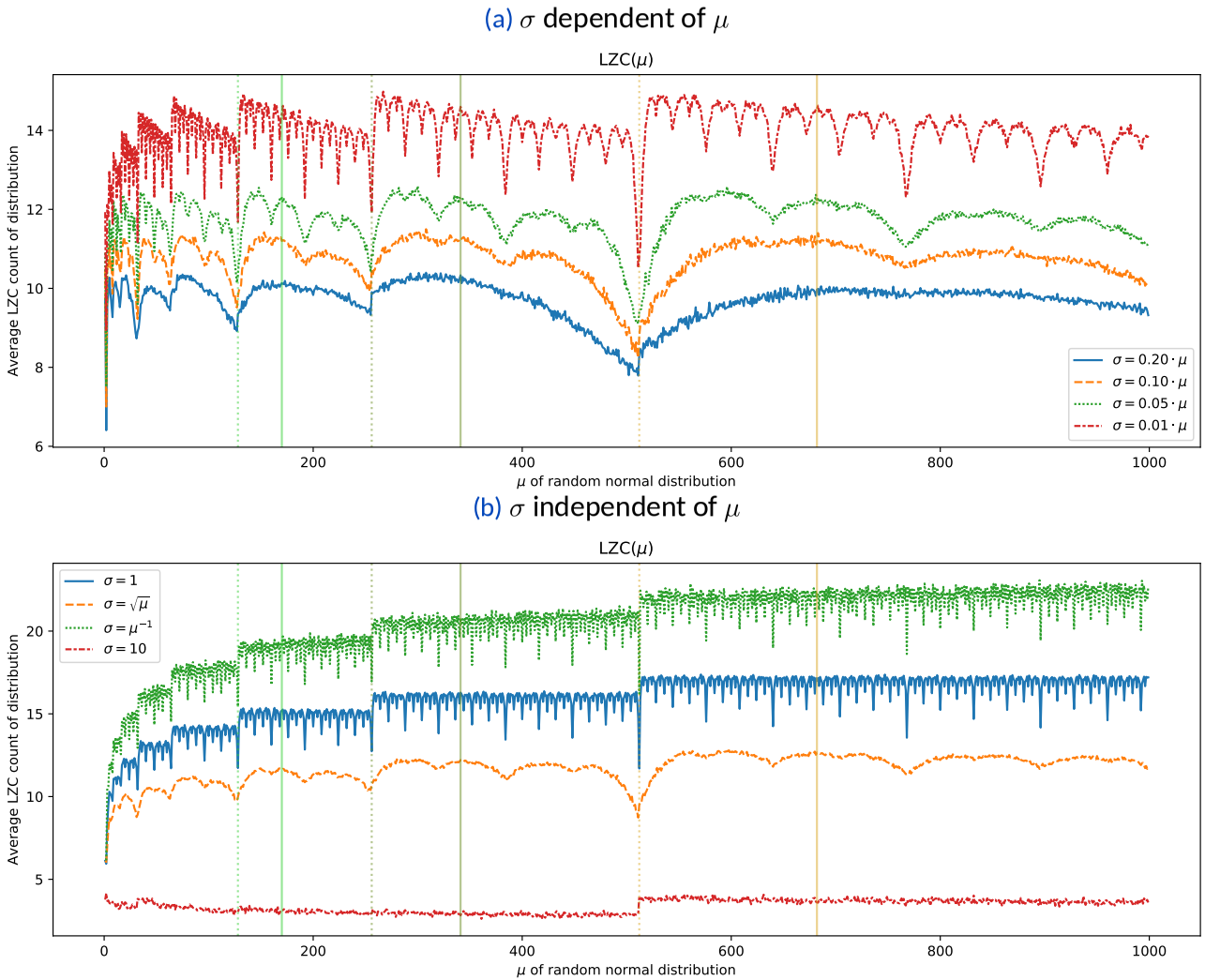


FIGURE 7.4 LZC for datasets with Gaussian distribution using a (a) σ dependent of μ and (b) σ independent of μ. The coloured vertical lines represent the resulting LZC for powers of two with (solid line) and without (dotted line) a shifted XOR calculation as described in Section 7.2.1.

7.4.3 Splitting of the Residual

In this section the residuals generated by XOR are analysed for statistical significance in the distribution of set and unset bits.

The results depicted in Figure 7.5 indicate that the distribution of set and unset values is not uniform. The horizontal blue lines indicate the uniform distribution. This suggests that information is still contained in the residual, as otherwise an even distribution of set and not set bits (e.g. white noise) would be expected. The number of set bits at the most significant positions of the residual occur at a much higher rate than the number of unset bits. Even with a length of six bits, there is a clear difference to the expected even distribution. This unequal distribution is caused by bit flips caused by the application of the XOR residual. This observation lead to the splitting of the residual into multiple streams described in Section 7.2.2.

OBSERVATION 7.4 There is a skewed distribution of the set and unset bits if XOR residual calculation is being used. This unequal distribution shows that there is information in the residual and that the compression factor can be further increased. A method to extract this information is to encode the number of FOCs separately.

7.4.4 Performance of Coding Methods

The transformation and coding methods analysed for this experiment are: Burrow-Wheeler-Transform (bwt), Delta difference (diff), Huffman coding (huff), Range coding (range), Move-To-Front (mtf), and Run-length Coding (rle). These coding methods are applied to the LZC and FOC. This experiment is conducted using the climate simulation output. Exemplary results for temperature are shown in Table 7.2, since the performance is similar for the other variables. The BWT transformation coupled with Range Coding performs best regarding LZC. Regarding FOC it performs only third best behind BWT+Huffman Coding and Huffman Coding.

Huffman Coding (without any transformation steps) performs best regarding FOC. This is due to the small value range of FOC. The downside of Huffman Coding is that the codelist used for coding the data must be transferred to the decoder. Such a necessity does not exist for Range Coding.

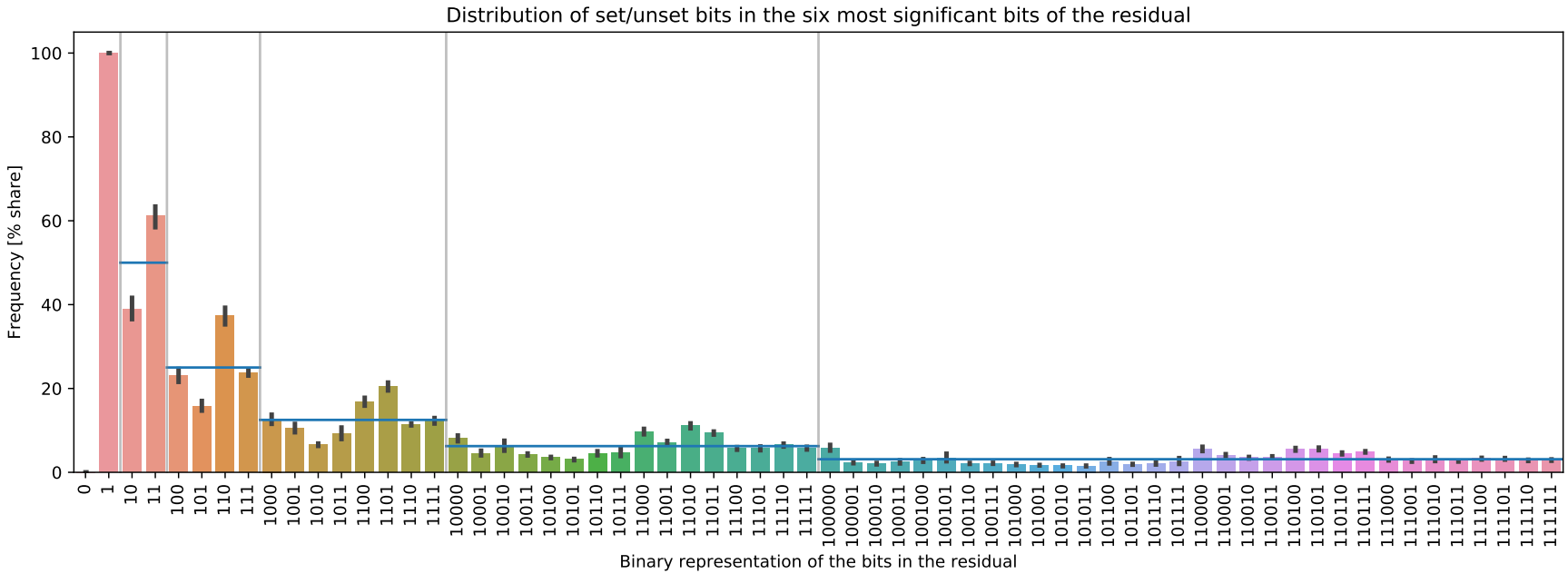


FIGURE 7.5 Distribution of set/unset bits in the six most significant bits of the residual. Horizontal lines show expected distribution if values would have been equally distributed (i.e. white noise). The vertical lines separate the individual bit lengths. The distribution was calculated using the ICON model simulation output. This figure is taken from Cayoglu et al. (2019b).

TABLE 7.2 File size of temperature data after transformation and coding schemes. The values are in Bytes. The abbreviations are: Burrow-Wheeler-Transform (bwt), Delta difference (diff), Huffman coding (huff), Range coding (range), Move-To-Front (mtf), Run-length Coding (rle). The best performing coding method for each category is highlighted.

Transformation and Coding Methods	**FOC**	**LZC**
bwt_diff_huff	29 715 223	41 347 759
bwt_diff_range	30 187 482	35 727 381
bwt_huff	21 625 804	50 204 194
bwt_mtf_diff_huff	32 046 132	45 767 747
bwt_mtf_diff_range	32 531 946	40 367 196
bwt_mtf_huff	25 382 523	37 223 861
bwt_mtf_range	25 958 407	32 657 611
bwt_mtf_rle_diff_huff	44 054 552	50 393 489
bwt_mtf_rle_diff_range	44 836 681	50 698 553
bwt_mtf_rle_huff	32 794 335	38 277 196
bwt_mtf_rle_range	33 716 265	38 402 906
bwt_range	22 043 891	**28 862 090**
diff_huff	29 451 797	37 695 831
diff_range	29 720 133	36 274 895
huff	**21 625 389**	50 203 689
mtf_diff_huff	31 908 557	46 008 528
mtf_diff_range	32 408 277	44 602 439
mtf_huff	25 374 374	35 935 034
mtf_range	25 782 694	34 990 704
mtf_rle_diff_huff	43 577 596	51 026 248
mtf_rle_diff_range	44 320 799	51 588 757
mtf_rle_huff	32 380 683	38 119 328
mtf_rle_range	33 313 109	38 667 389
range	22 489 931	42 918 321

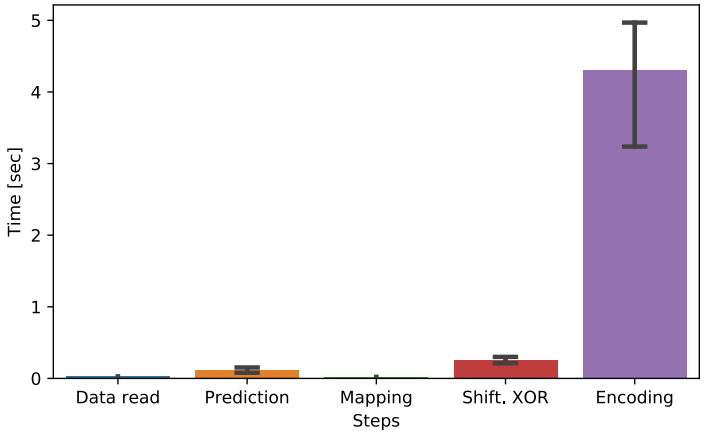


FIGURE 7.6 Time spend in each step of the compression algorithm. This figure is taken from Cayoglu et al. (2019b).

OBSERVATION 7.5 The BWT algorithm coupled with Range Coding outperforms most coding methods. While it is not optimal for FOC coding, it still performs good. With regard to the added value of using a single coding process for FOC and LZC, BWT+Range Coding is chosen.

7.4.5 Comparison of `pzip` and `fpzip`

In the last experiment `pzip` and `fpzip` are compared. The direct comparison of `pzip` and `fpzip` regarding CF and throughput is shown in Table 7.3. This experiment is conducted using the climate simulation output.

As can be seen from the table, in almost all cases the proposed algorithm outperforms `fpzip` in relation to CF. The only exception is cloud water, where `pzip` achieves a CF of 67.59 and `fpzip` 73.28. This is due to the high number of fill values in the cloud water data. Due to the coding scheme used by `fpzip`, it can compress exact predictions better than `pzip`. Future research must determine whether an alternative scheme should be used in such a case. In every other case `pzip` outperforms `fpzip` by ~ 10% on average. The performance of `pzip` for cloud ice content should be emphasised. The proposed algorithm achieves an improvement of 36.9% compared to `fpzip`.

Although the compression factor is better, the throughput of `fpzip` cannot be achieved with the current implementation of `pzip`. On average the `pzip` implementation is about six times slower than `fpzip`.

TABLE 7.3 CF and throughput [MiB/s] of climate simulation data using the proposed algorithm `pzip` and `fpzip` (higher is better). The best performing algorithm for each variables is coloured.

	Compression factor			Throughput [MiB/s]		
Climate variable	**fpzip**	**pzip**	**change [%]**	**fpzip**	**pzip**	**change [factor]**
Geopotential	9.96	**11.36**	+14.0	**125.68 ± 2.28**	35.13 ± 0.20	3.58
Vertical velocity	2.04	**2.22**	+8.8	**60.31 ± 0.63**	9.25 ± 0.15	6.52
Potential vorticity	2.24	**2.42**	+8.0	**67.12 ± 1.40**	9.94 ± 0.11	6.75
Cloud water	**73.28**	67.59	−7.8	**211.53 ± 7.97**	57.40 ± 0.52	3.69
Cloud ice content	2.87	**3.93**	+36.9	**84.84 ± 0.46**	16.88 ± 0.19	5.03
Specific humidity	2.60	**2.78**	+6.9	**67.62 ± 1.79**	10.88 ± 0.12	6.21
Temperature	3.23	**3.48**	+7.7	**75.14 ± 1.09**	12.35 ± 0.16	5.87
Virtual Temperature	3.23	**3.48**	+7.7	**71.42 ± 0.96**	12.16 ± 0.19	5.87
Zonal wind	2.23	**2.37**	+6.3	**61.73 ± 0.99**	9.50 ± 0.12	6.50
Vorticity	2.02	**2.17**	+7.4	**60.33 ± 1.13**	9.30 ± 0.10	6.48
Meridional wind	1.99	**2.18**	+9.5	**59.88 ± 1.75**	9.13 ± 0.14	6.56
∅ Average			+9.6			5.73

In order to analyse the bottleneck, each individual step of `pzip` was timed during execution. The result is depicted in Fig. 7.6. The majority of the time is spend in the coding step, specifically executing the BWT transformation. This is due to the memory footprint of BWT. While the time complexity of both algorithms are the same with $\mathcal{O}(n)$, the memory consumption is different. The currently known best implementation of BWT has a memory complexity of $\mathcal{O}(n \log \sigma)$ (Okanohara and Sadakane, 2009) with σ representing the number of elements in the alphabet. The time complexity of BWT is $\mathcal{O}(n)$ (Okanohara and Sadakane, 2009). The current implementation of `pzip` cannot use the L1, L2 and L3 caches of the CPU as effectively as `fpzip`. The reason for this is that the current setup of `pzip` is executed in block mode (see Section 2.1.1). This leads to an increased number of cache misses which in turn reduces throughput.

OBSERVATION 7.6 The compression factor of `pzip` is in most cases ~ 10% better than `fpzip` the state-of-the-art lossless compression algorithms for real-world climate data. The BWT transformation is the most time consuming task. More research is needed to optimise the coding step of the algorithm.

7.5 Summary

In this chapter different coding methods for data compression are analysed. It is shown that shifting the prediction and true value before calculating the residual results in a better compression factor with minimal additional computational costs. This shift enables the use of less sophisticated predictors with higher throughput.

The experimental results implicate that the compression performance is dependent on the value range covered by the data and its distribution. The shifted XOR calculation eliminates the disadvantages of XOR residual calculation by moving the data to a more favourable value range. Using XOR for residual calculation, results into a skewed distribution of set and unset bits. This non-uniform distribution suggests, that there is still information contained in the residual. By splitting the residual into LZC, FOC and the remaining residual, a decorrelation of this information is achieved. The

proposed coding scheme outperforms current state-of-the-art compression methods by ~ 10% with respect to the compression factor for the climate data used.

The time complexity of `fpzip` and `pzip` are the same, while the memory footprint of `pzip` is higher. Further research is needed for special case data such as cloud water, where the data consists mostly of fill values.

7.6 Code and Data Availability

The code of the proposed compression algorithm described above is available under GNU GPLv3 license at `https://github.com/ucyo/xor-and-residual-calculation` (Cayoglu, 2019b).

CHAPTER 8

Compression Framework

This chapter introduces a compression framework to enable scientists to design and develop custom compression algorithms. The results presented in this chapter have been published in parts in Cayoglu et al. (2018b).

8.1 Motivation

The proposed modular framework supports the creation of individual predictors, which can be customised and adjusted to the data at hand. The framework provides interfaces and customisable components, which are the building blocks to implement custom modules that are optimised for particular applications. Furthermore, the framework provides additional features such as the execution of benchmarks and validity tests for sequential and parallel execution of compression algorithms.

8.2 Proposed Method

The main goal of the framework is to provide state-of-the-art compression algorithms for domain scientists. It provides off-the-shelf solutions and a low barrier for customisation. In this section the structure of the proposed framework is described. The framework consists of two core components: objects and modifiers.

DEFINITION 8.1 (Object) Here, objects represent the current state of the data during the compression process (see Fig. 8.1). They may include metadata about previous states, but once they have been created they are immutable.

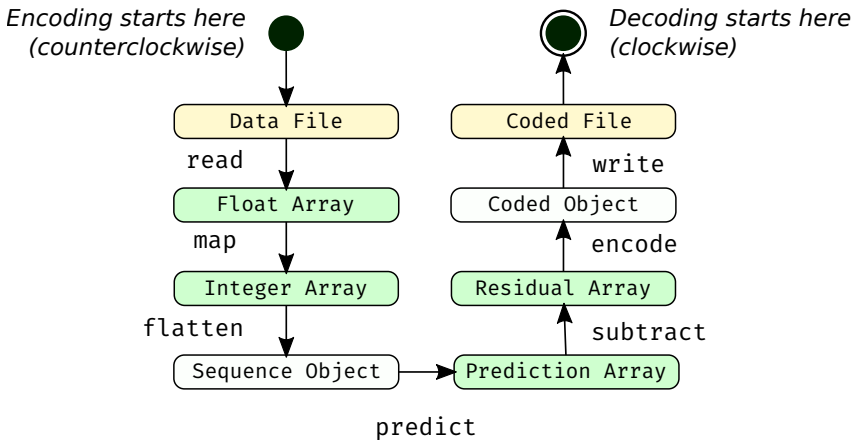


FIGURE 8.1 State diagram of a prediction-based compression algorithm. The label on the arrows define the transitions. The colour emphasises the similarity of the states. Data files are yellow, multi-dimensional arrays are green and custom elements white. This figure is taken from Cayoglu et al. (2018b).

DEFINITION 8.2 (Modifier) Modifiers operate on objects and are the only way to transform one object to another. Each modifier has exactly one method and can only operate on one kind of object. This setup prevents mistakes by allowing only a single way of interaction. A prediction-based compression algorithm consists of five modifiers with the following tasks:

- **Mapper**
  Mapping floating-point values to integers
- **Sequencer**
  Transforms an array into a data stream
- **Predictor**
  Predicts next datum on the data stream, based on past values
- **Subtractor**
  Calculates the residual between prediction and true value
- **Coder**
  Prepares residuals to be written on disk

The modifiers are designed for the tasks detailed in Section 2.1.4. The objects are the outcome of these tasks. The default compression function of the framework is shown in Algorithm 8.1.

Since the interface of each modifier is standardised it is easy to replace each modifier of the compression algorithm by a custom implementation. Apart from these components the framework provides additional modules

```
1:  procedure COMPRESS(arr, mapper, sequencer, predictor, subtractor, ...)
2:      iarr ← MAPPER.MAP(arr)                                  ▷ Mapper
3:      seq ← SEQUENCER.FLATTEN(iarr)                           ▷ Sequencer
4:      pred ← ARRAY.NEW()
5:      for i = 0 to seq.size do
6:          p ← PREDICTOR.PREDICT()
7:          PRED.APPEND(p)                                      ▷ Predictor
8:          PREDICTOR.UPDATE(seq[i])
9:      end for
10:     rarr ← SUBTRACTOR.SUBTRACT(iarr, seq, pred)             ▷ Subtractor
11:     coded ← CODER.CODE(rarr)                                ▷ Coder
12:     return coded
13: end procedure
```

ALGORITHM 8.1 Compression function of proposed framework.

to help the domain scientist during the design phase of a compression algorithm. These are for ensemble predictors, quality assessment, parallel compression and random subsetting. These additional modules are described in the following.

Ensemble Predictors. It is rather unlikely that there is one predictor that dominates all other predictors. For example, if it is known that a certain predictor performs well for temperature, but bad for greenhouse gas ozone, the framework should provide the possibility to switch between predictors or to average the result of the predictors (see Chapter 5).

For these cases the framework supports ensemble predictors. An ensemble predictor is defined by a list of predictors, a cost function and if necessary a consolidation method. The cost function determines the rank of the predictors. A consolidation function defines how the various predictions of the ensemble should be consolidated.

An example for an ensemble predictor is given in Algorithm 8.2. Here the predictors are ranked based on their performance prior to the current data point i.e. **last best method** (see Section 5.2.1) given a predefined traversal sequence (step four in Section 2.1.4). Please note the similarities in the syntax of Alg. 8.1 and Alg. 8.2. Since there is no distinguishing property of ensemble and non-ensemble predictors, the framework supports the nesting of ensemble predictors. An ensemble predictor may consist of several other ensemble predictors.

```
1: function PREDICT( )
2:     pred ← defaultpredictor
3:     if lastbestpredictor ≠ Null then
4:         pred ← lastbestpredictor
5:     else
6:         pred ← predictors[lastbestpredictor]
7:     end if
8:     return PRED.PREDICT()
9: end function
10:
11: function COST(prediction, truth)
12:     return ABS(truth − prediction)
13: end function
14:
15: function UPDATE(truth)
16:     predictions ← HASHMAP.NEW()
17:     for all p in predictors do
18:         prediction ← P.PREDICT()
19:         predictions[p] ← COST(prediction, truth)
20:     end for
21:     sorted ← SORTBYVALUE(predictions, ascending = True)
22:     lastbestpredictor ← sorted[0]
23: end function
```

ALGORITHM 8.2 An ensemble compression algorithm using the best predictor from previous prediction.

Quality Assessment. The Quality Assessment (QA) module provides information about the achievable compression ratio of the current setup and dataset. QA hooks into the compression process at step five (see Section 2.1.4) and calculates the LZC of the residual array. The residual array provides enough information about the performance of the predictors and expected CR. The average LZC can then be compared to the Shannon Entropy (Shannon, 1948) of the dataset. The Shannon Entropy quantifies the average amount of information represented by a random datum of the dataset.

Parallel Compression. The framework provides an additional module which can support the domain scientist in search for a compression method: parallel processing. The proposed framework contains a parallelization module, which can either chunk the data in blocks and compress each on a different thread or run a different predictor on each thread with the same input file. This leads to a less time-consuming development of a compression algorithm.

Random Subsetting. Since the design of a compression algorithm is an iterative process, it would be a daunting task for the scientist to have to compress gigabytes of data on each test, only to realise that a certain parameter needs to be fixed or a predictor eliminated. Therefore, the framework supports random subsetting of datasets. The subsetting is defined by size, error margin and possible dimension constraints.

These features help the scientist define a custom compression method for their data. The framework defines the necessary components and helper modules for customisation and grading of compression algorithms, while at the same time providing easy to use predefined algorithms. In the next section the actual implementation of the framework is described.

8.3 Implementation

An implementation of the framework is available at Cayoglu (2018b). The provided framework is implemented in Python 3 and uses as backend modules `scipy` (Oliphant, 2007), `pandas` (McKinney, 2010) and `xarray` (Hoyer and Hamman, 2017). It has been tested with files in NetCDF format with Climate and Forecast Metadata Conventions. The use of established open source

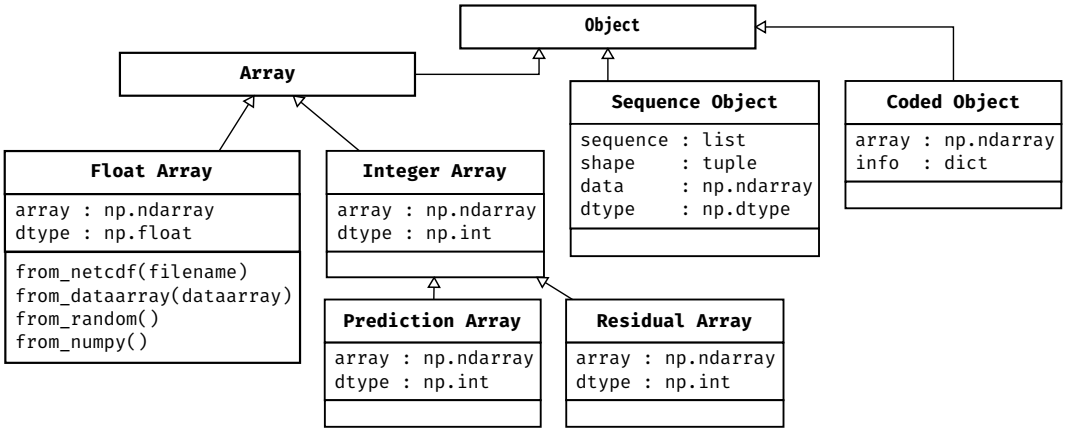


FIGURE 8.2 UML class diagram for object components. This figure is taken from Cayoglu et al. (2018b).

software provides a good basis for uptake, future cooperations and possible extensions of the framework. The details of the implementation of the core components are presented here.

The class diagram used for the implementation of the object components is shown in Fig. 8.2. As described in Section 8.2 the objects do not have the possibility to mutate itself or others. Except for `Float Array`, none of the objects has methods to manipulate its contents. The additional methods implemented in `Float Array` are for initialisation from common data types such as `numpy` (Oliphant, 2006) arrays or netcdf (Rew and Davis, 1990) data files.

The `Prediction Array` and `Residual Array` inherit from `Integer Array`. While these classes do not provide additional functionality compared to the `Integer Array`, they are necessary to provide strong distinction of objects on which each modifier can operate.

Figure 8.3 depicts possible modifiers to be used as components of the framework. This is a none exhaustive list of modifiers and should exemplify the large number of possible options in designing a compression algorithm. The modifiers which are implemented at the time of publication are emphasised. A description for each modifier is included in the documentation of the implementation.

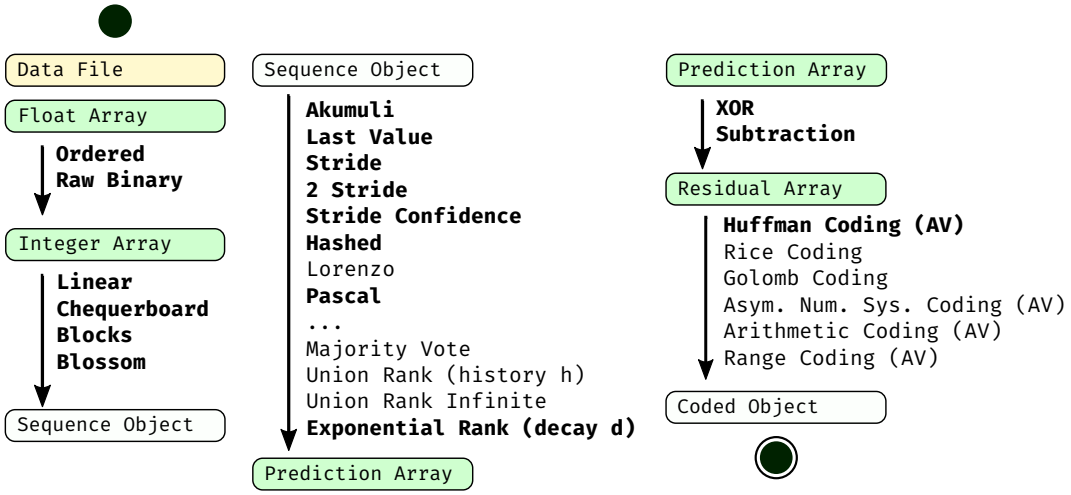


FIGURE 8.3 A none exhaustive list of modifiers to exemplify the large number of possible combinations. Emphasised are the modifiers which are implemented and part of the framework at the time of publication. This figure is taken from Cayoglu et al. (2018b).

8.4 Summary

In recent years, climate sciences have experienced a breakthrough in terms of possible fine-granular simulations. Next-generation climate models make it possible to carry out high-resolution simulations on HPC systems. This led to a significant increase of storage space. In this chapter a modular framework for the compression of climate data is presented.

The framework provides all necessary components to design, test and grade various prediction-based compression algorithms. It also supports the use of ensemble predictors to merge predictions based on different predictors, quality assessment methods to help assess the performance of the prediction methods, parallel compression for concurrent execution of predictors as well as random subsetting for unbiased result acquisition during the development of a compression algorithm. Although the framework is mainly used by climate researchers, it is conceivable to use it in conjunction with other structured floating-point data.

8.5 Code and Data Availability

An implementation of the framework described above is available under GNU GPLv3 license at `https://github.com/ucyo/cframework` (Cayoglu, 2018b).

CHAPTER 9

Conclusions and Outlook

This chapter gives an overview of the main contributions of this thesis and possible future research directions.

9.1 Conclusions

The goal of this thesis is to provide insights and contributions to the state-of-the-art in compression algorithms for structured floating-point data. The main challenge in the compression of floating-point data is the candidate space for making a prediction compared to e.g. textual data. All compression algorithms use the same principle: Identify and remove redundant information in the decorrelation step (based on a context), optionally further align the data in the approximation step, and find a more compact representation in the coding step. This thesis provides contributions to each of these three steps.

The analyses of variance, entropy and mutual information show that there is no ideal context for value prediction. The ideal context depends on the time, resolution and location of the data point. The described analyses for the identification of redundant information and the introduction of information spaces and contexts contribute to the decorrelation of the data. The use of information spaces helps to identify new patterns and relationships within variables for the current data. Using IS achieves an improvement of 10% on average for LZC and a reduction of the standard deviation for the compression factor by an average of more than 20% for the climate data used.

The lossy ARIMA compression algorithm provides a novel approximation algorithm for stationary time-series data. This method allows the integration of prior knowledge about the interactions of the variables into

the encoder. These two contributions close the circle and allow the compression algorithm to work with prior knowledge and information from the data available.

A thorough literature review indicates that a significant part of the prediction-based compression algorithms for floating-point data uses XOR for the residual calculation. This kind of residual calculation is improved by the novel coding algorithm described in this thesis. The analysis and subsequent comparison with state-of-the-art compression algorithms shows that the coding algorithm presented in this thesis achieves the best compression factors for structured floating-point data. The proposed coding scheme outperforms current state-of-the-art compression methods by ~ 10% with respect to the compression factor for the climate data used.

These contributions are of course meaningless if they cannot be used by the community. In the spirit of Open Science, all contributions have been made public as well as open source and can be reproduced by all interested parties. Please read the last section of each chapter for access to the code and/or data. In addition, all core contributions are published in a framework that simplifies the development of a custom compression algorithm.

This research was initiated by the needs of climate science, but the application of its contributions is not limited to it. The results of this thesis can be used to develop or improve any compression algorithm for structured floating-point data.

9.2 Outlook

The following is a collection of ideas that could be investigated to advance the compression of floating-point data.

Real Number Representation. The currently most used representation for real numbers is the IEEE754 floating-point standard 754-2008 (2008) first introduced in 1985. As already described in the background section, this form of representation has weaknesses when comparing two numbers or calculating the difference. There are other ways for describing real numbers: fixed-point representation (e.g. Q number format), floating-bars, or posits. An in-depth analysis of these representations and their respective differences could help solve the problem with powers of two that the IEEE754 floating-point has.

Coding of LZC/FOC. The coding algorithm described in this thesis still has redundant information in its residual. Preliminary experiments show that there is high mutual information between LZC and FOC. Future research could analyse if the sum of these values behaves differently compared to each single stream. Then one value (e.g. FOC) could be inferred from the sum and the remainder (e.g. LZC). This could further improve the compression factor.

Routing Problem of Ideal Traversal Paths. Two decisions are crucial for the performance of a prediction-based compression algorithm: traversal path and prediction method. The traversal path defines what information is available for calculating the prediction. The prediction method defines how this information is used to make a final prediction. Both of these problems are strongly coupled. The first part of the problem could be formulated as a routing problem: Each data point is a node with 8ø edges (number of adjacent cells in a tesseract) and the value difference of both nodes as weights for the edges. The shortest path through all nodes is now the smoothest curve and therefore the most predictable. The difficulty then no longer lies in the prediction of the data points, but in the compact representation of the route since this has to be passed on to the decoder.

Compression Using Adaptive Quantisation. Quantisation is a process of mapping values from one domain space to the other e.g. by rounding or truncation. Usually the source domain is continuous and the destination discrete. Instead of a predefined interval in which the simulation model writes its output, an adaptive quantisation method could be developed, which decides if an output should be generated based on the data. This decision could be based on a grade, which classifies how well the current data can be interpolated or predicted. This could be developed as a lossless or lossy compression algorithm. It only depends on the grading model and the extent of the quantisation applied to the data.

Machine Learning for Lossy and Lossless Compression. Since a couple years, machine learning is in the centre of attention across different scientific fields. Neural networks achieve unprecedented results in object recognition and classification of images and several other fields. The most promis-

ing models for compression are 3D convolutions and recurrent models e.g. long short-term memory (LSTM) models. Currently these models are considered for the whole compression process on its own e.g. Bellard (2019) and not as an extension for established algorithms. There should be more research focusing on using LSTM for parts of a prediction-based compression algorithm. This can be either for the prediction step (e.g. memory of previous values) or for the definition of the traversal path (e.g. memory of good traversal dimensions). Both should lead to a better prediction and therefore a better compression factor.

Predictor Based on Lagrange, Hermite, and Birkhoff Interpolation. There are already several prediction-based compression algorithms using Lagrange interpolation for one dimensional predictors with very good compression results e.g. Robinson (1994). A natural extension of the Lagrange interpolation are the multivariate interpolation techniques of Hermite and Birkhoff. Hermite interpolation considers the actual value and its derivative for multivariate interpolation, while Birkhoff also considers missing values. An in-depth analysis of these interpolation techniques for multivariate prediction could increase the prediction quality. These results could be integrated into the information spaces described in this thesis.

This page is intentionally left blank

APPENDIX A

Variance Analysis

In this appendix further results of the variance analysis described in Section 4.3.2 are shown.

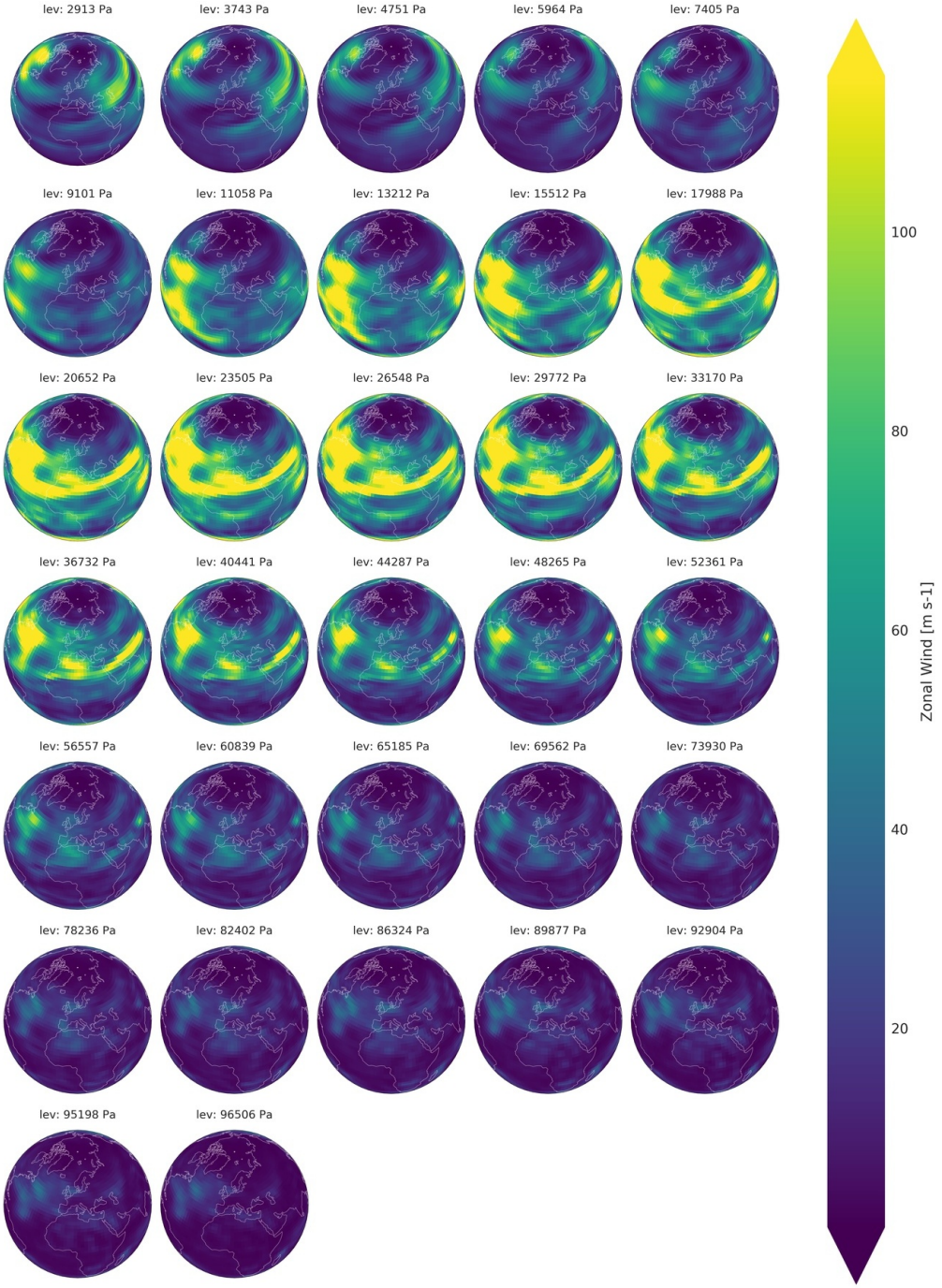


FIGURE A.1 Short-term variance analysis for specific humidity across time for the northern hemisphere.

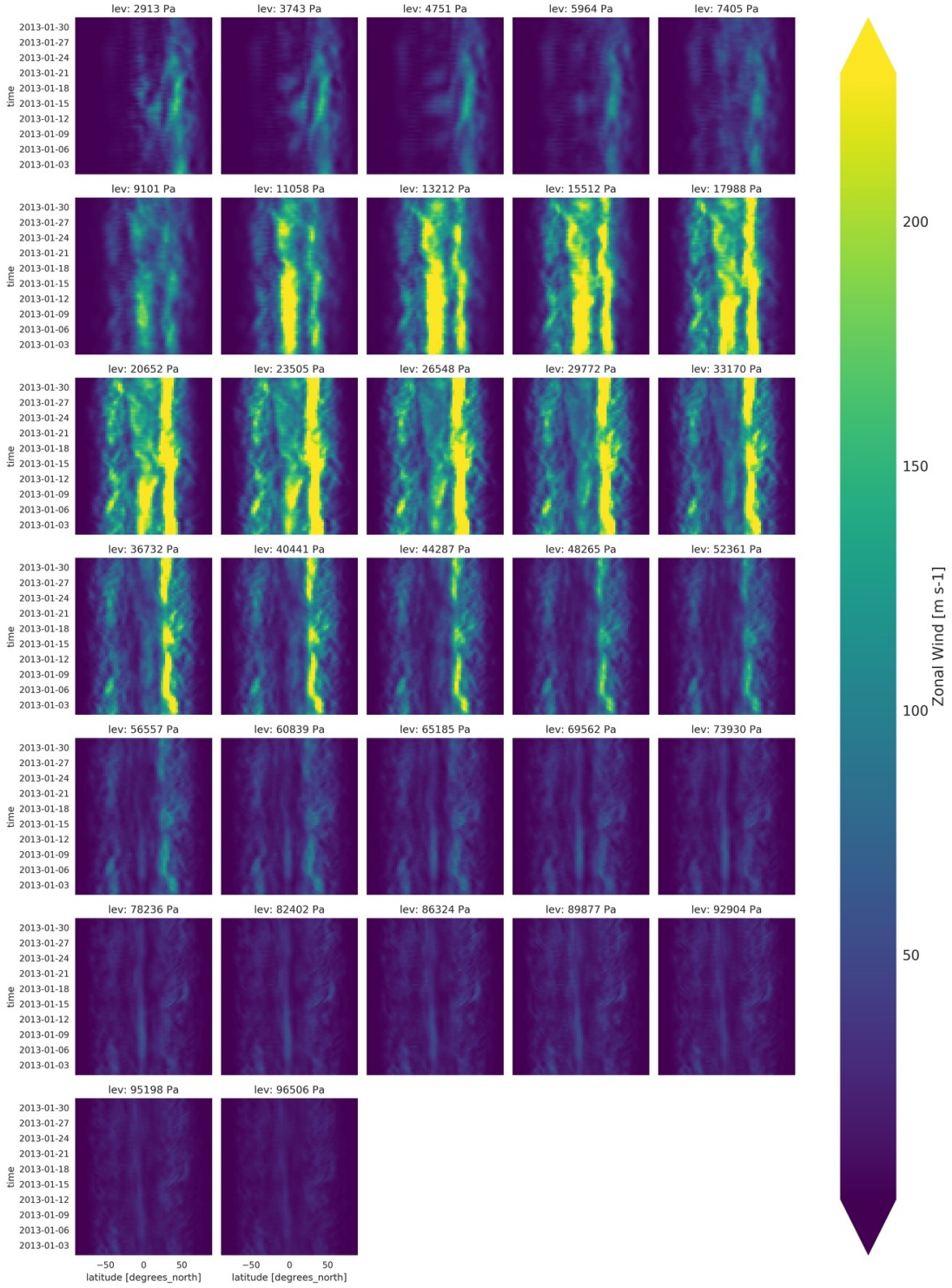


FIGURE A.2 Short-term variance analysis for specific humidity across longitude along the latitudes.

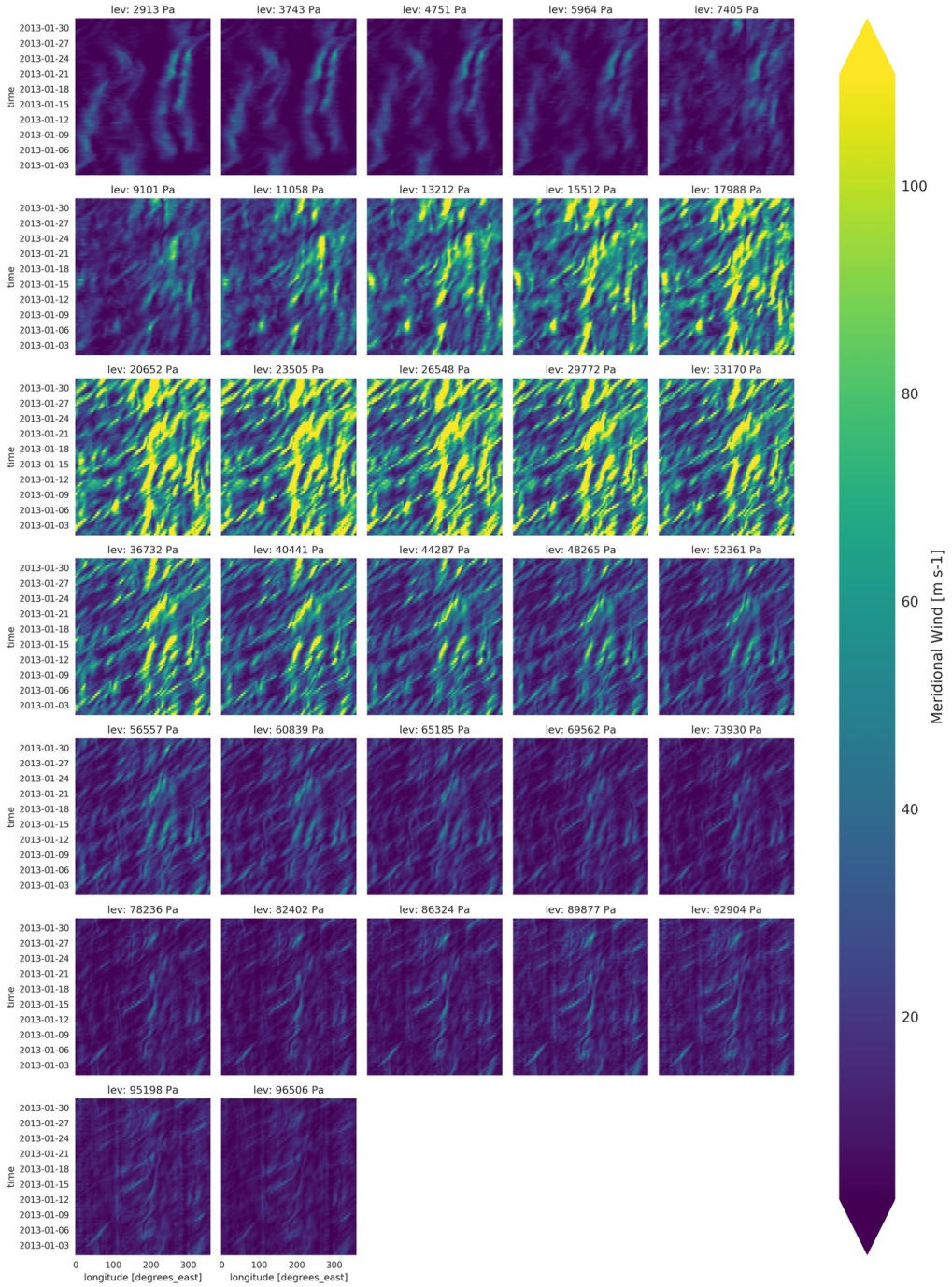


FIGURE A.3 Short-term variance analysis for specific humidity across latitudes.

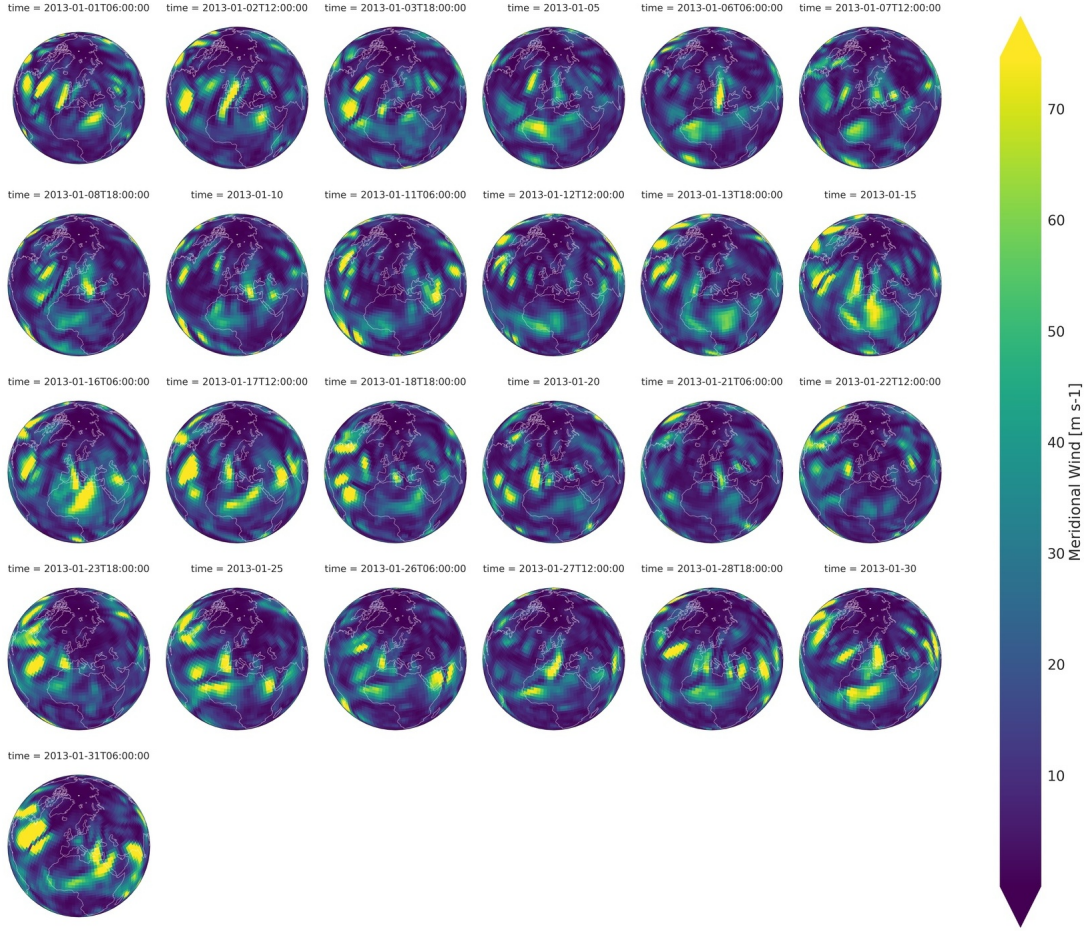


FIGURE A.4 Short-term variance analysis for meridional wind across altitudes for the northern hemisphere.

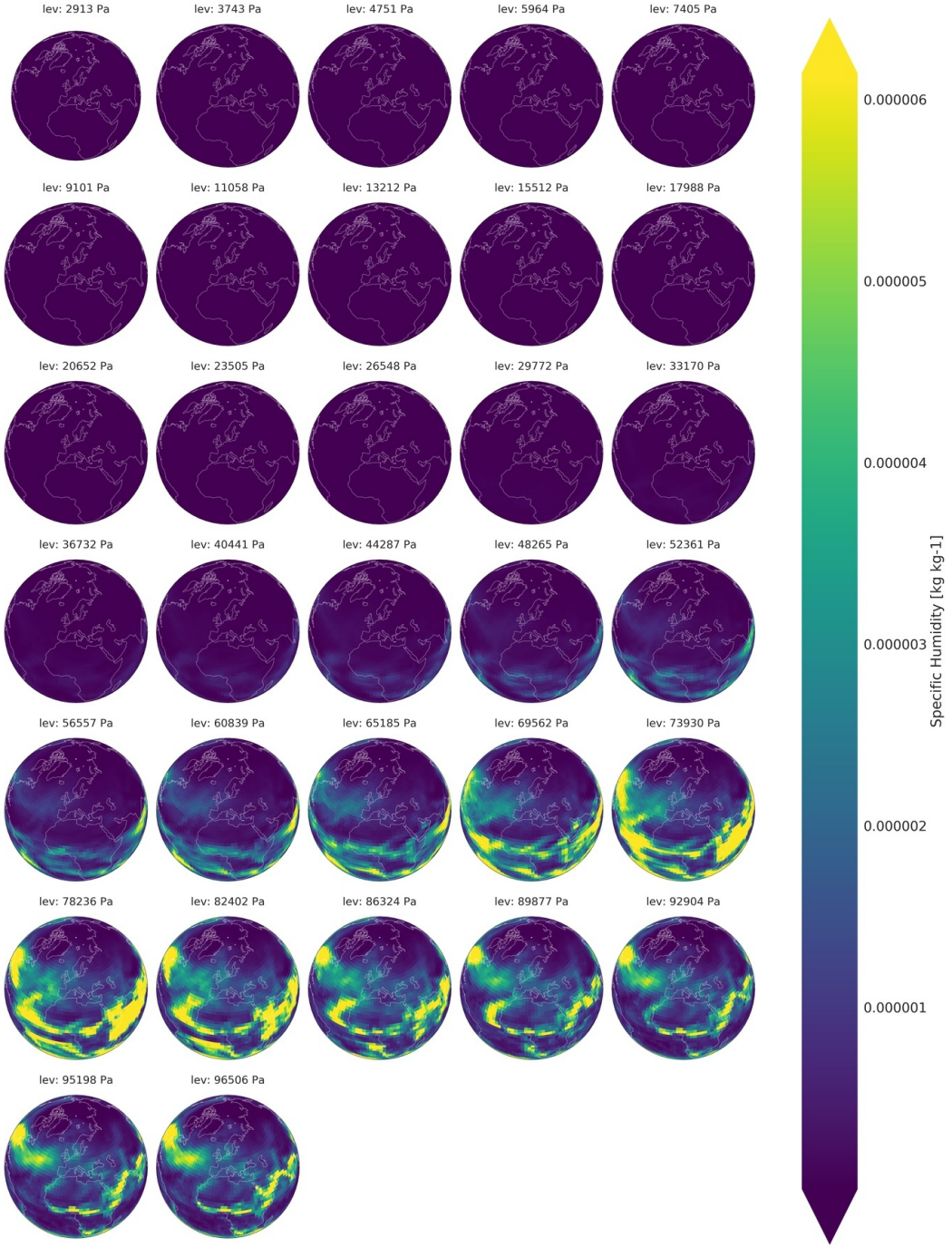


FIGURE A.5 Short-term variance analysis for zonal wind across time for the northern hemisphere.

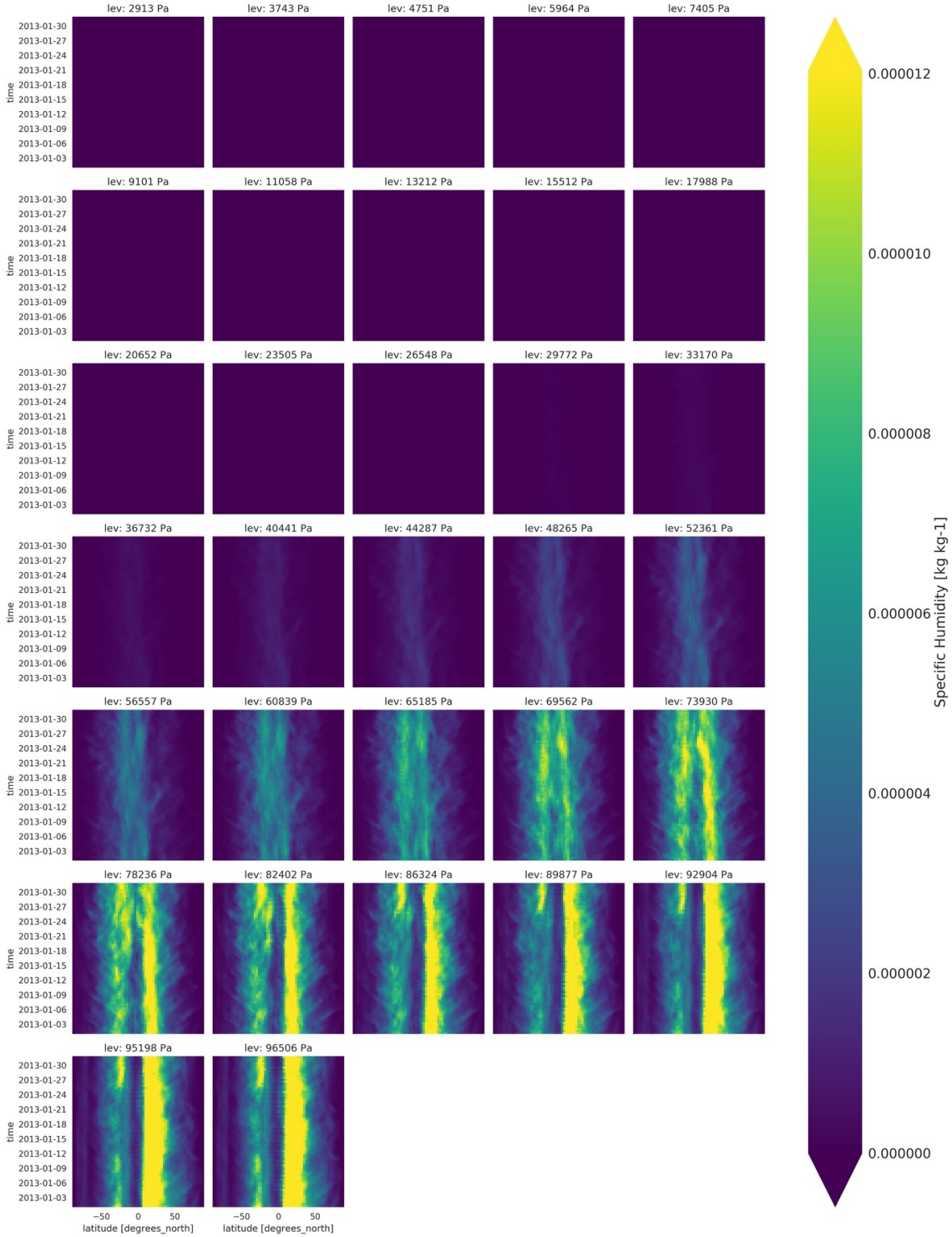


FIGURE A.6 Short-term variance analysis for zonal wind across longitude along the latitudes.

APPENDIX B

Pascal Predictor

In this appendix the pascal predictor is discussed in more detail. Given a Sequence $S = s_1 s_2 \dots s_{n-1} s_n$ with n elements, the `Pascal k` predictor makes a prediction for an element s_i using the last k elements in the sequence with $k \ll n$. The predictor is based on an prediction method used in audio compression (Robinson, 1994) and polynomial interpolation. The Pascal k is the optimal predictor for data without white noise and on a uniform grid which can be described by a polynomial function f of degree $i - 1$ (Eq. B.1). The coefficients of Pascal 1-5 are shown in Table B.1. The name `Pascal` has been chosen, because the coefficients can also be derived from Pascal's triangle. In the following the conducted experiments are described.

$$f(x) = \sum_{j=1}^{i} a_j \cdot x^j \tag{B.1}$$

The coefficients of Pascal **k** predictor can predict a polynomial function of order $k - 1$ exactly.

LEMMA B.1 **Given the n-th order backwards difference $\nabla_h^n[p](x)$ the optimal coefficients are $p(x) = \sum_{i=1}^{n} (-1)^{i+1} \binom{n}{i} p(x - i)$ for uniform spacing $h = 1$.**

TABLE B.1 Coefficients for Pascal *k* predictor using the last k values for prediction of s_i.

Predictor	Formula
`Pascal 1`	$\hat{s}_i = s_{i-1}$
`Pascal 2`	$\hat{s}_i = 2\, s_{i-1} - s_{i-2}$
`Pascal 3`	$\hat{s}_i = 3\, s_{i-1} - 3\, s_{i-2} + s_{i-3}$
`Pascal 4`	$\hat{s}_i = 4\, s_{i-1} - 6\, s_{i-2} + 4\, s_{i-3} - s_{i-4}$
`Pascal 5`	$\hat{s}_i = 5\, s_{i-1} - 10\, s_{i-2} + 10\, s_{i-3} - 5\, s_{i-4} + s_{i-5}$

Proof This can be shown using finite differences (which are zero in orders higher than those of the polynomial function):

$$\begin{aligned}
\nabla_h^n[p](x) &= \sum_{i=0}^{n}(-1)^i \binom{n}{i} p(x-ih) \text{ with } h=1 \text{ and } \nabla_h^n[p](x) := 0 \\
0 &= \sum_{i=0}^{n}(-1)^i \binom{n}{i} p(x-i) \\
0 &= -1^0 \binom{n}{0} p(x) + \sum_{i=1}^{n}(-1)^i \binom{n}{i} p(x-i) \\
p(x) &= \sum_{i=1}^{n}(-1)^{i+1} \binom{n}{i} p(x-i)
\end{aligned}$$

■

This page is intentionally left blank

Bibliography

754-2008, I. (2008). IEEE 754-2019 - IEEE Standard for Floating-Point Arithmetic. IEEE, https://standards.ieee.org/content/ieee-standards/en/standard/754-2019.html.

Ahmed, N., Natarajan, T., and Rao, K. R. (1974). Discrete cosine transform. **IEEE transactions on Computers**, 100(1):90–93.

Aiazzi, B., Alba, P., Alparone, L., and Baronti, S. (1999). Lossless compression of multi/hyper-spectral imagery based on a 3-D fuzzy prediction. **IEEE Trans. Geosci. Remote Sens.**, 37(5 pt 1):2287–2294, ISSN: 01962892, DOI: 10.1109/36.789625, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=789625.

Ainsworth, M., Klasky, S., and Whitney, B. (2017). Compression Using Lossless Decimation: Analysis and Application. **SIAM J. Sci. Comput.**, 39(4):B732–B757, ISBN: 0001409107, ISSN: 1064-8275, DOI: 10.1137/16m1086248, http://epubs.siam.org/doi/10.1137/16M1086248.

Akaike, H. (1974). A New Look at the Statistical Model Identification. **IEEE Trans. Automat. Contr.**, 19(6):716–723, ISBN: 0018-9286 VO - 19, ISSN: 15582523, DOI: 10.1109/TAC.1974.1100705.

Al Tamimi, A. K., Jain, R., and So-In, C. (2008). SAM: A simplified seasonal ARIMA model for mobile video over wireless broadband networks. In **Multimedia, 2008. ISM 2008. Tenth IEEE International Symposium on**, pages 178–183. IEEE.

Alakuijala, J., Obryk, R., Stoliarchuk, O., Szabadka, Z., Vandevenne, L., and Wassenberg, J. (2017). Guetzli: Perceptually Guided JPEG Encoder. pages 1–13, http://arxiv.org/abs/1703.04421.

Alakuijala, J. and Szabadka, Z. (2016). Brotli compressed data format. Technical report, http://www.rfc-editor.org/info/rfc7932.

Allasia, G., Cavoretto, R., and De Rossi, A. (2018). Hermite–Birkhoff interpolation on scattered data on the sphere and other manifolds. **Appl. Math. Comput.**, 318:35–50, ISBN: 9780735414389, ISSN: 00963003, DOI: 10.1016/j.amc.2017.05.018.

Alted, F. (2010). Why modern cpus are starving and what can be done about it. **Computing in Science & Engineering**, 12(2):68.

Baker, A. H., Hammerling, D. M., Mickelson, S. A., Xu, H., Stolpe, M. B., Naveau, P., Sanderson, B., Ebert-Uphoff, I., Samarasinghe, S., De Simone, F., Carbone, F., Gencarelli, C. N., Dennis, J. M., Kay, J. E., and Lindstrom, P. (2016). Evaluating lossy data compression on climate simulation data within a large ensemble. **Geosci. Model Dev.**, 9(12):4381–4403, ISSN: 19919603, DOI: 10.5194/gmd-9-4381-2016, http://www.geosci-model-dev-discuss.net/gmd-2016-146/.

Bartels, R. H., Beatty, J. C., and Barsky, B. A. (1987). **An introduction to splines for use in computer graphics & geometric modeling**. Morgan Kaufmann Publishers Inc.

Bellard, F. (2019). Lossless Data Compression with Neural Networks Long Short-Term Memory Model. pages 1–10, https://bellard.org/nncp/nncp.pdf.

Box, G. E. and Jenkins, G. M. (1976). Time series analysis, control, and forecasting. **San Francisco, CA: Holden Day**, 3226(3228):10.

Bruylants, T., Munteanu, A., and Schelkens, P. (2015). Wavelet based volumetric medical image compression. **Signal Process. Image Commun.**, 31:112–133, ISBN: 0923-5965, ISSN: 09235965, DOI: 10.1016/j.image.2014.12.007.

Burden, R. L. and Faires, J. D. (1997). Numerical analysis, brooks. **Cole Pub**, 7.

Burrows, M. and Wheeler, D. J. (1994). A block-sorting lossless data compression algorithm.

Burtscher, M. and Ratanaworabhan, P. (2008). Fpc: A high-speed compressor for double-precision floating-point data. **IEEE Transactions on Computers**, 58(1):18–31.

Cappello, F. and Lindstrom, P. (2018). Workshop on lossy compression for scientific data. https://europar2018.org/workshops-and-tutorials/tutorial-compression. [Online; accessed 22-08-2019].

Cayoglu, U. (2017). Repository for "Adaptive Lossy Compression of Complex Environmental Indices Using Seasonal Auto-Regressive Integrated Moving Average Models". https://github.com/ucyo/adaptive-lossy-compression. [Online; accessed 22-August-2019].

Cayoglu, U. (2018a). Repository for "Concept and Analysis of Information Spaces to Improve Prediction-Based Compression". https://github.com/ucyo/informationspaces. [Online; accessed 22-August-2018].

Cayoglu, U. (2018b). Repository for "Prediction-based Compression Framework". https://github.com/ucyo/cframework. [Online; accessed 27-May-2018].

Cayoglu, U. (2019a). Repository for "Data Analysis and Identification of Redundant Information". https://github.com/ucyo/climate-data-analysis. [Online; accessed 17-September-2019].

Cayoglu, U. (2019b). Repository for "On the Application of XOR for Residual Calculation in Prediction-based Compression". http://github.com/ucyo/xor-and-residual-calculation/. [Online; accessed 11-April-2019].

Cayoglu, U., Braesicke, P., Kerzenmacher, T., Meyer, J., and Streit, A. (2017). Adaptive Lossy Compression of Complex Environmental Indices Using Seasonal Auto-Regressive Integrated Moving Average Models. In **2017 IEEE 13th International Conference on e-Science (e-Science)**, pages 315–324. DOI: 10.1109/eScience.2017.45. [best paper award].

Cayoglu, U., Braesicke, P., Kerzenmacher, T., Meyer, J., and Streit, A. (2018a). Towards an optimised environmental data compression method for structured model output. In **EGU General Assembly Conference Abstracts**, volume 20, page 8609. https://meetingorganizer.copernicus.org/EGU2018/EGU2018-8609.pdf.

Cayoglu, U., Schröter, J., Meyer, J., Streit, A., and Braesicke, P. (2018b). A Modular Software Framework for Compression of Structured Climate Data. In **Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems**, SIGSPATIAL '18, pages 556–559, New York, NY, USA. ACM, ISBN: 978-1-4503-5889-7, DOI: 10.1145/3274895.3274897.

Cayoglu, U., Tristram, F., Meyer, J., Kerzenmacher, T., Braesicke, P., and Streit, A. (2018c). Concept and Analysis of Information Spaces to improve Prediction-Based Compression. In **2018 IEEE International Conference on Big Data (Big Data)**, pages 3392–3401. DOI: 10.1109/BigData.2018.8622313.

Cayoglu, U., Tristram, F., Meyer, J., Kerzenmacher, T., Braesicke, P., and Streit, A. (2019a). On Advancement of Information Spaces to Improve Prediction-Based Compression. In David, K., Geihs, K., Lange, M., and Stumme, G., editors, **INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik – Informatik für Gesellschaft**, pages 271–272, Bonn. Gesellschaft für Informatik e.V., ISBN: 978-3-88579-688-6, ISSN: 1617-5468, DOI: 10.18420/inf2019_39.

Cayoglu, U., Tristram, F., Meyer, J., Schröter, J., Kerzenmacher, T., Braesicke, P., and Streit, A. (2019b). Data Encoding in Lossless Prediction-Based Compression Algorithms. In **IEEE 15th International Conference on e-Science (e-Science)**. ISBN: 978-1-7281-2451-3, DOI: 10.1109/eScience.2019.00032.

Chattopadhyay, A. K. and Chattopadhyay, T. (2014). **Time series analysis**, volume 3 of **Springer Texts in Statistics**. Springer New York, New York, NY, ISBN: 978-0-387-75958-6, ISSN: 21991049, DOI: 10.1007/978-1-4939-1507-1_9, http://books.google.com/books?hl=en{&}lr={&}id=MrNY3s2difIC{&}oi=fnd{&}pg=PR2{&}dq=Time+series+analysis+with+applications+in+R{&}ots=UqDwRRMrk5{&}sig=Hz1koyO45IysG2YiM0rEO2Maza4http://link.springer.com/10.1007/978-0-387-75959-3.

Chebyshev, P. L. (1853). **Théorie des mécanismes connus sous le nom de parallélogrammes**. Imprimerie de l'Académie impériale des sciences.

Chen, P., Yuan, H., and Shu, X. (2008). Forecasting crime using the ARIMA model. In **Proc. - 5th Int. Conf. Fuzzy Syst. Knowl. Discov. FSKD 2008**, volume 5, pages 627–630. IEEE, ISBN: 9780769533056, DOI: 10.1109/FSKD.2008.222, http://ieeexplore.ieee.org/document/4666600/.

Claggett, S., Azimi, S., and Burtscher, M. (2018). Spdp: An automatically synthesized lossless compression algorithm for floating-point data. In **2018 Data Compression Conference**, pages 335–344. IEEE.

Collet, Y. (2011). Repository for "LZ4". https://lz4.github.io/lz4/. [Online; accessed 03-October-2019].

de Guenni, L. B., García, M., Muñoz, Á. G., Santos, J. L., Cedeño, A., Perugachi, C., and Castillo, J. (2017). Predicting monthly precipitation along coastal Ecuador: ENSO and transfer function models. **Theor. Appl. Climatol.**, 129(3-4):1059–1073, ISBN: 5930428506, ISSN: 14344483, DOI: 10.1007/s00704-016-1828-4, http://link.springer.com/10.1007/s00704-016-1828-4.

De la Cruz, L., Pallarès, E., Alins, J. J., and Mata, J. (1998). Self-similar traffic generation using a fractional ARIMA model. Application to the VBR MPEG video traffic. In **Telecommunications Symposium, 1998. ITS'98 Proceedings. SBT/IEEE International**, pages 102–107. IEEE.

Del Testa, D. and Rossi, M. (2015). Lightweight Lossy Compression of Biometric Patterns via Denoising Autoencoders. **IEEE Signal Process. Lett.**, 22(12):2304–2308, ISSN: 10709908, DOI: 10.1109/LSP.2015.2476667.

Deri, L., Mainardi, S., and Fusco, F. (2012). tsdb: A Compressed Database for Time Series. In Pescapè, A., Salgarelli, L., and Dimitropoulos, X., editors, **Traffic Monit. Anal.**, pages 143–156, Berlin, Heidelberg. Springer Berlin Heidelberg, ISBN: 978-3-642-28534-9.

Deutsch, L. P. (1996). Deflate compressed data format specification version 1.3.

Deutsch, P. and Gailly, J.-L. (1996). Zlib compressed data format specification version 3.3.

Di, S. and Cappello, F. (2016). Fast Error-Bounded Lossy HPC Data Compression with SZ. In **Proc. - 2016 IEEE 30th Int. Parallel Distrib. Process. Symp. IPDPS 2016**, pages 730–739. IEEE, ISBN: 9781509021406, DOI: 10.1109/IPDPS.2016.11, http://ieeexplore.ieee.org/document/7516069/.

Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. **Journal of the American statistical association**, 74(366a):427–431.

Duda, J. (2013). Asymmetric numeral systems: entropy coding combining speed of Huffman coding with compression rate of arithmetic coding. pages 1–24, http://arxiv.org/abs/1311.2540.

Dunton, A. M., Jofre, L., Iaccarino, G., and Doostan, A. (2019). Pass-efficient methods for compression of high-dimensional particle-laden turbulent flow data. pages 1–24, http://arxiv.org/abs/1905.13257.

Eichinger, F., Efros, P., Karnouskos, S., and Böhm, K. (2015). A time-series compression technique and its application to the smart grid. **VLDB J.**, 24(2):193–218, ISSN: 0949877X, DOI: 10.1007/s00778-014-0368-8.

Engelson, V., Fritzson, D., and Fritzson, P. (2002). Lossless compression of high-volume numerical data from simulations. In **Proc. DCC 2000. Data Compression Conf.**, volume 5, 11, page 574. IEEE Comput. Soc, ISBN: 0-7695-0592-9, ISSN: 1068-0314, DOI: 10.1109/dcc.2000.838221, http://ieeexplore.ieee.org/document/838221/.

Eugene, L. (2017). Akumuli Time-series Database. http://www.akumuli.org.

Fink, E. and Gandhi, H. S. (2011). Compression of time series by extracting major extrema. **J. Exp. Theor. Artif. Intell.**, 23(2):255–270, ISSN: 0952813X, DOI: 10.1080/0952813X.2010.505800.

Folk, M., McGrath, R. E., and Yeager, N. (1999). Hdf: an update and future directions. In **IEEE 1999 International Geoscience and Remote Sensing Symposium. IGARSS'99 (Cat. No. 99CH36293)**, volume 1, pages 273–275. IEEE.

Fout, N. and Ma, K. L. (2012). An adaptive prediction-based approach to lossless compression of floating-point volume data. **IEEE Trans. Vis. Comput. Graph.**, 18(12):2295–2304, ISBN: 1077-2626, ISSN: 10772626, DOI: 10.1109/TVCG.2012.194, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6327234.

French, K. R., Schwert, G. W., and Stambaugh, R. F. (1987). Expected stock returns and volatility. **Journal of financial Economics**, 19(1):3–29.

Garrett, M. W. and Willinger, W. (1994). Analysis, modeling and generation of self-similar VBR video traffic. In **ACM SIGCOMM computer communication review**, volume 24, 4, pages 269–280. ACM.

Goeman, B., Vandierendonck, H., and de Bosschere, K. (2002). Differential FCM: increasing value prediction accuracy by improving table usage efficiency. In **Proc. HPCA Seventh Int. Symp. High-Performance Comput. Archit.**, pages 207–216. IEEE Comput. Soc, ISBN: 0-7695-1019-1, ISSN: 1530-0897, DOI: 10.1109/hpca.2001.903264, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=903264http://ieeexplore.ieee.org/document/903264/.

Gok, A. M., Di, S., Alexeev, Y., Tao, D., Mironov, V., Liang, X., and Cappello, F. (2018). PaSTRI: Error-Bounded Lossy Compression for Two-Electron Integrals in Quantum Chemistry. In **2018 IEEE Int. Conf. Clust. Comput.**, volume September, pages 1–11. IEEE, ISBN: 978-1-5386-8319-4, ISSN: 15525244, DOI: 10.1109/CLUSTER.2018.00013, https://ieeexplore.ieee.org/document/8514854/.

Golomb, S. (1966). Run-length encodings (Corresp.). **IEEE Trans. Inf. theory**, 12(3):399–401.

Gomez, L. A. and Cappello, F. (2013). Improving floating point compression through binary masks. In **Proc. - 2013 IEEE Int. Conf. Big Data, Big Data 2013**, pages 326–331. ISBN: 9781479912926, DOI: 10.1109/BigData.2013.6691591.

Haar, A. (1910). Zur Theorie der orthogonalen Funktionensysteme - Erste Mitteilung. **Math. Ann.**, 69(3):331–371, ISSN: 00255831, DOI: 10.1007/BF01456326.

Hong, E. S. and Ladner, R. E. (2002). Group testing for image compression. **IEEE Transactions on Image Processing**, 11(8):901–911, ISSN: 1057-7149, DOI: 10.1109/TIP.2002.801124.

Howard, P. G. and Vitter, J. S. (1993). Fast and efficient lossless image compression. In **[Proceedings] DCC93: Data Compression Conference**, pages 351–360. IEEE.

Howard, P. G. and Vitter, J. S. (1994). Arithmetic Coding for Data Compression. **Proc. IEEE**, 82(6):857–865, ISBN: 0-8186-9202-2, ISSN: 15582256, DOI: 10.1109/5.286189, http://portal.acm.org/citation.cfm?doid=214762.214771.

Hoyer, S. and Hamman, J. (2017). xarray: N-D labeled arrays and datasets in Python. **Journal of Open Research Software**, 5(1), DOI: 10.5334/jors.148, http://doi.org/10.5334/jors.148.

Huamin Chen, Jian Li, and Mohapatra, P. (2005). RACE: time series compression with rate adaptivity and error bound for sensor networks. pages 124–133, ISBN: 0780388151, DOI: 10.1109/mahss.2004.1392089.

Huang, X., Ni, Y., Chen, D., Liu, S., Fu, H., and Yang, G. (2016). Czip: A Fast Lossless Compression Algorithm for Climate Data. **Int. J. Parallel Program.**, 44(6):1248–1267, ISSN: 08857458, DOI: 10.1007/s10766-016-0403-z, http://link.springer.com/10.1007/s10766-016-0403-z.

Hübbe, N., Wegener, A., Kunkel, J. M., Ling, Y., and Ludwig, T. (2013). Evaluating lossy compression on climate data. In **Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)**, volume 7905 LNCS, pages 343–356. ISBN: 9783642387494, ISSN: 03029743, DOI: 10.1007/978-3-642-38750-0_26, http://link.springer.com/10.1007/978-3-642-38750-0_26.

Huffman, D. A. (1952). A Method for the Construction of Minimum-Redundancy Codes. **Proc. IRE**, 40(9):1098–1101, ISBN: 1111022020101, ISSN: 00968390, DOI: 10.1109/JRPROC.1952.273898, http://ieeexplore.ieee.org/document/4051119/.

Hurrell, J. W. and Van Loon, H. (1997). Decadal variations in climate associated with the North Atlantic Oscillation. In **Climatic change at high elevation sites**, pages 69–94. Springer.

Hurwitz, M. M., Braesicke, P., and Pyle, J. A. (2011). Sensitivity of the mid-winter Arctic stratosphere to QBO width in a simplified chemistry–climate model. **Atmospheric Science Letters**, 12(3):268–272.

Ibarria, L., Lindstrom, P., Rossignac, J., and Szymczak, A. (2003). Out-of-core compression and decompression of large n-dimensional scalar fields. **Comput. Graph. Forum**, 22(3):343–348, ISBN: 0167-7055, ISSN: 01677055, DOI: 10.1111/1467-8659.00681, http://doi.wiley.com/10.1111/1467-8659.00681.

Isenburg, M., Lindstrom, P., and Snoeyink, J. (2005). Lossless compression of predicted floating-point geometry. **Comput. Des.**, 37(8):869–877, ISBN: 0010-4485, ISSN: 00104485, DOI: 10.1016/j.cad.2004.09.015, http://linkinghub.elsevier.com/retrieve/pii/S0010448504002544.

ITU-T.81 (1992). Digital compression and coding of continuous-tone still images – Requirements and guidelines. ITU, DOI: http://handle.itu.int/11.1002/1000/2633.

ITU-T.87 (1998). Lossless and near-lossless compression of continuous-tone still images . ITU, DOI: http://handle.itu.int/11.1002/1000/4395.

Jöckel, P., Tost, H., Pozzer, A., Brühl, C., Buchholz, J., Ganzeveld, L., Hoor, P., Kerkweg, A., Lawrence, M. G., Sander, R., Steil, B., Stiller, G., Tanarhte, M., Taraborrelli, D., van Aardenne, J., and Lelieveld, J. (2006). The atmospheric chemistry general circulation model ECHAM5/MESSy1: consistent simulation of ozone from the surface to the mesosphere. **Atmospheric Chemistry and Physics**, 6(12):5067–5104, DOI: 10.5194/acp-6-5067-2006, http://www.atmos-chem-phys.net/6/5067/2006/.

Josh Coalson, J. (2019). Free lossless audio compression (flac) library. https://xiph.org/flac/format.html.

Karadimitriou, K. and Tyler, J. M. (1998). The Centroid method for compressing sets of similar images. **Pattern Recognit. Lett.**, 19(7):585–593, ISSN: 01678655, DOI: 10.1016/S0167-8655(98)00033-6, http://www.sciencedirect.com/science/article/pii/S0167865598000336http://linkinghub.elsevier.com/retrieve/pii/S0167865598000336.

Kerzenmacher, T., Cayoglu, U., Kellmann, S., Kirner, O., Versick, S., Wang, S., and Braesicke, P. (2018). QBO influence on the ozone distribution in the extra-tropical stratosphere. In **EGU General Assembly Conference Abstracts**, volume 20, page 16565. https://meetingorganizer.copernicus.org/EGU2018/EGU2018-16565.pdf.

Krislock, N. and Wolkowicz, H. (2012). **Handbook on Semidefinite, Conic and Polynomial Optimization**, volume 166 of **International Series in Operations Research & Management Science**. Springer US, Boston, MA, ISBN: 978-1-4614-0768-3, ISSN: 08848289, DOI: 10.1007/978-1-4614-0769-0, http://link.springer.com/10.1007/978-1-4614-0769-0.

Kuschel, M., Kremer, P., Hirche, S., and Buss, M. (2006). Lossy data reduction methods for haptic telepresence systems. **Proc. - IEEE Int. Conf. Robot. Autom.**, 2006(May):2933–2938, ISBN: 0780395069, ISSN: 10504729, DOI: 10.1109/ROBOT.2006.1642147.

Lakshminarasimhan, S., Shah, N., Ethier, S., Klasky, S., Latham, R., Ross, R., and Samatova, N. F. (2011). Compressing the incompressible with ISABELA: In-situ reduction of spatio-temporal data. In **Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)**, volume 6852 LNCS, pages 366–379. ISBN: 9783642233999, ISSN: 03029743, DOI: 10.1007/978-3-642-23400-2_34, http://link.springer.com/10.1007/978-3-642-23400-2{_}34.

Langdon, G. and Rissanen, J. (1981). Compression of black-white images with arithmetic coding. **IEEE Transactions on Communications**, 29(6):858–867.

Legendre, A.-M. (1790). **Recherches sur l'attraction des sphéroïdes homogènes**.

Levinson, N. (1946). The wiener (root mean square) error criterion in filter design and prediction. **Journal of Mathematics and Physics**, 25(1-4):261–278, DOI: 10.1002/sapm1946251261, https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm1946251261.

Li, X. and Orchard, M. T. (2001). Edge-directed prediction for lossless compression of natural images. **IEEE Trans. Image Process.**, 10(6):813–817, ISBN: 0-7803-5467-2, ISSN: 10577149, DOI: 10.1109/83.923277.

Liang, X., Di, S., Tao, D., Chen, Z., and Cappello, F. (2018a). An Efficient Transformation Scheme for Lossy Data Compression with Point-wise Relative Error Bound. In **2018 IEEE Int. Conf. Clust. Comput.**, pages 179—-189.

Liang, X., Di, S., Tao, D., Li, S., Li, S., Guo, H., Chen, Z., and Cappello, F. (2018b). Error-Controlled Lossy Compression Optimized for High Compression Ratios of Scientific Datasets.

Lindstrom, P. (2014). Fixed-Rate Compressed Floating-Point Arrays. **IEEE Trans. Vis. Comput. Graph.**, 20(12):2674–2683, ISBN: 1077-2626, ISSN: 1077-2626, DOI: 10.1109/TVCG.2014.2346458, http://ieeexplore.ieee.org/document/6876024/.

Lindstrom, P., Chen, P., and Lee, E. J. (2016). Reducing disk storage of full-3D seismic waveform tomography (F3DT) through lossy online compression. **Comput. Geosci.**, 93(May):45–54, ISSN: 00983004, DOI: 10.1016/j.cageo.2016.04.009.

Lindstrom, P. and Isenburg, M. (2006). Fast and efficient compression of floating-point data. **IEEE Trans. Vis. Comput. Graph.**, 12(5):1245–1250, ISSN: 10772626, DOI: 10.1109/TVCG.2006.143, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4015488.

Liu, L., Ogino, M., and Hagita, K. (2017). Efficient Compression of Scientific Floating-Point Data and An Application in Structural Analysis. **Trans. Japan Soc. Comput. Eng. Sci.**, 2017(20170002):20170002–20170002, ISSN: 1347-8826, DOI: 10.11421/JSCES.2017.20170002, https://www.jstage.jst.go.jp/article/jsces/2017/0/2017{_}20170002/{_}article/-char/ja/.

Liu, S., Huang, X., Ni, Y., Fu, H., and Yang, G. (2014). A high performance compression method for climate data. In **Proc. - 2014 IEEE Int. Symp. Parallel Distrib. Process. with Appl. ISPA 2014**, pages 68–77. IEEE, ISBN: 9781479942930, DOI: 10.1109/ISPA.2014.18, http://ieeexplore.ieee.org/document/6924431/.

Lu, T., Liu, Q., He, X., Luo, H., Suchyta, E., Choi, J., Podhorszki, N., Klasky, S., Wolf, M., Liu, T., and Qiao, Z. (2018). Understanding and modeling lossy compression schemes on HPC scientific data. **Proc. - 2018 IEEE 32nd Int. Parallel Distrib. Process. Symp. IPDPS 2018**, (February):348–357, ISBN: 9781538643686, DOI: 10.1109/IPDPS.2018.00044.

Marin, O., Schanen, M., and Fischer, P. (2016). Large-scale lossy data compression based on an a priori error estimator in a Spectral Element Code. ISSN: 0903-1936, https://www.semanticscholar.org/paper/Large-scale-lossy-data-compression-based-on-an-a-in-Marin-Schanen/74301be975f17d8df319103ada85410645ad4fa2.

Martin, G. N. N. (1979). Range encoding: an algorithm for removing redundancy from a digitised message. In **Video Data Rec. Conf.**, number March, pages 24–27. ISSN: 05380006.

McKinney, W. (2010). Data structures for statistical computing in python. In van der Walt, S. and Millman, J., editors, **Proceedings of the 9th Python in Science Conference**, pages 51 – 56.

Moffat, A. (1991). Two-level context based compression of binary images. In **[1991] Proceedings. Data Compression Conference**, pages 382–391. IEEE.

Motta, G., Storer, J. A., and Carpentieri, B. (2000). Lossless image coding via adaptive linear prediction and classification. **Proceedings of the IEEE**, 88(11):1790–1796.

NASA (2019). Zonal bands. https://upload.wikimedia.org/wikipedia/commons/c/c6/Zonal_band.svg. [Online; accessed 25-August-2019].

Neves, A. J. and Pinho, A. J. (2009). Lossless compression of microarray images using image-dependent finite-context models. **IEEE Trans. Med. Imaging**, 28(2):194–201, ISBN: 0278-0062, ISSN: 02780062, DOI: 10.1109/TMI.2008.929095, http://ieeexplore.ieee.org/document/4591388/.

Nowack, P. J., Braesicke, P., Luke Abraham, N., and Pyle, J. A. (2017). On the role of ozone feedback in the ENSO amplitude response under global warming. **Geophysical Research Letters**.

Oelhaf, H., Sinnhuber, B.-M., Woiwode, W., Bönisch, H., Bozem, H., Engel, A., Fix, A., Friedl-Vallon, F., Grooß, J.-U., Hoor, P., Johansson, S., Jurkat-Witschas, Kaufmann, T., Stefan Krämer, M., Kraus, J., Kretschmer, E., Lörks, D., Marsing, A., Orphal, J., Pfeilsticker, K., Pitts, M., Poole, L., Preusse, P., Rapp, M., Riese, M., Rolf, C., Ungermann, J., Voigt, C., Volk, C. M., Wirth, M., Zahn, A., and Ziereis, H. (2019). POLSTRACC: Airborne experiment for studying the Polar Stratosphere in a Changing Climate with the high-altitude long-range research aircraft HALO. **Bulletin of the American Meteorological Society**. [accepted, to be published].

Okanohara, D. and Sadakane, K. (2009). A linear-time burrows-wheeler transform using induced sorting. In **SPIRE**.

Oliphant, T. E. (2006). **A guide to NumPy**, volume 1. Trelgol Publishing USA.

Oliphant, T. E. (2007). Python for scientific computing. **Computing in Science Engineering**, 9(3):10–20, ISSN: 1521-9615, DOI: 10.1109/MCSE.2007.58.

Olver, P. J. (2006). On multivariate interpolation. **Stud. Appl. Math.**, 116(2):201–240, ISSN: 00222526, DOI: 10.1111/j.1467-9590.2006.00335.x.

Pai, P.-F. and Lin, C.-S. (2005). A hybrid ARIMA and support vector machines model in stock price forecasting. **Omega**, 33(6):497–505.

Pavlov, I. (2019). 7zip library. https://www.7-zip.org/.

Peano, G. (1890). Sur une courbe, qui remplit toute une aire plane. **Mathematische Annalen**, 36(1):157–160.

Pearson, K. (1896). Mathematical Contributions to the Theory of Evolution. III. Regression, Heredity, and Panmixia. **Philos. Trans. R. Soc. London. Ser. A, Contain. Pap. a Math. or Phys. Character**, 187:253–318, ISSN: 02643952, http://www.jstor.org/stable/90707.

Pelkonen, T., Franklin, S., Teller, J., Cavallaro, P., Huang, Q., Meza, J., and Veeraraghaven, K. (2015). Gorilla: A Fast, Scalable, In-Memory Time Series Database. **Proc. VLDB Endow.**, 8(12):1816—-1827, http://dl.acm.org/citation.cfm?doid=2824032.2824078.

POLSTRACC (2019). POLar STRAtosphere in a Changing Climate (POLSTRACC). www.polstracc.kit.edu. [Online; accessed 2019-08-02].

Rabiner, L. and Schafer, R. (1978). Digital processing of speech signals.

Ratanaworabhan, P., Ke, J., and Burtscher, M. (2006). Fast lossless compression of scientific floating-point data. In **Data Compression Conf. Proc.**, number August, pages 133–142. IEEE, ISBN: 0-7695-2545-8, ISSN: 10680314, DOI: 10.1109/DCC.2006.35, http://ieeexplore.ieee.org/document/1607248/.

Rew, R. and Davis, G. (1990). NetCDF: An Interface for Scientific Data Access. **IEEE Comput. Graph. Appl.**, 10(4):76–82, ISBN: 0272-1716, ISSN: 02721716, DOI: 10.1109/38.56302.

Rice, R. and Plaunt, J. (1971). Adaptive variable-length coding for efficient compression of spacecraft television data. **IEEE Transactions on Communication Technology**, 19(6):889–897.

Richman, J. S. and Moorman, J. R. (2000). Physiological time-series analysis using approximate entropy and sample entropy. **Am. J. Physiol. Heart Circ. Physiol.**, 278(6):H2039–49, ISBN: 0363-6135 (Print) 0363-6135 (Linking), ISSN: 0363-6135, DOI: 10.1152/ajpheart.2000.278.6.H2039, http://ajpheart.physiology.org/content/278/6/H2039https://link.aps.org/doi/10.1103/PhysRevA.29.975http://www.ncbi.nlm.nih.gov/pubmed/10843903.

Robinson, T. (1994). SHORTEN: Simple lossless and near-lossless waveform compression. **Engineering**, pages 1–16.

Ryabko, B. Y. (1980). Data compression by means of a "book stack". **Problemy Peredachi Informatsii**, 16(4):16–21.

Salomon, D. and Motta, G. (2010). **Handbook of Data Compression**, volume 44. Springer London, London, ISBN: 978-1-84882-902-2, ISSN: 1751-8113, DOI: 10.1007/978-1-84882-903-9, http://link.springer.com/10.1007/978-1-84882-903-9.

Sasaki, N., Sato, K., Endo, T., and Matsuoka, S. (2015). Exploration of Lossy Compression for Application-Level Checkpoint/Restart. In **Proc. - 2015 IEEE 29th Int. Parallel Distrib. Process. Symp. IPDPS 2015**, pages 914–922. IEEE, ISBN: 9781479986484, ISSN: 1530-2075, DOI: 10.1109/IPDPS.2015.67, http://ieeexplore.ieee.org/document/7161577/.

Sauer, T. and Xu, Y. (2006). On Multivariate Lagrange Interpolation. **Math. Comput.**, 64(211):1147, ISSN: 00255718, DOI: 10.2307/2153487.

Schröter, J., Rieger, D., Stassen, C., Vogel, H., Weimer, M., Werchner, S., Förstner, J., Prill, F., Reinert, D., Zängl, G., Giorgetta, M., Ruhnke, R., Vogel, B., and Braesicke, P. (2018). ICON-ART 2.1: a flexible tracer framework and its application for composition studies in numerical weather forecasting and climate simulations. **Geosci. Model Dev.**, 11(10):4043–4068, DOI: 10.5194/gmd-11-4043-2018, https://www.geosci-model-dev.net/11/4043/2018/.

Schultz, M. H. (1970). Error bounds for polynomial spline interpolation. **Math. Comput.**, 24(111):507–507, ISSN: 0025-5718, DOI: 10.1090/S0025-5718-1970-0275025-9, http://www.ams.org/jourcgi/jour-getitem?pii=S0025-5718-1970-0275025-9.

Shannon, C. E. (1948). The mathematical theory of communication. 1963. **MD. Comput.**, 14(4):306–17, ISBN: 0252725484, ISSN: 0724-6811, DOI: 10.1002/j.1538-7305.1948.tb01338.x, http://www.ncbi.nlm.nih.gov/pubmed/9230594.

Srinivasan, K. and Reddy, M. R. (2010). Efficient preprocessing technique for real-time lossless EEG compression. **Electron. Lett.**, 46(1):26–27(1), ISSN: 0013-5194, DOI: 10.1049/el.2010.2349.

Sriraam, N. (2012). Correlation dimension based lossless compression of EEG signals. **Biomed. Signal Process. Control**, 7(4):379–388, ISSN: 17468094, DOI: 10.1016/j.bspc.2011.06.007, http://dx.doi.org/10.1016/j.bspc.2011.06.007.

Sriraam, N. and Eswaran, C. (2008). An adaptive error modeling scheme for the lossless compression of EEG signals. **IEEE Trans. Inf. Technol. Biomed.**, 12(5):587–594, ISSN: 10897771, DOI: 10.1109/TITB.2007.907981.

Tao, D., Di, S., Chen, Z., and Cappello, F. (2017a). Exploration of Pattern-Matching Techniques for Lossy Compression on Cosmology Simulation Data Sets. In **Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)**, volume 10524 LNCS, pages 43–54. ISBN: 9783319676296, ISSN: 16113349, DOI: 10.1007/978-3-319-67630-2_4, http://link.springer.com/10.1007/978-3-319-67630-2_4.

Tao, D., Di, S., Chen, Z., and Cappello, F. (2017b). Significantly Improving Lossy Compression for Scientific Data Sets Based on Multidimensional Prediction and Error-Controlled Quantization. In **2017 IEEE Int. Parallel Distrib. Process. Symp.**, pages 1129–1139. IEEE, ISBN: 978-1-5386-3914-6, DOI: 10.1109/IPDPS.2017.115, http://ieeexplore.ieee.org/document/7967203/.

Tao, D., Di, S., Liang, X., Chen, Z., and Cappello, F. (2018). Fixed-PSNR Lossy Compression for Scientific Data. **Proc. - IEEE Int. Conf. Clust. Comput. ICCC**, 2018-Septe:314–318, ISBN: 9781538683194, ISSN: 15525244, DOI: 10.1109/CLUSTER.2018.00048, http://arxiv.org/abs/1805.07384.

Tao, D., Di, S., Liang, X., Chen, Z., and Cappello, F. (2019). Optimizing Lossy Compression Rate-Distortion from Automatic Online Selection between SZ and ZFP. **IEEE Trans. Parallel Distrib. Syst.**, XX(X):1–1, ISSN: 1045-9219, DOI: 10.1109/tpds.2019.2894404.

Taquet, J. and Labit, C. (2012). Hierarchical oriented predictions for resolution scalable lossless and near-lossless compression of CT and MRI biomedical images. **IEEE Trans. Image Process.**, 21(5):2641–2652, ISBN: 1057-7149, ISSN: 10577149, DOI: 10.1109/TIP.2012.2186147.

Toderici, G., Vincent, D., Johnston, N., Hwang, S. J., Minnen, D., Shor, J., and Covell, M. (2017). Full resolution image compression with recurrent neural networks. In Gonzalez, T. F., editor, **Proc. IEEE Conf. Comput. Vis. Pattern Recognit.**, pages 5306—-5314, 2455 Teller Road, Thousand Oaks California 91320 United States of America. Chapman and Hall/CRC, ISBN: 9780429143793, ISSN: 08936080, DOI: 10.1201/9781420010749, http://methods.sagepub.com/book/neural-networkshttps://www.taylorfrancis.com/books/9781420010749.

Velegar, M., Erichson, N. B., Keller, C. A., and Kutz, J. N. (2019). Scalable diagnostics for global atmospheric chemistry using ristretto library (version 1.0). **Geoscientific Model Development**, 12(4):1525–1539, DOI: 10.5194/gmd-12-1525-2019, https://www.geosci-model-dev.net/12/1525/2019/.

Waring, E. (1779). VII. Problems concerning interpolations. **Philos. Trans. R. Soc. London**, 69:59–67, ISSN: 0261-0523, DOI: 10.1098/rstl.1779.0008.

Weinberger, M., Seroussi, G., and Sapiro, G. (2000). The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS. **IEEE Trans. Image Process.**, 9(8):1309–1324, ISBN: 1057-7149, ISSN: 10577149, DOI: 10.1109/83.855427, http://ieeexplore.ieee.org/document/855427/.

WMO (2013). General Regularly-distributed Information in Binary form. WMO FM 92-XII GRIB.

WMO (2019). WMO Integrated Global Observing System (WIGOS). https://public.wmo.int/en/about-us/vision-and-mission/wmo-integrated-global-observing-system. [Online; accessed 08-September-2019].

Wu, X. and Memon, N. (1997). Context-based, adaptive, lossless image coding. **IEEE Trans. Commun.**, 45(4):437–444, ISBN: 0090-6778, ISSN: 00906778, DOI: 10.1109/26.585919.

Zängl, G., Reinert, D., Rípodas, P., and Baldauf, M. (2015). The ICON (ICOsahedral Non-hydrostatic) modelling framework of DWD and MPI-M: Description of the non-hydrostatic dynamical core. **Q. J. R. Meteorol. Soc.**, 141(687):563–579, ISSN: 1477870X, DOI: 10.1002/qj.2378, http://doi.wiley.com/10.1002/qj.2378.

Ziv, J. and Lempel, A. (1977). A universal algorithm for sequential data compression. **IEEE Transactions on information theory**, 23(3):337–343.

This book was designed by Uğur Çayoğlu.
It was edited and set into type in Germany, and was
printed and bound by AStA in Karlsruhe, Germany.

The text face is Vollkorn, designed by Friedrich Althausen. It
is intended to be ‘modest and well working [with] dark and meaty
serifs and a bouncing and healthy look.’

The chapter face is Cantata One, designed by Joana Correia based in Porto.
Cantata One was originally inspired by hand written letters
made with a pointed pen on an old handmade map of New York City.

The section face is Lora, designed by Olga Karpushina and Alexei Vanyashin.
Lora is a serif with roots in calligraphy. It should convey the mood of a
modern-day story or an art essay.

The captions are set in Carlito, designed by Lukasz Dziedzic and is
based on Lato. It was initally issued by Google and is also known
as ‘Google’s Carlito font’.