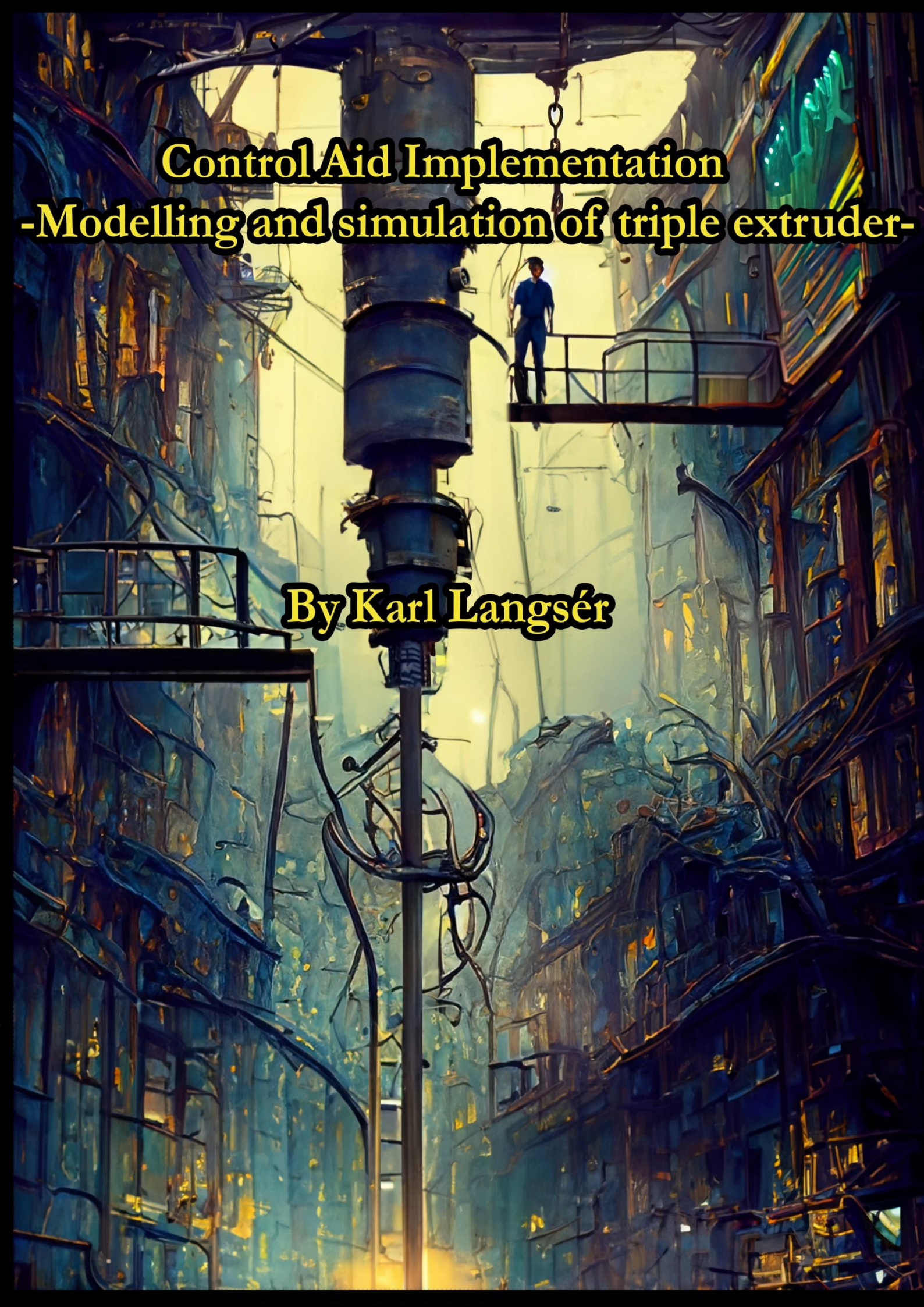
Control Aid Implementation
-Modelling and simulation of triple extruder-
By Karl Langsér

Control Aid Implementation

-Modelling and simulation of triple extruder-

by

Karl Langsér

Department of chemical engineering

Lund University

with

NKT HV Cables AB

for

Master Thesis

June 2023

Bernt Nilsson - Examiner

Niklas Andersson - Head Supervisor

Erik Lundström - Supervisor

Tommy Johansson – Supervisor

[The following section has been left empty on purpose]

Popular science summary

Modelling of cable extruders through Machine Learning

Introduction

In the production of their High Voltage Direct Current Cables (HVDC) and High Voltage Alternating Current Cables (HVAC), NKT uses triple extruders to create layers of insulation and semi-conduction in their cables to improve its electrical properties. To optimize the process, a machine learning model was created to simulate the cable-product based on the extrusion-head inputs.

Project

In simple terms, an extruder is a machine that presses out molten plastic in a mold-like fashion to acquire the desired dimensions and shape of the plastic product. These layers are extruded around the conductor (a copper or aluminium rod or wire) through three separate extruder screws that are all connected to a triple extrusion head. In the triple head the melts flow together around the conductor. Imagine putting a small rod in the middle of a toothpaste tube and pressing out the toothpaste around the rod, then we put that in the same manner in the middle of a larger toothpaste tube and press the toothpaste out again around our toothpaste coated rod. The toothpaste would in this example be the plastic and the final product will be a rod (conductor) with layers of toothpaste (plastic insulation) stacked on top of one another.

This extrusion process is controlled by manipulation of the extruder screw motor speeds; the screw is what drives the melt forward; the puller rate, which is the speed with which the extruded cable is pulled away from the triple head and the position of the distributor walls in the triple head. The distributor walls can be thought of as walls that keep the plastic melt from flowing together before the actual extrusion die.

To better control the process and optimize the cable dimensions so that the insulation can be thinner without running the risk of being too thin (which would destroy the cable), this project aimed to create a model that can predict how the extruder is affected by the inputs described above, with respect to layer thickness and centering of the conductor in each layer.

The model produced is a neural network of Extreme learning type. The Extreme part basically just means that the learning is a matrix operation and not weight tuning, which lets it learn extremely fast compared to models where each weight must be tuned.

The extreme learning machine was built in python, together with software that helps in the training of the model, as well as two different Graphical User Interface that can be used to simulate the model. One of the GUIs Simulate the thickness of each layer at four different points perpendicular to each other in the cable, as a line graph. The other GUI simulates the cross-section of the cable.

The model was tested and validated in the working range of the process with certain parts of the inputs unknown and was shown to capture the correct trend in the working range. If the unknown inputs matched those of the test and training sets, it would accurately simulate the real process. The model is a proof of concept of how simple it is to model complex dynamics with little computing power utilizing machine learning and serves as a foundation for building optimal controllers or other control aid for the extrusion process.

[The following section has been left empty on purpose]

Acknowledgements

I want to start by thanking NKT and Rickard Nilsson for letting me do my master thesis together with them. A special mention goes out to Tommy Johansson and Erik Lundström for acting as aids during the project. Another mention goes to Erik Lundström for helping me understand the process and which variables should be the focus of the project, which greatly reduced the time it would have taken to understand the results and how to get the model working.

For helping in scheduling a visit to see a start-up of the process line, and a walkthrough of the full process line, I want to thank Jennie Aborres.

I also want to thank Bernt Nilsson and Niklas Andersson for acting as Examinator and Head supervisor of this project, making it possible for me to conduct my master thesis.

Abstract

In the production of their High Voltage Direct Current Cables (HVDC) and High Voltage Alternating Current Cables (HVAC), NKT uses triple extruders to create layers of insulation and semi-conduction. A model to predict the effect of extruder inputs on the cable's insulation and semi-conducting layers has been created and trained to predict the extruder in discrete time. The project developed a deep learning extreme learning machine algorithm and proved that it has good enough accuracy and generalization to predict the cable states as a function of extruder motor and line speeds as well as distributor screw positions.

The project also explored how to evaluate and test a model, despite the fact it has insufficient data, by dividing it into two parts and letting them be trained independently, with a subset of unknown inputs. The results were satisfactory. A way of combining the models was also proposed but was not further explored.

While the model shows that it can accurately describe the extruder in the ranges of where it is trained, the data acquisition was poor and hindered the collection of good enough training sets to let the model predict over the whole input range. The model does, however, act as a proof of concept that can be further developed into a finished state. It also showed that it can still predict the correct trends of the extruder even if the model is acting outside of the range where it is trained.

Contents

1. Background and Introduction

In the production of their High Voltage Direct Current Cables (HVDC) and High Voltage Alternating Current Cables (HVAC), NKT uses triple extruders to create layers of insulation and semi-conduction. The layers are extruded around the conductor with a tripe extruder head which is controlled by a process operator.

To achieve optimal performance in the cables, the layers need to be centered around the conductor and have the right thickness. This is achieved by manual operation of the extruder input settings together with analogue inputs. Currently there is no clear model of how the extruder inputs affect the centering and the thickness of the three layers; the operators use heuristics and experience to achieve a good result. This means that the education of new process operators is expensive and time consuming and the process suboptimal.

This project aims to develop a model that is accurate enough to simulate the real-world process and make the extruder operation easier and more precise. The simulation and data will be presented to the operator in a graphical user interface (GUI).

2. Problem formulation

The first part of the project involves stating the problems, and the operations needed to be done to have a final product capable of fulfilling the goals of the project. This means clarifying the goals, and the problems that need to be solved to achieve them.

2.1. Goals and outcome

First, a model that can describe how the centering of the layers vary with the extruder inputs is needed. Second a model to describe how the layer thickness varies with the extruder input. These then need to be combined into a model that can describe both as a function of extruder inputs.

The model will then be used to simulate the process and present it in a graphical user interface (GUI) to help an operator control the process. It will be implemented in the process to continuously track the inputs to the extruder or help the operator upon start up; another use can be as an aid in educating new process engineers or operators.

2.2. Problems

The first problem is to formulate the first iteration of a mathematical model that can be used in describing the process. How do the process parameters vary with the independent variables? Are the inputs the only independent variables, and if there are more, do they add degrees of freedom or not?

Software that can fit the model to experimental data is needed to tune the process parameters of the model. How will this tool be built? How good does the model need to be before it can be used to achieve the desired result?

How will the model and data be presented to the operator? How will the GUI be built? And what information is useful to an operator?

2.3. Initial approach

A literature study was conducted to get initial information on how the process works and what tools there are to solve the problem. The purpose was to get an understanding and inspiration on how to tackle the task at hand. This literature study was continued all throughout the project.

From the initially gathered information, the process would be, roughly, a three-step process. First the data will need to be extracted from the database where it is located, then a model of choice needs to be decided on. The model then needs to be fitted to data to see how well it performs. This is iterated on until a satisfactory model is obtained.

The model should then be expanded into all aspects of the process and tuned so that it has a good enough prediction capacity. And if there is time and possibility improved as much as possible.

2.4. Additional problems

From observing the real process and dialog with personnel at NKT, the process of obtaining the data needed for some of the inputs will be the biggest issue when trying to figure out how the model should be trained. While there are experiments from an earlier job, it is not enough to be used as basis for model training.

3. Literature study

The first part of the process is understanding what variables affect the two parameters, centering, and thickness, in each layer. Are the parameters independent or dependent? What are the material properties of the layers, and how will they affect the model. What previous work has been done in modelling similar processes and what parameters have been looked at?

This part aims to explore what plastic extrusion is, and how it is utilized in industry, as well as what is known about the process of extrusion in terms of modelling and simulation.

3.1. Plastic extrusion

The extrusion process of polymeric materials is an integrated process with many components and different processes in a production line. The extent of the line may vary a lot depending on the application of the product and production methods. It is multi-variable dependent, and small off-sets in the operating parameters can yield useless products. For the plastic to, set accordingly; behave in a controllable manner; and produce a predictable result, the temperature profile in the process must be closely monitored and controlled[1].

The general process can be compartmentalized into a few different steps. First the plastic material is received, inspected, and stored. Here an overview of the inventory is done, to assure that there are no defects other than the expected ones. After this the material might go through a blending process where additives are added, it may also be done previously by the supplier. This is done so that the plastic has the right material properties for its purpose. Some resins must be dried before entering the extruder to eliminate polymer degradation due to moisture or to remove condensate of the surface caused by cold storage spaces[1, 2].

Material is then fed to the extruder; melted mixed and ready to be molded. It is then transported to the die where it is shaped into the required shape. After it has been extruded through the die, it is cooled down as it is pulled away from the die at constant speed, to attain the right shape and cross section. Secondary operations are normally applied after the puller[1]. The product is then inspected for defects and quality assured before it is shipped to the customer[1].

In wire coating, an extrusion method known as coextrusion is used to achieve the required characteristics and electrical properties in the wire. Coextrusion is achieved by having several extrusion-dies extrude onto each other in a simultaneous fashion.

The adhesion between the layers is driven by process factors, assuming an appropriate tie layer has been applied. The four major factors are tie layer thickness, functionality in the tie layer, melt temperature and contact time[3].

3.2. Parameters

Modelling of the process parameters centering, and layer thickness, seems to have varying complexity and differ a lot in terms of modelling. While the thickness seems to be somewhat easy to model, the initial centering of the cable is not[2]. An image describing these measurements is displayed below in figure 3.1.

The centering of the layers around the conductor is a process that is very non-linear and hard to predict with linear approximations. If this was the only problem a model could be made with presumably little difficulty using various methods, e.g. the extreme learning machine, as has been done previously[4], but to a different set of inputs. The tuning screws for the distributor walls are, however, an analogue input and are not monitored, making it difficult to estimate the impact of each input on the extruder. And fitting a model without the right training data can prove to be difficult.

3.3. Utilization in the cable industry

In the cable industry, extrusion can be utilized when creating the insulation and semi conductive layers in the cables.

The first step in the process involves melting the plastic that is to be extruded; it comes as granules that need to be tightly sealed from the outside to prevent contamination of the insulation that might cause performance loss[2]. The molten plastic is then transported to the extrusion die where it is pressed out in the shape that it is supposed to have. All the layers are typically co-extruded around the conductor and dimensions are closely monitored[1].

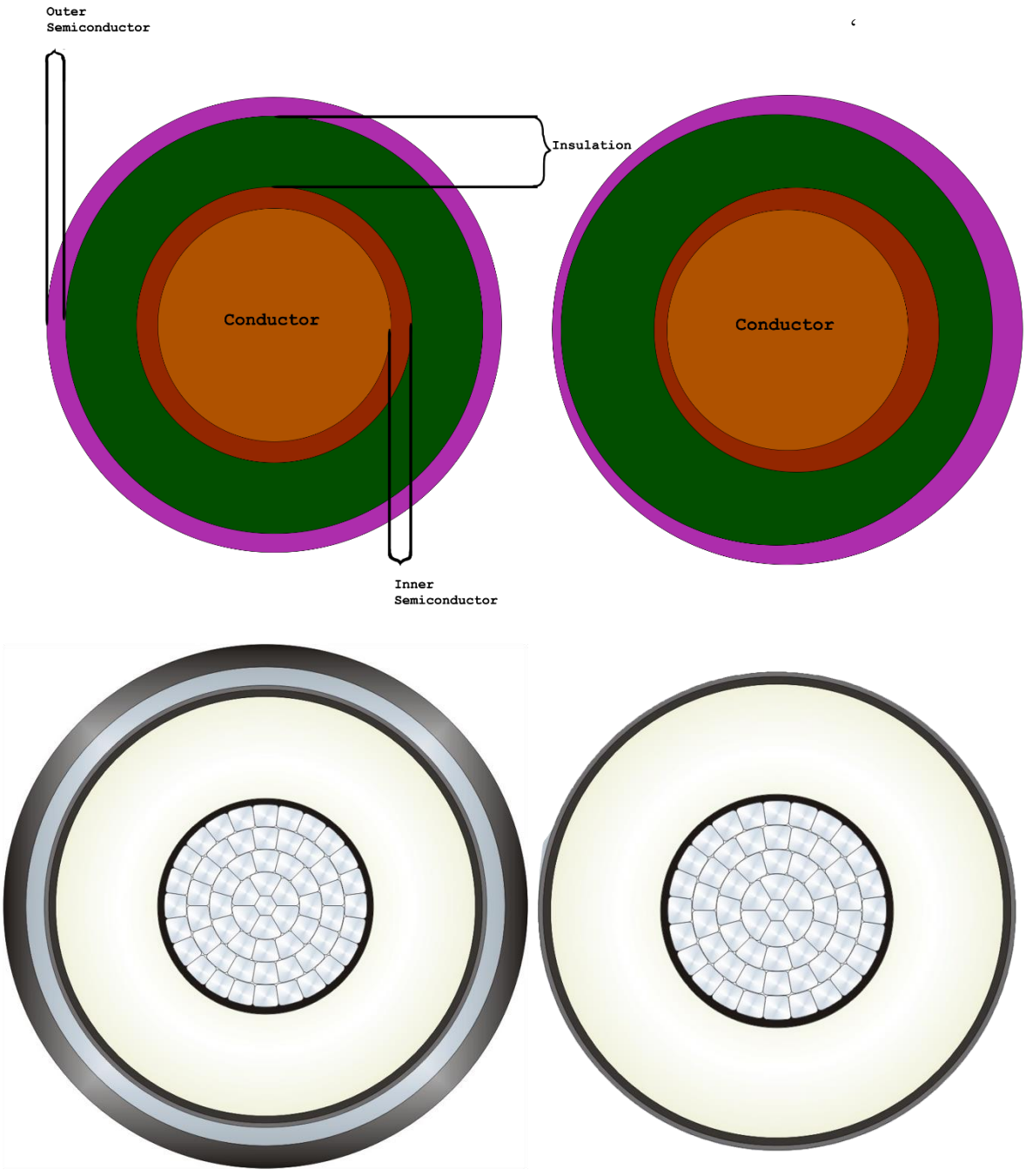

Figure 3.1: Image displaying the cross-section of the cable and the thickness as well as a display of what poor centering may look like. The bottom left of the picture shows a stylized picture of the real cable cross-section, and the bottom right shows which part is being modelled.

3.4. Extrusion modelling

Previous work has been done to try and model different types of extruder processes. K.S Boparai et al.[5], have modelled an extrusion process called fused deposition modeling, a type of 3D printing procedure. They utilized a response surface methodology (RSM), which is a statistical way of exploring the interconnectivity and relations between independent variables and a response variable[5]. The procedure can be viewed mathematically according to eq 3.1.

$$\begin{aligned} Y &= f(x_1, x_2, x_3, x_4 \dots, x_n) \pm \varepsilon \\ \eta &= f(x_1, x_2, x_3, x_4 \dots, x_n) \\ Y &= \eta \pm \varepsilon \end{aligned} \qquad \textbf{3.1}$$

Where Y is the desired response function and f is the function of independent input variables x, η is the response surface, and ε is the fitting error.

The response surface represents the expected response from the input set. The functional relationship can be determined by selecting a polynomial of higher order[5, 6]. They proposed a second order polynomial as the approximation for the response function. It can be written mathematically as eq 3.2.

$$Y = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n} a_{ii} x_i^2 + \sum_{i<j}^{n} a_{ij} x_i x_j + \varepsilon \qquad \textbf{3.2}$$

Where a_i is the linear effect of x_i, a_{ii} is the quadratic effect of x_i and a_{ij} and reveals the linear-by-linear interaction between x_i and x_j. The resulting response model accurately fit the experimental data and the goodness of fit was acceptable[5].

3.5. Extrusion control

Other attempts at modeling extruders have been made in attempts to control the process. Specifically in the cable extrusion industry attempts at automating conductor centering, and layer thickness tuning have been made.

Hui Li et al.[4], has done research on cable extrusion control, utilizing the Extreme learning machine (ELM) to program a Heuristic dynamic programming controller (HDP) for a system like the one shown in figure 3.2.

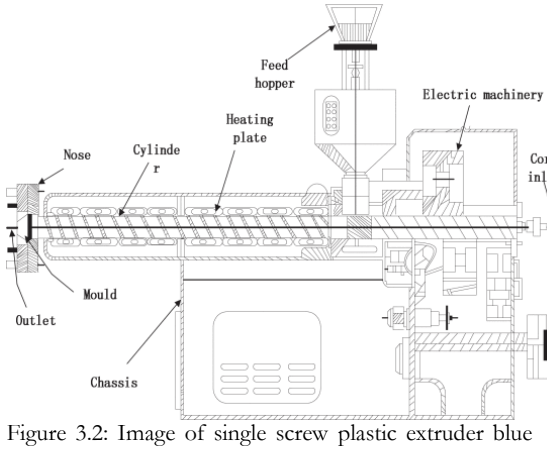

Figure 3.2: Image of single screw plastic extruder blue print. [4]

3.5.1 Extreme learning machine

The ELM concept is essentially a single feed forward neural network; A standard feedforward network (SLFN) with L hidden neurons (nodes), and N arbitrary samples ($x_i$and t_i) can be written as eq 3.3.

$$\begin{aligned} & x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in R^n, N \neq n \\ & t_i = [t_{i1}, t_{i2}, \dots, t_{im}]^T \in R^m \\ & \sum_{j=1}^{L} \beta_j g(w_j \cdot x_i + b_j) = o_i, i = 1, \dots, N \\ & (\beta_j, b_j) \in R \end{aligned} \qquad \textbf{3.3}$$

[4, 7]

Where x_i and t_i are input and target values. o_i is the output of the ELM and w_j is the input weight-matrix between the hidden layer neurons and the input layer neurons. g is the activation function for the hidden layer which can be a number of nonlinear functions, e.g., sigmoid or multi-quadratic; b_j is the bias for the hidden neurons and β is the output weight-matrix[7].

The learning process of the ELM is basically to minimize an error, most commonly a mean square error between the output of the ELM and the expected target value. It can be depicted mathematically as equation 3.4.

$$E = \sum_{j=1}^{L} (o_i - t_i)^2, i = 1, \dots, N \qquad \textbf{3.4}$$

Where E is the error that is to be minimized. The prediction model of the extruder was based on the ELM.

3.5.2 Improving the ELM

The robustness of the ELM can be further improved by utilizing the theory of Bartlett[8]. It states that the robustness of the ELM increases when the norm of the output weights decreases. This gives rise to a new optimization problem by utilizing ridge regression in the ELM seen in eq 3.5[9].

$$E = o_i - t_i, i = 1, \dots, N$$
$$\min_{E,\beta} \frac{1}{2}\|\beta\|_2^2 + C\,\frac{1}{2}\|E\|_2^2 \qquad \textbf{3.5}$$

Since the problem is no longer a simple least square minimization with respect to only the prediction error the MP pseudo inverse of the hidden layer matrix can no longer be used to calculate the output weights. Instead, the solution can be obtained by equation 3.6[10].

$$\beta = \left(\frac{I}{C} + H^T H\right)^{-1} H^T Y, \mathrm{L} < \mathrm{N}$$
$$\beta = H^T \left(\frac{I}{C} + HH^T\right)^{-1} \cdot Y, \mathrm{L} > \mathrm{N} \qquad \textbf{3.6}$$
$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix}$$
$$h = g\left(a_j \cdot x_i + b_j\right)$$

Where C is the error penalty factor, or generalization factor, determining how much weight should be put on model accuracy[10].

3.5.3 Heuristic dynamic programing

The prediction model achieved based on the ELM could accurately fit the experimental data from the extruder and gave a good description of the real process, even better than some back propagation network that it was evaluated against[4].

An HDP controller was created based on the ELM prediction model. The purpose of the Model network in HDP is to predict the next state of the system. It is then implemented in an HDP learning algorithm which sets the weights of the different learning machines by letting the states propagate forward through the network until the weights minimize a cost function[4].

3.5.4 How good is ELM for cable extrusion control?

Hui Li et al., concluded in their report that the ELM based HDP controller could accurately fit the experimental data and program the weights of the HDP, thus making it possible to control the extruder in this way.

3.6. The Deep ELM

When on the topic of ELMs and their ability to train single feed forward networks quickly and accurately with good generalization, it is worth exploring the value of Deep Extreme Learning Machine (DELM). To understand the DELM, the individual parts of the algorithm must first be explored.

3.6.1 The Autoencoder ELM

The first key in training a multi-layered ELM is to have some way of producing unsupervised inputs for each hidden layer of the ELM. This is done with an Autoencoder ELM(ELM-AE). It works by first encoding the inputs into a feature space, and then reconstructs the original data as best as possible through least square optimization. This generates the parameters of the hidden layers, by repeated process[10]. The whole process can be described as equation 3.7.

$$x_i = [x_{i1}, x_{i2}, \dots, x_{in}]^T \in R^n, N \neq n$$
$$\sum_{j=1}^{L} \beta g\left(a_j \cdot x_i + b_j\right) = x_i^T, i = 1, \dots, N \qquad \textbf{3.7}$$
$$a^T a = I, b^T b = 1$$

The autoencoder lets the algorithm learn unsupervised, meaning it is not constrained to finding a solution. This means it is trained to unknown or hidden patterns in the training data sets. So the key feature that discerns ELM-AE from the regular ELM is that the input weights and biases are orthogonal matrixes, and that it finds patterns and clusters without a set direction[10]. Since this way of programming the hidden layers does not require backpropagation or tuning, it also retains the fast-rendering speeds of the ELM while still adding the ability to increase resolution of noisy and outlier data from the autoencoder[10].

3.6.2 Solving the output layer

The next step in the DELM is to solve the output layer by minimizing the least square mean error. As with a normal ELM, it can be solved by

finding the solution to the problem in equation 3.8.

$$\beta = (H^T H)^{-1} H^T Y, \mathrm{L} < \mathrm{N}$$

$$\beta = H^T (H H^T)^{-1} \cdot Y, \mathrm{L} > \mathrm{N} \qquad \textbf{3.8}$$

$$H = \begin{bmatrix} h(x_1) \\ \vdots \\ h(x_N) \end{bmatrix}$$

Which can be done, as stated previously, by computing the Moore Penrose inverse of the last hidden layer matrix, or as in eq 3.6. if a regularization parameter is used. Another approach would be to treat the last hidden layer as an unknown and introduce a kernel instead[9, 10]. This would be done as equation 3.9[10].

$$h(x_i)H = K_{ELM,i} = \begin{bmatrix} K(x_i, x_1) \\ \vdots \\ K(x_i, x_N) \end{bmatrix}$$

$$\Omega_{ELM} = HH^T = \left[K_{ELM,1} \dots K_{ELM,N}\right] \qquad \textbf{3.9}$$

$$H\beta = K_{ELM}\left(\frac{I}{C} + \Omega_{ELM}\right)^{-1} Y$$

The kernel function K can be chosen among several different ones depending on what the machine is trying to do. This version of an ELM with a kernel is known as KELM or Kernel ELM[9, 10]. Introducing a kernel is generally done when the machine is unsure if the input it is given is incorrect or not and is more common in classification problems.

The last hidden layer would then be the inputs to the ELM, or KELM in the specific case of the DELM[10]. This proposed way of creating a deep ELM is depicted below in figure 3.3.

Each hidden layer acts as the input for the next and is trained sequentially by the ELM-AE. All layers except the last are therefore trained unsupervised while the last layer is supervised[10].

3.7. Historian data retrieval

Wonderware Historian receives SQL queries from clients through its Data Retrieval subsystems, which can locate the requested data and perform the needed processing of the data. In the case of configuration and event data, normal SQL queries can be used since the data is stored inside SQL Server database tables. Historical data must be retrieved from history blocks and then sent to clients as if it is stored in SQL Server tables.

The retrieval subsystem features include support for queries with all tag types, meaning all tag types can be included in the same query when retrieving data from the history table. Any combination of tags can be submitted in a single query. The strings can be of both fixed and variable length; All internal time computation and manipulation is done using the `Win32 FILETIME` type. The Resolution of `FILETIME` is 100 nanoseconds. All times are handled internally as UTC but the conversion to and from local time are handled going in and out of retrieval, so the external interface is local time[11].

3.8. Measuring equipment

To measure the layer thickness at four points perpendicular to each other in the extruder layers, NKT uses Sikora X-RAY 8000 with an addition for NXT. This type is specifically made to use in measuring medium, high and extra high voltage cables with e.g., XLPE as insulation. The measuring accuracy of the instrument is $\pm 15\ \mu m$, but is done on the hot measurements, and the data that is logged is the converted cold measurements of the cable[2, 12].

[The following section has been left empty on purpose]

.

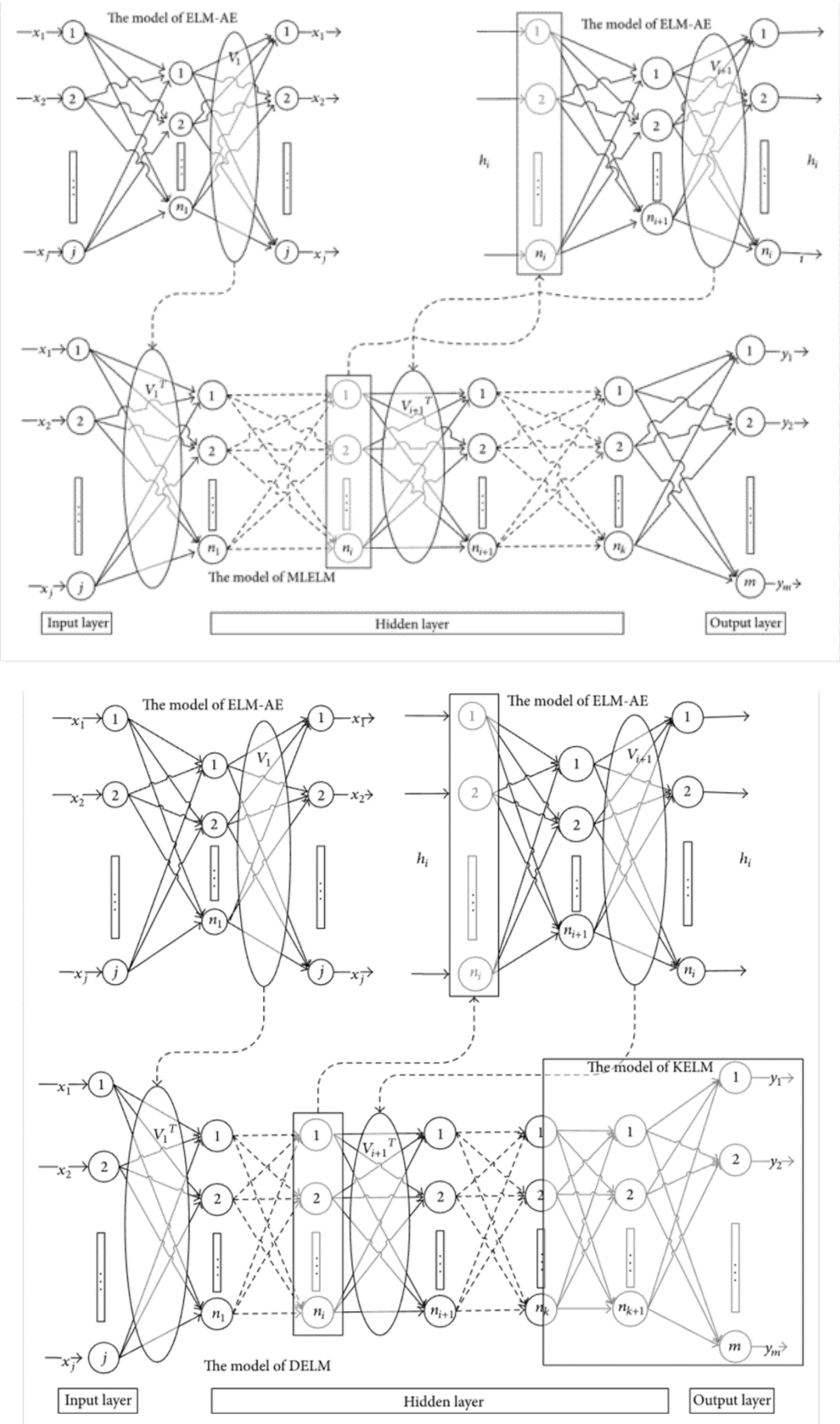

Fig 3.3: Overarching structure of Deep Extreme Learning Machine algorithm and training of Auto Encoders. [10]

4. Project and method

At the start of the project a visit to NKT was conducted where the process was observed up close together with dialog with NKT engineers. This gave a lot of insight into what parts of the project would prove useful, and possible to achieve. The first iteration action plan was therefore altered somewhat in accordance with the new information.

4.1. What is useful?

The initial project formulation described the goals as being a pretty much finished model that can simulate the centering and the thickness. But after the literature study, this was pushed off as a "best case scenario" and a set of new goals were added.

A software that can teach and simplify the operating of the extruder head would provide useful; this would not require the modelling of the full process but only the effect of the tuning screws on the layer thickness. While such a tool would serve little purpose to an experienced operator, it would be a handy tool for new operators to quickly learn how the tuning should be done.

Making a model and proving that it can predict the data and generalize over a greater span with high accuracy, even if the training data cannot be acquired in full would still provide useful intelligence but also a place from where it is possible to continue work on gathering data, that then can be used in training the model. Hence a training tool that can be used by other people with minimal learning, should be designed.

The initially proposed two models, based on physical components and properties as a continuous model equation were abandoned since it was clear that a black-box approach was needed due to the lack of knowledge of the intricacies of the process.

4.2. Process

The extruders that are going to be modelled are triple head extruders, that co-extrude the 2 semi-conducting layers and the insulation layer at the same time onto the conductor.

The initial part of the process involves feeding plastic granules to a hopper that melts the plastic material. They are then tempered and extruded onto the conductor with the three extrusion heads.

The position of the cable in the extrusion head is tuned using in total twelve screws, turned to set the position of the die distributors. The centering is usually done at startup of the process. These screws are not monitored through any input and are analogue in their function[2].

The crosslinking of the polymer is what gives the cables its mechanical properties. It takes place in a vulcanization zone right after the extrusion head. Once the polymer has been crosslinked that cable is set, and there is no way to change the composition of the cable.

4.3. Insulation

The reason for insulation in the cables is to lower the magnetic field interferences. This is done so that a higher transmission voltage can be put over the cable, which will make for less losses in the cable over long distances. Insulation is generally accompanied by Semi-conducting layers to smoothen the resistance gap between the insulation and the conductive layers[2, 13].

4.4. XLPE

The insulation material that is used in the cables is cross linked polyethylene (XLPE). The vulcanization process of the polymer starts with a catalyzing agent, a peroxide (commonly DCP), and a Low-Density Polyethylene (LDPE). The DCP forms a free radical when exposed to heat and steals one of the hydrogen atoms from the LDPE. This lets the LDPE create a bond with other hydrogen sparce LDPE and create a crosslinking structure[14]. The process can be seen below in figure 4.1.

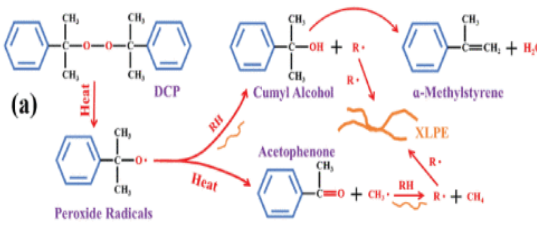

Figure 4.1: XLPE synthesis with DCP and LDPE. [14]

It is important to carefully monitor the extruder screw and die temperatures to avoid degradation of the XLPE, but still keep it in a molten state where it can be extruded at the correct pressure profile in the die barrel and the crosshead[15].

Since the process is catalyzed with a peroxide, the turnover of peroxide can be measured to obtain the degree of cross linking in the cable. If the degree of cross linking is too low, the cable is scrapped[2].

Because of the mechanical effects, shape memory etcetera, of the crosslinked polymer, it is important that the reaction only takes place within the vulcanization zone of the line. If the cross linking happens to early, or in the extruder screw itself, the cable will be ruined. This calls for careful manipulation and control of the temperature gradient through the extrusion line. The number of vulcanization zones also vary between different extruders [2].

The cross-linking process is well understood and the effects of the cross-linking on the layer dimensions and position can be very accurately calculated. Thus, the measurements taken from the melt, right after the extrusion head, can be used to calculate the final dimensions of the semi-conductive and insulating layer.

4.5. Making the software

The first part of making the software is to read the data from NKT's database to the machine kernel. The data is then going to be used to fit a mathematical model that will be used to simulate the process.

NKT uses an SQL Historian database where all the signal data is stored. To get the data from the database a python package called pandas was used and, after the data is downloaded and sorted, it is returned as a pandas `dataframe`. The data was queried from the database using `pyodbc`, which is a built-in function in python.

Since the historian database only registers changes in the process when the change is sufficiently large, data can be sparce with a lot of time between the data points. Thus, the software would need to handle both the raw data and interpolated data.

The actual interpolation of the data is done locally on the database by a function written by NKT. The interpolations are linear but should describe the real data sufficiently.

This is all created in a function called, `getdata` or `getdatainterp` for the interpolated data. These functions build a string, using database tags and time stamps. This is done according to how the local function is designed to pass the inputs. The functions are part of the `Functions` package created for the project.

After the literature study, as previously mentioned, it was apparent that a model formulation based on physical components and material properties would be close to impossible to achieve. A different approach was therefore adopted with a response surface methodology, where different types of polynomials are fitted to the data instead, to see if the dynamic could be described by a polynomial model.

4.6. UI for choosing training sets

As described in section 3, the Wonder Works Historian client does the time conversion and handling of historical data blocks, so a normal query with tag names and time stamps in local time can be used to query the data from the database.

For this an interface called `Training Wheels` was created. It lets the user choose a time stamp according to a template. It also lets the user choose which output tag the model is going to use as a target variable for the training. This means that the model can be trained to each variable at a time. Depending on what type of model is to be fitted, the interface lets the user decide on what parameters should be used in the training. It is depicted below in figure 4.2.

[The following section has been left empty on purpose]

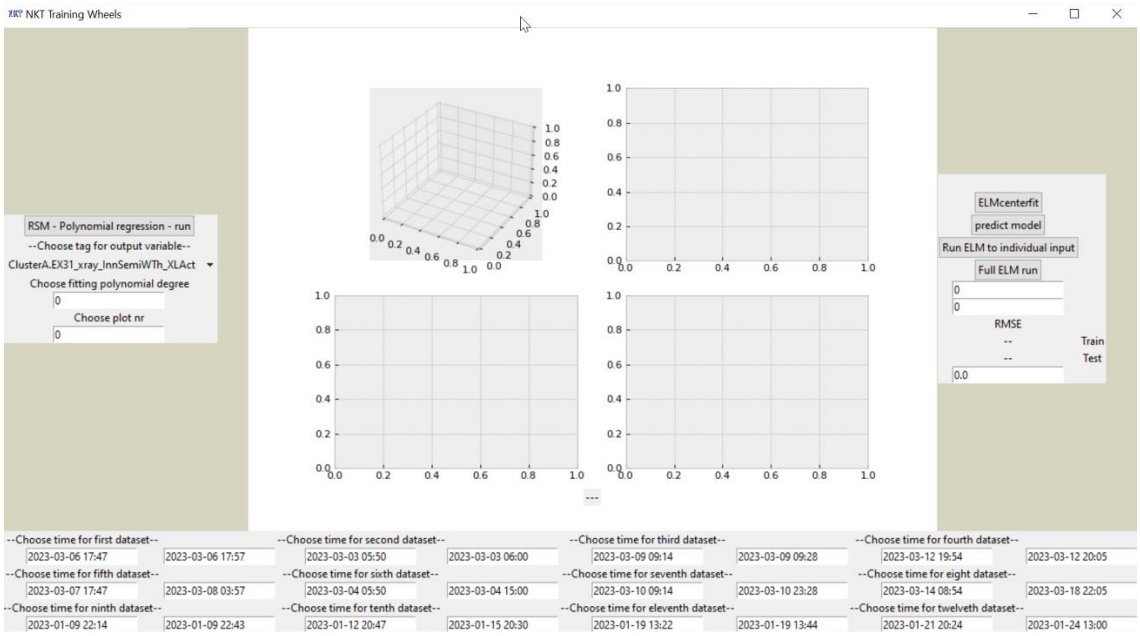

Figure 4.2: Image of Training Wheels interface created for the project.

The plotting part of the interface is there to detect anomalies in data and give a quick glimpse of the prediction and how good the fit is, but it needs to be compared with other metrics to determine model accuracy. During the training, the predictions of the model are plotted together with the actual logged output of the process. The feature is only meant to give the user a first grasp, and to understand the fit, the RMSE of the model needs to be looked at as well, together with an analysis of the data sets.

The UI also gives the user an option to fit all the parameters of the model at once. While running, the UI updates the user on where in the training process it is by telling the user which output it is training to and if it is in the process of reading data from the database or calculating the least square fit.

4.7. Polynomial regression model

The polynomial regression model was introduced to see if such a model could accurately describe the, straight forward, but complex dynamics of the extruder. The model was implemented using linear regression as a base.

First the fitting data was transformed into polynomial feature form according to equation 4.1.

$$x = (x_{1i}, \dots, x_{ni}), i = 1, \dots, M$$

$$C = \begin{bmatrix} 1_1 & \cdots & \prod_{i=1}^{M} x_{1i}^{d_i} \\ \vdots & \ddots & \vdots \\ 1_n & \cdots & \prod_{i=1}^{M} x_{ni}^{d_i} \end{bmatrix}, s.t. \sum d_i = d \qquad \textbf{4.1}$$

Where x_i is the input, and i represents the nr of the input for M inputs. C is a polynomial matrix that contains the evaluation of each term in the polynomial based on the inputs, for every set of inputs i. d is the degree of the polynomial.

This transformed data is then split into training sets and test sets. It is done to try to avoid an overfitting problem. The training set is used to fit coefficients of a polynomial of degree d according to equation 4.2. This was done using the `SKlearn` function `PolynomialFeatures` together with `fit_transform`.

$$P = (c_1, \dots, c_j)$$

$$PC = \begin{bmatrix} c_1 1_1 & \cdots & c_j \prod_{i=1}^{M} x_{1i}^{d_i} \\ \vdots & \ddots & \vdots \\ c_1 1_n & \cdots & c_j \prod_{i=1}^{M} x_{ni}^{d_i} \end{bmatrix}, s.t. \sum d_i = d \qquad \textbf{4.2}$$

Where j is the nr of polynomial terms and P is a vector with the coefficients of the polynomial. The model used to fit the polynomial is a linear regression model function called `LinearRegression`.

4.8. ELM model

The ELM was built manually in python and training sets were constructed with the help of `SciKit-Learn`. It was implemented as described in equation 3.3, with iteration over the number of hidden nodes that would best fit the data while avoiding over-fit. The learning process was a Mean Square Error minimization problem, where the data sets were split up into training sets and test sets, with a ratio of 70:30, and tried over different random initializations of the network; The optimal number of hidden nodes varied for each output signal.

The resulting ELM model consists of a set of input weights, output weights and biases for each output that is being measured in the extruder head. An illustration of the model can be seen in Fig 4.3.

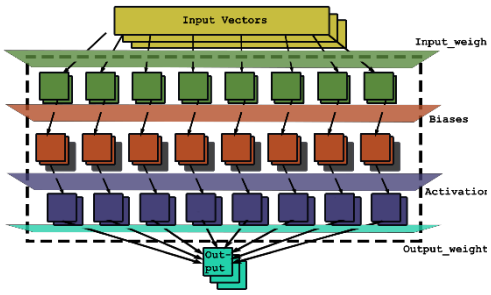

Figure 4.3: ELM model structure.

The model was trained to a few different data sets with a single time stamp to see if the training algorithm worked. The resulting model is described in equation 4.3.

$$\begin{aligned} x_{ce} &= [x_{ce1}, \dots, x_{ce12}]^T \\ x &= [x_1, x_2, x_3, x_4, x_{ce}]^T \in R^n \\ h &= g(w_j \cdot x + b_j) \\ \sum_{j=1}^{L} \beta_j H &= o \end{aligned} \qquad \textbf{4.3}$$

With x_{ce} representing all the inputs from the twelve centering screws and x the complete set of inputs.

The model output weight is computed analytically using the Moore-Penrose inverse, seen in equation 4.4. The implementation of this in python is done using the `scipy` function `Pinv`. Depending on how the network is initialized, it can generate different output matrixes β, which does seem to influence the generalization capabilities of the model.

$$\begin{aligned} H^{\dagger} &= pseudo\ inverse\ of\ H \\ H &= g(w_j \cdot x_i + b_j) \\ \beta &= H^{\dagger} Y \end{aligned} \qquad \textbf{4.4}$$

Where Y is the training set outputs values of the process, for each input set i. H is a vector containing the value of all transformations for all inputs and hidden nodes.

The model is then written to text files that store the output weights, input weights and the biases. This is done using the function `savetxt` in python. Since the model is written as strings, it needs to be reverted to a pandas `dataframe` when calling the model. To achieve this, the strings are split at select points and sorted into a list, the list is then converted into a `dataframe`. Each string is then converted to a float using `float`.

The reason the ELM was favored in this project over a classical back-propagation algorithm with gradient tuning is due to the lack of time and need for fast training.

4.9. A model of two parts

While working on the model, it became apparent that it would be difficult to obtain any data at all from the centering screws. So, to be able to continue the project a new direction was taken. To be able to train the model, while lacking critical data from some of the inputs it was divided into two parts.

The first part of the model is only trained to data sets where there is no centering done. It is basically a model that only describes the output as a function of the three screw motor speeds and the line speed, in total four inputs.

The second part of the model would be describing the output as function of the centering screws, or the angles between the conductor and the distributors in the extruder head.

These two parts could then be combined into a single model, describing the full extruder head. This would be done by letting the output of the first part be part of the input of the second part. The model can be seen in figure 4.4.

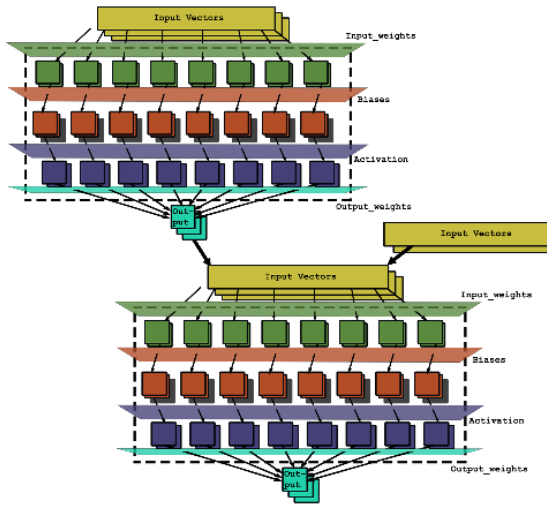

Figure 4.4: Illustration of the full model, combined from two separatley trained models.

The postulated upside with this approach is that, even if one part of the model works poorly, the other one would still work independently of the other, since they would be trained independently. And both parts could be simulated and evaluated separately, since they are each technically a full model, some of the inputs are just unknowns. It would however require the extruder to be run at the specific conditions that fit each part of the model to simulate it as one unit.

So, to conclude, the centering part (second part) of the model will only work in the linear regions of the motor and line speeds that the first part of the model works on, and vice versa. The window of where this working region for the second part of the model is, is unknown in the current training algorithm and the two parts can't be combined unless it is.

4.10. Finding the first training sets

The difficult part of this project was to find the appropriate data sets to use as training sets for the model. The following idea was formulated to find good enough data sets for the first part of the model, and the shortcomings they will have, to still be able to evaluate if the model works well enough.

4.10.1 Data acquisition first part

Depending on what type of cable is produced in each extruder, different tools are used for the extruder heads. These tools might influence the process, but little is known about the specific effects. Because of this, the test sets should include at least one data set that has a heterogeneous tool set, or it will most likely be slightly over-fitted to a specific toolset. Another way of getting around the problem is training the model to only homogeneous sets and create several models for different setups. Alas, these toolsets are not a part of the model and therefore, all the data sets need to belong to the same type of cable; the specifics of the cable and the properties might vary with the cable type. This should be done to avoid model errors due to an input (toolset) that is not captured by the model.

The reason that the toolsets are not included as inputs is because of time constraints and access constraints. It is, however, something that could be added if the data is accessible. Worth noting is that the model might fit the data quite well from different tool sets, even though it doesn't recognize it as a parameter, if the difference in output dependance on each tool set is small.

Since the data that is logged on to the database doesn't include the manual centering screws, each set must only contain a period where there was no centering done at all. This should not be too hard to capture since most operators only start centering after the line is running at operating conditions. This can be double checked by plotting the dataset to see if there is collinearity with the motor speeds.

Thus, the time stamps were initially chosen to a line start of the process. A line start is where they fire up the extruders and increase the speed of the motor and line to the operating point.

However since the process doesn't completely stabilize until a couple of hours after the start[2], it also introduces a new constraint on the data sets used in training. Initially it was postulated that this dynamic would be captured by the model as it was trained to line starts only, however since

it is dependent on both parts of the model, the training sets for both parts would need to be from the process when it is stable and when it is not, since the model won't know the difference in the unknown input, it just knows there is a difference. But the data still needs to be taken from a run where no centering has been done, which means a data set where no centering has been done for the full duration of the line would need to be found. This is neigh impossible, since it can't be seen by plotting it, since the collinear relationships method fails as noise together with the slow dynamic of the extruder makes it hard to see if there is any correlation.

The last difficulty is to capture the full effect of different base settings. A base setting will be defined here as the state that the extruder head is in as the process is started. This entails the centering screw positions and distributor toolset used. To make this happen, each training set should contain a different base setting. This is very hard to achieve since the data is not logged anywhere or noted by the operators. For example, in one case they might keep the settings they had on the previous run because it already had good centering, and in some cases the head has been taken apart between line starts and has a different setting[2]. This means that the model might have a hidden bias towards a certain setting which will later turn up in validation.

One con in only getting the data from successful runs is that each cable has a "recipe" which is used to get the right cable dimensions. This means that the model only works in specific linear combinations of the motor and lines speeds if it doesn't generalize well enough. It is a problem if the model should be used to simulate changes outside of these linear ranges.

So, if data sets that follow all these criteria can be found, it should allow the first part of the model to work independently of the second and simulate the model in a specific working range which would allow it to also be validated.

4.10.2 Data for second part

The only data available for the second part that is extensive enough(barely) to use as training for the second part of the model is an old test run made in an old job, so the model will have to do with that.

4.10.3 Trend

The program used to find the time stamps for each data set that is going to be used in training is called Trend and the interface is shown in figure 4.5.

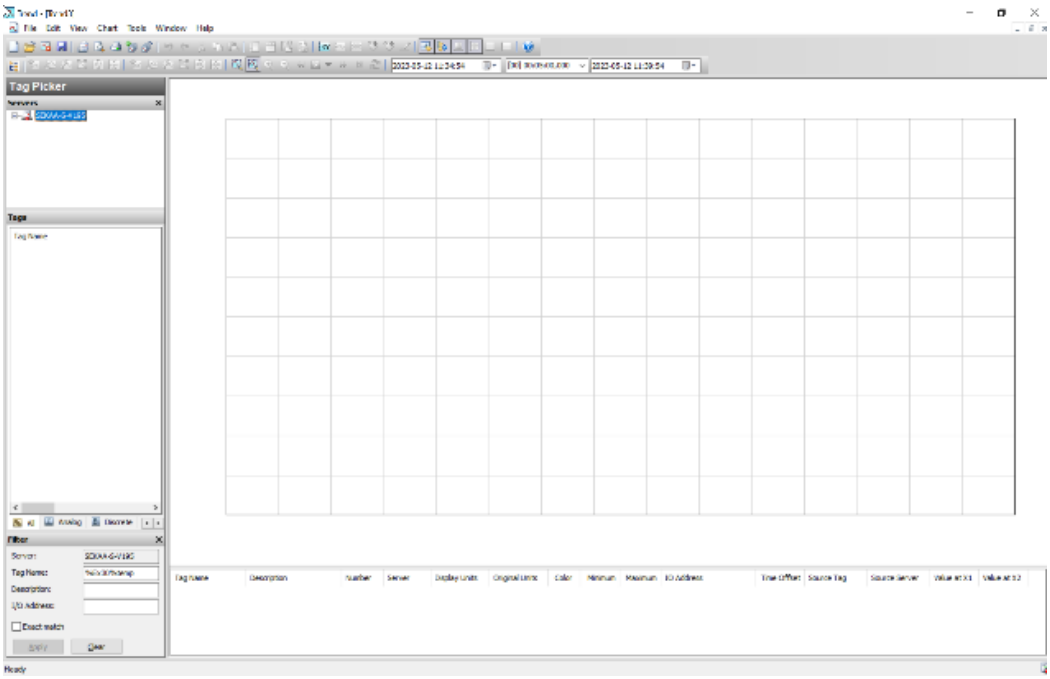

Figure 4.5: Trend interface used to find data sets.

The data is plotted as a line graph with a normalized value on the y-axis. Each tag is displayed in its own unit.

4.11. The optimal model

The optimal approach, when training the neural network would be to train the whole model simultaneously to all the input variables. This would create a much simpler learning algorithm and it would be much simpler to implement. It would also be easy to implement a discrete time derivative in the training sets as well, to better capture the dynamics of the system as it is simulated.

4.12. Simulating the model

When the model is trained and ready, a simulation GUI can read the model from the text files and save it. The model is then passed to a response function together with inputs from the simulation GUI, where the motor speeds, line speed, and the centering screws can be adjusted. The response function solves the model and returns the model output to the simulation GUI and plots it.

To view a sort of instant response of the system, this loop is then animated to generate time invariant responses to the inputs, and plotted vs a timeline. This means that the timeline does not show the actual response time of the system only the input time, since the real system is slow, and has delayed reactions to the inputs being given. It shows the slow input response instantly, which can be crucial if the model is to be used as an operating aid in the future and fast inputs are needed to not go below nominal layer thickness.

The model can also be simulated in discrete time by returning the output of the model as an input. The model then needs to be trained to the discrete time derivative, by shifting the training vector by one step, making each input vs output response a discrete time derivative of the time between each measurement. This shift is described in equation 4.5.

$$x_i = \begin{bmatrix} x_{1,1}, x_{1,2}, \dots, x_{1,n} \\ \vdots \\ x_{N-1,1}, x_{N-1,2}, \dots, x_{N-1,n} \end{bmatrix} \in R^n$$
$$t_i = \begin{bmatrix} x_{2,1}, x_{2,2}, \dots, x_{2,n} \\ \vdots \\ x_{N,1}, x_{N,2}, \dots, x_{N,n} \end{bmatrix} \in R^m \quad \textbf{4.5}$$
$$T = 30s, N \neq n$$

Where T is the real time between each data point and represents the discrete time step. It must be noted however that to get such a discrete time response the training data needs to be interpolated to generate equal time steps.

Since the model is not trained to a time variable, it would only be possible to show the actual time response of the system if a time variable was added to the model, to keep track of the measuring delay. This is not possible since it would require a much better understanding of the process and how it varies in real time, and equations of the dynamics formulated.

The full GUI exists in three parts. The first part is the training environment, where the model can be retrained to different data sets. Then there is the simulation of the Xray signals as a line graph, with the nominal values of the layers. Then the last part is a centering UI which shows the cross section of the cable and how it changes with the extruder inputs.

The purpose of the line graph simulation is to show a clearer image of how far off from the nominal value the extruder output is, to give a more intuitive picture of how good the layer thickness is in the cable. An image of the line graph simulation is shown below in figure 4.6.

The centering GUI calculates each layer cross section based on the values of the simulated Xray signals. This means that it won't show an accurate representation of what the cross section looks like, but rather show the trend of how it changes when inputs are changed.

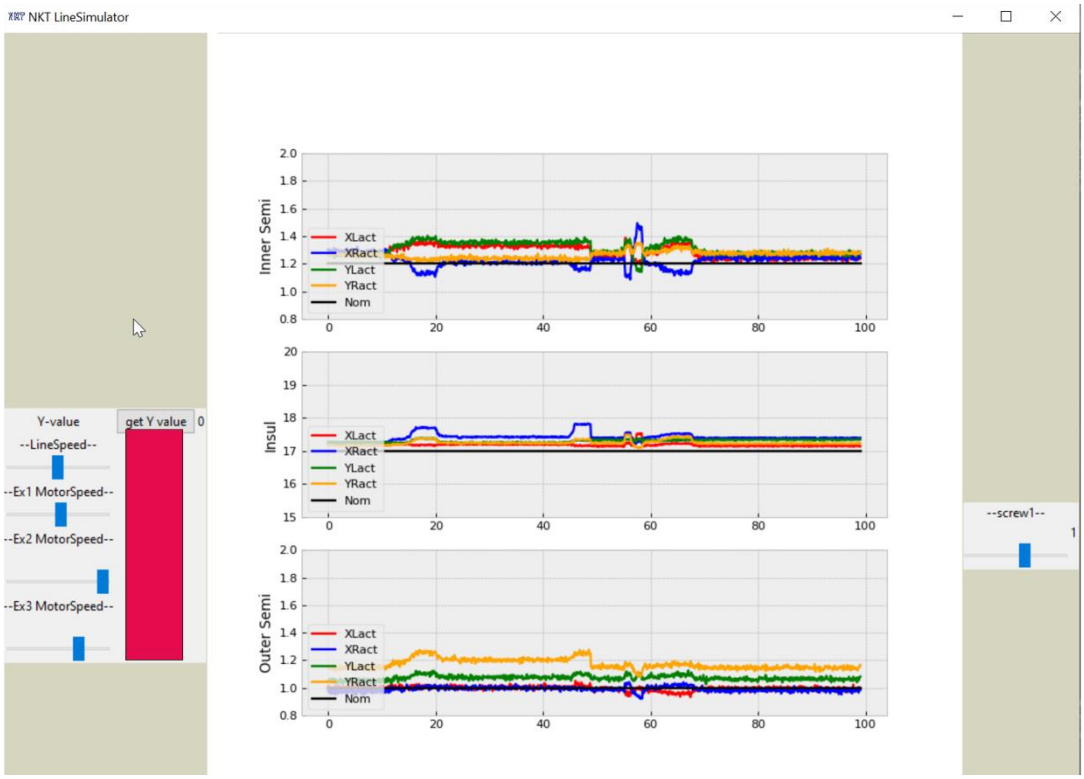

Figure 4.6: Line graph simulator of layer thickness Xray measurements. This is an unfinished itteration. The motor and line speeds have been blocked due to confidentiality.

4.13. Cross-section graphics

While it, at first glance, might not seem like too big of an issue, it turns out that it is quite difficult to show a graphical representation of the cable cross sections, by just going off 4 values. The question then becomes, how can the cross section of each one of the layers be calculated from only the 4 points of measurement that is available and still show an intuitive picture of how the conductor moves within the cable.

First an approach where a circle was divided into 4 parts; each part is a quarter slice of the circle calculated by making a circle section between each combination of nearby data points. The data points in question are the 4-layer thickness values represented as coordinates in the plot. This approach didn't really give an intuitive picture of how the centering of the layers around the conductor moves as the layer thickness varies.

So instead of drawing the circle sections as mentioned above, the circle was first drawn as a perfect circle based on the average radius of the circle assuming it is completely symmetrical. Then it is distorted by the ratio between the actual distance from the center of the perfect circle to the measured point in each direction, and the radius of the prefect circle. This is done for all four of the data points. Finally, the circle is moved on the plot so that the center of the circle is where the center of the perfect circle would be if the data points of the perfect circle would align with the actual data points, basically a moving oval.

Along with this, small text windows are placed at roughly the point of each measured output in the circle, which show the value of the output. Unlike the line graph, this figure is animated at a set interval together with a discrete time signal and shows the discrete time response of an input change but speed up to about 30 times the real discrete time step. It can be seen below in figure 4.7.

The screw inputs are in the real process tied together and the corresponding backside screw should be changed to match its frontside screw. The position of each extruder is shown as well to give a better view of where the cable is in real space.

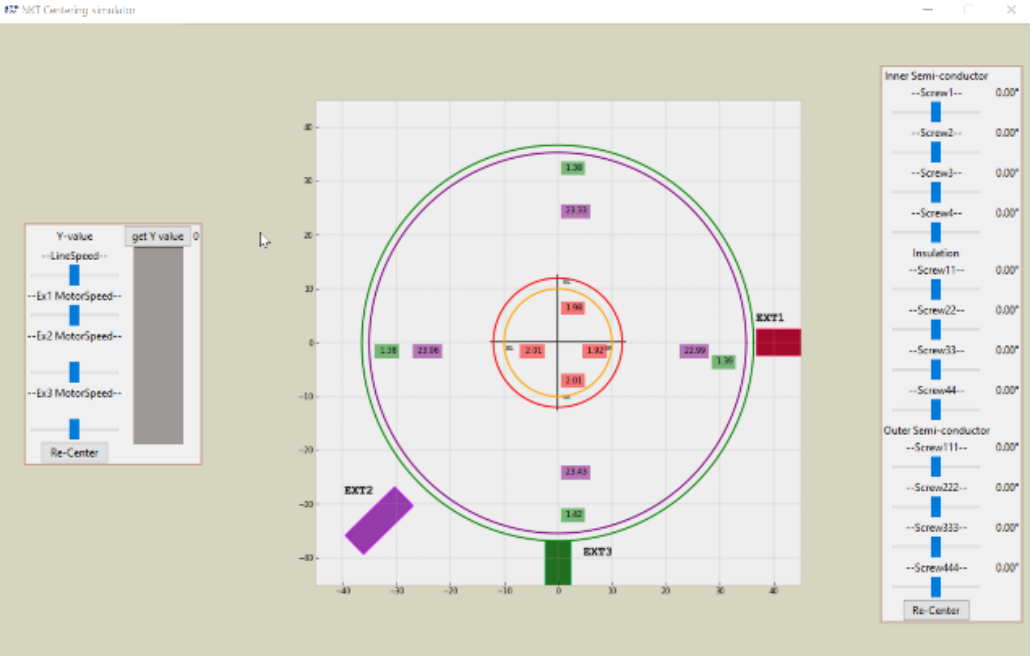

Figure 4.7: Image of the centering simulation tool, the line speeds and motor speeds have been blocked from view due to confidentiality.

4.14. Improved model-robustness

To further increase the robustness of the model, to reduce error propagation and increase the generalization, the model was expanded to include a robustness parameter also known as a regularization parameter. This method is known as ridge regression. Since the Moore Penrose pseudo inverse can no longer be used to calculate the least square, a function calculating it had to be added. The function calculates the least square solution and thus the semi-inverse of the dual optimization problem according to equation 4.6.

$$H^{-1} = \begin{cases} \left(\frac{I}{C} + H^T H\right)^{-1} \cdot H^T, L < N \\ H^T \left(\frac{I}{C} + HH^T\right)^{-1}, L > N \\ H^{-1}, L = N \end{cases} \quad \textbf{4.6}$$

Where C is the regularization parameter. This approach forces the model to not overfit the training data, by giving a penalty to the size of magnitude of the output weights.

While the model is now quite robust for the data fitting, it still can't solve the issue of underlying data patterns and with the second part taking outputs and returning them as inputs for the next iteration of the process simulation, this will still give cause to error propagation when the model receives outlier inputs outside the range of where it is trained, since the inputs are unknown. To solve this the model is once again expanded, this time into a deep learning algorithm.

4.15. Deep learning dual ELM

The deep learning model is built on the theory of ELM and takes inspiration from the Deep Learning Extreme Learning Machine. The model uses a total of six hidden layers where four are trained unsupervised and two are trained supervised. Once again, the direction of utilizing a non-propagating training algorithm is due to a lack of time and the need for quick retraining. The hidden layers were introduced to help cluster inaccurate data in the second part.

The overarching structure of the model is the same as the first model, with two independent parts. And to combine them. it would then take the output of the first part as part of the input in the second part just like in the first model iteration.

To train the unsupervised layers, an autoencoder ELM is used. It is implemented in the same way as described in equation 3.7. To create the autoencoder input layers `SciPy's` function `ortho_group` is used. To create the biases, a gaussian random matrix is created and each

element is squared. Each element in the matrix is then divided by the norm of the vector. Lastly to regain the initial distribution, each element is square rooted, and the sign of the original value is restored. The output layer is then calculated with equation 4.6. Each layer is trained to the output of the previous layer according to equation 4.7. The unsupervised layers are all but the last, for V number of layers.

$$\begin{aligned} &H_i = a_i \cdot X_i + B, for\ some\ input\ vector\ X \\ &X_i = vH_i, i = 1, \dots, V-1 \\ &X_{i+1} = gvX_i \end{aligned} \quad \textbf{4.7}$$

Where g is the activation function, which is a standard rectified linear unit function, B is the bias, v is the weights and a is the AE input weights. The last layer is trained by an ELM with a regularization parameter. The resulting model is described in equation 4.8 and shown in figure 4.8.

$$\begin{aligned} &H_V = X\prod_{i=1}^{V-1} v_i g \\ &Y = \beta \cdot K(H_V) = f(X_1, w) \\ &w = [v, K, \beta] \end{aligned} \quad \textbf{4.8}$$

Where X_1 are the inputs to the first layer, and Y is the output of the last layer. v are the weights and K is the ELM feature map obtained in training of the last layer, represented by the green, orange, and blue layer in figure 4.8.

The combined model can be described mathematically as equation 4.9 and is depicted in figure 4.9.

$$\begin{aligned} &Y_{1,part1} = f(x_{1,ij}, w_1) \\ &Y_{2,part1} = ff(Y_{1,part1}, x_{2,ij}, w_2) \end{aligned} \quad \textbf{4.9}$$

The computation time and learning time for the deep version is almost as fast as the regular ELM.

With this model a thought experiment was conducted. If the first part of the model is trained to something that is completely independent of the second part of the model, theoretically it should be able to reproduce the inputs of the first part, motor- and line speeds from the experiments used in training the second part. This would let the full model be tested as well, instead of just being a postulated idea. Thus, it could be trained to the amount of plastic extruded per unit length of cable pulled, which should be independent of other parts of the extruder.

The reason such an experiment should be possible is, now with an unsupervised part of the model, it should detect irregularities in the linear pattern and correctly cluster them, to increase the accuracy of the output from outliers and noise. Slightly inaccurate predictions from the model should therefore suffice in training. So hopefully the first part of the model can be trained with data from the theoretical "recipes" that is used to achieve certain layer thicknesses together with bleed runs and, in that way, generate data outside of the normal runs, thus eliminating the need for test runs with the full cable. This model could then be used to reproduce the unknown data for the second part of the model, thus making the combination possible. Due to lack of time, however, this remained as nothing more than an idea.

[The following section has been left empty on purpose]

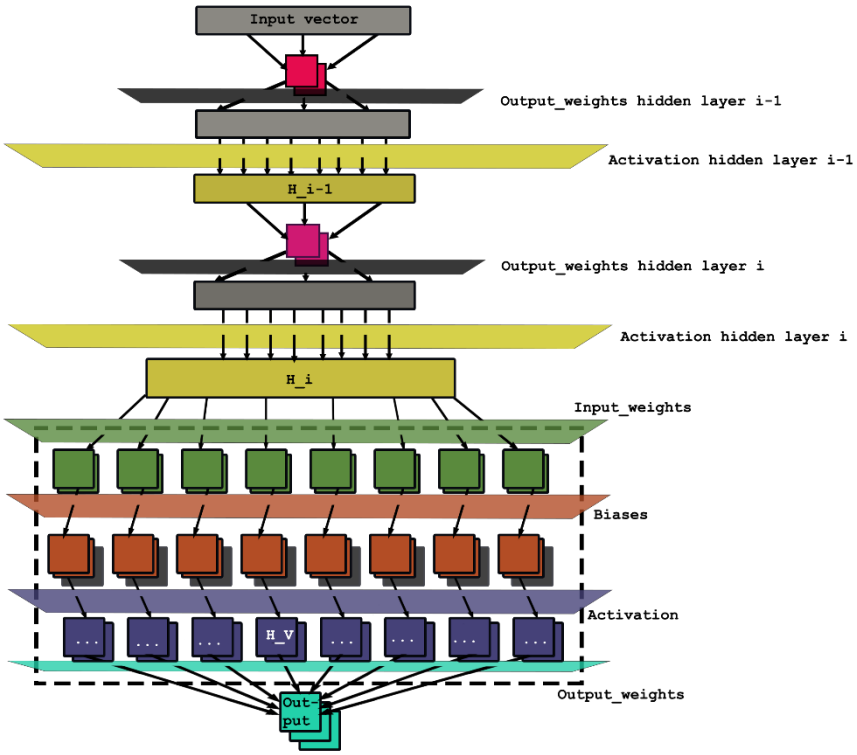

Fig 4.8: Single thread of the model, with 3 hidden layers.

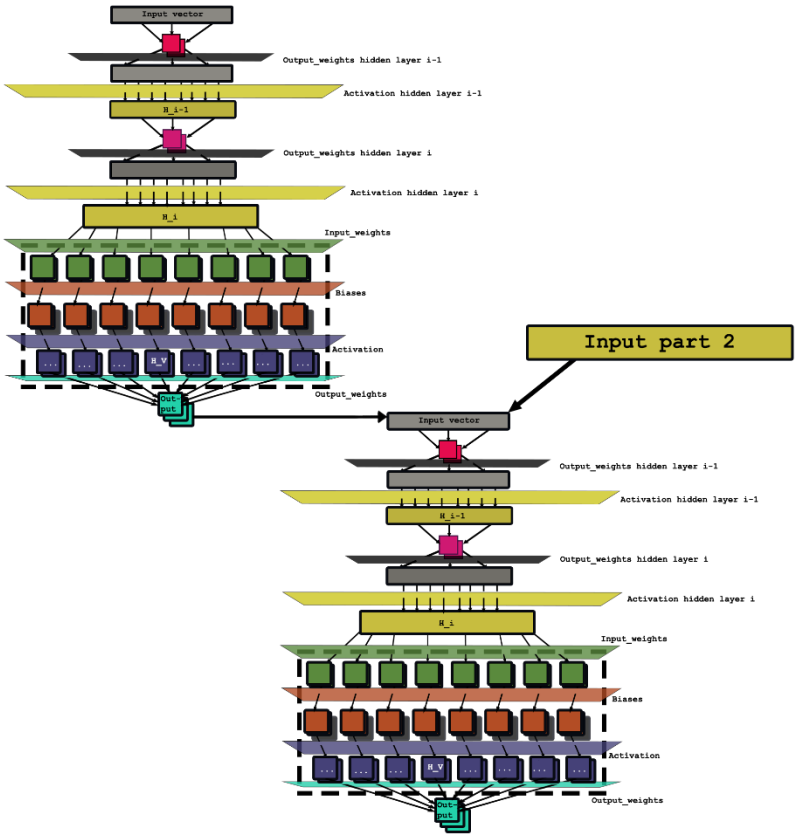

Fig 4.9: Combined model description.

5. Model results and implementation

The first model, the polynomial model, had a good fitting capability to the data that it was trained to, but ran into an issue of slight over-fitting. An acceptable standard of Root Mean Square Error (RMSE) was set to two times that of the RMSE for the train sets as an initial target base, but equal to the train set RMSE would be optimal. The initial testing performance of the polynomial model is shown in figure 5.1 to a made-up test set of data based on motorspeed1, with only one input and one output.

While the RMSE is quite low on the training set displayed in the picture, it can clearly be seen that it is off trend at several points in the prediction, which means that it is overfitting too much to capture the process dynamics accurately enough. At first it was hard to understand why the polynomial model was unable to reach an acceptable level of generalization.

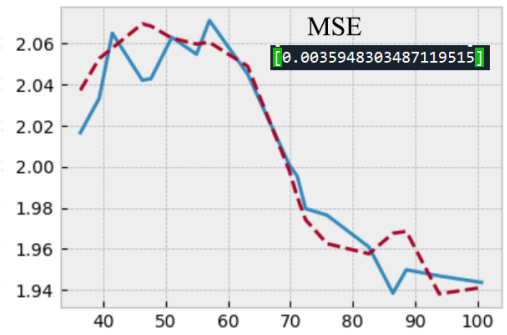

Fig 5.1: Polynomial model prediction for the inner semi-conductor layer test set. Blue line is the model prediction and red line is the actual output. The values on the axis are transformed values, the MSE value is the relative value between the test and training set.

A theory was that maybe these types of models are more sensitive to the training data than, say, an ELM, since not all the independent variables that are affecting the extruder output are included in the training sets. It may also be the case that it is more sensitive to outliers than the ELM. For reference the same test-training set was given to the ELM, and the results can be seen below in figure 5.2.

It also seems like the overfit increases with the number of inputs. A trend like this can be seen when increasing the number of inputs that is given to the polynomial model to fit. When comparing the testing RMSE to the training RMSE a trend, where the larger the training accuracy is compared to the test accuracy, the more the model seems to have overfitted the data. All of this is consistent with the theory from section 3. This is displayed in table 5.1 and shown in figures 5.3, 5.4 and 5.5.

These images show why it is important to not only look at how well the prediction accuracy is, because even if it is good for both the training data and the test data, it might be the case that it has overfitted the training data. The mean error on the test set might then be low because of e.g., measuring noise that happens to weakly correlate with the prediction, but the actual input to the real value lacks correlation. This can be identified by looking at the scatter plot or line plot to see if there is correlation or not. A model with poor robustness can act as a high pass filter amplifying high frequency signals, which causes a severely unstable model. So, the relative accuracy of a well-chosen test set always needs to be compared to the training accuracy.

The opposite might also be the case, where the model is underfitted. It then acts as a low pass filter with too low of a breakthrough frequency, then it filters out important changes and causes the model to not respond at all to inputs that it should respond to.

[The following section has been left empty on purpose]

Table 5.1: Extruder predictions with 4 inputs.

	Deg	Training MSE (1)	Test MSE (1)	ΔMSE (1)
Polynomial model				
Set 1	**3**	0.010	0.044	0.034
Set 2	**3**	0.011	0.024	0.013
Set 3	**3**	0.004	0.048	0.044
Set 1	**2**	0.004	0.015	0.011
Set 2	**2**	0.010	0.026	0.015
Set 3	**2**	0.003	0.009	0.006
	Nodes			
ELM model				
Set 1	**100 n**	0.005	0.005	0.000
Set 2	**1000 n**	0.011	0.011	0.000
Set 3	**1000 n**	0.002	0.003	0.001

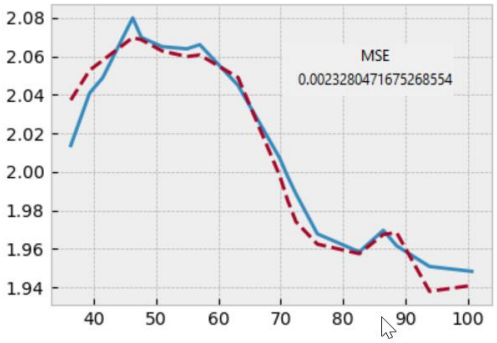

Figure 5.2: ELM model prediction for the inner semi-conductor layer test set. Blue line is model prediction and red line is the actual output. The values on the axis are transformed values.

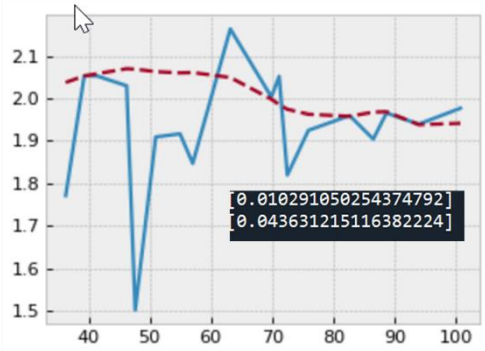

Figure 5.3: polynomial model prediction for the inner semi-conductor layer test set. Blue line is model prediction and red line is the actual output. The values on the axis are transformed values.

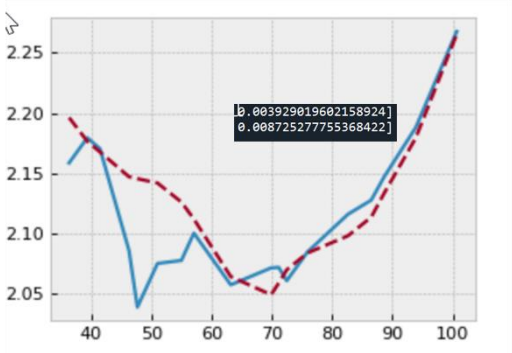

Figure 5.4: Polynomial model prediction for the insulation layer test set. Blue line is model prediction and red line is the actual output. The values on the axis are transformed values.

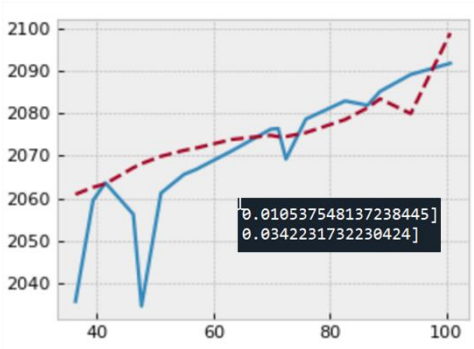

Figure 5.5: polynomial model prediction for the outer semi-conductor layer test set. Blue line is model prediction and red line is the actual output. The values on the axis are transformed values.

When analyzing the data, it is clear that there is multicollinearity in the data sets, caused by noise and how the line is started, which is most likely the cause of why the polynomial model is not working. Of course, there might be better ways to implement this type of model, but at some point, a choice of model must be made, to finish the project on time.

So, in the end, the polynomial model doesn't measure up to the ELM, which even at a single extruder motor speed signal, gives better generalization than what the polynomial regression model gives, and the gap grows as the number of inputs increases. The accuracy of the test set is also a lot better for the ELM. Thus, it was decided to not take the polynomial regression model any further and focus purely on machine learning with neural networks.

When expanding the training sets to multiple runs of the extruder, the generalization of the ELM stays quite good if nodes are chosen carefully. The more homogeneous data sets with regards to which cable is produced; this probably means

centering screw positions and tool sets are the same; the lower the relative RMSE can go at the cost of overfitting. But if one data set is much larger than the others, it will also cause a type of "fake" fit, since the error of the other data sets are diluted, but still retain the overfitting. This is obviously very bad. An example of this is shown in figure 5.6.

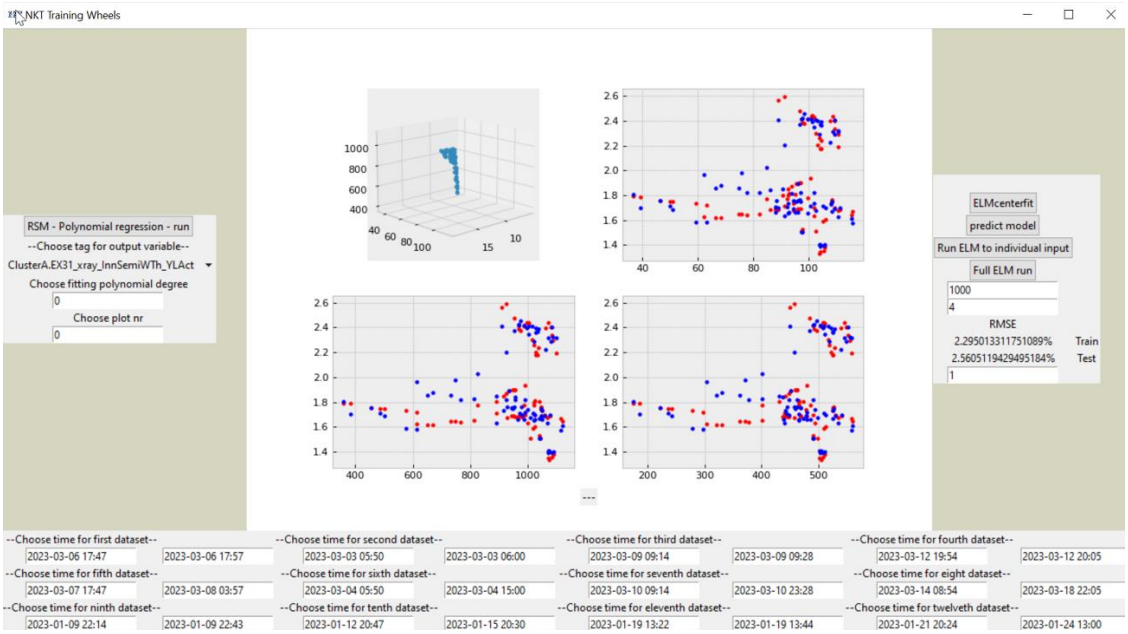

Figure 5.6: Results of training for one Xray signal. Blue are model predictions and red are the measured response. Top left shows the extruder operating inputs.

As can be seen in the three-dimensional graph in the top left, all the data points contain the same type of linear run and cable product.

The generalization is good when looking only at the RMSE values, but when looking at the plot, a subset of the data has been overfitted due to it containing way more data points than the other ones. This can be seen by looking at the scatter plot and how it seems to have weak to no correlation at the lower speeds, which is not what is expected. In this specific case the data set containing data from the lower motor speeds only contains a fifth of the data points of the set containing the higher speeds. Because of this, it is important to choose data sets of not too much different sizes, so that the fitting capabilities can be read from the RMSE values. This could in theory also be solved by adding a penalty term to larger data sets. The bottleneck for the model, even when only looking at the logged inputs, is in this specific case, still to a degree the lack of extruder screw input data.

It's worth noting however that extreme caution needs to be taken when choosing the number of hidden nodes in the regular ELM, since it seems to be very sensitive to the number of nodes. This becomes more of a problem as the data gets more nonlinear, noisy, and inaccurate as in the case of the distributors. The regular ELM is therefore not enough to describe the centering and thus should not be used.

5.0.1 The multicollinear pattern problem

As can be seen in figure 5.7. The extruders are always started in a linear way, so historical data from line runs doesn't train the model how to operate outside of these linear settings if there are non-linear deviations because of multicollinearity. Since the model is only supposed to work in a range close to the operating range, it might not be a big issue. But it is important to remember that the performance of the model would most likely be better even in the operating range with a broader range of data sets.

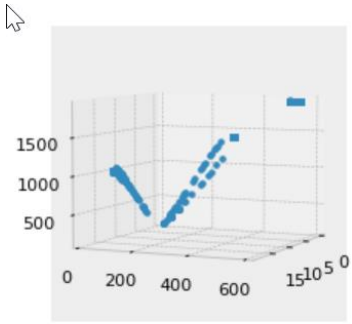

Figure 5.7: Extruder line input data plotted against each other. The multicollinear properties of the data set is obvious, each "line" is a separate cable produced.

The data that is being logged might also be better if it was taken as hot measurements when it passes the X-rays. Otherwise, variations in the post-extruder part of the line might affect how the model performs.

5.1. Ridge regressor model

When introducing the ridge regression with emphasis on minimizing the output weight norms, according to bartlett[8], the generalization as well as the model accuracy should improve even for asymmetric training data. This is empirically proven when looking at the same data sets as in figure 5.6, but with the penalty term added; the model performs a lot better which can be seen in figure 5.8.

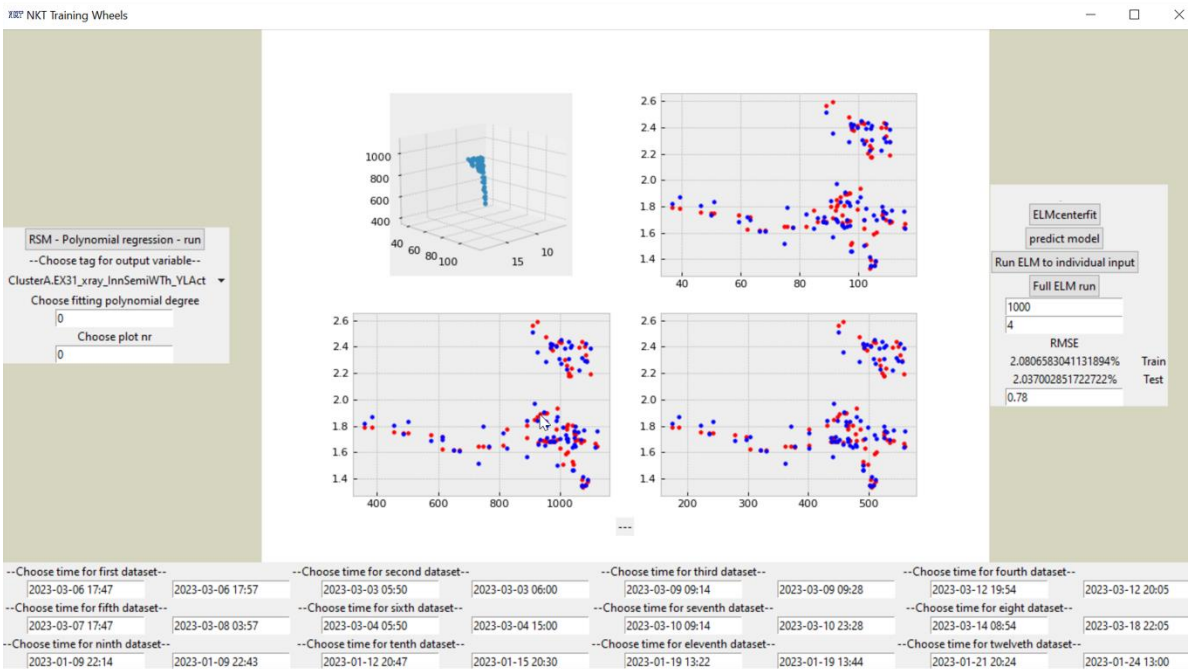

Figure 5.8: Results of training for one Xray signal. Blue are model predictions and red are the measured response. Top left shows the extruder operating inputs.

The RMSE may only seem slightly better at first glance, but, with true weights to the data points it is most likely much better.

When the number of data sets increases, the nature of how operators and control hardware control the extruders create a lot of noise and errors due to how the centering of the extruder is set; they basically create known unknowns for the inputs that aren't part of the training sets; Also, when utilizing data from several different cable types, which means different tool sets in the extruder head, it creates a difficult learning environment for the model. This is partly because of unlabeled inputs but also since multicollinearity is present combined with the number of data points approaching two-hundred thousand. With ridge regression, the latter turns out not to be that big of a problem, but the former requires carefully chosen training sets.

This is demonstrated when giving the model several runs of different cables, where, most likely, all the runs pertaining to the same cable

produced, should have the same settings on the centering screws; this conclusion is drawn from looking at the trend and might not be a hundred percent accurate. So, all the runs from cable A would have the same settings, and all the runs from cable B would have the same settings. But A and B might have different settings to each other. Such an experiment is shown in figure 5.9.

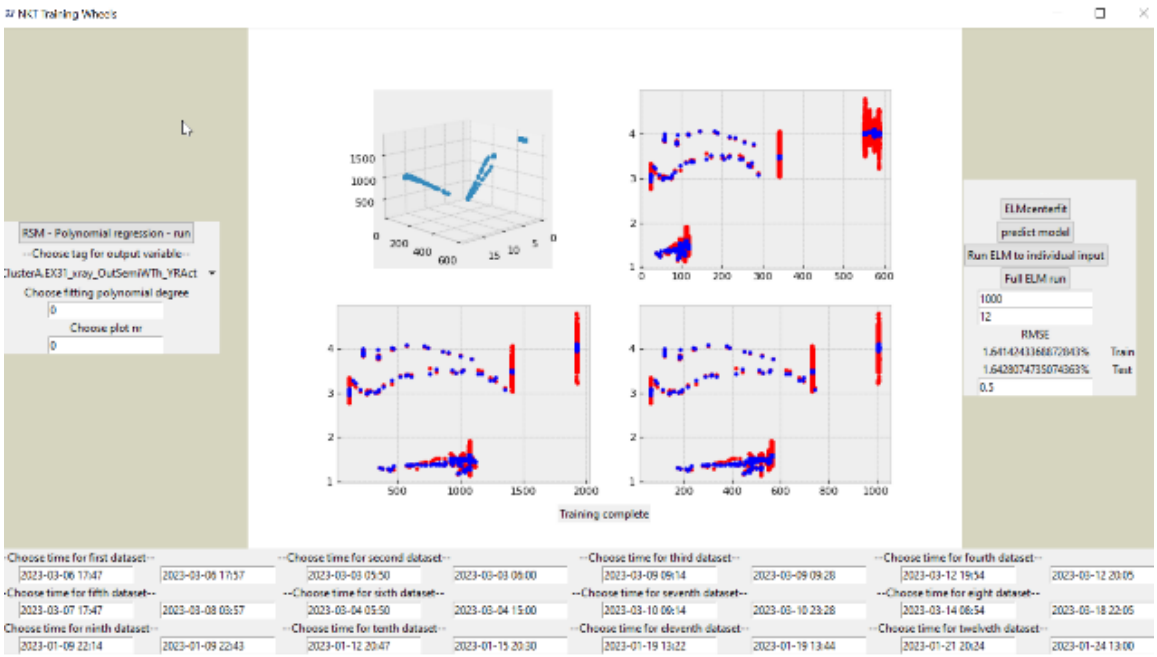

Figure 5.9: Results of training for one Xray signal. Blue are model predictions and red are the measured response. Top left shows the extruder operating inputs. The data sets used are from different line runs.

While, in this experiment the model just attributes these differences to the different motor and line speed combinations shown in the top left graph. It proves that if a label were to be added for the screws, the model would be able to accurately label and predict these inputs as well. What can also be seen is that the model filters out measurement errors and noise in the continuous part of the process where the output seems to follow a random distribution. An index label can also be added to represent these unknown screw positions.

An interesting observation is that it does not seem to matter if the data set chosen has centering or not. The only difference seems to be that around the point the centering was done, a local cluster of predictions with larger error than the mean is found. In a way the model seems to just assume that these points are inaccurate, which may be convenient. Why this happens is unknown, but it might just be that the datapoints affected by centering are too few. This can be seen in figure 5.10.

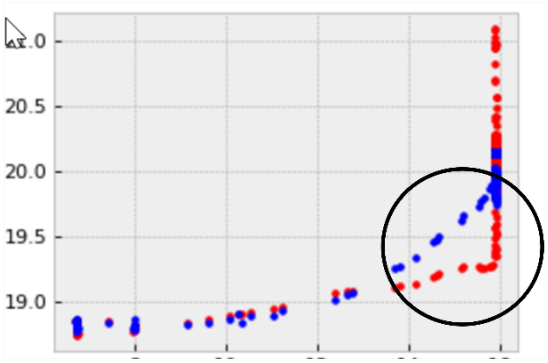

Figure 5.10: Extruder prediction and measured value, Blue is the prediction and red is the measurement.

By looking at the data in Trend it looks like there has been centering done in the region that is circled in the figure. It still seems to capture the correct trend, but with a slower dynamic. The trend should therefore hopefully still be captured when using data where centering has been done.

5.2. Can the independent model be used?

While the ELM can still categorize and accurately predict the data sets that it is given, it can, for the data it is not given, at best only label each new data set to a specific index. In this case, the index represents a specific centering screw, and tool set. And as previously mentioned, since the centering screw positions and the tools used for the extruder head is not logged anywhere and could not be obtained in the training of the model, the model cannot be used to predict the real process if the settings of the extruder head do not happen to coincide with the one represented by the chosen index. This is not a problem in the combined model, where the screws are modelled as well since each part can discern the settings of the other one. But it means that the model is only good for validation or looking at the trend, due to local uncertainties in intervals where an unknown input has been altered.

5.3. Single model clarification

At this point some clarification is needed to understand what has been done this far. The model that has been evaluated this far is basically a single model that describes the full extruder head from inputs to outputs. However, it has a subset of inputs that are unknown meaning that they only describe the extruder at the specific values of these inputs as they were in the training sets, and these values are unknown. In what has been coined the "first part" in this report, these unknown inputs are the tools and the centering screws. The initial idea was that the accuracy of the model can still be evaluated even if the training sets that are needed for it cannot be acquired in full since the training sets have been chosen so that the difference in the unknown inputs have a multicollinear relationship with each type of data set; the model basically attributes the effect of the unknown inputs to something that is known and strongly correlated to the unknowns. It turns out that, if the purpose is only to see if the trend is correct, even if this correlation is not present, the model still works. And the model can be fitted to any data set with good results, but it will fail to capture the dynamic in areas where the unknown inputs have been changed, but it does capture the correct trend.

5.4. Modelling centering screws first iteration

To test if the second part of the model that is lacking data sets work, old data from previous measurements were used in the task of training the model. The problem with this data is that it contains only a single run of the extruder, and only seven screw adjustments. The likelihood of a model being accurate with this data is almost null, but it can at least give an indication of how well the model can generalize the data it is given and how well it predicts the data that it does have.

The first iteration of the model gave very poor results, the generalization was poor and as demonstrated in figure 5.11, the errors propagating through the system causes the unstable model to go haywire as soon as the inputs deviate even slightly from the inputs in the training range as can be seen when simulating it.

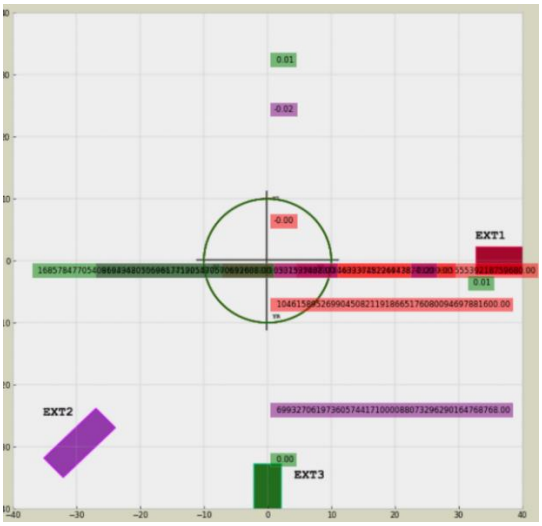

Figure 5.11: simulation of first iteration of models second part, with a 5% input deviation from the range in which its trained.

A deviation of 5% of screw angle outside of the range of the training inputs results in a cable with an average asymmetric layer thickness of eighty quadrillion lightyears. This version of the model would most likely not be good enough even with a full bank of training sets and tailored experiments. The fitting accuracy of a noise induced input set was practically zero.

5.5. Modelling centering screws second iteration

To lower the risk of error propagation and increase the resistance to noisy or inaccurate inputs together with increased robustness, a

regularization parameter was added to the second part of the model as well, and the model was expanded to a deep layer machine, which is earlier described in section 4. This version performed significantly better than the previous iteration and gave good accuracy as well as acceptable robustness. The training results of this version can be seen in table 5.2. The noise induced in the test was set to 20% more than the measurement inaccuracy of the Sikora and the approximated white noise of 30%. This is to see if the unsupervised layers are doing their job.

As can be seen the prediction accuracy is quite good. The largest error can be seen in outer semiconductor which is to be expected since the adjustment of this layer contains the least number of data points.

Table 5.2: Model results of second part.

	Training Error (%)	***Test Error (%)***	***Noisy Error (%)***
InnSem XL	0.908	0.928	0.934
InnSem XR	1.202	1.400	1.208
InnSem YL	0.810	0.872	0.817
InnSem YR	0.985	0.896	1.009
Insul XL	0.366	0.332	0.412
Insul XR	0.555	0.609	0.577
Insul YL	0.779	0.791	0.798
Insul YR	0.845	0.890	0.864
OutSem XL	2.654	2.460	2.675
OutSem XR	2.654	2.508	2.654
OutSem YL	3.107	3.068	3.139
OutSem YR	3.156	3.229	3.164

To showcase how well it generalizes the process the model was simulated with inputs 1200% outside of the fitting range. The resulting cross section can be seen in figure 5.12.

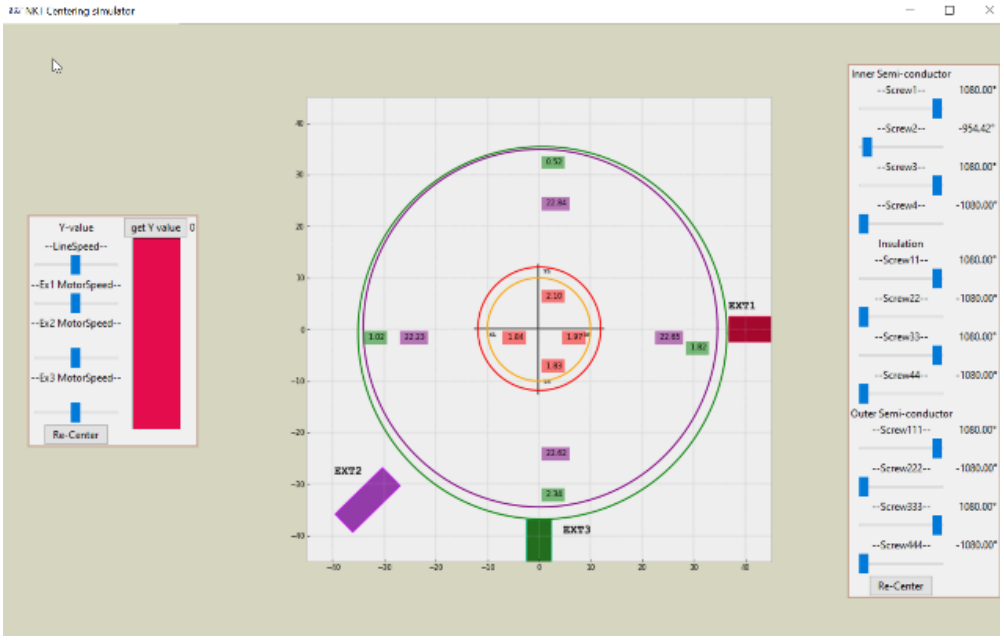

Figure 5.12: Second iteration of the second part of the model, simulated 1200% outside the range of training inputs, It must be noted that the simulation is done for Ex30 and not Ex31 so the position of the extruders are incorrect in the picture.

While the values are probably not correct, the trend is the same as that of the experimental run. And the values are consistent with logged runs of the extruder. If the process is run according to the experimental run, it has a largest deviation from the real value of 3%. This deviation is in the outer semiconductive layer. And the trend is the same as in the simulation in the figure above. This shows that with more data a model with a very high level of accuracy can be obtained if the data is available to use in training. Also, with the model having unsupervised learning, it would hopefully let the model cluster and find patterns in the data that lets it recognize and accurately predict noisy input signals, cold-measurement errors and sort out outlier datapoints. The effect would be that the tests run to acquire the data, won't need to be as rigorous and if the data can be logged, it could be low resolution and still be used in the training.

5.6. Observations

By far, the most difficult layer to predict accurately is the inner semiconducting layer. While the other two layers seem somewhat consistent regardless of what the flaws of the training have been. For example, when fitting the model to three different cable runs, the outer semi-conductor and the insulation still behave normally. This is not to be confused with them being true to the real process, but the variation in thickness doesn't fluctuate to unreasonable values. The inner semi-conductor is a different story. When only using the standard ELM model, as soon as the model goes outside of the linear range, even when it is in between two valid ranges the model becomes increasingly inaccurate. This is fixed with ridge regression, but the fit is generally worse than the other two when there is equal amount of data points in the training sets for all the inputs.

The general RMSE values of the model vs the real output is generally around 2% higher than when fitting the other 2 layers which indicates that the model does worse at predicting the inner semi-conductor with accuracy. This could be due to several reasons. The measurement could be noisier than the others which could cause the model to fit it worse. It could also just be more sensitive to the different inputs to the extruder.

5.7. How to get data for unknown inputs

The easiest problem to solve in the model is the issue of how the process is started and what data can be obtained from the history blocks. As mentioned earlier the line starts are done in a linear fashion as can be seen in figure 5.7.

Experiments where all but one extruder input is run at a constant, while one at a time each input would be adjusted over a range which encases all the possible operating ranges that the extruder will be ran on, would give the model the data it needs to generalize over the whole process. Because of the ridge regression this should only need to be done a couple of times for each input.

To solve the issue of the lack of centering screw data, a way of monitoring the screw would need to be added. The best way for model fitting would probably be to install sensors that can directly measure the angle between the conductor and the distributors in the extruder head. Not only can they be logged directly to the data base, but they are also far more precise than just looking at the screw. These sensors could also be added to just measure the angle from the sensor position, it doesn't really matter what angle it measures, as long as that angle can be converted to an input that can be understood by an operator.

Another possible way of solving the issue is to film the startup process and train another classification algorithm to the task of recognizing the screw positions. Then all that would be needed to monitor the inputs is a camera that records an image on the screw state, and the algorithm would recognize which position the screw is in. This would obviously require someone to create the model and train it first.

5.8. Validation

To validate how the first part of the model works independently in the operating ranges of the extruder is quite easy. All the data from random runs of the extruder that have not been part of the training or testing can be downloaded from the historian database and predicted. Thus, it is easy to see if the model is predicting accurately or not.

To estimate the accuracy of the model outside of the operating ranges is more difficult since it would require the extruder to be ran outside the operating ranges as a test. This might obviously be expensive, and it may not even be that interesting to see if the model is accurate outside of the operating range, or at least, not for the operating aid. But, to show what the model does in this range, the model was ran outside the operating range by between 10-30% on each extruder motor speed. The result can be seen in figure 5.13.

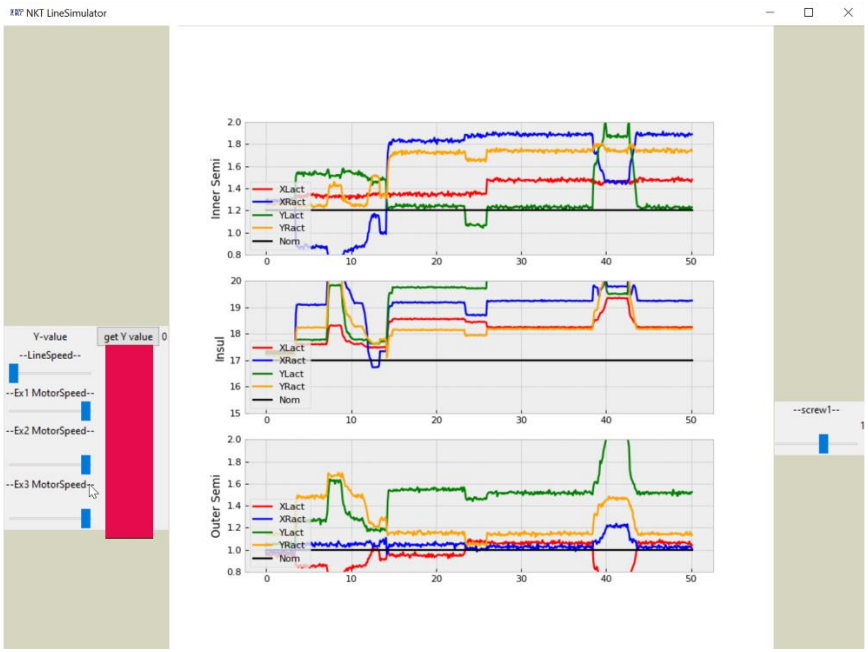

Figure 5.13: First part of the model ran 10-30% outside the extruder motor speed operating range. Simulating Ex31.

One quite peculiar point to make here is that the output seems to somewhat reverse the thickness order of the inner semiconducting layer when only the motor speed of the outer semiconducting layer is run outside of the operating range. When the model is run between the operating ranges but in the same linear fashion, the expected trend where the layers are the thinnest on the left side in the x direction and the left side on the y direction is observed. The former is where the melt flows together in the die. This is all displayed in figure 5.14.

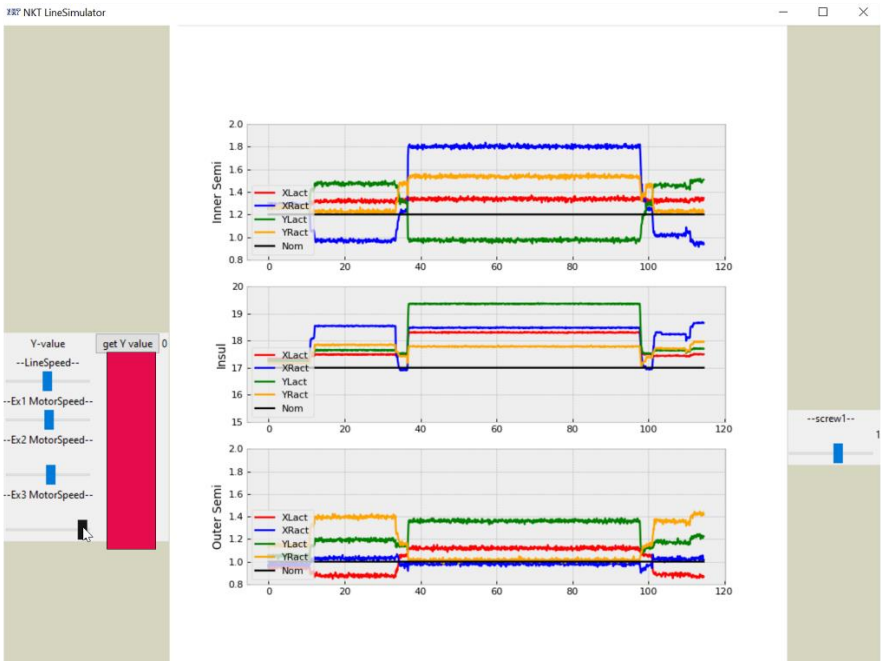

Figure 5.14: First part of the model ran between the extruder motor speed operating range for motor 1 and motor 2 and outside for motor 3. Simulating Ex31.

Once again, the expected output is not completely correct for all the data sets given as validation, which can be confidently attributed to the fact that these sets are ran with different extruder distributor settings (extruder screw positions, tool set). But it does seem to follow the correct trend. What this shows is that the model correctly filters out noise and manages to capture the slower frequency changes well.

The validation of the second part of the model is, in this project, quite pointless since the model has not been trained to sufficient data sets.

5.9. Interesting find

The tools used seem to have a much larger impact on the centering than initially thought. It was already known that the distributor position versus the conductor, i.e., what the centering screws affect, has a large impact. But it seems like maybe the toolset has a large impact as well. When letting the model predict different cables where the same motor and line speeds have been used, it seems like the centering can vary by a lot just based on the index being different (a different type of cable produced). This is displayed when fitting the data to sets of different cables, where there looks to have been no centering done. It is shown below in figure 5.15.

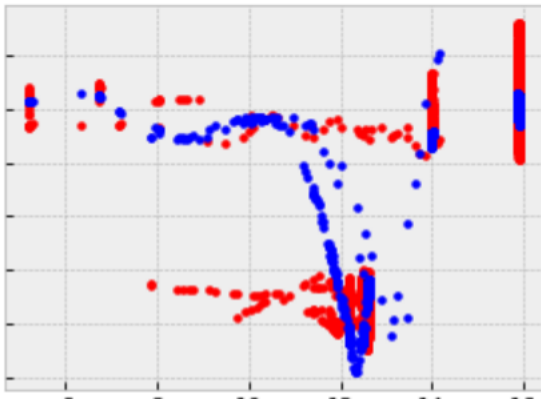

Figure 5.15: Extruder prediction of set where there most likely has been no centering, but cable types differ. Exhibits sign of underfit, even though nothing suggests it should be.

It is not completely clear what is causing this behavior, it might be the case that centering has indeed been done on these runs, and the method used to detect it is flawed, but the theory is as stated above that it is something else that is causing it, since the model seems to fail to distinguish the difference in outputs which may be to some linear relationship with the tools. It also may just be that the model finds some strange relationship in the data that happens to work in the regression.

6. Conclusion and future prospects

A model that can accurately predict and generalize over the data that it is given as training has been successfully achieved through the means of a deep learning extreme learning machine algorithm with ridge regression (DRELM). However, good enough training to train the model to predict in a wide enough range of inputs, so that it can be used as an operating tool for NKT, could not be acquired. It must also be pointed out that when there is enough data for all the inputs available the DRELM should not be needed over the ELM with ridge regression, since there would be no need to classify if the data belongs to different base settings anymore, but the efficiency of both are very similar.

6.1. Reflection

The model has obvious flaws in how it is trained. And while the DRELM does a good job of adapting to the training sets it is given, it will only ever be as good as the training it gets.

To truly make a model that can accurately describe the process together with the time dependance, a few things would need to be added. First a time parameter needs to be included; the data that is generated does have time stamps from when it is taken, but it is only when the reading from the measuring equipment is taken, so the delay would need to be estimated in the time stamps. The data is also converted to post vulcanization values, or so-called cold measurements, by computation and this would need to be included.

However, the purpose of this model is not to describe the model in real-time, only to estimate the effects of extruder inputs on the steady state output. To do this, it only needs to capture how the steady state of the cable cross-section changes upon input changes which can be done with a discrete time model. Data from the screws would also have to be logged in a way that can be accessed by a computer, so it can be used when training the model. This could be done manually by operators with pen and paper, or in custom

software and then be transferred by hand to digital sheet, or directly by measuring equipment.

The better the data sets that can be obtained, the better the model will perform. If it was possible to run experiments tailored to each input type and ways that the extruder heads can be set up, it would make the performance of the model better, but as have been shown in this project, the quality of experiments can be quite poor, and it would still work.

The input changes of the distributor screws would have to be measured against a baseline of the distributors, which can act as a reference for the input changes or taken as an angle measurement. Better documentation on what toolset is used in the extruder would also be needed. So, to put it simple, most problems could easily be solved by just monitoring and logging the distributor data to the historian database.

6.2. Future work

This leads to the subject of how to continue this project if it is deemed worth. The current model has already proven to be able to accurately predict layer thickness of the data it is trained to. It has good generalization capabilities, and all that is needed is more data sets.

Extensive trial runs would need to be done to gather data to further validate the model and see how well the second part of the model works. The easiest would be to use data from bleed tests to train the first part of the model or substitute it for the model that each cable "recipe" is taken from and train the model to reconstructed data. Many more tests would also have to be run so there is enough data to train the model.

If all the data can be obtained simultaneously from the database or by other means, the model should be run as a single part with all the inputs and outputs, the only reason to run the model as two separate parts is when lacking data from one or more inputs.

As previously mentioned, the best way would be to install sensors or other measuring equipment and start logging the distributor data. This would passively generate model training data, which has been proven in this project to be good enough to eventually create a model that works.

6.3. Things the model can achieve

As touched upon in the literature study, a possibility with this type of model might be that the process can be controlled without the help of an operator. With an accurate enough model, a heuristic dynamic programming algorithm could easily be set up to generate optimal controllers for NKTs extruder. This could in turn let the process be controlled much more precisely than a human ever could. On top of that, the network could possibly be taught how to counteract white noise in the sensors used to monitor the process input signals, which could create a smoother cable surface. Further classifier algorithms may also be possible to make, to automate crash procedures or other types of quality control. An example of this would be to feed the model data to a topology scanner, another AI could then be taught to predict the topology of the cable purely based on model output. The AI could then tell the operator in real time what the effects of what is currently happening are and suggest a change to the operator of how to fix it. This could hypothetically let operators run the process much closer to the nominal values without running the risk of going under it and ruining the cable.

There is also the possibility of quick retraining, and the model can be run and trained to the start-up of new extruder lines, to quickly understand how the process should be operated or identifying faults.

6.4. Take aways

NKT has a lot to gain from better tracking of their process parameters, both in terms of creating sophisticated software to aid in process operations, but also for a better understanding of how the process works, which would be especially useful to identify problems and solutions in start up of new extruder lines.

References

[1] J. R. Wagner, E. M. Mount, and H. F. Giles, "1 - Extrusion Process," in *Extrusion (Second Edition)*, J. R. Wagner, E. M. Mount, and H. F. Giles Eds. Oxford: William Andrew Publishing, 2014, pp. 3-11.

[2] E. Lundström, K. Langsér, Ed., ed, 2023.

[3] J. R. Wagner, E. M. Mount, and H. F. Giles, "40 - Coextrusion Applications," in *Extrusion (Second Edition)*, J. R. Wagner, E. M. Mount, and H. F. Giles Eds. Oxford: William Andrew Publishing, 2014, pp. 449-466.

[4] H. Li, L. Liu, W. Luo, X. Zhang, and M. Luo, "Research on Optimization Method of Wire and Cable Extrusion Control System," in *2019 Chinese Automation Congress (CAC)*, 22-24 Nov. 2019 2019, pp. 1703-1708, doi: 10.1109/CAC48633.2019.8997078.

[5] K. S. Boparai, R. Singh, and H. Singh, "Modeling and optimization of extrusion process parameters for the development of Nylon6–Al–Al2O3 alternative FDM filament," *Progress in Additive Manufacturing,* vol. 1, no. 1, pp. 115-128, 2016/06/01 2016, doi: 10.1007/s40964-016-0011-x.

[6] D. C. Montgomery, *Design and Analysis of Experiments*, 9th ed. Wiley, 2017.

[7] J. Wang, S. Lu, S.-H. Wang, and Y.-D. Zhang, "A review on extreme learning machine," *Multimedia Tools and Applications,* vol. 81, no. 29, pp. 41611-41660, 2022/12/01 2022, doi: 10.1007/s11042-021-11007-7.

[8] P. L. Bartlett, "The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network," *IEEE Transactions on Information Theory,* Article vol. 44, no. 2, pp. 525-536, 1998, doi: 10.1109/18.661502.

[9] X. Ding, J. Liu, F. Yang, and J. Cao, "Random compact Gaussian kernel: Application to ELM classification and regression," *Knowledge-Based Systems,* vol. 217, p. 106848, 2021/04/06/ 2021, doi: https://doi.org/10.1016/j.knosys.2021.106848.

[10] S. Ding, N. Zhang, X. Xu, L. Guo, and J. Zhang, "Deep Extreme Learning Machine and Its Application in EEG Classification," *Mathematical Problems in Engineering,* vol. 2015, p. 129021, 2015/05/27 2015, doi: 10.1155/2015/129021.

[11] *Wonderware Historian Retrieval Guide*: Schnieder Electric Software, 2017. Accessed on: 2/20.

[12] "X-RAY 8000 ADVANCED/NXT." Sikora International Corporation. https://sikora.net/en/products/xray8000nxt/ (accessed 04/19, 2023).

[13] M. G. Andersson *et al.*, "Highly Insulating Polyethylene Blends for High-Voltage Direct-Current Power Cables," *ACS Macro Letters,* vol. 6, no. 2, pp. 78-82, 2017/02/21 2017, doi: 10.1021/acsmacrolett.6b00941.

[14] F. B. Meng *et al.*, "Insulation Properties and Interfacial Quantum Chemical Analysis of Cross-Linked Polyethylene Under Different Degassing Time for HVDC Cable Factory Joint Applications," *IEEE Transactions on Dielectrics and Electrical Insulation,* vol. 30, no. 1, pp. 271-278, 2023, doi: 10.1109/TDEI.2022.3226136.

[15] R. Nejdl and V. Mentlík, "Influence of tools setting on processing quality of XLPE core insulation," in *Proceedings of the 2014 15th International Scientific Conference on Electric Power Engineering (EPE)*, 12-14 May 2014 2014, pp. 407-410, doi: 10.1109/EPE.2014.6839456.