Doctoral theses at NTNU, 2023:206

Erlend Torje Berg Lundby

Data-Driven Dynamical Modeling in the Face of Data Limitations

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Erlend Torje Berg Lundby

Data-Driven Dynamical Modeling in the Face of Data Limitations

Thesis for the Degree of Philosophiae Doctor

Trondheim, June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-7114-4 (printed ver.)
ISBN 978-82-326-7113-7 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2023:206

Printed by NTNU Grafisk senter

Summary

Data-driven modeling has experienced an enormous increase in popularity recent years due to the ever-growing data abundance, access to cheap computational resources and great advances in algorithms and methodology. This has lead to great accomplishments done by Machine Learning (ML) models within a range of domains. This extraordinary success has not gone unnoticed within different process industries. Motivated by the potential to improve efficiency, reduce energy consumption, prevent severe incidents, costly downtime and more, stakeholders in the industries are now looking to the ML community for solutions. However, unlike a range of the successful showcases of data-driven modeling approaches, process industries typically struggle with challenges of using ML models due to limited access to informative data. Moreover, the safety critical nature of most industrial processes put additional requirements on models' ability to generalize to the broader operational window, as well as knowledge about when the models fail. This does in general not coincide with highly complex ML models, which typically requires vast data to generalize, as well as suffering from low interpretability, challenging the trustworthiness of these types of models when applied in dynamical processes.

This thesis presents a set of approaches utilizing Data-Driven Models (DDMs) of dynamical processes under the constraint of limited data. A novel hybrid modeling approach, combining Physics Based Models (PBMs) and compressed sensing was developed and utilized to estimate unmodeled dynamics in a measured signal sampled at low frequency in an aluminum electrolysis simulator. Subtracting the PBM estimate from the measurements leaves us with a manipulated coarsely sampled signal representing unmodeled dynamics. This manipulation enables the powerful tool of compressed sensing, which can now estimate the unmodeled dynamics from measurements sampled at a much lower frequency than what the Shannon-Nyquist theorem requires. Another hybrid modeling approach presented in this thesis combines PBMs with Neural Networks. The NN is used to correct the modeling errors of a PBM by adding a corrective source term to the set of governing Ordinary Differental Equations (ODEs) known as the PBM. The method

combines the best of both modeling approaches, while eliminating some of their weaknesses. That is, the resulting hybrid modeling approach keeps the interpretability of the PBM, and correct for its error using the NN. Moreover, the PBM simplifies complexity of the learning problem for the NN, leading to a reduced need for data. The modeling approach is showcased on a set of high dimensional ODEs representing the mass and energy balance of the aluminum electrolysis.

Moreover, a purely data-driven modeling approach was used in another work of this PhD. The effect of sparsity promoting ℓ_1 regularization on the generalizability, interpretability, and training stability when modeling a high dimensional set of ODEs was studied and compared to a densely connected NN. The results show that the ℓ_1 regularization significantly reduces the complexity of the model, making the model more interpretable. Moreover, the results show that the sparse NN generalizes better, and yield more stable convergence under the constraints of limited access to training data. Building on this work, we introduce skip-connections to a NN for modeling high-dimensional nonlinear dynamics. The combination of sparsity promoting regularization and skip-connections significantly improves model accuracy and predictive stability for models trained on limited data. The case studies in both works were conducted on a set of high-dimensional ODEs representing the aluminum electrolysis.

Finally, the problem of maximizing the information content inherent in the training set was addressed. The goal was to excite the system dynamics to obtain the most informative data for training Deep Neural Networks (DNNs). We present a novel framework that samples a set of simulated informative state-action trajectories distributed around the state-action space. This enables utilizing a novel static Batch Mode Deep Active Learning (BMDAL) acquisition formulation to choose the most informative regions in the state-action space in which to exite the system dynamics. The case study show that the proposed method can outperform state-of-the-art random based sampling methods in terms of providing training data such that DNNs faster converges to acceptable model performaces in terms of accuracy and generalization.

The work in this thesis has contributed with methodology that addresses some challenges related to limitation of data which prevents the use of DDMs in process industries that could potentially be highly beneficial. However, a wide range of challenges still needs to be addressed, related to noise and disturbances, low observability as well as safety concerns.

Sammendrag

Datadrevet modellering får stadig mer oppmerksomhet som følge av tilgang på økende mengde data og billige databehandlingsressurser, samt store fremskritt innen algoritmer of metodikk. Dette har ført til at maskinlæringsmodeller har løst en rekke kompliserte modelleringsproblem innen ulike domener. Denne suksessen har fått oppmerksomhet blant flere prosessindustrier, som nå ser på muligheten til å bruke maskinlæring for å blant annet øke produksjonsrater, redusere energiforbruk, forutse og dermed unngå alvorlighe hendelser i produksjon og dyrbar nedetid og så videre. Prosessindustrien møter imidlertid store utfordringer knyttet til sikkerhetskritiske aspekter samt manglede tilgang på data. Avanserte maskinlæringsmodeller trenger typisk mye data for å kunne generalizere, samtidig som de som regler er vansklige å tolke på grunn av deres kompleksitet. Kombinasjonen av disse faktorene kompliserer bruken av maskinlæringsmodeller i prosessindustrien.

Denne avhandlingen presenterer en rekke bidrag med datadrevne modeller som på ulike vis addresserer utfordringer knyttet til å modelere dynamiske prosesser med begrenset tilgang på data. En ny hybrid modell som kombinerer en fysikkbasert modell med compressed sensing kan estimere periodisk, stasjonær umodellert dynamikk i et signal som er målt ved svært lave frekvenser. I casestudiet ble det brukt en enkel simuleringsmodell av aluminumselektrolysen, og det målte signalet var metallhøyden av smeltet aluminum i elektrolysecellen. En annen hybrid modell presentert i avhandlingen bruker et neuralt nettverk til å rette opp modelleringsfeilen til en fysikkbasert modell. Den resulterende hybride modellen kombinerer det beste av begge modellene samtidig som den eliminerer noen av ulempene de bærer. Det vil si, den hybride modellen beholder tolkbarheten til den fysikkbaserte modellen, samtidig som det neurale nettverket reduserer modellfeilen i den fysikkbaserte modellen. Det neurale nettverket trenes på manipulert data, det vil si residualet mellom målinger fra prosessen og fysikkbaserte estimater av målingene. Dette reduserer kompleksiteten i læringsproblemet og fører til at det kreves mindre data for å oppnå gode datadrevne modeller. Casestudiet ble utført på et sett av høydimensjonelle ordinære differensiallikninger (ODE'er) som representerer masse-,

og energibalansen i aluminumselektrolysen.

I tillegg til å bruke hybride modeller for å løse utfordringer knyttet til manglende data presenterer denne avhandlingen også rene datadrevne tilnærminger. Effekten av den kompleksitetsreduserende ℓ_1 regulariseringen på neurale nettverks evne til å generalisere, samt nettverkens treningsstabilitet og tolkbarhet ble analysert i et av bidragene. Resultatene av casestudiet viste at ℓ_1 regulariserte nettverk oppnådde bedre generaliseringsevner, samt mer stabil treningskonvergens sammenliknet med nettverk trent uten denne regulariseringen. Videre gjorde den reduserte kompleksiteten at tolkbarheten til modellene økte. Casestudien ble utført på samme høydimensjonelle set av ODE'er som i sistnevnte hybride modell. I det påfølgende arbeidet som bygger på disse resultatene så introduserte vi skip-connections til modellstrukturen for å modellere det samme dynamiske systemet. Det viste seg i casestudien at modeller med skip-connections som trenes med ℓ_1 regularisering oppådde økt modellnøyaktighet, samt prediktiv stabilitet for modeller trent med begrensede mengder data.

Det sise bidraget i denne avhanlingen tar for seg informasjonsinnholdet i treningsdataen. Målet er å eksitere et dynamisk system på en slik måte at man får mest mulig informasjon ut av det eksiterte dynamiske systemet. Dette gjør man for å sample data som skal brukes til å trene neurale nettverk. Vi presenterer et nytt rammeverk basert på active learning. Først sampler man et sett av simulerte, informative state-action baner fordelt rundt om i tilstandsrommet. Dette muliggjør å utnytte en ny statisk aktiv læringsformulering som søker finne de mest informative områdene i tilstandstommet for så å eksitere systemet i disse områdene. Resultatene viser at active learning metoden kan utkonkurrere de beste eksitasjonsmetodene basert på tilfeldig eksitasjon.

Arbeidet i denne avhandlingen har bidratt med metodikk som addreserer noen av utfordringene knyttet til manglende treningsdata som vanskligjør bruken av datadrevne modeller i prosessindustri. Det er imidlertid en rekke utfordinger som ikke er addressert, knyttet til målestøy, prosessforstyrrelser, lav observerbarhet samt sikkerhetskritiske aspekter.

Contents

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (PhD) at the Norwegian University of Science and Technology (NTNU), Trondheim.

The work presented has been conducted at the Department of Engineering Cybernetics (ITK), NTNU. The project supervisor has been Professor Jan Tommy Gravdahl, and co-supervisor Professor Adil Rasheed, both from the Department of Engineering Cybernetics, and co-supervisor Dr. Ivar Johan Halvorsen, senior researcher at SINTEF Digital. This work was supported by the industry partners Borregaard, Elkem, Hydro, Yara and the Research Council of Norway through the project TAPI: Towards Autonomy in Process Industries, project number 294544.

Acknowledgements

I would like to express my sincere gratitude to Professor Jan Tommy Gravdahl, my main supervisor, for seamlessly administering my PhD project and enabling me to focus solely on research. Moreover, Jan Tommy has been open to my ideas and motivated me to explore and pursue my research interests. His profound academic insights have played a key role in sharpening the contributions in this thesis. I would also like to extend my gratitude to my co-supervisor, Professor Adil Rasheed, for being an extraordinary resource during my PhD work. Adil has dedicated numerous hours with me writing academic articles, formalizing research ideas, and teaching me fundamental theories, as well as discussing topics relevant to my research. I deeply admire your exceptional work ethic and unwavering passion for helping all your students reach their fullest potential. Additionally, I want to convey my appreciation to my co-supervisor, Dr. Ivar Johan Halvorsen. Ivar possesses extensive industry knowledge, which he has shared through interesting discussions in both formal meetings and informal conversations before and after football practices. Ivar's perspectives have been invaluable in shaping the research and understanding the broader motivations behind our contributions. To all of my

supervisors, I want to express my heartfelt gratitude for your open-mindedness, which has made it effortless for me to freely voice my concerns and opinions on various topics. This has been crucial in fostering the trusting relationship I have with each of you, and for that, I am truly grateful.

I have had the privilege to work with highly skilled individuals at ITK. First and foremost, I would like to thank Haakon Robbinson for the immensely valuable collaborations we have had over an extended period during my PhD. Further, I want express my gratutde to Professor Sebastien Gros and Dr. Dirk Peter Reinhardt for assisting me in my latest work. I want to thank Emil Johannesen Haugstvedt and Alberto Miño Calero for the collaboration we had together. In addition, I would like to convey my appreciation to all my friends at ITK who have been crucial in fulfilling my social needs throughout the PhD period.

Lastly, I want to express my deepest appreciation for my parents, Rigmor and Halvdan, who have always supported me in every conceivable way and have shown me unconditional love. And to my lovely love Gro, who stood by my side through the toughest parts of my PhD.

Abbreviations

AL Active Learning.

AN-RFMSE Average Normalized Rolling Forecast Mean Squared Error.

APRBS Amplitude-modulated Pseudo-Random Binary Signal.

ARX Auto Regressive with eXternal input.

BMDAL Batch Mode Deep Active Learning.

BPDN Basis Pursuit Denoising.

CoSAMP Compressive Sampling Matching Pursuit.

CoSTA Corrective Source Term Approach.

DCT Discrete Cosine Transform.

DDM Data-Driven Model.

DeepAL Deep Active Learning.

DL Deep Learning.

DMBAL Diverse Mini-Batch Active Learning.

DNN Deep Neural Network.

DOF Degrees of Freedom.

DS Dantzig selector.

EKF Extended Kalman Filter.

GP Gaussian Process.

HAM Hybrid Analysis and Modeling.

HTP Hard Thresholding Pursuit.

IHT Iterative Hard Thresholding.

LASSO Least-Absolute Shrinkage Selection Operator.

ML Machine Learning.

MLP Multi-Layer Perceptron.

MPC Model Predictive Control.

MSE Mean Squared Error.

NED North East Down.

NLP Natural Language Processing.

NMPC Nonlinear Model Predictive Control.

NN Neural Network.

OCP Optimal Control Problem.

ODE Ordinary Differential Equation.

OMP Othogonal Matching Pursuit.

PBM Physics-Based Model.

PDE Partial Differential Equation.

PE Persistency of Excitation.

PGML Physics Guided Machine Learning.

PINN Physics Informed Neural Network.

PWA Piecewise Affine.

PWL Piece-Wise Linear.

QCBP Quadratic Constraint Basis Pursuit.

ReLU Rectified Linear Unit.

RFMSE Rolling Forecast Mean Squared Error.

RIP Restricted Isometry Property.

RK4 Runge-Kutta 4.

RMSE Rooted Mean Squared Error.

SINDy Sparse Identification of Nonlinear Dynamics.

Chapter 1

Introduction

1.1 Motivation

Process modeling is a cornerstone of a wide range of industrial processes, serving several objectives. That is, process models are used in design of controllers, process optimization, state estimation, decision support, designing process plants etc. This has motivated businesses throughout the years to invest in developing and maintaining models that describe their respective underlying dynamical systems. First principle Physics-Based Models (PBMs) have been widely utilized in the modeling of complex dynamical systems. Despite their effectiveness in representing the underlying mechanisms, their use has been met with challenges arising from an incomplete understanding of the phenomena, the computationally intensive nature of the models, and the inherent uncertainty associated with the factors influencing the dynamics. In light of these challenges, there has been a growing interest in data-driven modeling approaches as an alternative to PBMs. The increased availability of copious amounts of data, the accessibility of cheap computational resources, and the advancement of algorithms and methodology have further catalyzed the adoption of Data-Driven Models (DDMs) in various scientific and engineering applications. This includes for example material science [1], biomechanics [2], production of biofuels [3], reservoir modeling in oil and gas industry [4], bioengineering [5], drug discovery [6] and more [7]. The key advantage of DDMs is their ability to accurately model complex phenomena directly from the data, even in cases where a comprehensive understanding of the underlying mechanisms is lacking.

Learning and deploying DDMs in industrial processes is challenging due to various factors, including the nonlinearity and high dimensionality of process dynamics

which pose a significant difficulty for DDMs. Although Neural Networks (NNs) are inherently suitable for modeling such complexities, they are highly flexible and demand vast and varied datasets to ensure accurate and generalized model training. Complex DDMs also suffer in general from low interpretability making it difficult to understand and validate the predictions made by the DDMs. Furthermore, DDMs have shown to be unstable to train, affecting prediction properties [8].

The primary production of aluminum involves the extraction of the metal from alumina through the electrolytic process known as the Hall-Héroult process [9]. The aluminum electrolysis is a high dimensional, highly nonlinear industrial process with many interrelated physio–chemical sub–processes. Intensive research have lead to the development of complex PBMs of different sub–processes of aluminum electrolysis. In [10], a mass and energy balance model based on the first law of thermodynamics was developed. Authors of [11] use fundamental equations from fluid dynamics and electromagnetism to derive PBMs for the stationary magnetohydrodynamic flow and heaving of the liquid metal caused by the current induced magnetic field. A 3D multi-scale model predicting bubble flow derived from physical laws describing motions of gas in liquid, bubble growth based on chemical equations and Farraday's law and the coalscence of micro bubbles based on the probability of collision of bubbles was presented in [12]. The authors of [13] presented a multiscale, multiphysics modeling framework that combines magnetohydrodynamic model in [11], a mesoscale model to describe bubble behavior, and a full cell bath flow model into a full cell alumina distribution model.

Despite great advances in modeling the aluminum electrolysis from first-principles, state-of-the-art models still suffer from strong modeling bias due to the difficult modeling nature of the process. There are strong initiatives to improve modeling accuracy of the aluminum process. For example, increased insight in the process dynamics can potentially improve decision support, state estimation, and consequently process optimization while maintaining safe operation. Therefore, the industry is investing in research to map the utility of machine learning in the aluminum process. However, the aforementioned challenges of training and implementing DDMs are particulary present in the aluminum electrolysis. The environment in the aluminum electrolysis cell is extremely harsh due to the high temperatures and highly corrosive electrolyte [14, 15]. Sensor systems generally do not survive long in this environment, and most of the measurements taken of state variables in the process are done manually by process operators. Due to the cost of manual sampling, these measurements are taken at rare time-instants during operation. In addition, data collected from process operation in industrial settings is often limited due to the need for stability and safety control measures. This limited

information is often inadequate for training robust and accurate DDMs that can effectively handle the intricacies of high-dimensional and complex processes, such as aluminum electrolysis. Furthermore, the safety-critical nature of the aluminum electrolysis process requires highly reliable models to be utilized. The black-box nature of complex DDMs is generally not suitable for such requirements, and relying on barely interpretable DDM predictions cannot be justified.

The focus of this thesis is on addressing the aforementioned challenges related to data-driven modeling of dynamical systems under the constraints of limited data. The aluminum electrolysis is given significant attention, and most case studies are conducted on nonlinear Ordinary Differential Equations (ODEs) representing the aluminum electrolysis. In Chapter 7 however, the research is conducted on a surface vessel simulator. In principle, the scientific contributions of the thesis can be extended to any system of ODEs.

1.2 Background

The purpose of this section is to provide context for the contributions made in the thesis by briefly explaining and justifying the methodologies used in the research.

1.2.1 Hybrid modeling

By carefully observing physical phenomena, theories can be developed to understand the underlying system, which are then condensed into mathematical equations that can be solved to make predictions about the system. This is, as mentioned above, known as PBM. PBMs possess several advantages, including their intuitive and explainable nature, their ability to generalize well to situations where assumptions are upheld, and the availability of mature theories for analyzing their properties, such as stability and robustness to uncertainties and noise. However, accurate modeling of many real-world systems is computationally demanding. Assumptions may be made to reduce complexity and minimize computational requirements, particularly when developing control systems. Additionally, there may be limitations in accurately describing all aspects of observations, resulting in an incomplete, unfaithful, or overly simplified representation of the original system.

Data-Driven Modeling on the other side relies on direct approximation of the underlying function from measurement data, rather than an understanding of the underlying physics. In recent years, the rapid advancements in machine learning have led to a significant increase in the availability of data, enabling the development of DDMs for a wide range of applications, such as the identification of aluminum electrolysis processes [16] and the detection of contamination for polymer pellet quality control [17]. DDMs offer a high degree of flexibility and can often achieve exceptional accuracy with minimal computational requirements. This makes them

particularly useful in situations where a complete understanding of the underlying physics is lacking. However, it is widely recognized that these models tend to have poor generalization capabilities and often fail when presented with data that is not well represented by the training data. Furthermore, many types of DDMs require large amounts of data to be effective.

Due to these limitations, there is a preference for more transparent multivariate statistical models that can provide greater insight into industrial processes, such as estimating the internal state of an aluminum smelting process during operation [18]. To address the shortcomings of DDMs and PBMs, a new breed of modeling is emerging called hybrid analysis and modeling (HAM) [19]. The modeling approach combines PBMs and DDMs in order to utilize both the interpretability, generalizability, and robust foundation of PBMs and the accuracy, efficiency, and automatic recognition capabilities of DDMs. Hybrid models can balance the advantages and disadvantages of first-principles models and data-driven models and therefore have several advantages over these classes of models, such as higher prediction accuracy, better calibration properties, enhanced extrapolation properties and better interpretability [20]. Hybrid models can be designed in many different ways to exploit the advantages of these models. In [21], a deep neural network is used to estimate the process parameters utilized in a first-principle model. In [22] correction terms are inferred from the data and added to a model structure consisting of first-principle knowledge and the learned correction terms. The Physics-Informed Neural Network (PINN) utilizes the known first-principles knowledge expressed in Partial Differential Equations (PDEs) and their corresponding boundary conditions to regularize a neural network that is trained to approximate the solution of the PDE [23]. The authors of [24] introduce a method called physics-guided machine learning (PGML). The training of the Deep Neural Network (DNN) is augmented with simplified theories relevant to the system's dynamics. Instead of passing these features as input to the network, they are concatenated with the hidden layers of the network, avoiding loss of information in earlier layers. A third method, Sparse Identification of Nonlinear Dynamics (SINDy) [25], which is inspired by compressed sensing, seeks to express the physics of a sampled process as a sparse combination of candidate functions chosen from a large dictionary. This has inspired other methods that search for sparse solutions ([26, 27]), and other works optimize this search by trying to discover symmetries in the data [28]. Deep symbolic regression methods, as presented in literature such as [29] and [30], involve treating a neural network as an expression tree and directly optimizing it to derive a closed-form equation. These approaches utilize different activation functions at each layer of the network to represent a library of permissible functions. See ([31, 32, 33, 34]) for more in-depth reviews of this field. While showing some success, many HAM approaches suffer from issues

such as increased computational cost for training and inference, difficult training convergence, and overfitting. The Corrective Source Term Approach (CoSTA) is a simple yet general and effective method that can combine a PBM and a DDM by summing the predictions from both models. The DDM is then typically trained on the residual of the data that the PBM did not capture. In [35] theoretically justified that CoSTA can correct for a variety of modeling errors. CoSTA introduces model interpretability and simplifies the DDM learning problem by canceling out known physics using a PBM.

1.2.2 Compressed sensing

The Shanon-Nyquist sampling criterion demands a sampling frequency of at least twice the fastest frequency component in a signal in order to accurately estimate the signal and avoid aliasing [36]. This criterion is related to some of the major challenges in the estimation, control, and data sampling for learning in the aluminum electrolysis process. It is desirable to use advanced process control systems in the primary production of aluminum, such as non-linear model predictive control (NMPC) to optimize the operation [37]. But the performance of the NMPC is dependent on accurate state estimation, something that is a particularly difficult task in aluminum electrolysis due to low sampling rates, low observability, and difficult modeling conditions [38]. Moreover, the low sampling rates limit information in the data and make it difficult to learn the process dynamics from sampled data.

As discussed above, the cost of sampling is way to high to solve these problems by increasing the sampling rate to the Nyquist rate. Fortunately, an emerging and gradually more popular signal processing technique called Compressed sensing [39] aims to reconstruct low frequency sampled signals by bypassing the Shannon-Nyquist criterion. Compressed sensing (also referred to as compressive sensing, compressive sampling, and sparse recovery) is a rapidly evolving field of research that aims to estimate high-dimensional signals from low-dimensional measurement vectors. This creates an underdetermined system of linear equations with an infinite number of solutions. However, by exploiting sparsity or compressibility in certain domains, it becomes possible to estimate high-dimensional signal vectors from low-dimensional measurement vectors [40]. Compressed sensing provides a framework that allows for the estimation of signals using far fewer measurements than required by the Nyquist criterion. The strong potential of compressed sensing has stimulated research interest of the topic in a wide range of scientific fields. Examples are in medical imaging such as dynamic MRI [41], biomedical applications such as ECG signals[42] and EEG signals [43], and in communications systems such as wireless sensor networks [44] to mention a few. The field of compressed sensing has given rise to the development of compressive system identific-

ation, which addresses the problem of identifying systems from a limited number of observations. In [45], compressive system identification was used to identify Auto Regressive with eXternal input (ARX) models for Linear Time-Invariant and Linear Time-Variant systems with a large number of inputs and unknown delays. Another method, presented in [46], combines compressed sensing and dynamic mode decomposition to identify reduced-order models on downsampled spatial measurements of high-dimensional systems, and to reconstruct full-state dynamic modes associated with the model, and it was extended to actuated systems. In [47], a method that incorporates physical knowledge into compressed sensing was developed to reduce the volume of data and number of sensors needed for modeling and monitoring the temperature field of an additive manufacturing process. Further research in the field of compressive system identification can be found in [48, 49, 50, 51, 52].

1.2.3 Neural Networks

DNNs are highly flexible DDMs that have the ability to approximate a wide range of complex and high-dimensional relationships and patterns directly from data. The hidden layers of DNNs provide intermittent representations of the input which are reused in later layers. By adding more layers, a DNN can represent functions of increasing complexity. Therefore, DNNs are the standard in NN modeling.

In 1943, neurophysiologist Warren McCulloch and mathematician Walter Pitts suggested how NNs might work by modeling a simple NN using electrical circuits [53]. In 1958, Frank Rosenblatt introduced an implementation of the perceptron [54], allowing for simple neural networks to be trained. In 1969, Marvin Minsky and Seymour Papert pointed out that single perceptrons are not capable of solving the XOR problem [55]. They argued that to solve the XOR problem using perceptrons, they must be stacked in multiple layers. That is, in addition to an input and output layer, at least one hidden layer is needed to solve the XOR problem. The lack of methods to train such multilayer models led to the first major drop in popularity of NNs [56]. In 1974, Paul Werbos suggested that backpropagation could be used to train deep neural networks. Backpropagation allows one to calculate the gradient of a cost function with respect to the individual weights in the network. This is done by utilizing the chain rule, computing the gradient one layer at a time, and iterating backward from the output layer to the input layer. This discovery has been indispensable for many subsequent advances within deep learning (DL). The DL community has since faced major challenges that have slowed research progress and led to a significant decrease in research funding and interest. These periods have been followed by periods of revitalized enthusiasm due to solutions to pressing challenges or the emergence of novel and imaginative contributions. This encompasses an array of highly efficacious DNN architectures

and ways of training them, the exploding increase in computational capability, and the increasing access to vast data.

The Multi-Layer Perceptron (MLP) is a feed-forward NN composed of fully connected layers. The MLP can be seen as the most intuitive and general form of NNs because of its simplicity and applicability to a wide range of tasks. However, different learning problems have different challenges and characteristics. Therefore, introducing inductive bias to the neural network structure can be crucial to achieving acceptable generalization error, or in some cases, being able to solve the problem at all. Convolutional Neural Networks (CNN) are highly effective in tasks like object detection in extremely high dimensional inputs like images. While the MLP would naively look for correlations between all pixels in the image, the CNN uses convolutions and pooling to extract local and sparse features. The utilization of shared weights and local connections within the CNN is employed to fully leverage the 2D input-data structures. This simplifies the learning problem and reduces computational time. Recurrent neural networks (RNNs) are another highly popular DL architecture, able to capture dependencies in sequentially structured data. RNNs can use variable length inputs due to the feedback mechanism that reuses previous inputs. A main issue of training RNNs is their sensitivity to the exploding and vanishing gradient problem [57]. The problem can arise when the reduplications of several large and small derivatives potentially cause the gradients to explode or decay. The long-short-term memory (LSTM) network [58] is a recurrent DNN that addresses the issues of exploding and vanishing gradients by introducing forget-gates in each processing unit so that the aforementioned sensitivity decays over time. These DNN structures illustrate inductive bias is incorporated into DL to achieve highly effective models.

DL is achieving great success in several applications such as computer vision, natural language processing (NLP), medicine, and more [59]. Lately, there has been an increased interest in using NNs to model nonlinear dynamical systems, due to their highly expressive power. Examples are the use of NNs to model the simulated dynamics of a pressurized water nuclear reactor [60], identification of the dynamics of the bioethanol production and purification process [61], prediction of chemical reactions [62], and determination of chlorinated compounds in fish [63]. In the aluminum electrolysis process, DNNs have been applied to predict essential variables that are difficult to measure continuously. In [64], a dense single layer neural network with more than 200 neurons was used to simulate bath chemistry variables in aluminum electrolysis. The paper mainly addresses the training speed of neural networks using an extreme learning machine. In [65] dense neural networks were used to predict variables in the electrolysis cell. The study took into account the changing properties of the electrolysis cells by collecting data over the

course of their life cycle. In [66], dense neural networks with two hidden layers were used to model the properties of the carbon anode.

However, using NNs to model the dynamics of a physical process entails a number of challenges. NNs typically need vast data in order to generalize well. While applications like NLP and computer vision typically have access to enormous datasets, data from physical processes like aluminum electrolysis is typically scarce and contains limited information. Due to the cost of sampling and exciting the dynamics, the potential experimental budget is a limiting factor. Moreover, the safety-critical nature of the aluminum electrolysis is not compatible with standard black-box NNs. NNs also turn out to be unstable in training. That is, NNs with the same model structure trained on the same data but with different parameter initialization converge to models with different prediction capabilities. While NNs have proved to be highly effective, several challenges remains in order to fully exploit their potential in safety-critical systems like the aluminum electrolysis.

1.2.4 Active Learning

Active Learning (AL) aims to maximize model accuracy with a minimal amount of data by acquiring the most informative training data [67]. AL has been utilized in many fields, including image recognition [68], text classification [69], and object detection [70], to name a few. In these scenarios, large unlabeled datasets are available, and the AL algorithm aims to choose the most informative samples among the unlabeled data. The goal is then to reduce the costly labeling process which is performed by human domain experts. In the context of dynamical systems, labeling output data may not incur significant expenses. However, data obtained from a processes under closed-loop feedback control often lacks the necessary information to identify NN models with acceptable performance, including accuracy and generalizability to operational regions of the input space, known as the state-action space. Obtaining informative datasets of dynamical processes is costly due to for example interruption of the production or operation, and expenses of measuring certain states. Moreover, exciting the dynamical systems to regions with high model uncertainty can induce unforeseen and potentially severe incidents to the process and its surroundings. While the safety-critical nature of using NNs in controlled processes is a major research challenge itself, approached by for example techniques within reachability analysis [71, 72], we limit the scope of the work presented in Chapter 7 to the informativeness of the sampled data, addressing challenges related to costs of sampling large datasets in dynamical systems.

Introducing AL methods to experimental design for system identification introduces additional challenges to the AL problem. That is, most AL methods address static acquisition problems where, in principle, any location in the input space is

directly accessible, or a dataset is sampled in advance. For dynamical systems, on the other hand, reaching a desired location in the state-action space requires system excitation through control inputs. The topic of optimal excitation has been addressed in the research field known as optimal experiment design [73]. In light of this, optimal experiment design can be seen as a subfield of AL, or they can be seen as related research topics. Anyways, AL, which originates from the computer science community provides a wide range of information-theoretic approaches as well as a well-defined learning framework, providing great inspiration to researchers working with system identification.

Authors of [74] propose AL strategies for the identification of a Gaussian Process (GP) model inspired by information theoretic measures. The most promising work they propose suggests optimizing a sequence of control inputs that maximize the predictive differential entropy along the state trajectory; a method outperforming state-of-the-art experimental design methods. The work of identifying GP models was extended in [75], to also include global explorations. The global search for initial states is done by exploring the informativeness of short trajectories from candidate initial states. When an informative initial state is acquired, the local exploration maximizes the predictive entropy along the state trajectory as in [74]. AL is also applied to acquire data that efficiently identify linear models by solving an Optimal Control Problem (OCP) that maximize the minimal eigenvalues of covariates of states [76]. An active learning approach to identify a restricted class of nonlinear dynamical modes whose state transitions depend linearly on a known feature embedding of state-action pairs was investigated in [77]. However, a well-defined formulation to utilize AL in acquiring data for identifying DNN models of dynamical systems is yet to be defined.

1.3 Outline and contributions

1.3.1 Preliminaries (Chapter 2)

This chapter presents the preliminary theory used throughout the thesis. This includes simulation models used in the case studies as well as fundamental Machine Learning (ML) theory utilized in different contributions presented in the thesis. Two performance metrics used throughout the thesis are defined at the end of the chapter.

1.3.2 Hybrid modeling combining first principle model and compressed sensing (Chapter 3)

[78] **E. T. B. Lundby**, A. Rasheed, I. J. Halvorsen, J. T. Gravdahl, "A novel hybrid analysis and modeling approach applied to aluminum electrolysis process". In:

Journal of Process Control 105 (2021), pp. 62–77. ISSN: 0959-1524.

Aluminum electrolysis cells are subject to severe conditions that necessitate multiple manual measurements. The costs associated with manual sampling significantly impact operational expenses, resulting in a low frequency of measurements. Due to the highly complex and interrelated nature of the aluminum electrolysis, state-of-the-art PBMs, despite building on excellent system knowledge, are insufficient to accurately express the full physics in the cells. The combination of inadequate prediction models and low sampling rates makes state estimation in the aluminum electrolysis highly challenging, which degrades the potential of optimal process operation.

This chapter presents a novel hybrid modeling approach combining a PBM with the powerful signal estimation technique called compressed sensing to estimate a stationary disturbance from a signal measured at low sampling rates. Compressed sensing requires that, in order to estimate a coarsely sampled signal, the signal of interest must be sparse in some domain. Starting with a non-sparse coarsly sampled signal, a PBM subtract the known physics from the measured signal. This manipulation produces a much sparser residual signal representing the unmodeled dynamics in the measured signal. Since the residual signal can be sparsely represented in some transformed domain, compressed sensing can be utilized to estimate the unmodeled dynamics from low frequency measurements. The low frequency, manipulated signal is used as inputs to a compressed sensing algorithm, producing a high-fidelity estimate of the unmodeled dynamics in the originally measured signal. The novelty in the work is two-fold. First, compressed sensing is introduced to the field of aluminum electrolysis. Second, the novel hybrid modeling approach enabling compressed sensing to estimate the unmodeled dynamics of a measured non-sparse signal presented. To illustrate how the novel hybrid model can be utilized in state estimation, the estimate of the unmodeled signal is integrated in an Extended Kalman Filter (EKF) to increas accuracy of the state estimation.

1.3.3 Hybrid modeling combining first principle model and deep learning (Chapter 4)

[79] H. Robinson[1], **E. T. B. Lundby**[1], A. Rasheed, J. T. Gravdahl, "A novel corrective-source term approach to modeling unknown physics in aluminum extraction process" Conditional Acceptance: Engineering Applications of Artificial Intelligence. 2023, Available in: arXiv preprint arXiv:2209.10861 (2022).

State-of-the-art process models of the aluminum electrolysis are mostly derived from first principles yielding PBMs. The accuracy of these models often suffer

[1] Equal contributions

due to partly understanding of the process, numerous modeling assumptions, and uncertainty in the model coefficients. NNs are highly flexible DDMs with the capability to model complex input-output relations directly from data. However, NNs typically requires vast data to reach desirable model accuracy and generalizability. Moreover, the complex structure of NNs indicate that these models suffer from low interpretability.

In this chapter, we present a novel approach to combine the best of both physics-based and data-driven modeling approaches while eliminating some of their weaknesses. That is, a NN is used to correct a misspecified PBM of the Hall–Héroult process by using a corrective source term added to the set of governing ODEs. Residual data, meaning data obtained from subtracting the PBM estimates of states from measurements of the states is used as training data for the NNs. Utilizing PBMs in this manner reduces complexity of the learning problem for the NNs, which again reduces the need for data, while maintaining a high level of interpretability in the PBM. We compare this approach with an end-to-end learning approach and an ablated physics-based model, and show that the proposed hybrid method is more accurate, consistent, and stable for long-term predictions.

1.3.4 Modeling dynamics using sparse neural networks (Chapter 5)

[80] **E. T. B. Lundby**, A. Rasheed, I. J. Halvorsen, J. T. Gravdahl "Sparse deep neural networks for modeling aluminum electrolysis dynamics". In: Applied Soft Computing 134 (2023), p. 109989. ISSN: 1568-4946.

The remarkable capacity of DNNs to fit arbitrary nonlinear functions from data with minimal expert intervention has made them a widely popular choice for modeling complex nonlinear processes. However, they are almost always overparameterized and challenging to interpret due to their internal complexity. In addition, the optimization process to determine the learned model parameters can be unstable, as it may get trapped in bad local minima.

This chapter showcases the effectiveness of sparse regularization techniques in substantially decreasing model complexity. We illustrate this by applying these techniques to the aluminum extraction process, which is a highly nonlinear system comprising numerous interrelated subprocesses. We train a DNN with ℓ_1 regularization, which promotes sparsity, and then evaluate the impact of this regularization on generalizability, interpretability, and training stability in comparison to a densely connected DNN. Our findings reveal that sparse regularization significantly diminishes model complexity compared to a corresponding densely connected neural network. We contend that this renders the model more interpretable, and demonstrate that training an ensemble of sparse DNNs with varying parameter

initializations frequently converges to similar model structures with comparable learned input features and similar prediction capabilities. Moreover, the empirical study demonstrates that the resulting sparse models exhibit superior generalization from small training sets in comparison to their dense counterparts.

1.3.5 Modeling dynamics using Neural Networks with skip-connections (Chapter 6)

[81] **E. T. B. Lundby**[1], H. Robinson[1], A. Rasheed, J. T. Gravdahl, "Sparse neural networks with skip-connections for identification of aluminum electrolysis cell" Pending Review: IEEE CDC. (2023) Available in: arXiv preprint arXiv:2301.00582 (2023)

The limited access to informative data in the aluminum electrolysis combined with the data-hungry nature of NNs possible put restrictions on the utility value of such models as process models in aluminum extraction. However, the NN community continues to conduct quantum leaps within NN methodology, improving NNs in different applications. Huang et. al. [82] proposed DenseNet, connecting each layer to all consecutive layers through concatenated skip-connections, obtaining significant improvements in object recognition tasks performed by a convolutional neural network.

Building on this work, this chapter introduces modifications of of a feed-forward NN structure in the form of skip-connections from intermediate layers to the output layer for modeling a set of ODEs representing the aluminum electrolysis. Moreover, we combine the modified NN structure with sparsity promoting ℓ_1 regularization. We demonstrate that the combination of skip-connections and ℓ_1 regularization can greatly improve both the accuracy and the stability of the models for datasets of varying sizes.

1.3.6 Deep active learning in experimental design for nonlinear system identification (Chapter 7)

[83] **E. T. B. Lundby**, A. Rasheed, I. J. Halvorsen, D. Reinhardt, S. Gros, J. T. Gravdahl, "Deep active learning for nonlinear system identification", In arXiv preprint arXiv:2302.12667 (2023)

DNNs have great potential in modeling dynamical systems directly from data. However, this typically requires access to vast data spanning the regions of the input space in which the system will operate. Data obtained from a process under closed-loop feedback control typically contains little variation, and thus the information content of the data is often limited. Training DNNs using the data

[1] Equal contributions

from a closed-loop process to identify nonlinear system dynamics will therefore typically lead to models with bad generalization properties. Obtaining vast and informative data from a process with state-of-the-art experimental design methods typically involves large additional costs.

This chapter proposes an optimal experimental design method that aims to efficiently sample informative data from a dynamical system. The proposed method is a deep active learning framework that iterative obtains informative data that can be used to identify DNNs. The framework uses local explorations to obtain a set of simulated candidate state-action trajectories to be evaluated by a global exploration. The global exploration is a novel batch acquisition optimization problem formulation for dynamical systems used to choose the most informative areas of the input space of a dynamical system known as the state-action space. Results show that the novel framework outperforms state-of-the-art random-based excitation methods.

1.3.7 Other contribitions

During my PhD, I supervised Emil J. Haugstvedt - a master's student at Engineering Cybernetics in his pre-project. In this work he investigated different sparsity promoting pruning techniques for DNNs, and their effect on model accuracy and generalization capability of the resulting DNNs that were used to model the high-dimensional dynamical system described in Section 2.1.2 . The comparative study includes hard and soft thresholding, pruning and regrowing, and ℓ_1-regularization. The study concludes that the ℓ_1 regularization is the most crucial factor for the model performance compared to other techniques. Moreover, it was found that combining ℓ_1 regularization with other pruning techniques could both improve and impair model performance over both ℓ_1 regualrization and the additional technique alone, depending on the additional pruning technique. An article named "A comparative study of sparsity promoting techniques in neural network for modeling non-linear dynamics" that describes the work is under review at IEEE access.

Chapter 2

Preliminaries

2.1 Simulation models

2.1.1 Simple aluminum electrolysis model

This section describes a simple simulator of the aluminum electrolysis cell representing the mass balance in the cell. The simulator is used in the case study in Chapter 3.

Aluminum is extracted in the Hall-Héroult process, which involves dissolving alumina (Al_2O_3) into an electrolytic bath mainly consisting of cryolite (Na_3AlF_6). An electric current is sent through the cell, and aluminum ions in the electrolyte are reduced to aluminum. The aluminum reduction cell consists of basic components of an electrolysis cell such as anode, cathode, and electrolyte. One or several carbon anodes are immersed into the electrolyte, also known as the bath. This is illustrated in Figure 2.1. In this figure, the main components of an aluminum cell and a rough sketch of its design is presented. The carbon anodes are consumed in the process. This is expressed in the overall reaction of aluminum production in the Hall-Héroult process:

$$2Al_2O_3 + 3C \rightarrow 3CO_2(g) + 4Al(l), \tag{2.1}$$

where C is the carbon in the reaction from the anode, CO_2 is carbon dioxide, and Al is aluminum. The g indicates gas and the l indicates liquid. The molten aluminum is considered as the cathode in the reaction.

On the sidewalls of the cell, a ledge of frozen electrolyte called side ledge is formed. The side ledge works as a thermal insulator but most importantly as a protecting layer preventing the sidewall from corroding.

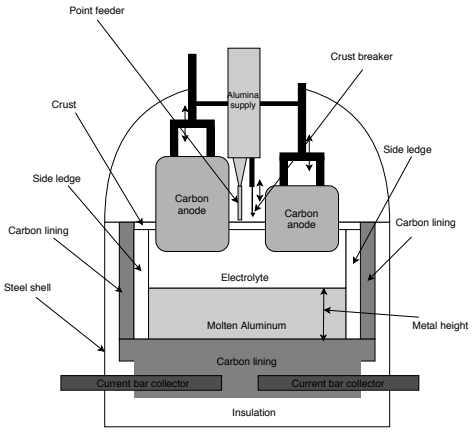


Figure 2.1: Conceptual drawing of an aluminum electrolysis cell.

The mass balance of the aluminum cell was derived in [84]. Let m be the mass of aluminum in the cell, such that:

$$\frac{dm}{dt} = \dot{m}_{gen}(t) - \dot{m}_{out}(t), \tag{2.2}$$

where $\dot{m}_{gen}(t)$ is the mass rate of aluminum generated at time instant t, and $\dot{m}_{out}(t)$ is mass rate of aluminum taken out of the cell. This can be approximated to a discrete model:

$$\Delta m(k) = \Delta t(\dot{m}_{gen}(k) - \dot{m}_{out}(k)), \tag{2.3}$$

where $k = \{1, 2, 3, ..\}$ represent discrete time instants and Δt is the sampling time for when the model is updated. The mass of aluminum generated per time unit is given by Faraday's law:

$$\dot{m}_{gen}(k) = \frac{\Delta Q(k) \cdot CE \cdot M_{Al}}{F \cdot z \cdot \Delta t}, \tag{2.4}$$

where CE is the current efficiency, $M_{Al} = 26.98g/mol$ is the molecular mass of aluminum, $F = 96485.3329C/mol$ is Faraday's constant and $z = 3$ is the number of electrones involved the reaction generating one aluminum atom. The charge ΔQ transferred from time instant $k-1$ to k is calculated by the time integral of the current I given by:

$$\Delta Q = \int_{k-1}^{k} I(\tau) d\tau. \tag{2.5}$$

Assuming that the current I is constant within a time interval Δt, generated mass rate can be formulated as:

$$\dot{m}_{gen}(k) = \frac{CE \cdot M_{Al}}{F \cdot z} I(k). \tag{2.6}$$

Furthermore, assuming constant current efficiency during a time interval Δt, the mass generated during a time interval Δt can be formulated as

$$m_{gen}(k) = \dot{m}_{gen}(k)\Delta t \tag{2.7}$$

The mass flow rate out of the cell relates to metal tapping. Assuming constant flow rate during the tapping yields:

$$m_{out}(k) = \Delta t \cdot \dot{m}_{out}(k) \tag{2.8}$$

The metal height h_m is a function of the average cross-section area of the cell and the volume of molten aluminum in the cell. It is assumed that the cross-section area is uniform. This follows from the assumption of horizontal side walls and uniform side ledge thickness, see Figure 2.2. The metal height can be formulated as:

$$h_m = \frac{V_{Al}(m_{Al}, \rho_{Al})}{A_{Al}(l, w, x_{sl})}, \tag{2.9}$$

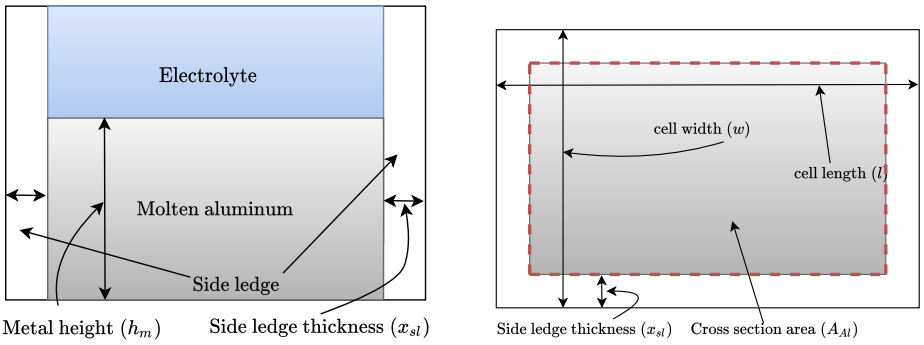
where V_{Al} is the volume of molten alumninum, m_{Al} is the mass of molten aluminum in the cell and $\rho_{Al} = 2200kg/m^3$ is the density of molten aluminum. A_{Al} is the cross section area where the cell is occupied by aluminum, l is the cell length, w is the cell width and x_{sl} is the side ledge thickness. V_{Al} is given by:

$$V_{Al} = \frac{m_{Al}}{\rho_{Al}}. \tag{2.10}$$

Figure 2.2b show a snap shot of the cross section of the electrolytic according to the simulation. Figure 2.2a illustrates that the side ledge thickness is uniform. These drawings illustrates that the uniform cross section area of aluminum is given by:

$$A_{Al} = (l - 2x_{sl}) \cdot (b - 2x_{sl}) \tag{2.11}$$

The thickness of the frozen electrolyte known as side ledge x_{sl} is determined by a complex interaction between heat balance, amperage, bath composition, cell resistance, movements in the bath induced by magnetism, bubbles, and more ([85]). Therefore, this variable is challenging to estimate. As stated above, this variable relates to the cross-section area A_{Al} and, consequently, to the metal height h_m. Therefore, estimating the metal height with compressed sensing techniques can reveal information about the side ledge thickness. The side ledge is crucial in preventing corrosion of the sidewalls. Thus this information is of great value. The side ledge thickness, which represents unmodeled dynamics in the cell, is simulated as a sum of cosine waves.



(a) Aluminum cell from the side

(b) Aluminum cell from above

Figure 2.2: Geometry of simulated aluminum cell. The side walls are assumed horizontal and the side ledge thickness is assumed uniform. In real aluminum cells, the bottom of the side walls are in general sloping walls. Furthermore, the side ledge thickness is in general not uniform.

2.1.2 Complex aluminum electrolysis model

This section presents the simulation model used in the case studies in Chapter 4, 5 and 5. The complete ODEs are presented, while derivations of-, and additional information about the simulator is found in Appendix A. An overview of the physical plant is shown in Figure 2.3. A PBM of the plant can be derived from the mass/energy balance of the system. We omit this step and present the model directly. The internal dynamics of the aluminum electrolysis cell are described by a set of Ordinary Differential ODEs, with the general form:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{2.12}$$

where $\mathbf{x} \in \mathbb{R}^8$ is the state vector, $\mathbf{u} \in \mathbb{R}^5$ are external inputs, and $f(\mathbf{x}, \mathbf{u})$ describes the nonlinear dynamics. Table 2.1 shows the names of the internal states and external inputs. The intrinsic properties of the Al_2O_3+AlF_3+Na_3 AlF_6 mixture are determined by the mass ratios of x_2 (Al_2O_3) and x_3 (AlF_3), written as:

$$\begin{aligned} c_{x_2} &= x_2/(x_2 + x_3 + x_4) \\ c_{x_3} &= x_3/(x_2 + x_3 + x_4) \end{aligned} \tag{2.13}$$

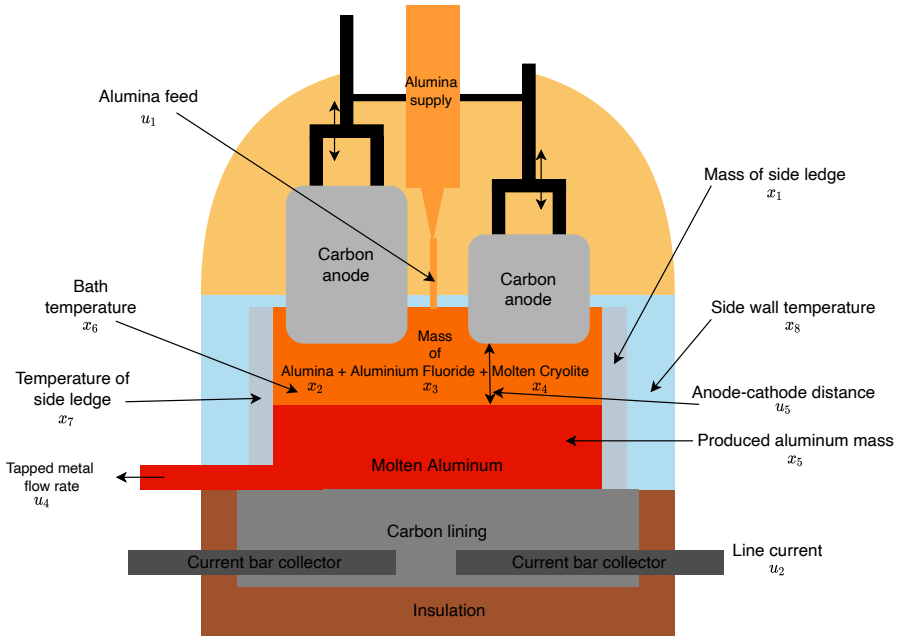


Figure 2.3: Schematic of the setup.

We then define the following quantities:

$$g_1 = 991.2 + 112c_{x_3} + 61c_{x_3}^{1.5} - 3265.5c_{x_3}^{2.2} - \frac{793c_{x_2}}{-23c_{x_2}c_{x_3} - 17c_{x_3}^2 + 9.36c_{x_3} + 1} \tag{2.14a}$$

$$g_2 = \exp\left(2.496 - \frac{2068.4}{273 + x_6} - 2.07c_{x_2}\right) \tag{2.14b}$$

$$g_3 = 0.531 + 3.06 \cdot 10^{-18}u_1^3 - 2.51 \cdot 10^{-12}u_1^2 + 6.96 \cdot 10^{-7}u_1 - \frac{14.37(c_{x_2} - c_{x2,crit}) - 0.431}{735.3(c_{x_2} - c_{x2,crit}) + 1} \tag{2.14c}$$

$$g_4 = \frac{0.5517 + 3.8168 \cdot 10^{-6}u_2}{1 + 8.271 \cdot 10^{-6}u_2} \tag{2.14d}$$

$$g_5 = \frac{3.8168 \cdot 10^{-6}g_3g_4u_2}{g_2(1 - g_3)} \tag{2.14e}$$

g_1 is the liquidus temperature T_{liq} defined in Equation (A.59), g_2 is the electrical conductivity κ defined in Equation (A.33), g_3 is the bubble coverage ϕ defined in Equation (A.37), g_4 is the bubble thickness d_{bub} defined in Equation (A.36)), and g_5 is the bubble voltage drop U_{bub} defined in Equation (A.35). The expressions and

Table 2.1: Table of states and inputs

Variable	Physical meaning	Unit	Variable	Physical meaning	Unit
x_1	mass side ledge	kg	x_2	mass Al_2O_3	kg
x_3	mass AlF_3	kg	x_4	mass $Na_3\,AlF_6$	kg
x_5	mass metal	kg	x_6	temperature bath	$^\circ C$
x_7	temperature side ledge	$^\circ C$	x_8	temperature wall	$^\circ C$
u_1	Al_2O_3 feed	kg/s	u_2	Line current	kA
u_3	AlF_3 feed	kg/s	u_4	Metal tapping	kg/s
u_5	Anode-cathode distance	cm			

coefficients of the physical properties in Equation (2.14) are taken from different academic works that have estimated these quantities. These academic works are cited in Appendix A.

The full PBM can now be written as a set of 8 ODEs:

$$\dot{x}_1 = \frac{k_1(g_1 - x_7)}{x_1 k_0} - k_2(x_6 - g_1) \tag{2.15a}$$

$$\dot{x}_2 = u_1 - k_3 u_2 \tag{2.15b}$$

$$\dot{x}_3 = u_3 - k_4 u_1 \tag{2.15c}$$

$$\dot{x}_4 = -\frac{k_1(g_1 - x_7)}{x_1 k_0} + k_2(x_6 - g_1) + k_5 u_1 \tag{2.15d}$$

$$\dot{x}_5 = k_6 u_2 - u_4 \tag{2.15e}$$

$$\dot{x}_6 = \frac{\alpha}{x_2 + x_3 + x_4}\left[u_2 g_5 + \frac{u_2^2 u_5}{2620 g_2} - k_7(x_6 - g_1)^2 + k_8\frac{(x_6 - g_1)(g_1 - x_7)}{k_0 x_1} - k_9\frac{x_6 - x_7}{k_{10} + k_{11}k_0 x_1}\right] \tag{2.15f}$$

$$\dot{x}_7 = \frac{\beta}{x_1}\left[\frac{k_9(g_1 - x_7)}{k_{15}k_0 x_1} - k_{12}(x_6 - g_1)(g_1 - x_7) + \frac{k_{13}(g_1 - x_7)^2}{k_0 x_1} - \frac{x_7 - x_8}{k_{14} + k_{15}k_0 x_1}\right] \tag{2.15g}$$

$$\dot{x}_8 = k_{17}k_9\left(\frac{x_7 - x_8}{k_{14} + k_{15}k_0 \cdot x_1} - \frac{x_8 - k_{16}}{k_{14} + k_{18}}\right) \tag{2.15h}$$

The constants (k_0, .., k_{18}, α, β) in Equation (2.15) are described and given numerical values in Table 2.2.

Table 2.2: Constants in the simulator

Constant	Physical meaning	Numeric value	Unit
k_0	$1/(\rho_{sl}A_{sl})$	$2 \cdot 10^{-5}$	m/kg
k_1	$2k_{sl}A_{sl}/\Delta_{fus}H_{cry}$	$7.5 \cdot 10^{-4}$	$(\mathrm{kg \cdot m})/(^\circ\mathrm{C \cdot s})$
k_2	$h_{bath-sl}A_{sl}/\Delta_{fus}H_{cry}$	0.18	$\mathrm{kg}/(^\circ\mathrm{C \cdot s})$
k_3	$0.002\frac{M_{Al_2O_3}\cdot CE}{z\cdot F}$	$1.7 \cdot 10^{-7}$	$\mathrm{kg}/(\mathrm{A \cdot s})$
k_4	$C_{Na_2O}\frac{4M_{AlF_3}}{3M_{Na_2O}}$	0.036	–
k_5	$C_{Na_2O}\frac{2M_{cry}}{3M_{Na_2O}}$	0.03	–
k_6	$0.002\frac{M_{Al}\cdot CE}{z\cdot F}$	$4.43 \cdot 10^{-8}$	$\mathrm{kg}/(\mathrm{A \cdot s})$
k_7	$k_2 \cdot c_{p_{cry,\,liq}}$	338	$\mathrm{J}/(^\circ\mathrm{C}^2\cdot\mathrm{s})$
k_8	$k_1 \cdot c_{p_{cry,\,liq}}$	1.41	$(\mathrm{J \cdot m})/(^\circ\mathrm{C}^2\cdot\mathrm{s})$
k_9	A_{sl}	17.92	m^2
k_{10}	$1/h_{bath-sl}$	0.00083	$(\mathrm{s \cdot m}^2\cdot^\circ\mathrm{C})/\mathrm{J}$
k_{11}	$1/(2k_{sl})$	0.2	$(\mathrm{s \cdot m}\cdot^\circ\mathrm{C})/\mathrm{J}$
k_{12}	$k_2 \cdot c_{p_{cry,\,s}}$	237.5	$\mathrm{J}/(^\circ\mathrm{C}^2\cdot\mathrm{s})$
k_{13}	$k_1 \cdot c_{p_{cry,\,s}}$	0.99	$(\mathrm{J \cdot m})/(^\circ\mathrm{C}^2\cdot\mathrm{s})$
k_{14}	$x_{wall}/(2k_{wall})$	0.0077	$(\mathrm{m}^2\cdot\mathrm{s}\cdot^\circ\mathrm{C})/\mathrm{J}$
k_{15}	$1/(2k_{sl})$	0.2	$(\mathrm{m \cdot s}\cdot^\circ\mathrm{C})/\mathrm{J}$
k_{16}	T_0	35	$^\circ\mathrm{C}$
k_{17}	$1/(m_{wall}c_{p,\,wall})$	$5.8 \cdot 10^{-7}$	$^\circ/\mathrm{J}$
k_{18}	$1/h_{wall-0}$	0.04	$(\mathrm{s \cdot m}^2\cdot^\circ\mathrm{C})/\mathrm{J}$
α	$1/c_{p_{bath,\,liq}}$	$5.66 \cdot 10^{-4}$	$(^\circ\mathrm{C \cdot kg})/\mathrm{J}$
β	$1/c_{p_{cry,\,sol}}$	$7.58 \cdot 10^{-4}$	$(^\circ\mathrm{C \cdot kg})/\mathrm{J}$
$c_{x_{2,crit}}$		0.022	–

2.1.3 Simulation model of surface vessel

We use the standard 3-Degrees of Freedom (3-DOF) model of a marine craft, which is a simplified model of a real vessel. The state of the vessel is described by the pose vector $\boldsymbol{\eta} = [x\, y\, \psi]^\top \in \mathbb{R}^3$ and the velocity vector $\boldsymbol{\nu} = [v_1\, v_2\, r]^\top \in \mathbb{R}^3$. The pose vector describes the position and orientation of the vessel in the North-East-Down (NED) frame with x and y being the position in the North and East directions, respectively, and ψ being the heading angle. The velocity vector describes the velocity of the vessel in the body frame with v_1 and v_2 being the velocity in the surge and sway directions, respectively, and r being the yaw rate. Figure 2.4 illustrates the surface vessel:

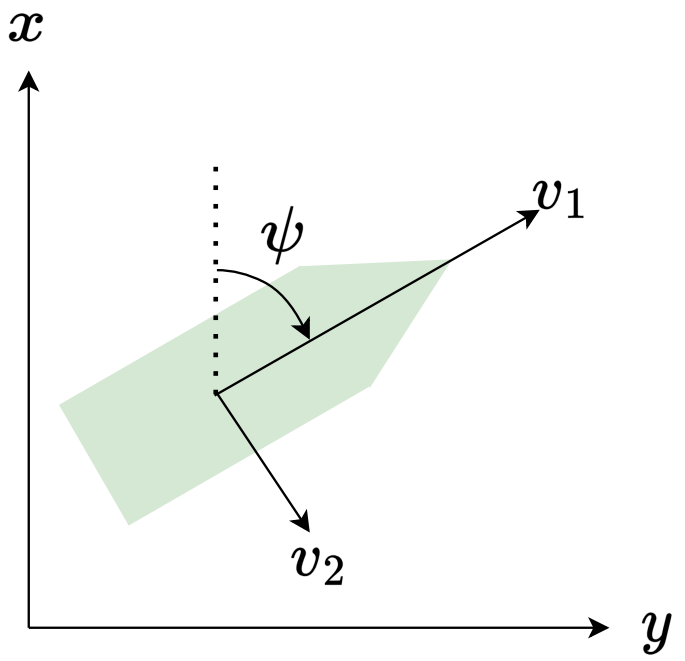


Figure 2.4: 3-DOF surface vessel in the NED frame.

The model is formulated as a nonlinear system of ODEs as follows (see [86] for details):

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{R}(\psi)\boldsymbol{\nu} \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau}. \end{aligned} \tag{2.16}$$

where $\mathbf{R}(\psi) \in \mathrm{SO}(3)$ is the rotation matrix from the body frame to the NED frame. The mass matrix $\mathbf{M} \in \mathbb{R}^{3\times3}$, the Coriolis matrix $\mathbf{C}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$, and the damping matrix $\mathbf{D}(\boldsymbol{\nu}) \in \mathbb{R}^{3\times3}$ are given by [87] and express the inertia and Coriolis matrices as:

$$\mathbf{M} = \begin{bmatrix} m_{11} & 0 & 0 \\ 0 & m_{22} & m_{23} \\ 0 & m_{32} & m_{33} \end{bmatrix}, \quad \mathbf{C}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & 0 & c_{13}(\boldsymbol{\nu}) \\ 0 & 0 & c_{23}(\boldsymbol{\nu}) \\ c_{31}(\boldsymbol{\nu}) & c_{32}(\boldsymbol{\nu}) & 0 \end{bmatrix},$$
$$\mathbf{D}(\boldsymbol{\nu}) = \begin{bmatrix} d_{11}(\boldsymbol{\nu}) & 0 & 0 \\ 0 & d_{22}(\boldsymbol{\nu}) & d_{23}(\boldsymbol{\nu}) \\ 0 & d_{32}(\boldsymbol{\nu}) & d_{33}(\boldsymbol{\nu}) \end{bmatrix} \tag{2.17}$$

where the elements $c_{ij}(\boldsymbol{\nu})$ and $d_{ij}(\boldsymbol{\nu})$ are given by

$$
\begin{aligned}
c_{13}(\boldsymbol{\nu}) &= -m_{22}v_2 - m_{23}r \\
c_{23}(\boldsymbol{\nu}) &= m_{11}v_1 \\
c_{31}(\boldsymbol{\nu}) &= -c_{13}(\boldsymbol{\nu}) \\
c_{32}(\boldsymbol{\nu}) &= -c_{23}(\boldsymbol{\nu})
\end{aligned}
\qquad
\begin{aligned}
d_{11}(\boldsymbol{\nu}) &= -X_{v_1} - X_{|v_1|v_1}|v_1| - X_{v_1v_1v_1}v_1^2 \\
d_{22}(\boldsymbol{\nu}) &= -Y_{v_2} - Y_{|v_2|v_2}|v_2| - Y_{|r|v_2}|r| - Y_{v_2v_2v_2}v_2^2 \\
d_{23}(\boldsymbol{\nu}) &= -Y_r - Y_{|v_2|r}|v_2| - Y_{|r|r}|r| \\
d_{32}(\boldsymbol{\nu}) &= -N_{v_2} - N_{|v_2|v_2}|v_2| - N_{|r|v_2}|r| \\
d_{33}(\boldsymbol{\nu}) &= -N_r - N_{|v_2|r}|v_2| - N_{|r|r}|r| - N_{rrr}r^2
\end{aligned}
\tag{2.18}
$$

The constant coefficients in (2.18) are summarized in Table 2.3. The kinematics of the vessel is given by

$$
\dot{\boldsymbol{\eta}} = \mathbf{R}(\psi)\mathbf{v} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ r \end{bmatrix}, \tag{2.19}
$$

and its dynamics are governed by

$$
\dot{\boldsymbol{\nu}} = \mathbf{M}^{-1}\left(\boldsymbol{\tau} - \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} - \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu}\right) \tag{2.20}
$$

We are interested in learning the dynamics of the vessel in response to given forces and moments, which we consider as control input. This simulation model is utilized in the case study in Chapter 7, where the following notation is used:

$$
\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \quad \text{with} \quad \mathbf{x} = \boldsymbol{\nu} \quad \text{and} \quad \mathbf{u} = \boldsymbol{\tau}, \tag{2.21}
$$

i.e. $\mathbf{x} \in \mathbb{R}^3$ is the state vector and $\mathbf{u} \in \mathbb{R}^3$ is the control input, with dynamics described by (2.20).

Table 2.3: Values and units of the parameters of the vessel.

Constant	Value	Unit	Constant	Value	Unit
m_{11}	2389.657	kg	m_{22}	2533.911	kg
m_{23}	62.386	kg	m_{32}	28.141	kg
m_{33}	5068.910	$\mathrm{kg \cdot m^2}$	X_{v_1}	-27.632	$\mathrm{kg \cdot} s^{-1}$
$X_{\lvert v_1\rvert v_1}$	-110.064	$\mathrm{kg \cdot s^{-1}}$	$X_{v_1v_1v_1}$	-13.965	$\mathrm{kg \cdot s^{-1}}$
Y_{v_2}	-52.947	$\mathrm{kg \cdot s^{-1}}$	$Y_{\lvert v_2\rvert v_2}$	-116.486	$\mathrm{kg \cdot s^{-1}}$
$Y_{v_2v_2v_2}$	-24.313	$\mathrm{kg \cdot s^{-1}}$	$Y_{\lvert r\rvert v_2}$	-1540.383	$\mathrm{kg \cdot s^{-1}}$
Y_r	24.732	$\mathrm{kg \cdot s^{-1}}$	$Y_{\lvert v_2\rvert r}$	572.141	$\mathrm{kg \cdot s^{-1}}$
$Y_{\lvert r\rvert r}$	-115.457	$\mathrm{kg \cdot s^{-1}}$	N_{v_2}	3.5241	$\mathrm{kg \cdot s^{-1}}$
$N_{\lvert v_2\rvert v_2}$	-0.832	$\mathrm{kg \cdot s^{-1}}$	$N_{\lvert r\rvert v_2}$	336.827	$\mathrm{kg \cdot s^{-1}}$
N_r	-122.860	$\mathrm{kg \cdot s^{-1}}$	$N_{\lvert r\rvert r}$	-874.428	$\mathrm{kg \cdot s^{-1}}$
N_{rrr}	0.000	$\mathrm{kg \cdot s^{-1}}$	$N_{\lvert v_2\rvert r}$	-121.957	$\mathrm{kg \cdot s^{-1}}$

2.2 Neural networks

A DNN is a supervised machine learning algorithm [56] that can be denoted by

$$\mathbf{y} = \hat{\mathbf{f}}(\mathbf{x}; \boldsymbol{\theta}), \tag{2.22}$$

where $\mathbf{y} \in \mathbb{R}^s$ is the output vector of the network model and s its length. $\mathbf{x} \in \mathbb{R}^d$ is the input vector to the network model, and d is the input dimension. Here, $\boldsymbol{\theta} \in \mathbb{R}^p$ denotes all trainable parameters in the network model where p is the number of the parameters. Each layer *j+1* operates on the output vector from the previous layer $\mathbf{Z}^j \in R^{L_j}$ and outputs a vector $\mathbf{Z}^{j+1} \in \mathbb{R}^{L_{j+1}}$:

$$\mathbf{Z}^{j+1} = \sigma(\mathbf{W}^{j+1}\mathbf{Z}^j + \mathbf{b}^{j+1}). \tag{2.23}$$

$\mathbf{W}^{j+1} \in \mathbb{R}^{L_{j+1} \times L_j}$ is called the weight matrix, $\mathbf{b}^{j+1} \in \mathbb{R}^{L_{j+1}}$ is the called the bias vector of layer $j+1$, $\boldsymbol{\theta} = \{\boldsymbol{\theta}^{\mathbf{1}},\ ...,\ \boldsymbol{\theta}^{j+1},\ ...\}$, and $\boldsymbol{\theta}^{j+1} = \{\mathbf{W}^{j+1},\ \mathbf{b}^{j+1}\}$. σ is a non-linear activation function. That is,

$$\begin{aligned}\sigma(\mathbf{W}^{j+1}\mathbf{Z}^j + \mathbf{b}^{j+1}) = (\sigma(W_1^{j+1}\mathbf{Z}^j + b_1^{j+1}),\ ...,\\ \sigma(W_i^{j+1}\mathbf{Z}^j + b_i^{j+1}),\ ...,\ \sigma(W_{L_{j+1}}^{j+1}\mathbf{Z}^j + b_{L_{j+1}}^{j+1}))^T,\end{aligned} \tag{2.24}$$

where W_i^{j+1} are the row vectors of the weight matrix $\mathbf{W}_{j+1}$ and b_i^{j+1}, $i = 1,\ ...,\ L_{j+1}$ are the entries of the bias vector $\mathbf{b}^{j+1}$. Thus, the activation function calculates the output of each neuron in layer $j+1$ as a nonlinear function of the weighted sum of outputs from the neurons in the previous layer plus a bias. Each neuron outputs one value, and the weight in the consecutive layer determines the importance of the output of each neuron in the current layer. The nonlinear activation function σ can, for example, be the sigmoid function, hyperbolic tangent function or the binary step function to mention a few. For the last decade or so, the popularity of the Piece-Wise Linear (PWL) activation function Rectified Linear Unit (ReLU) has grown exponentially. This is in part due to its computational simplicity, representational sparsity and non-vanishing gradients. The ReLU activation function is given by:

$$\sigma(z) = max\{0,\ z\}. \tag{2.25}$$

ReLU is the only activation function used in the work of this thesis.

2.2.1 Sparse neural networks and regularization

Dense neural networks are often overparameterized models, meaning that they have more parameters than can be estimated from the data and thus often suffer from overfitting. In [88], it is shown empirically that randomly initialized dense

neural networks contain subnetworks that can improve generalization compared to the dense networks. These subnetworks, characterized by significantly fewer non-zero trainable parameters than their dense counterpart, are called sparse neural networks. Their utility can further be seen in terms of increased computational performance for inference and training, and increased storage and energy efficiency. Typically large-scale models that require millions to billions of parameters and arithmetic operations can highly benefit from such sparsification. To conclude sparsification of complex models will lead to simpler models which are relatively easier to interpret, generalize, and train.

There are many existing schemes and methods for training sparse neural networks. Coarsely speaking, model sparsity can be divided into structured sparsity, which includes pruning for example entire neurons, and unstructured sparsity, which deals with pruning individual weights. Furthermore, methods can be classified into one out of three: *data-free* (such as magnitude pruning [89]), *data-driven* (such as selection methods based on the input or output sensitivity of neurons [90]) and *training-aware* methods (like weight regularization) based on the method's way of selecting candidates for removal. A comprehensive review is given in [91]. Among the methods that can be used to sparsify a complex network, regularization techniques are the most popular ones, with a solid mathematical foundation. In regularization, penalty terms $R(\mathbf{w})$ defined on the weights are added to the cost function C:

$$C(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}) = L(\mathbf{y}_i, \hat{\mathbf{f}}(\mathbf{x}_i; \boldsymbol{\theta})) + \lambda R(\mathbf{w}). \tag{2.26}$$

The vector $\boldsymbol{\theta} = \{\mathbf{w}, \ \mathbf{b}\}$ denotes the adaptable parameters, namely the weights $\mathbf{w}$ and biases $\mathbf{b}$ in the network model. Furthermore, $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ is the training data, $L(\cdot, \cdot)$ is the loss function to minimize and the positive scalar coefficient λ is a hyperparameter to weight the terms $L(\cdot, \cdot)$ and $R(\cdot)$. The standard choice of loss function $L(\cdot, \cdot)$ for regression tasks is the Mean Squared Error (MSE):

$$L(\mathbf{y}_i, \hat{\mathbf{f}}(\mathbf{x}_i; \boldsymbol{\theta})) = (\mathbf{y}_i - \hat{\mathbf{f}}(\mathbf{x}_i; \boldsymbol{\theta}))^2. \tag{2.27}$$

In the training process, the cost function is minimized to find optimal values of the parameters:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left\{ \frac{1}{N} \sum_{i=1}^{N} C(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta}) \right\}. \tag{2.28}$$

The most intuitive sparsity promoting regularizer is the ℓ_0 norm, often referred to as the sparsity norm:

$$R_{\ell_0}(\mathbf{w}) = ||\mathbf{w}||_0 = \sum_i \begin{cases} 1 & w_i \neq 0, \\ 0 & w_i = 0. \end{cases} \tag{2.29}$$

The ℓ_0 norm counts the number of nonzero weights. Unfortunately, the ℓ_0 norm has several drawbacks that make it less suitable for optimization. The ℓ_0 norm is nondifferentiable and sensitive to measurement noise. Furthermore, in terms of computational complexity, the problem of ℓ_0 norm is shown to be NP-hard [92]. The ℓ_1 norm is a convex relaxation of the ℓ_0 norm, and is given by:

$$R_{\ell_1}(\mathbf{w}) = ||\mathbf{w}||_1 = \sum_i |w_i|. \tag{2.30}$$

Due to its geometrical characteristics, ℓ_1 minimization is sparsity promoting. However, the ℓ_1 norm usually does not reduce the weights to zero but rather to very small magnitudes. Thus, magnitude pruning can be applied after ℓ_1 minimization to achieve true sparse models.

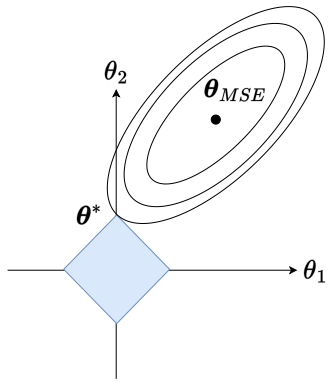
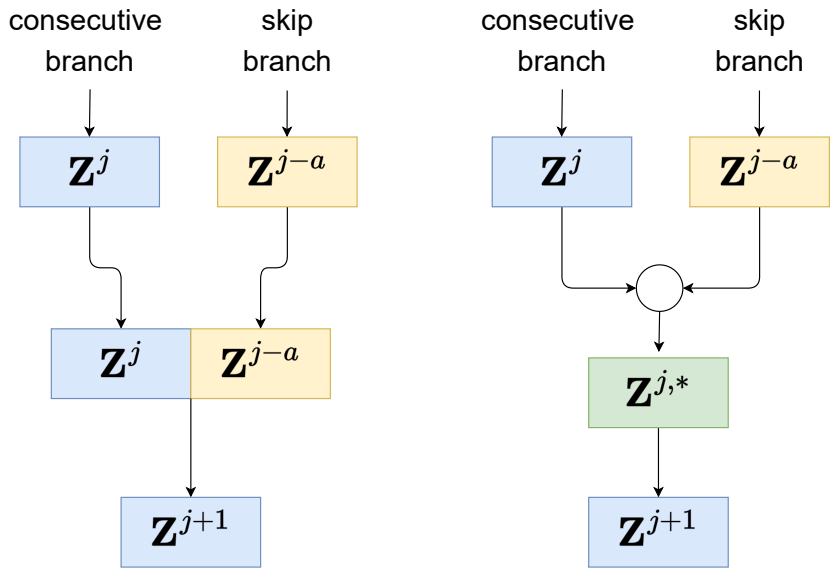


Figure 2.5: Illustration of ℓ_1 regularization. θ_1 and θ_2 are the model parameters. $\boldsymbol{\theta}_{MSE}$ is the MSE estimate. The ellipses show the contours of the error from the MSE estimate. The blue diamond illustrates the ℓ_1 constraints. $\boldsymbol{\theta}^*$ is the parameter estimate when ℓ_1 regularization is added to the optimization.

Fig. 2.5 illustrates how ℓ_1 regularization can lead to a sparse solution. The contours represented by the ellipse correspond to the mean squared error part ($L(\mathbf{y}_i, \hat{\mathbf{f}}(\mathbf{x}; \boldsymbol{\theta}))$) of the cost function $C(\mathbf{x}_i, \mathbf{y}_i, \boldsymbol{\theta})$ while the contours represented by the rhombus correspond to the regularization term. The stronger the regularization parameter λ the more the weights θ will be pushed towards the origin. It can be clearly seen that the two contours corresponding to the two parts of the cost function has a good chance interesting on the axis which will results in many θ values being zero.

2.2.2 Skip-connections

The representation capacity is generally understood to increase with both the depth and the width (the number of neurons in each layer), although early attempts to train very deep networks found them challenging to optimize using backpropagation due to the vanishing gradients problem. One of the major developments that enabled researchers to train deep NNs with many layers is the *skip connection*. A skip connection is simply an additional inter-layer connection that bypasses some of the layers of the network. This provides alternate pathways through which the loss can be backpropagated to the early layers of the NN, which helps mitigate the issues of vanishing and exploding gradients, which were major hurdles to training deeper models. In general, there are two fundamental ways skip connections are joined to the network after passing some layers. These are *addition*, as proposed in [93], also known as residual skip connection, and concatenation, as proposed in [82].



(a) Concatenated skip connection

(b) Residual skip connection

Figure 2.6: Illustrating the difference between concatenated and addition based skip connections. A skip connection from layer $j-a$ skip $a-1$ consecutive layers. After added or concatenated with hidden representation $\mathbf{Z}^{j}$, a regular nonlinear transformation is conducted to produce the next hidden transformation $\mathbf{Z}^{j+1}$.

Figure 2.6 illustrates the difference between the two. In the work of this thesis,

only concatenated skip connections are considered. The nonlinear transformation with concatenated skip connections as illustrated in Figure 2.6a can be formulated as:

$$\mathbf{Z}^{j+1} = \sigma(\mathbf{W}^{j+1,'}[\mathbf{Z}^{j}, \mathbf{Z}^{j-a}]) + \mathbf{b}^{j+1,'}). \tag{2.31}$$

$\mathbf{W}^{j+1,'}$ is the extended weight matrix, $\mathbf{b}^{j+1,'}$ is the extended bias vector, and $[\mathbf{Z}^{j}, \mathbf{Z}^{j-a}]$ is the concatenation of vector $\mathbf{Z}^{j}$ and $\mathbf{Z}^{j-a}$.

2.2.3 Deep Active Learning

Deep Active Learning (DeepAL) has emerged as a combined approach between DL and AL, addressing DL specific challenges within AL. This mainly includes dealing with over-confident uncertainty estimates of NN predictions, efficiently data-acquisition of data batches rather than the traditional AL one-by-one query method, and the joint optimization of the NN model and AL algorithm [67]. The majority of research on DeepAL focuses on static acquisition problems. Static acquisition problems refer to scenarios where data is already available, and any point in the input space can be acquired directly. Examples of such problems are visual data processing such as image classification [94] and object detection [95], NLP such as machine translation [96], text classification [97] and semantic analysis [98]. The static acquisition problem imply that there exists an unlabeled dataset $\mathcal{U} = \{\mathcal{Z}\}$ with c input samples $\mathcal{Z} = \{\mathbf{z}_1,\ \mathbf{z}_2,\ ...,\ \mathbf{z}_c\}$. The goal of DeepAL in the static acquisition problem is to acquire as few as possible of the unlabeled data in $\mathcal{U}$ for labeling by choosing the most informative samples. That includes designing a query strategy $Q, \mathcal{U} \xrightarrow{Q} \mathcal{L}$, where $\mathcal{L} = \{\mathcal{Z}, \mathcal{Y}\}$ is a labeled dataset, and $\mathcal{Y}$ are labels corresponding to inputs $\mathcal{Z}$. The query strategy can be expressed in terms of an acquisition function a_{batch} which acquires a batch $\mathcal{B}^* = \{z_1^*,\ z_2^*,\ ...,\ z_b^*\}$ of samples to be labeled. The batch based query called Batch Mode Deep Active Learning (BMDAL) is the foundation of DeepAL. The DeepAL scheme is an iterative acquisition scheme, and one acquisition step is generally defined by:

$$\mathcal{B}^* = \underset{\mathcal{B} \subseteq \mathcal{U}}{\operatorname{argmax}} \quad a_{batch}(\mathcal{B}, \hat{\mathbf{f}}(\mathcal{L})), \tag{2.32}$$

Here $\hat{\mathbf{f}}(\mathcal{L})$ is the NN. $\mathcal{L}$ is the labeled data up until the given acquisition step, and the notation $\hat{\mathbf{f}}(\mathcal{L})$ indicate that the NN is trained on this data. The acquisition function a_{batch} is in general a function of the NN that is trained on the currently acquired data since the informativeness of new samples can be evaluated using this NN.

The acquisition function a_{batch} defines the query strategy of the AL scheme. There exists a range of different query strategies in AL. Here we will shortly describe the strategies relevant to the case study. *Uncertainty-based* strategy is one of the most

popular strategies in AL. The strategy aims to select samples in which the model predictions are most uncertain about. Uncertainty-based AL methods are typically computationally efficient and easy to implement. Moreover, these methods typically provides highly informative samples. One of the most utilized uncertainty-based methods calculates the predictive entropy $H[\mathbf{y}|\mathbf{x}, \mathcal{L}]$ for a given sample $\mathbf{x}$. However, there are some concerns about applying uncertainty-based sampling strategies in BMDAL. Acquiring a batch of the most informative samples using an uncertainty measure can lead to a batch of very similar samples. Moreover, strategies of this type are often focused on examples close to a decision boundary, making it vulnerable to adverserial attacks [67]. Hence, a *Hybrid strategy* is often preferred, accounting for diversity in the sampled data. A method called Diverse Mini-Batch Active Learning (DMBAL) [99] adds informativeness to the optimization of a K-means algorithm in the weights of each candidate sample. In the DMBAL algorithm, informative estimates obtained by some informative measures are assigned as weights to the corresponding candidate samples. In each acquisition step, a batch $\mathcal{B}$ of b samples closest to the centroids of the weighted K-means algorithm is added to the training set.

2.3 Compressed sensing

The Shannon-Nyquist theorem states that the signal information is preserved if it is sampled uniformly at a rate at least two times faster than its bandwidth. In different applications, signal acquisition is prohibitive due to the cost of measuring the signal or simply because sensors do not sample the signal at rates as high as required by the Shannon-Nyquist theorem. Compressed sensing provides an alternative to Shannon-Nyquist sampling when the estimated signal is sparse or compressible [39]. Consider the signal $\mathbf{x}$ of length n represented in a transform basis Ψ such that $\mathbf{x} = \Psi\mathbf{s}$. A sparse signal $\mathbf{x}$ can in some basis Ψ be represented by $k << n$ nonzero coefficients in $\mathbf{s}$. A compressible signal $\mathbf{x}$ can be approximated by $k << n$ coefficients in $\mathbf{s}$. That is, when coefficients in $\mathbf{s}$ are sorted according to magnitude, they decay rapidly after the k'th coefficient. Compressed sensing addresses the problem of estimating a signal $\mathbf{x}$ of length n from m linear measurements $\mathbf{y}$ when $m << n$ by finding a solution to an underdetermined linear system:

$$\mathbf{y} = \Phi\mathbf{x}, \tag{2.33}$$

where Φ is the measurement matrix of size $m \times n$.

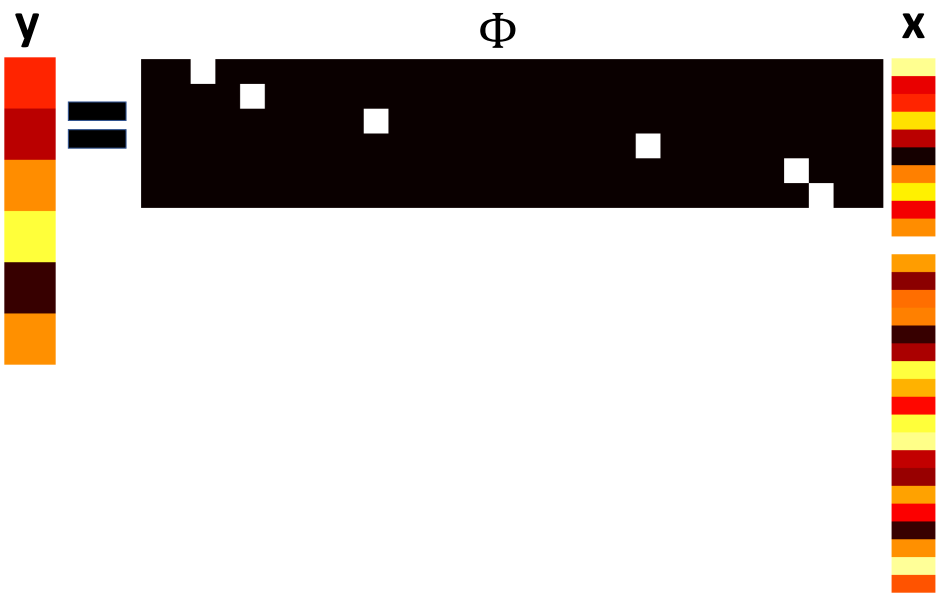


Figure 2.7: Matrix illustration of the linear underdetermined system $\mathbf{y} = \Phi\mathbf{x}$. Since the system is underdetermined and the signal $\mathbf{x}$ is in general not sparse, the system can in general not be solved or have infinitely many solutions. The left vector illustrates the measurements $\mathbf{y}$, the matrix in the middle illustrates the measurement matrix Φ and the vector on the right illustrates the signal $\mathbf{x}$. The black areas in Φ are zero entries, and the white areas are ones. Φ maps values between $\mathbf{x}$ and $\mathbf{y}$. Colors in $\mathbf{x}$ and $\mathbf{y}$ correspond to numeric values and range from small values corresponding to dark and large values corresponding to light.

Figure 2.7 illustrates the linear system in matrix form. This expresses that the number of equations (m) is much smaller than the number of unknowns (n).

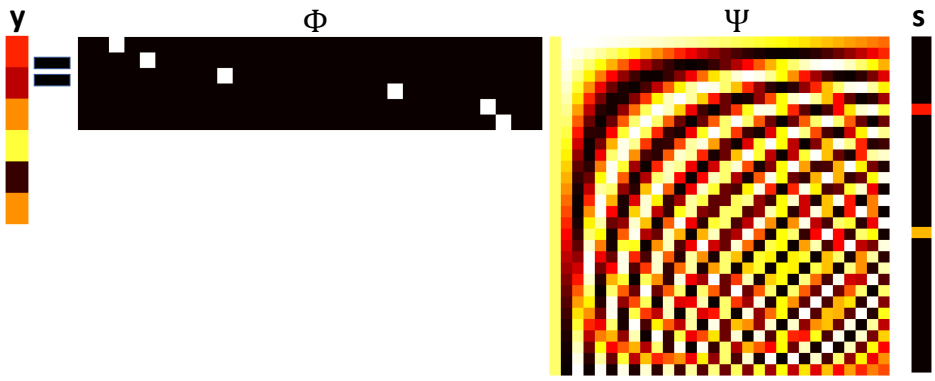
Now, consider the mapping of the signal $\mathbf{x}$ from time or space domain to a transform basis Ψ

$$\mathbf{x} = \Psi\mathbf{s}, \tag{2.34}$$

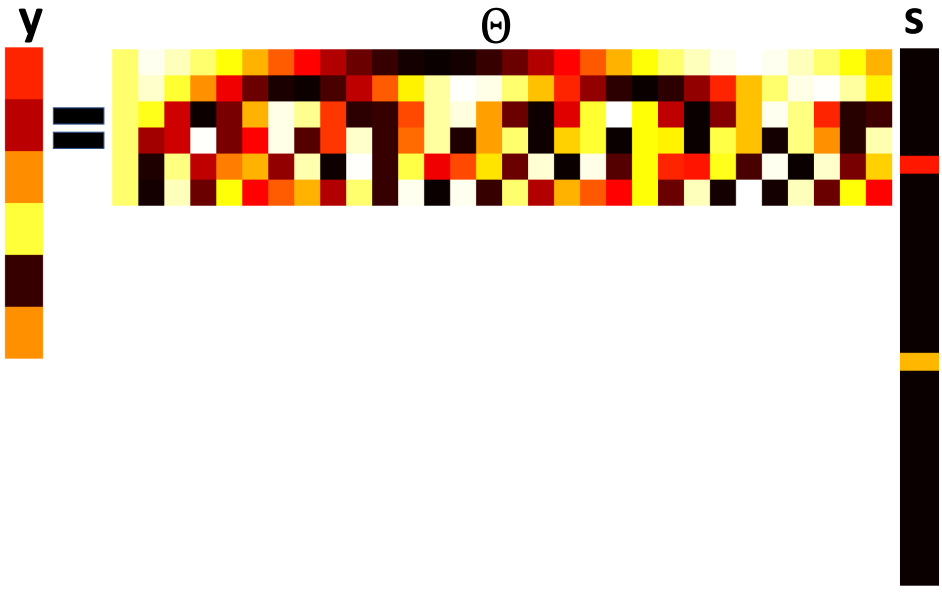
where $\mathbf{s}$ are the coefficients representing the signal $\mathbf{x}$ in Ψ. The resulting linear system is given by

$$\mathbf{y} = \Theta\mathbf{s}, \tag{2.35}$$

where $\Theta = \Phi\Psi$, $\Theta \in \mathbb{R}^{\mathrm{m}\times\mathrm{n}}$ This transformation is illustrated in Figure 2.8. The rows of Φ, $\{\phi_j\}_{j=1}^{m}$, represent the measurement vectors, while the columns of Ψ, $\{\psi_j\}_{j=1}^{n}$, represent orthonormal basis vectors. If an appropriate basis Ψ is chosen so that the signal $\mathbf{x}$ is sparse in this domain, a solution for $\mathbf{s}$ of equation (2.35) can



(a) Represents the linear system $\mathbf{y} = \Phi\Psi\mathbf{s}$. Decompose the signal $\mathbf{x}$ into a basis matrix Ψ and basis coefficients $\mathbf{s}$.



(b) Represents the linear system $\mathbf{y} = \Theta\mathbf{s}$. This is the resulting system to be solved in compressed sensing.

Figure 2.8: Matrix illustration of how the signal $\mathbf{x}$ is represented in terms of a basis $\Psi\mathbf{s}$. Hence the measurements $\mathbf{y}$ are represented in terms of the measurement matrix Φ, basis matrix Ψ and basis coefficients $\mathbf{s}$. In both (a) and (b), the leftmost vector is the measurements $\mathbf{y}$ and the rightmost vector is the coefficient vector $\mathbf{s}$. In $\mathbf{s}$, the black areas are zero entries. In (a), the middle-left matrix is the measurement matrix Φ. The black areas in Φ are zero entries, and the white areas are ones. The middle-right matrix is the transform basis Ψ. In this case, Ψ is the Discrete Cosine Transform (DCT). In (b), the middle matrix is the matrix product $\Theta = \Phi\Psi$. Despite the resulting system in (b) being underdetermined, the system can be solved for $\mathbf{s}$ if a suitable basis is found so that $\mathbf{s}$ is sparse. That is the case in this illustration, where $\mathbf{s}$ is two-sparse. Colors in $\mathbf{y}$, Ψ and $\mathbf{s}$ correspond to numeric values and range from small values corresponding to dark and large values corresponding to light.

be found from far less measurements than if the original system in equation (2.33) was to be solved for $\mathbf{x}$.

2.3.1 Low complexity structures

As mentioned, the linear system in equation (2.33) has fewer equations than unknowns, thus it is underdetermined. However, by utilizing the fact that the signal of interest $\mathbf{x} \in \mathbb{R}^n$ belongs to a low-dimensional subspace of dimension k, the system can still be solved. In other words, low complexity structures allows for recovering a signal $\mathbf{x}$ by solving the underdetermined system $\mathbf{y} = \Phi\mathbf{x}$. This touches the core of compressed sensing. All signals $\mathbf{x} \in \mathbb{R}^n$ can be expressed in a basis $\{\psi_i\}_{i=1}^n$ in terms of n coefficients $\{s_i\}_{i=1}^n$ as $\mathbf{x} = \sum_{i=1}^n s_i\psi_i = \Psi\mathbf{s}$. If the signal is k-sparse, it can be expressed by k nonzero coefficients s_i. This can be expressed mathematically as $||\mathbf{s}||_0 \leq k$, where $||\cdot||_0$ is the ℓ_0 pseudonorm expressing the number of nonzero coefficients. The set of all sparse signals is the union of $\binom{n}{k}$ k-dimensional subspaces spanned by k basis vectors. This gives the union-of-subspaces model and can be formulated mathematically as:

$$\mathbf{s} \in \bigcup_{S\subset[n],|S|=k} W_S =: \Sigma_k. \tag{2.36}$$

Here W_S is one subset of Ψ indexed by the index set $S \subset [n]$ with cardinality $|S| = k$. Hence Σ_k is the union of subspaces that correspond to vectors with at most k nonzero coefficients ([100]).

As an intuitive example, consider the canonical example in Figure 2.9, where the signal lives in $\mathbb{R}^3$. The union of subspaces spanned by maximum two vectors (2-sparse vectors) is illustrated by the three 2-dimensional subspaces $\mathbf{W}_1$, $\mathbf{W}_2$ and $\mathbf{W}_3$ in $\mathbb{R}^3$. Considering the k-sparse signal in an n dimensional space, the number of possible k dimensional subspaces the solution can live in is $\binom{n}{k} \approx klog(n/k)$. This is an important quantity in compressed sensing in terms of required measurements.

To include sets that do not necessarily form subspaces, a more general notion is needed for low-complexity structures. If the set of basis vectors is replaced with an arbitrary compact set, the signal models generated is referred to as *simple sets*:

Definition 2.3.1. Simple set: Let $\mathcal{A} \subset \mathbb{R}^n$ be an origin-symmetric compact set, and $k \in \mathbb{R}$. Then a set $\mathcal{K} \subset \mathbb{R}^n$ of vectors on the form

$$\mathbf{x} = \sum_{i=1}^{k} c_i\mathbf{a}_i, \quad c_i \geq 0, \mathbf{a}_i \in \mathcal{A} \tag{2.37}$$

is called a simple set. Since elements in $\mathcal{K}$ are conic combinations of at most k

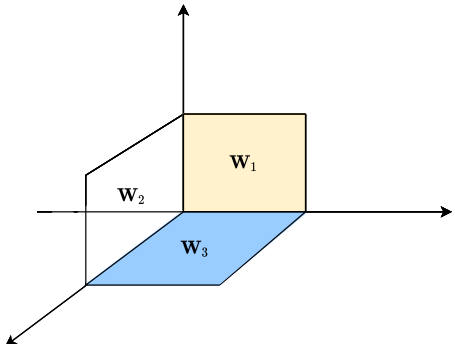


Figure 2.9: Canonical example, 2-dimensional subspaces in $\mathbb{R}^3$

elements in $\mathcal{A}$, $\mathcal{K}$ can be described as follows: $\mathcal{K} = cone_k(\mathcal{A})$. Moreover, since $\mathcal{K}$ is generated by the set $\mathcal{A}$, $\mathcal{A}$ is called an *atomic set*.

Again, consider the canonical example, where $\mathcal{A} = \{\pm \mathbf{e}_i\} \subseteq \mathbb{R}^n$. The simple set $\mathcal{K} = cone_k(\mathcal{A})$ correspond to the set $\Sigma_k(\mathbb{R}^n)$ of k-sparse vectors. Furthermore, introducing the notion of *atomic norm*, which is important in terms of compressed sensing:

Definition 2.3.2. Atomic norm: The function

$$||\mathbf{x}||_{\mathcal{A}} = inf \left\{ \sum_{\mathbf{a} \in \mathcal{A}} c_{\mathbf{a}} : \mathbf{x} = \sum_{\mathbf{a} \in \mathcal{A}} c_{\mathbf{a}} \mathbf{a}, c_{\mathbf{a}} \geq 0 \ \forall \mathbf{a} \in \mathcal{A} \right\} \tag{2.38}$$

associated with an atomic set $\mathcal{A} \subset \mathbb{R}^n$ is called the atomic norm of $\mathcal{A}$ at $\mathbf{x}$.

A general strategy in compressed sensing is to recover or estimate simple sets through *atomic norm minimization* (ANM):

$$\min_x \quad ||\mathbf{s}||_{\mathcal{A}} \quad \text{s.t.} \quad \mathbf{y} = \Theta \mathbf{s}. \tag{2.39}$$

2.3.2 Restricted isometry property

Consider the matrix $\Theta = \Phi\Psi$ in equation (2.35). In general, this matrix is rank deficient and hence loses information. However, Θ can be shown to preserve information in sparse and compressible signals if it satisfies the Restricted Isometry Property (RIP).

Definition 2.3.3. RIP: A matrix $\mathbf{A}$ is said to satisfy the RIP of order k if

$$(1 - \delta)||\mathbf{x}||_2^2 \leq ||\mathbf{A}\mathbf{x}||_2^2 \leq (1 + \delta)||\mathbf{x}||_2^2 \tag{2.40}$$

For all $\mathbf{x} \in \Sigma_k$ with $\delta > 0$.

The intuition of the RIP is that for any index set $\mathbf{S} \subset [n]$ with cardinality $|\mathbf{S}| \leq k$, the submatrix with columns of $\mathbf{A}$ indexed by $\mathbf{S}$: $\mathbf{A}_S$ approximately acts like an isometry on the set of k-sparse vectors. Direct construction of the measurement matrix Φ such that $\Theta = \Phi\Psi$ involves verifying equation (2.40) for all $\binom{n}{k}$ sparse vectors $\mathbf{x} \in \Sigma_k$. However, RIP can be achieved with high probability by selecting Φ as a random matrix [39]. This implies that to successfully estimate the signal of interest, a reasonable sampling strategy would be to sample the variable of interest at random time instants.

2.3.3 Signal estimation techniques

Given the linear system $\mathbf{y} = \Theta\mathbf{s}$, there are infinitely many coefficient vectors $\mathbf{s}$ that are consistent with the $m \ll n$ number of measurements. Therefore, to find the correct or approximate solution $\mathbf{s}$, it is necessary to exploit the a priori knowledge of sparsity or compressibility of the signal. This indicates minimizing the number of nonzero coefficients in $\mathbf{s}$ so that $\Theta\mathbf{s}$ still is consistent with the measurements $\mathbf{y}$. In its most explicit form, this minimization is done through ℓ_0-minimization:

$$\min_{s} \quad ||\mathbf{s}||_0 \quad \text{s.t.} \quad \mathbf{y} = \Theta\mathbf{s}. \tag{2.41}$$

The optimization program in equation (2.41) shows remarkable results as it is only dependent on $m = 2k$ independent measurements to recover $\mathbf{s}$ [101]. However, ℓ_0-minimization is provably NP-hard [92]. Furthermore, the solution of a ℓ_0-minimization problem can be highly sensitive to measurement noise and sparsity defects [100]. A key insight in compressed sensing is the convex relaxation of the optimization program in (2.41). The closest convex relaxation of (2.41) is the ℓ_1-minimization known as *Basis Pursuit* (BP) [102]:

$$\min_{s} \quad ||\mathbf{s}||_1 \quad \text{s.t.} \quad \mathbf{y} = \Theta\mathbf{s}. \tag{2.42}$$

The convex minimization problems such as the one in (2.42) guarantee a more stable solution that can also be solved in polynomial time. However, with this relaxation, there comes a cost in terms of increased number of required measurements, $m = \mathcal{O}(k \cdot log\,(n/k))$ as it relies on the RIP ([101]). Both optimization programs in (2.41) and (2.42) are formulated for exact reconstruction of the signal $\mathbf{x}$ due to the equality constraint. To account for measurement noise, the equality constraint can be replaced by an inequality constraint such as in Quadratic Constraint Basis Pursuit (QCBP):

$$\min_{s} \quad ||\mathbf{s}||_1 \quad \text{s.t.} \quad ||\mathbf{y} - \Phi\mathbf{x}||_2^2 \leq \eta \tag{2.43}$$

Here, η represents the noise level of the measurements. Therefore, it is desirable to estimate this noise level to get the most precise estimate of the signal of interest. Other convex programs are the Least-Absolute Shrinkage Selection Operator (LASSO) [103], the Dantzig selector (DS) [104] and Basis Pursuit Denoising (BPDN) [100]. Recovery guarantees are usually strongest for convex optimization programs. However, these programs become less practical as the problem increases in size. Thresholding algorithms represent a compromise between theoretical guarantees and efficient predictable running times. Thresholding algorithms can be divided into *hard* and *soft* thresholding. The following hard thresholding algorithm Hard Thresholding Pursuit (HTP) has been involved in the case study in Chapter 3:

Algorithm 1: HTP

Result: $\mathbf{s}^i$
Input: $\Theta\ \mathbb{R}^{m\times n}$, $\mathbf{y} \in \mathbb{R}^m$, $k \in [n]$;
Initializalize: $\mathbf{s}^0 \leftarrow 0$, $i \leftarrow 0$;
while *While condition* **do**
    instructions;
    $\mathbf{v}^{i+1} \leftarrow \mathbf{s}^n - \mu\Theta^T(\Theta\mathbf{s} - \mathbf{y})$ (Gradient descent step);
    $\mathbf{G}_{n+1} \leftarrow H_k(\mathbf{v^{i+1}})$ (Support identification) ;
    $\mathbf{s}^{i+1}_{\mathbf{G}_{n+1}} \leftarrow \Theta^{\dagger}_{\mathbf{G}_{n+1}}\mathbf{y}$ (Least squares update) ;
    $i \leftarrow i + 1$;
end

Here, H_k is a hard thresholding operator, identifying the index set $G \subset [n]$ which support the k largest values of $\mathbf{s}$, and zeroing out any values supported on $\bar{G}$. μ is a hyperparameter proportional to the gradient descent term. In the most basic case of the HTP algorithm, $\mu = 1$. In a more general case of the HTP algorithm, the hyperparameter $\mu \neq 1$. Another hard thresholding algorithm is Iterative Hard Thresholding (IHT) [105], which HTP is based on. The main difference between the two is that HTP converges faster than IHT [100]. Examples of soft thresholding algorithms are Smoothing proximal gradient method [106], and Fast iterative shrinkage-thresholding algorithm [107]. One of the most generic classifications of recovery algorithms split the algorithms into three different classes. Two of them are mentioned already, namely convex optimization and thresholding algorithms. Another class of recovery algorithms is iterative greedy algorithms [100]. Some of the most famous greedy methods are Orthogonal Matching Pursuit (OMP) [108] and Compressive Sampling Matching Pursuit (CoSaMP) [109].

2.4 Performance metrics

Accurate long-term predictions are crucial in the process industry for several reasons. The scarcity of expensive measurements implies that process models play a vital role in providing reliable forecasts without relying on feedback from measurements. Additionally, precise and stable long-term predictions contribute to decision support and process optimization. The choice of prediction horizon length depends on factors such as the model's objective, system open-loop stability, and measurement sampling rates. This section introduces performance metrics to evaluate model predictions for different prediction horizon lengths.

It is assumed that the initial condition $\mathbf{x}(t_0)$ are given to the models. Then the consecutive n time steps of the states are estimated $\{\hat{\mathbf{x}}(t_1),\ ...,\ \hat{\mathbf{x}}(t_n)\}$. This is called a *rolling forecast*. The model estimates the time derivatives of the states $d\hat{\mathbf{x}}_\mathbf{i}/dt$ based on the current state $\mathbf{x}(t_i)$ and control inputs $\mathbf{u}(t_i)$ and initial conditions $\mathbf{x}_0 = \mathbf{x}(t_0)$, or the estimate of the current state variables $\hat{\mathbf{x}}(t_i)$ if $t > t_0$:

$$\frac{d\hat{\mathbf{x}}(t_i)}{dt} = \begin{cases} \hat{\mathbf{f}}(\hat{\mathbf{x}}(t_i),\ \mathbf{u}(t_i)), & \text{if } t_i > t_0 \\ \hat{\mathbf{f}}(\mathbf{x}_0(t_i),\ \mathbf{u}(t_i)), & \text{if } t_i = t_0, \end{cases} \tag{2.44}$$

where $\hat{\mathbf{f}}(\cdot,\cdot)$ is the model. Then, the next state estimate $\mathbf{x}(t_{i+1})$ is calculated as

$$\hat{\mathbf{x}}(t_{i+1}) = \hat{\mathbf{x}}(t_i) + \frac{d\hat{\mathbf{x}}(t_i)}{dt} \cdot \Delta T. \tag{2.45}$$

2.4.1 Rolling forecast error measure

The rolling forecast can be computed for each of the states x_i for one set of test trajectories $\mathcal{S}_{test}$. However, presenting the rolling forecast of multiple test sets would render the interpretation difficult. By introducing a measure called Average Normalized Rolling Forecast Mean Squared Error (AN-RFMSE) that compresses the information about model performance, the models can easily be evaluated on a large number of test sets. The AN-RFMSE is a scalar defined as:

$$\text{AN-RFMSE} = \frac{1}{p}\sum_{i=1}^{p}\frac{1}{n}\sum_{j=1}^{n}\left(\frac{\hat{x}_i(t_j) - x_i(t_j)}{std(x_i)}\right)^2, \tag{2.46}$$

where $\hat{x}_i(t_j)$ is the model estimate of the simulated state variable x_i at time step t_j, $std(x_i)$ is the standard deviation of variable x_i in the training set $\mathcal{S}_{train}$, p is the number of state variables estimated, and n is the number of time steps the normalized rolling forecast MSE is averaged over. Hence, for every model $\hat{\mathbf{f}}_j$ and every test trajectory in the set of test trajectories $\mathcal{S}_{test}$, there is a corresponding AN-RFMSE value.

2.4.2 Model stability measure

When performing a rolling forecast, the predicted state may reach a region of the state space where the model is unstable. This may arise because the model is failing to generalize to this region, or because the system is unstable in this region. At this point, the predicted trajectory will diverge, and the error will grow exponentially. This phenomenon is referred to as a *blow-up*. The open loop instability of the model can be quantified by counting the number of blow-ups that occur within a finite time horizon. In this thesis, a blow-up is defined using the following criterion:

$$\underset{j<n}{max}\left[\frac{1}{p}\sum_{i=1}^{p}\left(\frac{|\hat{x}_i(t_j)-x_i(t_j)|}{std(x_i)}\right)\right]>3 \tag{2.47}$$

where p is again the number of estimated state variables and n is the number of time steps to consider. In other words, a blow-up is said to occur when the mean absolute error for all states and timesteps exceeds three standard deviations of the test set. Although this estimate is conservative, the number of blow-ups is not underestimated due to the exponential growth of the error.

Chapter 3

Hybrid modeling combining first principle model and compressed sensing

This chapter is based on the following publication:

[78] **E. T. B. Lundby**, A. Rasheed, I. J. Halvorsen, J. T. Gravdahl, "A novel hybrid analysis and modeling approach applied to aluminum electrolysis process". In: Journal of Process Control 105 (2021), pp. 62–77. ISSN: 0959-1524.

It presents a novel hybrid modeling approach that enables estimating stationary unmodeled dynamics in coarsly sampled signals.

3.1 Introduction

The cost and challenges of taking measurements of important state variables cause low-frequency measurements and low observability in aluminum electrolysis. This makes state estimation particularly challenging. Using highly precise predictive models can greatly enhance the accuracy of the state estimation, potentially reducing the need for frequent measurements. However, the task of enhancing predictive modeling remains a formidable challenge for both PBM and DDM modeling approaches, owing to the complex and interrelated nature of high-dimensional processes, as well as the scarcity of available data. In the pursuit of addressing these challenges, compressed sensing stands out as an interesting signal reconstruction technique. Exploiting that the signal of interest is sparse or compressible in some domain makes the problem of estimating a high dimensional signal vector from a low dimensional measurement vector possible [40]. Compressed sensing offers

a framework that enables estimating signals from far fewer measurements than required by the Nyquist criterion. Therefore, by introducing compressed sensing to estimate a measured signal in aluminum electrolysis one can possible minimize the number of expensive manual measurements taken from that signal and at the same time achieve a high-resolution estimate of the coarsely measured signal. That being said, directly applying compressed sensing to signals in the aluminum electrolysis does not guarantee success. In general, compressed sensing is used on signals or systems that are already sparse. This is rarely the case in the time-series data sampled from the aluminum extraction process, and utilizing the power of compressed sensing is not straightforward.

To summarize, there is a need for improved state estimation in aluminum electrolysis. At the same time, the cost of taking manual measurements is high, and should preferably be taken at an as low rate as possible. Thus, the main objective of this chapter is to minimize the number of expensive manual measurements needed to improve state estimation of essential variables in aluminum electrolysis cells like the side ledge thickness. To realize the objective, two research questions are formulated. These research questions are:

- How can sampled data be manipulated so that the powerful tool of compressed sensing can be utilized for estimating unmodeled dynamics from sparsely sampled data?
- How can a high-fidelity signal estimate of the unmodeled dynamics be utilized in a Kalman filter to improve the accuracy of the estimated system states?

The case study in the chapter is conducted on the simplified aluminum simulator presented in Section 2.1.1. The dynamics of this sub-process can be expressed in terms of two state variables: the aluminum mass in the cell and the thickness of frozen electrolyte on the inside of sidewalls known as the side ledge. The side ledge works as a protective layer for the sidewall against the corrosive electrolyte and molten metal at high electrolysis temperature. Furthermore, the side ledge works as thermodynamic insulation. Thus, it is essential for both safe and efficient operations. It is practically impossible to measure the side ledge profile in operating electrolysis cells [110]. However, the side ledge can be estimated from measurements of the metal height since it affects the displacement volume for the aluminum in the cell and the metal height, see Figure 2.2. The metal height measurements have to be sampled manually with a dipstick, observing the molten metal mark. These measurements are taken at low sampling rates.

In this chapter, we present a novel hybrid modeling aproach. The approach sug-

gests a way of manipulating a sampled signal with a first principle PBM to take advantage of compressed sensing. That is, estimating a signal from far fewer measurements than what is required by the Nyquist criterion. Thus, instead of estimating the measured signal, the method suggests estimating the dynamics in the signal that the first principle model does not capture. This unmodeled dynamics in the sampled signal is much sparser than the sampled signal, and therefore it is required much fewer measurements to estimate this signal with compressed sensing. The novel hybrid modeling approach presented in this study can estimate stationary, periodical unmodeled dynamics in a sampled signal from few, randomly sampled measurements. The estimate of the unmodeled dynamics is then utilized in an EKF to improve the accuracy of estimating the side ledge thickness.

The structure of the chapter is as follows. In Section 3.2, the EKF used to merge estimates and measurements is presented. In Section 3.3, there is a detailed description of the novel hybrid modeling method and a description of how simulated data is generated for analysis. In Section 3.4 the results from analysis are presented and in Section 3.5 conclusions are made.

3.2 Extended Kalman filter

The Kalman filter is a set of equations that provide a recursive solution of the least-squares method. It supports estimates of past, present, and future states based on measurements, models, and uncertainties in the system. Although the Kalman filter was initially derived for linear systems, it has been extended to nonlinear systems through online Taylor expansions of the nonlinear system. This extension is referred to as EKF. The EKF addresses the problem of estimating the state $\mathbf{x}$ of a nonlinear system

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) \tag{3.1}$$

$$\mathbf{y}_k = h(\mathbf{x}_k, \mathbf{v}_k), \tag{3.2}$$

where $f(\cdot)$ and $h(\cdot)$ are in general nonlinear functions. $\mathbf{w}_k$ represents the process noise, $\mathbf{v}_k$ represents the measurement noise, $\mathbf{y}_k$ represents the measurement, and $\mathbf{u}_k$ represent the control input. The subscript indicates at what time step the variable is sampled or estimated. The process and measurement noise are assumed to be normally distributed random variables with zero mean and covariances $\mathbf{Q}$ and $\mathbf{R}$ respectively:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{3.3}$$

$$\mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}). \tag{3.4}$$

The Kalman filter algorithm can roughly be divided into two steps, where equations of the Kalman filter fall into the groups of either *time update* or the *measure-*

ment update. Equations in the time update stage are responsible for projecting the current state estimate $\hat{\mathbf{x}}_k$ and error covariance estimate $\mathbf{P}_k$ forward in time to get the *a priori* state estimate $\hat{\mathbf{x}}_{k+1}^-$ and covariance estimate $\mathbf{P}_{k+1}^-$. Equations in the measurement stage are responsible for feedback from measurements to correct the *a priori* estimates, hence give the *a posteriori* estimates $\hat{\mathbf{x}}_k$ and $\mathbf{P}_k$ ([111]). The *a priori* and *a posteriori* error covariance estimates are defined by:

$$\mathbf{P}_k^- = E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)(\mathbf{x}_k - \hat{\mathbf{x}}_k^-)^T\right] \tag{3.5}$$

$$\mathbf{P}_k = E\left[(\mathbf{x}_k - \hat{\mathbf{x}}_k)(\mathbf{x}_k - \hat{\mathbf{x}}_k)^T\right] \tag{3.6}$$

The algorithm is as follows:

Algorithm 2: EKF

Time update:;
$\hat{\mathbf{x}}_{k+1}^- = \hat{\mathbf{x}}_k + \Delta t \cdot f(\hat{\mathbf{x}}_k, \mathbf{u}_k)$;
$\mathbf{P}_{k+1}^- = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{Q}_k$;
Measurement update:;
$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$;
$\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - h(\hat{\mathbf{x}}_k^-, \mathbf{u}_k))$;
$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^-$;

$\mathbf{A}_k$ is the Jacobian matrix of $f((\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k))$ with respect to $\mathbf{x}$, $\mathbf{H}_k$ is the jacobian matrix of $h(\mathbf{x}_k, \mathbf{v}_k)$ with respect to $\mathbf{x}$ and $\mathbf{K}_k$ is the Kalman gain. Δt is the sampling time and $\hat{\mathbf{x}}_{k+1}^-$ is calculated according to the *forward Euler method*.

3.3 Method and data generation

3.3.1 Set-up for data generation and pre-processing

The cell dimensions in the simulated cell are assumed to be $l = 20m$ as the length and $w = 2m$ as the width. The initial side ledge thickness has been set to $x_{sl} = 0.08m$ and is assumed to be uniform along all sidewalls. The unmodeled dynamics are expressed as variations in the side ledge. They are assumed to be two cosine waves with frequencies 2[per day] and 5[per day] with associated amplitudes of respectively $0.02m$ and $0.01m$. The fact that stationary periodical signals have been used to simulate unmodeled dynamics can be justified by the nature of the process. The aluminum electrolysis is a semi-batch process with periodical control inputs that induce periodical dynamics on the system states. Examples of these periodical control inputs are alumina feed and anode change. This causes

periodical dynamics in the side ledge thickness. The stationarity assumption is an approximation that can be justified for shorter periods. The initial mass of molten aluminum in the cell is set to $m_{al} = 14,700kg$. Given the dimensions of the cell, the initial side ledge thickness, and the density of molten aluminum, this corresponds to an initial metal height of $h_m[0] = 0.183m$. The line current is set to $I = 330kA$ and is assumed constant during the simulation. In the current work, a current efficiency of $CE = 0.95$ is used and assumed to be constant. Equation (2.6) with these inputs yields the mass of aluminum produced in the cell to be $2524kg/day$ ($\sim 2500kg/day$). The amount of aluminum mass tapped at each tapping is done according to the following control strategy:

$$m_{out} = \begin{cases} m_{ref} + k \cdot (h_{meas} - h_{ref}), & \text{if } h_{meas} \text{ not older than 5 hours} \\ m_{ref}, & \text{if } h_{meas} \text{ is older than 5 hours.} \end{cases} \tag{3.7}$$

where $m_{ref} = 2500kg$ and $h_{ref} = 18cm$. h_{meas} is the measured metal height. Metal is tapped from the cell every $18 - 48$ hours. The line current I and current efficiency CE is used as input to Equation (2.3) to generate a high resolution timeseries of the rate of generated aluminum mass.

As described in Section 2.3.2, having a random measurement matrix, Φ will ensure successful recovery of the signal with high probability. Therefore, measurements of the metal height are sampled at random time instants. The average sampling rate is varied for different simulations to test the limit of required data. This is also the case for the standard deviation of the measurement noise, which is varied to test the reconstruction algorithms robustness against noise. The measurement noise is assumed to be *Gaussian white*. In the simulation, the measurements are values of the simulated metal height chosen at random time instants. Each measurement has an added value drawn from a normal distribution with zero mean and a standard deviation chosen for that simulation. The simulations were conducted for a period corresponding to 100 days with a small timestep of $\Delta t = 5minutes$.

3.3.2 Novel hybrid framework

The proposed method is a novel hybrid approach that utilizes first principle system knowledge to manipulate a measured signal. The manipulated data is the residual between the measured signal and an estimate of the measured signal calculated by a physics-based model. This residual represents the unmodeled dynamics in the measured signal. The signal representing the unmodeled dynamics in the measured signal is much sparser than the measured signal itself. Therefore, much fewer measurements are required to estimate the unmodeled dynamics in the measured

signal compared to estimating the measured signal with compressed sensing techniques. The novel method is limited to estimating stationary unmodeled dynamics. A compressed sensing technique is used to estimate the sparse residual and thereby gaining information about the periodic disturbances. This information is provided to an EKF as a pseudo measurement, leading to an increase in state estimation accuracy. Figure 3.4 illustrates the novel hybrid framework developed in the work related to this chapter.

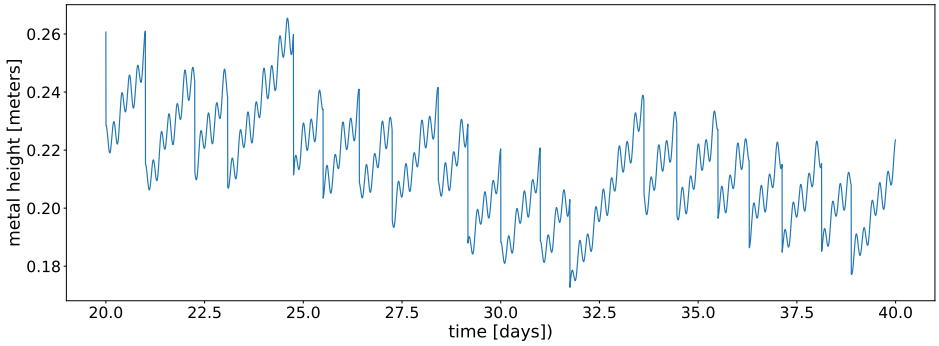
Sparsification of a signal

The dynamics of the metal height is in its nature a non-sparse signal in the DCT. This is due to the discontinuities of the saw-tooth shape in the signal. Figure 3.1b, which shows the DCT of the metal height signal, illustrates this point. Moreover, the metal height signal is non-stationary due to the non-regular tapping of metal. The compressed sensing method used in this work estimates the frequency components of a signal based on measurements of that signal. Since the metal height signal is non-stationary, the frequency components will continuously vary. This makes it much more difficult to predict the signal in the future. Instead, by utilizing this first principle model based on equation (2.6) and the available knowledge about the amount of aluminum tapped from the cell, it is possible to sparsify the signal and remove non-stationarities from the signal.

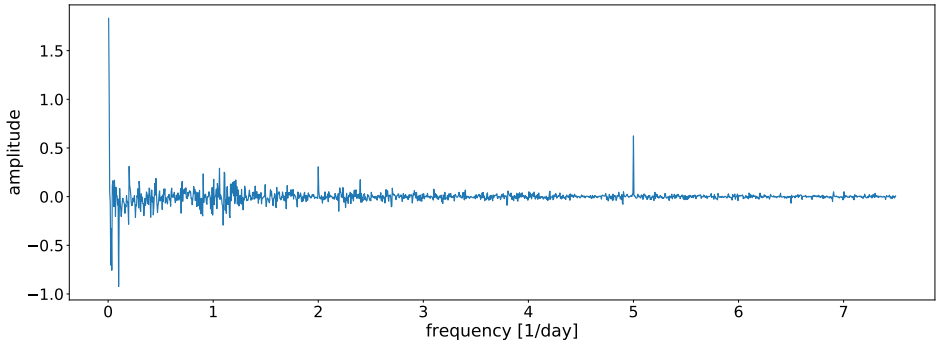
The idea is to estimate the signal h_{unmod} in Figure 3.2b that represent the unmodeled dynamics in the metal height signal by using the data points $h_{residual}$ as input in a compressed sensing algorithm. Subtracting the estimated signal h_{model} based on the physical model from the measured metal height h_{meas} gives a signal $h_{residual}$ representing the unmodeled dynamics and measurement noise. This signal h_{unmod} is much more sparse in the DCT domain than the metal height signal h_m. This can be seen by comparing Figure 3.2c and Figure 3.1b. Given that DCT is used as a basis, estimating the new signal h_{unmod} with compressed sensing requires much fewer datapoints than what is required for estimating the metal height h_m with compressed sensing. Furthermore, by including a first principle model in the estimation process and leaving estimation of only the unmodeled dynamics to compressed sensing, the robustness of the state-of-the-art estimate is preserved. Figure 3.3 illustrates the signal representation in terms of the matrix representation from Figure 2.8b.

Integrated solution

In Figure 3.4, a schematic representation of the integrated solution is presented. The integrated solution is composed of a physics-based model f, a compressed sensing signal estimation algorithm, a Kalman filter, and a metal height measure-

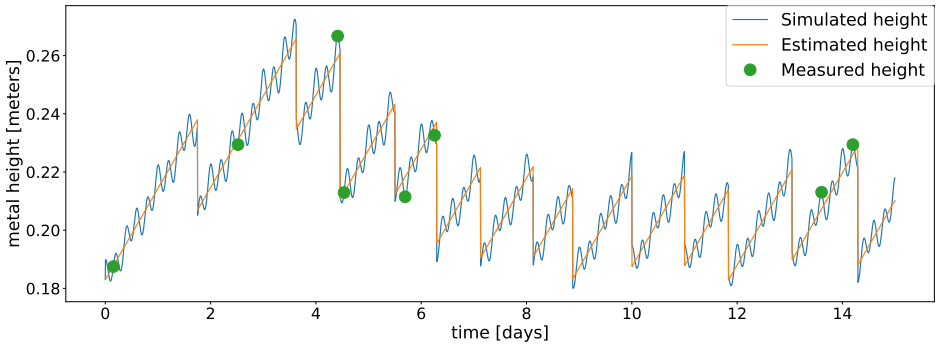


(a) Simulated height from day 20 to day 40

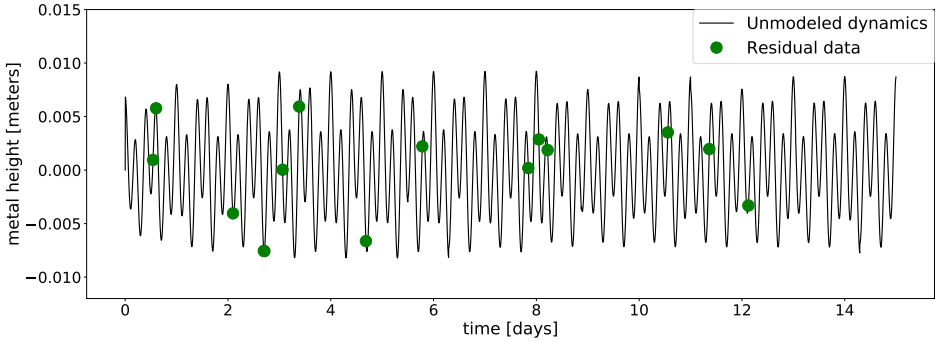


(b) DCT of simulated height.

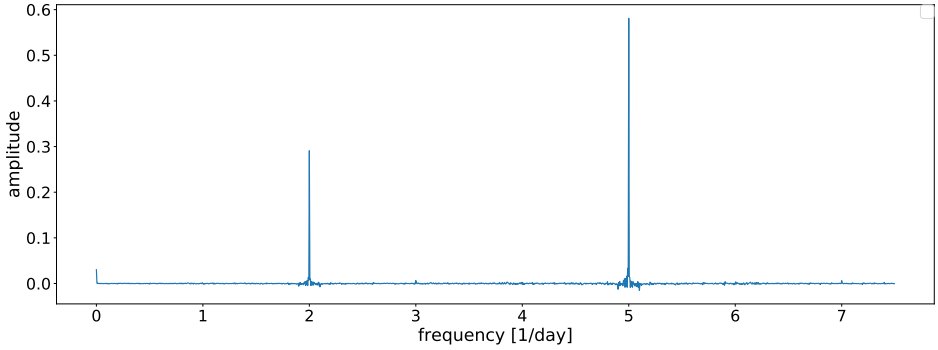
Figure 3.1: In Figure (a), the simulated height h_m from day 20 to day 40 is shown. In (b), the DCT of h_m, excluding the zero-frequency component, is shown. Frequency components of the unmodeled dynamics are those at 2 [per day] and 5 [per day]. Clearly, the transform is not sparse in the Cosine transform domain. Therefore it will be difficult to estimate the signal h_m from a limited number of measurements given the DCT as a transform basis.



(a) Simulated height and estimated height from first principle model



(b) Unmodeled dynamics. Residual between measured and estimated metal height.



(c) DCT of unmodeled dynamics.

Figure 3.2: In (a), the orange graph represents the estimated metal height h_{model} based on Faraday's law and knowledge of the amount of metal tapped. The blue graph represents the simulated metal height h_m. The green points represent measured metal height h_{meas}. In (b), the black graph represent the difference between simulated and estimated metal height $h_{unmod} = h_m - h_{model}$, whereas the green points represent the difference between measured and estimated metal height at the time instants when the measurements where taken $h_{residual} = h_{meas} - h_{model}$.

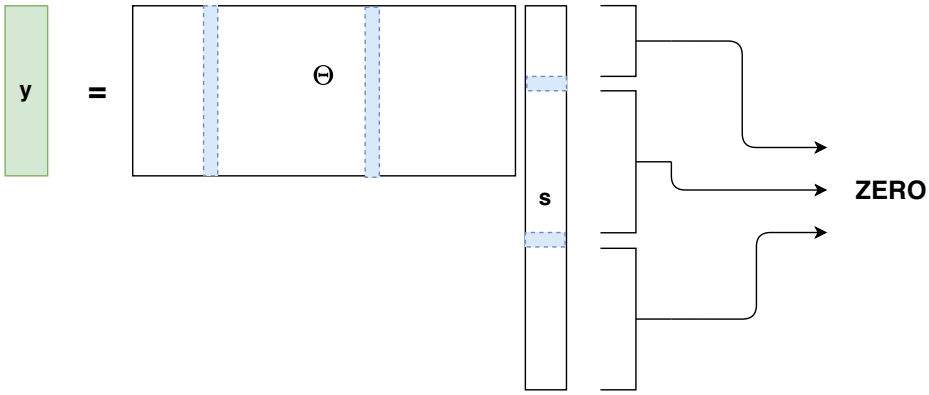


Figure 3.3: The signal illustrated on matrix form. The blue columns of Θ corresponds to the blue nonzero coefficients in $\mathbf{s}$.

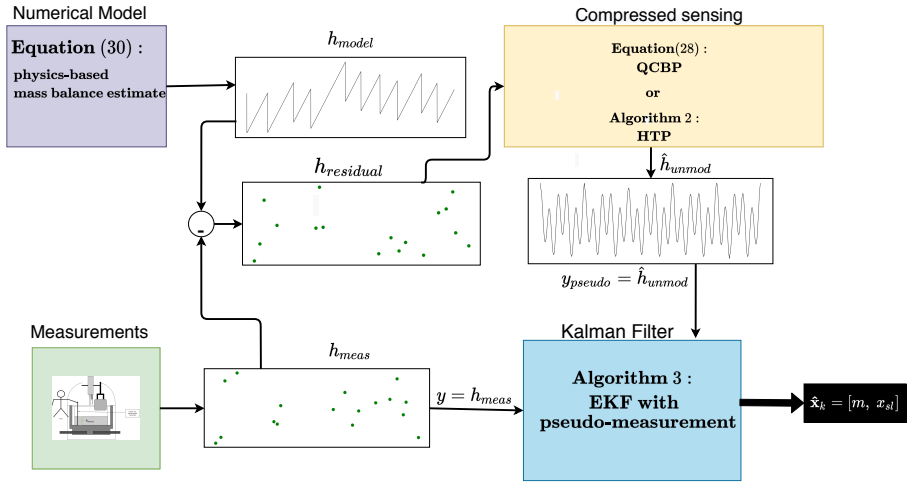


Figure 3.4: **Methodology:** The numerical model generates high temporal resolution time-series of the height h_{model}. From the measurements, coarse resolution time-series of height h_{meas} is obtained. The difference between them when height measurements are taken is referred to as $h_{residual}$. $h_{residual}$, which has the same time resolution as h_{meas} is provided as an input to the compressed sensing algorithm to generate a signal estimate and hence high-resolution estimate h_{unmod} of the unmodeled dynamics expressed in the height signal. h_{unmod} is then provided to an EKF as a pseudo measurement to estimate the side ledge thickness. The numerical model f and the height measurement h_{meas} are also provided to the EKF to estimate the state vector $\mathbf{x}$.

ment. The physics-based model is as follows:

$$f = \frac{dm}{dt} = \dot{m}_{in} - \dot{m}_{out} = \frac{CE \cdot M_{Al}}{F \cdot z} I[k] - \dot{m}_{tapped}[k]. \tag{3.8}$$

f estimates the mass flow in the electrolytic cell based on the inputs from the line current $I[k]$ and flow of tapped metal $\dot{m}_{tapped}[k]$ at each time instant k. Given an estimate of the mass of aluminum and displacement volume in the cell, a metal height estimate can be calculated. This gives the first principle estimate h_{model} visualized in Figure 3.2a. Since the dynamics of the side ledge thickness affects the displacement volume in the cell, it also affects the metal height in the cell. The lack of a model estimating the dynamics of the side ledge thickness causes the unmodeled dynamics in the height signal h_{unmod} visualized in Figure 3.2b. The compressed sensing signal estimation method estimates the unmodeled dynamics expressed in the height signal h_{unmod} using the manipulated datapoints $h_{residual}$ in Figure 3.2b. This estimate is then used as a pseudo measurement for the Kalman filter to estimate the side ledge thickness x_{sl}. In terms of a state-space model, the input vector is defined as:

$$\mathbf{u}_k = [u_{1,k},\ u_{2,k}] = \left[\frac{\dot{m}_{in,k}}{\rho},\ \frac{\dot{m}_{out,k}}{\rho}\right], \tag{3.9}$$

the state vector is defined as:

$$\mathbf{x}_k = [x_{1,k},\ x_{2,k}] = [m,\ x_{sl}], \tag{3.10}$$

and the measurement is defined as

$$y = h_{meas}. \tag{3.11}$$

This gives the following state-space model:

$$\begin{bmatrix}\dot{x}_1 \\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}u_1 - u_2 + v_1 \\ v_2\end{bmatrix}, \tag{3.12}$$

$$y = \frac{x_1}{Area(x_2)} + w. \tag{3.13}$$

$\mathbf{v} = [v_1,\ v_2]$ is process noise, whereas w is the measurement noise.

Furthermore, the pseudo measurement representing the estimated unmodeled dynamics is defined as:

$$y_{pseudo} = \hat{h}_{unmod}. \tag{3.14}$$

y_{pseudo} is used as a high-frequency measurement when a signal estimate is available. The pseudo measurement has corresponding Kalman filter Covariance matrices

for the pseudo measurement $\mathbf{Q}_{pseudo}$ and $\mathbf{R}_{pseudo}$. Since the physics-based model is assumed to have great abilities in estimating the mass balance of the cell, the element of $\mathbf{Q}_{pseudo}$ representing the process noise covariance for the mass estimate Q_{11} is set to zero. The element of $\mathbf{Q}_{pseudo}$ representing the process noise covariance of the side ledge thickness Q_{22} is set to one. The measurement noise covariance $\mathbf{R}_{pseudo}$ is much smaller than Q_{22}, indicating that the pseudo measurement y_{pseudo} is trusted much more than the *a priori* estimate of the side ledge thickness. Thus, the *a posteriori* estimate of the side ledge thickness will be greatly influenced by the pseudo measurement. Since $Q_{11} = 0$, the *a posteriori* estimate of the mass will not be influenced by the pseudo measurement.

Algorithm 3: EKF with pseudo measurement

Time update:;
$\hat{\mathbf{x}}_{k+1}^{-} = \hat{\mathbf{x}}_k + \Delta t \cdot f(\hat{\mathbf{x}}_k, \mathbf{u}_k)$;
$\mathbf{P}_{k+1}^{-} = \mathbf{A}_k \mathbf{P}_k \mathbf{A}_k^T + \mathbf{Q}_k$;
if *Meaurement* y_k *is available* **then**
  Measurement update:;
  $\mathbf{K}_k = \mathbf{P}_k^{-} \mathbf{H}_k (\mathbf{H}_k \mathbf{P}_k^{-} \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$;
  $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_k (\mathbf{y}_k - h(\hat{\mathbf{x}}_k^{-}, \mathbf{u}_k))$;
  $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^{-}$;
else
  $\mathbf{K}_{k,pseudo} = \mathbf{P}_{k,pseudo}^{-} \mathbf{H}_k (\mathbf{H}_k \mathbf{P}_{k,pseudo}^{-} \mathbf{H}_k^T + \mathbf{R}_{k,pseudo})^{-1}$;
  $\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_k^{-} + \mathbf{K}_{k,pseudo} \mathbf{y}_{k,pseudo}$;
  $\mathbf{P}_{k,pseudo} = (\mathbf{I} - \mathbf{K}_{k,pseudo} \mathbf{H}_k) \mathbf{P}_{k,pseudo}^{-}$
end

Algorithm 3 describes how the pseudo measurements y_{pseudo} are incorporated into the EKF. y_{pseudo} is treated as a measurement in the EKF. In a regular measurement update, when the a posteriori estimate $\hat{\mathbf{x}}_k$ is calculated, a model estimate $h(\hat{\mathbf{x}}_k^{-}, \mathbf{u}_k)$ of the variable $\mathbf{x}$ is subtracted from the measurement to include both measurement and model estimate in the posterior estimate. However, since y_{pseudo} is the estimate of the unmodeled dynamics, there is no model estimate of this signal. Thus, the posterior estimate after the pseudo measurement is included only depends on the a priori estimate and the pseudo-measurement, see Algorithm 3.

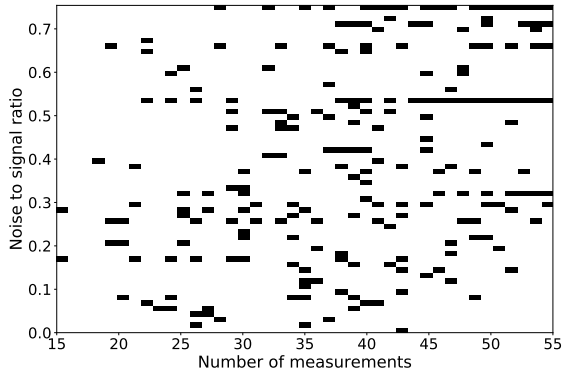
3.4 Results

In this section, the results from a case study of the hybrid method explained in Section 3.3.2 are presented. The data generation of the simulated data used in the case study is described in Section 3.3.1. The unmodeled dynamics that the hybrid

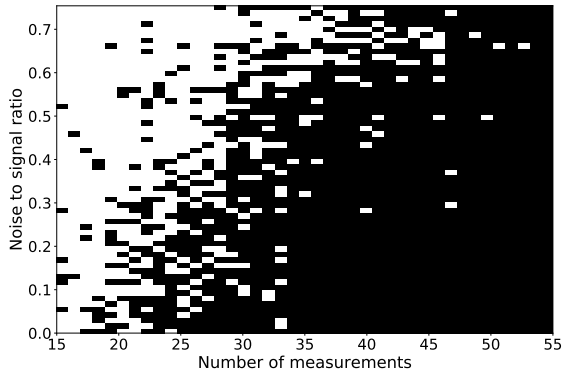
modeling framework aims to estimate is a periodical, stationary signal composed of two frequency components. As stated in the introduction, it is of interest to minimize the number of measurements and at the same time improve predictive modeling. Therefore Section 3.4.1 investigates the number of measurements required to successfully estimate the unmodeled dynamics with the novel hybrid method. Furthermore, the robustness against measurement noise of the method is assessed. Section 3.4.2 illustrates how the estimate of the unmodeled dynamics provided by the novel method affects the state estimation in the Kalman filter. Throughout the study, two different compressed sensing techniques were applied. These are the QCBP optimization program and two versions of the HTP algorithm.

3.4.1 Noise and measurement study

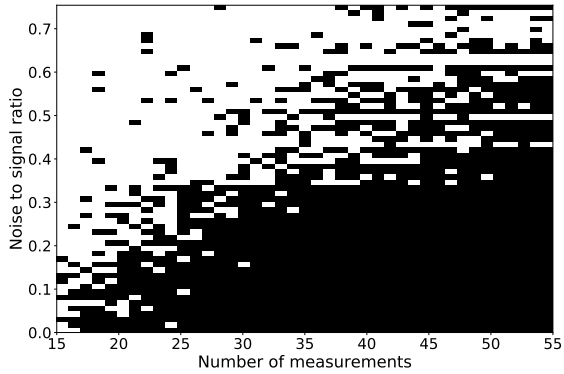
The performance of the hybrid modeling approach given two different compressed sensing techniques is assessed in this section. The performance measure in Figure 3.5 is a binary value stating if the correct signal support for the unmodeled dynamics was found by the compressed sensing algorithm or not. Signal support means the coefficients defining the signal in the sparse, transformed domain. The performance measure in Figure 3.6 is the Rooted Mean Squared Error (RMSE). The performance in both Figure 3.5 and Figure 3.6 is tested for several measurements and noise levels. The HTP algorithm explicitly requires that the algorithm search for a signal support with a given number of coefficients. Since the QCBP program is an optimization program minimizing the ℓ_1-norm of the support coefficients $\mathbf{s}$ with constraints on the quadratic error between measurement and estimated signal, it is not expected that the program will find a signal support with the exact same number of coefficients as the correct solution. In general, some coefficients that are not part of the correct solution can be expected to be included in the estimate calculated by the QCBP program. Therefore, the requirement for the QCBP program to succeed in the performance test presented in Figure 3.5 is that it finds the correct support and that the largest erroneous estimated coefficient is smaller than 0.3 times the smallest of the correct coefficients. Figure 3.5 shows how QCBP and HTP estimation strategies perform to find the correct support or basis coefficients in the simulations with different levels of measurement noise and different number of measurements used in the estimation. In the HTP algorithm, tuning the hyperparameter μ turns out to be of great importance. The amplitude of the unmodeled dynamics estimated is small in magnitude ($< 0.01[m]$). Therefore, the gradient descent term $\mu\Theta^T(\Theta\mathbf{s} - \mathbf{y})$ for the basic implementation of HTP with $\mu = 1$ becomes very small after the first iteration and converges. Implementing the HTP algorithm with $\mu >> 1$ estimates the correct support in many more cases than the basic test with $\mu = 1$. Figure 3.5b and Figure 3.5c show that the success of the method is dependent on both the number of measurements used and the measure-



(a) Performance plot of the HTP algorithm. Hyperparameter $\mu = 1$



(b) Performance plot of the HTP algorithm. Hyperparameter $\mu = 700$



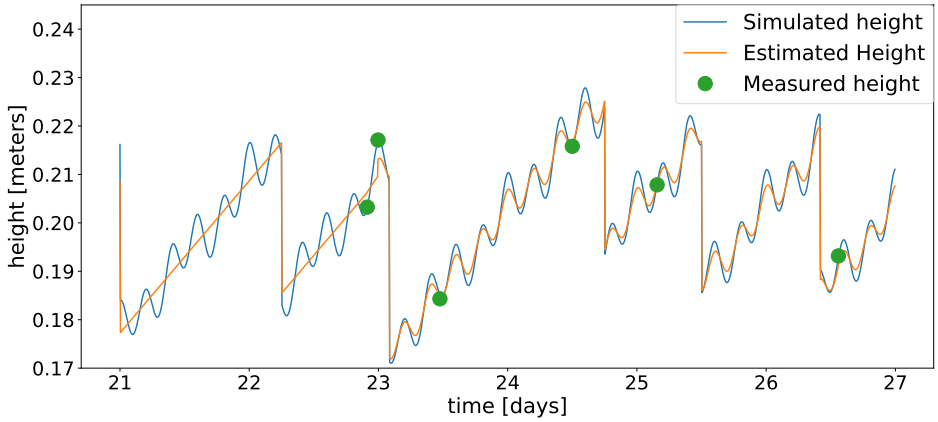
(c) Performance plot of QCBP

Figure 3.5: Performance plot of the HTP algorithm and QCBP optimization program. On the horizontal axis, the number of measurements of metal height in the simulation varies, while the noise to signal ratio varies along the vertical axis. The noise to signal ratio is defined as the ratio between the standard deviation of the measurement noise and the average amplitude of the unmodeled dynamics signal for any given simulation. For a given noise to signal ratio and a given number of measurements, the color black indicates that the reconstruction algorithm found the correct support for the signal.

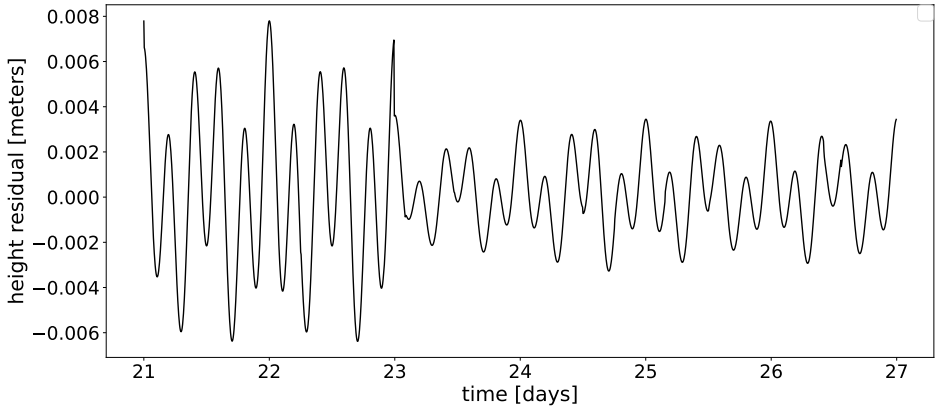
ment noise. The figures show a clear tendency that the number of measurements required for a successful signal estimation increases with increasing noise. Figure 3.6 shows the RMSE for the estimated signals from the QCBP program in Figure 3.6a and the RMSE for HTP algorithm with $\mu = 700$ in Figure 3.6b. Simulations with different levels of noise and number of measurements are included in the test. Figure 3.5 and Figure 3.6 are results from the same test. Figure 3.6a shows that the RMSE increases proportionally with the measurement noise for the QCBP program. The explanation for this can be that the optimization program in QCBP has an inequality constraint allowing for a certain maximum quadratic error between the estimate and the measured values proportional to the standard deviation of the noise. Furthermore, comparison of Figure 3.5c and 3.6a indicates that the RMSE does not seem to be significantly dependent on the number of measurements used to estimate the signal as long as the correct support is estimated. The same can be said about the HTP algorithm, which has consistently low values of the RMSE as long as the correct support is found. Figure 3.5b and 3.6b show that the RMSE is significantly lower when the correct support is found compared to when the correct support is not found. The QCBP program has an inequality constraint that allows for a certain quadratic error between measured and estimated values that limits the RMSE regardless of if the correct support is found or not. The HTP algorithm does not have this limitation in RMSE in the search for correct support. Hence, in general, the RMSE will be significantly larger if HTP finds the wrong support for the signal. For the QCBP program, this difference is not that clear due to the quadratic constraint.

3.4.2 State and signal estimates

In this section, the estimated metal height, height residual as well as the state estimates of aluminum mass and side ledge thickness is presented. In Figure 3.7 and Figure 3.8, simulated values and estimates when using the QCBP optimization are presented, whereas in Figure 3.9 and Figure 3.10 simulated values and estimates when using the HTP algorithm are presented. In both cases, a noise to signal ratio equal to 0.215 was used. The time interval for the figures includes the measurement from which the compressed sensing reconstruction algorithm finds the correct support of the unmodeled dynamics. In the case presented, the HTP algorithm needed a few more measurements than the QCBP algorithm to find the correct support for the unmodeled dynamics. That is, the QCBP program needed 25 measurements of the signal to find the correct support for the signal while the HTP algorithm needed 28 measurements to find the correct support for the signal. Therefore, the time frame shown in Figure 3.7 and Figure 3.8 differs from the time frame in Figure 3.9 and Figure 3.10.



(a) Simulated and estimated height.



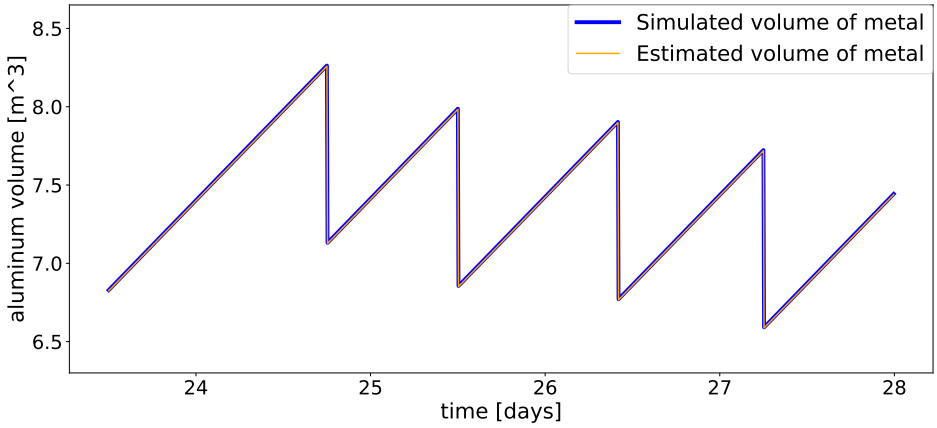
(b) Height residual.

Figure 3.7: In (a), the blue graph represents the simulated metal height h_m, whereas the orange graph represents the estimated metal height, estimated by the Kalman filter. The green points are the measured values of the metal height. In (b), the residual between simulated metal height h_m and Kalman filter-estimated metal height $\hat{h}_m$ is shown. The unmodeled dynamics in the metal height are estimated from the QCBP program.

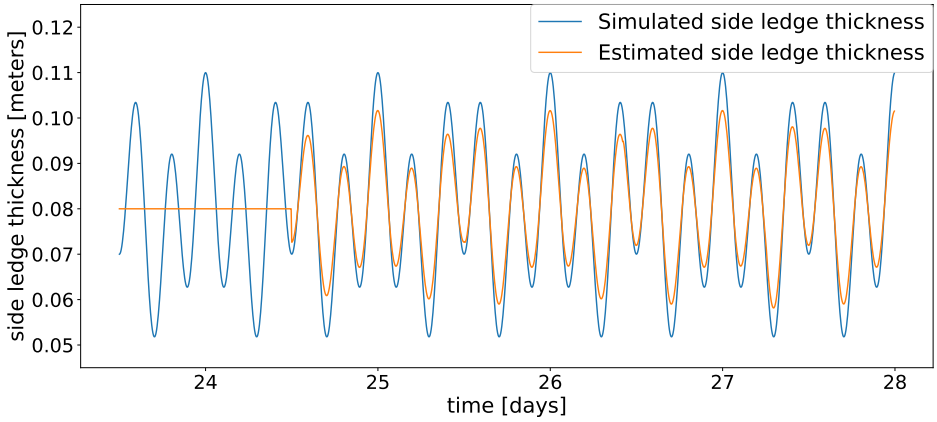
Figure 3.7a shows the simulated and the estimated metal height. The unmodeled dynamics are estimated by the QCBP optimization program using data points $h_{residual}$, plotted as green points in Figure 3.2b. The estimate is then fed into the Kalman Filter as a pseudo measurement, where the covariance matrices $\mathbf{Q}_{pseudo}$ and $\mathbf{R}_{pseudo}$ are tuned such that only the side ledge thickness $\mathbf{x}_{sl}$ is updated. Figure 3.7b shows the residual between simulated and estimated metal height. Figure

3.7a shows that the estimated metal height changes character and simultaneously follows the simulated metal height as the green data point is measured at day 23. Looking at Figure 3.7b, it is clear to see the height residual $h_m - \hat{h}_m$ decreases significant at this point.

Figure 3.8 shows the estimated and simulated states $\mathbf{x}$ as they are defined in the EKF. The estimate of the unmodeled dynamics is based on the same estimate as in Figure 3.7.



(a) Simulated and estimated volume.



(b) Simualted and estimated side ledge thickness.

Figure 3.10: In (a), the simulated and estimated volume of aluminum in the cell is shown. The simulated thickness of the side ledge is shown in (b). Simulated values are in blue while estimated values are in orange. The side ledge thickness is estimated from the HTP algorithm.

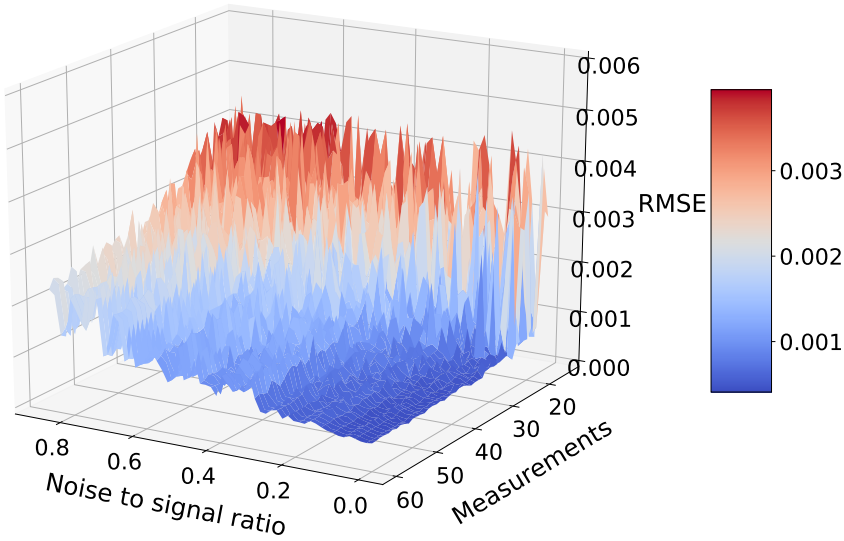
Figure 3.9 corresponds to Figure 3.7 and Figure 3.10 corresponds to Figure 3.8. The difference is that the estimate in Figure 3.9 and 3.10 is calculated by the HTP algorithm, while the estimates in Figure 3.7 and 3.8 is calculated by the QCBP program. The HTP algorithm estimates a signal with a much smaller RMSE than the QCBP algorithm for this specific case. Figure 3.6, shows that this is the case for the simulations where the correct support is found.

3.5 Conclusion

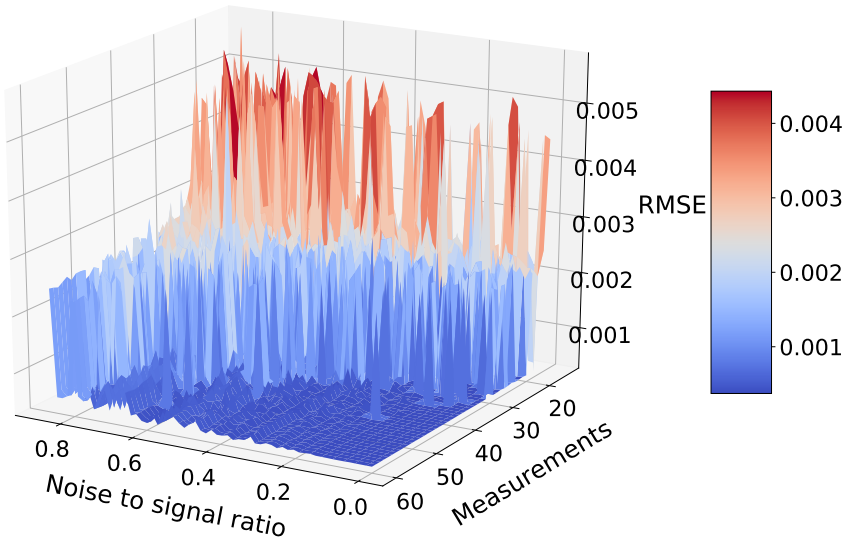
This chapter introduces a novel hybrid modeling method that addresses the problem of estimating stationary, periodical, unmodeled dynamics from a non-sparse, non-stationary signal measured at low sampling rates. The two research questions investigated have been satisfactorily addressed. These are pointed out below:

- The first research question sought an answer regarding the potential manipulation of the coarsely sampled signal so that compressed sensing techniques could be utilized for signal estimation. It was observed that most of the non-sparsity in the measured signal was due to the linear increase caused by the aluminum production and discontinuities due to sudden drops of the amplitude in the signal resulting from a regular tapping of the molten aluminum. This resulted in the measurement signal being non-sparse even in the frequency domain. Fortunately, the cause and effect of these linear increases and discontinuities were very well captured by the physics-based model. Simply subtracting the estimated metal height signal based on a physics-based model from the measured metal height signal yields manipulated measurements representing a new signal, namely the unmodeled dynamics of the metal height. As shown in Section 3.4.1, compressed sensing show promising results in estimating this residual signal from a limited number of measurements with Gaussian measurement noise.
- The second research question pertained to utilizing the estimated signal in a Kalman filter to improve the accuracy of the state estimation. The proposed method answered this by including the estimated signal as a pseudo measurement into the Kalman filter. The pseudo measurement is treated as a separate high-resolution measurement with corresponding Covariance matrices tuned according to the uncertainty about the state estimates. It was demonstrated that the state estimate of one of the variables improved significantly when a signal estimate of the unmodeled dynamics is available.

In the proposed approach, only stationary unmodeled dynamics are considered. This is because the compressed sensing techniques used in the method only consider stationary signals.

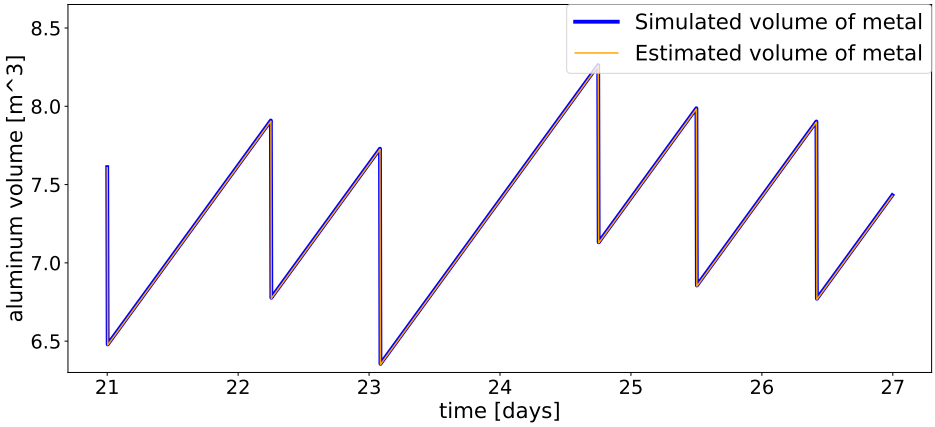


(a) RMSE for QCBP estimation technique

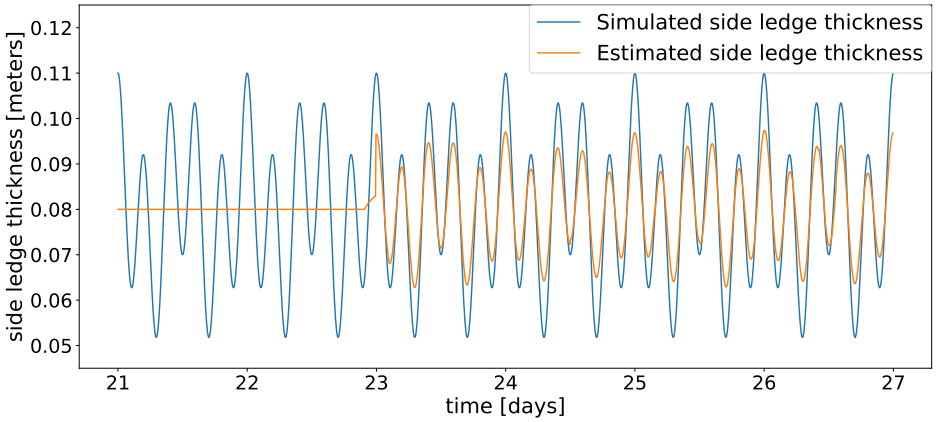


(b) RMSE for the HTP algorithm. Hyperparameter $\mu = 700$

Figure 3.6: RMSE for simulations with varying measurement noise and number of measurements used in estimation of the signals.

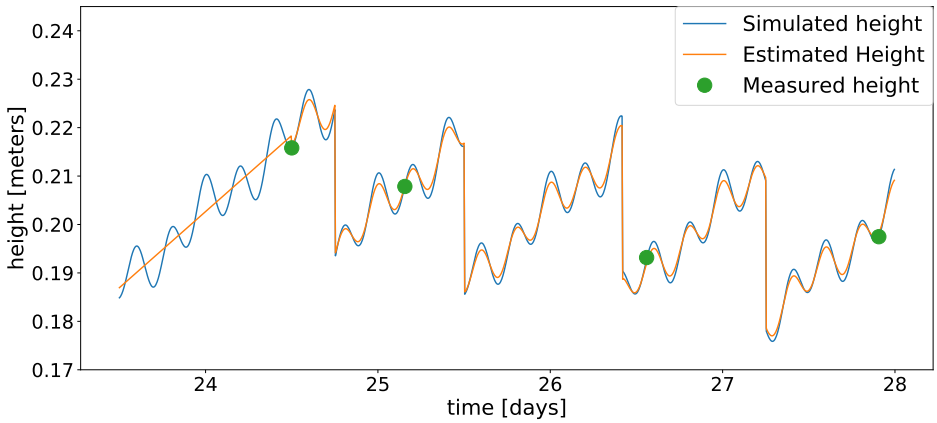


(a) Simulated and estimated volume.

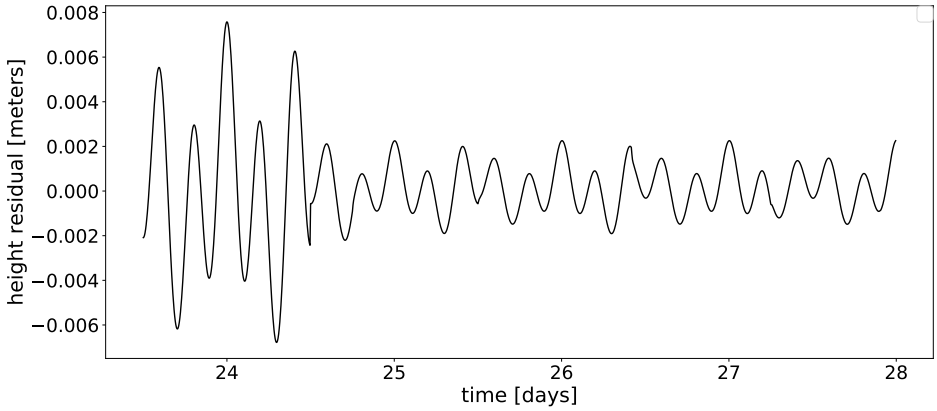


(b) Simualted and estimated side ledge thickness.

Figure 3.8: In (a), the simulated and estimated volume of aluminum in the cell is shown. The simulated thickness of the side ledge is shown in (b) Simulated values are in blue while estimated values are in orange. The side ledge thickness is estimated from the QCBP program.



(a) Simulated and estimated height.



(b) Height residual.

Figure 3.9: In (a), the blue graph represent the simulated metal height h_m, whereas the orange graph is the estimated metal height, estimated by the Kalman filter. The green points are the measured values of the metal height. In (b), the residual between simulated metal height h_m and Kalman filter-estimated metal height $\hat{h}_m$ is shown. The unmodeled dynamics in the metal height is estimated from the HTP algorithm.

Chapter 4

Hybrid modeling combining first principle model and deep learning

This chapter is based on:

[79] H. Robinson[1], **E. T. B. Lundby**[1], A. Rasheed, J. T. Gravdahl, "A novel corrective-source term approach to modeling unknown physics in aluminum extraction process" Conditional Acceptance: Engineering Applications of Artificial Intelligence. (2023), Available in: arXiv preprint arXiv:2209.10861 (2022).

It presents a hybrid modeling approach, where a DNN aims to correct a mispecified PBM formulated as a high-dimensional set of ODEs. The DNN is added to the set of ODE and trained on the residual data between measurements and PBM estimates.

4.1 Introduction

State-of-the-art modeling of aluminum electrolysis used in decision support and state estimation is mainly based on first principle PBMs. As mentioned earlier, these models are known for their comprehensible and interpretable nature, their ability to generalize well to situations where assumptions hold true, and the existence of established methods for evaluating their properties such as robustness and stability in the presence of uncertainty and noise. But the process is characterized by several nonlinear sub–processes that interact with each other. While some of

[1] Equal contributions

the sub–process are based on solid theoretical background and sound assumptions, other sub-processes are more challenging to model and can be seen as disturbances in the process. The strong interaction effects between sub–processes imply that modeling errors in one sub–process propagate to other sub–processes and eventually cause significant uncertainty in state estimates and future predictions. DDMs have the ability to model complex phenomena directly from data. However, these methods typically require vast and diverse data, which is typically not available from regular process operation and very expensive to acquire. Moreover, DDMs struggle to generalize to unseen data and typically have low interpretability.

Recent developments in the field of hybrid modeling have drawn attention to the potential benefits of combining the strengths of both first principle PBMs and DDMs. In this study, we examine the effectiveness of the CoSTA, which involves adjusting the output of a discretized PBM by utilizing a DDM that has been trained to correct the errors of the base model. This method is an intuitive and efficient way to leverage existing models, and have been used in different ways. For example, in [35], it was demonstrated that CoSTA works for simple, one-dimensional heat transfer problems. In [112], the same work was extended to 2D and was also demonstrated that the CoSTA model has inbuilt sanity check mechanism. The method use prior knowledge to make the learning problem more simple in terms of complexity and also potentially reduce dimensionality by substracting PBM estimates from the training data.

In this work, we extend and apply CoSTA to correct a misspecified PBM of a complex aluminum extraction process simulation. We build on previous work by:

- Extending CoSTA to multidimensional problems: The previous works utilizing CoSTA were limited to modeling a single state temperature in either one or two-dimensional heat transfer.
- Successfully applying CoSTA to a system with external control inputs: None of the previous work involved any control inputs. In the current work, five inputs are used to excite the system.
- Investigating the predictive stability of CoSTA relative to end-to-end learning, and showing that a hybrid approach can yield more trustworthy models.
- Demonstrating that CoSTA is applicable to a system with complex coupling between different states and inputs: The system considered here involves eight states and five inputs which form a set of eight ordinary equations which are highly coupled. The previous works involving heat transfer involved only one partial differential equations hence the potential of CoSTA to couple problems was never evaluated earlier.

In the case study, the novel method is applied on the aluminum electrolysis simulator described in Section 2.1.2.

This chapter is structured as follows. Section 4.2 presents the CoSTA approach applied. We then outline the methodology of the work in Section 4.3, namely how the data was generated, how the models were trained, and how they were evaluated. In Section 4.4 we present the results, and give a detailed discussion about the behavior of the process and the models. We then summarise our findings and outline future work in Section 4.5.

4.2 Corrective source term approach (CoSTA)

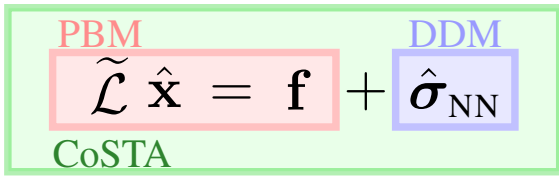


Figure 4.1: CoSTA combines PBM and DDM into a unified model by adding a NN-generated corrective source term to the governing equation of the PBM.

In this section we outline the CoSTA approach, illustrated in Figure 4.1. Suppose we want to solve the following general problem:

$$\mathcal{L}\mathbf{x} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{4.1}$$

where $\mathcal{L}$ is a differential operator, $\mathbf{x}$ is the unknown state of the system that we wish to compute, and $\mathbf{f}(\cdot,\cdot)$ is a source term that depends on the state $\mathbf{x}$ and external inputs $\mathbf{u}(t)$.

Assume now that we have a PBM designed to predict $\mathbf{x}$, and let $\tilde{\mathbf{x}}$ denote the PBM's prediction of the true solution $\mathbf{x}$. If $\tilde{\mathbf{x}} \neq \mathbf{x}$, there is some error in the PBM, and this error must stem from at least one of the following misspecifications in the model:

1. Incorrect $\mathbf{f}$ in Equation (4.1), replaced by $\tilde{\mathbf{f}}$.
2. Incorrect $\mathcal{L}$ in Equation (4.1), replaced by $\widetilde{\mathcal{L}}$.
3. A combination of the above.
4. Discretization of $\mathcal{L}$, replaced by $\mathcal{L}_{\mathcal{D}}$[1].

[1]Derived using, for example, finite differences. This is necessary when Equation (4.1) lacks analytical solutions, which is almost always the case.

Note that case 4 is also mathematically equivalent to misspecifying $\mathcal{L}$. For example, $\frac{\partial}{\partial t}$ could be approximated using a forward finite difference. We can write this using the difference operator Δ_h, such that h is the time step and $\frac{1}{h}\Delta_h f(t) = (f(t+h)-f(t))/h$. We can therefore limit our discussion to Cases 1 and 2 without loss of generality.

Suppose now that the PBM-predicted solution $\tilde{\mathbf{x}}$ is given as the solution of the following system:

$$\widetilde{\mathcal{L}}\tilde{\mathbf{x}} = \tilde{\mathbf{f}} \tag{4.2}$$

This formulation encompasses both Case 1 ($\widetilde{\mathcal{L}} = \mathcal{L}$ and $\tilde{\mathbf{f}} \neq \mathbf{f}$), Case 2 ($\widetilde{\mathcal{L}} \neq \mathcal{L}$ and $\tilde{\mathbf{f}} = \mathbf{f}$), and combinations thereof (for $\widetilde{\mathcal{L}} \neq \mathcal{L}$ and $\tilde{\mathbf{f}} \neq \mathbf{f}$). Furthermore, suppose we modify the system above by adding a source term $\hat{\sigma}$ to Equation (4.2), and let the solution of the modified system be denoted $\hat{\tilde{\mathbf{x}}}$. Then, the modified system reads

$$\widetilde{\mathcal{L}}\hat{\tilde{\mathbf{x}}} = \tilde{\mathbf{f}} + \hat{\boldsymbol{\sigma}} \tag{4.3}$$

and the following theorem holds.

Theorem Let $\hat{\tilde{\mathbf{x}}}$ be a solution of Equations (4.3), and let $\mathbf{x}$ be a solution of Equations (4.1). Then, for both operators $\widetilde{\mathcal{L}}$, $\mathcal{L}$ and both functions $\mathbf{f}$, $\tilde{\mathbf{f}}$, such that $\hat{\tilde{\mathbf{x}}}$ and $\mathbf{x}$ are uniquely defined, there exists a function $\boldsymbol{\sigma}$ such that $\hat{\tilde{\mathbf{x}}} = \mathbf{x}$.

Proof: Define the residual $\boldsymbol{\sigma}$ of the PBM's governing equation (4.2) as[2]

$$\boldsymbol{\sigma} = \widetilde{\mathcal{L}}\mathbf{x} - \tilde{\mathbf{f}}. \tag{4.4}$$

If we set $\hat{\boldsymbol{\sigma}} = \boldsymbol{\sigma}$ in Equation (4.3), we then obtain

$$\begin{aligned} \widetilde{\mathcal{L}}\hat{\tilde{\mathbf{x}}} &= \tilde{\mathbf{f}} + \hat{\boldsymbol{\sigma}} && (4.5)\\ &= \tilde{\mathbf{f}} + \widetilde{\mathcal{L}}\mathbf{x} - \tilde{\mathbf{f}} && (4.6)\\ &= \widetilde{\mathcal{L}}\mathbf{x} && (4.7)\\ \implies \quad \hat{\tilde{\mathbf{x}}} &= \mathbf{x} + \mathbf{c} && (4.8) \end{aligned}$$

where $\mathbf{c}$ is a function of independent variables. We can eliminate $\mathbf{c}$ by setting appropriate boundary conditions. ■

[2]Instead of defining the residual in terms of the approximate solution (e.g. as is done in truncation error analysis [113, chapter 8]), we define $\boldsymbol{\sigma}$ by inserting the true solution into Equation (4.1). Our proof is simpler, and fit well with real systems where state measurements are more readily available than the true governing equations.

This shows that we can always find a corrective source term $\hat{\boldsymbol{\sigma}}$ that compensates for any error in the PBM's governing equation (4.2) such that the solution $\hat{\tilde{\mathbf{x}}}$ of the modified governing equation (4.3) is equal to the true solution $\mathbf{x}$. This observation is the principal theoretical justification of CoSTA. As illustrated by Figure 4.2, the CoSTA approach should be applicable to many physical problems that can be described using differential equations.

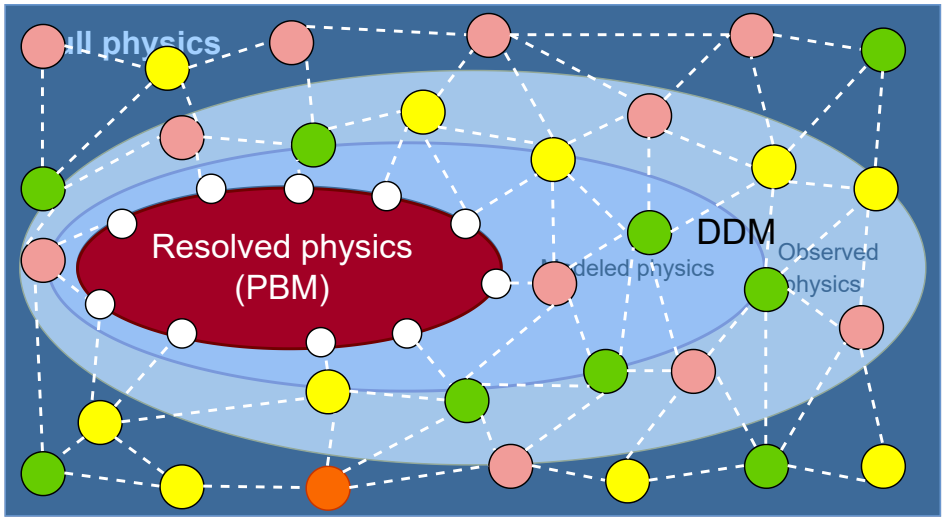


Figure 4.2: CoSTA: It maximizes the utilization of the well known PBM while correcting for the unknown using DDM. In the CoSTA, PBM is described by the set of differential equations describing the state of the system after the introduction of errors as explained in Section 4.3.2. We call it resolved physics and represent it by the red ellipse. All the unaccounted physics and unintended numerical discretization errors are captured using the data-driven corrective source terms.

4.3 Method and experimental setup

This section explains how data is generated, how modeling errors are induced in the PBM, the modeling approaches applied in the case study, and how the models were evaluated.

4.3.1 Inducing error in the PBM

For this case study, we use simulation data generated from Equation (2.15) in order to validate the CoSTA method. To that end, we make a further simplification: we ignore Equation (2.14a) and set the liquidus temperature g_1 to a constant.

$$g_{1,PBM} = 968^{\circ}C. \tag{4.9}$$

We refer to the resulting model as the *ablated PBM*. This choice was made because the model is particularly sensitive to errors in g_1. Inspecting Equation (2.15) shows that the ablated PBM will incorrectly predict the evolution of $[x_1, x_4, x_6, x_7, x_8]$. As we will see later in Section 4.4 and Figure 4.6, this can lead to errors of roughly $5°\text{C}$ in g_1, and 500kg in the side ledge mass x_1 (a relative error of 10%). The aim of the case study is to develop a DDM to correct the ablated PBM using measurement data sampled from the true model.

4.3.2 Data generation and preprocessing

The dynamical system data is generated by integrating the set of non-linear ODE's in Equation (2.15) representing the system dynamics using the fourth-order numerical integrator Runge-Kutta 4 (RK4) with a fixed timestep $\Delta T = 10s$. One time-series simulation starts at an initial time t_0 with a set of initial conditions $\mathbf{x}(t_0)$, and last until a final time $T = 5000 \times \Delta T$. For the slow dynamics of the aluminum process, a sampling time of $10s$ turns out to be sufficiently fast with negligible integration errors. Higher sampling frequencies would lead to unnecessary high computational time and large amounts of simulation data. The initial conditions for each trajectory were uniformly sampled from the ranges shown in Table 4.1. Each simulation generates a set of trajectories consisting of 8 states and 5 inputs. 40 simulated trajectories are used for training the models, and 100 simulated trajectories are used as the test set. This relatively large number of test cases was to chosen to allow us to explore the statistics of how the model performs.

Figure 4.3 shows the time series evolution of the entire training set and test set. The training set trajectories are blue while the test set trajectories are orange. The figures show that the range of the training set covers the range of the test set. This indicates that models are evaluated on interpolation cases in the test set.

Table 4.1: Initial conditions for system variables. For x_2 and x_3, concentrations c_{x_2} and c_{x_3} are given.

Variable	Initial condition interval
x_1	$[2060,\ 4460]$
c_{x_2}	$[0.02,\ 0.05]$
c_{x_3}	$[0.09,\ 0.13]$
x_4	$[11500,\ 16000]$
x_5	$[9550,\ 10600]$
x_6	$[940,\ 990]$
x_7	$[790,\ 850]$
x_8	$[555,\ 610]$

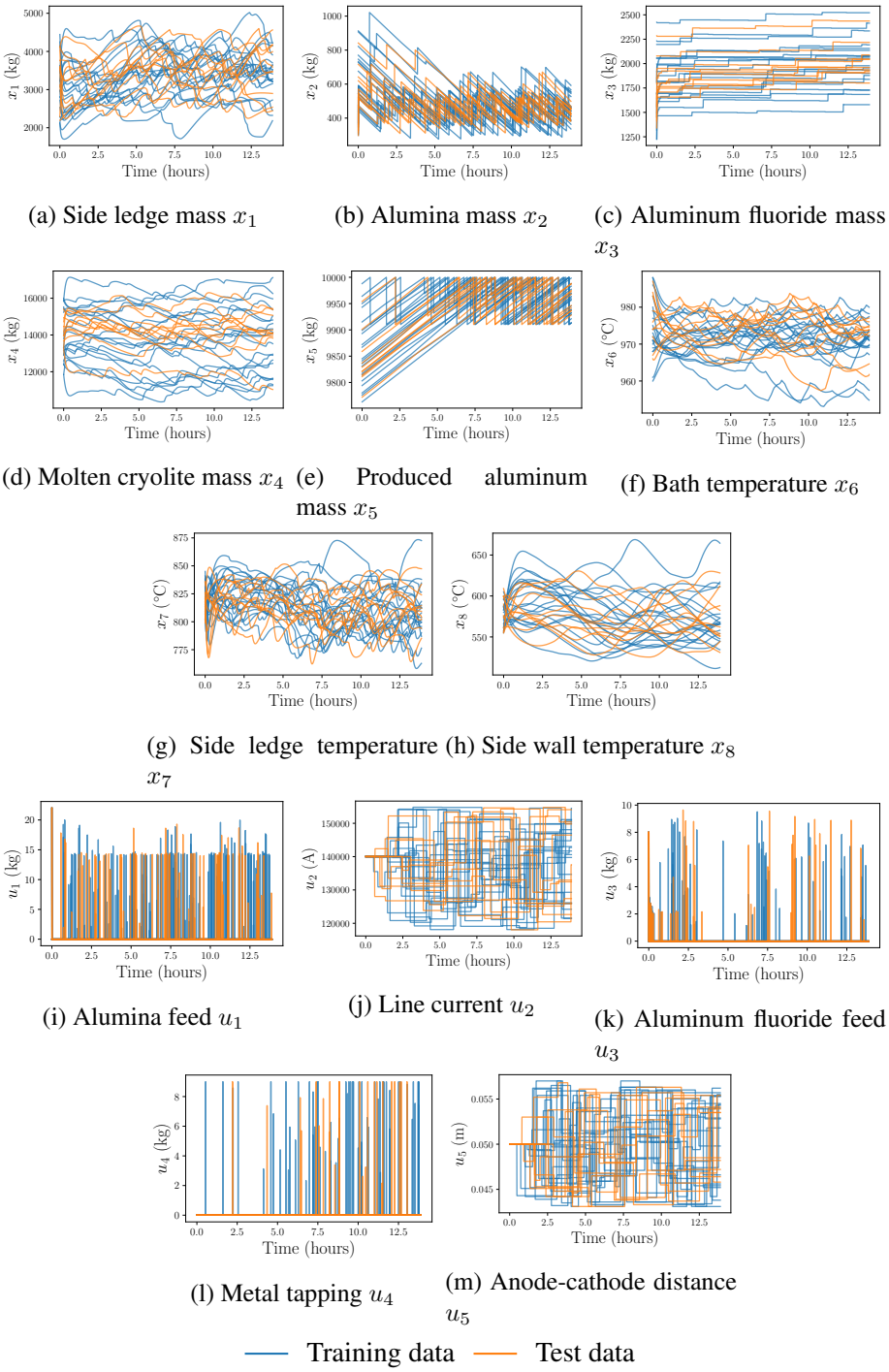


Figure 4.3: Training and test set trajectories of the system states. Only 10 random sample test trajectories are shown here to make the figures clearer.

Estimation of the regression variable $\dot{\mathbf{x}}$

The ODEs in Equation (2.15) are time invariant. This means that at time $k+1$, $\dot{\mathbf{x}}_{k+1}$ in general only depends on the current state and input $(\mathbf{x}_k, \mathbf{u}_k)$ at time k. In other words, the system has the Markov property. Therefore, the datasets are listed in pairs $\mathcal{D} = \{(\mathbf{x}_k, \mathbf{y}_k)\} = \{(\mathbf{x}_k, \mathbf{u}_k), \dot{\mathbf{x}}_k\}$. This does not always hold in practice, and the state vector must therefore be augmented with additional information, i.e. lookback states from previous time steps. Takens' Theorem gives an upper bound on the number of necessary lookback states [114]. The time derivatives at time k are estimated as the forward difference $\dot{\mathbf{x}}_k = (\mathbf{x}_{k+1} - \mathbf{x}_k)/h$, where h is the time step. In this work we use $h = 10\text{s}$. This numerical derivative induces a discretization error. However, since the dynamics of the aluminum electrolysis is slow, this error is considered negligible. In a pure DDM approach, the datasets are listed in pairs as follows:

$$\mathcal{D}_{\text{DDM}} = \{((\mathbf{x}_k, \mathbf{u}_k), \dot{\mathbf{x}}_k)\}_1^N, \tag{4.10}$$

with N training pairs. In other words, the DDM aims to map $(\mathbf{x}_k, \mathbf{u}_k)$ to $\dot{\mathbf{x}}_k$. For the corrective source term model presented in Equation (4.3), the target variable is the error of the PBM model:

$$\mathcal{D}_{\text{CoSTA}} = \{(\mathbf{x}_k, \mathbf{y}_{k,\,\text{CoSTA}})\} = \{(\mathbf{x}_k, \mathbf{u}_k), (\dot{\mathbf{x}}_k - \hat{\dot{\mathbf{x}}}_{k,\text{PBM}})\}_1^N. \tag{4.11}$$

Input signal generation

While machine learning models are extremely useful for function approximation and interpolating data, they naturally do not always extrapolate properly and are highly dependent on on the quality and variety of the data that they are trained on. Due to this it is vital that the training data covers the intended operational space of the system. Here, the operational space means the region of the state space which the system operates in, meaning state and input vectors $[\mathbf{x}^T, \mathbf{u}^T]^T$ observed over time. The data should capture the different nonlinear trends of the system covered by the operational space. For systems *without exogenous inputs*, variation can only be induced by simulating the system with different initial conditions $\mathbf{x}(t_0)$. For systems *with exogenous inputs*, the initial conditions are generated in the same way. Moreover, the input vector $\mathbf{u}$ will excite the system dynamics. The aluminum process has a feedback controller that ensures safe and prescribed operation. However, operational data from a controlled, stable process is generally characterized by a low degree of variation which is insufficient for effective system identification. A well-known convergence criterion for the identification of linear time-invariant systems is Persistency of Excitation (PE). A signal $\mathbf{x}(t_k)$ is PE of order L if all sub-sequences $[\mathbf{x}(t_k), \ldots, \mathbf{x}(t_k + L)]$ span the space of all possible

sub-sequences of length L that the system is capable of generating. While the PE criterion is not directly applicable to nonlinear systems, sufficient coverage of the dynamics is required for successful system identification [115, 116].

To push the system out of its standard operating conditions, we add random perturbations to the control inputs. In general, each control input i is given by:

$$u_i = \text{Deterministic term + Random term.} \tag{4.12}$$

The control inputs $u_1,\ u_3$ and u_4 are impulses. The random term is zero for these control inputs when the deterministic term is zero. The deterministic term is a proportional controller. The control inputs u_2 and u_5 are always nonzero. These control inputs have constant deterministic terms and a random term that changes periodically. The random term stays constant for a certain period ΔT_{rand} before changing to a new randomly determined constant. Choosing the period ΔT_{rand} is a matter of balancing different objectives. On the one hand, it is desirable to choose a large period ΔT_{rand} so that the system can stabilize and evolve under the given conditions to reveal the system dynamics under the given conditions. On the other hand, it is desirable to test the systems under many different operational conditions. By empirically testing different periods ΔT_{rand}, and seeing how the dynamics evolve in simulation, it turns out that setting $\Delta T_{rand} = 30\Delta T$ is a fair compromise between the two. In this study, we generate the random disturbances using the Amplitude-modulated Pseudo-Random Binary Signal (APRBS) method [117]. Table 4.2 gives the numerical values of the deterministic term of the control

Table 4.2: Equations used to control the aluminum process

Input	Deterministic term	Random term interval	ΔT_{rand}
u_1	$3 \cdot 10^4(0.023 - c_{x_2})$	$[-2.0, 2.0]$	ΔT
u_2	$1.4 \cdot 10^4$	$[-7 \cdot 10^3, 7 \cdot 10^3]$	$30 \cdot \Delta T$
u_3	$1.3 \cdot 10^4(0.105 - c_{x_3})$	$[-0.5,\ 0.5]$	ΔT
u_4	$2(x_5 - 10^4)$	$[-2.0,\ 2.0]$	ΔT
u_5	0.05	$[-0.015,\ 0.015]$	$30 \cdot \Delta T$

input, the interval of values for the random terms, and the duration ΔT_{rand} of how long the random term is constant before either becoming zero (u_1, u_3, u_4) or changing to a new randomly chosen value (u_2, u_5).

4.3.3 Modeling approaches

The case study compares three different modeling approaches, namely a PBM approach using an ablated PBM, the hybrid modeling approach CoSTA - combining

the ablated PBM with a DNN, and a purely DDM approach - modeling the entire set of ODEs with DNNs. Comparing the CoSTA approach with a PBM and a DDM approach will reveal the effect of the CoSTA approach.

Ablated PBM

As previously discussed, we are interested in modeling scenarios where the PBM does not capture the full underlying physics of the system. For this case study, the PBM is described by Equation (2.15), with modifications to induce modeling errors so that the PBM deviates from the simulation model. That is, Equation (2.14a) describing the liquidus temperature g_1 was ignored, and g_1 was set to a constant, as shown in Equation (4.9). The resulting model, which ignores the dynamics of the liquidus temperature, is referred to as the *ablated PBM*, or PBM. The PBM used in the case study is a set of ODEs on the general form:

$$\hat{\dot{\mathbf{x}}}_k = \mathbf{f}_{\text{PBM}}(\mathbf{x}_k,\ \mathbf{u}_k), \tag{4.13}$$

where $\hat{\dot{\mathbf{x}}}_k$ is the PBM estimate of the time derivative at timestep k, $f_{\text{PBM}}(\cdot,\ \cdot)$ is the ablated PBM, and $\mathbf{x}_k$ and $\mathbf{u}_k$ are state variables and control inputs at timestep k.

DDM

The DDM approach models the time derivative of the state, and has the following form:

$$\hat{\dot{\mathbf{x}}}_k = \mathbf{f}_{\text{DDM}}(\mathbf{x}_k,\ \mathbf{u}_k). \tag{4.14}$$

In the case study, the DDM $\mathbf{f}_{\text{DDM}}(\cdot,\ \cdot)$ is a DNN. The DNN is trained on the training set $\mathcal{D}$ in Equation (4.10).

CoSTA

The case study aims to develop a DDM to correct the ablated PBM using measurement data sampled from the true model. The resulting hybrid model CoSTA presented on a general form in Equation (4.3) consists of the PBM in Equation (4.13) and a DDM that is meant to correct the misspecified PBM. The CoSTA in this case study used to model the set of ODEs in Equation (2.15) is given by:

$$\hat{\dot{x}}_k = \mathbf{f}_{\text{CoSTA}}(\mathbf{x}_k,\ \mathbf{u}_k) = \mathbf{f}_{\text{PBM}}(\mathbf{x}_k,\ \mathbf{u}_k) + \mathbf{f}_{\text{corr}}(\mathbf{x}_k,\ \mathbf{u}_k). \tag{4.15}$$

$\mathbf{f}_{\text{corr}}$ is a DNN that aims to correct the errors in the PBM. The parameters of the corrective source term $\mathbf{f}_{\text{corr}}$ are learned through training, using the manipulated training set in Equation (4.11).

4.3.4 Training

The PBM in CoSTA will typically reduce the complexity of the learning problem compared to the learning problem of a pure DDM approach. In light of this, it is interesting to see the effects of sparsity promoting ℓ_1 regularization, which is known to lead to sparser models and better generalization with less available data, while maintaining model accuracy. The case study includes two versions of the DNN in the CoSTA, namely:

- Dense $\mathbf{f}_{\text{corr}}$
- Sparse $\mathbf{f}_{\text{corr}}$

where $\mathbf{f}_{\text{corr}}$ is the corrective source term in Equation (4.15). In order to compare the results with a purely DDM approach, the same two versions of complexity are used in the DDM approach in Equation (4.14), that is:

- Dense $\mathbf{f}_{\text{DDM}}$
- Sparse $\mathbf{f}_{\text{DDM}}$

The architecture of all networks, both in CoSTA and the DDM approach, was $[13, 20, 20, 20, 20, 8]$ (13 inputs, 8 outputs, 4 hidden layers with 20 neurons each). The `ReLU` activation function was used for all layers except the output layer, which had no activation function. The same architecture was used for all networks for a fairer comparison. The models were trained on the training set using the total-loss function shown in Equation (2.26), where the loss function $L(\cdot, \cdot)$ is the MSE as shown in Equation (2.27). The ADAM optimizer, a popular SGD method with adaptive learning rates proposed by [118], was used with the following default parameters: Initial learning rate $\eta = 10^{-3}$, Gradient forgetting factor $\beta_1 = 0.9$, and Gradient second-moment forgetting factor $\beta_2 = 0.999$. The dense networks were trained with $\lambda = 0$, and sparse networks with $\lambda = 10^{-4}$, where λ is the ℓ_1 regularization parameter. All models were trained for 100 epochs (an epoch is defined as one full pass over the dataset). This value was chosen empirically during initial training attempts by inspecting training loss curves and selecting a rough average of optimal training times. An alternative would be *early stopping*, where training is terminated when performance on an additional validation dataset begins to drop. However, [119] and [120] showed that early stopping could have an additional regularizing effect by constraining the parameter space. Therefore, early stopping was not used to maintain similar training conditions over all experiments.

4.4 Results and discussion

As described in Section 4.3.2, the test set consists of 100 simulated trajectories, each with a length of 5000 timesteps, with a timestep of $\Delta T = 10sec$. Each model is used to perform a rolling forecast given the initial conditions and input signal of each test trajectory. The experiments were repeated 10 times with different random initializations of the trainable parameters to increase the statistical significance of the results. That is, each of the four model types consisting of DNNs has 10 model instances. The 4 model types evaluated in the case study that consists of DNNs are the dense and sparse DNN model structures in the CoSTA("CoSTA dense" and "CoSTA sparse" in Figure 4.4 and Figure 4.5), and dense and sparse DNN model structures in the DDM approach ("DDM dense" and "DDM sparse" in Figure 4.4 and Figure 4.5). These model structures are described in Section 4.3.4. The last model type that was evaluated, namely the ablated PBM has no trainable parameters, and remained the same in all experiments. Hence, the PBM has only 1 model instance.

Figure 4.4 shows violin plots of the AN-RFMSE values defined in Equation (2.46) for all model types. The AN-RFMSE violin plots are shown at three different prediction horizons to demonstrate the models' short-term, medium-term, and long-term performance. The AN-RFMSE values included in constructing a single violin plot for a given horizon and model type are based on the forecasts of the test-set trajectories up until the given forecasting horizon, and blow-ups are omitted as outliers. A possible issue here is that excluding these AN-RFMSE values favors the models that have many blow-ups. However, the results also show high blow-up rates correlate with high AN-RFMSE. Figure 4.5 shows the frequency of blow-ups for each model type, based on the blow-up measure defined in Equation (2.47).

These results show that, on average, all DDM and CoSTA models have a lower AN-RFMSE than the ablated PBM in the short and medium term. However, all DDM and CoSTA models experience some blow-ups in the long term, which the PBM model does not. The dense DDM fared the worst, with 27.3% of long-term forecasts blowing up. The sparse DDM marginally improves on the AN-RFMSE, but we found that the blow-up rate was significantly reduced in the long term compared to the dense DDM. Both dense and sparse CoSTA models were significantly more accurate than the DDM models. The sparse CoSTA had similar accuracy to the dense CoSTA models in the short and medium term. However, the sparse CoSTA model had no blow-ups in the short and medium term and had half the blow-up rate of the Sparse DDM in the long term. These experiments demonstrate that CoSTA can reliably correct misspecified PBMs and significantly improves predictive stability compared to end-to-end learning. The base PBM does

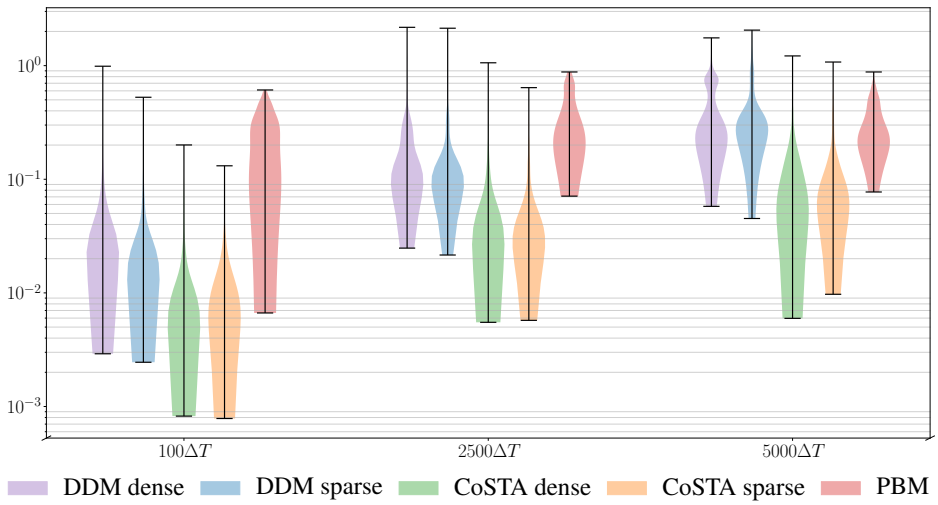


Figure 4.4: Violin-plot of the AN-RFMSE for all model types for 100 different initial conditions and inputs signals. The width of the bar reflects the distribution of the data points, and the error bars represent the range of the data. The error is shown after 3 different times to compare the performance in the short, medium, and long-term. We trained 10 different instances for each model type for statistical significance. We see that CoSTA improves the predictive accuracy over the whole trajectory. Introducing sparse regularization appears to improve performance for DDM, but only appears to affect CoSTA models in the long term, where sparse CoSTA appears to have less variance.

not exhibit any blow-up issues, suggesting that the blow-ups can be attributed to the NNs used in this work. If long-term forecasts are required (> 3000 timesteps), we recommend combining the CoSTA approach with a sanity check mechanism to detect potential blow-ups.

Figure 4.4 and Figure 4.5 summarize the main results of the case study as they include forecasts of all test-set trajectories. To illustrate and elaborate on the effect of the results when forecasting without feedback from measurements, we have included plots of forecasts of a single, representative test trajectory. Figure 4.6 shows the mean predictions for each model type for the representative test trajectory, along with a 99.7% confidence interval to show the spread of the predictions from the 10 instances of each model type. Only the DDM and CoSTA models trained with ℓ_1 regularization are shown for clarity. Before discussing the differences between the models, we will describe the system's dynamics and how the incorrect PBM behaves in comparison. First, note that all variables are non-negative,

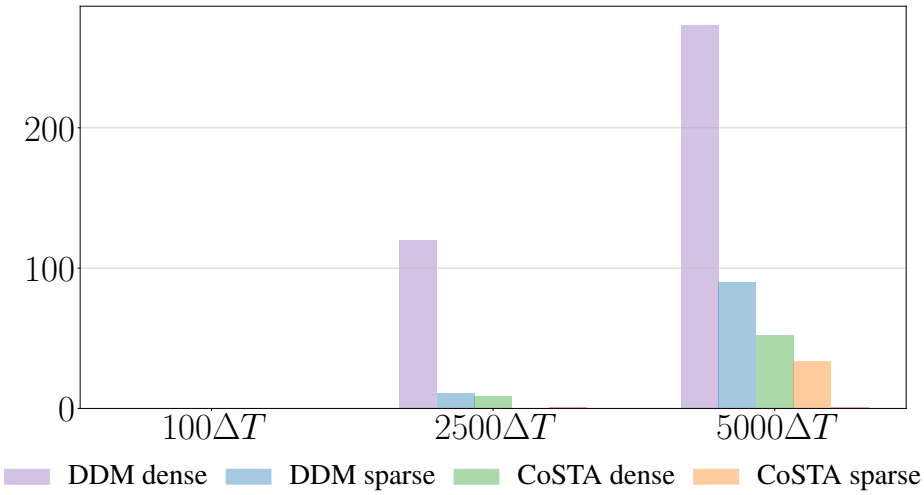


Figure 4.5: Bar chart of the number of times model estimates blow up and diverges. The plot is for all model types for 100 different initial conditions and input signals. The number of blow-ups was counted after 3 different times to compare the performance in the short, medium, and long-term. We trained 10 different instances for each model type for statistical significance. We see that applying CoSTA greatly increases the predictive stability in the long term. That is, the number of blow-ups for CoSTA models is far less than the number of blow-ups for DDM. However, PBM does not suffer from significantly fewer blow-ups than CoSTA.

reflecting different physical quantities in the system, i.e., mass, temperature, and current. Inspecting Equation (2.15), we see that the states x_2, x_3, and x_5 are linearly dependent on u_1, u_2, u_3, and u_4. We refer to these as the *linear states*, and the rest as the *nonlinear states*.

Liquidus temperature g_1: Figure 4.6i shows the true liquidus temperature g_1 (in black) and the constant PBM estimate of the liquidus temperature (in red dotted line). The liquidus temperature g_1, which is the temperature at which the bath solidifies, is determined by the chemical composition of the bath. That is, g_1 is determined by the mass ratios between x_2, x_3, and x_4. The fact that PBM assumes g_1 to be constant induces modeling errors for the PBM.

Mass of side ledge x_1: Figure 4.6a shows the mass of frozen cryolite ($Na_3\,Al\,F_6$), or side ledge. The solidification rate $\dot{x}_1$ is proportional to the heat transfer Q_{liq-sl} through the side ledge ($Q_{liq-sl} \sim \left(\frac{g_1 - x_7}{x_1}\right)$) minus the heat transfer $Q_{bath-liq}$ between the side ledge and the bath ($Q_{bath-liq} \sim (x_6 - g_1)$). The solidification rate $\dot{x}_1$ is dependent on the value of g_1, and therefore the PBM incorrectly predicts the mass rate $\dot{x}_1$. In Figure 4.6a we see that the PBM modeling error for x_1 starts to increase after approximately one hour. This is simultaneously as the true liquidus temperature g_1 drifts away from the constant PBM estimate of g_1, see Figure 4.6i. As we can see, the PBM overestimates g_1. Therefore, the PBM also overestimates the heat transfer out of the side ledge, leading to an overestimate of the amount of cryolite that freezes, and hence an overestimate of the increase in side ledge mass. However, this modeling error is limited by the effect that an increased side ledge mass (and therefore increased side ledge thickness) leads to better isolation. Thus, the PBM estimate of the heat transfer through the side ledge Q_{liq-sl} is inversely proportional to the x_1 estimate, and the modeling error of x_1 reaches a steady state for a constant modeling error in g_1. In addition to modeling errors due to errors in the g_1 estimate, modeling errors of x_6 and x_7 propagates as modeling errors in $\dot{x}_1$.

Both the mean of DDM and the mean of CoSTA models appear to correctly predict the response of x_1. However, both model classes show a growing spread. While the error spread of both model classes appears to grow over time, the DDM error grows roughly twice as fast. Furthermore, both CoSTA and DDM show some cases where the error bound becomes significantly large, meaning that one or more of the models fail. For the DDM models, these cases appear more frequently, and the errors are larger than for the CoSTA models. Figures 4.6f and 4.6g shows that these error peaks often coincide with the peaks in the bath temperature x_6 and the side ledge temperature x_7.

Mass of alumina x_2: Figure 4.6b shows the mass of aluminum in the bath. Equation (2.15) shows that $\dot{x}_2$ (mass rate of Al_2O_3) is proportional to u_1 (Al_2O_3 feed), and negatively proportional to u_2. Figure 4.6b shows that this yields a saw-tooth response that rises as u_1 spikes, and decays with a rate determined by u_2. This state has no dependence on g_1, nor any dependence on other states that depends on g_1. Therefore the PBM (and CoSTA) predict this state with no error. On the other hand, the spread of the DDM models grows over time, with the mean error eventually becoming significant.

Mass of aluminum fluoride x_3: The x_3 state (mass of AlF_3) acts as an accumulator, rising when AlF_3 is added to the process (u_3 spikes), and falling when Al_2O_3 is added to the process (u_1 spikes). The latter is caused by impurities (Na_2O) in the Alumina (Al_2O_3) reacting with AlF_3, generating cryolite (Na_3AlF_6). As can be seen in Figure 4.6c, the latter effect is relatively small. Despite this, the DDM appears to correctly model these decreases. However, the DDM models become less and less accurate as time goes on. The PBM and CoSTA model x_3 with no error.

Mass of molten cryolite x_4: This state represents the mass of molten cryolite in the bath, where $\dot{x}_4 = k_5 u_1 - \dot{x}_1$. The first term represents additional cryolite generated by reactions between impurities in the added alumina (u_1) and AlF_3 (x_3). The second term describes how the cryolite can freeze (x_1) on the side ledge, which can melt again as the side-ledge temperature x_7 increases. As can be seen in Figure 4.6d the response of x_4 therefore mirrors that of x_1, with relatively small upturns when alumina is added (u_1). Inspecting Figure 4.6a, we see that the models have essentially identical behavior. Incorrectly estimating x_4 causes some issues. The mass ratio c_{x_2} (see Equation (2.13)) is important in terms of determining the cell voltage U_{cell}. A forecasting error of x_4 will propagate as a forecasting error of c_{x_2}, leading to inaccurate estimates of the cell voltage U_{cell}. This is elaborated when discussing the bath temperature x_6.

Mass of produced metal x_5: This linear state also has a saw-tooth characteristic, growing at a rate proportional to the line current (u_2), and falling when metal is tapped (u_4 spikes). Looking at Figure 4.6e, the DDM models have similar error dynamics to the other linear states, while the PBM and CoSTA models have virtually no error.

Temperature in the bath x_6: There are several possible sources of PBM modeling errors of the bath temperature x_6. As discussed earlier, since the PBM overestimates the side ledge thickness due to a modeling error of g_1, it follows that the PBM overestimates the thermal insulation of the side ledge. This leads to an overestima-

tion of the bath temperature, as the heat transfer out of the bath is underestimated. In Figure 4.6f, we see this overestimate of x_6 provided by the PBM after approximately one hour, simultaneously as the PBM starts to overestimate the side ledge mass x_1.

Furthermore, the change in bath temperature $\dot{x}_6$ is determined by the energy balance in the bath. The energy balance in the bath consists of several components, namely the electrochemical power P_{el} which adds energy to the system, the heat transfer from the bath to the side ledge $Q_{bath-sl}$ which transports energy out of the bath, and the energy $E_{tc,liq}$ required to break inter-particle forces in the frozen cryolite liquidus temperature. The electrochemical power $P_{el} = U_{cell} \cdot u_2$ is the product of the cell voltage U_{cell} and the line current u_2. The cell voltage is given by $U_{cell} = \left(g_5 + \frac{u_2 u_5}{2620 g_2}\right)$, where g_5 is the bubble voltage drop, and $\frac{u_2 u_5}{2620 g_2}$ is the voltage drop due to electrical resistance in the bath. The bubble voltage drop g_5 increases exponentially when the operation gets close to an anode effect. Anode effects occurs when the mass ratio of alumina - c_{x_2} is reduced to the a critical mass ratio of alumina $c_{x_2,crit} \sim 2$. This can explain overestimate error peaks in the x_6 estimate, which are most present for the DDM models. As we can see in Figure 4.6f the peaks of the error band for the DDM happens simultaneously with overestimates of x_4 (see Figure 4.6d), indicating that the DDM wrongly predict anode effects in these cases. Moreover, the voltage drop due to electrical resistance is given by $\frac{u_2 u_5}{2620 g_2}$, where u_2 is the line current, u_5 is the Anode-Cathode Distance (ACD), $2620[m^2]$ is the total surface of the anodes and g_2 is the electrical conductivity. Within reasonable operational conditions, $\frac{1}{g_2}$ can be approximated as a function that increases linearly with the increasing mass ratio of alumina c_{x_2}. The modeling error in x_4 can therefore propagate to x_6. After approximately eight hours, the error bound of CoSTA models shows that one of the CoSTA models calculates an instantaneous overestimate of x_6, followed by an instantaneously underestimate of x_6. A possible explanation is that the CoSTA model first erroneously predicts the anode effect. The underestimate of x_6 that instantaneously follows can possibly be caused by an underestimate of c_{x_2} that is lower than $c_{x_2,crit}$ which leads to negative P_{el} values in the model.

Temperature in the side ledge x_7: The change of temperature in the side ledge $\dot{x}_7$ is determined by the heat balance in the side ledge. That is, the heat transfer from the bath to the side ledge Q_{liq-sl}, the heat transfer from the side ledge to the side wall $Q_{sl-wall}$, and the energy $E_{tc,sol}$ required to heat frozen side ledge to liquidus temperature from side ledge temperature. The change of side ledge temperature depends on the side ledge thickness x_1, the bath temperature x_6, the side ledge temperature x_7, the wall temperature x_8 and the liquidus temperature

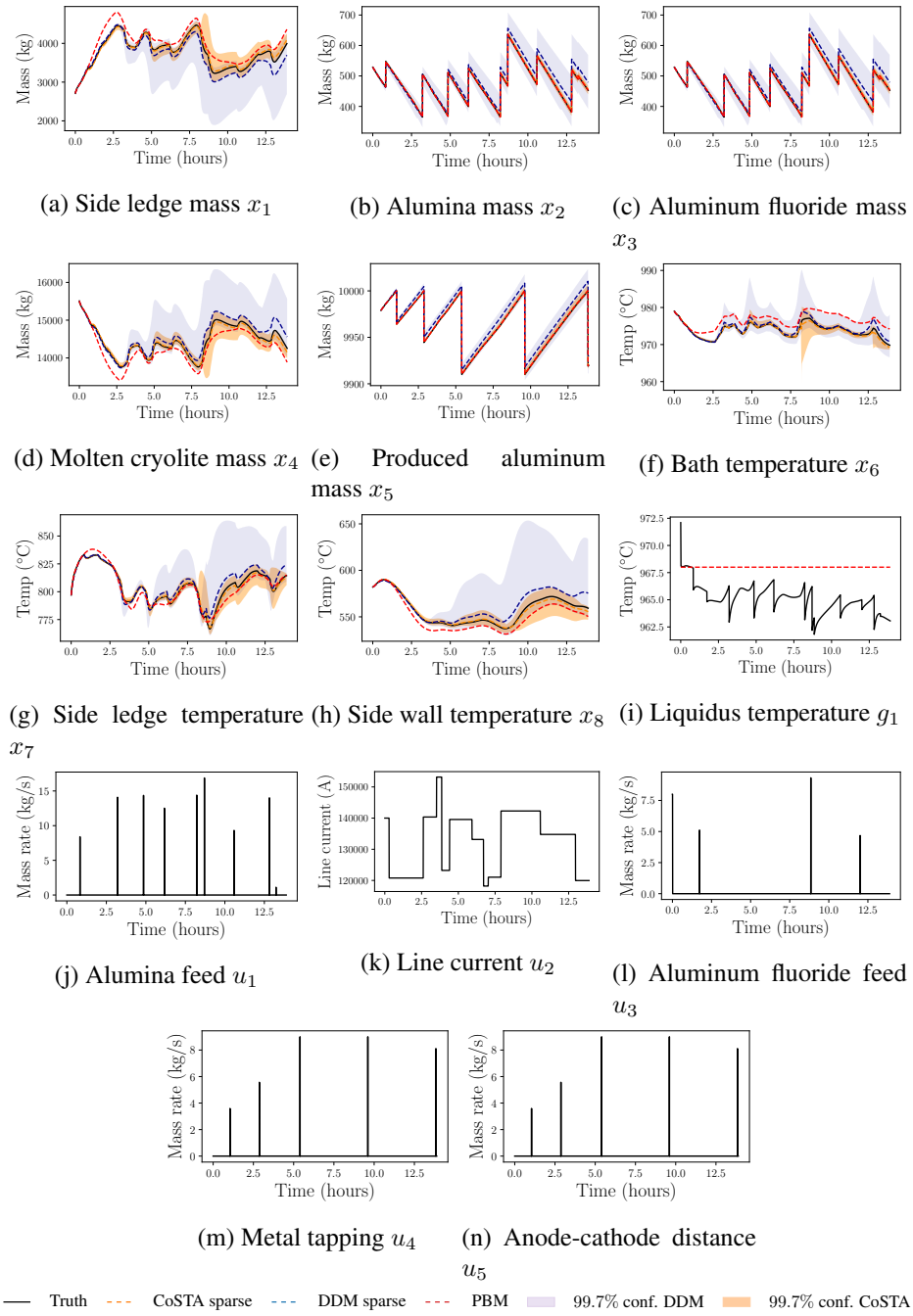
g_1. As argued above, for the PBM modeling errors in $x_1,\ x_6,\ x_7, x_8$, and g_1 will propagate as modeling errors in the side ledge temperature change $\dot{x}_7$. For the DDM and CoSTA models, the error bounds for the modeling errors of x_7 shown in Figure 4.6g are mostly growing simultaneously with error spikes in the error bound of x_6, presumably caused by erroneously predicted anode effects, as explained above.

Temperature in the wall x_8: Figure 4.6h shows that The temperature of the side wall x_8 is changing according to the heat transfer from the side ledge to the wall $Q_{sl-wall}$, and the heat transfer from the wall to the ambient Q_{wall-0}. Changes in the wall temperature $\dot{x_8}$ depend on the side ledge temperature x_7, the wall temperature x_8, and the side ledge thickness x_1. PBM modeling errors of these states at time k propagate as modeling errors in the side wall temperature x_8 in the next time step, $k+1$. Hence, the PBM will, with correct inputs always model the correct $\dot{x}_8$ since the PBM model of $\dot{x}_8$ is equal to the simulator.

4.5 Conclusions and future work

In this work, we presented a recently developed approach in modeling called the Corrective Source Term Approach (CoSTA). CoSTA belongs to a family of HAM tools where PBMs and DDMs are combined to exploit the best of both approaches while eliminating their weaknesses. The method was applied to model an aluminum extraction process governed by very complex physics. First, a detailed high-fidelity simulator was used to generate a dataset treated as the ground truth. Then, an ablated model was created by setting an internal variable of the simulator to a constant. Finally, the ablated model was supplemented with a corrective source term modeled using a NN that compensated for the ignored physics. The main conclusions from the study are as follows:

- CoSTA, in all the scenarios investigated, could correct for the ignored physics and was consistently more accurate in predicting all 8 states of the process compared to both the PBM and DDM over a reasonably long time horizon.
- Both end-to-end learning and CoSTA were able to capture the complex coupling between states and inputs.
- CoSTA consistently yields more stable predictions when compared to pure DDM, despite the numerous input signals being very sparse and discontinuous.
- Regularizing the networks using ℓ_1 regularization was found to be effective in improving model stability in both DDM and CoSTA.



(a) Side ledge mass x_1 (b) Alumina mass x_2 (c) Aluminum fluoride mass x_3

(d) Molten cryolite mass x_4 (e) Produced aluminum mass x_5 (f) Bath temperature x_6

(g) Side ledge temperature x_7 (h) Side wall temperature x_8 (i) Liquidus temperature g_1

(j) Alumina feed u_1 (k) Line current u_2 (l) Aluminum fluoride feed u_3

(m) Metal tapping u_4 (n) Anode-cathode distance u_5

Figure 4.6: Rolling forecast of a representative test trajectory. 10 CoSTA models with sparse corrective NN's, 10 DDM's consisting of sparse NN models, as well as a PBM, are predicting the test set trajectories given the initial conditions and the input vector at any given time.

One significant benefit of the CoSTA approach is that it can maximize the utilization of domain knowledge, leading to reliance on black-box DDM for modeling only those physics that are either not known or are poorly known. Although it remains to be investigated in future work, it can be expected that much simpler models will be sufficient for modeling the corrective source terms. These source terms can then be investigated to achieve additional insight giving more confidence in the model. Even if it is not possible to interpret the source terms, it should still be possible to place bounds on their outputs using domain knowledge. This can be used as an inbuilt sanity check mechanism in the system. For example, since we know the amount of energy put into the system, the source terms for the energy equation will be bounded, so any NN-generated source term violating this bound can be confidently rejected, making the models more attractive for safety-critical applications like the one considered here. Another topic that would be worth investigating is the robustness of the method to noise.

Chapter 5

Modeling dynamics using sparse neural networks

This chapter is based on the publication:

[80] **E. T. B. Lundby**, A. Rasheed, I. J. Halvorsen, J. T. Gravdahl "Sparse deep neural networks for modeling aluminum electrolysis dynamics". In: Applied Soft Computing 134 (2023), p. 109989. ISSN: 1568-4946.

The contribution consider the effect of the sparsity promoting ℓ_1 regularization on generalizability, interpretability and training stability of DNNs compared to densely trained DNNs.

5.1 Introduction

NNs are challenging to interpret, difficult to generalize to solve previously unseen problems, and unstable to train. These are critical shortcomings to overcome before the models can be used in high-stake applications. We discuss each of these briefly.

Interpretability: This can be defined as the ability of a model to express itself in human interpretable form [121]. A simple model like linear regression having very few trainable parameters can be a good example of an interpretable model. However, a DNN, constituting millions of trainable parameters, can be extremely difficult or almost impossible to interpret. We can attempt to remedy this by training DNNs with a sparsity prior, thereby reducing the number of parameters and revealing a more parsimonious structure.

Generalizability: This refers to the model's ability to predict outcome values for

unseen data from the same distribution as the training data. Highly complex and overparameterized models are prone to overfitting, meaning they do not generalize to unseen data not adequately represented during the training. The overfitting can be mitigated by increasing the amount and variety of data. However, in many complex physical systems, the cost and challenges of data acquisition limit the amount of training data. As a result, the trained DNN can fail to generalize. However, if the overparameterization issue is addressed, then there is tentative evidence that the DNN will be better at generalization [122].

Stability: The training of a DNN requires solving an optimization problem in a multidimensional space. Depending upon the complexity of the problem, the dimension can easily be in thousands or even millions, and multiple local minima might be encountered. Even the same DNN with just a slightly different initialization of the parameters can end up in very different minima, and the risk of ending up in a bad minimum is high. By stability in the context of the current work, we mean that the optimization leads to a reasonable minimum, which, even if not global, yields a similar loss value. On the contrary, an unstable DNN will be the one where the optimization process yields inconsistent results, gets stuck in bad local minima, or fails to converge to an acceptable parameter configuration.

This chapter addresses the challenges of generalizability, interpretability, and stability by training sparse neural networks. Authors in [122] show empirically that sparse neural networks can generalize better than dense neural networks on classification tasks. This line of reasoning is intuitively related to Occam's Razor, and even early research such as [123] has investigated trimming small neural networks to increase interpretability. However, most recent research in deep learning focuses on high-dimensional data and uses architectures with millions to billions of parameters. Fully interpreting these models is unfortunately intractable and is not given much attention in this work. On the other hand, dynamical systems can often be expressed in a relatively low dimensional state space despite their rich and complex behavior. This makes them a good candidate for further investigation. Unfortunately, the research on sparse neural networks for modeling dynamical systems is limited. The authors in [124] propose a sparse Bayesian deep learning algorithm for system identification. The method was tested on a simulator of a cascade tank [125] with two states and one input, and on a simulator of coupled electric drives [126] with three states and one input. Besides this, little research has been done, and the critical shortcomings of DNN mentioned above remain mostly unaddressed. In this work, we attempt to address the following research questions:

- What effect can sparsity-promoting regularization have on the complexity of neural networks?

- Can one generate insight from the interpretation of sparse neural networks, or are they as difficult to interpret as their dense counterparts?
- Can sparsity promoting regularization improve the data efficiency of neural networks?
- Can the model uncertainty of neural networks be reduced, and their accuracy improved so that they are better suited for modeling the complex dynamics over both short and long-term horizons?

The study is applied to the aluminum electrolysis simulator described in Section 2.1.2.

The chapter is structured as follows. Section 5.2 presents theory on model complexity of ReLU networks. Section 5.3 presents the method applied in the paper and the experimental setup of the simulator for data generation. In section 5.4, the results are presented and discussed. Finally, in section 5.5, conclusions are given, and potential future work is presented.

5.2 Region bounds for piecewise affine neural networks

The complexity of NNs with Piecewise Affine (PWA) activation functions such as ReLU can be analyzed by looking at how the network partitions the models' input space to an exponential number of linear response regions [127, 128]. For each region in the input space, the PWA neural network has a linear response for the output. Authors in [129] present asymptotic upper and lower bounds for maximum number of regions for a DNN with ReLU activation:

$$\begin{aligned} &\text{Lower} : \Omega\left(\left(\frac{n}{d}\right)^{(L-1)d} n^d\right), \\ &\text{Upper} : \mathcal{O}\left(n^{dL}\right). \end{aligned} \tag{5.1}$$

d is the input dimension, L is the number of hidden layers, and n is the number of activation functions or neurons in each layer. The bounds in Equation (5.1) are valid for networks with the same number of neurons in each layer. The bonds for networks with an uneven number of neurons show similar exponential results and are thus not included for convenience. Equation (5.1) illustrates the exponential relation between the input dimension, number of neurons, and the depth of the network. For realistic amounts of data sampled from a physical system, the number of linear regions that a relatively small dense neural network partition the input space into exceeds the sampled data by several orders of magnitude. Thus, in order to generalize to larger areas of the models' input space, the number of regions needs to be reduced drastically. This motivates sparsifying the model.

5.3 Method and experimental setup

5.3.1 Training with sparsity promoting regularization

In this chapter, sparse DNN models are utilized to predict state variables in the aluminum electrolysis simulator. All the weight matrices in a DNN model are ℓ_1 regularized to impose sparsity. Figure 5.1 illustrates how weights are enumerated according to their input and output nodes. Layer j has i nodes and layer $(j+1)$ has r nodes. Layer $j = 0$ corresponds to the input layer, and consist of measured or estimated states $\mathbf{x}(t)$ and control inputs $\mathbf{u}(t)$ at time step t. The output layer consists of the estimated time derivatives of the states $\dot{\mathbf{x}}(t)$ at time step t.

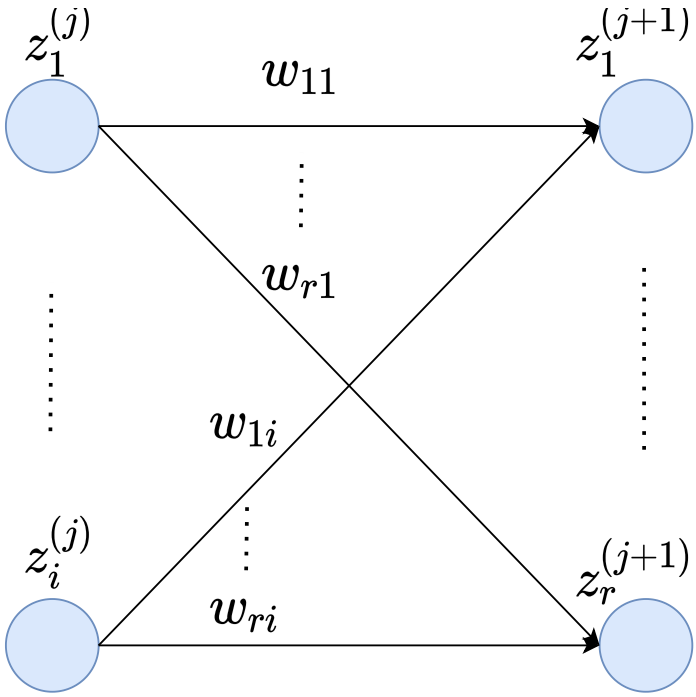


Figure 5.1: Enumerated weights according to their input and output nodes between layer j and $(j+1)$

The weight matrix $\mathbf{W}_{j+1}$ corresponds to the weights that connect layer j to $j+1$.

$\mathbf{W}_{j+1}$ is arranged as follows:

$$\mathbf{W}_{j+1} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1i} \\ w_{21} & w_{22} & \dots & w_{2i} \\ \vdots & \ddots & & \\ w_{r1} & w_{r2} & \dots & w_{ri} \end{bmatrix}. \tag{5.2}$$

Regularization terms $R_{\ell_1,j+1}$ are defined for each weight matrix $\mathbf{W}_{j+1}$:

$$R_{\ell_1,j+1} = \sum_{i,k \in W_{j+1}} |w_{i,k}|. \tag{5.3}$$

where $w_{i,k}$ are the model weights from layer j to $j+1$, or equivalently, the elements of $\mathbf{W}_{j+1}$. The regularization terms for each layer are added to the cost function:

$$\begin{aligned} \mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \Bigg\{ & \frac{1}{N} \sum_{i=1}^{N} (\mathbf{y_i} - \mathbf{f}(\mathbf{x_i}))^2 + \lambda_1 R_{\ell_1,1} ... \\ & + \lambda_{j+1} R_{\ell_1,j+1} + ... + \lambda_L R_{\ell_1,L} \Bigg\}, \end{aligned} \tag{5.4}$$

where $\frac{1}{N} \sum_{i=1}^{N} (\mathbf{y_i} - \mathbf{f}(\mathbf{x_i}))^2$ is the MSE, and $\lambda_{j+1}, j = 0, \ ..., 3$ is a layer-specific hyperparameter that determines how the weights in $\mathbf{W}_{j+1}$ are penalized.

5.3.2 Experimental setup and data generation

Data for the aluminium electrolysis process is generated by integrating the non-linear ODEs given by Eqs. (2.15a) - (2.15h) with a set of chosen initial values for the state variables $\mathbf{x}(t_0)$, and fourth-order RK4 algorithm. The initial conditions of each variable x_i for each simulation are randomly chosen from a given interval of possible initial conditions given in Table 5.1. For x_2 and x_3, concentrations c_{x_2} and c_{x_3} are given. Data-driven models depend on a high degree of variation in the training data to be reliable and valid in a large area of the input space. The input signal determines how the system is excited and thus what data is available for modeling and parameter estimation. Operational data from a controlled, stable process is generally characterized by a low degree of variation. Even large amounts of data sampled over a long period from a controlled process can not guarantee that the variation in the training data is large enough to ensure that the trained model generalizes to unseen data. Therefore, random excitations are added to the input signals to increase the variation in the sampled data. The intuition is that the random excitation will push the dynamics out of the standard operating condition so that variation in the training data increases. In general, each control input i is given by:

$$u_i = \text{Deterministic term + Random term.} \tag{5.5}$$

Table 5.1: Initial conditions for system variables

Variable	Initial condition interval
x_1	$[3260,\ 3260]$
c_{x_2}	$[0.02,\ 0.03]$
c_{x_3}	$[0.10,\ 0.12]$
x_4	$[13500,\ 14000]$
x_5	$[9950,\ 10000]$
x_6	$[975,\ 975]$
x_7	$[816,\ 816]$
x_8	$[580,\ 580]$

The control inputs u_1, u_3 and u_4 are impulses. The random term is zero for these control inputs when the deterministic term is zero. The deterministic term is a proportional controller. The control inputs u_2 and u_5 are always nonzero. These control inputs have constant deterministic terms and a random term that changes periodically. The random term stays constant for a certain period ΔT_{rand} before changing to a new randomly determined constant. Choosing the period ΔT_{rand} is a matter of balancing different objectives. On one hand, it is desirable to choose a large period ΔT_{rand} so that the system can stabilize and evolve under the given conditions to reveal the system dynamics under the given conditions. On the other hand, it is desirable to test the systems under many different operational conditions. By empirically testing different periods ΔT_{rand}, and seeing how the dynamics evolve in simulation, it turns out that setting $\Delta T_{rand} = 30\Delta T$ is a fair compromise between the two. Table 5.2 gives the numerical values of the

Table 5.2: Control functions

Input	Deterministic term	Random term interval	ΔT_{rand}
u_1	$3e4(0.023 - c_{x_2})$	$[-2.0, 2.0]$	ΔT
u_2	$14e3$	$[-7e3, 7e3]$	$30 \cdot \Delta T$
u_3	$13e3(0.105 - c_{x_3})$	$[-0.5,\ 0.5]$	ΔT
u_4	$2(x_5 - 10e3)$	$[-2.0,\ 2.0]$	ΔT
u_5	0.05	$[-0.015,\ 0.015]$	$30 \cdot \Delta T$

deterministic term of the control input, the interval of values for the random terms, and the duration ΔT_{rand} of how long the random term is constant before either becoming zero (u_1, u_3, u_4) or changing to a new randomly chosen value (u_2, u_5). One simulation i with a given set of initial conditions is simulated for 1000 time

steps, and each time step $\Delta T = 30s$. The simulation generates the data matrix as in Equation (5.6):

$$\mathbf{X} = \begin{bmatrix} x_1(0) & x_2(0) & ... & x_8(0) & u_1(0) & ... & u_5(0) \\ x_1(1) & x_2(1) & ... & x_8(1) & u_1(1) & ... & u_5(1) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_1(k) & \dots & x_i(j) & \dots & u_1(j) & \dots & u_5(j) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_1(999) & x_2(999) & ... & x_8(999) & u_1(999) & ... & u_5(999) \end{bmatrix}. \tag{5.6}$$

The number j within the parenthesis of variable i indicates the time step for when $x_i(j)$ is sampled. The target values are then calculated as

$$\mathbf{Y} = \begin{bmatrix} \frac{x_1(1)-x_1(0)}{\Delta T} & \dots & \frac{x_8(1)-x_8(0)}{\Delta T} \\ \frac{x_1(2)-x_1(1)}{\Delta T} & \dots & \frac{x_8(2)-x_8(1)}{\Delta T} \\ \vdots & \ddots & \vdots \\ \frac{x_1(k)-x_1(j-1)}{\Delta T} & \dots & \frac{x_8(j)-x_8(k-1)}{\Delta T} \\ \vdots & \ddots & \vdots \\ \frac{x_1(1000)-x_1(999)}{\Delta T} & \dots & \frac{x_8(1000)-x_8(999)}{\Delta T} \end{bmatrix}. \tag{5.7}$$

Each training set $\mathcal{S}_k$ from simulation k are put in input and outputs are put in pairs:

$$\mathcal{S}_k = [\mathbf{X}, \mathbf{Y}] = \begin{bmatrix} [\mathbf{x}^T(0), \mathbf{u}^T(0)]^T, & \mathbf{y}(0) \\ \vdots & \vdots \\ [\mathbf{x}^T(j), \mathbf{u}^T(j)]^T, & \mathbf{y}(j) \\ \vdots & \vdots \\ [\mathbf{x}^T(999), \mathbf{u}^T(999)]^T, & \mathbf{y}(999) \end{bmatrix}. \tag{5.8}$$

The training sets from each simulation are normalized before they are stacked

$$\mathcal{S}_{stack} = \left[\mathcal{S}_1^T,\ \mathcal{S}_2^T,\ ...,\ \mathcal{S}_k^T,\ ....,\ \mathcal{S}_n^T\right]^T. \tag{5.9}$$

Here, n is the number of simulated time-series $\mathbf{X}$. The number of time series simulations n varies in the experiments to evaluate the model performance as a function of training size. Then, all input-output pairs in the stacked training set are shuffled:

$$\mathcal{S}_{train} = \text{shuffle}(\mathcal{S}_{stack}). \tag{5.10}$$

The shuffled training set is put in mini-batches $[\mathbf{X}_{batch,\,i},\ \mathbf{Y}_{batch,\,i}]$, and the models are trained on these mini-batches. The test set also consists of several time series simulations generated in the same way described above. The test set is given by:

$$\mathcal{S}_{test} = \{\{X_1\},\ \{X_2\},\ ...,\ \{X_p\}\}. \tag{5.11}$$

$\mathcal{S}_{test}(i) = \{X_i\} = \{([\mathbf{x}_k,\ \mathbf{u}_k]\}_{k=1}^{1000}$ is one simulated time series, and $\{\mathbf{x}_k\}_{k=1}^{1000}$ is being forcasted by the models. In all experiments, 20 models of each dense and sparse networks with different initialization are trained on the training set and then evaluated on the test set.

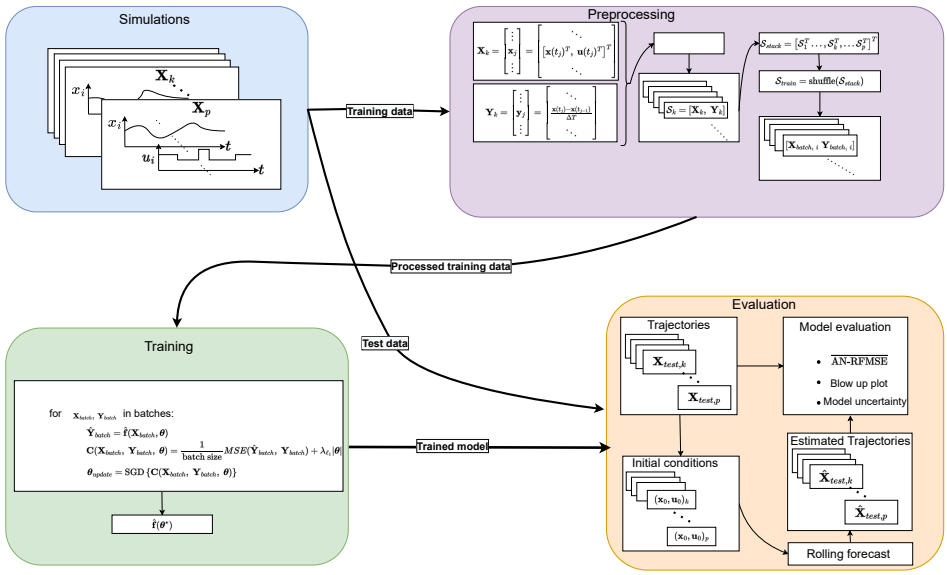


Figure 5.2: Schematic presentation of the experimental setup. This includes the data simulation, preprocessing of training data, model training, and model evaluation. In the first step (blue box), training and test data are simulated. Each frame in the blue box corresponds to a simulation of the dynamics from some random initial conditions. The training and test data are separated, where the test data is given directly to the evaluation part of the case study, while the training data is sent to preprocessing. In the preprocessing stage, input features from a given simulation are arranged in an input matrix. Output features are calculated (see Equation (5.7)) before they are put in an output matrix corresponding to the input matrix. Then, both input and output features are normalized and put in pairs. After all the simulations are arranged in input-output pairs, all pairs are shuffled before being put in mini-batches for training. In the training procedure, the model parameters are optimized on the mini-batches. The trained models are then sent to the evaluation stage. In this stage, models are given the initial conditions from the test set. In addition, the models are given the control inputs in the test set at every time step. Given the initial conditions as well as control inputs at every time-step of the test set trajectory, the models produce a forecast of the test set trajectories. The estimated trajectories are compared to the test set trajectories and evaluated according to accuracy measures, uncertainty in terms of disagreement between models with different initial parameters trained on the same data with the same hyperparameters, and according to the number of blow-ups of a given model type (models trained with same hyperparameters), meaning the number of times that model type estimate diverges from the test set they are estimating.

Figure 5.2 summarizes the workflow in the case study. Each step is briefly explained in the figure text, and more thoroughly throughout Sec. 5.3.

5.4 Results and discussion

In this section, we present and discuss the main findings of the work. In doing so, we will analyze the results from the perspective of interpretability, generalizability, and training stability.

5.4.1 Interpretability perspective

As discussed earlier, the interpretability of a model is the key to its acceptability in high-stake applications like the aluminum extraction process considered here. Unfortunately, highly complex dense neural networks having thousands to millions of parameters which were the starting point for the modeling here are almost impossible to interpret. Figure 5.3 shows the model structure of a dense DNN model learned for the generated data. The figure illustrates how densely trained neural networks yield uninterpretable model structures.

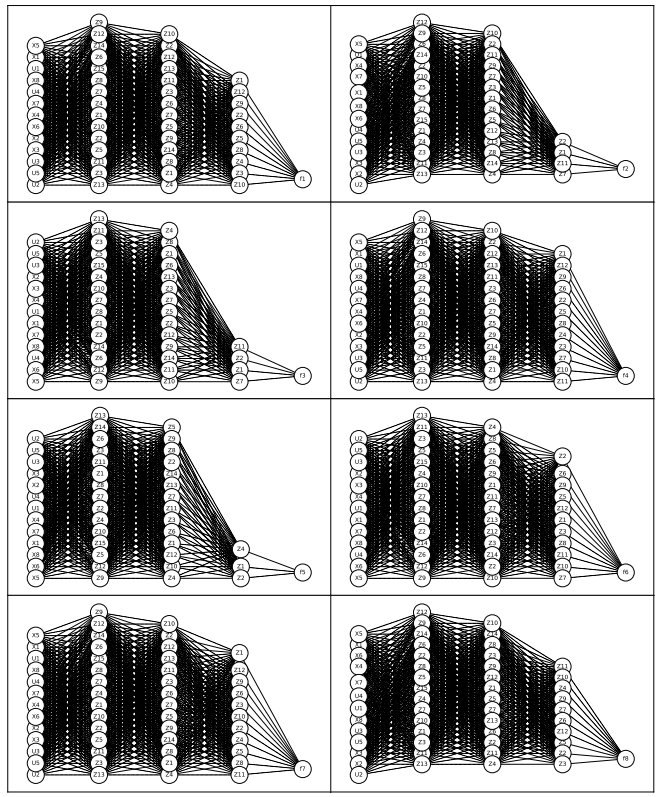


Figure 5.3: Model structure of each output function $\{f_1, .., f_8\}$ for one of the trained dense neural network models. The circles represent neurons, inputs, and outputs in the model. X_i represent system variable i in the input layer, U_i represents control input i in the input layer, and Z_i represents latent variable i in the layer it is visualized. The directed edges indicate weights in the model.

Fortunately, through the regularization, it was possible to significantly reduce the model complexity resulting in a drastically reduced number of trainable parameters (see the Figs. 5.4-5.11). It can be argued that the reduced model complexity of sparse neural networks increases the model interpretability. With domain knowledge about the aluminum electrolysis process, the sparse models can be evaluated as we will do in the remainder of this section.

The results related to the interpretability aspect are presented in the form of model structure plots which can be used to explain the input-output mapping of the models. If the model structures are very sensitive to initialization, then their interpretation will not make sense. Therefore, 100 DNNs with different initialization are

trained independently and their common trends are emphasized in the discussions. We now present each of the model outputs $\{\hat{f}_1(\mathbf{x}, \mathbf{u}),\ ...,\ \hat{f}_8(\mathbf{x}, \mathbf{u})\}$. It is worth mentioning that these outputs are estimates of each of the time derivatives of the states $\{\dot{x}_1,\ ...,\ \dot{x}_8\}$ respectively.

Table 5.3: Frequency of learned features for each output $\{\hat{f}_1,\ ...,\ \hat{f}_8\}$. Each column i corresponds to an output $\hat{f}_i$ of the neural network. Each row element j correspond to one of the features $\{x_1,\ x_2,\ ...,\ x_8,\ u_1,\ u_2,\ ...,\ u_5\}$. The value of the table element (i, j) is the percent of how many out of one hundred models of the output f_i that feature j occurs for DNNs with ℓ_1 regularization.

Feature	Output functions							
	$\hat{f}_1$	$\hat{f}_2$	$\hat{f}_3$	$\hat{f}_4$	$\hat{f}_5$	$\hat{f}_6$	$\hat{f}_7$	$\hat{f}_8$
x_1	86	1	1	86	2	100	99	100
x_2	100	2	2	100	1	100	100	100
x_3	100	2	2	100	0	93	100	87
x_4	100	0	0	100	0	87	100	87
x_5	2	0	0	2	0	2	2	2
x_6	100	2	2	100	1	100	100	87
x_7	22	1	1	21	1	7	89	89
x_8	100	1	1	100	2	87	100	100
u_1	100	100	100	100	0	100	100	87
u_2	4	0	0	4	1	100	18	18
u_3	2	0	100	2	0	2	4	4
u_4	2	0	0	2	100	3	3	3
u_5	3	0	0	3	1	100	18	18

Model output $\hat{f}_1$

The simulation model for the first output f_1 defined in Equation (2.15a) is a function of the features $\{x_1, x_2, x_3, x_4, x_6, x_7\}$. f_1 can further be divided into three sums $f_1 = h_1(x_1, x_2, x_3, x_4, x_7) + h_2(x_6) + h_3(x_2, x_3, x_4)$, where $h_1 = k_1 \frac{g_1(x_2, x_3, x_4) - x_7}{k_0 x_1}$, $h_2 = -k_2 x_6$ and $h_3 = g_1(x_2, x_3, x_4)$. g_1 is a nonlinear function defined in Equation (2.14a).

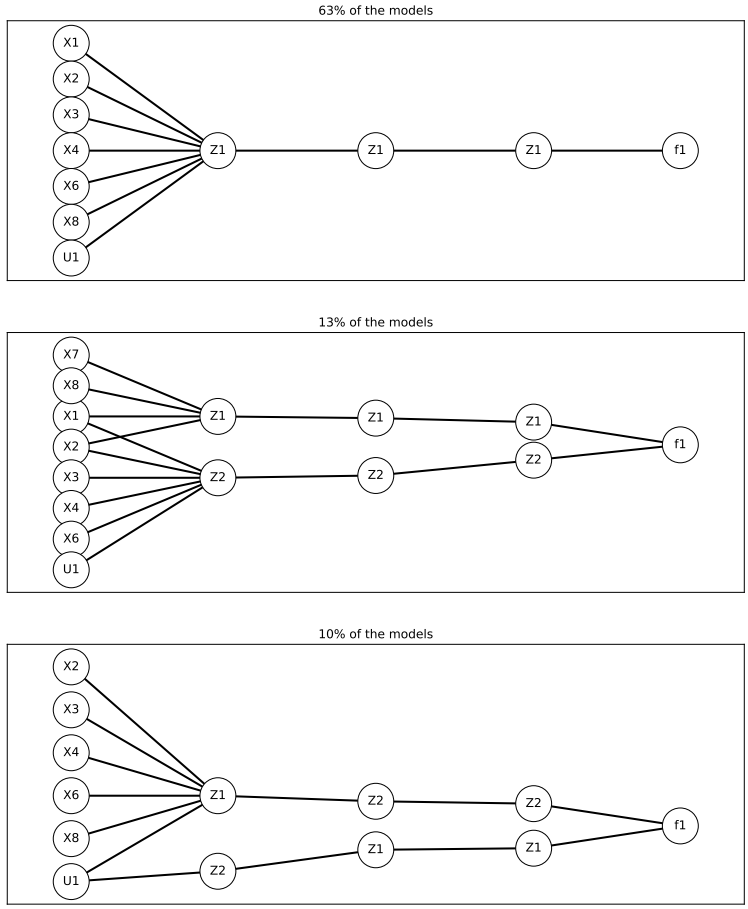


Figure 5.4: Most common learned structures for $\hat{f}_1(\mathbf{x}, \mathbf{u})$ for DNNs with ℓ_1 regularization..

Figure 5.4 shows the three most common learned structures of the first output of the neural network $\hat{f}_1(\mathbf{x}, \mathbf{u})$. In total, these structures account for 86% of all learned structures of $\hat{f}_1$. The top structure forms the resulting structure for 63% of the models. It is a function of seven inputs $\hat{f}_1 = f(x_1, x_2, x_3, x_4, x_6, x_8, u_1)$. All input features are connected to the same neuron in the first layer. Moreover, it is only one neuron in each hidden layer. The upper bound in Equation (5.1) states that this model structure only has $n^{dL} = 1^{7 \cdot 3} = 1$ region. This is equivalent to stating that the model collapses to one linear model. The middle and the bottom structures of Figure 5.4 has more than one neuron in the hidden layers. However, the structures can be divided into two disconnected subnetworks since the hidden neurons are

not connected before they are added in the output layer. Hence, also these models collapse to linear models with a single region. This means that the neural networks do not capture the nonlinear dynamics in the simulation model. All structures of Figure 5.4 are erroneously including x_8 and u_1 in their feature basis. Besides, x_7 which is present in the simulation model f_1 is not found by the top and bottom structures in Figure 5.4. In fact, x_7 is found only in 22% of the models $\hat{f}_1$ according to Table 5.3. The exact cause of this erroneous feature selection is not trivial. However, x_8 which is the wall temperature correlates highly with the side ledge temperature x_7. Thus, x_8 can possible have been learned as a feature of f_1 instead of x_7. Moreover, the alumina feed u_1 on the other hand affects the time derivative of the mass of alumina $\dot{x}_2$, the time derivative of mass of aluminum fluoride $\dot{x}_3$ and the time derivative of cryolite $\dot{x}_4$ directly. All these variables affect $\dot{x}_1$ through the liquidus temperature $g_1(x_2, x_3, x_4)$. To understand how this might cause the learning algorithm to find u_1 as a feature of $\hat{f}_1$, consider the following: let u_1 be zero until time t. Then, $\{x_2, x_3, x_4\}$ will be updated due to u_1 at the next sampled time step $t + 1$. However, the fourth-order Runge Kutta solver splits the sampling interval into 4 smaller intervals $\{t + 0.25,\ t + 0.5,\ t + 0.75,\ t + 1\}$ and solve the ODE eqautions at all these time steps. Thus, the state variables $\{x_2, x_3, x_4\}$ are updated already at time $t + 0.25$. Since $\dot{x}_1$ is depending on these variables, x_1 will be updated at $t + 0.5$. Therefore, at time $t + 1$, when data is sampled, x_1 would also be changed. Hence, the learning algorithm finds u_1 to affect the time derivative $\dot{x}_1$. This could might have been solved by shortening the sampling interval. Furthermore, x_1 not included as a feature in 14% of the models. This might be a combination of parameter initialization and that x_1 is multiplied by the small constant $k_0 = 2 \cdot 10^{-5}$.

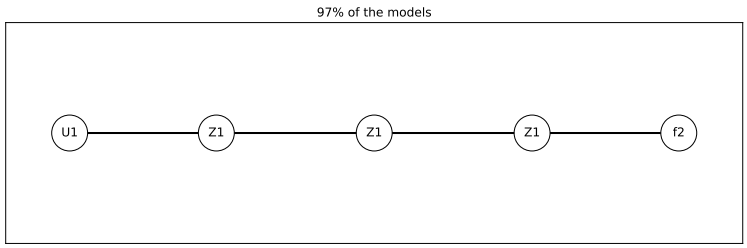


Figure 5.5: Most common learned structure for $\hat{f}_2(\mathbf{x}, \mathbf{u})$ for DNNs with ℓ_1 regularization. 97% of f_2 ends up with this structure.

Model output $\hat{f}_2$

Figure 5.5 shows the most common learned structure among the models $\hat{f}_2$ that models the time derivative of alumina $f_2 = \dot{x}_2$. 97% of the structures end up as the structure in Figure 5.5. The simulation model f_2 in Equation (2.15b) is a linear model dependent on $\{u_1, u_2\}$. The learned models $\hat{f}_2$ only finds u_1 as the relevant feature. The reason for this might be that u_2 is proportional to a very small constant $k_3 = 1.7 \cdot 10^{-7}$. Variations in the line current u_2 are not big enough to be significant for the learning algorithm. The dynamics caused by u_2 are captured in a bias in the models.

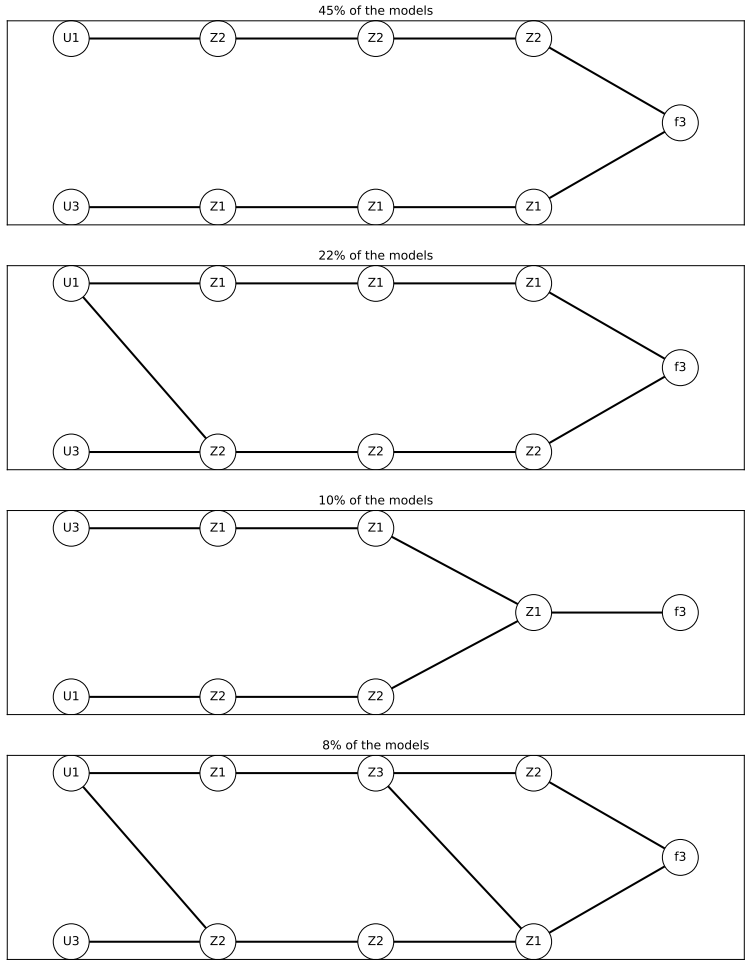


Figure 5.6: Most common learned structures for $\hat{f}_3(\mathbf{x}, \mathbf{u})$ for DNNs with ℓ_1 regularization.

Model output $\hat{f}_3$

Figure 5.6 shows the four most common learned structures of $\hat{f}_3$. $\hat{f}_3$ models the time derivative of the aluminum fluoride mass $\dot{x}_3$ in the cell. The simulation model $\dot{x}_3 = f_3$ in Equation (2.15c) is a linear model depending on the features $\{u_1, u_3\}$, and in all structures in Figure 5.6, only these two features are found. As shown in Table 5.3, these features are found in 100% of the trained models. The structures found are mainly linear models. However, in the second and fourth structures, some weights connect the features in intermediate layers.

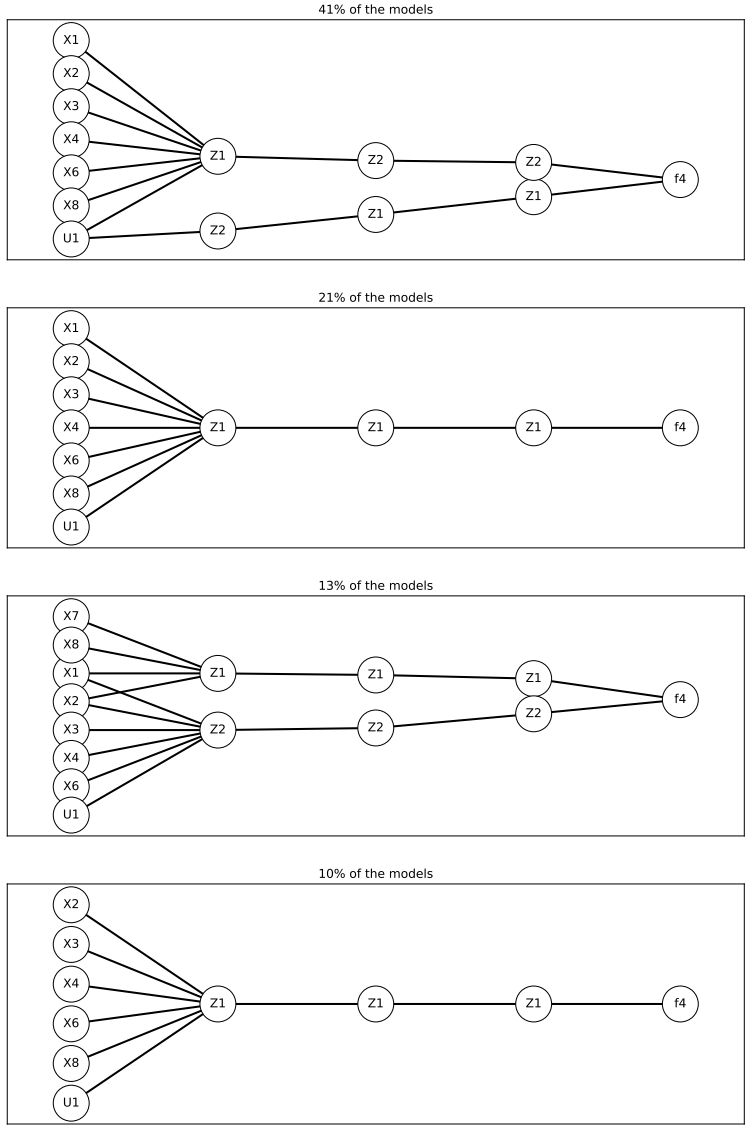


Figure 5.7: Most common learned structures for $\hat{f}_4(\mathbf{x}, \mathbf{u})$ for DNNs with ℓ_1 regularization.

Model output $\hat{f}_4$

Figure 5.7 shows the four most common learned structures among models $\hat{f}_4$ that models the mass rate of liquid cryolite Na_3AlF_6 in the bath, namely $\dot{x}_4$. The

DNNs yielding this structure are ℓ_1 regularized. $\dot{x}_4$ is simulated by the simulation model $\dot{x}_4 = f_4$ in Equation (2.15c). f_4 consist of the features $\{x_1, x_2, x_3, x_4, x_6, x_7, u_1\}$. Table 5.3 show that $\{x_2, x_3, x_4, x_6, x_8, u_1\}$ are included in 100% of the learned models, x_1 is included in 86% of the models and x_7 is included in only 21% of the models. As for $\hat{f}_1$, x_8 is erroneously included in the basis of the model. This might be explained by the fact that x_7 and x_8 are highly correlated and that the wrong feature is included. The structures in Figure 5.7 are all forming linear models. However, the simulation model $\dot{x}_4 = f_4$ is partly nonlinear. Thus, the approximation $\hat{f}_4$ oversimplifies the dynamics. This might be caused by a high weighting of the sparse regularisation term. If the loss function is less penalized, there is room for more weights in the model and, therefore, more nonlinearities.

Model output $\hat{f}_5$

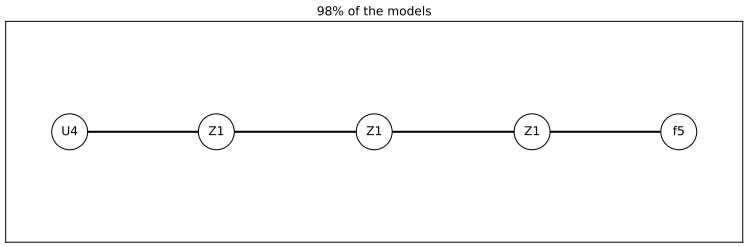


Figure 5.8: Most common learned structures for $\hat{f}_5(\mathbf{x}, \mathbf{u})$ for DNNs with ℓ_1 regularization.

Figure 5.8 shows the most commonly learned structure among the ℓ_1 regularized DNNs that models $\hat{f}_5$. The structures in the figure include 98% of the learned model structures. $\hat{f}_5$ is modeling the mass rate of produced aluminum in the cell $\dot{x}_5$. The x_5 time series are produced by the simulation model $\dot{x}_5 = f_5$ in Equation 2.15e. f_5 is a linear model dependent on the features $\{u_2, u_4\}$. However, most of the model structures are only depending on u_4. This can be caused by the fact that u_2 is proportional to a very small constant $k_6 = 4.43 \cdot 10^{-8}$. Thus, variations in u_2 might not be large enough for the learning algorithm to find u_2 significant as a basis for $\hat{f}_5$.

Model output $\hat{f}_6$

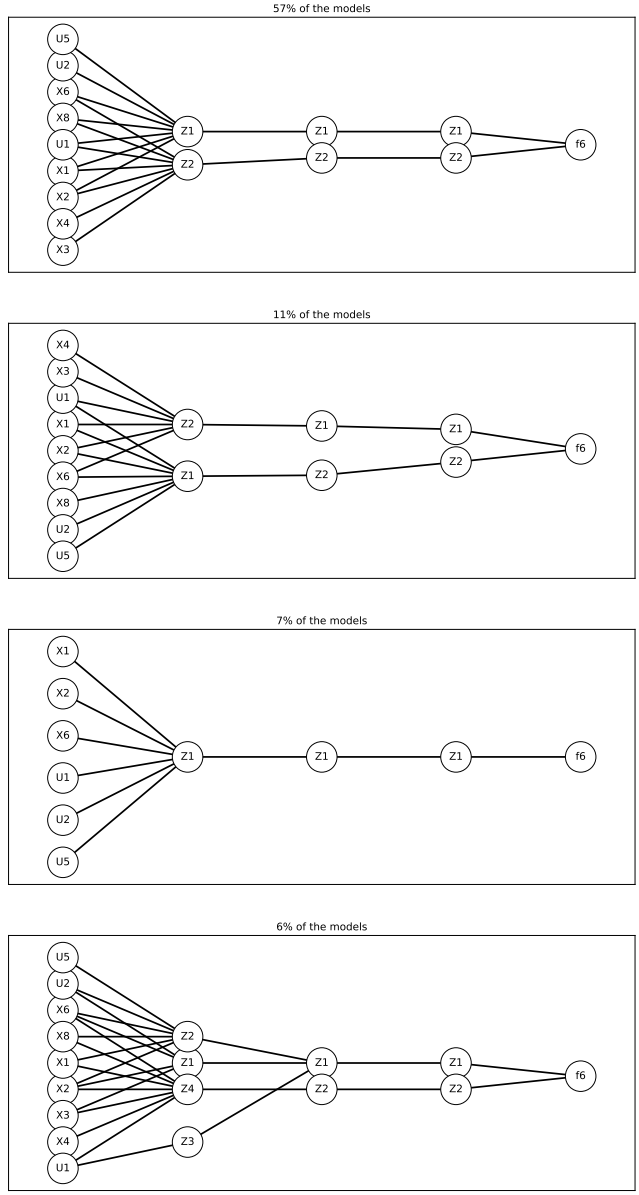


Figure 5.9: Most common learned structures for $\hat{f}_6(\mathbf{x}, \mathbf{u})$ for DNNs with ℓ_1 regularization.

Figure 5.9 shows the most common model structures of $\hat{f}_6$. $\hat{f}_6$ models the bath temperature time derivative $\dot{x}_6$. The bath temperature x_6 is simulated by the ODE in Equation (2.15f). It is a nonlinear equation depending on $\{x_1, x_2, x_3, x_4, x_6, x_7, u_2, u_5\}$. The most common structure, learned by 57% of the models $\hat{f}_6$ is illustrated in the top plot of Figure 5.9. This structure has the basis $\{x_1, x_2, x_3, x_4, x_6, x_8, u_1, u_2, u_5\}$. Hence, it finds u_1 and x_8, which is erroneously found in many of the structures above. A possible explanation for this trend is given above. The structure has two neurons in the first layer, one with the basis $\{x_1, x_2, x_6, x_8, u_1, u_2, u_5\}$ and one with the basis $\{x_1, x_2, x_3, x_4, x_6, x_8, u_1\}$. Since the model collapses to a linear model, all terms are summed in the end. Thus, the separation of the basis is thus of little importance. The second plot from above in Figure 5.9 is the second most common structure, and accounts for 11% of the models $\hat{f}_6$. It has the same feature basis as the most common structure, but they are arranged differently in the first layer. However, since both model structures are linear models, this arrangement is of minor importance. The third structure in Figure 5.9 which account for 7% of the structures of $\hat{f}_6$ has the basis $\{x_1, x_2, x_6, u_1, u_2, u_5\}$. Compared to the other structures, $\{x_3, x_4, x_8\}$ is not present. Since it only happens rarely, this is maybe partly caused by bad parameter initialization. The fourth most common structure, which accounts for 6% of the structures, has the same feature basis as the first and the second most common structures. This structure is plotted in the bottom of Figure 5.9. While all other structures in Figure 5.9 have one linear response region, the fourth most common model structure models some nonlinearities. That is, neurons in the first hidden layer are connected in the second hidden layer. Hence, the input space must be divided into several linear response regions for this structure.

Model output $\hat{f}_7$

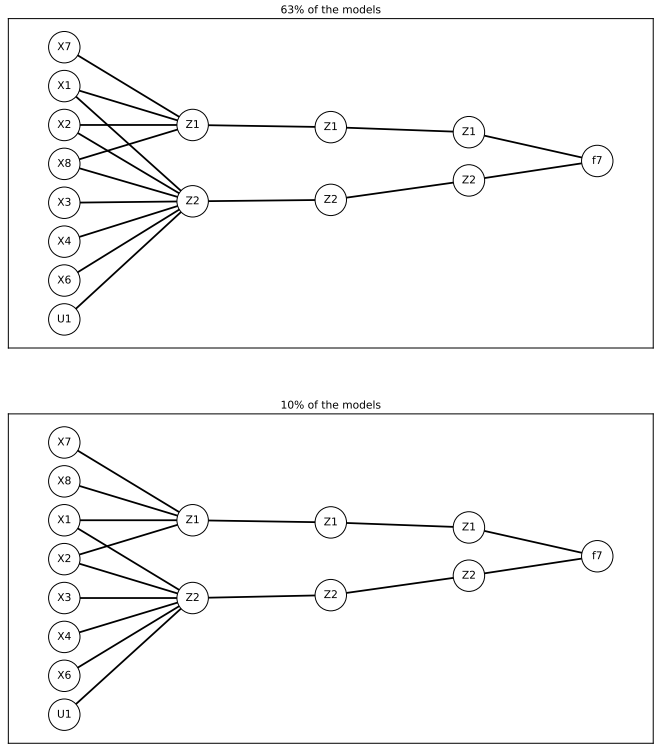


Figure 5.10: Most common learned structures for $\hat{f}_7(\mathbf{x}, \mathbf{u})$ for DNNs with ℓ_1 regularization.

Figure 5.10 shows the two most common structures for the models $\hat{f}_7$. $\hat{f}_7$ models the time derivative of the side ledge temperature $\dot{x}_7$. $\dot{x}_7 = f_7$ is simulated by the ODE in Equation (2.15g). $\dot{x}_7$ is depending on the feature basis $\{x_1, x_2, x_3, x_4, x_6, x_7, x_8\}$. Table 5.3 states that the basis $\{x_2, x_3, x_4, x_6, x_8, u_1\}$ is present for 100% of the models, x_1 is present for 99% of the models and x_7 is present in 89% of the models. The top plot in Figure 5.10 show the structure that accounts for 63% of the models. The bottom plot account for 10% of the model structures of $\hat{f}_7$. These two structures collapse to linear models, and have the same feature basis $\{x_1, x_2, x_3, x_4, x_6, x_7, x_8, u_1\}$, but have minor differences in how weights are connected between input layer and first hidden layer. u_1 is also for this model output erroneously found as a basis, and a possible explanation is mentioned above.

Model output $\hat{f}_8$

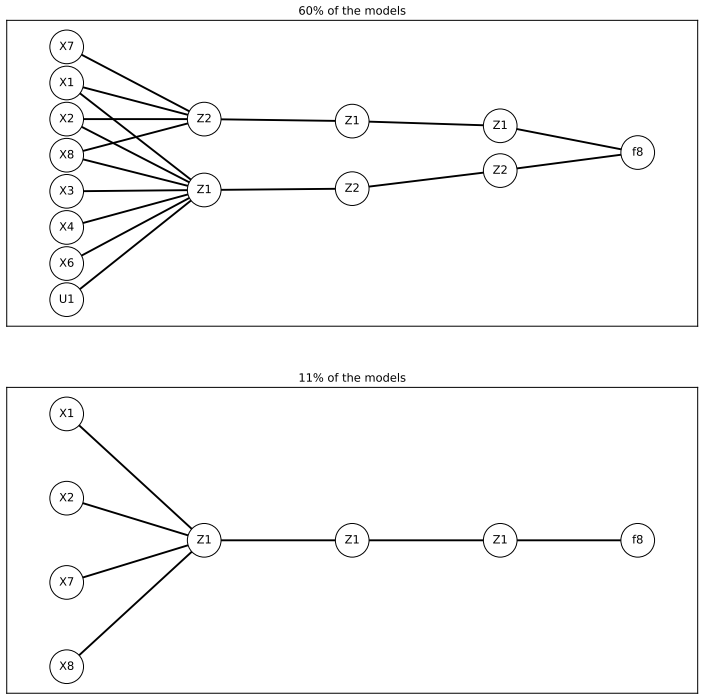


Figure 5.11: Most common learned structures for $\hat{f}_8(\mathbf{x}, \mathbf{u})$ for DNNs with ℓ_1 regularization.

Figure 5.11 show the two most common model structures for the last model output $\hat{f}_8$. $\hat{f}_8$ models the time derivative of the wall temperature $\dot{x}_8 = f_8$, which is simulated in Equation (2.15h). $\dot{x}_8$ depends on the feature basis $\{x_1, x_7, x_8\}$. However, the most common learned structure for $\hat{f}_8$ has the basis $\{x_1, x_2, x_3, x_4, x_6, x_7, x_8, u_1\}$. This is the exact same structure as the most common learned structure for $\hat{f}_7$. Therefore, a possible explanation is that $\hat{f}_8$ adapts the same parameters as $\hat{f}_7$ in some cases as they highly correlates. The bottom plot in Figure 5.11 show the second most common learned structure of the model output $\hat{f}_8$. This structure is learned by 11% of the models $\hat{f}_8$. The feature basis for this structure is $\{x_1, x_2, x_7, x_8\}$, and reminds more of the actual basis. In this structure, there is only one erroneous learned feature, namely x_2. Figure 5.4-5.11 and Table 5.3 show that the sparse learning is quite consistent in finding the same feature basis and structure with similar characteristics. However, some differences that affect the models are present.

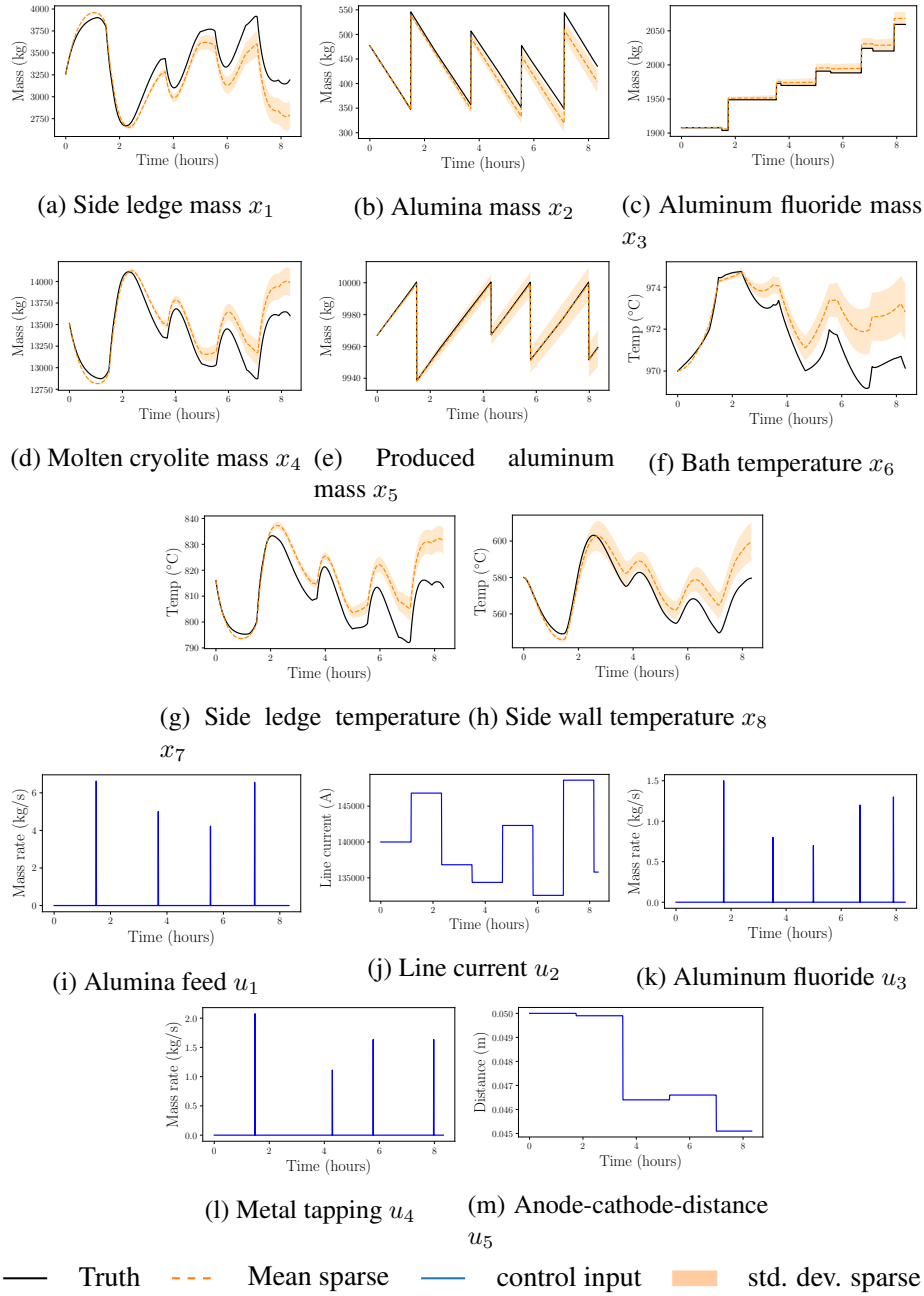
It is clear that doing a similar analysis for models in Figure 5.3 is impossible as all interconnections make the models a black box. On average, 93% of the weights of dense DNNs are pruned in the sparse DNN models. For the outputs $\hat{f}_1,\ \hat{f}_4,\ \hat{f}_6,\ \hat{f}_7$ and $\hat{f}_8$, approximately 40% of the input features are pruned of the model structures. For $\hat{f}_2,\ \hat{f}_3$ and $\hat{f}_5$, $85-95\%$ of the input features are pruned. For all outputs of the sparse DNN models, around $85-95\%$ of the neurons are pruned at each layer. In a neural network, the number of matrix operations only decreases if neurons are pruned. That is, removing a neuron in layer j is equivalent to removing a row in weight matrix $\mathbf{W}_j$ and a column in weight matrix $\mathbf{W}_{j+1}$. The dense models have a compact model structure, where most weights are nonzero. The dense DNN models in the case study have the shapes 13-15-14-12-8. The first number is the number of features, the second, third, and fourth numbers are the number of neurons in hidden layers, and the last is the number of outputs. This shape gives 669 matrix operations in a forward pass. An average sparse DNN, has the shape $13-6-6-6-8$. This gives 198 matrix operations. Thus, the number of matrix operations in the forward pass of a sparse DNN model is reduced by approximately 70%.

5.4.2 Generalizability perspective

This section investigates the joint impact of the training data quantity and model sparsity on the models' performance on a set of test trajectories in terms of accuracy and predictive stability. The performance measures defined in Section 2.4, namely the AN-RFMSE defined in Equation (2.46) and the instabillity measure defined in Equation (2.47) are used.

Comparison of sparse and dense rolling forecast

Figure 5.12 and Figure 5.13 show the performance of the ensembles of 20 sparse and 20 dense DNN models with different parameter initialization forecasting the state variables $\mathbf{x}$ in one of the time series in the test set $\mathcal{S}_{test}(i) = \{\mathbf{X}_i\}$ as defined in Equation (5.11). The models are trained on a data set
$\mathcal{S}_{train} = \{\{\mathbf{X}_1,\ \mathbf{Y}_1\},\ \{\mathbf{X}_2,\ \mathbf{Y}_2\},\ ...,\ \{\mathbf{X}_{10},\ \mathbf{Y}_{10}\}\}$ consisting of ten time series $\{\mathbf{X}_1,\ ..,\ \mathbf{X}_{10}\}$ with 999 time steps each.



(a) Side ledge mass x_1 (b) Alumina mass x_2 (c) Aluminum fluoride mass x_3

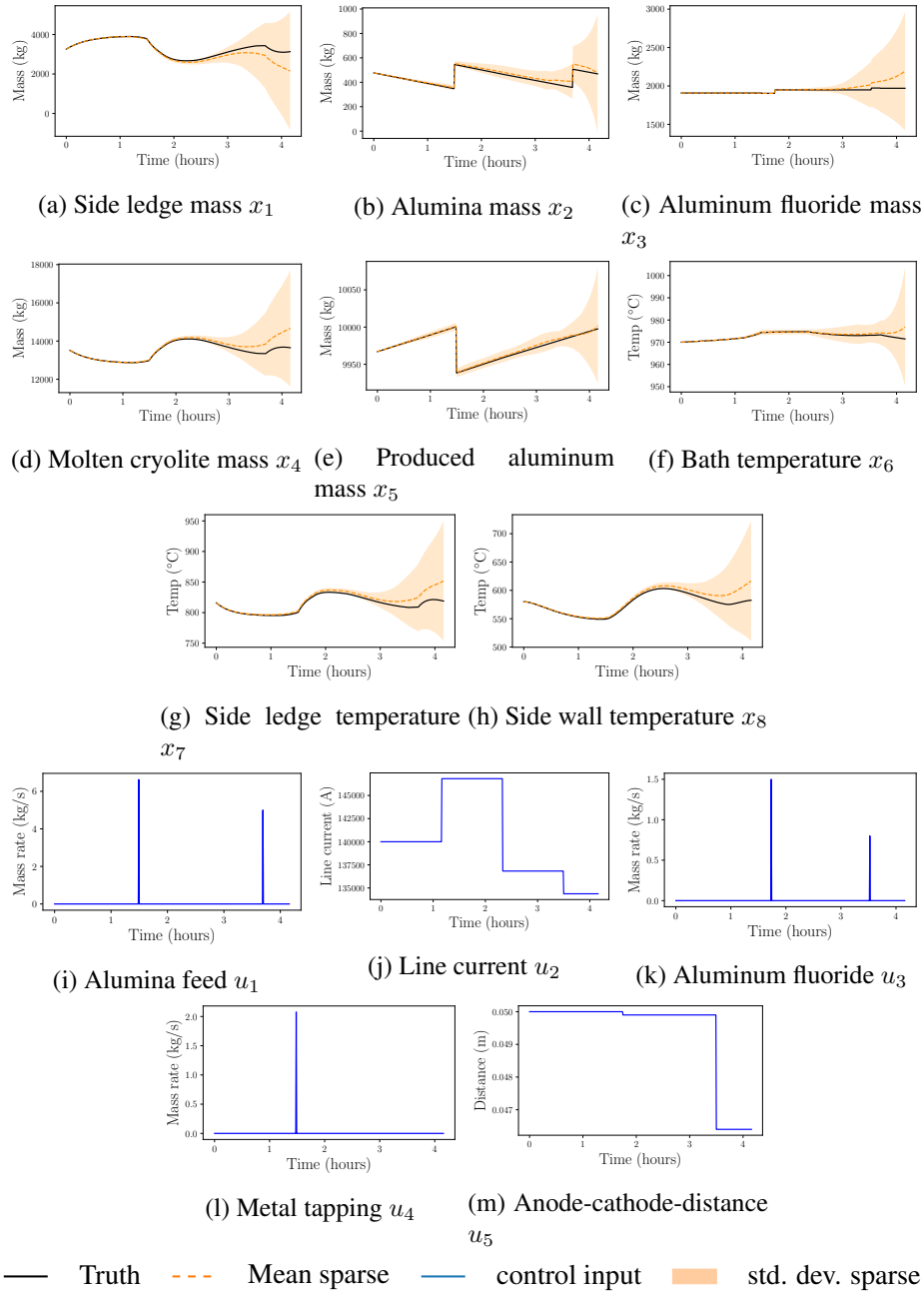
(d) Molten cryolite mass x_4 (e) Produced aluminum mass x_5 (f) Bath temperature x_6

(g) Side ledge temperature x_7 (h) Side wall temperature x_8

(i) Alumina feed u_1 (j) Line current u_2 (k) Aluminum fluoride u_3

(l) Metal tapping u_4 (m) Anode-cathode-distance u_5

Figure 5.12: Sparse rolling forecast of state variables $\{x_1, .., x_8\}$ at each time instant. The true values of $\mathbf{x}$ and control inputs $\mathbf{u}$ are taken from one simulated set of test set trajectories $\mathbf{X}_i \in \mathbf{X}_{test}$. The orange dotted line shows the average of 20 forecasts calculated by 20 sparse neural network models with different parameter initialization. The orange band shows the standard deviation of the same 20 forecasts calculated by 20 sparse models.



(a) Side ledge mass x_1 (b) Alumina mass x_2 (c) Aluminum fluoride mass x_3

(d) Molten cryolite mass x_4 (e) Produced aluminum mass x_5 (f) Bath temperature x_6

(g) Side ledge temperature x_7 (h) Side wall temperature x_8

(i) Alumina feed u_1 (j) Line current u_2 (k) Aluminum fluoride u_3

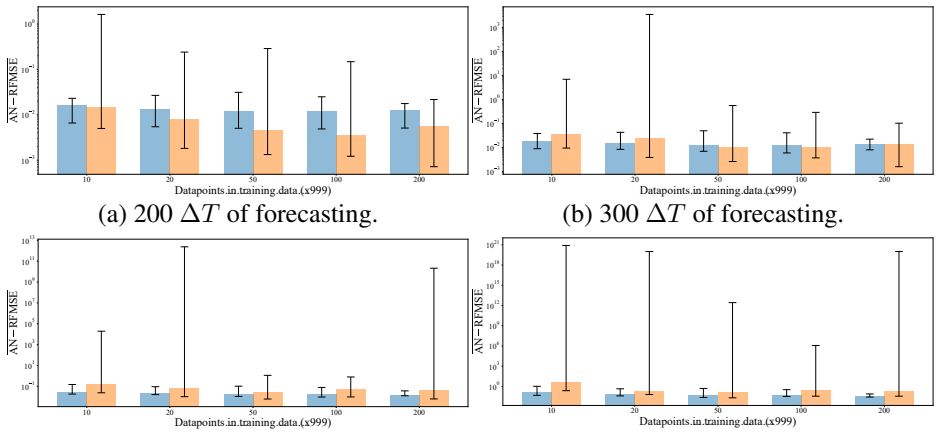
(l) Metal tapping u_4 (m) Anode-cathode-distance u_5

Figure 5.13: Dense rolling forecast of state variables $\{x_1, .., x_8\}$ at each time instant. The true values of $\mathbf{x}$ and control inputs $\mathbf{u}$ are taken from one simulated set of test set trajectories $\mathbf{X}_i \in \mathbf{X}_{test}$. The orange dashed line shows the average of 20 forecasts calculated by 20 dense neural network models with different parameter initialization. The orange band shows the standard deviation of the same 20 forecasts calculated by 20 dense models. The forecast is shown until some of the dense model estimates starts to diverge from true values.

Figure 5.12 and Figure 5.13 indicate that the forecasts of sparse and dense models are showing similar performance for the first time steps after they are given the initial conditions. However, while the forecasts calculated by sparse models show a consistently slow drift from the simulated values of $\mathbf{x}$, the mean and standard deviation of forecasts calculated by dense models suddenly drifts exponentially. The narrow banded standard deviation of sparse neural networks can indicate that these models converge to models with similar characteristics during training despite different parameter initialization. Furthermore, the consistently slow drift between the sparse DNN model forecast of $\mathbf{x}$ and the true values of $\mathbf{x}$ indicate that the sparse models are generalizing better as they are showing good forecasting capabilities in a broader region than the dense DNN models. Figure 5.12 and Figure 5.13 show some interesting results that indicate better generalization of sparse DNN models than dense DNN models and that the convergence of model parameters for sparse DNN models are more robust to random initialization than dense DNN models are.

Impact of the training data quantity and prediction horizon

Figure 5.14 shows the median, maximum and minimum values of AN-RFMSE among test set trajectories.



(a) 200 ΔT of forecasting.

(b) 300 ΔT of forecasting.

(c) 500 ΔT of forecasting.

(d) 1000 ΔT of forecasting.

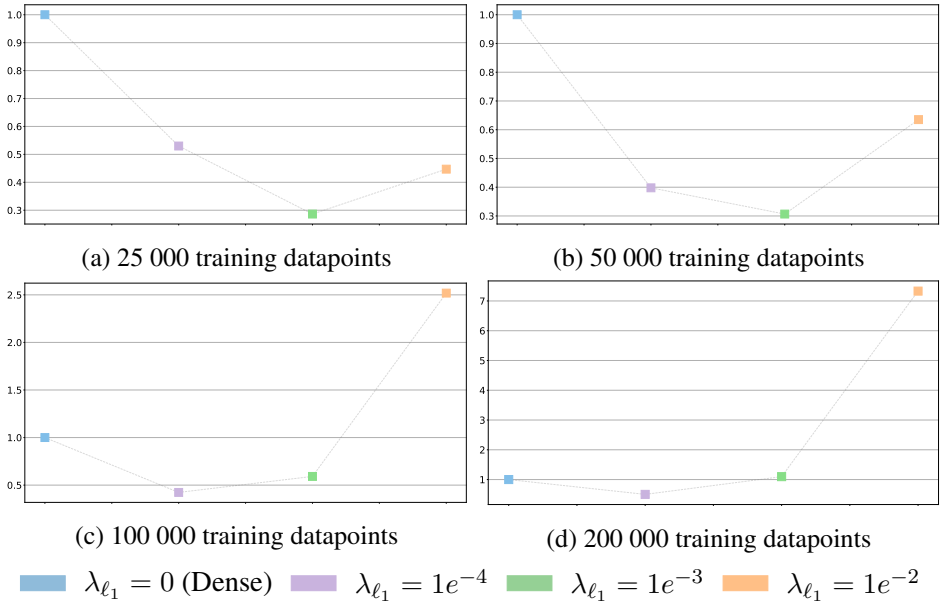
Median AN-RFMSE sparse Median AN-RFMSE dense

Figure 5.14: Median, maximum, and minimum AN-RFMSE values. There are five ensembles of models of both sparse and dense DNNs trained on data sets with different sizes. For each ensemble of models, there is a corresponding colored bar indicating median values and an error bar indicating maximum and minimum AN-RFMSE values. Moreover, the size of the training sets is indicated on the x-axis of the subplots. The blue bar shows the median AN-RFMSE value among 20 sparse DNN models over 20 test set trajectories. The orange bar shows the median AN-RFMSE value among 20 dense DNN models over 20 test set trajectories. For each subfigure, the AN-RFMSE values are calculated for a given number of timesteps reported in the captions of each subfigure (5.14a - 5.14d). Notice the logarithmic scale of the y-axis.

Figure 5.14 contains a good amount of information about the model performance of dense DNN models and the sparse DNN models with weight penalty $\lambda_{\ell_1} = 1e^{-2}$. Figs. 5.14a to 5.14d report median and extreme AN-RFMSE values over four different time horizons for five ensembles of models trained on five data sets with different sizes. Hence, the results in Figure 5.14 show how dense and sparse DNN models perform with varying amounts of training data over varying time horizons. Figs. 5.14a to 5.14d show that the ensembles of **sparse models** trained on data sets with varying size show similar results, both in terms of median and extreme AN-RFMSE values. However, as Figure 5.14d shows, there seems to be a small trend that model ensembles trained with more data perform slightly better over longer forecasting horizons. Furthermore, the band between the minimum and maximum AN-RFMSE values is overall relatively small for all ensembles of models and all forecast horizons for sparse models. The converging

behavior of the performance of ensembles of sparse models as a function of the amount of data in the training set indicates that only small amounts of data are required to gain significance for the model parameters. While the sparse models show stable performance across ensembles of models with different amounts of training data and slowly increasing AN-RFMSE values proportional to the length of the forecasting horizon, the same cannot be said about the performance of the dense models. When considering the **dense models**, Figs. 5.14a to 5.14d indicate that there is a trend where both median, minimum and maximum AN-RFMSE values decrease significantly as sizes of training set decreases. This expected trend indicates that the performance improves with increasing dataset size. However, the trend is not consistent for all ensembles of dense DNN models for all forecasting horizons. Furthermore, the maximum AN-RFMSE values for ensembles of dense DNN models for longer forecasting horizons such as in Figure 5.14c and Figure 5.14d show that the AN-RFMSE exponentially increases for some of the models within the ensemble. This indicates that the dense DNN models are likely to have some input regions where the model output is not sound. If the model estimate enters a region with poorly modeled dynamics, the model estimate might drift exponentially. For short-term prediction, that is in Figure 5.14a and Figure 5.14b, the trend is that dense models show better performance for median and minimum values than sparse models, especially within the ensembles with large training sets. This may be because dense models have more flexibility in terms of more parameters. However, it is important to state that the weights of the sparse models evaluated in Figure 5.14 are especially hard penalized, indicating that the flexibility of these models are limited. For longer forecasting horizons (Figure 5.14c and Figure 5.14d), sparse models are always showing better performance than dense models in terms of median AN-RFMSE values. This is a typical example of a bias-variance trade-off. For all forecasting horizons and within all groups of training set sizes, sparse models are always showing smaller maximum AN-RFMSE values.

While it is valuable to have models that can give reasonable estimates in the long term, short-term prediction accuracy can be given extra attention since the models typically perform best on shorter horizons, and can therefore be used more aggressively for optimal control. As observed above, dense models tend to give better median accuracy than sparse models with hard ℓ_1 regularization in shorter prediction horizons if trained on larger datasets. We, therefore, conducted a study where we compared dense models with sparse models trained with different levels of weight regularization with different sizes of the training set. Figure 5.15 show the mean prediction accuracy of an ensemble of dense models and three different ensembles of sparse models with different degree of weight penalization (namely $\lambda_{\ell_1} = 1e^{-4}$, $\lambda_{\ell_1} = 1e^{-3}$ and $\lambda_{\ell_1} = 1e^{-2}$). Each ensemble consists of 20 mod-



(a) 25 000 training datapoints

(b) 50 000 training datapoints

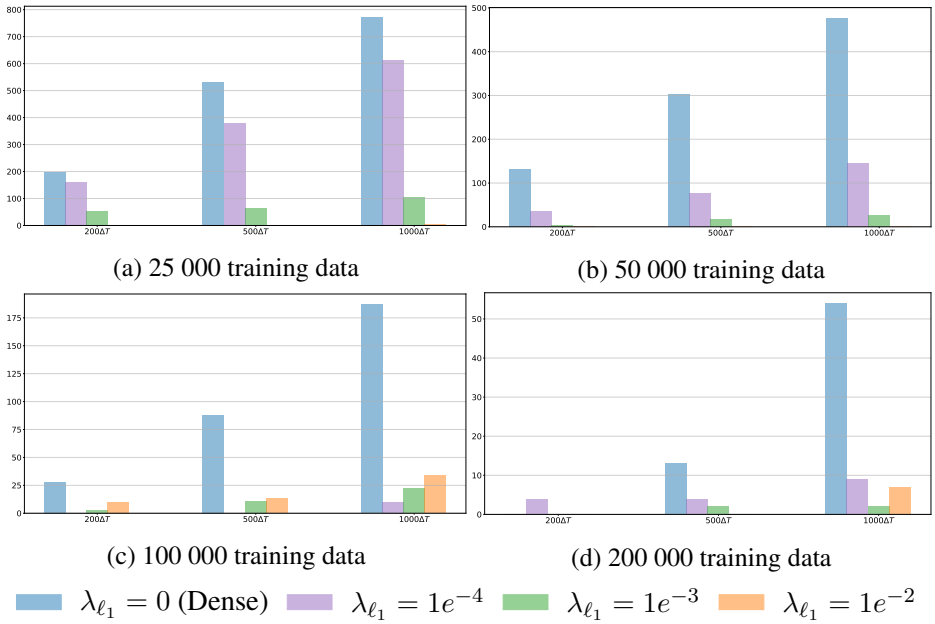
(c) 100 000 training datapoints

(d) 200 000 training datapoints

Figure 5.15: Median prediction error ratio for short prediction horizon of $200\Delta T$. All median prediction errors are divided by the median prediction error of dense models. Thus, dense models will always have a prediction error ratio of 1.

els, and each of the models forecast 200 timesteps of 50 different test trajectories. The median accuracy is expressed in terms a ratio called median prediction error ratio. This ratio is given by of the ratio between the median prediction error of the given ensemble of models and the median prediction error of the ensemble of dense models. This means that the median prediction error ratio of dense models is always one, and that smaller ratio indicates higher median accuracy. For low and medium data sizes (Figure 5.15a and 5.15b), the median accuracy of all ensembles of sparsely regularized models are more accurate than dense models also in the short term. In this data regime, the different sparse models show similar prediction capabilities. In Figure 5.15c, the median prediction error ratio for models trained on 100 000 data points are plotted. In this data regime, the median accuracy ratio between medium sparse models ($\lambda_{\ell_1} = 1e^{-3}$ and $\lambda_{\ell_1} = 1e^{-4}$) and dense models remains quite similar to the median accuracy ratio between dense and medium sparse models in the low data regime. Moreover, the prediction error ratio of very sparse models ($\lambda_{\ell_1} = 1e^{-2}$) and the other models increase significantly. This is probably because the model accuracy of very sparse models converges at low data limits. In contrast, medium sparse and dense models can exploit their nonlinear prediction capabilities when large amounts of training data are available. In Figure 5.15d, median prediction error ratios for models trained on 200 000 data samples are presented. At this point, sparse models trained with $\lambda_{\ell_1} = 1e^{-4}$ has a median model prediction error ratio of 0.5, sparse models trained with $\lambda_{\ell_1} = 1e^{-3}$ has a prediction error ratio of 1.1 and sparse models trained with $\lambda_{\ell_1} = 1e^{-2}$ has a prediction error ratio of approximately 7. In this large data regime, it seems clear that the medium sparse, and dense models are very accurate. In contrast, the accuracy of very sparse models converges for small training datasets. Furthermore, the sparse model trained with the smallest sparsity regularization parameter ($\lambda_{\ell_1} = 1e^{-4}$) still outperforms the dense models. This study illustrates that dense models require enormous amounts of data to outperform sparse models.

To quantify the exponential drift or blow-up of model estimates for different levels of sparsity, we run a test on ensembles of models trained on datasets with different data sizes. The measure used in Figure 5.16 is defined in Equation (2.47).



(a) 25 000 training data

(b) 50 000 training data

(c) 100 000 training data

(d) 200 000 training data

Figure 5.16: Effect of regularization on the number of blowups for different amount of training data

Figure 5.16 shows the frequency of blowups for models with different degree of ℓ_1 regularization ranging from $\lambda_{\ell_1} = 0$ giving dense DNNs to $\lambda_{\ell_1} = 1e^{-2}$ giving very sparse DNNs. The models are trained on four different datasets, and for each dataset and each value of regularization parameter λ_{ℓ_1}, an ensemble of 20 DNNs are trained. The models are then tested on 50 different simulated test trajectories, yielding 1000 possible blowups for each ensemble of models. The training datasets consist of respectively 25 000, 50 000, 100 000 and 200 000 datapoints to show the effect of sparsfication on preventing blow-ups for a range of training set sizes. In each subfigure of Figure 5.16, the number of blow-ups are given after three different prediction horizons. For the models trained on the smallest training set (show in Figure 5.16a and Figure 5.16b) the trend is clearly that the higher degree of sparsity, the lesser the blow ups. For models trained on larger datasets (Figure 5.16c and Figure 5.16d) the dense models are still having significantly more blow ups than all the sparse models. However, when we compare very sparse models ($\lambda_{\ell_1} = 1e^{-2}$) with medium-, and little sparse models ($\lambda_{\ell_1} = 1e^{-3}$ and $\lambda_{\ell_1} = 1e^{-3}$ respectively), the difference becomes less significant. This can maybe be explained by the fact that the amount of data converges to a sufficient level also for the models trained with smaller sparsity promoting regularization on

the parameters.

5.4.3 Training stability perspective

Sparsification on a large ensemble of neural networks gives similar sparse structures. This has been shown in the structure plots in the Figs.5.4-5.11. Furthermore, Table 5.3 show that sparse models to a large extent finds the same feature basis for each of the model outputs $\{\hat{f}_1,\ ...,\ \hat{f}_8\}$.

Moreover, Figure 5.14 clearly indicates for all forecasting horizons that the uncertainty bounds for dense models are much larger than those for sparse models. For the longer horizons, some of the dense models tend to blow up. This is probably due to that the model enters a region of the input space where it overfits. This can be seen as poor generalization to that specific area. Figure 5.16 quantify this by showing that state estimates calculated by dense models tend to blow up much more frequently than sparse models for all forecasting horizons and all training dataset sizes used in the case study. This indicates that the risk of finding bad minimas is higher for dense models. Furthermore, sparse models are more likely to converge to reasonable minimas with smaller training data than dense models. Hence, the study shows that sparse models have better training stability than dense models with limited data.

5.5 Conclusions and future work

This chapter presents a sparse neural network model that approximates a set of nonlinear ODEs based on time series data sampled from the system variables included in the ODEs. The set of nonlinear ODEs represents an aluminum electrolysis simulator based on the mass and energy balance of the process. This includes nonlinear and interrelated models of electrochemical and thermal subprocesses. The sparsity in the model is achieved by imposing sparsity using ℓ_1 regularization on the weights. The main conclusions from the study can be itemized as follows:

- ℓ_1 regularization drastically reduces the number of parameters in the DNN. In our case we witnessed a 93% reduction in the parameters compared to the corresponding dense DNN for a regularization parameter of $\lambda_{\ell_1} = 1e^{-2}$.
- The sparse neural network was more interpretable using the domain knowledge of the aluminium electrolysis process. In contrast, the dense neural networks were completely black-box.
- Sparse neural networks were consistently more stable than their dense counterparts. This was reflected in the model uncertainty estimates based on a large ensemble of models. Furthermore, dense model estimates tend to di-

verge from the states that they are estimating way more often than sparse models. This means that the parameters of sparse neural networks are more likely to end up at a reasonable minima than the parameters of dense DNN with limited training data.

- For small to medium amounts of data, even the most sparse models have better median prediction accuracy than the dense models for a short prediction horizon.
- For medium to large amounts of training data, sparse models with low weight penalization still has better prediction accuracy than dense models in the short term. However, dense models outperform very sparse models in terms of median prediction accuracy in short prediction horizons. For longer prediction horizons, sparse models outperform dense models both in terms of higher median accuracy.

While the sparse models show promising results within interpretability and generalizability, there is still a high potential for improvement. There is a desire to increase prediction accuracy and decrease the bias of the sparse models. This might be addressed by investigating other sparsity structures at different layers that better compromise the bias-variance trade-off. One possible direction is to inject simplified theories known from first principle into the neural network to possibly increase accuracy. Lastly, we have not addressed the additional challenges related to the presence of noise.

Chapter 6

Modeling dynamics using Neural Networks with skip-connections

This chapter is based on the article:

[81] **E. T. B. Lundby**[1], H. Robinson[1], A. Rasheed, J. T. Gravdahl, "Sparse neural networks with skip-connections for identification of aluminum electrolysis cell" Pending Review: IEEE CDC. (2023) Available in: arXiv preprint arXiv:2301.00582 (2023)

Here, skip connections are introduced in the DNN model structure. Furthermore, the combined effect of skip-connections and sparsity promoting ℓ_1 regularization is examined.

6.1 Introduction

Recent research has found that using sparser networks may be the key to training models that can generalize across many situations. In particular, [88] showed empirically that for any dense architecture, there is a very high probability that there is a sparse subnetwork which will train faster and generalize better than the full model. This is known as the Lottery Ticket Hypothesis, and many methods of sparsification can be seen as attempts to somehow extract such a "winning lottery ticket" from an initially dense network. There have been numerous advances in this field, and we refer to [91] for a recent and comprehensive review. In this work we use the well-known ℓ_1 regularization for sparsification of the model.

A challenge related to the use of NNs is the choice of architecture and hyper-

[1] Equal contributions

parameters. Typical networks have multiple layers which are densely connected, although this can vary between domains. Choosing an appropriate architecture is an art, generally involving trial and error to improve performance and avoid overfitting. It is commonly understood that the early layers of a neural network have a significant impact on the overall performance of the network, but deep networks often suffer from the vanishing or exploding gradient problem which prevents effective training of these early parameters [56]. Skip-connections were originally proposed by [93] as a way to circumvent this, by introducing a shorter path between the early layers and the output. They were not only found to enable the training of significantly deeper networks, but [130] also demonstrated that they may help improve training convergence.

In this work, we investigate the effects of adding skip connections and ℓ_1 regularization on the accuracy and stability of these models for short, medium, and long horizons. We address the following questions:

- How do skip connections affect the stability and generalization error of neural networks trained on high-dimensional nonlinear dynamical systems?
- How does sparsity affect stability and generalization error for neural networks with skip connections that model nonlinear dynamics?
- How does the amount of training data affect neural networks with skip connections compared to neural networks without skip connections?

We make the following contributions:

- We perform a black box system identification of an aluminum electrolysis cell using different NN architectures.
- We demonstrate that the accuracy and open-loop stability of the resulting models is greatly improved by using ℓ_1 weight regularization and incorporating skip connections into the architecture.
- This advantage is consistent across datasets of varying sizes.

The chapter is structured as follows. Section 6.2 describes the proposed method and the experimental setup. In Section 6.3, results are presented and discussed, and in Section 6.4, conclusions are made.

6.2 Method and setup

In this section, we present all the details of data generation and its preprocessing, and the methods that are required to reproduce the work. The steps can be briefly summarized as follows:

- Use Equation (2.15) with random initial conditions to generate 140 trajectories with 5000 timesteps each. Set aside 40 for training and 100 for testing. Construct 3 datasets by selecting 10,20 and 40 trajectories respectively.
- For each model class and dataset, train 10 instances on the training data.
- Repeat all experiments with ℓ_1 regularization, see loss function in Equation (6.2).
- Use trained models to generate predicted trajectories along the test set and compare them to the 100 test trajectories.

6.2.1 InputSkip

In this work, we utilize a modified DenseNet architecture as proposed by [82], where the outputs of earlier layers are concatenated to all the consecutive layers. We simplify the structure such that the model only contains skip connections from the input layer to all consecutive layers. We call this architecture InputSkip, which has reduced complexity compared to DenseNet.

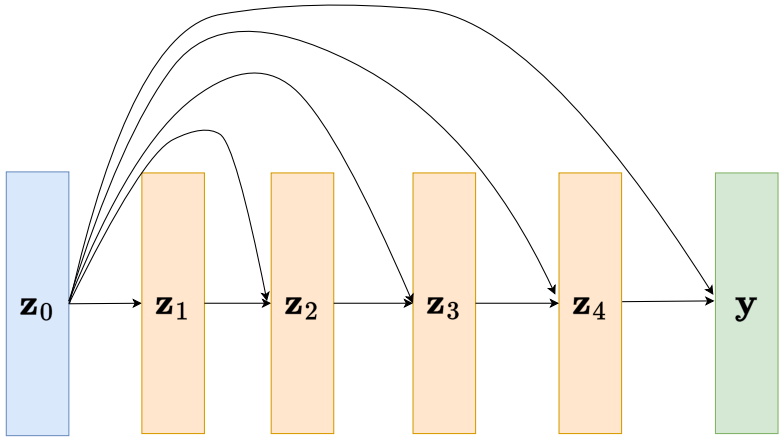


Figure 6.1: InputSkip architecture with 4 hidden layers

The model InputSkip model structure is illustrated with 4 hidden layers in Figure 6.1. This design is motivated by the fact that the output of each layer (including the final output) becomes a sum of both a linear and a nonlinear transformation of the initial input $\mathbf{x}$. Hence, the skip connections from the input layer to consecutive layers facilitate the reuse of the input features for modeling different linear and nonlinear relationships more independently of each other. The structure can be described mathematically as follows:

$$\begin{aligned} \mathbf{z}_1 &= \boldsymbol{\sigma}_1(\mathbf{W}_1\mathbf{z}_0 + \mathbf{b}_1) \\ \mathbf{z}_i &= \boldsymbol{\sigma}_i\left(\mathbf{W}_i \begin{bmatrix} \mathbf{z}_{i-1} \\ \mathbf{z}_0 \end{bmatrix} + \mathbf{b}_i\right), i > 1, \end{aligned} \tag{6.1}$$

where $\mathbf{z}_0$ is the input layer, $\mathbf{z}_i$ for $i > 0$ are hidden layers and $\mathbf{y}$ is the output layer. The output of each layer (excluding the first) becomes a sum of a linear and a nonlinear transformation of the initial input x. Hence, the skip connections from the input layer to consecutive layers facilitate the reuse of the input features for modeling different linear and nonlinear relationships more independently.

6.2.2 Data generation

Equation (2.15) was discretized using the RK4 scheme with a fixed timestep $h = 10\,\mathrm{s}$ and numerically integrated on the interval $[0, 5000h]$. We used uniformly randomly sampled initial conditions from the intervals shown in Table 6.1 to generate 140 unique trajectories. We set aside 40 trajectories for training and 100 of the trajectories as a test set. The 40 training trajectories were used to create 3 datasets of varying sizes (small, medium, large), namely 10, 20, and 40 trajectories. In total, the datasets contained 50000, 100000, and 200000 individual data points respectively.

Equation (2.15) also depends on the input signal $\mathbf{u}$. In practice, this is given by a deterministic control policy $\mathbf{u} = \boldsymbol{\pi}(\mathbf{x})$ that stabilizes the system and keeps the state $\mathbf{x}$ within some region of the state space that is suitable for safe operation. We found that this was insufficient to successfully train our models, because the controlled trajectories showed very little variation after some time, despite having different initial conditions. This lack of diversity in the dataset resulted in models that could not generalize to unseen states, a situation that frequently arose during evaluation. To inject more variety into the data and sample states $\mathbf{x}$ outside of the standard operational area, we used a stochastic controller

$$\boldsymbol{\pi}_s(\mathbf{x}) = \boldsymbol{\pi}(\mathbf{x}) + \mathbf{r}(t)$$

that introduced random perturbations $\mathbf{r}(t)$ to the input. These perturbations were sampled using the APRBS method proposed by [117] for nonlinear system identification.

Table 6.1: Initial conditions intervals for $\mathbf{x}$

Variable	Initial condition interval
x_1	$[2060,\ 4460]$
c_{x_2}	$[0.02,\ 0.05]$
c_{x_3}	$[0.09,\ 0.13]$
x_4	$[11500,\ 16000]$
x_5	$[9550,\ 10600]$
x_6	$[940,\ 990]$
x_7	$[790,\ 850]$
x_8	$[555,\ 610]$

In system identification it is typical to optimize the model to estimate the function $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$. However, this is not feasible for Equation (2.15) because the inputs $\mathbf{u}$ are not differentiable. Instead, we discretize the trajectories using the forward Euler difference and use this as the regression variable:

$$\mathbf{y}_k = \frac{\mathbf{x}_{k+1} - \mathbf{x}_k}{h}$$

The datasets are then constructed as sets of the pairs $([\mathbf{x}_k, \mathbf{u}_k], \mathbf{y}_k)$.

6.2.3 Training setup

We optimize the models by minimizing the following loss function using stochastic gradient descent:

$$\mathbf{J}_\theta = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} (\mathbf{y_i} - \hat{\mathbf{f}}(\mathbf{x_i}, \mathbf{u_i}))^2 + \lambda \sum_{j=1}^{L} |\mathbf{W}_j| \tag{6.2}$$

where $\mathcal{B}$ is a *batch* of randomly sampled subset of indices from the dataset, L is the number of layers of the NN, and λ is the regularization parameter. This loss function is the sum of the MSE of the model $\hat{\mathbf{f}}$ with respect to the regression variables $\mathbf{y}$, and the ℓ_1 norm of the connection weight matrices $\mathbf{W}_i$ in all layers. We used a batch size of $|\mathcal{B}| = 128$. We used the popular ADAM solver proposed by [118] with default parameters to minimize Equation (6.2).

6.3 Results and discussions

We characterize the different model classes (PlainDense, PlainSparse, InputSkipDense, InputSkipSparse) by estimating their blow-up frequencies and their Rolling Forecast Mean Squared Error (RFMSE) on the validation data. The blow-up frequency is an interesting measure since it can indicate how stable the model is

in practice. The measures used in this section are defined in Section 2.4 as the AN-RFMSE defined in Equation (2.46), and the instability measure defined in Equation (2.47).

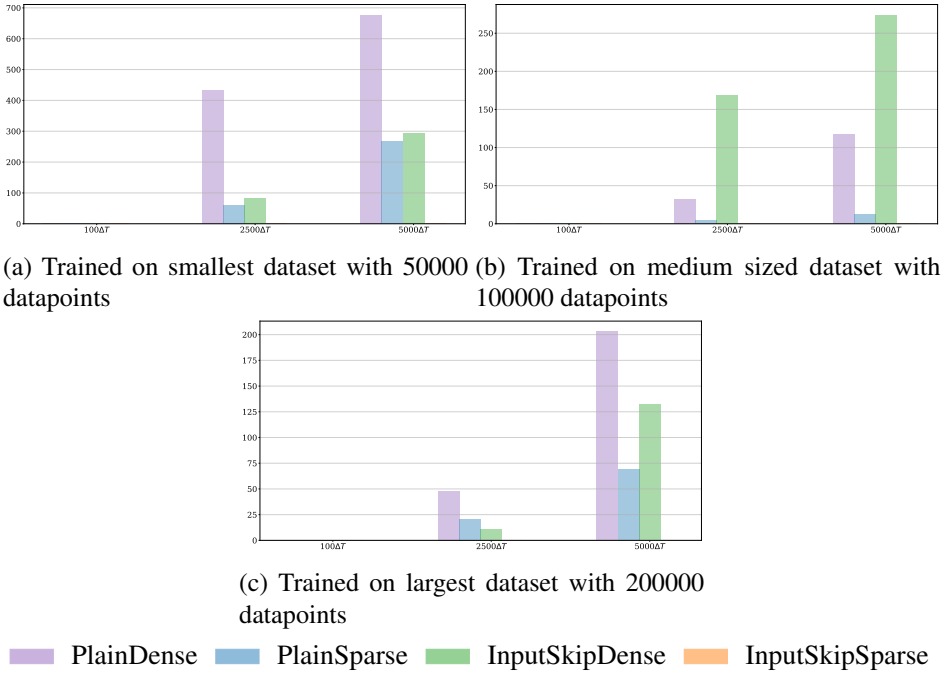
We perform a Monte Carlo analysis by training 10 instances of each model class and evaluating these on 100 trajectories randomly generated using the true model, yielding 1000 data points for each model class. We repeat the experiments for 3 different dataset sizes to study the data efficiency of the models.

Figure 6.2 presents the total number of blow-ups recorded within each model class after $100h$, $2000h$, and $5000h$ (short, medium, and long term respectively). For simplicity, blow-ups were detected by thresholding the computed variance of a predicted trajectory and manually inspected. It is clear that for short time horizons all the models exhibit robust behavior independently of the size of the training datasets. However, for medium and long time horizons, PlainDense, PlainSparse, and InputSkipDense architectures exhibit a significant number of blow-ups and therefore instability. Figure 6.2a - 6.2c show that PlainDense is generally the most unstable, with up to 67% of all trajectories resulting in a blow-up. For the smallest amount of training data (Figure 6.2a) PlainSparse and InputSkipDense have similar blow-up frequencies. For larger datasets, the PlainSparse architecture shows significantly better stability than both PlainDense and InputSkipDense. InputSkipDense and PlainDense both show better stability with increasing amounts of training data in terms of fewer blow-ups. However, both these dense models still suffer from significant amounts of blow-ups.

In comparison, almost no blow-ups are recorded when using the InputSkipSparse architecture, even for the small training dataset. In Figure 6.2, the orange bars corresponding to the blow-up frequency of InputSkipSparse models are not visible for any of the training sets due to the significantly lower number of blow-ups. For InputSkipSparse models trained on the smallest dataset, only 3 out of 1000 possible blow-ups were reported for the longest horizon. Apart from that, no blow-ups were reported for the InputSkipSparse models.

Only a few blow-ups were recorded after $5000h$ in the medium term.

Figure 6.3 presents a violin plot of the accuracy of each model class, expressed in terms of RFMSE over different time horizons. Only the plot for the smallest dataset (50000 points) is shown, due to the results being very similar. A larger width of the violin indicates a higher density of that given RFMSE value, while the error bars show the minimum and maximum recorded RFMSE values. The model estimates that blew up (see Figure 6.2) are not included. In this way, we estimate the generalization performance of the models only within their regions of



(a) Trained on smallest dataset with 50000 datapoints

(b) Trained on medium sized dataset with 100000 datapoints

(c) Trained on largest dataset with 200000 datapoints

Figure 6.2: Divergence plot: Number of trajectories that blow-up over different time horizons. The total number of trajectories is 1000, so the values can be read as a permille.

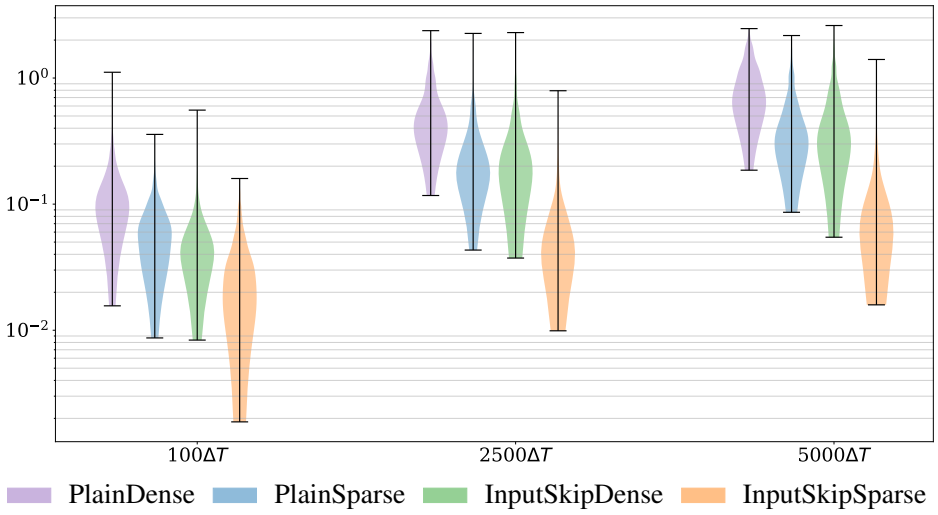


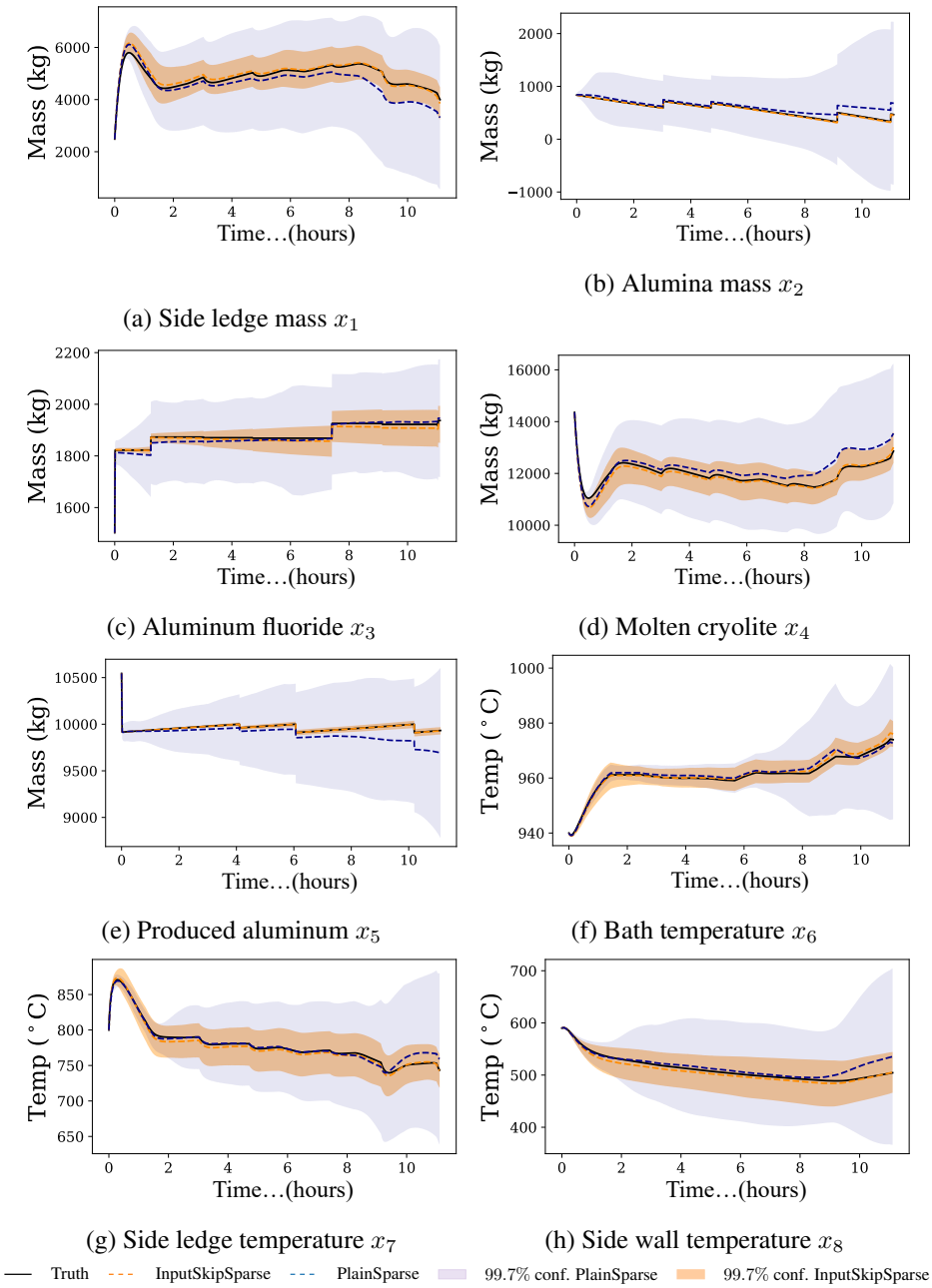
Figure 6.3: Model accuracy expressed in terms of AN-RFMSE over different horizons. Ten models of each of the model types (PlainDense, PlainSparse, InputSkipDense, InputSkipSparse) are trained on the smallest dataset of 50000 data points. The model estimates that blow up (see Figure 6.2) are excluded. The plot shows that sparse models with skip connections (InputSkipSparse) are consistently more accurate than both sparse and dense models without skip connections.

stability. Note that the violin plots for model classes with many blow-ups are made using fewer samples, and can be seen as slightly “cherry-picked". Nonetheless, the InputSkipSparse architecture consistently yields more accurate results, up to an order of magnitude better than the others in the long term.

6.4 Conclusion and future work

In this work, we compared the performance of two different model structures trained both with and without sparsity promoting ℓ_1 regularization. The two model types are standard MLPs, and a more specialized architecture that includes skip connections from the input layer to all consecutive layers. This yields four different model structures, which we call PlainDense, PlainSparse, InputSkipDense, and InputSkipSparse. The main conclusions of the chapter are as follows:

- NNs with skip connections are more stable for predictions over long time horizons compared to standard MLPs. Furthermore, the accuracy of NNs with skip connections is consistently higher for all forecasting horizons.



(a) Side ledge mass x_1

(b) Alumina mass x_2

(c) Aluminum fluoride x_3

(d) Molten cryolite x_4

(e) Produced aluminum x_5

(f) Bath temperature x_6

(g) Side ledge temperature x_7

(h) Side wall temperature x_8

Figure 6.4: Rolling forecast of a representative test trajectory

- The application of sparsity-promoting ℓ_1 regularization significantly improves the stability of both the standard MLP and InputSkip architectures. This improvement was more apparent for models with the InputSkip architecture.
- The InputSkipSparse showed satisfactory stability characteristics even when the amount of training data was restricted. This suggests that this architecture is more suitable for system identification tasks than the standard MLP structure.

The case study shows that both sparsity-promoting regularization and skip connections can result in more stable NN models for system identification tasks while requiring less data, as well as improving their multi-step generalization for both short, medium, and long prediction horizons. Despite the encouraging performance of the sparse-skip networks, we can not guarantee similar performance for noisy data, as we have only investigated the use of synthetic data devoid of any noise. However, such a study will be an interesting line of future work. This case study also has relevance beyond the current setup. In more realistic situations, we often have a partial understanding of the system we wish to model (see Equation (2.15)), and only wish to use data-driven methods to correct a PBM when it disagrees with the observations (e.g. due to a faulty assumption). As shown in [79], combining PBMs and data-driven methods in this way also has the potential to inject instability into the system. Finding new ways to improve or guarantee out-of-sample behavior for data-driven methods is therefore of paramount importance to improve the safety of such systems.

Chapter 7

Deep active learning in experimental design for nonlinear system identification

This chapter is based on the article in:

[83] **E. T. B. Lundby**, A. Rasheed, I. J. Halvorsen, D. Reinhardt, S. Gros, J. T. Gravdahl, "Deep active learning for nonlinear system identification", In arXiv preprint arXiv:2302.12667 (2023).

The contribution propose a novel DeepAL acquisition scheme. A set of simulated state-action trajectories are obtained through local explorations. This simulated set of state-action trajectories enables a novel static BMDAL acquisition scheme that aims to acquire the batch of most informative regions among the candidates. The dynamics are then explored locally in the acquired regions to acquire data from the system.

7.1 Introduction

While having extrordinary capabilities to model complex input-output relations, DNNs typically need vast data to obtain levels of accuracy and generalizability acceptable to apply them in safety-critical systems. AL aims to maximize model accuracy with a minimal amount of data by acquiring the most informative training data [67]. This has been noticed by researches aiming to identify the dynamics of different systems with more simple ML models such as GP models [74], linear models [76], and restricted class of nonlinear models [77]. However, the research on active learning for system identification of NN models is, to the best of the

authors' knowledge, highly limited. To that end, we extend the work of AL used to acquire the most informative data for system identification to NNs. That is,

- In equation (7.7), we formalize a general BMDAL acquisition scheme for dynamical system identification referred to in this work as global exploration. The scheme is on the static deep active learning batch acquisition form presented in equation (2.32) from [67]. The static acquisition scheme for dynamical systems is enabled through local explorations, obtaining a set of simulated informative candidate state-action trajectories distributed around the state-action space. The novel DeepAL scheme builds upon the AL scheme presented in [75] that iteratively searches for the single most informative state-action trajectory for identifying a GP model through local and global explorations.
- The novel formulation of the static BMDAL acquisition for dynamical systems is utilized in a novel framework presented in Fig. 7.2. Ensembles of NNs are used to produce uncertainty estimates that assess the informativeness of single state-action trajectories.
- The general nature of the proposed BMDAL formulation allows for a wide range of query strategies to be applied. This is demonstrated by using two different query strategies in the case study.

The AL algorithm is showcased on the simulated dynamics of a 3 Degree Of Freedom (DOF) surface vessel with three states and three control inputs, yielding an input space of six dimensions. The simulator represents the dynamics of the MilliAmpere ferry[87], which is an experimental platform owed by NTNU. The simulation model is presented in Section 2.1.3.

7.2 Ensembles of neural networks

Ensemble learning includes methods that combine multiple models in making predictions. The main premise in ensemble learning is that errors made by individual models are likely to be compensated by other models such that the overall ensemble prediction on average improves prediction accuracy over individual models [131]. Deep ensembles, consisting of multiple DNNs, have gained significant attention in recent years due to their improved accuracy, uncertainty estimation, and robustness to out-of-distribution data. There are two well known methods of training ensembles of NNs; by bootstrapping, where the ensemble methods are trained on different bootstrap samples of the dataset, and by multiple random seeds, where the model parameters of the members are intialized with different random parameters and then trained on the entire dataset. While the bootstrapping method has

been found to hurt performance of the NNs, using random initializations turn out to be a promising approach [132, 133]. In [134], the success of random initialization in deep ensembles is explained by the fact that this method explore diverse modes of the function space. That is, Bayesian neural networks, which do not perform as good as deep ensembles, only explore the proximity of one single mode of the function space. Ensembles can provide both point estimates and uncertainty estimates. The point estimate can be calculated as an average of the predictions. Consider an ensemble of M NNs with different parameter initialization. When forecasting several timesteps ahead without feedback from measurements, we let each ensemble member estimate individual state forecast trajectories. The state forecast ${}^{j}\hat{\mathbf{x}}_{t+1}$ at time $t+1$ provided by an ensemble member $\hat{\mathbf{f}}_j$ is given by forward Euler integration:

$$ {}^{j}\hat{\mathbf{x}}_{t+1} =^{j} \hat{\mathbf{x}}_t + \hat{\mathbf{f}}_j({}^{j}\hat{\mathbf{x}}_t, \mathbf{u}_t; \boldsymbol{\theta})\Delta T, \tag{7.1} $$

where ΔT is the time step. Then, the average prediction of M predictions calculated by individual NN ensemble members at timestep $t+1$ is given by:

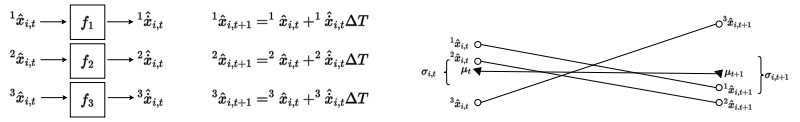
$$ \boldsymbol{\mu}_{t+1} = \frac{1}{M}\sum_{j=1}^{M} {}^{j}\hat{\mathbf{x}}_{t+1}, \tag{7.2} $$

Uncertainty-based strategies are utilized in the proposed AL method. Therefore, the main motivation of using deep ensembles in this work is to obtain an uncertainty estimate. NN predictions have two sources of uncertainty, namely model uncertainty (also known as epistemic uncertainty), and data uncertainty (also known as aleatoric uncertainty). Usually, these types of uncertainties are modeled separately. In this study, the data is sampled from a fully observed process without process disturbances or measurement noise. Thus, only model uncertainty is considered here. The model uncertainty is caused by shortcomings in the model. This includes errors in the *training procedure* such as bad training hyperparameters (learning rate, batch size, regularization etc.), *insufficient model structure*, or *lack of information in the data* [135]. In general, there are four different types of methods of estimating the uncertainty of a NN based on whether the NNs are deterministic or stochastic, and whether a single NN or multiple NNs are used to estimate the uncertainty. These are *single deterministic methods*, *Bayesian methods*, *ensemble methods* and *test-time augmentation methods* [135]. Ensemble methods have proven to be attractive in quantifying uncertainty of NNs. Ensemble methods are in several works compared to Bayesian methods. In [136], it is argued that ensemble based methods preform better than Bayesian Monte-Carlo Dropout approximations in DeepAL due to more calibrated predictive uncertainties. Both [137] and [138] came to the same conclusions, particularly under dataset shift.

A simple way to quantify the uncertainty of these predictions is to calculate the empirical variance of NN predicitions for each output of the network

$$\boldsymbol{\sigma}_{t+1}^2 = \frac{1}{n}\sum_{j}^{n}({}^{j}\hat{\mathbf{x}}_{t+1} - \boldsymbol{\mu}_{t+1})^2. \tag{7.3}$$

Here, ${}^{j}\hat{\mathbf{x}}_{t+1}$ is the NN prediction made by ensemble member j of $\mathbf{x}_{t+1}$, and $\boldsymbol{\mu}_{t+1}$ is the mean ensemble prediction at time step $t+1$.



(a) Forecasts calculated by ensemble members

(b) Fusion of individual ensemble forecasts

Figure 7.1: Ensemble

Figure 7.1a illustrate how individual ensemble members independently calculate a rolling forecast of the state vector, and Figure 7.1b illustrates how the set of trajectories calculated by the models in the ensemble produce a mean estimate and an uncertainty estimate.

7.3 Deep active learning in dynamical systems

Although data from a dynamical system may be readily available from production or operation, it often provides limited information and is not well-suited for the purpose of system identification. Due to the physical nature of dynamical systems, arbitrary points in the state-action space cannot be directly accessed. In order to sample given data points from the state-action space, the dynamics must be excited by control inputs. This is a dynamic acquisition problem. In an attempt to maximize the information contained in this sampling process, an OCP can be defined over a finite horizon, maximizing some measure of information. This is here referred to as the *local exploration*. As the name indicates, this optimization is only efficient for shorter horizons, limiting the method to explore the dynamics in the proximity of the initial state. However, when identifying the input-output mapping of a nonlinear dynamical system, the entire operational window of the system must be explored.

Assuming a set of input data is already available, AL offers a robust approach for selecting the most informative data points from the input space. In the current acquisition problem for dynamical systems, we do not have access to a pre-sampled

dataset. However, by locally exploring different parts of the input space, a set of simulated state-action trajectories can be obtained. With an available simulated dataset, a static AL acquisition problem for dynamical systems can be formulated. This is referred to as *global exploration*. The global exploration acquires the batch of initial states corresponding to the batch of state-action trajectories that maximizes a global batch acquisition function. Following the acquisition of a set of initial states through global exploration, a subsequent round of local exploration is conducted for each state in the batch. This local exploration entails a longer optimization horizon compared to the initial search conducted for all candidates during the global exploration. This is because the computational complexity of the OCP increases significantly with the horizon, making it necessary to restrict the horizon to a relatively short length when optimizing for all candidates prior to the global exploration. When control trajectories are obtained from the final local explorations these trajectories are applied to the real system from the corresponding acquired initial states. This is done under the assumption that the system is driven to each initial state using a specific control law. As the system evolves under the applied control sequences, data on the system states is collected.

Local exploration

Data sampled from a dynamical system should be properly excited by a control signal to obtain informative data that can be used for system identification. Local exploration refers to the dynamic AL acquisition problem of finding a control trajectory that informatively excites the system. Given an initial state $\mathbf{x}_0$ from where the dynamical system is excited, the local exploration can be formulated as an open loop finite horizon OCP , which yields a sequence of control inputs $\{\mathbf{u}\}_{k=0}^{T-1,\,*} = \{\mathbf{u}_0,\ \mathbf{u}_1,\ ...,\ \mathbf{u}_{T-1}\}^*$. In the context of active learning, the objective function is an acquisition function a_{local} that measures the informativeness of the sequence of forecasted states $\{\hat{\mathbf{x}}_t\}_{t=1}^{T} = \{\hat{\mathbf{x}}_1,\ \hat{\mathbf{x}}_2,\ ...,\ \hat{\mathbf{x}}_T\}$ given the candidate sequence of control inputs $\{\mathbf{u}_k\}_{k=0}^{T-1} = \{\mathbf{u}_0,\ \mathbf{u}_1,\ ...,\ \mathbf{u}_{T-1}\}$ and an initial state $\mathbf{x}_0$:

$$\begin{aligned} \{\mathbf{u}_t\}_{t=0}^{T-1,\,*} &= \underset{\{\mathbf{u}_t\}_{t=0}^{T-1}}{\operatorname{argmax}}\ a_{local}\left(\mathbf{x}_0,\ \{\mathbf{u}_t\}_{t=0}^{T-1}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right) \\ s.t. \quad \hat{\mathbf{x}}_{t+1} &= \hat{\mathbf{f}}(\hat{\mathbf{x}}_t, \mathbf{u}_t; \boldsymbol{\theta}) \end{aligned} \tag{7.4}$$

where $\hat{\mathbf{x}}_0 = \mathbf{x}_0$. The standard strategy in Model Predictive Control (MPC) formulation is to only apply the first control input in the sequence and then solve the OCP again for each consecutive timestep until the end of the horizon. This scheme requires $T-1$ optimizations to obtain one sequence of control inputs, and is therefore computational expensive. An alternative that is computationally feasible is to optimize for the entire control sequence one time and apply the control sequence

obtained from that one solution of the OCP. The authors of [74] developed an active learning scheme for a GP model. They suggested to maximize the sum of differential entropy of the GP model predictions over the control horizon of T steps, such that $a_{local} = \sum_{i=0}^{T-1} H[\hat{\mathbf{f}}(\hat{\mathbf{x}}_i, \mathbf{u}_i; \boldsymbol{\theta})]$. The differential entropy of variable y is defined by [139]

$$H(y) = -\int_{\mathcal{Y}} p(y) log(p(y)) dy, \tag{7.5}$$

where $p(y)$ is the probability density function. In this case, the probability density function represent the distribution over the predictions. If the variable y is Gaussian distributed, the differential entropy is given by

$$H_{Gaussian}(y) = \frac{1}{2} log(2\pi exp(\sigma^2(y)), \tag{7.6}$$

where σ^2 is the variance of the given prediction.

Global exploration

Exciting the system dynamics is essential to obtain informative data from a dynamical system. The local exploration formulated as an OCP in equation (7.4) provides a sound basis for exciting the problem locally. However, when the goal is to obtain the most informative data from the entire input space, solely depending on the optimization in equation (7.4) is inefficient. That is, the computational complexity increases drastically with optimization horizon. This put restrictions on how long the optimization horizon can be, and therefore also the area that a optimized state-action trajectory can span. Moreover, the uncertainty of state forecasts typically increases with each time step. This is highly relevant in the local exploration formulation since the corresponding OCP typically aims to maximize some uncertainty measure. With high levels of uncertainty, the actual states are likely to deviate from predicted states after longer horizons. Thus, the efficacy of local explorations as defined above is typically limited to exploring dynamics locally. Authors in [75] suggest partitioning the search problem into global and local explorations for actively learning a GP model. Building upon the work in [75], equation (7.7) provides a general formulation of the DeepAL optimization problem for dynamical systems, acquiring an optimal batch rather than single initial states. The global exploration consider a set $\mathcal{X} = \{\mathbf{x}_{0,1}, \mathbf{x}_{0,2}, \ldots, \mathbf{x}_{0,c}\}$ of c candidate initial states. For each of the candidate initial states $\mathbf{x}_{0,i}$, an optimal control trajectory $(\mathbf{u}_{\mathbf{t}_{t=0}}^{T-1,*})_i$ is obtained by optimizing the OCP in equation (7.4). With an initial condition $\mathbf{x}_{0,i}$ and the obtained control trajectory $(\mathbf{u}_{\mathbf{t}_{t=0}}^{T-1,*})_i$, the corresponding forecasted state trajectory $\left(\{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right)_i$ is estimated by the model. One acquisition step of the global exploration is generally described in the follow-

ing DeepAL optimization formulation:

$$\begin{aligned} \mathcal{B}^* = \underset{\mathcal{B}\subseteq\mathcal{X}}{argmax}\ a_{global}\left(\mathcal{B},\ \left\{\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right\}_{i=1}^{b}\right) \\ \left\{\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right\}_{i=1}^{b} \subseteq \mathcal{XU} \end{aligned} \tag{7.7}$$

where $\mathcal{B} = \{\mathbf{x}_{0,\,1},\ \mathbf{x}_{0,\,2}, ...,\ \mathbf{x}_{0,\,b}\}$ is a candidate batch of initial conditions, and $\left\{\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right\}_{i=1}^{b}$ is the corresponding batch of simulated state-action trajectories.
$\mathcal{XU} = \left\{\left\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right\}_1, \left\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right\}_2, ..., \left\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right\}_c\right\}$ is the set of simulated candidate state-action trajectories. The acquired a batch $\mathcal{B}^* = \{\mathbf{x}^*_{0,\,1},\ \mathbf{x}^*_{0,\,2},\ ...,\ \mathbf{x}^*_{0,\,b}\}$ of initial conditions corresponds to the batch simulated state-action trajectories $\left(\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right)_{i=1}^{b}$ that maximize some global batch acquisition function a_{global}. Since simulated state-action trajectories are already sampled from in a local exploration scheme, the global exploration becomes a static acquisition problem on the form of the standard DeepAL scheme presented in equation (2.32).

7.4 Method and setup

In this section, the experimental setup, as well as the methods used in the case study are presented. The data is generated by integrating the nonlinear ODEs in equation (2.21) with a set of initial values for the states $\mathbf{x}_0$ using the fourth-order Runge-Kutta (RK4) numerical integration algorithm. In the DeepAL method presented in this work, a batch of initial states are chosen from a set of candidate states according to the optimization in equation (7.7). The query strategies defined by the global acquisition function a_{global} are described in Section 7.4.1. The control trajectories that excite the system dynamics are acquired in the local exploration scheme defined in equation (7.4). The local acquisition function a_{local} in this scheme is an *uncertainty-based strategy* also described in detail in Section 7.4.1. The benchmark method chooses the set of initial conditions randomly. Moreover, the control input trajectory from each initial state are chosen according to the APRBS used to identify nonlinear dynamics with NNs in works like [117], [81] and [79]. In each loop of the AL scheme, a batch of $b = 10$ time series $\{\mathbf{X}_1,\ \mathbf{X}_2,\ ...,\ \mathbf{X}_i,\ ...,\ \mathbf{X}_b\}$ are obtained. The $i'th$ time series is obtained by simulating the dynamics from an initial condition $\mathbf{x}_0 = [x_1(0),\ x_2(0),\ x_3(0)]$ over a

horizon of $T = 15$ with timesteps $\Delta T = 0.1sec$. This yields a time series $\mathbf{X}_i$:

$$\mathbf{X}_i = \begin{bmatrix} x_1(0) & x_2(0) & x_3(0) & u_1(0) & u_2(0) & u_3(0) \\ x_1(1) & x_2(1) & x_3(1) & u_1(1) & u_2(1) & u_3(1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(t) & x_2(t) & x_3(t) & u_1(t) & u_2(t) & u_3(t) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1(T-1) & x_2(T-1) & x_3(T-1) & u_1(T-1) & u_2(T-1) & u_3(T-1) \\ x_1(T) & x_2(T) & x_3(T) & nan & nan & nan \end{bmatrix} \tag{7.8}$$

Hence, the control inputs $\mathbf{u}(t)$ are defined until timestep $t = T - 1$. That is, at the last step there is no need for a control input since there is no next state to be calculated. For each time series i in the batch, the output label $\mathbf{Y}_i$ for training is calculated by the forward Euler formula:

$$\mathbf{Y}_i = \begin{bmatrix} \frac{x_1(1)-x_1(0)}{\Delta T} & \frac{x_2(1)-x_2(0)}{\Delta T} & \frac{x_3(1)-x_3(0)}{\Delta T} \\ \frac{x_1(2)-x_1(1)}{\Delta T} & \frac{x_2(2)-x_2(1)}{\Delta T} & \frac{x_3(2)-x_3(1)}{\Delta T} \\ \vdots & \vdots & \vdots \\ \frac{x_1(T)-x_1(T-1)}{\Delta T} & \frac{x_2(T)-x_2(T-1)}{\Delta T} & \frac{x_3(T)-x_3(T-1)}{\Delta T} \end{bmatrix} \tag{7.9}$$

We define a new matrix $\mathbf{X}_i^{'}$ that contains all but the last row of the $i'th$ time series $\mathbf{X}_i$. Then $\mathbf{X}_i^{'}$ and $\mathbf{Y}_i$ are paired as inputs and outputs:

$$\mathcal{S}_i = [\mathbf{X}_i^{'},\ \mathbf{Y}_i]. \tag{7.10}$$

This is done for all simulations in the batch. Then the input-output pairs are stacked:

$$\mathcal{S}_{batch} = [\mathcal{S}_1^T,\ \mathcal{S}_2^T,\ ...,\ \mathcal{S}_b^T]^T, \tag{7.11}$$

before added to the training data $\mathcal{D}_{train}$. Before the training is conducted, the inputs and outputs in the training set $\mathcal{D}_{train}$ are normalized, shuffled, and put in mini-batches for training. In each loop of the learning scheme, an ensemble of $M = 10$ NNs are trained on all training data sampled up until that time.

7.4.1 Novel DeepAL scheeme for dynamical systems

The DeepAL acquisition scheme comprises a global exploration scheme and a local exploration scheme. The global exploration scheme will for each acquisition step in the AL loop choose a batch of b initial states $\mathcal{B}^* = \{\mathbf{x}_{0,\,1}^*,\ \mathbf{x}_{0,\,2}^*,\ ...,\ \mathbf{x}_{0,\,b}^*\}$ among a set of c candidates $\mathcal{X} = \{\mathbf{x}_{0,\,1},\ \mathbf{x}_{0,\,2},\ ...,\ \mathbf{x}_{0,\,c}\}$ according to the AL optimization problem defined in equation (7.7). For each of the candidate initial

states $\mathbf{x}_{0,\,i}$ a state-action trajectory is obtained according to the local exploration in equation (7.4). The query strategy is defined by the global batch acquisition function a_{global}, which quantifies the informativeness of batches of state-action trajectories corresponding to initial state candidates. A simple *uncertainty*-based acquisition function that sum of predictive entropies along all candidate trajectories is given by:

$$a_{global}\left(\mathcal{B}, \left(\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right)_{i=1}^{b}\right) = \sum_{i=1}^{b}\sum_{t=1}^{T} H([^{j}\hat{\mathbf{x}}_t]_{j=1}^{M}), \tag{7.12}$$

where $[^{j}\hat{\mathbf{x}}_t]_{j=1}^{M}$ is the set of ensemble forecasts at timestep t. Assuming that the ensemble predictions are approximately normally distributed around the mean prediction $\boldsymbol{\mu}_t$ given in equation (7.2), and that predicted states are uncorrelated, maximizing the entropy will become approximately the same as maximizing the empirical variance given in equation (7.3):

$$H([^{j}\hat{\mathbf{x}}_t]_{j=1}^{M}) \approx \boldsymbol{\sigma}_t^2. \tag{7.13}$$

In order to scale the optimization problem according to the magnitude of states in the state vector, the empirical variance of state k, $\sigma_{t,\,k}$, $k \in \{1,\ 2,\ 3\}$ can be divided by the standard deviation of the $k'th$ state based on the currently sampled dataset. defining the vector $\mathbf{s}_{inv} = [\frac{1}{std_1},\ \frac{1}{std_2},\ \frac{1}{std_3}]^T$, where std_k is the standard deviation of the $k'th$ state x_k, the resulting acquisition function can be defined as:

$$a_{global}\left(\mathcal{B}, \left(\{\mathbf{u}_t\}_{t=0}^{T-1,\,*}, \{\hat{\mathbf{x}}_t\}_{t=1}^{T}\right)_{i=1}^{b}\right) = \sum_{i=1}^{b}\sum_{t=1}^{T} \boldsymbol{\sigma}_t^2 \odot \mathbf{s}_{inv}, \tag{7.14}$$

where $\odot$ is the Hadamard product operator that takes the element-wise multiplication of the two vectors. The resulting acquisition function is purely uncertainty based and does not take into account the similarity between samples. Hybrid acquisition strategies takes into account both uncertainty of individual samples as well as the similarities between samples in a candidate batch $\mathcal{B}$. An intuitive hybrid acquisition method DMBAL adds informativeness to the optimization of a weighted K-means algorithm. That is, the algorithm acquires the closest sample to each of the b centroids found by a weighted K-means, where the weight is some informative measure. A simple adaption of the algorithm to the problem at hand is given in Algorithm 4: The method reuses the uncertainty measure defined in equation (7.14). In addition, the method aims to add diversity of samples by comparing the similarities of the candidate initial conditions in the modified DMBAL method. The c candidates in $\mathcal{X} = \{\mathbf{x}_{0,\,1}, \mathbf{x}_{0,\,2},\ ..., \mathbf{x}_{0,c}\}$ are at each acquisition step uniformly sampled from the intervals given in Table 7.1:

Algorithm 4: Diverse Mini-Batch Active Learning (DMBAL) in Global exploration

Input: Candidate initial conditions $\mathcal{X}$, Acquired dataset $\mathcal{D}_{train}$, pre-filter factor β, batch size/number of clusters b, Required level of model accuracy α

Train model on $\mathcal{D}_{train}$

while *required level of model accuracy* α *is not reached* **do**

- Get informativeness $\sum_{t=1}^{T} \boldsymbol{\sigma}_t^2 \odot \mathbf{s}_{inv}$ for simulated state-action trajectories corresponding to initial states in $\mathcal{X}$
- Prefilter top $\beta \cdot b$ informative samples
- Cluster $\beta \cdot b$ initial states to b clusters with weighted K-means
- Select batch $\mathcal{B}^*$ of b different initial states closest to the cluster centers
- Perform local exploration to obtain control input trajectories for each initial state in $\mathcal{B}^*$
- From initial conditions in $\mathcal{B}^*$, apply obtained control trajectories on system dynamics
- Add sampled data to training set $\mathcal{D}_{train}$
- Train model on all samples in $\mathcal{D}_{train}$

Table 7.1: Initial condition intervals for the states $\mathbf{x}$

Variable	Initial condition interval
x_1	$[-0.2,\ 1.4]$
x_2	$[-0.2,\ 0.2]$
x_3	$[-0.2,\ 0.2]$

The local exploration scheme obtains a sequence of control inputs by optimizing the dynamic acquisition problem formulated as an OCP in equation (7.4) from a given initial state. The local acquisition function a_{local} used in the optimization defined in equation (7.4) is the same as the uncertainty-based global acquisition function defined in equation(7.14), but only for a single initial state and the corresponding trajectory. That is, the local acquisition function is:

$$a_{local} = \sum_{t=1}^{T} \boldsymbol{\sigma}_t^2 \odot \mathbf{s}_{inv}. \tag{7.15}$$

A schematic illustration of the novel DeepAL scheme is presented in Fig. 7.2

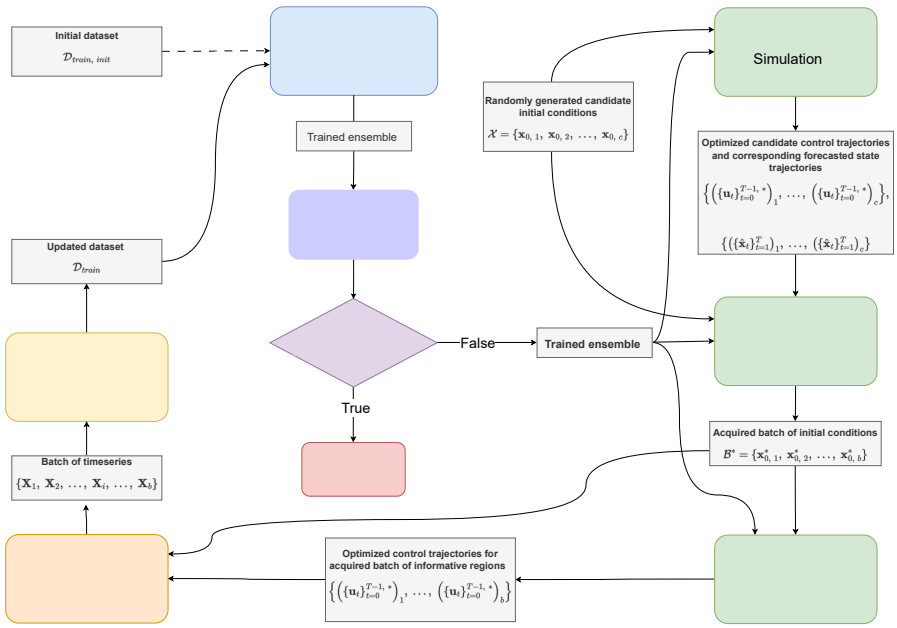


Figure 7.2: Schematic presentation of the DeepAL scheme. Given an initial dataset $\mathcal{D}_{init}$, NNs in an ensemble are trained. If the ensemble yields the required level of model accuracy, the AL scheme is terminated. If not, the ensemble is used in global and local explorations. First, local exploration and simulation procedure generates control trajectories for each initial state candidate in $\mathcal{X}$. The global exploration scheme then acquires a batch $\mathcal{B}^*$ of b initial states in which excitation of the dynamics would be most informative. Then, local explorations are conducted for each of the initial states in $\mathcal{B}^*$, yielding a set of b control trajectories. Given that the dynamics are driven to each of the initial conditions in $\mathcal{B}^*$, the corresponding control trajectories are applied to excite the dynamics. From this excitation process, time series are obtained, preprocessed and added to the training set $\mathcal{D}_{train}$. The ensemble is then trained on this training set, and the procedure is repeated until the required model accuracy on the test set is achieved, or a sampling budget is exhausted.

The OCP solved to generate control sequence is solved in the optimization framework CasADi [140], while the design and training of DNNs used in the optimization is done using the DL framework PyTorch [141]. The ML-CasADi package developed by authors of [142] is used to integrate the two frameworks.

7.4.2 Test set generation

The utilization of DNNs in modeling dynamical systems is driven by their capability to represent intricate relationships with a high degree of accuracy. When proper measures are taken to address safety considerations, they have the potential to enhance the optimality of MPC. As a result, evaluating the sampling strategies on a testset generated by using an optimal control policy is a subject of significant interest. The MPC used when generating the testset solves an OCP that minimize a quadratic cost function:

$$\begin{aligned} \{\mathbf{u}_0,\ ...,\ \mathbf{u}_{n-1}\} &= \min_{\mathbf{u}} \sum_{k=0}^{n-1} (\mathbf{x}_k - \mathbf{x}_{ref,\,k})^T \mathbf{Q} (\mathbf{x}_k - \mathbf{x}_{ref,\,k}) + \mathbf{u}_k \mathbf{R} \mathbf{u}_k, \\ \text{s.t.} \quad \dot{\mathbf{x}}_k &= \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k). \end{aligned} \tag{7.16}$$

$\mathbf{x}_{ref,\,k}$ is the desired reference signal, and $\mathbf{Q}$ and $\mathbf{R}$ are weighting matrices. The subscript k indicates the value of the variable at timestep k. The function $\mathbf{f}(\cdot, \cdot)$ is the simulation model itself. Given an initial condition, the optimization problem in equation (7.16) is solved for n steps. Both the sequence of states $\{\mathbf{x}_k\}_{k=0}^{n-1}$ and control inputs $\{\mathbf{u}_k\}_{k=0}^{n-1}$ are decision variables in the optimization, and can be extracted from the solution of the optimization. The testset consist of 100 time series with initial conditions uniformly sampled from the intervals in Table 7.1.

Table 7.2: Reference intervals for the states $\mathbf{x}_{ref}$

Variable	Reference values interval
x_1	$[-0.3,\ 1.3]$
x_2	$[-0.3,\ 0.3]$
x_3	$[-0.3,\ 0.3]$

Each of the time series are generated by ten optimizations of equation (7.16) each with a horizon of 50 timesteps. The final state $\mathbf{x}_n$ in one optimization is then the initial state of the next time series, such that the $i'th$ time series in the testset can be written as:

$$\mathbf{X}_{test,\,i} = \begin{bmatrix} x_1(0) & \dots & u_3(0) \\ \vdots & \ddots & \vdots \\ x_1(n-1) & \dots & u_3(n-1) \\ x_1(n) & \dots & u_3(n) \\ \vdots & \ddots & \vdots \\ x_1(10*n-1) & \dots & u_3(10*n-1) \end{bmatrix}, \tag{7.17}$$

where $n = 50$. The value of the references $\mathbf{x}_{ref}$ are uniformly drawn from Table 7.2 are constant for the optimization horizon. Hence each time series has ten different references over 500 timesteps. Hence, the testset can be written as:

$$\mathcal{D}_{test} = \{X_{test,\,1},\ X_{test,\,2},\ \ldots,\ X_{test,\,100}\} \tag{7.18}$$

7.5 Results and discussion

The case study presented in this section investigate the efficacy of global and local explorations compared to benchmark random sampling methods. Moreover, the study presents the effect of the global hybrid strategy DMBAL for different values of the prefilter hyperparameter β, where the special case of $\beta = 1$ can be considered as an uncertainty based acquisition strategy.

7.5.1 Information based and random sampling

In order to quantify the efficacy of global and local explorations in the proposed DeepAL sampling scheme compared to benchmark random sampling methods, we define three fundamental data acquisition schemes. All schemes use both a global and a local sampling method. The three fundamental methods are based on either an information theoretic approach or a random sampling strategy for both local and global exploration. Fig. 7.3 shows a schematic presentation of how the different schemes are derived from the two sampling strategies.

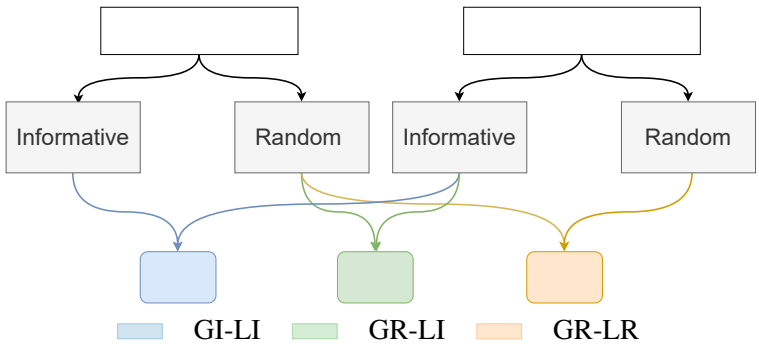
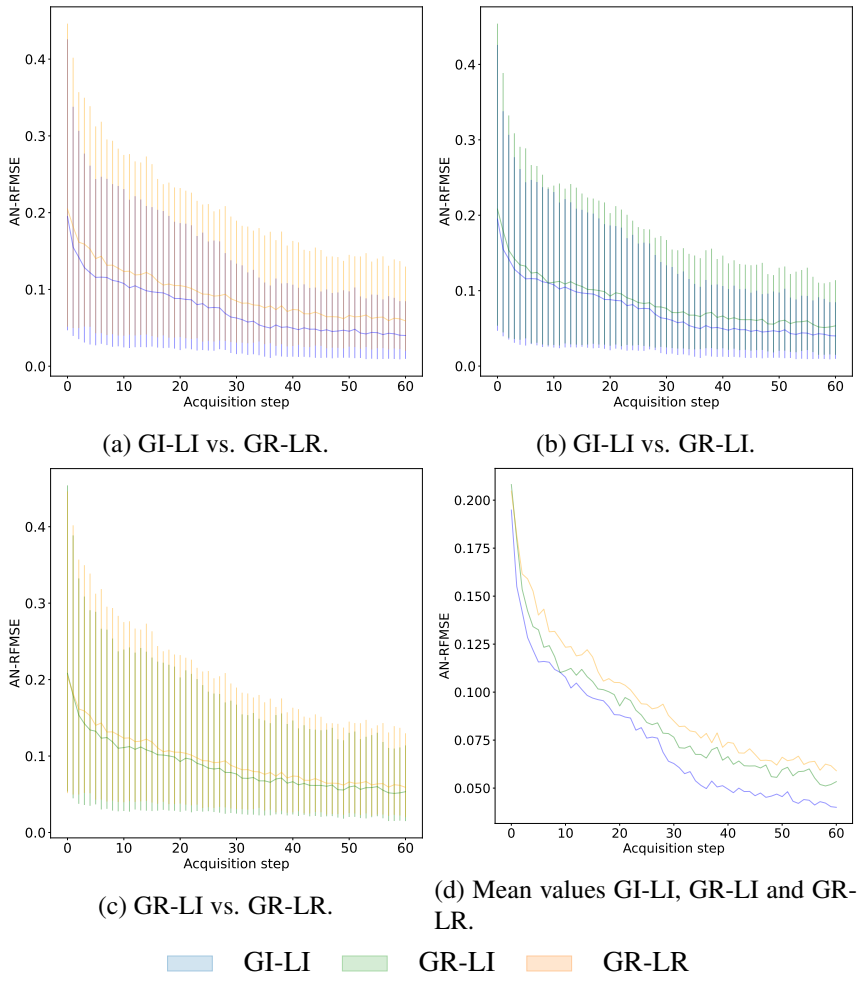


Figure 7.3: Schematic of how fundamental sampling schemes are derived from global and local sampling strategy. Global Informative Local Informative (GI-LI), Global Random Local Informative (GR-LI), and Global Random Local Random (GR-LR) are the derived methods that are investigated in the case study.

The fundamental sampling schemes are namely Global Informative Local Informative (GI-LI), Global Random Local Informative (GR-LI), and Global Random

Local Random (GR-LR). The GI-LI method is described in Section 7.4.1, and is using information theoretic sampling strategies both locally and globally. The global exploration in GI-LI acquires initial conditions from where to conduct the local explorations. The global exploration is using the local exploration method to assign measures of informativeness to the candidate initial conditions. The GR-LI method is globally random, and a batch of initial states is acquired by uniformly sampling within the interval of states presented in Table 7.1. The local exploration method of GR-LI is the exact same as the local sampling method used GI-LI. The GR-LR method uses the same random global strategy as GR-LI, and uses the APRBS method to excite the dynamics locally.



(a) GI-LI vs. GR-LR.

(b) GI-LI vs. GR-LI.

(c) GR-LI vs. GR-LR.

(d) Mean values GI-LI, GR-LI and GR-LR.

Figure 7.4: AN-RFMSE values for NN models trained on each batch of sampled data. The rolling forecast has a prediction horizon of 50 timesteps. The drawn line show the mean AN-RFMSE values, and the error bound show the 25 prosentile and 75 procentile of AN-RFMSE values for models trained on data sampled up until the AL loop specified on the x-axis. The GI-LI methods yields significantly lower mean and 75 procentile of AN-RFMSE values than the two competing methods. The GR-LI method slightly outperforms the GR-LR method, but this result is not as significant.

Fig. 7.4 presents the performance of the three sampling schemes after all data acquisition steps. The error bounds show the $25 - 75$ percentile of AN-RFMSE

values. The upper bound is particularly interesting since it gives an intuition about the models ability to generalize to a broader set of the test set trajectories. The results show that the GI-LI method outperforms the GR-LI and the GR-LR methods, both in terms of higher mean accuracy as well as significantly lower values for the 75 percentile of AN-RFMSE values. Moreover, the globally random, locally informative GR-LI method shows better performance in terms of higher mean accuracy an lower 75 percentile AN-RFMSE values than the purely random $GR - LR$ method. However, the superiority of GR-LI method over the GR-LR method is not as significant as the superiority of GI-LI over the two others, indicating that the globally informative step is of major importance.

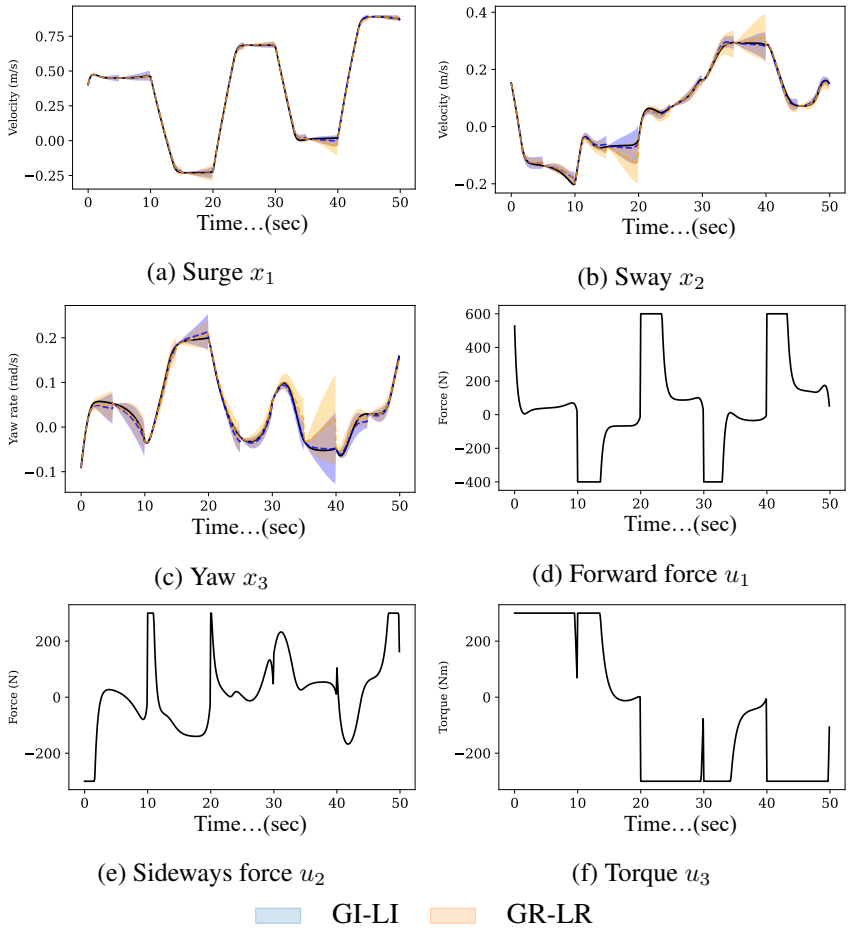
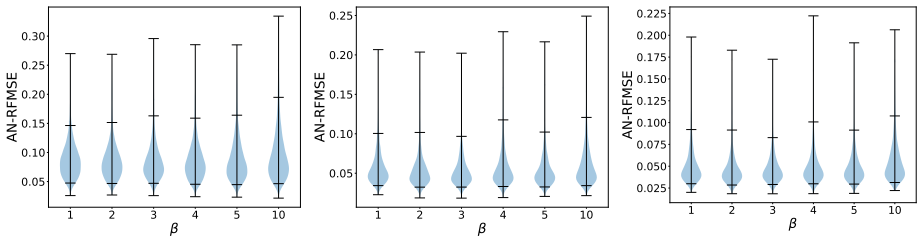


Figure 7.5: Rolling forecast of 50 timesteps. The black drawn lines in Fig. 7.5a-7.5a are state simulated states. The dotted lines show mean forecast values, and the uncertainty bounds show 99.7% confidence intervals. The black lines in Fig. 7.5d-7.5f are control inputs. The ensembles that forecast the states are trained on data based on the GI-LI and GR-LR methods. The plot illustrates how the GI-LI method gives better mean predictions as well as narrower and better calibrated uncertainty bounds.

Fig. 7.5 shows the mean and uncertainty bounds of ensemble forecasts trained on data sampled with GI-LI and GR-LR methods. The plots illustrate how the GI-LI might provide data that gives improved mean predictions as well as narrower and better calibrated uncertainty bounds.

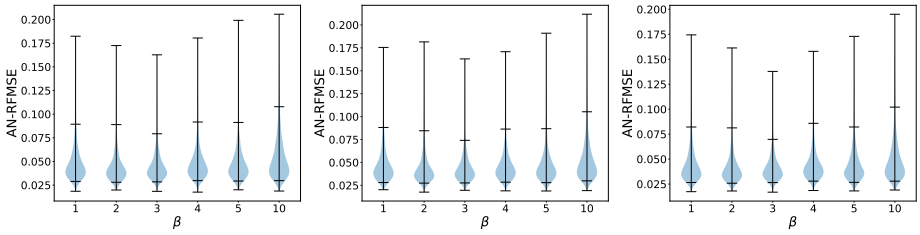
7.5.2 Uncertainty based and hybrid global strategy



(a) AN-RFMSE values for models trained between AL loop number ten and 30.

(b) AN-RFMSE values for models trained between AL loop number 30 and 40.

(c) AN-RFMSE values for models trained between AL loop number 40 and 45.



(d) AN-RFMSE values for models trained between AL loop number 45 and 50.

(e) AN-RFMSE values for models trained between AL loop number 50 and 55.

(f) AN-RFMSE values for models trained between AL loop number 55 and 60.

Figure 7.6: Violin plots of AN-RFMSE values corresponding to models trained on data sampled with the GI-LI method. The rolling forecast has a prediction horizon of 50 timesteps. All methods use the global DMBAL method for choosing initial conditions, but with different prefilter hyperparameter β. The two innermost horizontal lines of each violin plot show the 5 and 95 procentile, while the outermost horizontal lines show the extreme values. Each subplot includes AN-RFMSE values for models trained on data from a range of the AL loops. That is, Fig. 7.6a summarize the perfomance of the sampling methods in early stages, while Fig. 7.6c-7.6f summarize the performace later in the AL scheme.

Fig. 7.6 show the performance of GI-LI methods using the global DMBAL method with different prefilter parameter β for different stages of the AL scheme. The algorithm only considers the $b \cdot \beta$ samples with highest informativeness score according to a given informativeness measure, where a batch $\mathcal{B}^*$ of $b = 10$ initial conditions are chosen. Hence, $\beta = 1$ corresponds to simply picking the b samples with the highest informativeness score. Since the chosen informative measure is uncertainty based, using $\beta = 1$ means that the global method is uncertainty based. Using $\beta > 1$ means that the global algorithm takes into account diversity by choos-

ing the samples closest to centroid centers of a k-means algorithm, which means that the overall method is a hybrid AL strategy, combining uncertainty measures and diversity measures. Fig 7.6 shows that the average performance in all plots is approximately the same for all choices of β. In Fig. 7.6a and 7.6a showing performance for models trained on data acquired after respectively 10 to 30 AL loops and 30 to 40 AL loops, it is difficult to conclude any significant differences, other than that the method using $\beta = 10$ has higher extremum AN-RFMSE values, as well as more stretched distribution towards higher AN-RFMSE values, indicating that lower choices of β give better results. Fig 7.6c-7.6f show the performance of the method for different β values for later stages of the AL scheme. The most evident result is the tendency that the upper extreme values of AN-RFMSE have a minimum for $\beta = 3$, and that both lower and higher values of β give higher extreme AN-RFMSE values. Moreover, it seems like the 95 percentile also is the lowest for $\beta = 3$. Moreover, again, using high values of β turns out to give worse result as $\beta = 10$ show the worst performance. The overall results from the conducted case study indicate that using the DMBAL algorithm in the global acquisition scheme is of minor importance compared to acquiring a batch of samples with the highest uncertainty scores. However, choosing the right values of the prefilter hyperparameter β, the modeling errors in extreme cases can potentially be reduced. The limited value of the hybrid method used in this study might be explained by that it only compares the similarity of initial states rather than similarities between candidate trajectories. Moreover, the shortcomings of uncertainty based methods are known to be more present if the batch size is greater. With the currently choice of batch size $b = 10$ trajectories being sampled at each acquisition step, the power of hybrid strategies might not be present.

7.6 Conclusions and future work

The main conclusions from the work can be itemized as follows:

- The globally random, locally informative based GR-LI strategy show slightly better results than a globally random, locally random strategy in terms of mean accuracy an lower values of the 75 procentile of AN-RFMSE values, indicating better generalization. However, the novel GI-LI DeepAL scheme significantly outperform GR-LI and GR-LR schemes both in terms of mean accuracy and 75 procentile values of AN-RFMSE. This indicates that global explorations are of major importance with respect to achieving higher accuracy and generalization.

- The DMBAL approach, which emphasizes diversity in the selection of samples, might reduce the upper bound of extreme error values, provided that the

prefilter hyperparameter is carefully chosen. This method is compared to simply selecting the top b samples based on some uncertainty measure, when sampling a batch of initial conditions globally. However, the DMBAL approach does not exhibit significant improvements over the global uncertainty based method beyond this in the given case study. The method only compares the similarity between initial states of the candidate trajectories, rather than similarity between the whole candidate trajectories. This leaves out potentially important information.

The novel DeepAL framework is flexible and allows for a range of AL aqcuisition strategies. Conducting a comparative study including different AL acquisition functions in the framework would be highly interesting as it could increase our knowledge about efficient sampling of dynamical systems. Global hybrid strategies that can consider similarity of entire state trajectories are of particular interest since the currently tested hybrid strategy that only compares initial conditions seems to be of limited value.

Chapter 8

Conclusions and further work

This thesis presents a set of novel contributions, including data-driven and hybrid methodologies that address the challenges of using ML to model dynamical systems suffering from data scarcity. The contributions include utilizing system knowledge to reduce the complexity of the learning problem, assessing the effect of different model structures and the model complexity of NNs, and maximizing the information content in training data obtained from experimental design.

Chapter 3 presents a novel hybrid modeling approach. The method can estimate periodical and stationary unmodeled dynamics from a non-stationary measured signal sampled at low rates. We illustrate that the obtained high-fidelity signal estimate can be utilized in a state observer like the EKF to improve accuracy of the state estimates in the dynamical system. The linear nature of the compressed sensing signal model makes the method highly intepretable. However, while the method can cancel non-stationarities captured by the PBM, the method is sensitive to non-stationarities in the unmodeled dynamics. In complex processes like aluminum electrolysis, capturing all non-stationarities in a PBM is highly unlikely. Therefore, it is of scientific interest to investigate the possibility of using basis transforms that can sparsely represent non-stationary signals. If such basis transforms were found, a signal estimate could be estimated offline, such that high-fidelity data from a coarsely sampled signal could be obtained. This data could be used as training data for other ML models. However, depending on the compressed sensing model in real-time to provide a forecast of system dynamics for longer horizons means that we would rely on the signal to be constant in the transformed domain. This is a big assumption. However, if we can assume that the signal is slowly varying, the signal estimate could be used in state estimation for shorter horizons. Researching the possibility of extending the work to model slowly-varying

signals could also make the approach more attractive.

The remaining contributions in this thesis comprise DNNs that partly or completely model ODEs directly from input-output data. The hybrid modeling approach discussed above is inherently sensitive to unforeseen incidents or control actions deviating from the periodical pattern since the model does not account for these inputs. On the other hand, the DNNs included in the remaining models aim to learn a mapping from an input space containing all states and inputs affecting the dynamical system to the time derivative of the states. With access to control inputs at each time step, time integration of the models can provide state forecasts. These models aim to capture the underlying dynamics.

Chapter 4 introduces the hybrid modeling approach CoSTA approach to modeling aluminum electrolysis. That is, a corrective source term is added to the terms of a PBM, correcting for modeling errors in the PBM. We demonstrate that the method outperforms a purely data-driven approach with the same training data size. We argue that this can be explained by the reduced complexity of the modeling problem, compared to model all dynamics with a data-driven approach.

Pure data-driven modeling approaches was presented in Chapter 5 and 6. Chapter 5 examines the impact of sparsity promoting ℓ_1 regularization on the generalizability, interpretability, and training stability of DNNs compared to a densely connected NN. The findings of the study indicate that the use of ℓ_1 regularization leads to a significant reduction in model complexity, which enhances its interpretability. Additionally, the results demonstrate that the sparse neural network exhibits superior generalization performance, and achieves more stable convergence, especially in scenarios where the amount of training data is limited. The work in Chapter 6 builds on the previous work on sparse DNNs, and introduces skip-connections to model nonlinear dynamics. The combination of sparsity promoting regularization and skip-connections significantly improves model accuracy and training stability for a model trained on limited data. The case studies in both works were conducted on a set of high-dimensional ODEs representing the aluminum electrolysis.

The work on DNNs discussed above demonstrates that DNNs can obtain accurate models of a set of simulated high-dimensional ODEs representing the aluminum electrolysis. However, the case study assumes access to all state variables at each time step of the time series used as training data. Accessing measurements of state variables in the aluminum electrolysis is typically expensive, or even impossible due to the electrolysis cell's harsh environment. This can restrict the possibility of applying the models proposed to the aluminum electrolysis. In that sense, the work conducted might be of greater value to processes where all states affecting the dynamics are measurable. However, DNNs can possibly be used to model sub-

processes in the electrolysis, given that variables affecting the sub-process can be modeled.

Chapter 7 presents the last contribution in this thesis. The chapter presents a novel DeepAL framework for acquiring informative data used in identifying DNN models of dynamical systems. The acquisition framework is decomposed into a local and a global exploration. The local exploration explore different regions of the input space by obtaining control informative trajectories corresponding to each region, and then simulates state-action trajectories in these regions. This enables a novel static BMDAL acquisition formulation that evaluates the acquired set of candidate trajectories, and choose the set of regions that maximize some batch acquisition function. The dynamics of the chosen regions are then explored. The case study demonstrates that the novel sampling framework can outperform state-of-the-art random based sampling methods for system identification. The general formulation of the novel static BMDAL acquisition problem allows for a range of query strategies to be applied, potentially improving the performance of the method. However, it is inherent in searching for informative data in dynamical systems that the system will encounter regions of the input space where the model is uncertain. This can impose a safety risks which must be addressed if using the novel framework on real systems.

Appendix A

Simulation model

In this section we will follow a purely physics-based approach to deriving the equations. At appropriate places we will highlight the challenges and assumption

The dynamical system simulated in this study is generated by the set of ordinary differential equations (ODE's) in Eqs. (2.15a) - (2.15h). This system of equations is derived from a simplified model of an aluminum electrolysis cell. This model comprises simplified energy and mass balance of the electrolysis cell. The model considers an energy balance based on sideways heat transfer, energy transfer between the side ledge and bath due to the melting and freezing of cryolite, and energy input as a function of electrical resistance in the electrolyte and voltage drop due to bubbles. The mass balance includes mass transfer between side ledge and bath, the input of Al_2O_3 and AlF_3, production of metal and consumption of the raw material Al_2O_3, and tapping of metal from the electrolysis cell. In Table 2.1, the system states and inputs are described. The purpose of the simulation model is not to mimic the exact dynamics of an aluminum electrolysis cell but rather to generate nonlinear dynamics similar to what occurs in real aluminum electrolysis.

A.1 Heat capacity

Heat capacity is a measure of the amount of thermal energy a body of a certain material can store for a given temperature and volume and is given by the definition:

$$C = \frac{\delta q}{\delta T}. \tag{A.1}$$

$C[J/^\circ C]$ is the heat capacity, $\delta q[J]$ is an infinitesimal heat quantity and $\delta T[^\circ C]$. Specific heat capacity c_p is heat capacity at constant pressure per unit of mass:

$$c_p = \left(\frac{dh}{dT}\right)_p, \tag{A.2}$$

where c_p $[J/(kg^\circ C]$ is the specific heat capacity, h $[J/kg]$ is specific enthalpy and T $[^\circ C]$ is temperature. The subscript p indicates constant pressure. In the process of aluminum electrolysis, the pressure can be assumed to be constant at $p = 1$ $[atm]$.

A.2 Energy and mass balance

The first law of thermodynamics known as the energy conservation principle states the following [143]:

$$\frac{dE_i}{dt} = \dot{E}_{in,\,i} - \dot{E}_{out,\,i}. \tag{A.3}$$

$\frac{dE_i}{dt}$ $[W]$ is the change of energy of species i in the system, $\dot{E}_{in,\,i}$ $[W]$ is the energy input rate and $\dot{E}_{out,\,i}$ $[W]$ is the energy output rate of species i the system. System is here used synonymous to control volume. The energy of the system can be transferred through heat, work or through the energy associated with the mass crossing the system boundary. This can be expressed as follows:

$$\dot{E}_{in,\,i} = \dot{Q}_{in,\,i} + \dot{W}_{in,\,i} + \dot{m}_{in,\,i}e_{in,\,i} \tag{A.4}$$

$$\dot{E}_{out,\,i} = \dot{Q}_{out,\,i} + \dot{W}_{out,\,i} + \dot{m}_{out,\,i}e_{out,\,i} \tag{A.5}$$

where $\dot{Q}_{in,\,i}$ $[W]$ and $\dot{Q}_{out,\,i}$ $[W]$ are the rates of heat in and out of the system and $\dot{W}_{in,\,i}$ $[W]$ and $\dot{W}_{out,\,i}$ $[W]$ is the rate of work generated on the system. $\dot{m}_{in,\,i}$ $[kg/s]$ and $\dot{m}_{out,\,i}$ $[kg/s]$ is the mass rate into the system and out of the system respectively, whereas $e_{in,\,i}$ $[J/kg]$ and $e_{out,\,i}$ $[J/kg]$ is the specific energy of the mass entering and leaving the system. The specific energy can be formulated as:

$$e = u + \frac{1}{2}v^2 + gz, \tag{A.6}$$

where u $[J/kg]$ is the specific internal energy, $\frac{1}{2}v^2$ $[J/kg]$ is the specific energy related to velocity v $[m/s]$, and gz $[J/kg]$ is the specific energy related to elevation difference z $[m]$.

The change of system energy $\frac{dE_i}{dt} = \dot{E}_i$ can be written as:

$$\frac{dE_i}{dt} = \frac{d(m_i e_i)}{dt}, \tag{A.7}$$

where m $[kg]$ is the mass of the system and e_i $[J/kg]$ is the specific energy of the system. Since the relevant control volumes are related to an aluminum electrolysis, it is reasonable to neglect the terms $\frac{1}{2}v^2$ and gz. In this work, $\dot{Q} = Q_{in,\,i} - Q_{out,\,i}$ is defined as positive when net heat is provided to the system, and $W = W_{in,\,i} - W_{out,\,i}$ is positive when work is added to the system. Thus, the resulting energy equation can be formulated as:

$$\frac{d(m_i u_i)}{dt} = m\frac{du_i}{dt} + u_i\frac{dm_i}{dt} = \dot{m}_{in,\,i}u_{in,\,i} - \dot{m}_{out,\,i}u_{out,\,i} + \dot{Q} + \dot{W}. \quad \text{(A.8)}$$

Work W $[J]$ is organized transfer of energy. W can be divided into several types of work [144]:

$$W = W_{flow} + W_{\Delta V} + W_s + W_{el} + W_{other}. \quad \text{(A.9)}$$

W_{flow} is the work associated with the volume displacements of streams that enter and exit the system, $W_{\Delta V}$ is the work associated with changes of the volume of the system, W_s is the mechanical work supplied using movable machinery, W_{el} is the electrochemical work supplied when the system is connected to an external electric circuit. W_{other} is the sum of other types of work, for example if surface areas changes or electromagnetic work. For an aluminum electrolysis, $W_{\Delta V} = W_s = W_{other} \approx 0$. W_{flow} is given by:

$$W_{flow} = pV, \quad \text{(A.10)}$$

where p is pressure and V is volume. Enthalpy H $[J]$ is given by:

$$H = U + pV. \quad \text{(A.11)}$$

where $U = m \cdot u$ $[J]$. Furthermore, $H = h \cdot m$. Thus:

$$m\frac{du_i}{dt} + u_i\frac{dm_i}{dt} = \dot{m}_{in,\,i}h_{in,\,i} - \dot{m}_{out,\,i}h_{out,\,i} + \dot{Q} + \dot{W}_{el}. \quad \text{(A.12)}$$

Assuming that the flow work is neglectable compared to the other quantities in the energy equation for aluminum electrolysis gives that $H \approx U$. Recall that $c_p = \frac{dh}{dT}$. Hence:

$$\frac{du_i}{dt} \approx \frac{dh_i}{dt} = \frac{\partial h_i}{\partial T_i}\frac{dT_i}{dt} = c_{p_i}\frac{dT_i}{dt}, \quad \text{(A.13)}$$

where $\frac{dT}{dt} = \dot{T}_i$ $[^\circ C]$ is the temperature derivative with respect to time. This yields:

$$\frac{dT_i}{dt} = \frac{1}{m_i c_{p_i}}\left(\dot{m}_{in,\,i}h_{in,\,i} - \dot{m}_{out,\,i}h_{out,\,i} + \dot{Q} + \dot{W}_{el} - u_i\frac{dm_i}{dt}\right) \quad \text{(A.14)}$$

The mass rate equation can be formulated as:

$$\frac{dm_i}{dt} = \dot{m}_{in} - \dot{m}_{out} + \sum_{j=1}^{n_j} r_{i,j}, \tag{A.15}$$

where $r_{i,j}$ is the reaction rate of species i being produced or consumed in reaction j. Assuming that the contents of the control volume to be perfectly mixed and assuming that the flow work is neglectable gives:

$$u_i \approx h_i = h_{out,\,i}. \tag{A.16}$$

Hence, the resulting temperature specific energy equation for component i in a control volume is given by:

$$\frac{dT_i}{dt} = \frac{1}{m_i c_{p_i}} \left(\dot{m}_{in,i}(h_{in,i} - h_{out,i}) + \dot{Q} + \dot{W}_{el} - h_{out,\,i} \sum_{j=1}^{n_j} r_{i,\,j}. \right) \tag{A.17}$$

The latter equation states that the time derivative of the temperature in the control volume is dependent on the composition of species in the control volume. It is assumed that the temperature is equal for all components in the control volume. Furthermore, it is assumed that there is a common heat loss $\dot{Q}$ from a control volume to other control volumes, and that electrical power $\dot{W}_{el}$ is performed on the whole control volume instead of on different components in the control volume. When different components are mixed in a control volume, the enthalpy of this mix is more complex than adding the enthalpy of individual species. However, the complexity of mixed enthalpy is left out of this simulation model. The heat capacity of a mix of components in a control volume $c_{p_{cv}}$ is simplified to be constant despite of that $c_{p_{cv}}$ varies with composition and temperature in the control volume. The values for different species and control volumes are taken from [145] and [146]. Thus, the simplified simulation equation for the temperature derivative in a control volume is given by:

$$\begin{aligned} \frac{dT_{cv}}{dt} = \frac{1}{m_{cv} c_{p_{cv}}} \Bigg(& \left[\sum_{i=1}^{n_i} \dot{m}_{in,i}(h_{in,i} - h_{out,i}) \right] + \dot{Q} + \dot{W}_{el} \\ & - \sum_{i=1}^{n_i} \left[h_{out,\,i} \sum_{j=1}^{n_j} r_{i,\,j} \right] \Bigg) \end{aligned} \tag{A.18}$$

where m_{cp} is the sum of masses in the control volume.

A.3 Heat transfer

Heat transfer $\dot{Q}[W]$ will from this point be referred to as Q, meaning that the dot is omitted. the In the process of aluminum electrolysis, the two most important principles for heat transfer are convection and conduction. **Conduction** is heat transfer through molecular motion within a solid material. The expression for conduction is given by

$$Q = -k \cdot A \cdot \frac{\partial T}{\partial x}. \tag{A.19}$$

Q $[W]$ is heat transferred, A $[m^2]$ is the area the heat is transferred through $\frac{\partial T}{\partial x}$ $[^\circ C/m]$ is the temperature gradient in the direction x that the heat is transferred, and k $[W/(m^\circ C)]$ is the thermal conductivity, a material dependent proportionality constant. For a fixed cross-section area, the one dimensional **steady state heat flow** through a wall of thickness x $[m]$ from $x = 0$ with temperature T_1 to $x = 1$ with temperature T_1 integrates to:

$$Q = k \cdot A \cdot \frac{T_1 - T_2}{x}, \tag{A.20}$$

where $T_1 > T_2$. Thermal conductive resistance for a plane wall can be extracted from the latter expression:

$$R_{cond} = \frac{x}{k \cdot A}, \tag{A.21}$$

where $R_{cond}[^\circ C/W]$ is the thermal resistance, $x[m]$ is the thickness of the solid material in the direction heat is transferred, and k and A are as mentioned above. Thermal conductive analysis is analogous to an electrical circuit, where the temperature difference is analogous to the potential difference V, the heat flow is analogous to the electrical current I and thermal resistance is analogous to electrical resistance R_{el}. **Convection** is the heat transfer through the mass motion of a fluid. Heat transfer between a surface at temperature T_s and a fluid at a bulk temperature T_f is due to convection. Convection can be formulated as:

$$Q = h \cdot A \cdot (T_s - T_f), \tag{A.22}$$

where $A[m^2]$ is the contact surface between a solid surface and the liquid, the heat transfer coefficient h $[W/(m^2)^\circ C]$ is the proportionality constant between the heat flux and the thermodynamic driving force for the flow of heat, i.e. the temperature difference $(T_s - T_f)$.

Thermal resistance can be defined for a fluid R_{conv}, and is given by:

$$R_{conv} = \frac{1}{h \cdot A}. \tag{A.23}$$

As for electrical circuits, thermal resistances can be coupled in series, and the reciprocal of the total resistance equals the sum of reciprocals of individual resistances:

$$\frac{1}{R_{tot}} = \sum_{i=1}^{N} \frac{1}{R_i} \tag{A.24}$$

Eq. A.24 together with the assumption of stationary heat transfer makes it possible to calculate the heat transfer from one edge to the other through several resistors in series as the temperature difference between the edges divided by the sum of reciprocals of individual resistors:

$$Q = \frac{T_1 - T_{N+1}}{\sum_{i=1}^{N} R_i}, \tag{A.25}$$

where $T_1 > T_2 > ... > T_{N+1}$ is temperature and R_i are the resistors. In the simulation model, the heat transfer is assumed to be piecewise stationary, meaning that the heat transfer is assumed to be constant from the middle of one control volume to the middle of the adjacent control volume. However, the heat transfer is not assumed to be stationary through several control volumes. Thus, there are separate energy balances for each control volume. Heat transfer is only considered through the side walls of the electrolysis cell.

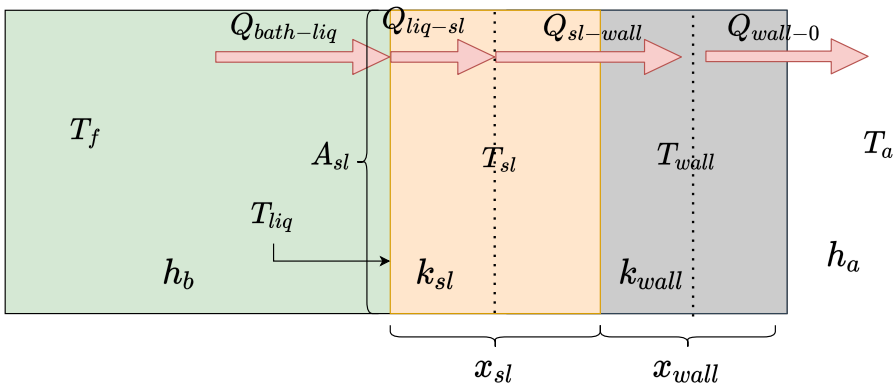


Figure A.1: Convection and conduction trough several materials

Convective heat transfer $Q_{bath-liq}$ from the bath to the surface of the side ledge

$$Q_{bath-liq} = h_{bath-sl} A_{sl} (T_{bath} - T_{liq}). \tag{A.26}$$

Conductive transfer Q_{liq-sl} from surface of side ledge to the center of the side

ledge

$$Q_{liq-sl} = \frac{2k_{sl}A_{sl}(T_{liq} - T_{sl})}{x_{sl}}. \tag{A.27}$$

Conductive heat transfer $Q_{sl-wall}$ from center of side ledge to the center of the side wall

$$Q_{sl-wall} = \frac{A_{sl}(T_{sl} - T_{wall})}{(x_{wall}/2k_{wall}) + (x_{sl}/2k_{sl})} \tag{A.28}$$

The heat transfer from the middle of the wall to the ambient Q_{wall-0} consists of conductive heat transfer from the middle of the wall to the surface of the wall, and the convection from the surface of the wall to the ambient air

$$Q_{wall-0} = \frac{A_{sl}(T_{wall} - T_0)}{(1/h_{wall-0}) + (x_{wall}/2k_{wall})}. \tag{A.29}$$

A.4 Electrochemical power

Electrochemical power $\dot{W}_{el}$ $[W]$, from now referred to as P_{el} is the amount of energy transferred to a system from a electrical circuit and is defined as:

$$P_{el} = U_{cell} \cdot I_{line}, \tag{A.30}$$

where U_{cell} $[V]$ is the applied cell voltage and I_{line} $[A]$ is the line current sent through the electrolyte. The cell voltage is composed of three different types of voltage contributions. these are the **decomposition voltage**, which is the theoretical minimum potential for the decomposition of alumina, **overvoltage**, meaning the excess voltage due to electrode polarization and **ohmic voltage drops**, due to resistance of various sections in the cell [147]. These contributions can be divided into smaller contributions caused by different effects in different parts of the cells. To make the mathematical expression in the resulting nonlinear simulation model less comprehensive, only **ohmic voltage drop** contributions are included. These are:

- Electrolyte voltage drop U_{el} $[V]$
- Bubble voltage drop U_{bub} $[V]$

The **voltage drop over the electrolyte** is due to the electrical resistivity of the electrolyte. Assuming uniform current density, the resistance of the elecrolyte is given by:

$$R_{el} = \frac{1}{\kappa}\frac{d}{A}. \tag{A.31}$$

R_{el} $[\Omega]$ is the electrical resistance, κ $[1/(\Omega m)]$ is electrical conductivity, d $[m]$ is the interpolar distance and A $[m^2]$ is the total surface of the anodes. The expression for electrical conductivity is given by [9]:

$$\kappa = exp\Bigg(2.0156 - \frac{2068.4}{T_{bath} + 273} + 0.4349 \cdot BR - 2.07C_{al_2O_3} - 0.5C_{CaF_2} - 1.66C_{MgF_2} + 1.78C_{LiF} + 0.77C_{Li_3AlF_6}\Bigg). \tag{A.32}$$

T_{bath} $[^\circ C]$ is the temperature of the electrolyte, BR $[-]$ is the bath ratio, while $C_x[-]$ is the concentration of substance x. BR is assumed to be constant at 1.2, $C_{MgF_2} = 0.01$, $C_{CaF_2} = 0.05$, $C_{LiF} = 0$ and $C_{Li_3AlF_6} = 0$. Thus, κ can be simplified to:

$$\kappa = exp\Bigg(2.496 - \frac{2068.4}{T_{bath} + 273} - 2.07C_{Al_2O_3}\Bigg). \tag{A.33}$$

The voltage drop due to resistance in the electrolyte is given by:

$$U_{EL} = R_{EL} \cdot I_{line} \tag{A.34}$$

Gas accumulation beneath the anode surface which reduces the cross-sectional area of the electrolyte in that zone. Thus, the effective resistivity increases and causes the so called **bubble voltage drop** U_{bub} [148]:

$$U_{bub} = \frac{d_{bub} \cdot j_A}{\kappa} \frac{\phi}{1 - \phi}. \tag{A.35}$$

j_A $[A/cm^2]$ is the anode current density d_{bub} $[cm]$ in the bubble layer thickness and ϕ $[-]$ is the bubble coverage as a fraction of the anode:

$$d_{bub} = \frac{0.5517 + j_A}{1 + 2.167j_A}, \tag{A.36}$$

and

$$\phi = 0.509 + 0.1823j_A - 0.1723j_A^2 + 0.05504j_A^3 + \frac{0.4322 - 0.3781BR}{1 - 1.637BR} + \frac{0.431 - 0.1437(x_{Al_2O3} - x^{AE}_{Al_2O3})}{1 + 7.353(x_{Al_2O3} - x^{AE}_{Al_2O3})} \tag{A.37}$$

$x_{Al_2O_3}$ $[-]$ is the weight percent of alumina in the bath and $x^{AE}_{Al_2O_3}$ $[-]$ is the weight percent of alumina at where the anode effect occurs, in this case it is assumed that $x^{AE}_{Al_2O_3} = 2.0$. Since the simulation model is simplified to only include contributions from U_{bub} and U_{EL}, the total applied cell voltage in the simulation model is given by:

$$U_{cell} = U_{EL} + U_{bub}. \tag{A.38}$$

A.5 Mass rates

The substances considered in the simulation model are alumina (Al_2O_3), aluminum fluoride (AlF_3) and cryolite (Na_3AlF_6) in the bath and liquid aluminum (Al) in the metal pad below the bath. Aluminum is extracted from alumina, which is dissolved in the electrolytic bath. In addition to alumina, carbon anodes are consumed in the net reaction, producing molten aluminum and carbon dioxide gas (CO_2):

$$2Al_2O_3 + 3C \rightarrow 3CO_2 + 4Al. \tag{A.39}$$

This reaction occurs at a rate according to Farraday's law for aluminum electrolysis [146]:

$$r_{al} = \frac{CE \cdot I_{line}}{z \cdot F}, \tag{A.40}$$

where r_{al} $[mol/s]$ is the reaction rate of the primary reaction of the Hall-Héroult process presented in Eq. A.39, $z = 12$ is the number of electrones in the reaction, $F = 96486.7[(A \cdot s)/mol]$ is the Faraday constant, and $CE[-]$ is the current efficiency, assumed constant at $CE = 0.95$. The fed alumina contains several impurities [146]. In the simulation model derived used in this work, only sodium oxide (Na_2O) is considered as additions in the feeded alumina. The content of Na_2O in alumina react as:

$$3Na_2O + 4AlF_3 \rightarrow 2Na_3AlF_6 + Al_2O_3. \tag{A.41}$$

3 Mol Na_2O reacts with 4 Mol of AlF_3 and produces 2 Mol of Na_3AlF_6 and 1 Mol of Al_2O_3. The reaction rate of the latter reaction r_{bath} $[kmol/s]$ can be formulated as:

$$r_{bath} = \frac{C_{Na_2O}}{3M_{Na_2O}} u_{Al_2O_3}, \tag{A.42}$$

where C_{Na_2O} $[-]$ is the weight percent of Na_2O in the alumina feed, M_{Na_2O} $[g/mol]$ is the molar mass of Na_2O and $u_{Al_2O_3}$ kg/s is the rate of alumina feed. The reaction in Eq. A.41 affects the mass balance of both AlF_3, Na_3AlF_6 and Al_2O_3. Therefore, r_{bath} is included in the mass balance equations of all these species. The general mass rate Eq. A.15 is used in the derivation the mass rate of side ledge, cryolite, alumina, aluminum fluoride and metal. The control volumes in which there are nonzero mass rates are the bath/electrolyte, the metal pad and the side ledge. The **mass rates in the electrolyte** are:

$$\dot{m}_{Al_2O_3} = (1 - C_{Na_2O})u_{Al_2O_3} - \frac{2}{1000} r_{al} M_{Al_2O_3} + r_{bath} M_{Al_2O_3}. \tag{A.43}$$

$\dot{m}_{Al_2O_3}$ $[kg/s]$ is the mass rate of alumina and $M_{Al_2O_3}$ $[g/mol]$ is the molar mass of alumina. $\frac{2}{1000} r_{al} M_{Al_2O_3}$ $[kg/s]$ is the reaction rate of alumina produced due to

the reaction in Eq. A.39 and $r_{bath}M_{Al_2O_3}$ $[kg/s]$ is the reaction rate of alumina due to the reaction in Eq. A.41. The mass rate of AlF_3 is given by:

$$\dot{m}_{AlF_3} = u_{AlF_3} - 4r_{bath}M_{AlF_3}, \tag{A.44}$$

where $\dot{m}_{AlF_3}$ $[kg/s]$ is the mass rate of aluminum fluoride, u_{AlF_3} $[kg/s]$ is the input rate of aluminum flouride and $4r_{bath}M_{AlF_3}$ $[kg/s]$ is the reaction rate of produced aluminum fluoride from the reaction in equation (A.41). The mass rate of cryolite in the bath is given by:

$$\dot{m}_{cry} = w_{fus} + 2r_{bath}M_{cry}. \tag{A.45}$$

$\dot{m}_{cry}$ $[kg/s]$ is the mass rate of cryolite in the electrolyte, $2r_{bath}M_{cry}$ $[kg/s]$ is the reaction rate of produced cryolite due to the reaction in Eq. A.41 and w_{fus} $[kg/s]$ is the mass rate of cryolite transferred between the side ledge and the bath. w_{fus} is given by:

$$w_{fus} = \frac{Q_{bath-liq} - Q_{liq-sl}}{\Delta_{fus}H_{cry}}. \tag{A.46}$$

$Q_{bath-liq}$ and Q_{liq-sl} is given in Eqs. A.26 and (A.27) respectively, and $\Delta_{fus}H_{cry}$ is the heat of fusion for cryolite, i.e. the amount of energy required to melt one kg of cryolite. The heat of fusion for cryolite at $1000°C$ is $\Delta_{fus}H_{cry} = 119495$ $[J/kg]$ [146], and is assumed to be constant in the simulation model. The **side ledge** is necessary to withstand the highly corrosive molten cryolite in the oven. The side ledge consists of *frozen* cryolite [146]. The mass rate of side ledge is therefore the transfer of cryolite between the electrolyte and side ledge due to melting and freezing:

$$\dot{m}_{sl} = -w_{fus}. \tag{A.47}$$

$\dot{m}_{sl}$ $[kg/s]$ is the mass rate of side ledge, and w_{fus} $[kg/s]$ is given above. The **mass rate of aluminum** is given by:

$$\dot{m}_{Al} = \frac{2}{1000}r_{al}M_{Al} - u_{tap}. \tag{A.48}$$

$\dot{m}_{Al}$ $[kg/s]$ is the mass rate of aluminum, $\frac{2}{1000}r_{al}M_{Al}$ $[kg/s]$ is the reaction rate of produced aluminum due to the reaction in equation (A.39), and u_{tap} $[kg/s]$ is the control input of tapping metal from the oven.

A.6 Temperature derivatives

Eq. (A.18) is used to calculate the temperature derivatives in the electrolyte, side ledge and side wall. As mentioned above, $\dot{Q} = Q$ and $\dot{W}_{el} = P_{el}$. In the **electrolyte**, the energy is transferred in and out of the control volume in many different

ways. Heat $Q_{bath-sl}$ is transferred through convection from the bath to the side ledge surface ($Q_{bath-liq}$) and from the side ledge surface to the center of the side ledge with conductive heat Q_{liq-sl}. The resulting heat transfer can be formulated as:

$$Q_{bath-sl} = \frac{T_{bath} - T_{sl}}{(x_{sl}/2k_{sl}A_{sl}) + 1/(h_{bath-sl}A_{sl})}. \tag{A.49}$$

Energy is transferred through mass transfer in several ways. The energy needed to heat and melt substances fed as control input u is given by:

$$\begin{aligned} E_u = \ & \Delta_{fus}H_{Al_2O_3}u_{Al_2O_3} + \Delta_{fus}H_{AlF_3}u_{AlF_3} \\ & + (T_{bath} - T_{in})(\bar{c}_{p_{Al_2O_3}}u_{Al_2O_3} + \bar{c}_{p_{AlF_3}}u_{AlF_3}). \end{aligned} \tag{A.50}$$

$\Delta_{fus}H_i$ $[J/kg]$ is the specific heat of fusion for substance i, and $\bar{c}_{p_i}$ $[J/(^\circ Ckg)]$ is the average heat capacity from T_{in} to T_{bath}. Energy is also transferred through mass transfer between the electrolyte and the side ledge. When the side ledge (frozen cryolite) melts into the bath, energy is required both to heat and melt the frozen cryolite. The energy required to heat the frozen cryolite is:

$$E_{tc,\,liq} = w_{fus}c_{p,cry,liq}(T_{bath} - T_{liq}). \tag{A.51}$$

w_{fus} $[kg/s]$ is as mentioned above the mass rate of cryolite between the bath and side ledge, and is positive when cryolite melts and is transferred to the bath. $c_{p,cry,liq}$ $[J/(^\circ Ckg)]$ is the heat capacity of molten cryolite T_{bath} $[^\circ C]$ is the bath temperature and T_{liq} $[^\circ C]$ is the liquidus temperature at which cryolite melts and freezes. $E_{tc,\,liq}$ $[W]$ is the energy required to heat molten cryolite from liquidus temperature to bath temperature. The subscript tc stands for temperature change and the subscript liq indicates that the substance is liquid. When $E_{tc,liq}$ is positive, it is because W_{fus} is positive, indicating that cryolite is melting. Thus, when $E_{tc,liq}$ is negative, cryolite is freezing. The energy needed to melt frozen cryolite is given by:

$$E_{sc} = w_{fus}\Delta_{fus}H_{cry}. \tag{A.52}$$

E_{sc} $[W]$ is the energy required to melt the mass of frozen electrolyte. The subscript sc stands for state change, meaning that it transitions between solid and liquid. $\Delta_{fus}H_{cry}$ $[J/kg]$ is the heat of fusion for cryolite as mentioned above. When E_{sc} is positive, cryolite is melted and energy is required whereas when E_{sc} is negative, cryolite freezes, and energy is released. Energy is required for the primary reaction in Eq. A.39 since it is endothermic:

$$E_{r_{al}} = r_{al}\Delta H_{r_{al}}. \tag{A.53}$$

$E_{r_{al}}$ $[W]$ is the amount of energy required for the reaction to take place, $r_{al}[mol/s]$ is as mentioned above the reaction rate of the primary reaction in the process,

and $\Delta H_{r_{al}} = 2197582\ [J/mol]$ is the enthalpy of reaction for (A.39). Since the reaction in equation (A.41) is exothermic, this reaction releases energy to its surroundings. This is given by:

$$E_{r_{bath}} = 1000 r_{bath} \Delta H_{r_{bath}}. \tag{A.54}$$

$E_{r_{bath}}\ [W]$ is the energy released due to the reaction in (A.41) $r_{bath}\ [kmol/s] = 1000[mol/s]$ is the reaction rate of (A.41) and $\Delta H_{r_{bath}} = -993283\ [J/mol]$ is the entalpy of reation for (A.41). The time derivative for the bath temperature is given by:

$$\dot{T}_{bath} = \frac{P_{el} - Q_{bath-sl} - E_u - E_{tc,\,liq} - E_{sc} - E_{r_{al}} - E_{r_{bath}}}{m_{bath} c_{p_{bath,\,liq}}}. \tag{A.55}$$

$m_{bath}\ [kg]$ is the mass of the liquid bath and $c_{p_{bath,\,liq}}\ [J/(^\circ C \cdot kg)]$ is the enthalpy of the liquid bath. For the **side ledge** control volume, the energy transfer is through melting and freezing of cryolite and heat transfer. The energy transfer related to melting and freezing of cryolite is given by:

$$E_{tc\ sol} = w_{fus} c_{p_{cry,\,sol}} (T_{liq} - T_{sl}). \tag{A.56}$$

$E_{tc\ sol}\ [W]$ is the energy required into the side ledge when frozen cryolite heats from side ledge temperature T_{sl} to liquidus temperature T_{liq}. $w_{fus}\ [kg/s]$ is given above and $c_{p_{cry,\,sol}}\ [J/(^\circ C \cdot kg)]$ is the heat capacity of solid cryolite. The time derivative of the side ledge temperature is given by:

$$\dot{T}_{sl} = \frac{Q_{liq-sl} - Q_{sl-wall} - E_{tc\ sol}}{m_{sl} c_{p_{cry,\,sol}}} \tag{A.57}$$

$\dot{T}_{sl}\ [^\circ C/s]$ is the temperature change in the side ledge, $Q_{liq-sl}\ [W]$ and $Q_{sl-wall}\ [W]$ is the heat in and out of the side ledge respectively. $m_{sl}\ [kg]$ is the mass of the side ledge, and $c_{p_{cry,\,sol}}\ [J/(^\circ C \cdot kg)]$ is the heat capacity of solid cryolite. There is no mass transfer through the **wall**. Therefore, the only energy transfer through this control volume is through heat transfer. The time derivative of the wall temperature $\dot{T}_{wall}\ [\circ C/s]$ is given by:

$$\dot{T}_{wall} = \frac{Q_{sl-wall} - Q_{wall-0}}{m_{wall} c_{p_{wall}}}. \tag{A.58}$$

$Q_{sl-wall}\ [W]$ is the heat from the side ledge to the wall, $Q_{wall-0}\ [W]$ is the heat from the wall to the ambient, $m_{wall}\ [kg]$ is the mass og the wall and $c_{p_{wall}}\ [J/(^\circ C \cdot kg)]$ is the heat capacity of the wall.

A.7 Liquidus temperature

In [149], the liquidus temperature T_{liq} was determined for primary crystalization of cryolite (Na_3AlF_6) in a system consisting of the bath components $Na_3AlF_6 - AlF_3 - LiF - CaF_2 - MgF_2 - KF$. The liquidus temperature was determined by thermal analysis in a vertical alumina tube furnace under argon atmosphere. An empirical cure was fitted, which is valid from temperatures $1011°C$ to approximately $800°C$. The curve is given by:

$$\begin{aligned} T_{liq} = \; & 1011 + 0.50[AlF_3] - 0.13[AlF_3]^{2.2} \\ & - \frac{3.45[CaF_2]}{1 + 0.0173[CaF_2]} \\ & + 0.124[CaF_2][AlF_3] - 0.00542\left([CaF_2][AlF_3]\right)^{1.5} \\ & - \frac{7.93[Al_2O_3]}{1 + 0.0936[Al_2O_3] - 0.0017[Al_2O_3]^2 - 0.0023[AlF_3][Al_2O_3]} \\ & - \frac{8.90[LiF]}{1 + 0.0047[LiF] + 0.0010[AlF_3]^2} \\ & - 3.95[MgF_2] - 3.95[KF]. \end{aligned} \tag{A.59}$$

$[x]$ denote the weight-% of component x. In the simulator, it is assumed that the following components are constant at values $[MgF_2] = 1\%$, $[CaF_2] = 5\%$, $[KF] = [LiF] = 0\%$. This yields:

$$\begin{aligned} T_{liq} = \; & 991.2 + 1.12[AlF_3] - 0.13[AlF_3]^{2.2} + 0.061[AlF_3]^{1.5} \\ & - \frac{7.93[Al_2O_3]}{1 + 0.0936[AlF_3] - 0.0017[AlF_3]^2 - 0.0023[AlF_3][Al_2O_3]}. \end{aligned} \tag{A.60}$$

A.8 Further simplifications in the simulation model

In addition to assumptions and simplifications accounted for in the article, some additional simplifications are made in the simulation model. The reason for this is to simplify the analytical expression in the ODE's used to simulate the dynamics of the simulation model. The ODE's will still describe a complex nonlinear system, but comparing predictive models with the simulation models, and thus analysing the performance of the novel predictive models will be clearer. From the expression for $\dot{T}_{bath}$ in (A.55), the terms E_u, E_{sc}, $E_{r_{Al}}$ and $E_{r_{bath}}$ are omitted. Thus, in the simulation model, the expression for $\dot{T}_{bath}$ is given by:

$$\dot{T}_{bath,\,sim} = \frac{P_{el} - Q_{bath-sl} - E_{tc,\,liq}}{m_{bath} c_{p_{bath,\,liq}}}. \tag{A.61}$$

This neglects some essential physical effects in the process. This is justified by the argument that the main purpose of this work is to evaluate data driven models on a complex nonlinear system, rather than simulating the dynamics of an aluminum electrolysis cell as good as possible.

Bibliography

[1] Kun Wang and WaiChing Sun. "A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning". In: *Computer Methods in Applied Mechanics and Engineering* 334 (2018), pp. 337–380. ISSN: 0045-7825.

[2] Yan Feng and Sorin Mitran. "Data-driven reduced-order model of microtubule mechanics". In: *Cytoskeleton* 75.2 (2018), pp. 45–60.

[3] Pamela P Peralta-Yahya, Fuzhong Zhang, Stephen B Del Cardayre and Jay D Keasling. "Microbial engineering for the production of advanced biofuels". In: *Nature* 488.7411 (2012), pp. 320–328.

[4] Soodabeh Esmaili and Shahab D. Mohaghegh. "Full field reservoir modeling of shale assets using advanced data-driven analytics". In: *Geoscience Frontiers* 7.1 (2016). Special Issue: Progress of Machine Learning in Geosciences, pp. 11–20. ISSN: 1674-9871.

[5] Vasilis Marmarelis, Georgios Mitsis, E Daskalaki, P Diem and S Mougiakakou. *Data-driven modeling for diabetes*. Lecture Notes in Bioengineering. Springer Berlin, Heidelberg, 2014. ISBN: 9783662523674.

[6] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams and Alán Aspuru-Guzik. "Automatic chemical design using a data-driven continuous representation of molecules". In: *ACS central science* 4.2 (2018), pp. 268–276.

[7] Francisco J. Montáns, Francisco Chinesta, Rafael Gómez-Bombarelli and J. Nathan Kutz. "Data-driven modeling and learning in science and engineering". In: *Comptes Rendus Mécanique* 347.11 (2019). Data-Based Engineering Science and Technology, pp. 845–855. ISSN: 1631-0721.

[8] Omer San, Adil Rasheed and Trond Kvamsdal. *Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution*. 2021. arXiv: 2103.14629 [physics.comp-ph].

[9] Kai Grotheim and Halvor Kvande. *Introduction to Aluminium Electrolysis-Understanding the Hall-Heroult Process*. Dusseldorf, Germany: Aluminium-Verlag, 1993.

[10] Vanderlei Gusberti, Dagoberte S Severo, Barry J Welch and Maria Skyllas-Kazacos. "Modeling the mass and energy balance of different aluminium smelting cell technologies". In: *Light Metals 2012*. Springer, 2012, pp. 929–934.

[11] Jinsong Hua, Christian Droste, Kristian E Einarsrud, Magne Rudshaug, Robert Jorgensen and Nils-Haavard Giskeodegard. "Revised benchmark problem for modeling of metal flow and metal heaving in reduction cells". In: *Light Metals* (2014), pp. 691–695.

[12] Meijia Sun, Baokuan Li and Linmin Li. "A multi-scale mathematical model of growth and coalescence of bubbles beneath the anode in an aluminum reduction cell". In: *Metallurgical and Materials Transactions B* 49.5 (2018), pp. 2821–2834.

[13] Kristian Etienne Einarsrud, Ingo Eick, Wei Bai, Yuqing Feng, Jinsong Hua and Peter J. Witt. "Towards a coupled multi-scale, multi-physics simulation framework for aluminium electrolysis". In: *Applied Mathematical Modelling* 44 (2017), pp. 3–24. ISSN: 0307-904X.

[14] He Song Li and Chang Wei Jiang. "Development and application of soft sensor model for heterogeneous information of aluminum reduction cells". In: *Control Engineering Practice* 19.10 (2011). ISSN: 09670661.

[15] Zhaohui Zeng, Weihua Gui, Xiaofang Chen, Yongfang Xie and Renchao Wu. "A mechanism knowledge-driven method for identifying the pseudo dissolution hysteresis coefficient in the industrial aluminium electrolysis process". In: *Control Engineering Practice* 102 (2020). ISSN: 09670661.

[16] A. Meghlaoui, J. Thibault, R.T. Bui, L. Tikasz and R. Santerre. "Neural networks for the identification of the aluminium electrolysis process". In: *Computers & Chemical Engineering* 22.10 (1998), pp. 1419–1428. ISSN: 0098-1354.

[17] You Peng, Birgit Braun, Casey McAlpin, Michael Broadway, Brenda Colegrove and Leo Chiang. "Contamination classification for pellet quality inspection using deep learning". In: *Computers & Chemical Engineering* 163 (2022), p. 107836. ISSN: 0098-1354.

[18] Nazatul Aini Abd Majid, Mark P. Taylor, John J.J. Chen and Brent R. Young. "Multivariate statistical monitoring of the aluminium smelting process". In: *Computers & Chemical Engineering* 35.11 (2011), pp. 2457–2468. ISSN: 0098-1354.

[19] Adil Rasheed, Omer San and Trond Kvamsdal. "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective". In: *IEEE Access* 8 (2020), pp. 21980–22012.

[20] Moritz von Stosch, Rui Oliveira, Joana Peres and Sebastião Feyo de Azevedo. "Hybrid semi-parametric modeling in process systems engineering: Past, present and future". In: *Computers and Chemical Engineering* 60 (2014), pp. 86–101. ISSN: 00981354.

[21] Mohammed Saad Faizan Bangi and Joseph Sang Il Kwon. "Deep hybrid modeling of chemical process: Application to hydraulic fracturing". In: *Computers and Chemical Engineering* 134 (2020). ISSN: 00981354.

[22] Dongheon Lee, Arul Jayaraman and Joseph S. Kwon. "Development of a hybrid model for a partially known intracellular signaling pathway through correction term estimation and neural network modeling". In: *PLoS Computational Biology* 16.12 (2020), pp. 1–31. ISSN: 15537358.

[23] Maziar Raissi, Paris Perdikaris and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[24] Suran Pawar, Omer San, Burak Aksoylu, Adil Rasheed and Trond Kvamsdal. "Physics guided machine learning using simplified theories". In: *Physics of Fluids* 33.1 (2021), p. 011701.

[25] Steven L. Brunton, Joshua L. Proctor and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the National Academy of Sciences* 113.15 (2016), pp. 3932–3937.

[26] Joseph Bakarji and Daniel M. Tartakovsky. "Data-driven discovery of coarse-grained equations". In: *Journal of Computational Physics* 434 (2021), p. 110219. ISSN: 0021-9991.

[27] Kathleen Champion, Bethany Lusch, J. Nathan Kutz and Steven L. Brunton. "Data-driven discovery of coordinates and governing equations". In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451. ISSN: 0027-8424.

[28] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu and Max Tegmark. “AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 4860–4871.

[29] Samuel Kim, Peter Y. Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Ceperic and Marin Soljacic. “Integration of Neural Network-Based Symbolic Regression in Deep Learning for Scientific Discovery”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.9 (2021), pp. 4166–4177.

[30] Hao Xu, Dongxiao Zhang and Nanzhe Wang. “Deep-learning based discovery of partial differential equations in integral form from sparse and noisy data”. In: *Journal of Computational Physics* 445 (2021), p. 110592. ISSN: 0021-9991.

[31] Rahul Rai and Chandan K. Sahu. “Driven by Data or Derived Through Physics? A Review of Hybrid Physics Guided Machine Learning Techniques With Cyber-Physical System (CPS) Focus”. In: *IEEE Access* 8 (2020), pp. 71050–71073.

[32] Laura von Rueden, Sebastian Mayer, Rafet Sifa, Christian Bauckhage and Jochen Garcke. “Combining Machine Learning and Simulation to a Hybrid Modelling Approach: Current and Future Directions”. In: *Advances in Intelligent Data Analysis XVIII*. Ed. by Michael R. Berthold, Ad Feelders and Georg Krempl. Cham: Springer International Publishing, 2020, pp. 548–560.

[33] Manuel Arias Chao, Chetan Kulkarni, Kai Goebel and Olga Fink. “Fusing physics-based and deep learning models for prognostics”. In: *Reliability Engineering & System Safety* 217 (2022), p. 107961. ISSN: 0951-8320.

[34] William Bradley, Jinhyeun Kim, Zachary Kilwein, Logan Blakely, Michael Eydenberg, Jordan Jalvin, Carl Laird and Fani Boukouvala. “Perspectives on the Integration between First-Principles and Data-Driven Modeling”. In: *Computers & Chemical Engineering* (2022), p. 107898. ISSN: 0098-1354.

[35] Sindre S. Blakseth, Adil Rasheed, Trond Kvamsdal and Omer San. “Deep neural network enabled corrective source term approach to hybrid analysis and modeling”. In: *Neural Networks* 146 (2022), pp. 181–199. ISSN: 0893-6080.

[36] C. E. Shannon. “Communication In The Presence Of Noise”. In: *Proceedings of the IEEE* 86.2 (1998), pp. 447–457.

[37] Steinar Kolås and Stein O. Wasbø. "A Nonlinear Model Based Control Strategy for the Aluminium Electrolysis Process". In: *Essential Readings in Light Metals: Volume 2 Aluminum Reduction Technology*. Ed. by Geoff Bearne, Marc Dupuis and Gary Tarcy. Cham: Springer International Publishing, 2016, pp. 825–829.

[38] Steinar Kolås. "Estimation in nonlinear constrained systems with severe disturbances". PhD thesis. Norwegian university of science and technology, 2008.

[39] Richard G. Baraniuk. "Compressive sensing". In: *IEEE Signal Processing Magazine* 24.4 (2007). ISSN: 10535888.

[40] Simon Foucart and Holger Rauhut. "An invitation to compressive sensing". In: *Applied and Numerical Harmonic Analysis*. 9780817649470. 2013.

[41] Ricardo Otazo, Emmanuel Candès and Daniel K. Sodickson. "Low-rank plus sparse matrix decomposition for accelerated dynamic MRI with separation of background and dynamic components". In: *Magnetic Resonance in Medicine* 73.3 (2015), pp. 1125–1136.

[42] Mohammed M Abo-Zahhad, Aziza I Hussein, Abdelfatah M Mohamed et al. "Compression of ECG signal based on compressive sensing and the extraction of significant features". In: *International Journal of Communications, Network and System Sciences* 8.05 (2015), p. 97.

[43] Kok-Kiong Poh and Pina Marziliano. "Compressive sampling of EEG signals with finite rate of innovation". In: *EURASIP journal on advances in signal processing* 2010 (2010), pp. 1–12.

[44] Xiao-Yang Liu, Yanmin Zhu, Linghe Kong, Cong Liu, Yu Gu, Athanasios V Vasilakos and Min-You Wu. "CDC: Compressive data collection for wireless sensor networks". In: *IEEE Transactions on Parallel and Distributed Systems* 26.8 (2014), pp. 2188–2197.

[45] Borhan M. Sanandaji, Tyrone L. Vincent, Michael B. Wakin, Roland. Tóth and Kameshwar Poolla. "Compressive System Identification of LTI and LTV ARX models". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. 2011, pp. 791–798.

[46] Zhe Bai, Eurika Kaiser, Joshua L. Proctor, J. Nathan Kutz and Steven L. Brunton. "Dynamic mode decomposition for compressive system identification". In: *AIAA Journal* 58.2 (2020), pp. 561–574. ISSN: 00011452.

[47] Yanglong Lu and Yan Wang. "Monitoring temperature in additive manufacturing with physics-based compressive sensing". In: *Journal of Manufacturing Systems* 48 (2018), pp. 60–70. ISSN: 02786125.

[48] Reinhard Heckel and Helmut Bolcskei. “Identification of sparse linear operators”. In: *IEEE Transactions on Information Theory* 59.12 (2013). ISSN: 00189448.

[49] Yannis Kopsinis, Konstantinos Slavakis and Sergios Theodoridis. “Online sparse system identification and signal reconstruction using projections onto weighted l1 balls”. In: *IEEE Transactions on Signal Processing* 59.3 (2011). ISSN: 1053587X.

[50] Yilun Chen, Yuantao Gu and Alfred O. Hero. “Sparse LMS for system identification”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. 2009.

[51] Yuantao Gu, Jian Jin and Shunliang Mei. “l0 norm constraint LMS algorithm for sparse system identification”. In: *IEEE Signal Processing Letters* 16.9 (2009). ISSN: 10709908.

[52] Nicholas Kalouptsidis, Gerasimos Mileounis, Behtash Babadi and Vahid Tarokh. “Adaptive algorithms for sparse system identification”. In: *Signal Processing* 91.8 (2011). ISSN: 01651684.

[53] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[54] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[55] Marvin Minsky and Seymour Papert. *Perceptrons*. Cambridge: MIT Press, 1969.

[56] Ian Goodfellow, Yoshua Bengio and Aaron Courville. *Deep learning*. MIT press, 2016.

[57] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[58] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[59] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie and Laith Farhan. “Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions”. In: *Journal of big Data* 8.1 (2021), pp. 1–74.

[60] Amine Naimi, Jiamei Deng, Altahhan Abdulrahman, Vineet Vajpayee, Victor Becerra and Nils Bausch. "Dynamic Neural Network-based System Identification of a Pressurized Water Reactor". In: 2020, pp. 100–104.

[61] Erasmo M. Rentería-Vargas, Carlos J. Zúñiga Aguilar, Jesse Y. Rumbo Morales, Felipe De Jesús S. Vázquez, Miguel De-La-Torre, José A. Cervantes, Estela S. Bustos and Manuela Calixto Rodríguez. "Neural Network-Based Identification of a PSA Process for Production and Purification of Bioethanol". In: *IEEE Access* 10 (2022), pp. 27771–27782.

[62] David Fooshee, Aaron Mood, Eugene Gutman, Mohammadamin Tavakoli, Gregor Urban, Frances Liu, Nancy Huynh, David Van Vranken and Pierre Baldi. "Deep learning for chemical reaction prediction". In: *Mol. Syst. Des. Eng.* 3 (3 2018), pp. 442–452.

[63] Vassilios D. Papadopoulos, Grigorios N. Beligiannis and Dimitra G. Hela. "Combining experimental design and artificial neural networks for the determination of chlorinated compounds in fish using matrix solid-phase dispersion". In: *Applied Soft Computing* 11.8 (2011), pp. 5155–5164. ISSN: 1568-4946.

[64] Patrizia R.S. Chermont, Fábio M. Soares and Roberto C.L. De Oliveira. "Simulations on the bath chemistry variables using neural networks". In: *TMS Light Metals*. Vol. 2016-January. 2016.

[65] Alan Marcel Fernandes de Souza, Fábio Mendes Soares, Marcos Antonio Gomes de Castro, Nilton Freixo Nagem, Afonso Henrique de Jesus Bitencourt, Carolina de Mattos Affonso and Roberto Célio Limão de Oliveira. "Soft sensors in the primary aluminum production process based on neural networks using clustering methods". In: *Sensors* 19.23 (2019). ISSN: 14248220.

[66] Dipankar Bhattacharyay, Duygu Kocaefe, Yasar Kocaefe and Brigitte Morais. "An artificial neural network model for predicting the CO 2 reactivity of carbon anodes used in the primary aluminum production". In: *Neural computing and Applications* 28.3 (2017), pp. 553–563.

[67] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen and Xin Wang. "A Survey of Deep Active Learning". In: *ACM Computing Surveys* 54.9 (Oct. 2021). ISSN: 0360-0300.

[68] Yarin Gal, Riashat Islam and Zoubin Ghahramani. "Deep Bayesian Active Learning with Image Data". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 1183–1192.

[69] Christopher Schröder and Andreas Niekler. “A survey of active learning for text classification using deep neural networks”. In: *arXiv preprint arXiv:2008.07267* (2020).

[70] Hamed H Aghdam, Abel Gonzalez-Garcia, Joost van de Weijer and Antonio M López. “Active learning for deep detection neural networks”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3672–3680.

[71] Jiameng Fan, Chao Huang, Xin Chen, Wenchao Li and Qi Zhu. “Reachnn*: A tool for reachability analysis of neural-network controlled systems”. In: *Automated Technology for Verification and Analysis: 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19–23, 2020, Proceedings*. Springer. 2020, pp. 537–542.

[72] Diego Manzanas Lopez, Patrick Musau, Nathaniel P Hamilton and Taylor T Johnson. “Reachability analysis of a general class of neural ordinary differential equations”. In: *Formal Modeling and Analysis of Timed Systems: 20th International Conference, FORMATS 2022, Warsaw, Poland, September 13–15, 2022, Proceedings*. Springer. 2022, pp. 258–277.

[73] Xavier Bombois, Michel Gevers, Roland Hildebrand and Gabriel Solari. “Optimal experiment design for open and closed-loop system identification”. In: *Communications in Information and Systems* 11.3 (2011), pp. 197–224.

[74] Mona Buisson-Fenet, Friedrich Solowjow and Sebastian Trimpe. “Actively Learning Gaussian Process Dynamics”. In: *Proceedings of the 2nd Conference on Learning for Dynamics and Control*. Vol. 120. Proceedings of Machine Learning Research. PMLR, 2020, pp. 5–15.

[75] Shengbing Tang, Kenji Fujimoto and Ichiro Maruta. “Actively Learning Gaussian Process Dynamical Systems Through Global and Local Explorations”. In: *IEEE Access* 10 (2022), pp. 24215–24231.

[76] Andrew Wagenmaker and Kevin Jamieson. “Active learning for identification of linear dynamical systems”. In: *Conference on Learning Theory*. PMLR. 2020, pp. 3487–3582.

[77] Horia Mania, Michael I Jordan and Benjamin Recht. “Active learning for nonlinear system identification with guarantees”. In: *arXiv preprint arXiv:2006.10277* (2020).

[78] Erlend T. B. Lundby, Adil Rasheed, Jan T. Gravdahl and Ivar J. Halvorsen. “A novel hybrid analysis and modeling approach applied to aluminum electrolysis process”. In: *Journal of Process Control* 105 (2021), pp. 62–77. ISSN: 0959-1524.

[79] Haakon Robinson, Erlend Lundby, Adil Rasheed and Jan T. Gravdahl. "A novel corrective-source term approach to modeling unknown physics in aluminum extraction process". In: *arXiv preprint arXiv:2209.10861* (2022).

[80] Erlend T. B. Lundby, Adil Rasheed, Jan T. Gravdahl and Ivar J. Halvorsen. "Sparse deep neural networks for modeling aluminum electrolysis dynamics". In: *Applied Soft Computing* 134 (2023), p. 109989. ISSN: 1568-4946.

[81] Erlend T. B. Lundby, Haakon Robinsson, Adil Rasheed, Ivar J. Halvorsen and Jan T. Gravdahl. "Sparse neural networks with skip-connections for nonlinear system identification". In: *arXiv preprint arXiv:2301.00582* (2023).

[82] Gao Huang, Zhuang Liu, Laurens Van Der Maaten and Kilian Q. Weinberger. "Densely Connected Convolutional Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269.

[83] Erlend T. B. Lundby, Adil Rasheed, Ivar J. Halvorsen, Dirk Reinhardt, Sebastien Gros and Jan T. Gravdahl. "Deep active learning for nonlinear system identification". In: *arXiv preprint arXiv:2302.12667* (2023).

[84] Håkon Viumdal, Saba Mylvaganam and David Di Ruscio. "System identification of a non-uniformly sampled multi-rate system in aluminium electrolysis cells". In: *Modeling Identification and Control* 35.3 (2014), pp. 127–146.

[85] Laszlo I. Kiss and Véronique Dassylva-Raymond. "Freeze thickness in the aluminum electrolysis cells". In: *Light Metals 2008 - Proceedings of the Technical Sessions Presented by the TMS Aluminum Committee at the TMS 2008 Annual Meeting and Exhibition*. TMS. New Orleans, LA., Mar. 2008, pp. 431–436.

[86] Thor I. Fossen. *Handbook of Marine Craft Hydrodynamics and Motion Control*. John Wiley & Sons, May 2011.

[87] Anders Aglen Pedersen. "Optimization Based System Identification for the milliAmpere Ferry". MA thesis. NTNU, 2019.

[88] Jonathan Frankle and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". In: *arXiv preprint arXiv:1803.03635* (2018).

[89] Michael Zhu and Suyog Gupta. *To prune, or not to prune: exploring the efficacy of pruning for model compression*. 2017.

[90] Xiaoqin Zeng and Daniel S Yeung. “Hidden neuron pruning of multilayer perceptrons using a quantified sensitivity measure”. In: *Neurocomputing* 69.7-9 (2006), pp. 825–837.

[91] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden and Alexandra Peste. “Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks.” In: *J. Mach. Learn. Res.* 22.241 (2021), pp. 1–124.

[92] Balas K. Natarajan. “Sparse approximate solutions to linear systems”. In: *SIAM Journal on Computing* 24.2 (1995), pp. 227–234. ISSN: 00975397.

[93] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[94] Keze Wang, Dongyu Zhang, Ya Li, Ruimao Zhang and Liang Lin. “Cost-Effective Active Learning for Deep Image Classification”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 27.12 (2017), pp. 2591–2600.

[95] Ying Li, Binbin Fan, Weiping Zhang, Weiping Ding and Jianwei Yin. “Deep active learning for object detection”. In: *Information Sciences* 579 (2021), pp. 418–433. ISSN: 0020-0255.

[96] Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos and Tom M Mitchell. “Competence-based curriculum learning for neural machine translation”. In: *arXiv preprint arXiv:1903.09848* (2019).

[97] Ye Zhang, Matthew Lease and Byron C. Wallace. “Active Discriminative Text Representation Learning”. In: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. AAAI’17. San Francisco, California, USA: AAAI Press, 2017, pp. 3386–3392.

[98] Sreyasee Das Bhattacharjee, Ashit Talukder and Bala Venkatram Balantrapu. “Active learning based news veracity detection with feature weighting and deep-shallow fusion”. In: *2017 IEEE International Conference on Big Data (Big Data)*. 2017, pp. 556–565.

[99] Fedor Zhdanov. “Diverse mini-batch active learning”. In: *arXiv preprint arXiv:1901.05954* (2019).

[100] Niklas Koep, Arash Behboodi and Rudolf Mathar. “An Introduction to Compressed Sensing”. In: *Applied and Numerical Harmonic Analysis*. Springer International Publishing, 2019, pp. 1–65.

[101] Richard G Baraniuk, Volkan Cevher, Marco F Duarte and Chinmay Hegde. "Model-based compressive sensing". In: *IEEE Transactions on information theory* 56.4 (2010), pp. 1982–2001. ISSN: 00189448.

[102] S. S. Chen, D. L. Donoho and M. A. Saunders. "Atomic decomposition by basis pursuit". In: *SIAM Review* 43.1 (Mar. 2001), pp. 129–159. ISSN: 00361445.

[103] Robert Tibshirani. "Regression Shrinkage and Selection Via the Lasso". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (Jan. 1996), pp. 267–288. ISSN: 0035-9246.

[104] Emmanuel Candes and Terence Tao. "The Dantzig selector: Statistical estimation when p is much larger than n". In: *The annals of Statistics* 35.6 (2007), pp. 2313–2351.

[105] Thomas Blumensath and Mike E. Davies. "Iterative hard thresholding for compressed sensing". In: *Applied and Computational Harmonic Analysis* 27.3 (Nov. 2009), pp. 265–274. ISSN: 10635203.

[106] Xi Chen, Qihang Lin, Seyoung Kim, Jaime G Carbonell, Eric P Xing et al. "Smoothing proximal gradient method for general structured sparse regression". In: *The Annals of Applied Statistics* 6.2 (2012), pp. 719–752.

[107] Amir Beck and Marc Teboulle. "A fast iterative shrinkage-thresholding algorithm for linear inverse problems". In: *SIAM journal on imaging sciences* 2.1 (2009), pp. 183–202.

[108] Joel A Tropp and Anna C Gilbert. "Signal recovery from random measurements via orthogonal matching pursuit". In: *IEEE Transactions on information theory* 53.12 (2007), pp. 4655–4666.

[109] Deanna Needell and Joel A Tropp. "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples". In: *Applied and computational harmonic analysis* 26.3 (2009), pp. 301–321.

[110] Anela M. Ivanova, Pavel A. Arkhipov, Olga Tkacheva and Yury P. Zaikov. "Experimental Studies of the Dynamic Formation of the Side Ledge in an Aluminum Electrolysis Cell". In: *Russian Metallurgy (Metally)* 2020.2 (2020). ISSN: 15556255.

[111] Greg Welch and Gary Bishop. "An Introduction to the Kalman Filter". In: *In Practice* 7.1 (2006), pp. 1–16. ISSN: 10069313.

[112] Sindre S. Blakseth, Adil Rasheed, Trond Kvamsdal and Omer San. "Combining physics-based and data-driven techniques for reliable hybrid analysis and modeling using the corrective source term approach". In: *Applied Soft Computing* 128 (2022), p. 109533. ISSN: 1568-4946.

[113] Randall J. LeVeque. *Finite-Volume Methods for Hyperbolic Problems*. 1st. Cambridge University Press, 2002.

[114] Floris Takens. "Detecting strange attractors in turbulence". In: *Dynamical Systems and Turbulence, Warwick 1980*. Ed. by David Rand and Lai-Sang Young. Berlin, Heidelberg: Springer Berlin Heidelberg, 1981, pp. 366–381. ISBN: 978-3-540-38945-3.

[115] Lennart Ljung. *System Identification: Theory for the User*. 2nd. Pearson, 1998. ISBN: 978-0136566953.

[116] Oliver Nelles. *Nonlinear system identification: from classical approaches to neural networks, fuzzy models, and gaussian processes*. Springer Nature, 2020.

[117] Maximilian Winter and Christian Breitsamter. "Nonlinear identification via connected neural networks for unsteady aerodynamic analysis". In: *Aerospace Science and Technology* 77 (2018), pp. 802–818. ISSN: 1270-9638.

[118] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[119] J. Sjöberg and L. Ljung. "Overtraining, Regularization and Searching for a Minimum, with Application to Neural Networks". In: *International Journal of Control* 62.6 (Dec. 1995), pp. 1391–1407. ISSN: 0020-7179. DOI: 10.1080/00207179508921605. URL: https://doi.org/10.1080/00207179508921605.

[120] Christopher M. Bishop. "Regularization and Complexity Control in Feed-Forward Networks". In: *International Conference on Artificial Neural Networks*. 1995, pp. 141–148. ISBN: 978-2-910085-19-3. URL: https://publications.aston.ac.uk/id/eprint/524/ (visited on 25/03/2023).

[121] Yu Zhang, Peter Tiňo, Aleš Leonardis and Ke Tang. *A Survey on Neural Network Interpretability*. 2021. arXiv: 2012.14261 [cs.LG].

[122] Shiwei Liu, Decebal Constantin Mocanu and Mykola Pechenizkiy. "On improving deep learning generalization with adaptive sparse connectivity". In: *arXiv preprint arXiv:1906.11626* (2019).

[123] Michael C Mozer and Paul Smolensky. "Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 1. Morgan-Kaufmann, 1988.

[124] Hongpeng Zhou, Chahine Ibrahim, Wei Xing Zheng and Wei Pan. "Sparse Bayesian deep learning for dynamic system identification". In: *Automatica* 144 (2022), p. 110489. ISSN: 0005-1098.

[125] Maarten Schoukens, P Mattson, Torbjörn Wigren and Jean-Philippe Noel. "Cascaded tanks benchmark combining soft and hard nonlinearities". In: *Workshop on nonlinear system identification benchmarks*. 2016, pp. 20–23.

[126] Torbjörn Wigren and Maarten Schoukens. *Coupled electric drives data set and reference models*. Department of Information Technology, Uppsala Universitet, 2017.

[127] Guido F Montufar, Razvan Pascanu, Kyunghyun Cho and Yoshua Bengio. "On the Number of Linear Regions of Deep Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence and K. Q. Weinberger. Vol. 27. Curran Associates, Inc., 2014.

[128] Razvan Pascanu, Guido Montufar and Yoshua Bengio. "On the number of response regions of deep feed forward networks with piece-wise linear activations". In: ICLR, 2014.

[129] Thiago Serra, Christian Tjandraatmadja and Srikumar Ramalingam. "Bounding and Counting Linear Regions of Deep Neural Networks". In: *CoRR* abs/1711.02114 (2017). arXiv: 1711.02114.

[130] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer and Tom Goldstein. *Visualizing the Loss Landscape of Neural Nets*. 2017. URL: https://arxiv.org/abs/1712.09913.

[131] Omer Sagi and Lior Rokach. "Ensemble learning: A survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 8.4 (2018), e1249.

[132] Jeremy Nixon, Balaji Lakshminarayanan and Dustin Tran. "Why are bootstrapped deep ensembles not better?" In: *"I Can't Believe It's Not Better!"NeurIPS 2020 workshop*. 2020.

[133] Balaji Lakshminarayanan, Alexander Pritzel and Charles Blundell. "Simple and scalable predictive uncertainty estimation using deep ensembles". In: *Advances in neural information processing systems* 30 (2017).

[134] Stanislav Fort, Huiyi Hu and Balaji Lakshminarayanan. "Deep ensembles: A loss landscape perspective". In: *arXiv preprint arXiv:1912.02757* (2019).

[135] Jakob Gawlikowski, Cedrique R N Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher et al. "A survey of uncertainty in deep neural networks". In: *arXiv preprint arXiv:2107.03342* (2021).

[136] William H Beluch, Tim Genewein, Andreas Nürnberger and Jan M Köhler. "The power of ensembles for active learning in image classification". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 9368–9377.

[137] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan and Jasper Snoek. "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift". In: *Advances in neural information processing systems* 32 (2019).

[138] Fredrik K Gustafsson, Martin Danelljan and Thomas B Schon. "Evaluating scalable bayesian deep learning methods for robust computer vision". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 318–319.

[139] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[140] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings and Moritz Diehl. "CasADi – A software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (2019), pp. 1–36.

[141] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019).

[142] Tim Salzmann, Elia Kaufmann, Jon Arrizabalaga, Marco Pavone, Davide Scaramuzza and Markus Ryll. "Real-time Neural-MPC: Deep Learning Model Predictive Control for Quadrotors and Agile Robotic Platforms". In: *arXiv preprint arXiv:2203.07747* (2023).

[143] Ibrahim Dincer and Calin Zamfirescu. "Chapter 1 - Fundamentals of Thermodynamics". In: *Advanced Power Generation Systems*. Ed. by Ibrahim Dincer and Calin Zamfirescu. Boston: Elsevier, 2014, pp. 1–53.

[144] Sigurd Skogestad. *Chemical and energy process engineering*. CRC press, 2008.

[145] M.P. Taylor, W.D. Zhang, V. Wills and S. Schmid. “A Dynamic Model for the Energy Balance of an Electrolysis Cell”. In: *Chemical Engineering Research and Design* 74.8 (1996), pp. 913–933.

[146] Tormod Drengstig. “On process model representation and AlF3 dynamics of aluminum electrolysis cells”. PhD thesis. Ph. D. thesis, Norwegian Univ. of Science and Tech.(NUST), 1997.

[147] Stefan W. Jessen. “Mathematical modeling of a Hall Héroult aluminium reduction cell”. MA thesis. Technical University of Denmark, DTU, DK-2800 Kgs. Lyngby, Denmark, 2008.

[148] Thomas M Hyde and Barry J Welch. “The gas under anodes in aluminium smelting cells. Part I: Measuring and modelling bubble resistance under horizontally oriented electrodes”. In: *LIGHT METALS-WARRENDALE-* (1997), pp. 333–340.

[149] Asbjørn Solheim, Sverre Rolseth, Egil Skybakmoen, Lisbet Støen, Åsmund Sterten and Trond Støre. “Liquidus temperature and alumina solubility in the system Na 3 AlF 6-AlF 3-LiF-CaF 2-MgF 2”. In: *Essential Readings in Light Metals: Aluminum Reduction Technology, Volume 2*. John Wiley & Sons, 2013, pp. 73–82.

ISBN 978-82-326-7114-4 (printed ver.)
ISBN 978-82-326-7113-7 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU

Norwegian University of
Science and Technology