CHALMERS
UNIVERSITY OF TECHNOLOGY

Predicting Fuel Consumption of Marine Vessels

Using Boosted Regression Trees and Structured State-space Sequence Models While Considering Weather Forecasts

Master's thesis in Complex Adaptive Systems

KARL LUNDGREN & ERIK NORLIN

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

Predicting Fuel Consumption of Marine Vessels

Using Boosted Regression Trees and Structured State-space Sequence Models While Considering Weather Forecasts

KARL LUNDGREN & ERIK NORLIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Automatic Control
Chalmers University of Technology
Gothenburg, Sweden 2024

Predicting Fuel Consumption of Marine Vessels
Using Boosted Regression Trees and Structured State-space Sequence Models While Considering Weather Forecasts
KARL LUNDGREN & ERIK NORLIN

Examiner and academic supervisor: Balázs Kulcsár (Chalmers)
Industrial supervisors: Simon Johansson (Cetasol), Fredrik Olsson (RISE), Luis Sanchez-Heres (RISE)

Master's Thesis 2024
Department of Electrical Engineering
Division of Systems and Control
Automatic Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: photograph of the Burö passenger ferry. Source: Västtrafik.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

Predicting Fuel Consumption of Marine Vessels
Using Boosted Regression Trees and Structured State-space Sequence Models While Considering Weather Forecasts
KARL LUNDGREN & ERIK NORLIN
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The increasing greenhouse gas (GHG) emissions from the maritime sector necessitate a reduction in fuel consumption to achieve the environmental sustainability goals adopted by IMO by 2050. A solution for this is to optimize the fuel consumption of marine vessels, which is enabled by prediction. Although physical models for predicting fuel consumption often fall short because of their inability to account for environmental factors, data-driven models such as XGBoost have shown promise in improving prediction accuracy. To accurately predict fuel consumption in marine vessels while considering weather forecasts, we explore the potential of XGBoost, a gradient-boosted tree ensemble, and S5, a structured state-space sequence model. We introduce a novel feature selection algorithm. We also apply balanced realization, an optimal state-space model reduction technique, to the S5 model, enhancing its efficiency, and analyze its effectiveness from a systems and control perspective. Our findings demonstrate the capabilities of these models in predicting fuel consumption, namely that XGBoost performs best, and highlight the importance of incorporating weather forecasts for improved accuracy, offering valuable insights for fuel optimization strategies in the maritime industry.

Keywords: fuel prediction and optimization, systems and control theory, marine weather and vessels, boosted regression trees, xgboost, structured state-space sequence models, s5, mamba, feature selection, balanced realization.

Acknowledgements

We would like to express our greatest gratitude towards our examiner and supervisor Balázs Kulcsár at the Department of Electrical Engineering at Chalmers for providing excellent guidance throughout our thesis, always showing optimism, kindness, and providing deep insights in systems and control theory that helped us tremendously. Special thanks is also devoted to Simon Johansson, data scientist at Cetasol, who has helped us understand the technical depth in this area of optimization, always being incredibly helpful, positive, and pushing the limits of our understanding in data science. Lastly, we would like to thank Fredrik Olsson and Luiz Sanchez-Heres at RISE for giving us the opportunity to take on this project, providing valuable feedback from a maritime perspective, and providing us with a fine workplace to conduct our thesis at. We wish you all the best for the future.

Karl Lundgren & Erik Norlin, Gothenburg, May 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

DPLR	Diagonal Plus Low Rank
EMAE	Envelope-weighted Mean Absolute Error
EMEDAE	Envelope-weighted Median Absolute Error
GBDT	Gradient Boosted Decision Trees
GBM	Gradient Boosting Machine
GBRT	Gradient Boosted Regression Trees
GELU	Gaussian Error Linear Unit
GHG	Green House Gas
HiPPO	High-order Polynomial Projection Operators
IMO	International Maritime Organization
JADOC	Joint Approximate Diagonalization Under Orthogonality Constraints
LTI	Linear Time-invariant
LegS	Scaled Legendre Measure
LPV	Linear parameter-varying
MAE	Mean Absolute Error
MIMO	Multiple Inputs Multiple Outputs
MRMR	Minimum Redundancy And Maximum Relevance
MSE	Mean Squared Error
NARX	Non-linear Autoregressive Exogenous Model
NPLR	Normal Plus Low Rank
PCA	Principal Component Analysis
RELU	Rectified Error Linear Unit
RNN	Recurrent Neural Network
SISO	Single Input Single Output
SOTA	State Of The Art
SSM	State-space Model
SVD	Singular Value Decomposition
S4	Structured State-space Sequence Model
S5	Simplified Structured State-space Sequence Model
UN	United Nations
XGBoost	eXtreme Gradient Boosting
XGB	XGBoost
ZOH	Zero-order Hold

Contents

List of Figures

List of Tables

1
Introduction

This chapter outlines the problem and why a solution is needed. The purpose and aim are defined as well as the limitations and ethics.

1.1 Background

In 2023, the global average temperature reached an all-time high record since the pre-industrial era (1850-1900), with an increase of 1.45 °C [1]. The trend of global warming, confirmed to be partly induced by human activities [2], tends to exacerbate extreme weather events and rising sea levels. These effects have serious consequences, for example, increased poverty, displacement, health risks, and species extinction [3].

One contributor to greenhouse gas (GHG) emissions is the maritime sector. According to the International Maritime Organization (IMO) [4], an agency of the United Nations responsible for regulating maritime safety, security, and pollution, the emissions of GHG of total shipping have increased 9.6 % from 2012 to 2018. During the same period, their share of global GHG emissions increased from 2.76 % to 2.89 %. In an effort to reduce maritime GHG emissions, the IMO increased its ambitions in 2023 by adopting a strategy to achieve net zero GHG emissions from international shipping by 2050 [5].

A solution to reduce the GHG missions in marine vessels is to optimize fuel consumption. With this, ferry operators can strategically plan speeds, which can result in lower costs for ferry services, improved operational efficiency, and reduced impact on the environment. In order to optimize fuel consumption for a given point in space and time, a mathematical function that determines fuel consumption must be evaluated. A simple approach to estimate fuel consumption given a certain speed is to use a physics-inspired model. However, according to Lang et al. (2022) [6], such models tend to produce underestimated results because they do not take into account environmental factors that affect the dynamics of the vessel. Data-driven predictive models, on the other hand, are able to make more accurate predictions because environmental effects are incorporated into the data that the model is trained on. The authors showed that, for their data and use case, *Extreme Gradient Boosting (XGBoost)* [7], a machine learning framework for boosted regression trees, known for having generated state-of-the-art (SOTA) results for prediction tasks, performed the best among various models when predicting fuel consumption of ships. These included linear regression, artificial neural networks, support vector machines, and physical models.

Furthermore, *structured state-space sequence models (S4)* [8], a novel approach to time series prediction, with its foundation in control theory, have received considerable attention within the deep learning community in recent years. These models still remain underexplored, both in terms of prediction of fuel consumption and in-depth analysis from a systems and control perspective.

1.2 Purpose

The purpose of this thesis is to provide more knowledge about how fuel consumption can be accurately predicted using boosted regression trees and structured state-space sequence models. In turn, the purpose of doing this is ultimately to allow for better fuel consumption optimization models. Furthermore, the purpose is also to bring new insights about structured state-space sequence models from a systems and control perspective, which there is a lack of.

1.3 Optimization of fuel consumption

In optimizing fuel consumption as defined here, the objective of interest is to provide online speed recommendations for a fixed-route marine vessel that optimize fuel consumption *per distance*. Already at the beginning of a journey, the fuel consumption per distance can be sequentially optimized for points in time until the end of the journey. As the journey progresses, this process can be repeated, updating the recommendations. This process can be performed by a combination of fuel prediction, integration techniques and, if there is a schedule to be kept, by using time constraints.

The reason for recommending speeds is that it is an intuitive control parameter for the captain of the vessel, while also allowing exact constraints of estimated time of arrival to be formulated. If there are no such constraints, the most appropriate control parameters are the engine power or engine speed, since these signals directly affect the engine's fuel consumption.

More precisely, an optimal speed can be obtained by predicting fuel consumption per distance over a range of speeds and choosing the speed corresponding to minimal fuel consumption. With this speed and knowing the usual route, the next course and position can be determined by numerical integration. At an integrated point in the future, another optimal speed can be determined in the same way. Performing this sequentially and noting the optimal speed for each point, a time series of optimal speeds can be constructed, which can be recommended to the captain already at the beginning of the journey. This process implies that, if any engine-related signal is to be used as a feature in the prediction of fuel consumption, it must first be predicted using speed to not break causality.

The subtlety here is the challenge when it comes to benchmarking the performance of predictive models for optimization. *Observed* speeds are necessary to be used for prediction because it is the only case where corresponding labels of fuel consumption per distance exist. If instead the speed was swept over a range in order to optimize, new data points would be introduced without corresponding labels. However, as any time series of speeds potentially could have been found by optimization already at the beginning of the journey, using observed speeds is technically similar to real-time prediction of optimal speeds. This validates the case for potential benchmarking.

In short, with observed speeds and fuel consumption of a fixed-route vessel, predictive models for fuel consumption can be created with corresponding test cases that mimic real-time prediction and optimization.

1.4 Aim

The aim is to investigate how XGBoost and the *simplified S4 (S5)* [9], can accurately predict the fuel consumption of a fixed-route vessel while also considering weather

to allow the possibility of optimizing fuel consumption with these models.

Additionally, S5 could be considered a substantial model due to its neural network structure and compressing the model could speed up the prediction time. For this reason, the objective also includes implementing balanced realization [10], a method for optimal compression of state-space models, on the S5 and evaluating its effectiveness from a systems and control perspective. Hence, contributing with new knowledge in this field.

To achieve this, the specific case of the Burö ferry, a fixed-route vessel operating in the Gothenburg archipelago, will be investigated. Time series sensor data collected from this vessel, together with forecasted weather data will be used. The aim is more specifically decomposed into the following objectives.

1. Conduct a statistical analysis to determine the dependence between weather and fuel consumption of the Burö ferry operating in the Gothenburg archipelago.
2. Develop a baseline and more advanced XGBoost and S5 models, with appropriate features, capable of predicting fuel consumption of the Burö ferry while accounting for weather disturbances.
3. Assess the performance of XGBoost and S5 for fuel prediction in terms of accuracy.
4. Perform balanced realization on a trained S5 model and assess its effectiveness from a systems and control perspective.

1.5 Limitations

Here, fuel consumption will be predicted but not optimized. The reason for this can be understood from Section 1.3.

1.6 Societal, ethical and ecological aspects

Enhancing predictive capabilities for fuel consumption can ultimately lead to better fuel efficiency and therefore to a more economically and environmentally sustainable maritime industry. With respect to data collection, the data used to train these predictive models is inherently untraceable to individuals, ensuring that integrity is not compromised.

2

Theory

The theoretical framework underpinning the thesis is explained in this chapter. It begins by establishing the fundamentals required within maritimity. The structure of the following sections is outlined similarly to how the pipeline of the implementation of fuel prediction would be; theory for data preparation is followed by theory for feature selection, which in turn is followed by theory defining XGBoost and S5. After this, performance benchmarking is described. Lastly, essential theory for the discussion is described.

2.1 Maritime prerequisites

2.1.1 Weather

The weather elements that are often encountered within maritimity are wind, waves, and current, where wind and waves can be broken down into the elements described in Table 2.1. The velocity of these weather elements is measured in $\mathrm{m\,s^{-1}}$ or knots and their corresponding directions are measured in degrees clockwise from north, where 0° means that the weather is coming from north, i.e. the vectors are pointing south. Each wave element has the properties height and period where wave height is the distance between a wave's top and valley, and period is the time for a wavelength to pass a specific point [11].

Table 2.1: Description of gust, wind waves, swell waves, and waves.

Weather element	Description
Gust	Burst of strong wind.
Wind waves	Waves induced by local wind, often short.
Swell waves	Smooth waves traveling longer distances.
Waves	Wind waves and swell waves combined.

The velocity, energy and power of a wave can be determined from its period and height [12]. Given gravity g $[\mathrm{m\,s^{-2}}]$, a wave period T [s], the phase velocity of the wave c_p $[\mathrm{m\,s^{-1}}]$ in deep waters, that is, when the depth of the water is greater than half the wavelength of the wave, is defined by

$$c_p = \frac{g}{2\pi} T. \tag{2.1}$$

Moreover, the energy of a wave E_{wave} [J m^{-2}] is determined by its height H [m], gravity g [m s^{-2}], and water density ρ [kg m^{-3}] as

$$E_{\text{wave}} = \frac{1}{16}\rho g H^2. \tag{2.2}$$

Combining the energy of a wave and the group velocity $c_g = c_p/2$ (deep water), the wave power P_{wave} [W m^{-1}] is then determined by

$$P_{\text{wave}} = E_{\text{wave}} c_g. \tag{2.3}$$

2.1.2 Vessel engine and navigation

Being able to interpret the performance of a vessel's engine and its movement is necessary for efficient navigation. Table 2.2 describes the fundamental measurements of the engine and navigation that provide captains with the essential information they need for operation. Unlike weather directions, 0° for heading and course means that their vectors are pointing towards the north rather than away from it.

Table 2.2: Description of vessel engine and navigation measurements.

Measurement	Description
Engine speed [RPM]	The rotational speed of the crankshaft.
Engine torque [N m]	The rotational force created by the crankshaft.
Engine power [W]	The power generated by the crankshaft.
Speed over ground [m s^{-1}]	The speed of the vessel measured by satellite.
Heading [°]	The direction in which the vessel is pointing, measured clockwise from the north.
Course [°]	The direction in which the vessel is moving, measured clockwise from north.
Longitude [°]	The east-west position from Greenwich
Latitude [°]	The north-south position from the equator

The engine speed ω and the engine torque τ are signals directly from the engine. With these, the engine power P_{engine} can be determined from

$$P_{\text{engine}} = \frac{2\pi\tau\omega}{60}. \tag{2.4}$$

2.1.3 The relationship between vessel speed and fuel consumption

The resistances acting on a vessel are composed of friction and pressure resistances of both air and water [13], [14]. Frictional resistance is the drag force induced by a fluid that flows and sticks to the surface of the vessel. The pressure resistance, on the other hand, consists of the resistance to drag and waves. Specifically, form drag comes from the pressure differences caused by the shape of the vessel, and the wave resistance comes from the making and breaking of waves. The power required for

the vessel to overcome all these resistances can be determined analytically. Starting with the general formula for the drag force F_{drag}

$$F_{\text{drag}} = \frac{1}{2}\rho v^2 C_D A, \tag{2.5}$$

where ρ is the density of the fluid, v is the velocity of the object through the fluid, C_D is the drag coefficient dependent on the Reynolds number measuring the smoothness of the fluid flow, and A is the reference area. Moreover, the equation for power P is

$$P = Fv. \tag{2.6}$$

The power P_{drag} required to overcome the drag of a (still) fluid can therefore be given by

$$P_{\text{drag}} = \frac{1}{2}\rho v^3 C_D A. \tag{2.7}$$

Applying this principle to all the drags that act on a vessel and putting them together results in a total power P_{total} with unique drag coefficients for friction and pressure and unique areas for air and water. The key point is that the relationship between the power required for the vessel to overcome all drags and its velocity is cubic and is also known as the *propeller law* [13], [14]. This is given by

$$P_{\text{total}} = kv^3, \tag{2.8}$$

for some constant k. However, note that this power does not equal the engine power required to overcome the drags because energy is converted into the system; an efficiency constant would have to be considered in k to determine this. Furthermore, engine power and fuel consumption are directly related and therefore this law can also describe the relationship between speed and fuel consumption. However, there is a caveat to this. This relationship only holds for calm weather; it may be infeasible to do this calculation at seas due to the complexity of environmental factors.

2.2 Data preparation

2.2.1 Correlation

A way of measuring the degree of association between two variables is to determine the **correlation** [15] between them. A positive correlation is a relationship between two variables in which if one variable increases, so does the other, and a negative correlation is a relationship in which if one variable increases, the other decreases.

When two variables are *collinear*, they are linearly dependent. *Multi-collinearity* is technically the same as the former but is used in the context when two prediction features are collinear.

Correlation does not mean causation, which means that one variable does not necessarily cause the effect on the other variable because they are correlated. Moreover, correlated features are dependent, but uncorrelated variables do not mean that the variables are independent. In other words, correlation is a subset of dependence.

2.2.1.1 Pearson's correlation coefficient

Fundamentally, **Pearson's correlation coefficient** [16] measures the *linear* dependence between two variables. For a population, given two random variables X and Y the Pearson's correlation coefficient $\rho_{X,Y}$ is determined as in

$$\rho_{X,Y} = \operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}, \tag{2.9}$$

if $\sigma_X \sigma_Y > 0$ where μ and σ are the mean and standard deviation of X and Y, respectively. $\rho_{X,Y} \in [-1,1]$ where -1 means perfect negative collinearity, +1 means perfect positive collinearity, and zero means that there is no linear dependence. For a sample this is known as **Pearson's** r denoted r_{xy} and defined as

$$r_{xy} = \frac{1}{n-1} \sum_{i=1}^{n} \left(\frac{x_i - \overline{x}}{s_x} \right) \left(\frac{y_i - \overline{y}}{s_y} \right), \tag{2.10}$$

where n is the sample size, x_i, y_i are the sample points, $\overline{x}$, $\overline{y}$ are the sample means and s_x, s_y are the sample standard deviations.

2.2.1.2 Spearman's rank correlation coefficient

In some cases the relationship between two variables might not be linear. **Spearman's rank correlation coefficient** [16] is a more general measure of correlation which indirectly captures the *non-linear* relationship between two variables. This is accomplished by determining the Pearson correlation coefficient of the rank variables $R(x)$, $R(y)$ instead. By ranking the values of the variables, non-linear changes become monotonically increasing. Hence, the relationship between the rank variables can be captured by the Pearson correlation coefficient. More precisely, for a population, the Spearman rank correlation coefficient ρ_s is described as

$$\rho_s = \rho_{R(X)R(Y)}, \tag{2.11}$$

whereas for a sample this is r_s defined as

$$r_s = r_{R(x)R(y)} = 1 - \frac{6 \sum_{i=1}^{n} (R(x_i) - R(y_i))^2}{n(n^2-1)}. \tag{2.12}$$

2.2.2 Principal component analysis

Data that appear to be high-dimensional can sometimes be described by a few latent factors, and therefore reducing the dimensionality of the data can be of interest. A method for doing this is **principal component analysis (PCA)** [17], which does this by projecting centered data to lower dimensions while preserving as much variance in the data as possible.

Consider a zero mean centered high-dimensional data set $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ for n data points and d dimensions. The *principal directions* $\boldsymbol{V} \in \mathbb{R}^{d \times d}$ contains columns with unit vectors that optimally project the data to lower dimensions such that the variance is maximized. For a basic case, each data point $\boldsymbol{x}_i \in \mathbb{R}^d$ is projected onto only the first principal direction $\boldsymbol{v}_1 \in \mathbb{R}^d$ as $y_i = \boldsymbol{v}_1^\top \boldsymbol{x}_i$, where $\boldsymbol{y}_1 = (y_1, y_2, \ldots, y_n)^\top \in \mathbb{R}^n$ is

the first *principal component*. This is done such that the reconstruction error, the mean square error (MSE), is minimized as

$$\mathrm{MSE}(\boldsymbol{v_1}) = \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|^2. \tag{2.13}$$

Here, $\tilde{\boldsymbol{x}}_i = \boldsymbol{v}_1\boldsymbol{v}_1^\top\boldsymbol{x}_i$ is the reconstruction in the original higher dimension. Expanding and simplifying yields

$$\begin{aligned}
\mathrm{MSE}(\boldsymbol{v_1}) &= \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i\|^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\|\boldsymbol{x}_i - \boldsymbol{v}_1 y_i\|^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - y_i\boldsymbol{v}_1)^\top(\boldsymbol{x}_i - y_i\boldsymbol{v}_1) \\
&= \frac{1}{n}\sum_{i=1}^{n}\Big(\boldsymbol{x}_i^\top\boldsymbol{x}_i - 2y_i\boldsymbol{v}_1^\top\boldsymbol{x}_i + y_i^2\boldsymbol{v}_1^\top\boldsymbol{v}_1\Big) \\
&= \frac{1}{n}\sum_{i=1}^{n}\Big(\boldsymbol{x}_i^\top\boldsymbol{x}_i - 2\Big(\boldsymbol{v}_1^\top\boldsymbol{x}_i\Big)^2 + y_i^2\Big) \\
&= \frac{1}{n}\sum_{i=1}^{n}\Big(\boldsymbol{x}_i^\top\boldsymbol{x}_i - \Big(\boldsymbol{v_1}^\top\boldsymbol{x}_i\Big)^2\Big).
\end{aligned} \tag{2.14}$$

Hence, minimizing the above is equivalent to maximizing the second term.
From basic statistics, the variance is defined as

$$\mathrm{Var}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2. \tag{2.15}$$

Hence, the variance of the projected data $\boldsymbol{y}_1$ is

$$\mathrm{Var}[\boldsymbol{y}_1] = \frac{1}{n}\sum_{i=1}^{n}\Big(\boldsymbol{v_1}^\top\boldsymbol{x}_i\Big)^2 - \left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{v_1}^\top\boldsymbol{x}_i\right)^2. \tag{2.16}$$

Since the mean of the data is zero, the variance is simply

$$\begin{aligned}
\mathrm{Var}[\boldsymbol{y}_1] &= \frac{1}{n}\sum_{i=1}^{n}\Big(\boldsymbol{v_1}^\top\boldsymbol{x}_i\Big)^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{v_1}^\top\boldsymbol{x}_i\boldsymbol{x}_i^\top\boldsymbol{v_1} \\
&= \boldsymbol{v_1}^\top\boldsymbol{C}\boldsymbol{v_1},
\end{aligned} \tag{2.17}$$

where $\boldsymbol{C} \in \mathbb{R}^{d\times d}$ is the covariance matrix. Thus,

$$\begin{aligned}
\boldsymbol{v}_1^* &= \operatorname*{argmin}_{\boldsymbol{v}_1}\mathrm{MSE}(\boldsymbol{v}_1) \\
&= \operatorname*{argmax}_{\boldsymbol{v}_1}\mathrm{Var}[\boldsymbol{y}_1].
\end{aligned} \tag{2.18}$$

Since $\boldsymbol{v}_1$ is a unit vector, the maximization problem is subjected to the constraint $\|\boldsymbol{v}\| = \boldsymbol{v}_1^\top \boldsymbol{v}_1 = 1$, which is relaxed with a Lagrange multiplier λ_1 as

$$\boldsymbol{v}_1^* = \underset{\boldsymbol{v}_1}{\operatorname{argmax}}\, \boldsymbol{v_1}^\top \boldsymbol{C} \boldsymbol{v_1} + \lambda_1 \Big(1 - \boldsymbol{v}_1^\top \boldsymbol{v}_1\Big). \tag{2.19}$$

First-order necessary conditions for optimality with respect to $\boldsymbol{v}_1$ give

$$\boldsymbol{C}\boldsymbol{v}_1^* = \lambda_1 \boldsymbol{v}_1^*. \tag{2.20}$$

Thus, $\boldsymbol{v}_1^*$ is the eigenvector of $\boldsymbol{C}$ corresponding to the maximal eigenvalue. The maximal variance of $\boldsymbol{y}_1$ is therefore equivalent to

$$\begin{aligned} \operatorname{Var}[\boldsymbol{y}_1]^* &= \boldsymbol{v}_1^{*\top} \boldsymbol{C} \boldsymbol{v}_1^* \\ &= \lambda_1 \boldsymbol{v}_1^{*\top} \boldsymbol{v}_1^* \\ &= \lambda_1. \end{aligned} \tag{2.21}$$

In the general case, this implies that the maximal variance of the j-th principal component $\boldsymbol{y}_j$ is equivalent to the j-th largest eigenvalue of $\boldsymbol{C}$ with the j-th eigenvector $\boldsymbol{v}_j^*$ corresponding to the j-th principal direction.

Moreover, $\boldsymbol{X}$ can be decomposed by the *singular value decomposition (SVD)* [18] as

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{2.22}$$

where $\boldsymbol{U} \in \mathbb{R}^{n \times n}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times d}$ containing the singular values along the $d \times d$ diagonal, and $\boldsymbol{V} \in \mathbb{R}^{d \times d}$. Hence,

$$\begin{aligned} \boldsymbol{C} &= \boldsymbol{X}^\top \boldsymbol{X} \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^\top \boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \\ &= \boldsymbol{V}\boldsymbol{\Sigma}^2 \boldsymbol{V}^\top \\ &= \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^\top, \end{aligned} \tag{2.23}$$

and

$$\begin{aligned} \boldsymbol{Y} &= \boldsymbol{X}\boldsymbol{V} \\ &= \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \boldsymbol{V} \\ &= \boldsymbol{U}\boldsymbol{\Sigma}. \end{aligned} \tag{2.24}$$

Thus, the principal directions can be obtained from the columns of $\boldsymbol{V}$, and the principal components can be obtained from the columns of $\boldsymbol{U}\boldsymbol{\Sigma}$ from the SVD of $\boldsymbol{X}$.

2.3 Feature selection for machine learning

Feature selection [19] is the process of identifying and removing as much irrelevant and redundant information from the data as possible, measured, for example, by correlation. Relevance, the extent of how well a feature describes the target, is preferred to be maximized because it can be difficult to establish relationships between the target and irrelevant features. Hence, irrelevant features do not necessarily help the model to accurately perform inference. Redundancy, which can be described as the similarity between two features, is preferred to be minimized. If one feature can be explained by another feature, it is sufficient to use one of them to describe the target. Removing irrelevancy and redundancy reduces the complexity of the model, making the training of the model lighter and more efficient, as well as making the model more interpretable.

2.3.1 Minimum redundancy and maximum relevance

A popular algorithm for feature selection is **minimum redundancy and maximum relevance (mRMR)** [20], which attempts to resemble the philosophy above. Algorithm 1 explains the details of this. As inputs, it takes features, a target, and the number of best features k to preserve. Then, for each feature, the relevance is calculated between the feature and the target. It continues to compute the redundancy between this feature and all other features and sums them to a total redundancy. The feature score is then determined by subtracting its relevance from its total redundancy. This procedure is repeated for all features. Finally, the scores are sorted in descending order, and the features corresponding to the k highest scores are selected.

Algorithm 1: mRMR

```
Input: features, target, k
for feature i in features do
    relevance = compute_relationship(feature i, target)
    redundancy = 0
    for feature j in features do
        redundancy += compute_relationship(feature i, feature j)
    end
    scores[feature i] = relevance - redundancy
end
Sort scores in descending order
Output: select the top k features
```

2.4 Defining XGBoost

2.4.1 Decision trees

A **decision tree** [21] is a binary tree-like flowchart structure used to represent a series of decisions and their potential outcomes. Each node in the tree splits data based on a yes-or-no question revolving a feature, and leads down to ending leaf nodes representing predicted outcomes. This structure can be leveraged as a supervised machine learning model for both regression and classification tasks where

training it yields an optimal way of structuring a decision-making binary tree given a set of features.

The process of training a decision tree is as following. At each node, the algorithm aims to split the data into subsets that are as homogeneous, or *pure*, as possible in terms of the target feature. There are a number of different ways to measure purity, one of them being *Gini impurity* which measures impurity as the likelihood of mislabeling data points within a subset. The algorithm considers all possible ways to split the data that forms a binary tree. It calculates the impurity reduction for each potential split and selects the split that leads to the greatest decrease in impurity.

One major advantage of decision trees is the simplicity of them. However, decision trees tend to be unstable, meaning that a slight change in the data can lead to very different series of splits. Also, they tend to severely overfit if the trees are too deep [22].

2.4.2 Boosting

Boosting [23] is a family of supervised machine learning algorithms and consists most commonly of chained weak learners to form one strong learner. A weak learner is a classifier that weakly correlates with the true classification, still though better than random guessing, whereas a strong learner correlates well with the true classification. Boosting is an iterative process. Every weak learner aims to improve the prediction from the previous learner and incorrect predictions are more heavily empathized in training.

2.4.2.1 Gradient boosting machine

In supervised machine learning we want to find some function $\hat{F}(\boldsymbol{x})$ that best approximates

$$\hat{F}(\boldsymbol{x}) = \underset{F}{\operatorname{argmin}}\, \mathbb{E}_{\boldsymbol{x},y}\left[\sum_{i=1}^{n} L(y_i, F(\boldsymbol{x}_i))\right], \tag{2.25}$$

given features $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, a target $\boldsymbol{y} \in \mathbb{R}^n$, a loss function L, n data points and d features. Based on boosting, **gradient boosting machine (GBM)** [24] chains a series of M number of weak learners $a(\boldsymbol{x})$ such that they are together converted to one strong learner $\hat{F}(\boldsymbol{x})$. Specifically, $\hat{F}(\boldsymbol{x})$ is defined by the weighted sum over all M weak learners as

$$\hat{F}(\boldsymbol{x}) = \sum_{m=1}^{M} \gamma_m a_m(\boldsymbol{x}) + \text{const.} \tag{2.26}$$

for some parameters γ_m. As in boosting, each weak learner aims to improve the prediction of the previous learners. Gradient boosting does this by adding the weak learner that minimizes the loss of the new prediction to the prediction of the previous learner as

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \left(\underset{a_m(\boldsymbol{x})}{\operatorname{argmin}}\left[\sum_{i=1}^{n} L(y_i, F_{m-1}(\boldsymbol{x}_i) + a_m(\boldsymbol{x}_i))\right]\right). \tag{2.27}$$

This problem can be simplified to a steepest to find a local minimum in the loss function by iterating over $F_m(\boldsymbol{x})$ as

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) - \gamma_m \sum_{i=1}^{n} \nabla_{F_{m-1}} L(y_i, F_{m-1}(\boldsymbol{x}_i)), \tag{2.28}$$

where the weak learner $a_m(x)$ is approximated to the negative gradients $g_m(\boldsymbol{x}_i)$ determined as

$$g_m(\boldsymbol{x}_i) = -\frac{\partial L(y_i, F_{m-1}(\boldsymbol{x}_i))}{\partial F_{m-1}(\boldsymbol{x}_i)}, \tag{2.29}$$

with the training set

$$\{\boldsymbol{x}_i, g_m(\boldsymbol{x}_i)\}_{i=1}^{n}. \tag{2.30}$$

The parameter γ_m is optimized as

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, F_{m-1}(\boldsymbol{x}_i) + \gamma a_m(\boldsymbol{x}_i)), \tag{2.31}$$

and the model is then updated accordingly

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \gamma_m a_m(\boldsymbol{x}). \tag{2.32}$$

Lastly, the final approximater becomes $\hat{F}(\boldsymbol{x}) = F_M(\boldsymbol{x})$.

2.4.2.2 Extreme gradient boosting

Introduced in 2016, **extreme gradient boosting** (**XGBoost**) [7], is a specific instance of gradient-boosted decision trees (GBDT) and regression trees (GBRT), which in turn are gradient-boosted type of models where each weak learner is a decision tree. The intuition of GBDT/GBRT is illustrated in Figure 2.1. It shows a sequence of decision trees in which each tree slightly improves the prediction of the previous one, causing the residual $r_i = y_i - \hat{y}_{m,i}$ to decrease sequentially. Together, they all form one strong learner that minimizes the residual $r_i = y_i - \hat{y}_{M,i}$.

The objective L to minimize for each tree (2.33) is similar to the one in the minimization problem (2.27) but with added regularization $\omega(a)$ which penalizes complexity of the trees in order to reduce overfitting.

$$L_m = \sum_{i=1}^{n} l[y_i, F_{m-1}(\boldsymbol{x}_i) + a_m(\boldsymbol{x}_i)] + \omega(a_m). \tag{2.33}$$

A second order Taylor approximation is performed on a convex error function l

$$L_m \approx \sum_{i=1}^{n} \left[l(y_i, F_{m-1}(\boldsymbol{x}_i)) + \left(g_m a_m + \frac{1}{2} h_m a_m^2 \right)(\boldsymbol{x}_i) \right] + \omega(a_m). \tag{2.34}$$

Removing constants with respect to $a(\boldsymbol{x})$ gives the new simplified objective

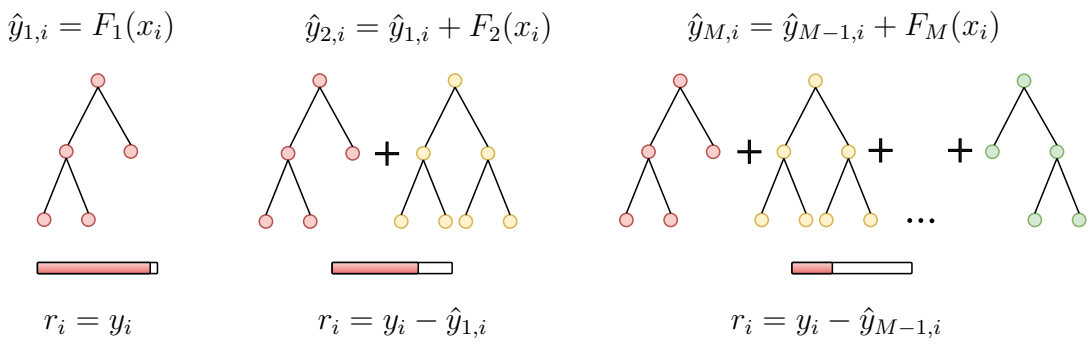

Figure 2.1: GBDT and GBRT consist of a sequence of decision trees, each tree improving the prediction of the previous one. Together they form one strong learner that minimizes the residual $r_i = y_i - \hat{y}_{M,i}$.

$$\tilde{L}_m = \sum_{i=1}^{n} \left(g_m a_m + \frac{1}{2} h_m a_m^2 \right) (\boldsymbol{x}_i) + \omega(a_m), \tag{2.35}$$

where $g_m(\boldsymbol{x}_i)$ and $h_m(\boldsymbol{x}_i)$ are the first and second derivatives of the loss

$$\begin{aligned} g_m(\boldsymbol{x}_i) &= \frac{\partial L(y_i, F_{m-1}(\boldsymbol{x}_i))}{\partial F_{m-1}(\boldsymbol{x}_i)}, \\ h_m(\boldsymbol{x}_i) &= \frac{\partial^2 L(y_i, F_{m-1}(\boldsymbol{x}_i))}{\partial F_{m-1}(\boldsymbol{x}_i)^2}. \end{aligned} \tag{2.36}$$

Moreover, each tree $a_m(\boldsymbol{x})$ is defined as

$$a_m(\boldsymbol{x}) = w_{q(\boldsymbol{x})} \in \mathbb{R}^T, q : \mathbb{R}^d \to \{1, 2, \ldots, T\}, \tag{2.37}$$

where leaf weights w is a vector containing the scores for the T number of leaf nodes, and q is a tree structure that assigns a data point $\boldsymbol{x}_i \in \mathbb{R}^d$ to a leaf node, where d is the number of features, to the corresponding leaf node index of the tree. Furthermore, the regularization $\omega(a)$ is defined according to

$$\omega(a) = \mu T + \frac{1}{2} \lambda \sum_{t=1}^{T} w_t^2, \tag{2.38}$$

with penalty constants μ and λ. Inserting the definition of the tree (2.37) and the regularization (2.38) into the objective (2.35) yields

$$\tilde{L}_m = \sum_{i=1}^{n} \left[g_m(\boldsymbol{x}_i) w_{q(\boldsymbol{x}_i)} + \frac{1}{2} h_m(\boldsymbol{x}_i) w_{q(\boldsymbol{x}_i)}^2 \right] + \mu T + \frac{1}{2} \lambda \sum_{t=1}^{T} w_t^2 \tag{2.39a}$$

$$= \sum_{t=1}^{T} \left[\left(\sum_{i \in I_t} g_m(\boldsymbol{x}_i) \right) w_t + \frac{1}{2} \left(\sum_{i \in I_t} h_m(\boldsymbol{x}_i) + \lambda \right) w_t^2 \right] + \mu T. \tag{2.39b}$$

Here, $I_t = \{i \mid q(\boldsymbol{x}_i) = t\}$ is the set of indices corresponding to data points that belong to node t and n is the number of data points. The summation index in (2.39b) is changed since data points within the same leaf node share the same score. Further letting $G_{m,t} = \sum_{i \in I_t} g_m(\boldsymbol{x}_i)$ and $H_{m,t} = \sum_{i \in I_t} h_m(\boldsymbol{x}_i)$ simplifies the expression to

$$\tilde{L}_m = \sum_{t=1}^{T} \left[G_{m,t} w_t + \frac{1}{2} \left(H_{m,t} + \lambda \right) w_t^2 \right] + \mu T. \tag{2.40}$$

First-order necessary conditions for optimality give

$$w_t^* = -\frac{G_t}{H_t + \lambda}, \tag{2.41}$$

resulting in the optimized objective

$$\tilde{L}_m^*(q) = -\frac{1}{2} \sum_{t=1}^{T} \frac{G_t^2}{H_t + \lambda} + \mu T. \tag{2.42}$$

That is, for a given tree structure q. Its measure can be interpreted as the impurity of the decision tree, but with the added benefit that it captures the complexity of it; the smaller the score, the better the model is. It should be noted that it is generally intractable to compute all possible tree structures. Instead in practise, one level of the tree at a time is optimized using a greedy algorithm; starting from one single leaf node, the optimal split is found by maximizing the loss reduction (score)

$$L = \frac{1}{2} \left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + H_R)^2}{H_L + H_R + \lambda} \right) - \mu, \tag{2.43}$$

where the first term is the loss of the left leaf, the second is the score of the right leaf, the third is the score on the original leaf, and the fourth is the regularization. Algorithm 2 describes this more precisely as following. First it takes the set of indices I_t for a node t, and a score of zero is initialized. Then for each feature k and data points x_{ik}, the loss reduction between before and after the split is computed. The score is updated to this loss reduction if it is greater than the old score. Lastly, the selected split at node t is the one corresponding to the maximum loss reduction.

Algorithm 2: XGBoost

Input: I_t
$G \leftarrow \sum_{i \in I_t} g(\boldsymbol{x}_i)$
$H \leftarrow \sum_{i \in I_t} h(\boldsymbol{x}_i)$
score = 0
for k=1 to d **do**
    $G_L \leftarrow 0$
    $H_L \leftarrow 0$
    Sort I_t in descending order by $\boldsymbol{x}_k$
    for i in I_t **do**
        $G_L \leftarrow G_L + g(x_{ik})$
        $H_L \leftarrow H_L + h(x_{ik})$
        $G_R \leftarrow G - G_L$
        $H_R \leftarrow H - H_L$
        score $\leftarrow \max\left(\text{score}, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda}\right)$
    end
end
Output: the split corresponding to the max score

To further reduce overfitting, in every boosting round when a new tree is added to the model, the leaf weights of the added tree are scaled by a small factor. This is called shrinkage and can be analogous to learning rate in stochastic optimization. It reduces the learning effect of the tree to leave room for other trees to enhance the model. Even more so, *bootstrap aggregating (bagging)* [25] on the features is also leveraged to reduce overfitting, meaning that each tree is trained using a random subset of the features, with replacement.

Building optimal decision trees is powerful but can also be computationally demanding. XGBoost supports several techniques for efficiency such as approximate algorithms for optimal split finding, an algorithm handling sparsity patterns, parallelization, cache-awareness, and data compression. Consequently, XGBoost is a scalable tree boosting system and shown state-of-the-art performance for many problems.

Here, y at one step ahead in the future is determined by its own history, the past and present of u, and the present ϵ.

2.5 Defining S5

2.5.1 State-space models

Introduced by Kalman (1960) [26], the **state-space model (SSM)** is a fundamental mathematical framework in control theory that describes how input signals of a linear time-invariant (LTI) system are mapped to output signals through latent states in time. It is governed by

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t), \end{aligned} \tag{2.44}$$

where $\boldsymbol{u}(t) \in \mathbb{C}^M$ are the input signals, $\boldsymbol{x}(t) \in \mathbb{C}^P$ are the latent states, and $\boldsymbol{y}(t) \in$

$\mathbb{C}^N$ are the output signals. $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$ are called *transition matrices* where $\boldsymbol{A} \in \mathbb{C}^{P\times P}$ is the state transition matrix describing the evolution of latent states over time, $\boldsymbol{B} \in \mathbb{C}^{P\times M}$ is the actuator matrix describing how the dynamics of the latent states are externally affected by the input signals, $\boldsymbol{C} \in \mathbb{C}^{N\times P}$ is the observation matrix and maps the latent states to the observed output states, and $\boldsymbol{D} \in \mathbb{C}^{N\times M}$ is a skip connection that relates the input directly to the output. An SSM is said to be single-input-single-output (SISO) for $M = N = 1$ and multiple-input-multiple-output (MIMO) for $M > 1$ and $N > 1$. A block diagram for a typical state-space model is shown in Figure 2.2

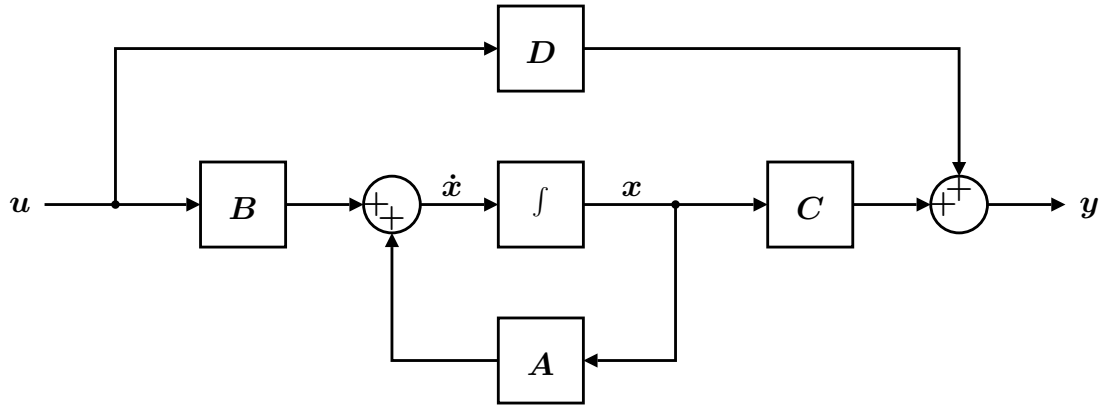

Figure 2.2: A typical state-space model.

Furthermore, in the time domain, the entire system can be characterized by the *impulse response* $\boldsymbol{g}(t) \in \mathbb{C}^{N\times M}$. Once known, it can be convolved with the input to obtain the output directly as in

$$\begin{aligned} \boldsymbol{y}(t) &= \boldsymbol{g}(t) * \boldsymbol{u}(t), \\ \boldsymbol{g}(t) &= \boldsymbol{C}e^{\boldsymbol{A}t}\boldsymbol{B} + \boldsymbol{D}\delta(t), \end{aligned} \tag{2.45}$$

where $\delta(t)$ is the Dirac delta function. Obtaining the output and impulse response from this is in many cases intractable due to performing convolution and matrix power operations. Instead, it is more convenient to compute the output $\boldsymbol{Y}(s) \in \mathbb{C}^N$ using the input $\boldsymbol{U}(s) \in \mathbb{C}^M$ and the *transfer function* $\boldsymbol{G}(s) \in \mathbb{C}^{N\times M}$ in frequency domain given by (2.46) where convolution and matrix powers corresponds to multiplication and inverting respectively.

$$\begin{aligned} \boldsymbol{Y}(s) &= \boldsymbol{G}(s)\boldsymbol{U}(s), \\ \boldsymbol{G}(s) &= \boldsymbol{C}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{B} + \boldsymbol{D}, \end{aligned} \tag{2.46}$$

where

$$s = \sigma + \mathrm{i}\omega \tag{2.47}$$

is a complex frequency, simply known as the Laplace variable.

Transforming (2.46) into the time domain by an inverse Laplace transform yields the impulse response and the time dependent output, and vice-versa:

$$\mathcal{L}(\boldsymbol{g}(t)) = \boldsymbol{G}(s) \iff \mathcal{L}^{-1}(\boldsymbol{G}(s)) = \boldsymbol{g}(t). \tag{2.48}$$

2.5.1.1 Discretization

To represent a discrete linear sequence as a state-space model, it is necessary to discretize (2.44) with step size Δ. This yields (2.49) with discretized transition matrices $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}$, and $\overline{\boldsymbol{D}}$.

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \overline{\boldsymbol{A}}\boldsymbol{x}_k + \overline{\boldsymbol{B}}\boldsymbol{u}_k, \\ \boldsymbol{y}_k &= \overline{\boldsymbol{C}}\boldsymbol{x}_k + \overline{\boldsymbol{D}}\boldsymbol{u}_k. \end{aligned} \tag{2.49}$$

One method of discretization is to assume **zero-order hold (ZOH)** [27] for the input, which means that the system samples the latent state given an input and holds this value until the system samples a new latent state given the next input. This results in a continuous output and follows an exact discretization in time domain of the state-space with transition matrices according to

$$\begin{aligned} \overline{\boldsymbol{A}} &= e^{\boldsymbol{A}\Delta}, \\ \overline{\boldsymbol{B}} &= \boldsymbol{A}^{-1}(\overline{\boldsymbol{A}} - \boldsymbol{I})\boldsymbol{B}, \\ \overline{\boldsymbol{C}} &= \boldsymbol{C}, \\ \overline{\boldsymbol{D}} &= \boldsymbol{D}. \end{aligned} \tag{2.50}$$

In case of a non-invertible $\boldsymbol{A}$, the input matrix instead takes the form

$$\overline{\boldsymbol{B}} = \left(\int_{\tau=0}^{\Delta t} \mathrm{e}^{\tau \boldsymbol{A}} \mathrm{d}\tau \right) \boldsymbol{B}. \tag{2.51}$$

2.5.1.2 Transformed state-spaces

An SSM (2.44) can be transformed to a different but equivalent SSM [10] as

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}}(t) &= \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}(t) + \tilde{\boldsymbol{B}}\boldsymbol{u}(t), \\ \boldsymbol{y}(t) &= \tilde{\boldsymbol{C}}\tilde{\boldsymbol{x}}(t) + \tilde{\boldsymbol{D}}\boldsymbol{u}(t), \end{aligned} \tag{2.52}$$

where

$$\begin{aligned} \tilde{\boldsymbol{x}}(t) &= \boldsymbol{T}^{-1}\boldsymbol{x}(t), \\ \tilde{\boldsymbol{A}} &= \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}, \\ \tilde{\boldsymbol{B}} &= \boldsymbol{T}^{-1}\boldsymbol{B}, \\ \tilde{\boldsymbol{C}} &= \boldsymbol{C}\boldsymbol{T}, \\ \tilde{\boldsymbol{D}} &= \boldsymbol{D}. \end{aligned} \tag{2.53}$$

Notice that the input $\boldsymbol{u}(t)$ and the output $\boldsymbol{y}(t)$ persist the same. A typical transformation and special case is that if $\boldsymbol{A}$ is diagonalizable by $\boldsymbol{T}$, being its eigenvectors, then the new SSM is a *diagonalized SSM*, where $\tilde{\boldsymbol{A}} = \boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues of $\boldsymbol{A}$.

2.5.1.3 Stability

The state dynamics of a state-space model captured by $\boldsymbol{A}$ can be categorized as stable or unstable. This is called *stability* [28]. A stable system means that the states converge to a stable point as $t \to \infty$, similar to a ball between two hills. If nudging the ball a bit, it will roll back to a stable point. In contrast, an unstable system means that its dynamics diverge and its values explode to $\pm\infty$ as $t \to \infty$.

The stability of a system can be mathematically determined by the eigenvalues of $\boldsymbol{A}$. A continuous linear system is stable if all real parts of the eigenvalues are negative and unstable if any real part of the eigenvalues is positive. A discrete system is stable if the magnitudes of the potentially complex eigenvalues are between -1 and 1, otherwise it is unstable. If any eigenvalue is complex with a non-zero imaginary part, the dynamics will exhibit oscillatory behavior.

Using methods from robust control theory, more advanced criteria of stability can be formulated, taking the whole system from input to output into account instead of only looking naively at the stability of the state transition matrix $\boldsymbol{A}$. One such example is the *small-gain theorem* [29] which states that a system is stable from input to output if

$$\|\boldsymbol{G}\|_{\mathcal{H}_\infty} \cdot \ldots \cdot \|\boldsymbol{H}\|_{\mathcal{H}_\infty} \leq 1, \tag{2.54}$$

where $\boldsymbol{G}$, $\boldsymbol{H}$ and potentially others, are connected transfer functions, and $\|\cdot\|_{\mathcal{H}_\infty}$ is the $\mathcal{H}_\infty$-norm defined as

$$\|\boldsymbol{G}\|_{\mathcal{H}_\infty} = \max_s \sigma_1(\boldsymbol{G}(s)). \tag{2.55}$$

2.5.1.4 Controllability and observability

Controllability and observability [30] describe to what extent the dynamical behavior of a state-space can be manipulated and understood.

Controllability is the ability to control the dynamics of a state-space to any desired state from an initial condition given the available inputs $\boldsymbol{u}(t)$. A controllable system gives the flexibility to manipulate dynamical behaviors.

Observability, on the other hand, is the ability to estimate any state from available output measurements $\boldsymbol{y}(t)$. An observable system can give a better understanding of the dynamical behaviors of a system.

Mathematically, their definitions are similar. The degree to which the states are controllable and observable can be determined by the *controllability* and *observability Gramians*, $\boldsymbol{W}_c(t)$ and $\boldsymbol{W}_o(t)$. These are described by

$$\begin{aligned} \boldsymbol{W}_c(t) &= \int_0^t e^{\boldsymbol{A}\tau}\boldsymbol{B}\boldsymbol{B}^* e^{\boldsymbol{A}^*\tau}\mathrm{d}\tau, \\ \boldsymbol{W}_o(t) &= \int_0^t e^{\boldsymbol{A}\tau}\boldsymbol{C}^*\boldsymbol{C} e^{\boldsymbol{A}^*\tau}\mathrm{d}\tau, \end{aligned} \tag{2.56}$$

and are most often evaluated where $t \to \infty$.

Basically, the larger the eigenvalues of $\boldsymbol{W}_c$ and $\boldsymbol{W}_o$, the more controllable or observable the system. However, calculating the Gramians by their definitions is often impractical. Instead, if the system is they can be obtained through the *Lyapunov equations*

$$\begin{aligned} \boldsymbol{A}\boldsymbol{W}_c + \boldsymbol{W}_c\boldsymbol{A}^* + \boldsymbol{B}\boldsymbol{B}^* = \boldsymbol{0}, \\ \boldsymbol{A}^*\boldsymbol{W}_o + \boldsymbol{W}_o\boldsymbol{A} + \boldsymbol{C}^*\boldsymbol{C} = \boldsymbol{0}. \end{aligned} \tag{2.57}$$

2.5.2 High-order polynomial projection operators

When modeling long-term, intricate temporal dependencies, integrating information from previous time-steps is crucial. Most popular machine learning models, however, all exhibit a common limitation: forgetfulness. For example, the attention mechanism has non-dynamic context windows while classical recurrent neural networks (RNNs) generally suffer from the vanishing gradient problem.

Given a signal evolving in time, i.e., a continuous function $\boldsymbol{f}(t) \in \mathbb{R}$, is it possible to construct a constant-sized representation $\boldsymbol{c}(t) \in \mathbb{R}^P$ which effectively encodes the entire history of $\boldsymbol{f}(t)$ from time 0 to t?

The **High-order Polynomial Projection Operators (HiPPO)** gives a solution to this, requiring two things: a measure to quantify the quality of the approximation, and an appropriate subspace. Making use of an orthogonal polynomial basis, a common tool from approximation theory, it was shown in [31] that not only can the compression $\boldsymbol{c}(t)$ be approximated, but can even be solved optimally in closed-form, according to some chosen measure.

More formally, a space of square integrable functions L^2 with probability measure μ on some domain $\mathcal{A}$ is a Hilbert space $\mathcal{H}_\mu$ with the inner product

$$\langle f, g \rangle_\mu = \int_{\mathcal{A}} \overline{f(x)} g(x) \mathrm{d}\mu(x), \tag{2.58}$$

where $\overline{f(x)}$ is the complex conjugate of $f(x)$. This induces a norm

$$\|f\|_{L^2(\mu)} = \langle f, f \rangle_\mu^{1/2}. \tag{2.59}$$

Although any subspace could be used, the polynomial basis, $\mathcal{G}$, is suitable and is a more general basis than, for example, the Fourier basis.

Since $f_{\leq t} \in \mathbb{R}_{\leq t}$, the HiPPO framework defines the probability measure on the same domain and also lets it vary in time, yielding the measure $\mu^{(t)}$.

An optimization problem can now be formed as

$$\underset{g^{(t)}}{\operatorname{argmin}} \left\| f_{\leq t} - g^{(t)} \right\|_{L^2(\mu^{(t)})}, \tag{2.60}$$

where $g^{(t)} \in \mathcal{G}$. A natural choice for $g^{(t)}$ are polynomials orthogonal with respect to the measure $\mu^{(t)}$, i.e., when Equation (2.58) equals zero for two polynomials f and g.

The coefficients of such polynomials can be expressed as the projection

$$c_n^{(t)} = \left\langle f_{\leq t}, g_n^{(t)} \right\rangle_{\mu^{(t)}}, \tag{2.61}$$

where $n < P$.

This projection can now be differentiated with respect to t, which yields a linear parameter-varying (LPV) differential equation.

$$\dot{\boldsymbol{c}}(t) = \boldsymbol{A}(t)\boldsymbol{c}(t) + \boldsymbol{B}(t)\boldsymbol{f}(t). \tag{2.62}$$

For the novel *scaled Legendre measure* (LegS), here defined on the half-open interval as

$$\mu^{(t)} = \frac{1}{t}\mathbb{I}_{(0,t]}, \tag{2.63}$$

and a basis of *Legendre polynomials*, Equation (2.62) becomes

$$\dot{\boldsymbol{c}}(t) = -\frac{1}{t}\boldsymbol{A}\boldsymbol{c}(t) + \frac{1}{t}\boldsymbol{B}\boldsymbol{f}(t), \tag{2.64}$$

which is still LPV; the time dependence is now found in the factors $\frac{1}{t}$, effectively giving less importance to old inputs. If discretized using zero-order hold, this turns into

$$\boldsymbol{c}(t+\Delta t) = \mathrm{e}^{\Delta t\boldsymbol{A}}\boldsymbol{c}(t) + \left(\int_{\tau=0}^{\Delta t} \mathrm{e}^{\tau\boldsymbol{A}}\mathrm{d}\tau\right)\boldsymbol{B}\boldsymbol{f}(t). \tag{2.65}$$

If $\boldsymbol{A}$ is invertible, this simplifies to

$$\boldsymbol{c}(t+\Delta t) = \mathrm{e}^{\Delta t\boldsymbol{A}}\boldsymbol{c}(t) + \boldsymbol{A}^{-1}\left(\mathrm{e}^{\Delta t\boldsymbol{A}} - \boldsymbol{I}\right)\boldsymbol{B}\boldsymbol{f}(t) \tag{2.66}$$

which has the closed-form solution

$$\boldsymbol{A}_{n,k} = -\begin{cases} \sqrt{(2n+1)(2k+1)} & \text{if } n > k \\ n+1 & \text{if } n = k \\ 0 & \text{if } n < k \end{cases}, \quad \boldsymbol{B}_n = \sqrt{2n+1}. \tag{2.67}$$

We follow the nomenclature of previous works [32] and call this the HiPPO-LegS matrix. Diagonal state dynamics is desirable to be able to run the system (2.49) recurrently using parallel scans [9], allowing efficient computations of the matrix powers that arise. Unfortunately, this matrix cannot be numerically stably diagonalized [33] as it is not a normal matrix and thus requires another option.

The HiPPO-LegS matrix can be also decomposed into a normal plus low-rank (NPLR) form

$$\boldsymbol{A} = \boldsymbol{A}^{(N)} - \boldsymbol{P}\boldsymbol{P}^{\top}, \tag{2.68}$$

where

$$\boldsymbol{A}^{(N)}_{n,k} = -\begin{cases} \sqrt{(n+\frac{1}{2})(k+\frac{1}{2})} & \text{if } n > k \\ \frac{1}{2} & \text{if } n = k \\ \sqrt{(n+\frac{1}{2})(k+\frac{1}{2})} & \text{if } n < k. \end{cases} \tag{2.69}$$

is the normal matrix and

$$\boldsymbol{P}_n = \sqrt{n + \frac{1}{2}} \tag{2.70}$$

is a low-rank matrix. Here $\boldsymbol{A}^{(N)}$ is called the HiPPO-LegS-N matrix, or just HiPPO-N for short. Using this instead of the HiPPO-LegS matrix was empirically found to perform equally well while also lending itself to be diagonalized with a unitary matrix by the spectral theorem

$$\boldsymbol{\Lambda}^{(D)} = \mathrm{eig}\left(\boldsymbol{A}^{(N)}\right). \tag{2.71}$$

2.5.3 Wiener models

A **Wiener model** [34], shown in Figure 2.3, is a linear dynamic system with a proceeding non-linearity. The linear system takes an input u, and the linear output is mapped to a non-linear output y through a non-linearity. The purpose of the non-linearity is to capture more complex dynamics of a system.

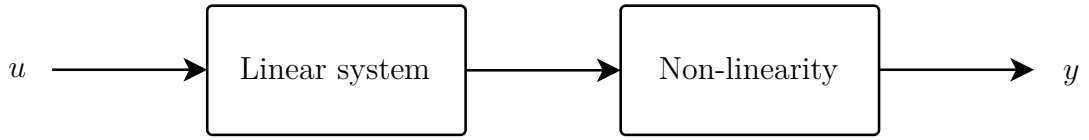

Figure 2.3: The Wiener model takes an input u to a linear system and maps the linear output to a non-linear output y.

2.5.3.1 Structured state-space sequence models

An approach to approximate non-linear dynamics is to construct an SSM as a neural network where its transition matrices are trainable parameters, and adding a sequence of dense layers and activations after the SSM. This is what the **structured state-space sequence model (S4)** [8] aims to do, which is nothing else than a *deep Wiener model* [35], if multiple S4s are chained sequentially. S4 initializes $\boldsymbol{A}, \boldsymbol{B}$ with the HiPPO-LegS framework, as this greatly improves performance by allowing the latent states to remember the history of the inputs described in Section 2.5.2.

By discretizing the SSMs within a deep S4, an impulse response for each SSM can be obtained in closed form by unrolling the dynamics to a vector $\overline{\boldsymbol{g}}$. This can be utilized specifically for training the model by obtaining the entire output sequence directly with convolution. This is in fact a naive approach and is computationally infeasible for long sequences of data due to repeated matrix exponentials of $\boldsymbol{A}$. To overcome this, $\overline{\boldsymbol{g}}$ is instead approximated in the frequency domain, greatly simplifying the computation. However, for this computation to be effective, $\boldsymbol{A}$ must be diagonal or parameterized as diagonal plus low rank (DPLR), that is, $\boldsymbol{\Lambda} - \boldsymbol{P}\boldsymbol{P}^\top$ for a diagonal $\boldsymbol{\Lambda}$ and a low-rank term $\boldsymbol{P}$. As mentioned previously, HiPPO-LegS cannot be diagonalized numerically stably since it is not a normal matrix, but it can be parametrized as NPLR as in (2.68). In turn, $\boldsymbol{A}^{(N)}$ can be diagonalized numerically stably, and by rearranging the terms, a numerically stable DPLR expression $\tilde{\boldsymbol{\Lambda}}_{\mathrm{DPLR}}$ can be decomposed from $\boldsymbol{A}$ as follows.

$$\begin{aligned} \boldsymbol{A} &= \boldsymbol{A}^{(N)} - \boldsymbol{P}\boldsymbol{P}^\top \\ &= \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^* - \boldsymbol{P}\boldsymbol{P}^\top \\ &= \boldsymbol{V}(\boldsymbol{\Lambda} - (\boldsymbol{V}^*\boldsymbol{P})(\boldsymbol{V}^*\boldsymbol{P})^*)\boldsymbol{V}^* \\ &= \boldsymbol{V}(\boldsymbol{\Lambda} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{V}^*, \end{aligned} \tag{2.72}$$

where $\boldsymbol{Q} = \boldsymbol{V}^*\boldsymbol{P}$. The SSMs within the model can be transformed into different but equivalent SSMs according to Section 2.5.1.2, where $\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{\Lambda}}_{\text{DPLR}} = \boldsymbol{\Lambda} - \boldsymbol{Q}\boldsymbol{Q}^*$.

With regard to dynamical stability, Goel et al. (2022) [36] showed that the stability of $\tilde{\boldsymbol{\Lambda}}_{\text{DPLR}}$, in terms of eigenvalues, depends only on the dynamical stability of $\boldsymbol{\Lambda}$, not on the low-rank term. They empirically found that allowing $\boldsymbol{\Lambda}$ to train freely, all eigenvalues remain stable on the basis of stable initialization of the eigenvalues, which $\boldsymbol{A}^{(N)}$ provides. Unfortunately, there is lack of theoretical evidence to support this as a fact. However, one can regularize the eigenvalues during training by taking the real parts to the exponent to penalize them when they are positive.

The S4 is inherently a SISO model, but allows for a pseudo-MIMO setting by stacking parallel S4 layers and mixing all outputs in the *proceeding* dense layers to model coupled non-linear dynamics. A more generalized and efficient architecture of S4 is the **simplified S4 (S5)** [9] which is a true MIMO model in which controllable states are coupled by the transition matrices, i.e. *inside* the SSMs. The true MIMO nature of S5 makes it perform just as well as S4 but with far fewer parameters. The complete architecture of the S5 is shown in Figure 2.4.

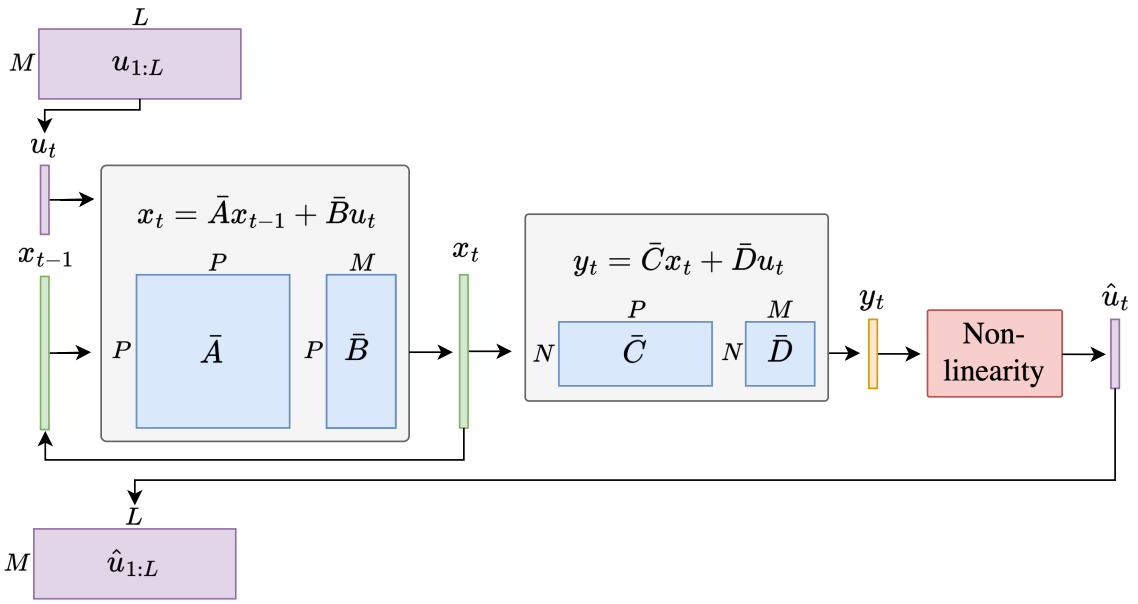

Figure 2.4: The architecture of S5 is analogous to a Wiener model; the linear block consists of a state-space layer, and the output goes through an arbitrary non-linearity. L here is the length of the time series to predict and $\hat{u}_t$ is the prediction, which can be used as an input to a proceeding S5-layer.

Although the convolutional approach is not available for S5, the computational and spatial complexities for both training and prediction are the same as for S4 using a parallel scanning algorithm. A benefit of training recurrently with this algorithm is

that it allows to train on irregularly sampled data, which is not possible with the convolutional approach in the case of S4. However, for the scanning algorithm to be efficient, $\boldsymbol{A}$ in this case must be diagonal. This is achieved by initializing $\boldsymbol{A}$ with $\boldsymbol{A}^{(N)}$ instead of HiPPO-LegS. Gu et al. (2022) [32] showed that in the limit of infinite latent state dimensions, an SSM initialized with HiPPO-N converges to an SSM initialized with HiPPO-LegS anyway, explaining why this initialization still performs well. By performing the numerically stable diagonalization $\boldsymbol{A}^{(N)} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^*$, the SSMs within S5 can, similarly to before, be transformed into different but equivalent SSMs according to Section 2.5.1.2, where $\tilde{\boldsymbol{A}} = \boldsymbol{\Lambda} = \boldsymbol{V}^{-1}\boldsymbol{A}^{(N)}\boldsymbol{V}$.

Since training is performed in the time domain, ZOH is used for discretization as this provides an exact discretization. Inserting the transformed transition matrices into the equation for ZOH (2.50) yields the following.

$$\begin{aligned}
\overline{\boldsymbol{\Lambda}} &= e^{\boldsymbol{V}^{-1}\boldsymbol{A}^{(N)}\boldsymbol{V}\boldsymbol{\Delta}}, \\
\overline{\boldsymbol{B}} &= (\boldsymbol{V}^{-1}\boldsymbol{A}^{(N)}\boldsymbol{V})(\overline{\boldsymbol{\Lambda}} - \boldsymbol{I})(\boldsymbol{V}^{-1}\boldsymbol{B}), \\
\overline{\boldsymbol{C}} &= \boldsymbol{C}\boldsymbol{V}, \\
\overline{\boldsymbol{D}} &= \boldsymbol{D},
\end{aligned} \tag{2.73}$$

where $\boldsymbol{\Delta} \in \mathbb{R}^P$ contains one time delta step for each latent state, also trainable parameters. S5 tends to perform better with this setting than having a single time delta step.

Regarding prediction, it is generally known that the inference speed of autoregressive models is one of their major limitations. This is because passing the whole sequence is required for every new sample. The S5 on the other hand performs prediction recurrently which only requires constant computation and memory per time step unlike conventional autoregressive models that scales with the context length.

2.5.4 Balanced realization

When state-space models are large, it may be desirable to compress them. This can be done with **balanced realization** [10] that transforms a state-space with a change of variable such that the input $\boldsymbol{u}(t)$ and output $\boldsymbol{y}(t)$ remain the same while also balancing the gramians of the system. This is desirable because it induces "dead" states in the system. These are states which now are neither particularly controllable or observable, which can then safely be removed by simple truncation without affecting the input-output energy (or variance) of the system too much, i.e., the transfer function will remain largely the same.

More specifically, given the state-space (2.44) let $\boldsymbol{x}(t) = \boldsymbol{T}\boldsymbol{z}(t)$ where $\boldsymbol{z}(t) \in \mathbb{C}^P$ and $\boldsymbol{T} \in \mathbb{C}^{P\times P}$, the state-space is transformed according to Section 2.5.1.2 as

$$\begin{aligned}
\dot{\boldsymbol{z}}(t) &= \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}\boldsymbol{z}(t) + \boldsymbol{T}^{-1}\boldsymbol{B}\boldsymbol{u}(t), \\
\boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{T}\boldsymbol{z}(t) + \boldsymbol{D}\boldsymbol{u}(t),
\end{aligned} \tag{2.74}$$

with new transition matrices

$$\begin{aligned}\hat{\boldsymbol{A}} &= \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T},\\ \hat{\boldsymbol{B}} &= \boldsymbol{T}^{-1}\boldsymbol{B},\\ \hat{\boldsymbol{C}} &= \boldsymbol{C}\boldsymbol{T},\\ \hat{\boldsymbol{D}} &= \boldsymbol{D}.\end{aligned} \tag{2.75}$$

It is called *balanced* because the controllability and observability Gramians, $\boldsymbol{W}_c$ and $\boldsymbol{W}_o$, are balanced in the frame of the transformed system. This transformation, along with the truncation, is shown in Figure 2.5.

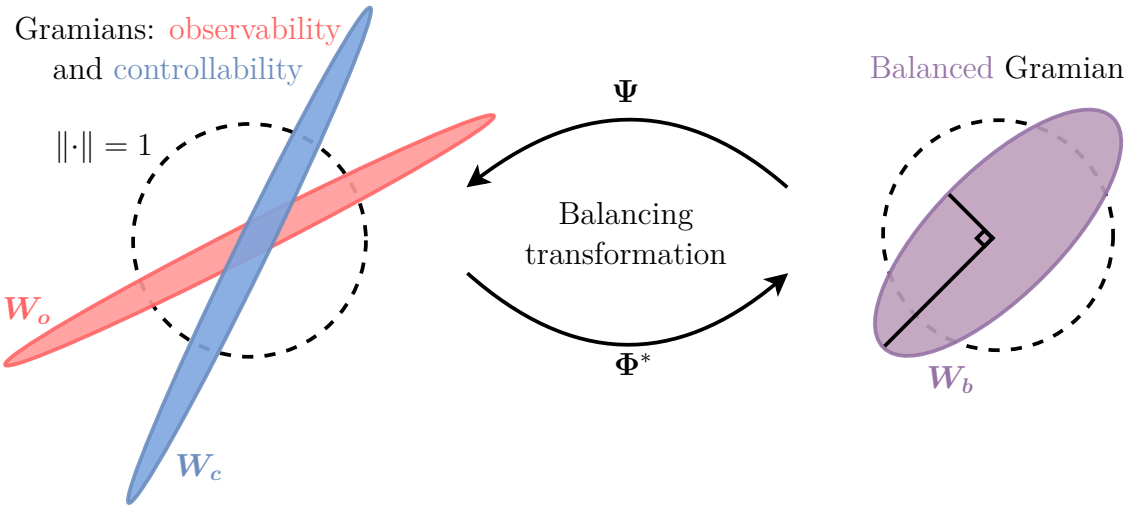

Figure 2.5: Balancing the controllability and observability gramians by a transformation.

Precisely, $\hat{\boldsymbol{W}}_c = \hat{\boldsymbol{W}}_o = \boldsymbol{\Sigma} \in \mathbb{C}^{P\times P}$ where $\boldsymbol{\Sigma}$ is diagonal and is referred to as the *Hankel matrix* containing the *Hankel singular values*. The system's new Gramians are obtained by inserting (2.53) into (2.56) with $t \to \infty$ and simplifying, yielding

$$\begin{aligned}\hat{\boldsymbol{W}}_c &= \boldsymbol{T}^{-1}\boldsymbol{W}_c\boldsymbol{T}^{-*},\\ \hat{\boldsymbol{W}}_o &= \boldsymbol{T}^{*}\boldsymbol{W}_o\boldsymbol{T}.\end{aligned} \tag{2.76}$$

For this evaluation of t, the Gramians are only feasible to compute for stable systems, since $\boldsymbol{A}$ is contained in the exponent in (2.56). Balanced realization is therefore performed only on the *stable* states of the system. Moreover, the product of the new Gramians are

$$\hat{\boldsymbol{W}}_c\hat{\boldsymbol{W}}_o = \boldsymbol{T}^{-1}\boldsymbol{W}_c\boldsymbol{W}_o\boldsymbol{T} = \boldsymbol{\Sigma}^2, \tag{2.77}$$

$$\Rightarrow \boldsymbol{W}_c\boldsymbol{W}_o\boldsymbol{T} = \boldsymbol{T}\boldsymbol{\Sigma}^2. \tag{2.78}$$

Notice that (2.78) is the equation for the eigendecomposition of $\boldsymbol{W}_c\boldsymbol{W}_o$ where $\boldsymbol{T}$ contains the eigenvectors, and the squared Hankel singular values are the eigenvalues. Since $\boldsymbol{W}_c$ and $\boldsymbol{W}_o$ can be computed from the original transition matrices, $\boldsymbol{T}$ can be retrieved from the eigendecomposition of $\boldsymbol{W}_c\boldsymbol{W}_o$. However, doing this could produce any scaling of the eigenvectors, which could result in $\hat{\boldsymbol{W}}_c \neq \hat{\boldsymbol{W}}_o$ even though (2.77) is

satisfied. Though, most computational software use standard scaling that scales the eigenvectors such that they have unit norm. Hence, the eigendecomposition yields

$$\begin{aligned} \boldsymbol{T}_u^{-1}\boldsymbol{W_c}\boldsymbol{T}_u^{-*} &= \boldsymbol{\Sigma}_c, \\ \boldsymbol{T}_u^{*}\boldsymbol{W_o}\boldsymbol{T}_u &= \boldsymbol{\Sigma}_o, \end{aligned} \tag{2.79}$$

with *unscaled* eigenvectors $\boldsymbol{T}_u$ and eigenvalues $\boldsymbol{\Sigma}_c$ and $\boldsymbol{\Sigma}_c$ respectively such that $\boldsymbol{\Sigma}_c \neq \boldsymbol{\Sigma}_o$ could be the case. Scaling the eigenvectors with $\boldsymbol{\Sigma}_s$ as

$$\boldsymbol{T} = \boldsymbol{T}_u\boldsymbol{\Sigma}_s \tag{2.80}$$

yields

$$\begin{aligned} \boldsymbol{T}^{-1}\boldsymbol{W_c}\boldsymbol{T}^{-*} &= \boldsymbol{\Sigma}_c\boldsymbol{\Sigma}_s^{-2}, \\ \boldsymbol{T}^{*}\boldsymbol{W_o}\boldsymbol{T} &= \boldsymbol{\Sigma}_o\boldsymbol{\Sigma}_s^{2}. \end{aligned} \tag{2.81}$$

For $\hat{\boldsymbol{W}}_c$ and $\hat{\boldsymbol{W}}_o$ to be equal,

$$\boldsymbol{\Sigma}_c\boldsymbol{\Sigma}_s^{-2} = \boldsymbol{\Sigma}_o\boldsymbol{\Sigma}_s^{2}, \tag{2.82}$$

$$\Rightarrow \boldsymbol{\Sigma}_s = \boldsymbol{\Sigma}_c^{1/4}\boldsymbol{\Sigma}_o^{-1/4}. \tag{2.83}$$

The unscaled eigenvectors $\boldsymbol{T}$ that truncate (2.52) can then be obtained as in (2.80).

From the eigendecomposition, the Hankel singular values are sorted along the diagonal of the Hankel matrix in descending order, meaning that the first column in $\boldsymbol{T}$ is the most controllable and observable state direction, the second is the second most, etc. Hence, keeping the first Q columns truncates the transformed system such that it keeps the amount of energy explained by the Q most controllable and observable states.

Next, the goal is to rewrite (2.74) in its truncated form. Let

$$\begin{aligned} \boldsymbol{T} &= \left[\boldsymbol{\Psi} \mid \boldsymbol{T}_t\right], \\ \boldsymbol{S} = \boldsymbol{T}^{-1} &= \left[\frac{\boldsymbol{\Psi}}{\boldsymbol{T}_t}\right] \end{aligned} \tag{2.84}$$

where t denotes the truncated part of the respective matrix, $\boldsymbol{\Psi} \in \mathbb{C}^{P\times Q}$, $\boldsymbol{T}_t \in \mathbb{C}^{P\times(P-Q)}$, $\boldsymbol{\Phi} \in \mathbb{C}^{Q\times P}$ and $\boldsymbol{S}_t \in \mathbb{C}^{(P-Q)\times P}$. (2.74) can therefore be rewritten as

$$\begin{aligned} \left[\frac{\dot{\tilde{\boldsymbol{x}}}(t)}{\dot{\boldsymbol{z}}_t(t)}\right] &= \left[\frac{\boldsymbol{\Phi}^*\boldsymbol{A}\boldsymbol{\Psi} \mid \boldsymbol{\Phi}^*\boldsymbol{A}\boldsymbol{T}_t}{\boldsymbol{S}_t\boldsymbol{A}\boldsymbol{\Psi} \mid \boldsymbol{S}_t\boldsymbol{A}\boldsymbol{T}_t}\right]\left[\frac{\tilde{\boldsymbol{x}}(t)}{\boldsymbol{z}_t(t)}\right] + \left[\frac{\boldsymbol{\Phi}^*\boldsymbol{B}}{\boldsymbol{S}_t\boldsymbol{B}}\right]\boldsymbol{u}(t), \\ \boldsymbol{y}(t) &= \left[\boldsymbol{C}\boldsymbol{\Psi} \mid \boldsymbol{C}\boldsymbol{T}_t\right]\left[\frac{\tilde{\boldsymbol{x}}(t)}{\boldsymbol{z}_t(t)}\right] + \boldsymbol{D}\boldsymbol{u}(t), \end{aligned} \tag{2.85}$$

where $\tilde{\boldsymbol{x}}(t) \in \mathbb{C}^Q$ contains the states to keep and $\boldsymbol{z}_t(t) \in \mathbb{C}^{\mathbb{P}-\mathbb{Q}}$ contains the truncated states. After truncation where the first Q columns of $\boldsymbol{T}$ are kept, i.e., $\boldsymbol{z}_t(t) = \boldsymbol{0}$, (2.74) can be approximated to

$$\begin{aligned}\dot{\tilde{\boldsymbol{x}}}(t) &= \boldsymbol{\Phi}^* \boldsymbol{A} \boldsymbol{\Psi} \tilde{\boldsymbol{x}}(t) + \boldsymbol{\Phi}^* \boldsymbol{B} \boldsymbol{u}(t), \\ \boldsymbol{y}(t) &= \boldsymbol{C} \boldsymbol{\Phi} \tilde{\boldsymbol{x}}(t) + \boldsymbol{D} \boldsymbol{u}(t),\end{aligned} \tag{2.86}$$

such that it preserves the energy corresponding to the Q most controllable and observable states.

2.5.5 Operator norms and error bounds

Technically, it is the *variance* which is to be preserved when truncating a system, but the term *energy* is still commonly used for historical reasons in electrical circuits [10].

It is often more intuitive to truncate a state-space based on the amount of energy in the system that is desirable to preserve rather than the number of states to keep. This can be done by investigating the Hankel singular values. As these are sorted in descending order in the diagonal Hankel matrix, $\boldsymbol{\Sigma}_s$, the cumulative sum of the Q first singular values divided by the sum of all singular values to obtain the fraction of preserved energy.

Using the $\mathcal{H}_\infty$ operator norm defined in (2.55) the balanced realization aims to solve the min-max problem

$$\begin{aligned}&\min_Q \|\boldsymbol{G} - \boldsymbol{G}_Q\|_{\mathcal{H}_\infty} \\ \text{s.t. } &\frac{\operatorname{cumsum}\left(\operatorname{diag}(\Sigma_s)_{1:Q}\right)}{\sum_i \operatorname{diag}\left(\Sigma_s\right)_i} > 1 - \epsilon,\end{aligned} \tag{2.87}$$

where ϵ is the largest tolerable fraction of the variance to be lost, Σ_s is the Hankel matrix and $\boldsymbol{G}_Q$ is the reduced order model.

There exist lower and upper bounds for the error of a chosen truncation:

$$\sigma_{Q+1} \leq \|\boldsymbol{G} - \boldsymbol{G}_Q\|_{\mathcal{H}_\infty} \leq 2 \sum_{i=Q+1}^{P} \sigma_i, \tag{2.88}$$

where σ are the Hankel singular values, i.e., the elements of $\operatorname{diag}(\Sigma_s)$.

2.6 Metric for benchmarking

2.6.1 Envelope-weighted mean absolute error

Envelope-weighted mean absolute error (EMAE) [37] is a dimensionless relative error metric in percentage form defined by

$$\text{EMAE} \equiv \frac{\sum_{i=1}^{N} |y_i - \hat{y}_i|}{\sum_{i=1}^{N} \max\left(y_i, \hat{y}_i\right)} \cdot 100, \tag{2.89}$$

for true values y, predictions $\hat{y}$ and N data points. Unlike other relative error metrics, EMAE is symmetric and scale-independent, which means that the same absolute errors yield the same unique relative error. It also has the property of never diverging at any point.

2.7 Miscellaneous

2.7.1 Nonlinear autoregressive exogenous model

The **nonlinear autoregressive exogenous model (NARX)** [38] is a non-linear autoregressive model including external inputs that describes the evolution of a time series. Given a variable of interest y to model in the future, external or *exogenous* inputs u that help predict y, noise ϵ, and a non-linear function F, NARX is defined accordingly:

$$y_t = F(y_{t-1}, y_{t-2}, y_{t-3}, ..., u_t, u_{t-1}, u_{t-2}, u_{t-3}, ...) + \epsilon_t. \tag{2.90}$$

2.7.2 The curse of dimensionality

The curse of dimensionality [39] is the concept that explains why the sparsity of available data increases as the dimensionality of the data increases.

More formally, given d dimensions and n data points where $X_d^i = (x_1^i, x_2^i, \ldots, x_d^i) \in \mathbb{R}^d \ \forall i = \{1, 2, \ldots, n\}$, and a query point $Q_d = (q_1, q_2, \ldots, q_d) \in \mathbb{R}^d$, the *Minkowski distance (L^p-norm)* between X_d^i and Q_d can be calculated according to

$$\left\| X_d^i - Q_d \right\|_p = \left(\sum_{j=1}^{d} \left| x_j^i - q_j \right|^p \right)^{\frac{1}{p}}, \tag{2.91}$$

which measures the dissimilarity between two points. The largest and smallest distances between Q_d and two other data points can be defined as

$$\begin{aligned} D_{d,\min} &= \min \left\{ \left\| X_d^i - Q_d \right\|_p \mid 1 \leq i \leq n \right\}, \\ D_{d,\max} &= \max \left\{ \left\| X_d^i - Q_d \right\|_p \mid 1 \leq i \leq n \right\}. \end{aligned} \tag{2.92}$$

Under the assumption that the dimensions of X_d^i and Q_d are independent and unique, then

$$\lim_{d \to \infty} \frac{|D_{d,\max} - D_{d,\min}|}{D_{d,\min}} = 0. \tag{2.93}$$

This means that as the number of dimensions grows large, the data points become so sparse that the distance between a query point and any data point converges to the same value. In other words, all data points become dissimilar. This is visualized in Figure 2.6.

In the practice of machine learning, this means that a predictive model will fail to generalize and have difficulty interpolating because there is no similarity in the data. However, *Hughes' phenomenon* [40], visualized if Figure 2.7, states that the accuracy of predictive models initially increases with dimensionality up to a certain point because the data become more separated, and this helps the model to discriminate between clusters of data. After this point, the accuracy decreases for the reason explained above.

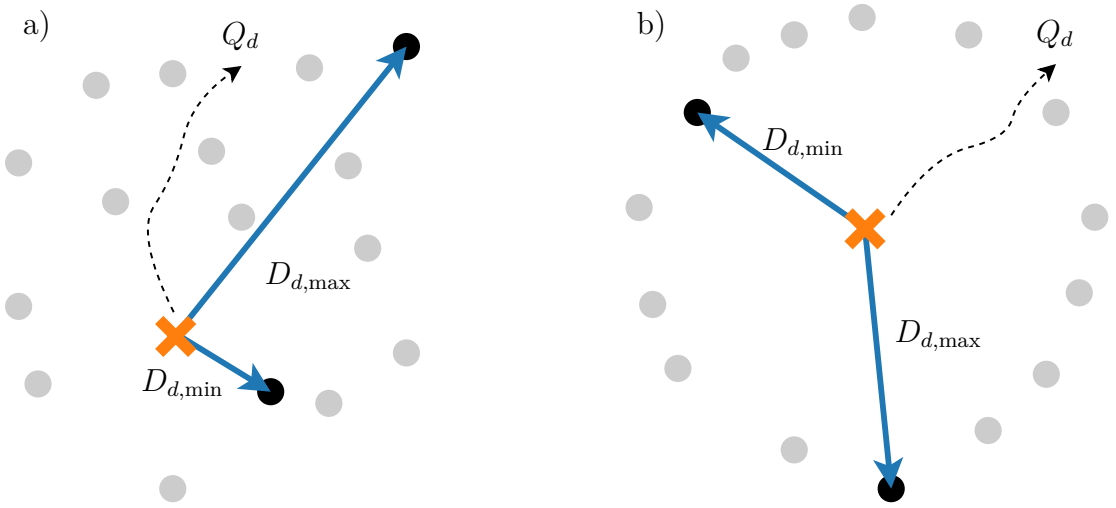

Figure 2.6: The case in 2.6a is when the dimensionality is low, and 2.6b is when the dimensionality is high. In 2.6a, there is a clear minimum and maximum distance between a query point and other data points. In 2.6b, the data points are so sparse and far away from each other that the distance between a query point and any other data point converges to the same value. The dashed lines indicate that the crosses are a query point Q_d.

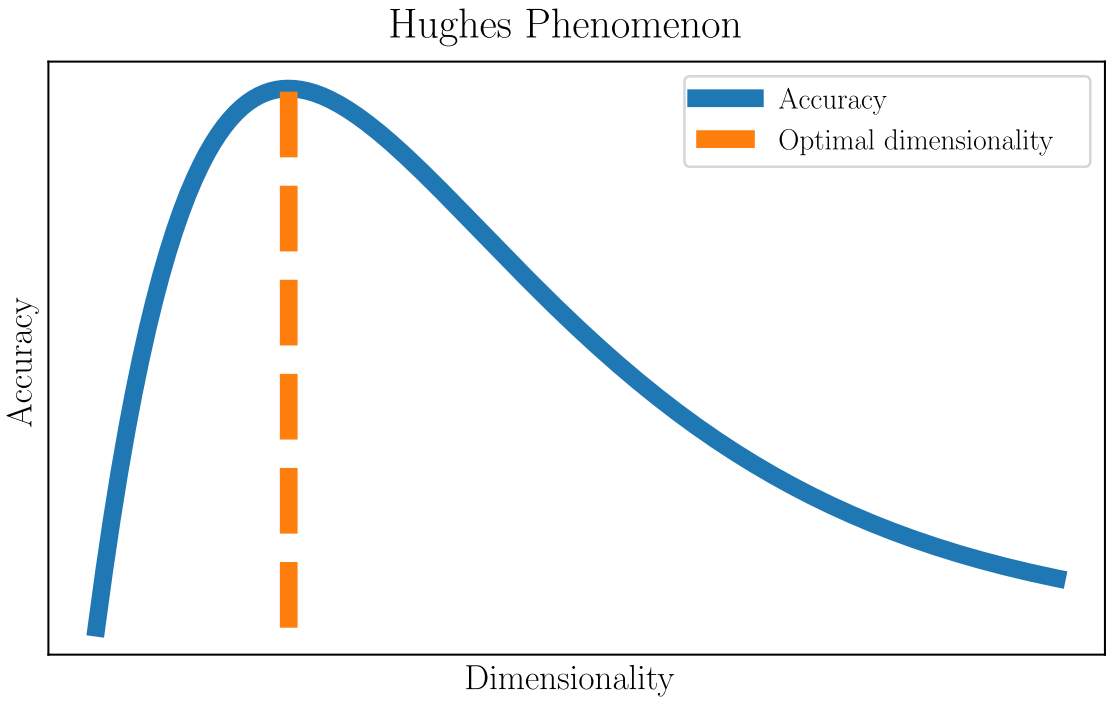

Figure 2.7: Model accuracy vs. dimensionality (number of features), usually has an optimum according to Hughes phenomenon.

3

Methods

In the following sections, the methods that were used to carry out the objectives of the thesis are presented. These include the methods used for preparing the Burö ferry data, selecting features for prediction of fuel consumption, implementing XGBoost and S5 for prediction, as well as carrying out balanced realization on a trained S5.

3.1 Preparing data

Time series data were collected from the Burö passenger ferry operating in the Gothenburg archipelago between December 2022 and August 2023. The routes, or *legs*, that Burö traveled during this period were Kalvsund-Grötö, Grötö-Framnäs, Framnäs-Öckerö, Öckerö-Kalvsund, Kalvsund-Öckerö and Grötö-Öckerö. Figure 3.1 shows how the speed of the ferry varies for some samples from all legs.

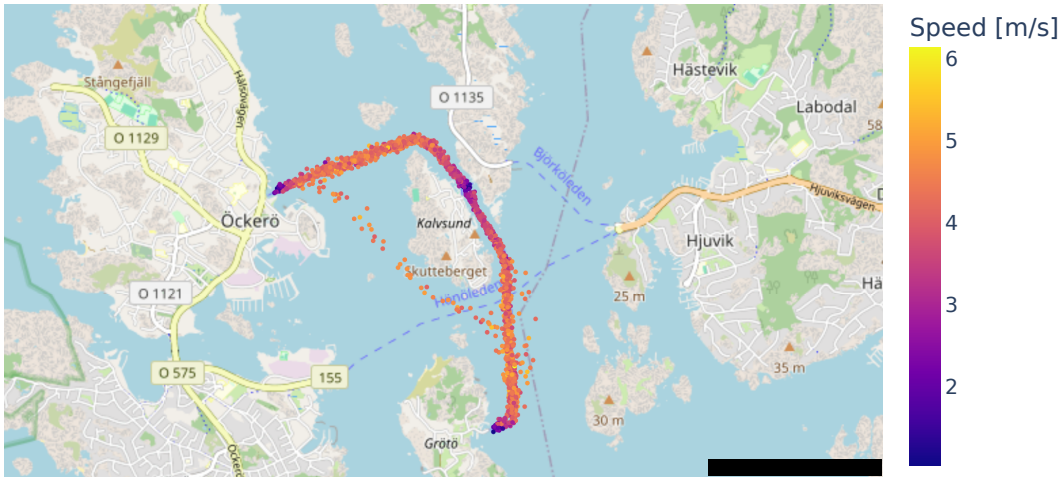

Figure 3.1: 2000 random samples of the vessel speed from all legs from the Burö ferry plotted on a map of the Gothenburg archipelago. Öckerö lies to the west, with Kalvsund and Framnäs positioned centrally and Grötö situated in the south.

The data consisted of engine signals, navigation measurements and forecasted weather

data, all shown in Table 3.1. The weather data were sampled hourly and approximated in large regions, making the weather data constant thereafter.

Table 3.1: The features from the original data set. Fuel per time step [$\mathrm{L\,s^{-1}}$] is the amount of fuel used between two time steps and delta distance [$\mathrm{m\,s^{-1}}$] is the distance travelled between two time steps. Steady-state and sailing state are both categorical features. Steady-state is binary and classifies if the vessel has constant speed or not. Sailing state classifies the vessel's state as either docked, docking, stopped or sailing.

Original features
Fuel per time step
Delta distance
Engine speed
Engine torque
Speed over ground
Acceleration
Longitude
Latitude
Steady-state
Sailing state
Leg name
Forecasted wind speed/direction
Forecasted gust speed/direction
Forecasted wind wave period/height/direction
Forecasted swell wave period/height/direction
Forecasted wave period/height/direction
Forecasted current speed/direction
Forecasted water temperature
Forecasted air temperature
Forecasted air pressure
Timestamp

Figure 3.2 shows how the fuel per time step, the engine speed and the speed over ground change over the course of a journey from Öckerö to Grötö. The data were prepared, which involved cleaning and feature engineering, to obtain high-quality data to help the models learn better.

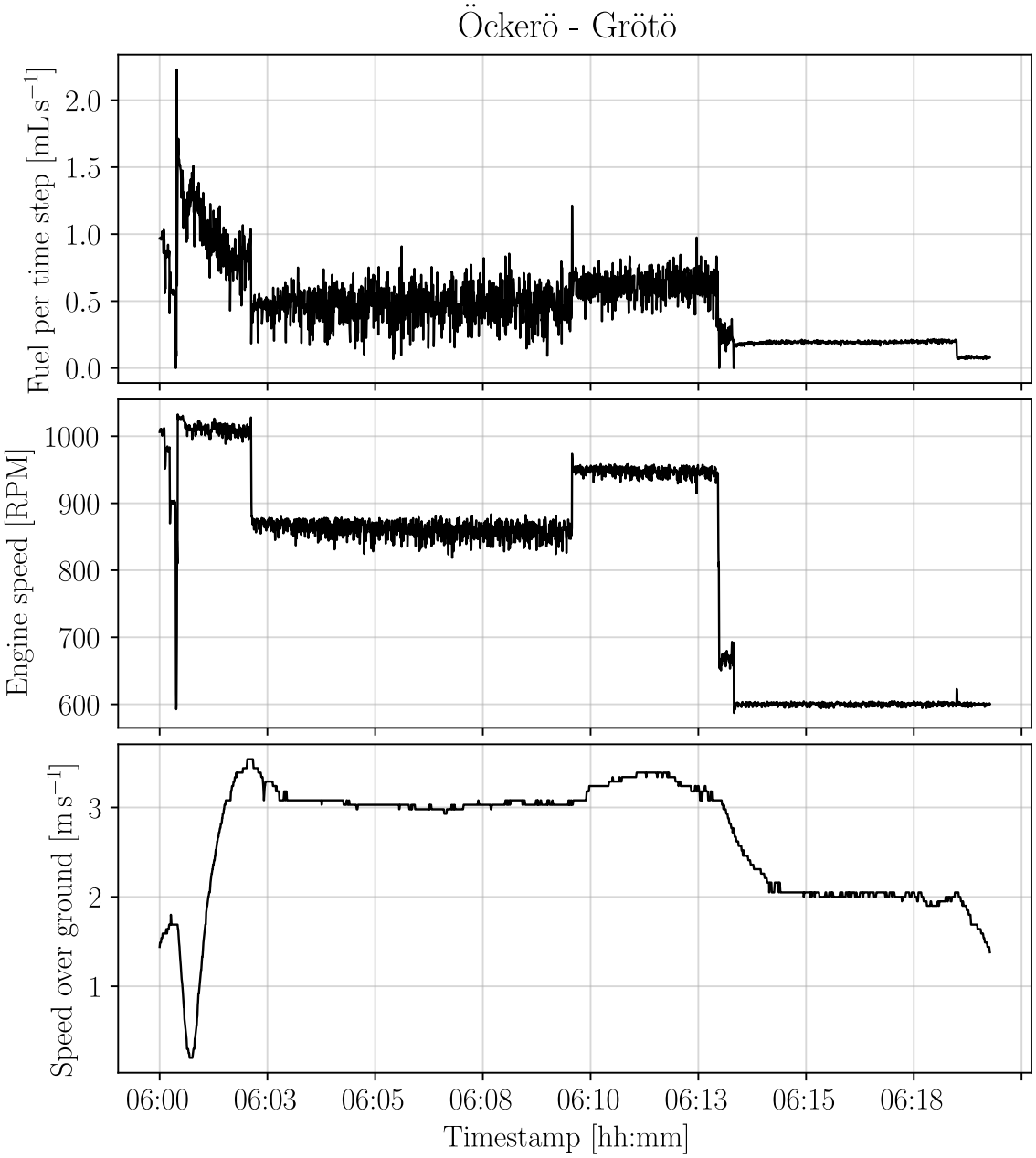

Figure 3.2: Unprocessed time series data of fuel per time step, engine speed and speed over ground of a 20 min long time series. Fuel per time step follows the engine speed closely whereas speed over ground follows it vaguely. There is much stochasticity in the engine signals, both from signal noise and seemingly mechanical and human behaviour. The dip in speed over ground at the start can be explained by a reversal maneuvering. Notice that there is a delayed response in speed over ground from engine speed.

3.1.1 Cleaning data

For cleaning, the raw data were resampled at $3\frac{1}{3}$ Hz and outliers and NaNs were removed. Arguably, the most important filtering procedure was to remove samples

in which the vessel had significant acceleration, specifically at the start and at the end of the journeys. This was important because the vessel's velocity can only determine the *required* power to overcome the drags as in (2.8), not the engine power itself. It is only when the resulting force of the drags and the thrust is zero, i.e. constant velocity, that the engine power and the required power are similar. If the vessel is accelerating, the engine power is greater than the required power and vice versa when decelerating. However, it is quite naive to filter on acceleration to be zero since this rarely happens in practise. Moreover, the models used for prediction had varying abilities for allowing filtering on acceleration due to the nature of their architectures. More on this in Sections 3.3 and 3.4.

Furthermore, states where energy optimization is uninteresting for the captain, such as non-sailing states, reversals and short journeys were also removed. Samples where the vessel was docked and in the process of docking were removed simply by keeping those in the sailing state true. Reversals of the vessel were removed by filtering out data points where the angle difference between heading and course was greater than 120° for more than 6 s. Time series that were shorter than one minute were removed and unique IDs were assigned to the remaining journeys to be able to divide the data randomly into training, validation, and test sets with respect to entire time series. In addition, a moving average with a window size of 10 s was swept over the data to smooth out the noise.

3.1.2 Engineering new features

All engineered features are summarized in Table 3.2. The target feature, fuel per distance [$\mathrm{L\,m^{-1}}$], was obtained by dividing fuel per time step [$\mathrm{L\,s^{-1}}$] by the delta distance [$\mathrm{m\,s^{-1}}$]. In addition, engine power was derived from (2.4).

Table 3.2: Features engineered from the original ones. The original forecasted weather features in Table 3.1 were replaced by their corresponding head and side components in the vessel's reference frame.

Engineered features
Fuel per distance
Engine power
Head/side wind velocity/pseudo-drag
Head/side gust velocity/pseudo-drag
Head/side wind wave velocity/pseudo-drag/power
Head/side swell wave velocity/pseudo-drag/power
Head/side wave velocity/pseudo-drag/power
Head/side current velocity/pseudo-drag
Wind wave energy
Swell wave energy
Wave energy

Figure 3.3 shows how fuel per distance, engine power, and speed on the ground change over a journey from Grötö to Öckerö. More importantly, it specifically explains how the engine power and the required power to overcome all drags relate in real-time.

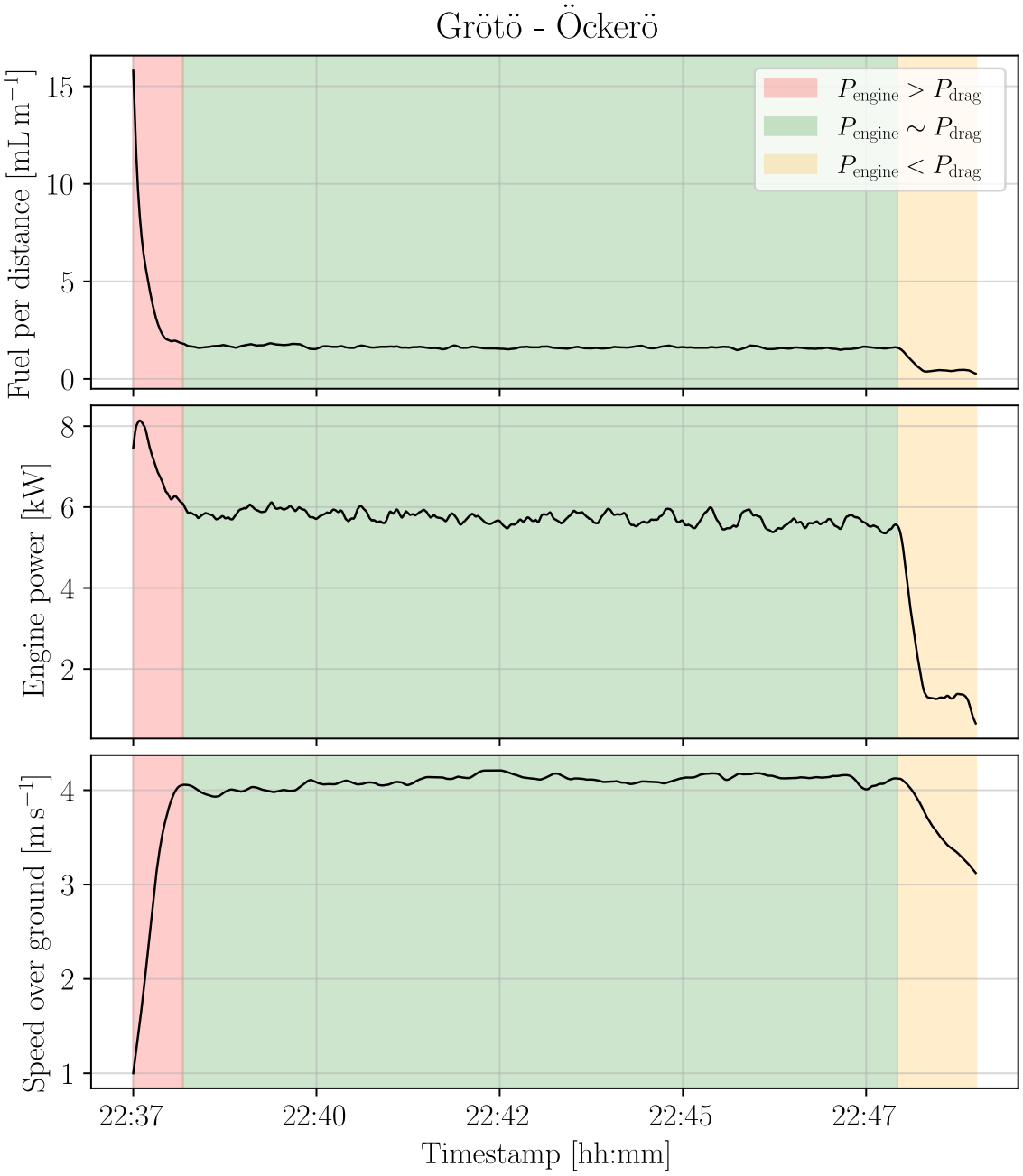

Figure 3.3: An 11 min long processed time series data of fuel per distance, engine power and speed over ground. There is less noise as a result of the smoothening. The speed of the vessel resembles these curves of the engine signals during cruising. Although the data was filtered to extract cruising, data prior to and after this are shown to illustrate mismatch between the engine power P_{engine} and the power required to overcome all drags P_{drag}. When Burö accelerates at the beginning, the engine signals are large for small speeds; $P_{\text{engine}} > P_{\text{drag}}$. As the speed converges to a steady state, the engine signals does so to; $P_{\text{engine}} \sim P_{\text{drag}}$. During these periods, it physically makes sense to approximate P_{engine} with vessel velocity as in (2.8). Toward the end of the journey, when Burö decelerates by almost eliminating P_{engine}, the engine signals are small for large speeds; $P_{\text{engine}} < P_{\text{drag}}$. Notice how different values the engine signals take for the same speeds in red and yellow.

Weather induces drag forces on the vessel, which have quadratic relationships with the respective weather velocities, as stated in Section 2.1.3. Current, wind, and wave velocities were squared to obtain these relationships, which are named *pseudo-drags* here. The wave velocities were determined according to (2.1) and these were also used to calculate the corresponding wave energies of (2.2) and wave powers of (2.3).

As mentioned in Section 1.3, the weather affects the fuel consumption of vessels, but it is really the weather in relation to the frame of the vessel (head and side components) that has external effects on the vessel. Since position and course in the future can be approximately determined through integration, forecasted weather decomposed into the head and side components of the vessel's reference frame can also be determined. Therefore, these features are valid for use in a predictive model when optimizing along a journey.

The head and side components were obtained by taking the cosine and sine of the angle differences between the course and the weather directions, respectively. Positive head components mean that the weather is coming from the front of the vessel (e.g. head wind) and negative from behind (e.g. tail wind). Fuel consumption was assumed to be not dependent on whether the weather came from the right or left side of the vessel. Therefore, the side components were always set to their absolute values to reduce the domains of these features.

3.1.3 Exploring feature dependence

To better understand how the features co-varied with fuel per distance, the correlations between the features were computed. More specifically, Spearman correlations were computed rather than Pearson's r to better capture nonlinear dependencies. However, the original weather features were discarded and only the feature-engineered head and side components of the weather were considered. The wave periods were also discarded because wave velocities were obtained instead.

3.2 Selecting features

Based on the correlations, feature selection was carried out in congruence with the theory of relevance and redundancy explained in Section 2.3. The mRMR algorithm has a fundamental flaw, though; a feature's score is its relevance subtracted with its total redundancy. The subtlety here is that if the most relevant feature gets a significantly large total redundancy, it scores the lowest and will be considered as the worst feature even though it is the strongest predictor.

In an attempt to overcome this issue, a new algorithm was developed. **Norlund's feature selection**, introduced here, similarly leverages relevance and redundancy to select the most appropriate features. The intuition of this algorithm is that features that are too redundant with the most relevant features are removed altogether. A threshold is used as an upper bound on redundancy and another threshold is used as a lower bound on relevance.

Algorithm 3 shows how this works. First, the relationships between all features are computed, e.g. correlation. From these, the features with their relevances are obtained and sorted in descending order. Then, for each relevant feature, features that are too redundant with the relevant feature are removed from relevant features. This makes the set of relevant features dynamically change and ensures that a removed feature is not considered in a later iteration. Lastly, too irrelevant features are removed, and this results in the final set of selected features. This algorithm was used for feature selection for both XGBoost and S5, with an exhaustive grid

search over thresholds for relevance and redundancy to obtain features that yielded better predictions and to account for Hughes' phenomenon. With all correlations precomputed, this grid search runs almost instantly, considering that the process will be bounded by the ranking procedure, which runs in $\mathcal{O}(n \log n)$ time required when calculating the Spearman correlations, where n is the number of data points.

Algorithm 3: Norlund's feature selection

```
Input: data, target, relevance and redundancy thresholds
Compute the relevances with target for all features
Sort features in descending order by relevance
for feature in feature set do
    if feature relevance < relevance threshold then
        Remove feature from feature set
    end
end
for relevant feature in feature set do
    Compute remaining redundancies between relevant feature and feature set
    for other feature in feature set do
        if other feature redundancy > redundancy threshold then
            Remove other feature from feature set
        end
    end
end
Output: feature set
```

3.3 Predicting using XGBoost

XGBoost was implemented in such a way that it mapped a set of features at a time t to the fuel per distance also at time t, that is, there is no time dependence. This property allowed for further filtering of the data on when $P_{\text{engine}} \sim P_{\text{drag}}$, which created gaps in the time series as a consequence. This was done because the cruise data still contained periods where $P_{\text{engine}} \not\sim P_{\text{drag}}$. As shown in Figure 3.2, the engine speed can change abruptly to which speed on the ground has a delayed response. For this reason, additional samples were removed where the steady state was not fulfilled, $|\text{acceleration}| > 0.005\,\text{m}\,\text{s}^{-2}$, $|\text{gradient of engine speed}| > 1\,\text{RPM}^2$ and the moving deviation of engine speed with a window size of 30 s was greater than 100 RPM. In addition, more outliers of fuel per distance and engine speed were removed as well as only keeping vessel speeds in the range $3\,\text{m}\,\text{s}^{-1}$ to $5\,\text{m}\,\text{s}^{-1}$, which are likely inside the range of speeds interesting to optimize over in this case.

Based on this processed data, three approaches to XGBoost were implemented. The first being a baseline model that only maps speed over ground to fuel per distance to set an initial benchmark with a straightforward approach. The other two were based on speed over ground, position, leg name, and weather features obtained from Norlund's feature selection, where the position and leg name were used to help the models recognize where the vessel is and what route it is taking.

The second approach mapped these multiple features directly to fuel per distance, while the third consisted of two XGBoost models; model one mapped speed over ground, weather, position, and leg name to engine power, and model two mapped the same features and predicted engine power to fuel per distance. The reason for this

two-step approach was that speed over ground was significantly better correlated with engine power than with fuel per distance. Since engine power and fuel per distance are almost perfectly collinear, fuel per distance could easily be predicted from engine power.

Specifically, XGBoost was implemented using the module `dmlc XGBoost`[1] on a GPU called *Nvidia GTX 1660 Super*. The models were trained for 1500 boosting rounds using early-stopping of 100 boosting rounds, causing the training to stop if the validation loss does not get any better within any 100 consecutive rounds. The runs were performed with the module's default parameter because these parameters produced similar results despite searching for optimal parameters. However, the mean absolute error (MAE) was used as a loss function, instead of the default setting with mean squared error (MSE), to make the training more robust to potential outliers.

3.4 Implementing the S5

The S5 model was implemented such that it mapped speed over ground, position, and weather from Norlund's feature selection at time t to fuel per distance at the same time t. However, since S5 can be seen as a recurrent neural network, its hidden states at time t partly depend on the hidden states at the previous time step $t-1$, i.e. S5 is sequential in its nature. This property allowed S5 to learn the underlying dynamics of the vessel.

The data were normalized for S5, as this helps the gradients when training neural networks, but were not filtered as with XGBoost, as this would create gaps in the time series. Although S5 is capable of handling irregularly sampled data, the training is more consistent if the data is regularly sampled. Additionally, cutting the time series into shorter ones would prevent S5's inner states from building up properly before it would perform reasonably. In other words, the duration of the warm-up phase of the inner states would be too large a fraction of this shorter time series.

The two-step approach implemented with XGBoost was not performed with S5 as this barely improved the accuracy of XGBoost while increasing the complexity by requiring a two-step approach using separate models. That method was therefore deemed unnecessary to proceed with here.

Regarding the implementation of the architecture, S5 was implemented in the open-source module `s5-pytorch`[2]. Its architecture is the same as shown previously (see Figure 2.4) and the non-linearity as it was implemented in the module is shown in Figure 3.4. The architecture was slightly modified such that there was only one final output from the non-linearity of the last S5-layer. Furthermore, hundreds of trials were conducted to find a well-performing set of hyperparameters. Briefly, the best run had three chained S5-layers, each with a state dimension of 512. The complete set of hyperparameters is described in Table A.1.

[1] `https://xgboost.readthedocs.io/en/stable`

[2] `https://github.com/i404788/s5-pytorch`

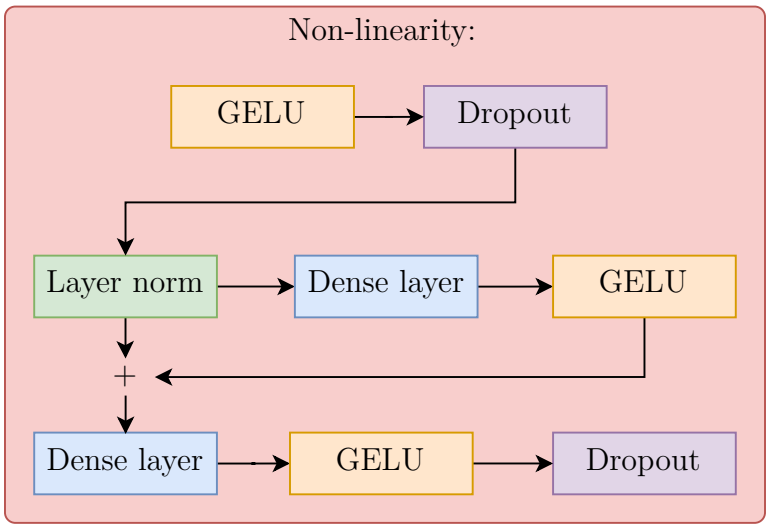

Figure 3.4: The output of the S5-SSM enters the first GELU activation of the non-linearity, and goes through a sequence of layer normalization, dense layers, GELU activation functions and dropouts.

The training was performed on the same GPU as mentioned before using early-stopping. Cosine annealing with warm restarts, a method to oscillate the learning rate throughout the training, was used to leverage exploration and exploitation, as this showed to improve performance. Similarly to XGBoost, MAE was also used here as a loss function.

To be on the safe side in terms of dynamical stability, the real part of the eigenvalues of the state transition matrices were used to regularize, as described in Section 2.5.3.1. More formally, a novel exponential RELU function was used:

$$e^{\mathbf{\Lambda}_i} \max(0, \mathbf{\Lambda}_i) \geq 0, \tag{3.1}$$

which penalizes positive eigenvalues slightly harder than the conventional RELU.

3.4.1 Compressing state-spaces using balanced realization

Balanced realization was performed on the S5 to compress the model according to Sections 2.5.1.4 and 2.5.4. 95% of the energy was preserved in each state-space of the S5-layers. Differences in prediction error, computation time, and transfer functions were analyzed before and after truncation.

The compressed and balanced system matrices $\hat{\boldsymbol{A}} \in \mathbb{C}^{Q\times Q}$, $\hat{\boldsymbol{B}} \in \mathbb{C}^{Q\times M}$, $\hat{\boldsymbol{C}} \in \mathbb{C}^{N\times Q}$, and the unmodified $\boldsymbol{D} \in \mathbb{R}^{N\times M}$ are recalled from (2.86) as

$$\begin{aligned} \dot{\tilde{\boldsymbol{x}}}(t) &= \overbrace{\boldsymbol{\Phi}^* \mathbf{\Lambda} \boldsymbol{\Psi}}^{\hat{\boldsymbol{A}}} \tilde{\boldsymbol{x}}(t) + \overbrace{\boldsymbol{\Phi}^* \boldsymbol{B}}^{\hat{\boldsymbol{B}}} \boldsymbol{u}(t), \\ \boldsymbol{y}(t) &= \underbrace{\boldsymbol{C}\boldsymbol{\Phi}}_{\hat{\boldsymbol{C}}} \tilde{\boldsymbol{x}}(t) + \boldsymbol{D}\boldsymbol{u}(t). \end{aligned} \tag{2.86}$$

However, unlike $\mathbf{\Lambda}$, this new balanced transition matrix $\tilde{\boldsymbol{A}}$ is no longer diagonal, as required by the parallel associative scans which the S5 architecture uses to efficiently

compute recurrences. However, the system can simply, yet again, be diagonalized by the spectral decomposition of $\hat{\boldsymbol{A}}$ as described in Section 2.5.1.2, yielding

$$\begin{aligned}\dot{\tilde{\boldsymbol{x}}}(t) &= \overbrace{\boldsymbol{V}^{-1}\hat{\boldsymbol{A}}\boldsymbol{V}}^{\hat{\boldsymbol{A}}_{\boldsymbol{V}}=\hat{\boldsymbol{\Lambda}}}\tilde{\boldsymbol{x}}(t) + \overbrace{\boldsymbol{V}^{-1}\hat{\boldsymbol{B}}}^{\hat{\boldsymbol{B}}_{\boldsymbol{V}}}\boldsymbol{u}(t),\\ \boldsymbol{y}(t) &= \underbrace{\hat{\boldsymbol{C}}\boldsymbol{V}}_{\hat{\boldsymbol{C}}_{\boldsymbol{V}}}\tilde{\boldsymbol{x}}(t) + \boldsymbol{D}\boldsymbol{u}(t).\end{aligned} \tag{3.2}$$

Now, the issue here is the time discretization step size $\boldsymbol{\Delta} \in \mathbb{R}^P$, which also needs to be truncated to the dimension $\mathbb{R}^Q$. There does not seem to exist any established theory on this. However, a naive but intuitive approach is to perform the transformation and truncation to $\boldsymbol{\Delta}$ exactly as was done to the state transition matrix:

$$\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}} = \underbrace{\boldsymbol{V}^{-1}\overbrace{\boldsymbol{\Phi}^*\boldsymbol{\Delta} \odot \boldsymbol{I}\boldsymbol{\Psi}}^{\hat{\boldsymbol{\Delta}}}\boldsymbol{V}}_{\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}}}. \tag{3.3}$$

Unfortunately, this results in a complex and nondiagonal matrix $\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}} \in \mathbb{C}^{Q\times Q}$, as opposed to the required $\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}} \in \mathbb{R}^Q$.

Now, recalling the equation for zero-order hold discretization of the state transition matrix as

$$\overline{\boldsymbol{A}} = e^{\boldsymbol{A}\Delta} \tag{2.50}$$

and looking at the exponent while inserting the new balanced and truncated matrices, it simplifies to

$$\hat{\boldsymbol{A}}_{\boldsymbol{V}}\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}} = \boldsymbol{V}^{-1}\boldsymbol{\Phi}^*\boldsymbol{\Lambda}\boldsymbol{\Psi}\cancel{\boldsymbol{V}\boldsymbol{V}^{-1}}\boldsymbol{\Phi}^*\boldsymbol{\Delta} \odot \boldsymbol{I}\boldsymbol{\Psi}\boldsymbol{V} = \boldsymbol{V}^{-1}\boldsymbol{\Phi}^*\boldsymbol{\Lambda}\boldsymbol{\Delta} \odot \boldsymbol{I}\boldsymbol{\Psi}\boldsymbol{V}, \tag{3.4}$$

which however, is not diagonal, as only the diagonalizing transformation of $\hat{\boldsymbol{A}}$ was applied. Instead, if the diagonalizing transformation of $\hat{\boldsymbol{A}}\hat{\boldsymbol{\Delta}}$ is applied directly, a discretized, complex, and diagonal state transition matrix $\overline{\boldsymbol{A}}$ would be obtained, just as required. However, this did not provide satisfactory results. For this reason, alternative, more or less theoretically sound transformations were tried.

Other options for forcing $\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}} \in \mathbb{R}^Q$, include using the singular values $\boldsymbol{\Sigma}$ of $\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}}$, simply taking the magnitude of the matrix and extracting the diagonal as $\operatorname{diag}\left|\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}}\right|$, and taking the magnitude of the eigenvalues as $\left|\operatorname{eig}\left(\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}}\right)\right|$. All of these approaches are compared and discussed further in Section 4.3.

3.5 Benchmarking

Based on EMAE, presented in Section 2.6.1, a new relative error metric is proposed: **envelope-weighted median absolute error (EMEDAE)** for time series benchmarking. Here, the median is used instead of the expected value:

$$\text{EMEDAE} \equiv \frac{\text{med}(|\boldsymbol{y} - \hat{\boldsymbol{y}}|)}{\text{med}(\max(\boldsymbol{y}, \hat{\boldsymbol{y}}))} \cdot 100, \tag{3.5}$$

where $\boldsymbol{y}$ and $\hat{\boldsymbol{y}} \in \mathbb{R}^N$ are the labels and predictions for N data points, respectively. This is motivated by the fact that the minimum absolute deviation is always the median of the target distribution [41], which a model learns to approximate during training if the mean absolute error is used as a loss function. It therefore makes sense to evaluate the model with a relative error that is calculated on the basis of the median to test its performance. This was the case for both XGBoost and S5.

Moreover, performance was not calculated for the first minute of the S5 predictions to let the inner states build up and stabilize, as would be the case if used in production. However, this was not the case for XGBoost as it was independent of time.

To express the error bounds (2.88) for the $\mathcal{H}_\infty$-norms of the truncation in a relative sense, the equation was min-max normalized such that

$$0 = \frac{\sigma_{Q+1} - \sigma_{Q+1}}{-\sigma_{Q+1} + 2\sum_{i=Q+1}^{P} \sigma_i} \leq \frac{\|\boldsymbol{G} - \boldsymbol{G}_Q\|_{\mathcal{H}_\infty} - \sigma_{Q+1}}{-\sigma_{Q+1} + 2\sum_{i=Q+1}^{P} \sigma_i} \leq \frac{-\sigma_{Q+1} + 2\sum_{i=Q+1}^{P} \sigma_i}{-\sigma_{Q+1} + 2\sum_{i=Q+1}^{P} \sigma_i} = 1. \tag{3.6}$$

4

Results

The results that are intended to fulfill the objectives of the thesis are provided in this chapter. The correlation between weather and fuel consumption of the Burö ferry, the performance of XGBoost and S5, and the corresponding prediction features that produce this are presented. The results required to perform and analyze balanced realization of S5 are also presented.

4.1 Correlations with fuel consumption

Table 4.1 shows Spearman correlations of the engine power and navigation signals with fuel per distance. These correlations are based on the filtering techniques used for XGBoost because these data are better processed for where $P_{\text{engine}} \sim P_{\text{drag}}$, as mentioned in Section 3.3, which therefore provides better information about the dependency between weather and fuel consumption.

Table 4.1: Spearman correlations between fuel per distance and the engine and navigation features, sorted in descending order by $|r_s|$ and based on the data with the filtering techniques used for XGBoost.

Fuel per distance	
Feature	r_s
Engine power	0.97
Speed over ground	0.44
Latitude	-0.06
Longitude	0.00

Engine power is almost perfectly collinear with fuel per distance, as expected. Without *any* filtering, speed over ground correlated with fuel per distance with 0.27. The table shows that applying the filtering techniques gives a 63 % better correlation, showing that preprocessing of the data was effective. Longitude and latitude barely correlate with fuel per distance, even though it showed that the use of these features improved the performance of the models.

Spearman correlations between fuel per distance and the weather features are presented in Table 4.2, also based on the filtering techniques for XGBoost for the same reason above. The highest correlations are the head components of the wind-related features (wind, gust, wind wave) with correlations 0.28 to 0.30.

A noticeable trend is that the head components of the weather correlate the most with fuel per distance. This makes physical sense since it is mainly the head components that induce the drags that the propeller of the vessel has to overcome. The

Table 4.2: Spearman correlations between fuel per distance and the weather features, sorted in descending order by $|r_s|$ and based on the data with the filtering techniques used for XGBoost.

Fuel per distance	
Feature	r_s
Head wind velocity/pseudo-drag	0.30
Head gust velocity/pseudo-drag	0.30
Head wind wave velocity/pseudo-drag/power	0.28
Head swell wave velocity/pseudo-drag	0.23
Head wave velocity/pseudo-drag/power	0.22
Head swell wave power	0.21
Head current velocity/pseudo-drag	0.20
Wind wave height/energy	0.18
Wave height/energy	0.18
Side wind wave power	0.16
Side wave power	0.14
Air pressure	-0.14
Side wind wave velocity/pseudo-drag	0.10
Side wind velocity/pseudo-drag	0.07
Side gust velocity/pseudo-drag	0.06
Water temperature	0.06
Swell wave height/energy	0.06
Air temperature	0.06
Side swell wave power	0.04
Side wave velocity/pseudo-drag	-0.04
Side current velocity/pseudo-drag	0.03
Side swell wave velocity/pseudo-drag	0.01

head components of the wind wave features have the most significant correlations (0.28) among the wave features, followed by the head swell wave/wave/current features with correlations of 0.20-0.23. The side components have weak correlations and do not show any clear patterns. Air pressure and air/water temperature also correlate weakly with fuel per distance.

The correlation between weather and fuel per distance is not significant for this case of the Burö ferry. The reason for this could be that the surrounding islands in the archipelago protect the ferry from harsher weather. Weather may have greater correlations with fuel per distance in places where there are harsher weather conditions.

With regard to weak correlation, it is important to emphasize that correlation is a subset of dependence and does not reveal dependence in its entirety. Thus, one cannot draw a full conclusion about the dependence on fuel per distance solely on correlation, but it may give an indication.

4.2 Prediction of fuel consumption

4.2.1 Selected features

Table 4.3 shows the features used that yielded the best results for XGBoost and their Spearman correlations with fuel per distance and engine power respectively, including the weather features obtained from Norlund's feature selection.

Table 4.3: Spearman correlations of the features with fuel per distance (left) and engine power (right) that yielded the best results for XGBoost, including the weather features obtained from Norlund's feature selection. They are sorted in descending order by $|r_s|$ and are based on the data with the filtering techniques for XGBoost.

Fuel per distance

Feature	r_s
Engine power	0.97
Speed over ground	0.44
Head gust velocity	0.30
Head swell drag	0.23
Head current drag	0.20
Wind wave height	0.18
Air pressure	-0.14
Side wind drag	0.07
Water temperature	0.06
Latitude	-0.06
Longitude	0.00

(a) Relevance threshold: 0.06, redundancy threshold: 0.64.

Engine power

Feature	r_s
Speed over ground	0.60
Head gust velocity	0.19
Wind wave height	0.17
Head swell drag	0.15
Air pressure	-0.14
Head current drag	0.12
Water temperature	0.08
Side wind drag	0.07
Longitude	-0.04
Latitude	-0.03

(b) Relevance threshold: 0.07, redundancy threshold: 0.64.

The weather features are the same in both cases, though with different correlations. This is not surprising since engine power and fuel per distance are almost perfectly collinear and will therefore produce similar features from Norlund's feature selection. Ironically, the correlation between speed over ground and engine power is 36 % better than between speed over ground and fuel per distance. This was the motivation to implement the two-step approach for XGBoost.

Similarly, Table 4.4 shows the features used that yielded the best results for S5 and their Spearman correlations with fuel per distance, also including the weather features obtained from the Norlund feature selection.

For all three selected feature sets, by only looking at the correlations it looks like some features are not important for the prediction of fuel per distance. However, for both cases of XGBoost and S5, it showed that the use of neither too few nor too many features produced better predictions, despite some having weak correlations. This can again be partly explained by what was said before about correlation being a subset of dependence. However, it can also be explained by Hughes' phenomenon explained in Section 2.7.2; up to a certain point, adding more predictive features helps predictive models discriminate between data clusters.

Table 4.4: Spearman correlations of all features with fuel per distance that yielded the best results for S5. These include the weather features obtained from Norlund's feature selection using a relevance threshold of 0.07 and a redundancy threshold of 0.7. They are sorted in descending order by $|r_s|$ and are based on data with the basic filtering techniques.

Fuel per distance	
Feature	r_s
Speed over ground	0.44
Head wind velocity	0.18
Head swell velocity	0.13
Head wave power	0.13
Head current velocity	0.12
Side wave velocity	-0.11
Water temperature	0.08
Wind wave height	0.07
Latitude	-0.06
Longitude	0.02

4.2.2 Performance

Table 4.5 shows the performance results obtained from the implementations of XGBoost and S5. The two-step approach for XGBoost performed the best, but was only 0.19 %pt. (2 %) better than predicting fuel per distance directly. However, only using speed performed 7.39 %pt. (88 %) worse than the two-step approach. Compared to the S5 with full states, the S5 performed 5.46 %pt. (65 %) worse than the two-step approach, but 1.93 %pt. (12 %) better than when XGBoost only uses speed.

Table 4.5: Performance of all models using EMEDAE for the test sets.

Model	EMEDAE [%]
XGBoost - speed only	15.81
XGBoost - speed, weather, position and leg	8.61
XGBoost - predicted power, speed, weather, position and leg	**8.42**
S5 - full state	13.88
S5 - truncated state, diag $\left\lvert\hat{\mathbf{\Delta}}_{\boldsymbol{V}}\right\rvert$	14.95
S5 - truncated state, $\sigma(\hat{\mathbf{\Delta}}_{\boldsymbol{V}})$	16.32
S5 - truncated state, $\left\lvert\text{eig}\left(\hat{\mathbf{\Delta}}_{\boldsymbol{V}}\right)\right\rvert$	19.30

The best performing truncation method was the approach using diag $\left|\hat{\mathbf{\Delta}}_{\boldsymbol{V}}\right|$ and performed 1.07 %pt. (8 %) worse than without any truncation. The worst method was $\left|\text{eig}\left(\hat{\mathbf{\Delta}}_{\boldsymbol{V}}\right)\right|$, which performed 5.42 %pt. (39 %) worse than with full states. The number of training rounds and test losses (MAE) corresponding to these results are presented in Table A.2.

In terms of inference speed, the truncated model averaged 26.95 batches per second, while the full model only managed 16.12 batches per second. This is a 67 % increase in prediction speed.

Figure 4.1 shows the models' predictions for one and the same time series of the test sets. There are gaps in the time series for XGBoost since XGBoost allowed for further filtering of the data, whereas it is whole for the case for S5. The results of these filtering techniques for XGBoost are clearly visualized here; the predictions of XGBoost are only done for samples where the speed is in steady state, since the fuel per distance is in steady state.

Furthermore, the S5 appears to be able to capture the dynamics of fuel per distance as it follows its curve throughout most of the journey. However, it fails to do this at the beginning of the time series, but this can be explained by the fact that the inner states build up during the first minute. As mentioned in Section 3.5, performance was not measured for the first minute of any time series for S5. The oscillatory behavior of the truncated S5 during this minute could be explained by the fact that the imaginary part of the eigenvalues of the transition matrices have gotten larger after truncation.

It can be noted that there is a clear trade-off between the models, namely that XGBoost allows for better data preparation but is naive in the sense that it does not account for any dynamics, whereas S5 does the opposite. Not done here, but it would be interesting to measure the performance of these models for the intersection of the predicted data points between XGBoost and S5.

4.3 Balanced realization of the S5

The trained state transition matrices $\mathbf{\Lambda}$ of the S5 revealed that they were indeed dynamically stable, allowing truncation over all states of the three state-space layers. Figure 4.2 illustrates the truncation process. More specifically, it shows the relative accumulative Hankel singular values and the normalized Hankel singular values against the Hankel singular value indices, respectively. Smaller indices correspond to larger energy. It tells how many of the largest singular values are needed to preserve 95 % of the energy of each state-space of the S5-layers. The figure implies that the amounts of states to keep are 387, 361, and 73 in the first, second, and third state-space layers, respectively.

4.3.1 Transfer functions

In Figure 4.3 a small excerpt of the S5 transfer functions before and after truncation is visualized, i.e., the response between the inputs and outputs of the state-space layers. However, the signals are physically uninterpretable due to the model's deep Wiener structure. It is therefore more interesting to analyze the shapes of the transfer functions and how well they match for the case of the full and truncated models. The spikes can be interpreted to show that the outputs of the state-space layers are very sensitive to the inputs.

In other words, the state-space layers by themselves are not robust, i.e. a minor change in the frequency of the input signal will produce a majorly different output response. Possible explanations for why the models still perform in a reasonable manner are that, firstly, the spikes could be canceled in later layers, since the state-space layers are chained. Secondly, the non-linearities could attenuate the outputs of the state-space layers.

4.3.2 Expressivity vs. overfitting

One theory is that S5 is possibly more expressive due to the spiky shape of the transfer functions, similar to the concept of *expressive power* [42], which is a highly desired property for artificial neural networks. However, the caveat is that expressivity often goes hand in hand with overfitting [43].

Furthermore, a clear advantage of the balanced realization method for model reduction was that it was able to improve the inference speed by 67 % while only decreasing prediction accuracy by 8 %. However, a more subtle advantage is the likely increased *robustness against overfitting*. Reducing the number of parameters quadratically by each state removed will likely have a large impact here: aiming for a minimal realization effectively makes the model more interpretable.

Yet another benefit when balancing the observability and controllability Gramians of the layers is possibly *normalizing the parameters* of the model. This would prevent vanishing or exploding gradients and results in a smoother loss landscape if one wishes to retrain the model later.

Additionally, regarding Figure 4.3 it can be seen that the expressivity of the truncated model was preserved. If allowed to speculate, in the event of a slightly overfitted model, *balanced realization could potentially even improve the performance on unseen data.*

4.3.3 $\mathcal{H}_\infty$-stability

The $\mathcal{H}_\infty$-norms for each state-space layer and the $\mathcal{H}_\infty$-error between the full and truncated transfer functions can be seen in Table 4.6. Only the best-performing transformation method for the step sizes, i.e., diag $\left|\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}}\right|$ was considered. Here, the normalized $\mathcal{H}_\infty$-errors are relative to the bounds, as calculated in (3.6). This metric shows that the truncation is quite successful; the error is only at most in the second percentile (1.54 %) within the lower and upper error bounds. Furthermore, these results show that nominal stabilities of the state-space layers of both the full and truncated models are not fulfilled, since the $\mathcal{H}_\infty$-norms are all greater than 1, as implied by the small-gain theorem (2.54).

Table 4.6: $\mathcal{H}_\infty$-norms and errors for each state-space layer of the S5.

Layer #	$\|\boldsymbol{G}\|_{\mathcal{H}_\infty}$	$\|\boldsymbol{G}_Q\|_{\mathcal{H}_\infty}$	$\|\boldsymbol{G}-\boldsymbol{G}_Q\|_{\mathcal{H}_\infty}$	Normalized $\|\boldsymbol{G}-\boldsymbol{G}_Q\|_{\mathcal{H}_\infty}$ [%]
1	41.55	41.48	2.55	0.92
2	154.65	154.72	1.68	0.63
3	11.25	11.25	0.51	1.54

In Figure 4.4 the largest singular value, $\overline{\sigma}(j\omega)$, as a function of frequency is shown. Intuitively, the maximum value here is, by definition, the $\mathcal{H}_\infty$-norm. By analyzing the maximum singular value of the systems at different frequencies, the MIMO response can be condensed from $N \times M$ such responses into a single one. Moreover, also here the jagged behavior for higher frequencies is clearly seen.

4.3.4 Transformations of the step sizes

As mentioned in Section 3.4.1, there seemed to be no established theory on handling already decided step sizes during a balanced realization, prompting us to try

different methods for projecting the dense and complex matrices down to a real and diagonal one, which can then be parameterized as a simple vector. In Figure 4.5 the distributions of the learned, truncated and then transformed step sizes $\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}}$ are shown. It can be seen that all the transformations result in very similar distributions.

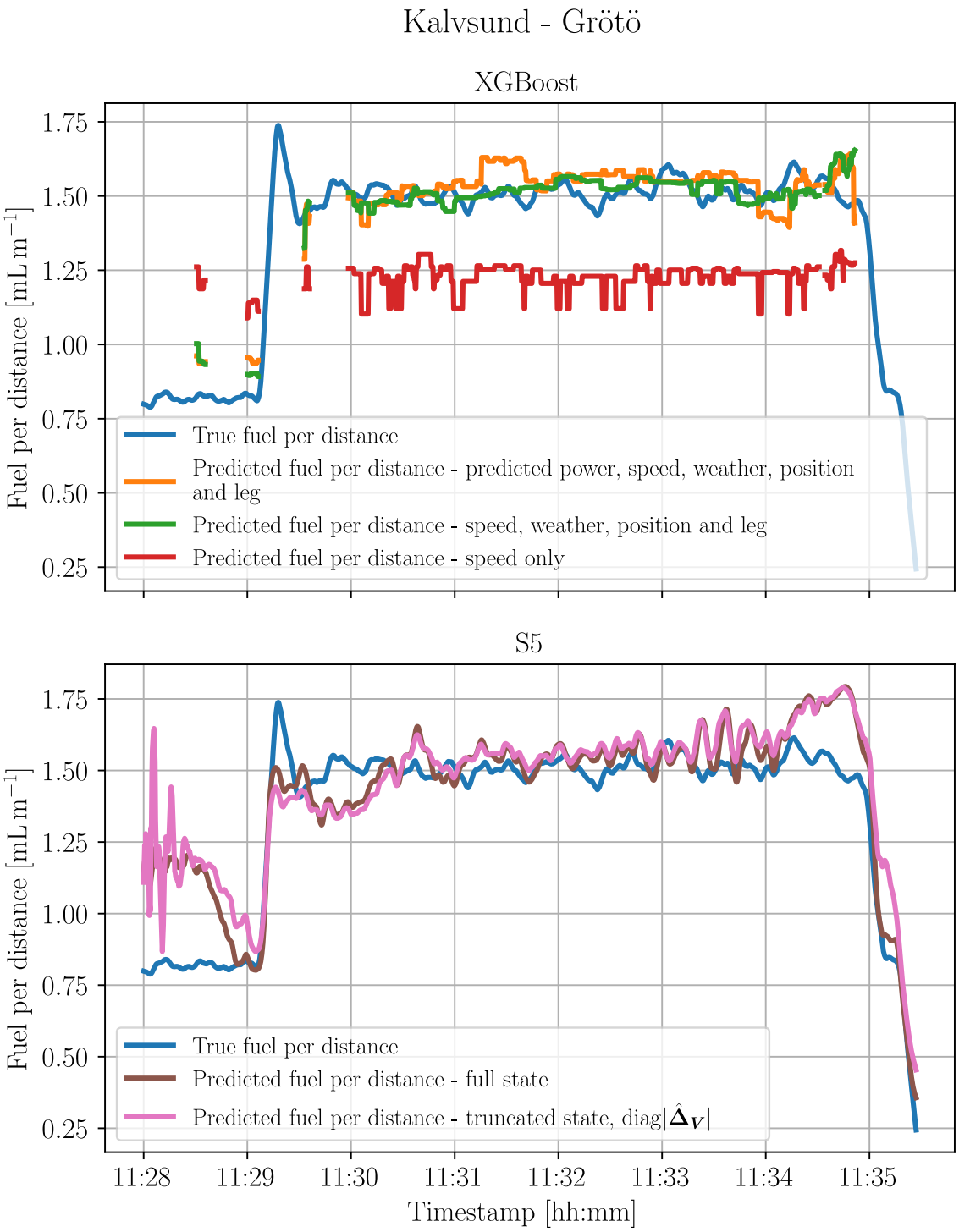

Figure 4.1: Predictions of a time series from the test sets by XGBoost (top) and S5 (bottom) models. Regarding truncation, only the best method: diag $\left|\hat{\mathbf{\Delta}}_V\right|$ is shown here to avoid overflow as they all showed similar trajectories.

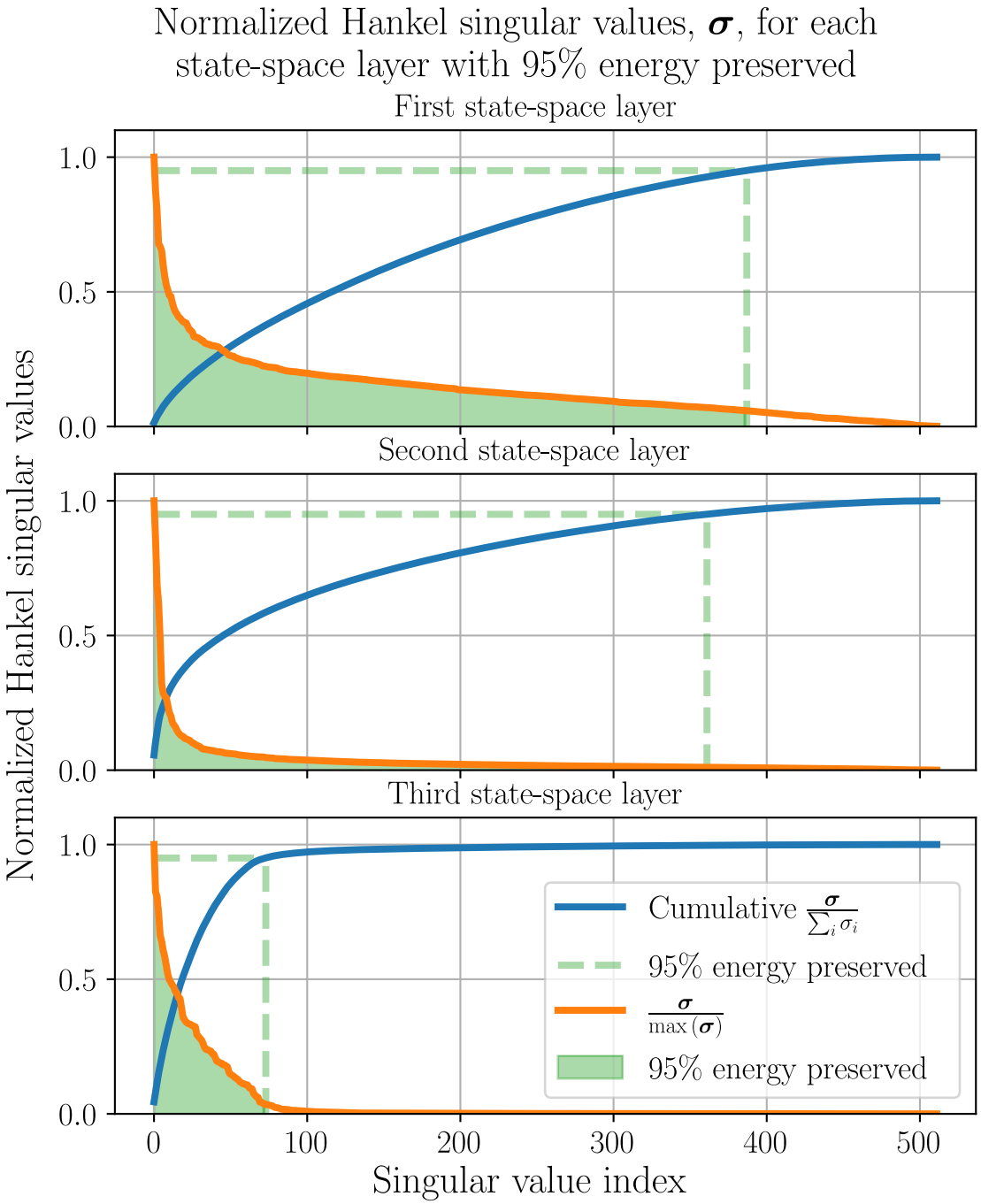

Figure 4.2: The normalized Hankel singular values against the singular value indices for each state-space layer of the trained S5, leveraged to perform balanced realization. The blue line shows the relative accumulative energy of the singular values, the orange line simply shows the normalized singular values, and everything in green indicate conservation of 95 % of the energy for each state-space.

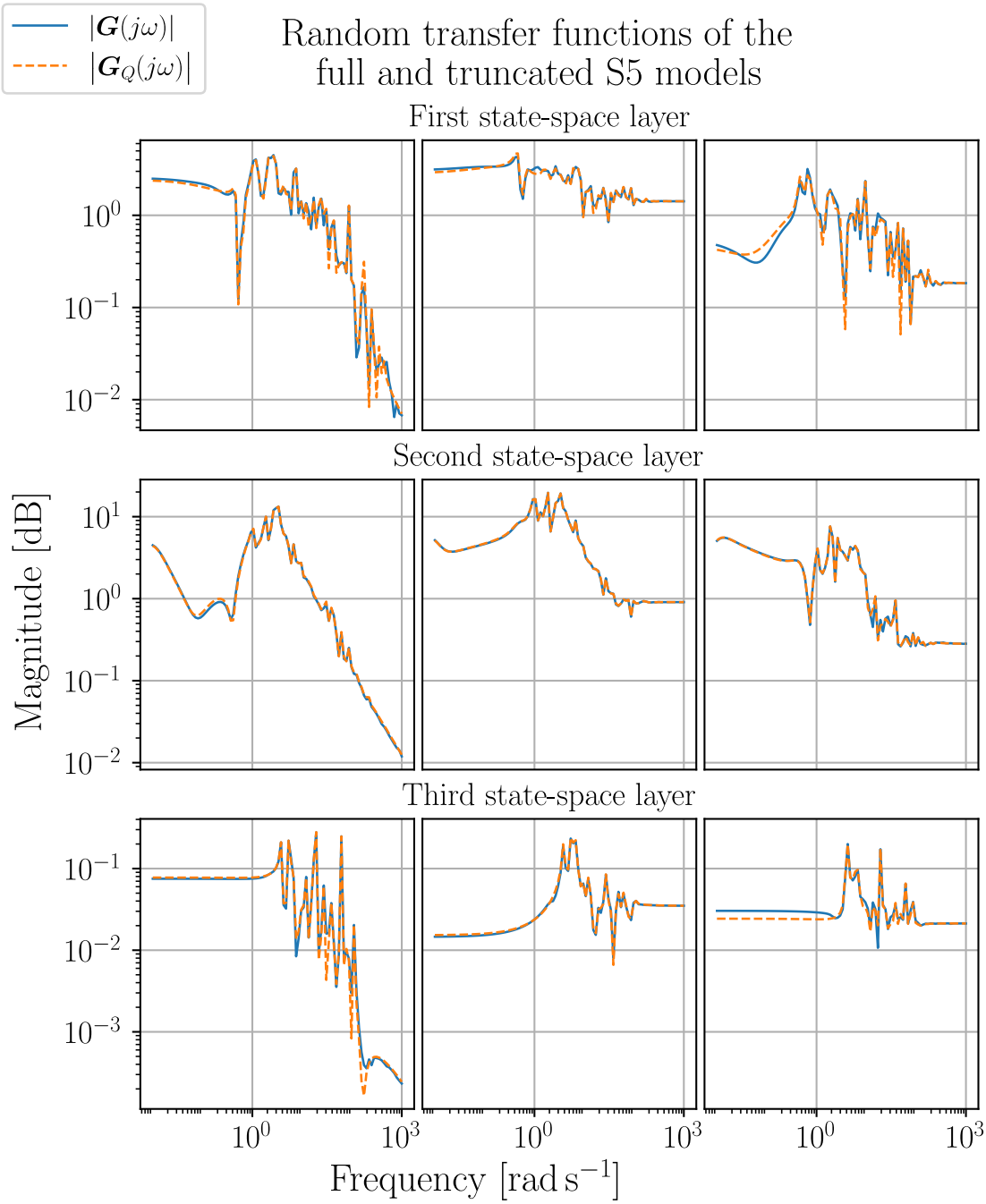

Figure 4.3: A selection of random pairs of input-output responses for the state-space layers.

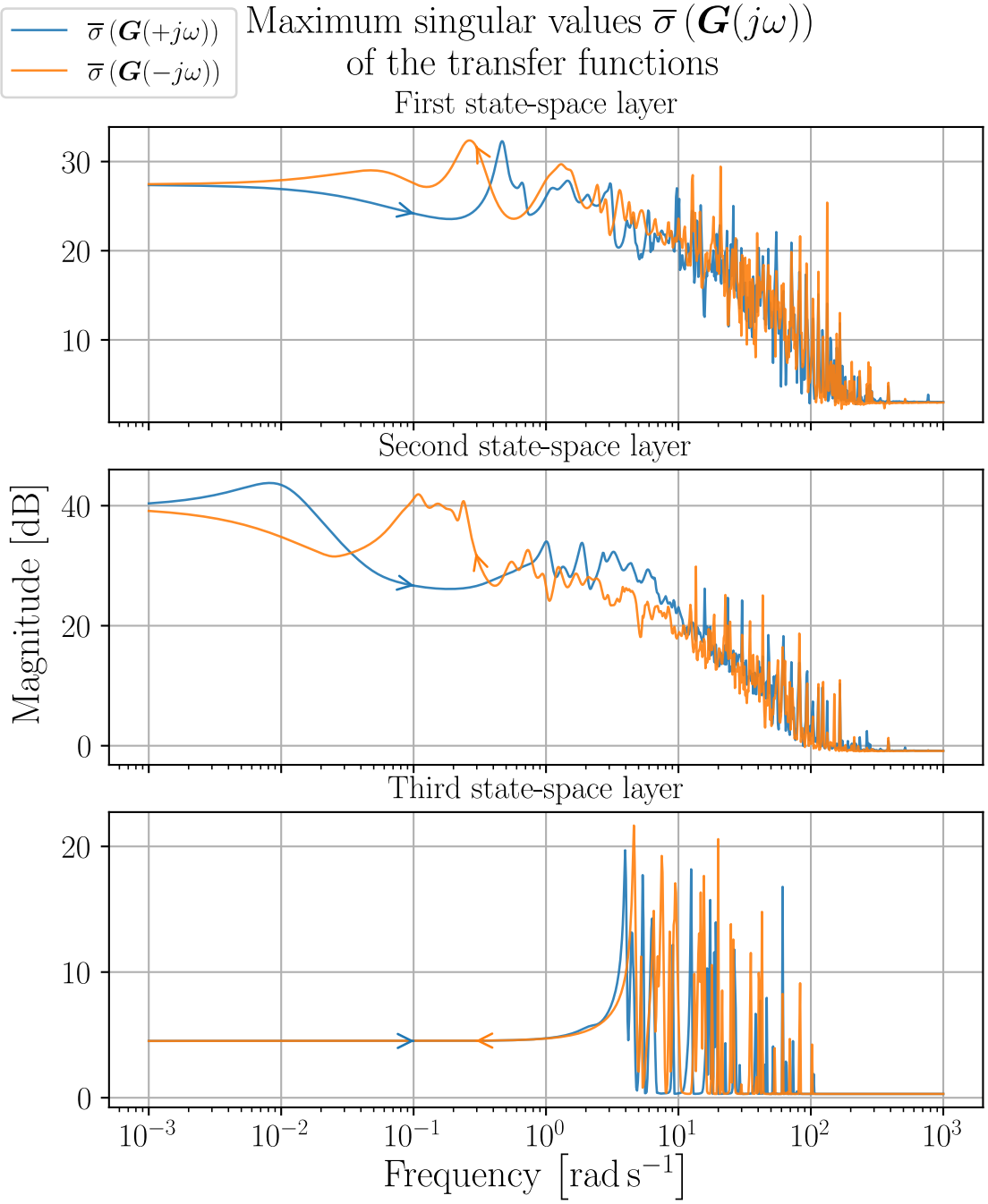

Figure 4.4: The largest singular value, $\overline{\sigma}(\boldsymbol{G}(j\omega))$, as a function of frequency. The logarithmic scale on the ω-axis requires the negative frequency to be plotted separately. Notice the arrows for the direction of increasing frequencies. However, negative frequencies are only a mathematical construct and are not physically interpretable. They are shown only for completeness. Also, only the full state model is plotted here because the response for the truncated model was identical and thus redundant to show.

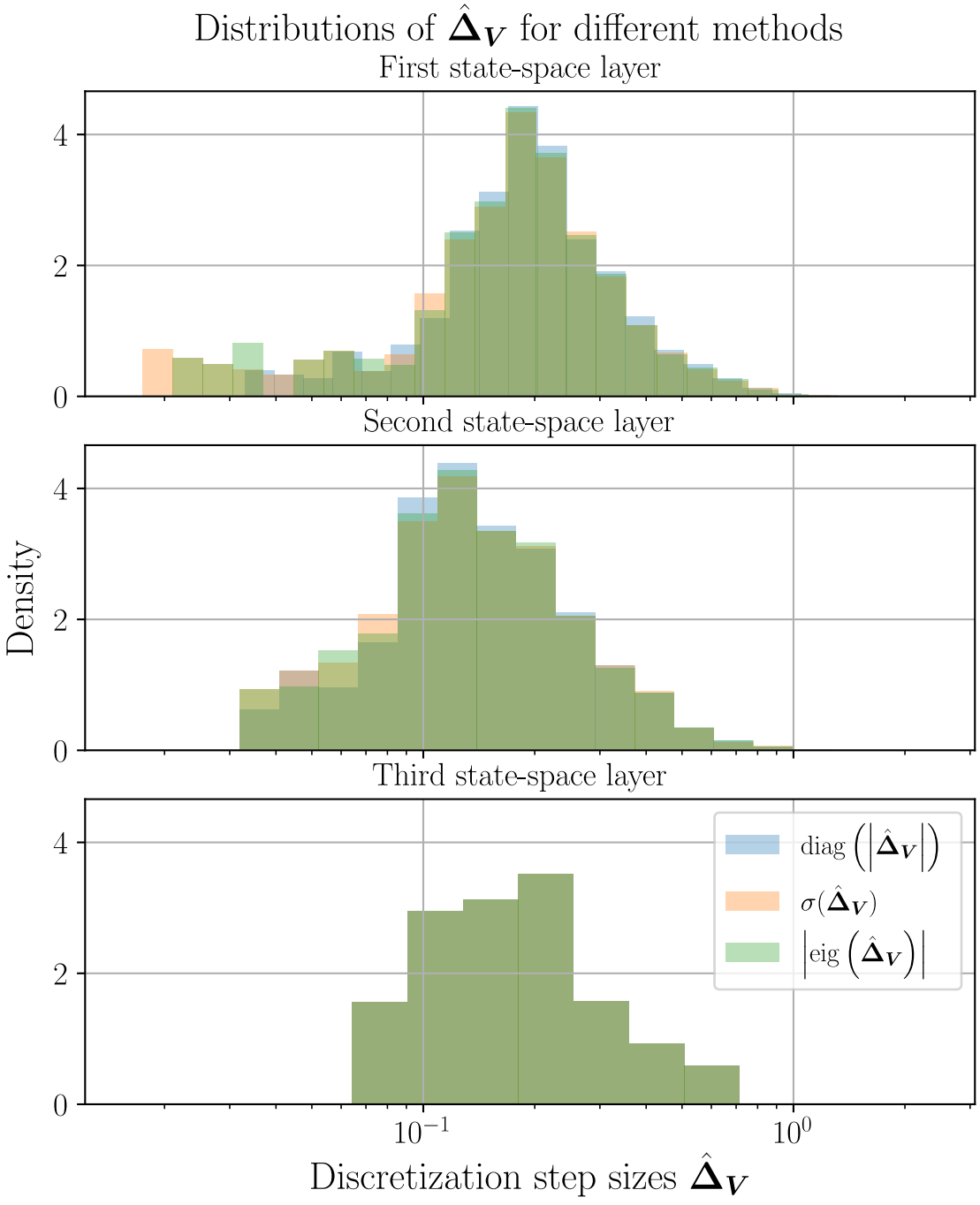

Figure 4.5: The distributions of $\hat{\mathbf{\Delta}}_{\boldsymbol{V}}$ for the three different truncation methods for each state-space layer, all very similar distributions in each layer.

5

Discussion

Many methods were tried that did not show to work, but instead provided valuable insights. Here, they are discussed including the importance of splitting the data by time series for XGBoost, experimentation with balanced realization, insights into PCA, and other approaches for time series prediction of fuel consumption that were deemed infeasible. Moreover, future work is also discussed here.

5.1 The importance of data splitting for XGBoost

Since the implementations of XGBoost were independent of time, the data was initially randomly divided into training, validation, and test sets *without* respect to the time series sequences. This produced extraordinary performance, but this was due to *data leakage* [44]. Consider a time series; a test or validation point could lie between two training points, which in turn could also lie between two other training points, and so on. Since the sample rate was relatively high, a group of consecutive data points was probably all very similar, which means that the model could confidently predict any test or validation point. This led to the model being able to predict the test and validation sets extremely well. However, this is not a valid comparison to real-time prediction. To make a correct assessment of the predictions, it was therefore very important to split the data with respect to time series, not individual data points, even though the architecture of XGBoost technically allowed the latter.

5.2 Experimentation with balanced realization

5.2.1 Bottlenecks between layers

The S5-layers presented potential bottlenecks due to the discrepancy between the number of inputs/outputs (10) and the much larger number of state dimensions (512). In neural networks, such bottlenecks, where information is densely compressed and then expanded, can lead to information loss and slower training.

Experimentation revealed a surprising result: increasing the bottleneck size while maintaining the same number of state dimensions actually decreased performance. Increasing the bottleneck size to match the state dimensions resulted in a nearly-linear relationship between Hankel singular values and their indices, indicating that truncation would cause significant energy loss. This suggests that when the number of outputs equals the number of states, the network learns to utilize each state by "load balancing", making them all important.

This finding hints at a potential relationship between the distribution of Hankel singular values (σ_i) and the ratio of state dimension (P) to output dimension (N).

When $P \gg N$, the weight of the singular values would probably cluster around a few elements, benefiting from truncation. In contrast, when $N \approx P$, the distribution is likely to flatten and spread out, requiring most or all states for energy conservation.

In the end, the size of the bottlenecks was set to the number of inputs, which yielded similar performance and while allowing for more efficient truncation.

5.2.2 The time discretization

With regard to the truncation of the time discretization steps, $\hat{\mathbf{\Delta}}_{\boldsymbol{V}}$, the most natural method for handling the projection back into a real and diagonal matrix would be not having to perform that transformation at all, i.e., only using (3.4), which in theory seems to work. However, for unknown reasons, this performed miserably.

Yet another approach to manage the issue of the complex and non-diagonal step size was to try to jointly diagonalize the pair $\hat{\mathbf{\Delta}}$ and $\hat{\boldsymbol{A}}$ using *Joint Approximate Diagonalization Under Orthogonality Constraints, (JADOC)* [45]. However, this did not work because joint diagonalization requires $\hat{\boldsymbol{A}}$ and $\hat{\mathbf{\Delta}}$ to commute, i.e., to both be *Hermitian*, which they are not.

5.3 Other approaches for time series prediction

Many different approaches were tested to predict fuel consumption. Some of them were recurrent time series *forecasting* of fuel per distance, that is, approaches that predicted fuel per distance ahead of time and using this forecast for the next time step. These approaches included fully recurrent forecasting of all features within the feature set as well as NARX models using XGBoost as a basis function and speed over ground, position and weather as exogenous inputs. The forecasts for these approaches were discovered to drift quickly, which is not surprising since the data contain a lot of variance, stochasticity, and complex underlying dynamics. If one were to use a time-dependent predictive model for a complex task such as prediction of fuel consumption of vessels, we assess that it would be wise to choose a structured state-space model because it is more capable of capturing the underlying dynamics of the vessel; a recurrent approach for this complex task is quite naive.

5.4 Insights into principal component analysis

Regarding S5, an interesting approach was tested that compressed the prediction features into fewer features using PCA. In this case, the strongest predictor, without doubt, was speed over ground, while the other features had similar correlations with the target. When performing PCA on *all* signals the high correlation between the new best signal and the target was significantly lower than that of speed over ground. Instead, other newly composed signals gained; the relevance of the features was distributed between all the components. This did not perform well, and another approach was proposed:

Instead, speed over ground was withheld and PCA was only performed on the remaining features which already had similar relevances. This performed much better and would be a viable approach when the computational budget is highly limited. In this case, not performing PCA at all performed better.

The explanation for this is due to the nature of PCA; the procedure is unsupervised and unaware of the target, and thus aims to maximize the variance *among all*

features without distinction or weighting.

5.5 Potential improvements

To include more data, suggestively data collected during the fall to also capture environmental factors during this season, as well as including data collected over multiple years, could potentially improve overall performance. The reason for this is that since the feature domain is relatively large, containing about 10 features, more data could potentially lead to better predictions during optimization. Again, this can be explained by the curse of dimensionality. Furthermore, including forecasted weather data of higher resolution, in the unlikely event that one can find it, would probably improve the accuracy of the predictions. Here, the weather data were updated every hour, and the spatial resolution was also quite low.

Furthermore, there exists a newly introduced correlation coefficient called *Chatterjee's* ξ [46], which, for the case of no rank ties, is defined as

$$\xi_n(X, Y) = 1 - \frac{3\sum_{i=1}^{n-1} |r_{i+1} - r_i|}{n^2 - 1}, \tag{5.1}$$

where r_i are the ranks of a random variable Y, sorted in such a way that $X_1 < \cdots < X_n$ and n is the number of data points.

This has many useful properties: It is asymmetric, it does not assume some distribution or relationship between variables, such as the linear relationship for Pearson's method or the monotonic relationship for Spearman's method, and it indicates 0 for no dependency and 1 for full dependency.

This allows Chatterjee's method to indicate non-linear, non-monotonic, and even oscillatory feature dependencies. Essentially, it tries to answer the question: "To what extent is Y a function of X?". If we had known about this earlier, it would probably have been used in place of Spearman's method. It would have been useful for the data exploration part, when not much is known about the feature dependencies and possibly also for the feature selection.

5.6 Future work

5.6.1 Optimizing fuel consumption

The ability to predict fuel consumption is of little use in itself. The next step would be to use the trained models in the optimization setting and investigate how the predictions behave when sweeping over ranges of speed over ground. Since the models use comprehensive sets of around 10 features, the feature domains could be quite large for the amount of data used in training. The prediction accuracy for the commonly seen data is reasonably good, as has been shown, but when sweeping the speed to optimize, some speeds could potentially be far out of the joint high-dimensional distributions, which could lead to poor predictions.

This can be motivated by the curse of dimensionality, as it could be easier to diverge from the distribution of the data in higher dimensions than in lower ones. With fewer features, the prediction accuracy for the test data is lower, but could potentially lead to more stable predictions when sweeping. However, with more features, the prediction accuracy for the test data is higher but could potentially lead to unstable predictions when sweeping. Thus, it could probably, counter-intuitively, exist

a *trade-off between prediction accuracy, optimization accuracy, and computational costs*. However, this remains to be investigated.

One could potentially use PCA on the feature set withholding speed over ground and only keep, perhaps, the best component, resulting in a tiny feature set of only two features. This would probably be suitable for sweeping without requesting predictions to be performed on data points out-of-distribution.

Moreover, sweeping the speed with the S5 is not straightforward and is likely a complicated process. One would have to generate many time series for different scenarios of cruising speeds and watch the steady-state response and optimize accordingly. This could potentially be an impractical approach, but it remains to be implemented and tested.

5.6.2 Further exploring balanced realization

With regard to balanced realization, even though the truncation of the time discretization steps was shown to be effective, further research is needed on how to properly truncate and transform the time discretization steps using methods more theoretically sound.

It would also be interesting to explore whether there are benefits in balancing the Gramians of each layer *during* training without truncating. This could potentially work as an advanced normalization and regularization technique for all structured state-space models, such as the newly introduced state-of-the-art *Mamba* model [47]. Probably, the balanced truncation would also be applicable to the Mamba, but its effectiveness and potential hindrances involving its selection mechanism remains to be investigated.

6

Conclusions

Here, the thesis is concluded, briefly stating the outcome of the intended objectives and what remains.

In this thesis, the predictive capabilities of XGBoost and S5 for fuel consumption have been investigated for the case of the fixed-route Burö ferry using speed over ground as a control parameter while considering weather forecasts.

Regarding the first objective: *to determine the statistical dependence between weather and fuel consumption of Burö.* It was discovered that the dependence between fuel consumption and forecasted weather is significant enough to produce substantially better results when these are incorporated as features in the models. The most significant being the wind-related features (wind, gust, and wind waves) decomposed into head components in the reference frame of the ferry.

Regarding the second and third objectives: *to develop XGBoost and S5 models capable of predicting fuel consumption of Burö with appropriate features while accounting for weather, and evaluate their performance in terms of accuracy.* XGBoost and S5 have been tested as time-independent and time-dependent predictive models, respectively, and have produced reasonably accurate predictions. In short, there is a clear trade-off between them, namely that XGBoost allows for better data preparation but is naive in the sense that it does not account for any dynamics, whereas S5 does the opposite. XGBoost performs best, 39 % better than S5, first by predicting the power of the engine and then the fuel consumption. In addition to speed over ground, appropriately selected features were position, leg, and various weather selected from Norlund's feature selection; a new feature selection algorithm introduced here.

Regarding the fourth objective: *to perform balanced realization on a trained S5 and assess its effectiveness from a systems and control perspective.* This was done to the implemented S5 in a novel way to preserve 95 % of the energy of each state-space. It was found that the model could be effectively compressed with only 8 % worse performance. To truncate the time discretization steps, different methods were experimented with and a naive approach was found to be the most effective.

In terms of stability, the state-space layers by themselves are not robust but are probably more expressive because of this. A theory is that the non-linearities are attenuating the state-space layers, explaining why it can produce reasonable results. The next step here would be to further investigate how to properly truncate the time discretization steps using a method more in line with already established theory within systems and control.

The developed methodology can be extended to the general case of any fixed-route vessel and remains to be tried on other cases. For Burö, the next step is to apply the models implemented in the fuel consumption optimization setting to investigate how the prediction behaves when speed is swept. This is vital since there could be

a trade-off between prediction and optimization accuracy explained by the curse of dimensionality.

Bibliography

[1] World Meteorological Organization (WMO), *WMO confirms that 2023 smashes global temperature record*, Accessed on [2024/03/07], Jan. 2024. [Online]. Available: https://wmo.int/news/media-centre/wmo-confirms-2023-smashes-global-temperature-record (return to page 1).

[2] Interngovernmental Panel on Climate Change (IPCC), Core Writing Team, H. Lee, and J. Romero, "Climate Change 2023: Synthesis Report," Geneva, Switzerland, 2023, p. 43. DOI: 10.59327/IPCC/AR6-9789291691647 (return to page 1).

[3] United Nations (UN), *Causes and effects of climate change*, Accessed on [2024/03/07]. [Online]. Available: https://www.un.org/en/climatechange/science/causes-effects-climate-change (return to page 1).

[4] International Maritime Organization (IMO), "Fourth Greenhouse Gas Study 2020," London, England, 2021, p. 29 (return to page 1).

[5] International Maritime Organization (IMO), *2023 IMO Strategy on Reduction of GHG Emissions from Ships*, Accessed on [2024/03/12], 2023. [Online]. Available: https://www.imo.org/en/OurWork/Environment/Pages/2023-IMO-Strategy-on-Reduction-of-GHG-Emissions-from-Ships.aspx (return to page 1).

[6] X. Lang, D. Wu, and W. Mao, "Comparison of supervised machine learning methods to predict ship propulsion power at sea," *Ocean Engineering*, vol. 245, p. 110 387, 2022. DOI: https://doi.org/10.1016/j.oceaneng.2021.110387 (return to page 1).

[7] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16, ACM, Aug. 2016. DOI: https://doi.org/10.48550/arXiv.1603.02754 (return to pages 1, 13).

[8] A. Gu, K. Goel, and C. Ré, *Efficiently modeling long sequences with structured state spaces*, 2022. arXiv: 2111.00396 [cs.LG] (return to pages 1, 22).

[9] J. T. Smith, A. Warrington, and S. Linderman, "Simplified state space layers for sequence modeling," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=Ai8Hw3AXqks (return to pages 2, 21, 23).

[10] S. L. Brunton and J. N. Kutz, "Data-driven science and engineering: Machine learning, dynamical systems, and control," in 2nd ed. Cambridge University Press, 2022, pp. 436–442. DOI: https://doi.org/10.1017/9781009089517.013 (return to pages 3, 18, 24, 27).

[11] Stormglass, *Weather*, Accessed on [2024/04/29]. [Online]. Available: https://docs.stormglass.io/#/weather (return to page 5).

[12] L. H. Holthuijsen, *Waves in oceanic and coastal waters*. New York, USA: Cambridge university press, 2007, pp. 123–136 (return to page 5).

[13] H. N. Psaraftis and S. Lagouvardou, "Ship speed vs power or fuel consumption: Are laws of physics still valid? regression analysis pitfalls and misguided policy implications," *Cleaner Logistics and Supply Chain*, vol. 7, 2023. DOI: https://doi.org/10.1016/j.clscn.2023.100111 (return to pages 6, 7).

[14] MAN Energy Solutions, *Basic principles of ship propulsion*, Accessed on [2024/04/29], Copenhagen, Denmark, 2023. [Online]. Available: https://www.man-es.com/docs/default-source/marine/tools/basic-principles-of-ship-propulsion_web_links.pdf?sfvrsn=12d1b862_14 (return to pages 6, 7).

[15] M. L. Samuels, J. A. Witmer, and A. A. Schaffner, *Statistics for the life sciences*. Pearson Education, 2016, vol. 5, pp. 523–530 (return to page 7).

[16] H. Yu and A. D. Hutson, "A robust spearman correlation coefficient permutation test," *Communications in Statistics: Theory and Methods*, vol. 53, no. 6, pp. 2141–2142, 2022. DOI: https://doi.org/10.1080/03610926.2022.2121144 (return to page 8).

[17] I. Jolliffe, *Principal Component Analysis* (Springer Series in Statistics), Second edition. New York, USA: Springer, 2002 (return to page 8).

[18] S. L. Brunton and J. N. Kutz, "Singular value decomposition (svd)," in *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2022, pp. 3–52. DOI: https://doi.org/10.1017/9781009089517.003 (return to page 10).

[19] M. A. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, University of Waikato, Hamilton, New Zealand, Apr. 1999, pp. 25–26 (return to page 11).

[20] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1226–1238, 2003. DOI: https://doi.org/10.1109/TPAMI.2005.159 (return to page 11).

[21] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, First edition. Chapman & Hall/CRC, 1984. DOI: https://doi.org/10.1201/9781315139470 (return to page 11).

[22] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, Second edition. New York, USA: Springer, 2009, pp. 305–317. DOI: https://doi.org/10.1007/978-0-387-84858-7 (return to page 12).

[23] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, Second edition. New York, USA: Springer, 2009, pp. 337–387. DOI: https://doi.org/10.1007/978-0-387-84858-7 (return to page 12).

[24] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, vol. 29, pp. 1189–1232, 2001. DOI: http://dx.doi.org/10.1214/aos/1013203451 (return to page 12).

[25] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996. DOI: https://doi.org/10.1007/BF00058655 (return to page 16).

[26] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960 (return to page 16).

[27] G. F. Franklin, J. D. Powell, M. L. Workman, *et al.*, *Digital control of dynamic systems*. Addison-wesley Reading, MA, 1998, vol. 3, pp. 44–46, 79–83, 222–223 (return to page 18).

[28] S. L. Brunton and J. N. Kutz, "Linear time-invariant systems," in *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2022, pp. 383–386. DOI: https://doi.org/10.1017/9781009089517.012 (return to page 19).

[29] T. Glad and L. Ljung, *Control Theory: Multivariable and Nonlinear Methods*, First edition. London, England: Taylor & Francis, 2000, pp. 18–19. DOI: https://doi.org/10.1201/9781315274737 (return to page 19).

[30] S. L. Brunton and J. N. Kutz, "Controllability and observability," in *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2022, pp. 389–396. DOI: https://doi.org/10.1017/9781009089517.012 (return to page 19).

[31] A. Gu, T. Dao, S. Ermon, A. Rudra, and C. Re, *Hippo: Recurrent memory with optimal polynomial projections*, 2020. arXiv: 2008.07669 [cs.LG] (return to page 20).

[32] A. Gu, A. Gupta, K. Goel, and C. Ré, *On the parameterization and initialization of diagonal state space models*, 2022. arXiv: 2206.11893 [cs.LG] (return to pages 21, 24).

[33] A. Gu, I. Johnson, A. Timalsina, A. Rudra, and C. Ré, *How to train your hippo: State space models with generalized orthogonal basis projections*, 2022. arXiv: 2206.12037 [cs.LG] (return to page 21).

[34] E.-W. Bai and F. Giri, "Introduction to block-oriented nonlinear systems," in *Block-oriented Nonlinear System Identification*. London: Springer London, 2010, pp. 3–11 (return to page 22).

[35] F. Bonassi, C. Andersson, P. Mattsson, and T. B. Schön, *Structured state-space models are deep wiener models*, 2023. arXiv: 2312.06211 [eess.SY] (return to page 22).

[36] K. Goel, A. Gu, C. Donahue, and C. Ré, *It's raw! audio generation with state-space models*, 2022. DOI: https://doi.org/10.48550/arXiv.2202.09729. arXiv: 2202.09729 [cs.SD] (return to page 23).

[37] A. Dolara, F. Grimaccia, S. Leva, M. Mussetta, and E. Ogliari, "Comparison of training approaches for photovoltaic forecasts by means of machine learning," *Applied Sciences*, vol. 8, no. 2, 2018, ISSN: 2076-3417. DOI: https://doi.org/10.3390/app8020228 (return to page 27).

[38] O. Nelles, "Nonlinear dynamic system identification," in *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Mod-*

els. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 547–577. DOI: https://doi.org/10.1007/978-3-662-04323-3_15 (return to page 28).

[39] D. Peng, Z. Gui, and H. Wu, *Interpreting the curse of dimensionality from distance concentration and manifold effect*, 2024. arXiv: 2401.00422 [cs.LG] (return to page 28).

[40] G. Hughes, "On the mean accuracy of statistical pattern recognizers," *IEEE Transactions on Information Theory*, vol. 14, no. 1, pp. 55–63, 1968. DOI: https://doi.org/10.1109/TIT.1968.1054102 (return to page 28).

[41] J. A. Hanley, L. Joseph, R. W. Platt, M. K. Chung, and P. Belisle, "Visualizing the median as the minimum-deviation location," *The American Statistician*, vol. 55, no. 2, pp. 150–152, 2001. DOI: https://doi.org/10.1198/000313001750358482 (return to page 41).

[42] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein, *On the expressive power of deep neural networks*, 2017. arXiv: 1606.05336 [stat.ML] (return to page 48).

[43] G. Tallec, E. Yvinec, A. Dapogny, and K. Bailly, "Fighting over-fitting with quantization for learning deep neural networks on noisy labels," in *2023 IEEE International Conference on Image Processing (ICIP)*, 2023, pp. 575–579. DOI: 10.1109/ICIP49359.2023.10222687 (return to page 48).

[44] S. Kaufman, S. Rosset, and C. Perlich, "Leakage in data mining: Formulation, detection, and avoidance," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '11, New York, USA: Association for Computing Machinery, 2011, pp. 556–563. DOI: https://doi.org/10.1145/2020408.2020496 (return to page 55).

[45] R. de Vlaming and E. A. W. Slob, *Joint approximate diagonalization under orthogonality constraints*, 2021. arXiv: 2110.03235 [math.NA] (return to page 56).

[46] S. Chatterjee, *A new coefficient of correlation*, 2020. arXiv: 1909.10140 [math.ST] (return to page 57).

[47] A. Gu and T. Dao, *Mamba: Linear-time sequence modeling with selective state spaces*, 2024. arXiv: 2312.00752 [cs.LG] (return to page 58).

A

Appendix 1

The hyperparameters used for the training of the S5 and the specific final results of all models are shown here.

Table A.1: Hyperparameters of the S5 implementation yielding the best result. T_0, T_{MULT} and $\eta_{\min}$ are the hyperparameters used for cosine annealing.

Hyperparameter	Value
State dimensions	512
S5 layers	3
Number of outputs in-between S5 layers	8
Initialization of discretization step sizes	$\boldsymbol{\Delta} \sim U(1 \times 10^{-3}, 1 \times 10^{-1})$
Dropout	0
Batch size	16
Maximal learning rate η	5×10^{-5}
T_0	5
T_{MULT}	2
$\eta_{\min}$	1×10^{-5}

Table A.2: The number epochs, including boosting rounds, and test losses corresponding to the best results for XGBoost and S5.

Model	Epochs	Test (MAE)
XGB: speed only	123	0.3220 [mL m^{-1}]
XGB: speed/weather/position/leg (to fuel)	1015	0.1808 [mL m^{-1}]
XGB: speed/weather/position/leg (to power)	726	0.6349 [kW]
XGB: pred. power/speed/weather/position/leg	358	0.1767 [mL m^{-1}]
S5: full state	152	0.2974 [mL m^{-1}]
S5: truncated state, diag $\lvert\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}}\rvert$	152	0.3141 [mL m^{-1}]
S5: truncated state, $\sigma(\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}})$	152	0.3428 [mL m^{-1}]
S5: truncated state, $\lvert\text{eig}\left(\hat{\boldsymbol{\Delta}}_{\boldsymbol{V}}\right)\rvert$	152	0.3933 [mL m^{-1}]

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY