



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

END-OF-DEGREE PROJECT
Degree in Chemical engineering
**SYSTEMATIC METHODOLOGY TO DEVELOP SURROGATE
MODELS FOR CHEMICAL PROCESSES USING ARTIFICIAL
NEURAL NETWORKS**



Report and Appendices

Author:	Guillem Rovira Herrero
Director:	Carlos Pozo Fernández
Call:	June 2022

Resum

Un dels principals problemes a l'hora de fer simulacions de processos químics és la gran càrrega computacional i l'elevat temps d'execució que es requereix degut a la complexitat d'aquests sistemes. Això és especialment problemàtic en el paradigma actual, on la connexió d'aquestes simulacions podria contribuir a la identificació d'oportunitats per a la simbiosis industrial i accelerar així la transició cap a una economia circular. En aquest context són especialment prometedors uns nous models, anomenats models de substitució, que puguin simplificar la mateixa tasca de simulació, reduint-ne el temps i la càrrega computacional.

En el present treball, es desenvoluparà una metodologia sistemàtica per aplicar mètodes de substitució mitjançant algorismes d'aprenentatge autònom, en concret, els coneguts com a xarxes neuronals artificials. Primerament, la falta de dades bibliogràfiques suficients per entrenar els models de substitució, ens portarà a simular el procés de piròlisi de polietilè amb Aspen HYSYS. Posteriorment, s'estudiaran dues alternatives per desenvolupar mètodes de substitució d'aquesta simulació. En la primera, es tractarà el procés com un sol conjunt, pel que es desenvoluparà un sol model de substitució que representi tot el procés. En la segona, el procés es tractarà per mòduls (de forma seqüencial), pel que es desenvoluparan tants models de substitució com mòduls es duguin a terme en el procés. En aquesta segona alternativa, els models de substitució es connectaran de manera seqüencial, pel que algunes de les variables de sortida predites per un model, s'utilitzaran com a variables d'entrada pel model del següent mòdul. Finalment, es compararà totes dues alternatives, d'acord segons el temps i la precisió que aconseguixin.

Resumen

Uno de los principales problemas a la hora de realizar simulaciones de procesos químicos es la gran carga computacional y el elevado tiempo de ejecución que se requiere debido a la complejidad de estos sistemas. Esto es especialmente problemático en el paradigma actual, en el que la conexión de estas simulaciones podría contribuir a la identificación de oportunidades para la simbiosis industrial y acelerar así la transición hacia una economía circular. En este contexto son especialmente prometedores unos nuevos modelos, llamados modelos de sustitución, que puedan simplificar la misma tarea de simulación, reduciendo su tiempo y carga computacional.

En el presente trabajo, se desarrollará una metodología para aplicar métodos de sustitución mediante algoritmos de aprendizaje autónomo, en concreto, los conocidos como redes neuronales artificiales. En primer lugar, la falta de datos bibliográficos suficientes para entrenar los modelos de sustitución nos llevará a simular el proceso de pirólisis de polietileno con Aspen HYSYS. Posteriormente, se estudiarán dos alternativas para desarrollar métodos de sustitución de esta simulación. En la primera, se tratará el proceso como un solo conjunto, por lo que se desarrollará un solo modelo de sustitución que represente todo el proceso. En la segunda, el proceso se tratará por módulos (de forma secuencial), por lo que se desarrollarán tantos modelos de sustitución como módulos se lleven a cabo en el proceso. En esta segunda alternativa, los modelos de sustitución se conectarán de forma secuencial, por lo que algunas de las variables de salida predichas por un modelo, se utilizarán como variables de entrada para el modelo del siguiente módulo. Por último, se comparará ambas alternativas, según el tiempo y la precisión que consigan.

Abstract

One of the main problems when performing chemical process simulations is the large computational load and the high execution time required due to the complexity of these systems. This is especially problematic in the current paradigm, where the connection of these simulations could contribute to the identification of opportunities for industrial symbiosis and thus accelerate the transition towards a circular economy. Particularly promising in this context are new models, called surrogate models, which can simplify the simulation task itself, reducing its time and computational burden.

In this paper, a methodology will be developed to apply surrogate models using machine learning algorithms, namely those known as artificial neural networks. First, the lack of sufficient literature data to train the surrogate models will lead us to simulate the polyethylene pyrolysis process with Aspen HYSYS. Subsequently, two alternatives will be studied to develop surrogate methods for this simulation. In the first, the process will be treated as a holistic system, so a single surrogate model representing the entire process will be developed. In the second alternative, the process will be treated in modules (sequentially), so as many surrogate models will be developed as modules are carried out in the process. In this second alternative, the surrogate models will be connected sequentially, so that some of the output variables predicted by one model will be used as input variables for the model of the next module. Finally, both alternatives will be compared according to the time and accuracy they achieve.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my project director Dr. Carlos Pozo, who has guided and mentored me throughout the project.

Next, I would like to mention Dr. Antonio Espuña, who has always been available to help with any doubts and has raised interesting questions and discussions about my project.

I would also like to make a special mention to my mother, Inés Herrero, who has provided me with help throughout the project, reviewing the text of the report and providing a different perspective.

Finally, I would like to express my gratitude to my friends and family who, with their unconditional support, have helped me during the course of these years, and have made all this possible.

Glossary

PE – Polyethylene

ANN – Artificial Neural Network

ReLU – Rectified Linear Unit

mse – Mean of the Squared Errors

NT – Number of Trays

RR – Reflux Ratio

Dflow – Distillate flow/rate

HK – Heavy Key

LK – Light key

NaN – Not a Number

df - DataFrame

Index

RESUM	I
RESUMEN	II
ABSTRACT	III
ACKNOWLEDGMENTS	IV
GLOSSARY	V
1. PREFACE	1
1.1. Project origin	1
1.2. Motivation and scope of the project	1
2. INTRODUCTION	3
2.1. Project objectives.....	3
3. INTRODUCTION TO MACHINE LEARNING	5
4. OVERVIEW OF SURROGATE MODELS	7
4.1. Artificial Neural Networks	7
5. METHODOLOGY	11
6. TOOLS	24
6.1. Commercial process simulator	24
6.2. Programming language and useful libraries	24
7. CASE STUDY: POLYETHYLENE WASTE PYROLYSIS	26
8. RESULTS AND DISCUSSION	33
8.1. Data generation.....	33
8.2. Pre-processing and cleaning	37
8.3. Building the architecture of the surrogate models.....	40
9. ENVIRONMENTAL EVALUATION	50
CONCLUSIONS	51
ECONOMIC EVALUATION	53
BIBLIOGRAPHY	57
APPENDIX A	61
A1. Discussion of the modular surrogate error	61
A2. Files used to build the surrogate models	63

1. Preface

1.1. Project origin

Process simulation and optimization are important steps for the design and operation of chemical manufacturing processes, since they can help to improve their performance in a reliable and economical manner (Ma et al. 2015) and help to understand which solutions are viable.

From a general Process System Engineering perspective, large networks are being built to compare different models economically and environmentally wise (Calvo-Serrano et al. 2019). However, combining rigorous simulation models into a large network is computationally very expensive and time-consuming (Y. F. Li et al. 2010). Hence, alternative tools such as surrogate models are emerging to simplify complex models, while maintaining an acceptable accuracy (Bhosekar and Ierapetritou 2018).

In this project, a methodology to build surrogate models for chemical processes will be developed, and the data needed to train these models will be automatically generated using a simulation process as case study, which will also be designed. The surrogate models will use machine learning algorithms, which try to emulate human intelligence by learning from the surrounding environment (el Naqa and Murphy 2015). In particular, artificial neural networks will be used to simplify part (i.e., the two last distillation columns) of the simulation of the pyrolysis of waste polyethylene.

1.2. Motivation and scope of the project

The urge to move towards an ecological transition and circular economy due to the impact of global warming has given rise new environmentally friendly solutions, and the chemical industry plays a key role in doing so. Many initiatives have emerged, being plastic waste treatment one of the most popular in the chemical industry. Global production of plastics has been continuously increasing from 1.5 million in 1950 to 367 million in 2020 (Statista 2022), and while cumulative plastic waste generation amounted to 6300 million between 1950 and 2015, only 9% was recycled (Geyer, Jambeck, and Law 2017). Nowadays tackling this problem has become an urgent necessity and highlights the importance of circular economy. The pyrolysis process is becoming a popular technique to treat plastic waste. By producing value-added products, pyrolysis overcomes the inherent disadvantage of landfilling while reducing the carbon footprint by up to 67%. (H. Li et al. 2021)

In this project, a complete simulation of a polyethylene pyrolysis process will be designed to study the performance of different surrogate models. However, only the last two distillation columns of the process will be replaced by the surrogate on this project, leaving the rest for future work.

The columns will be studied for some input variables (see Table 8.1) within a range of values (see Table 8.2). After generating the sampling data, two surrogate models will be trained and compared. The first one will be an artificial neural network that encompass both distillation columns concurrently (holistic surrogate model), while the second one will be composed by two different artificial neural networks, one for each distillation column, that will be connected through some bridge variables (modular surrogate model). The accuracy and time to develop both models will be analyzed, in order to determine which performs better according to these metrics. Finally, for the

best performing model, the number of hidden units for each hidden layer will be studied to determine the best configuration within a constant value of hidden layers.

2. Introduction

Following the idea of circular economy, plastic waste can be treated to be used as fuel or to recover its monomers, which can be reused to obtain polymers again. The case study addressed in this project will be a waste polyethylene pyrolysis process at 900°C, to recover various monomers. Usually, a previous simulation and optimization of the process is made, to satisfy some previously known requirements. On a plastic waste recovery plant, the requirements are usually to recover valuable monomers (high purity) given a known feed (Khoo 2019).

One of the purposes of this project is to explore the usefulness of surrogate models in replacing rigorous simulations. A methodology to build surrogate models using artificial neural networks will be elaborated and two different alternatives to build surrogate models (holistic vs modular) for chemical processes will be evaluated. The evaluation of the surrogates will encompass the computational time to generate the data and the accuracy of the surrogate models. The accuracy of the surrogate models will be measured using the mean of the squared errors. This function returns the error between the real output, retrieved by an Aspen HYSYS simulation, and the predicted output given by the trained surrogate, for each sampling point.

2.1. Project objectives

The project has four goals, which are listed below.

- To develop a systematic methodology to build surrogate models for chemical processes.
- To study the computational time required to generate the necessary data to train the surrogate model, considering two different approaches: holistic vs modular.
- To analyze and compare the accuracy of a holistic and a modular surrogate model.
- To determine the optimal number of hidden units, within a range of values, for a constant number of hidden layers, evaluated with the accuracy for every output variable.
- To deconstruct the methodology proposed in very specific actions so that it can be easily followed by somebody else, enabling the continuation of the proposed work in the future.

3. Introduction to machine learning

Tom Mitchell described machine learning as a computer program that “is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .” (Mitchell 2006)

Most machine learning problems can be assigned to one of the two following categories: supervised learning and unsupervised learning. In supervised learning, the relationship between an input and a known correct output are known in advanced, whereas, in unsupervised learning, we do not know what our results will look like.

In this project, all data will be generated by a previous HYSYS simulation and the relation between the input and the correct output will be known. Hence only supervised learning algorithms will be applied to model the surrogates. Further explanation on supervised learning is provided next.

Supervised learning problems are categorized into regression and classification problems. The difference between the two categories is that, in regression problems, we try to predict results within a continuous output, while in classification problems, we try to predict results with a discrete output. A typical example of a regression problem would be trying to predict the housing price given the size of the house. An example of a classification problem is spam filtering, given a text we try to determine if it is spam or not.

In any supervised learning problem, there is (i) a hypothesis function relating the output to a given input, and (ii) a cost function (also called loss function) measuring the accuracy of our hypothesis function. Hypothesis functions ($h_{\theta}(x)$) are defined by the input variables (x_i) and the weight parameters (θ_i). Weight parameters define the change of the slope of the hypothesis function with changes in the values of x_i . The cost function ($J(\theta)$) usually has a lambda λ parameter, also called regularization parameter, which determines how much the costs of our θ_i are inflated, thus helping the model not to overfit the data, i.e. the model will be less accurate without the use of the λ . If λ parameter is set to an extremely large value, the model will underfit the data, i.e., the model will be less accurate, too. The definition and training of the hypothesis function may give rise to three potential scenarios that will be explained below, together with some possible ways to improve them. An example of each scenario is shown in Figure 3.1.

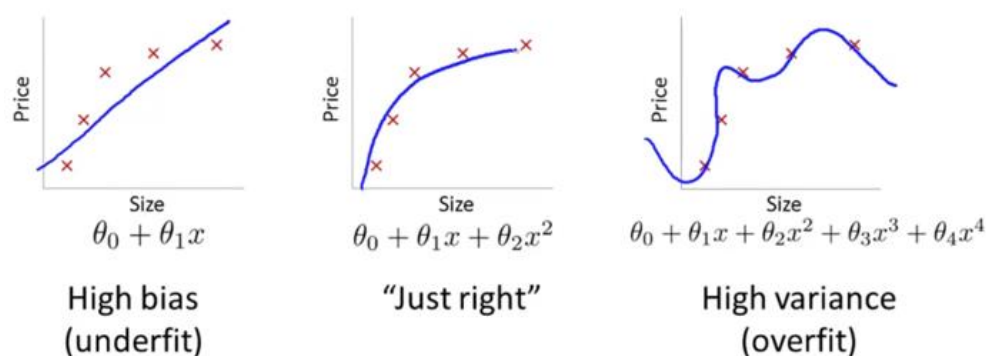


Figure 3.1. High bias and high variance models' comparison (Ng 2011)

In the first scenario, also referred to as high bias, our hypothesis function underfits the data. Adding more features or decreasing the regularization parameter of our cost function are common solutions to help reducing high bias. The second scenario, also referred to as high variance, happens when $h_{\theta}(x)$ overfits the data. Getting more training samples or trying smaller sets of features would help to reduce high variance. Contrary to high bias, increasing λ would help to lower high variance. The last one is the best possible scenario, which happens when our data fits just right, that meaning our hypothesis function does not have high bias nor high variance.

4. Overview of surrogate models

When designing complex industrial processes, we often manage computationally expensive simulation models that might be difficult to solve (i.e., convergence problems), optimize or integrate in larger process networks. To alleviate this burden, one can resort to so-called surrogate models, which are simplified approximations that mimic the relation between input and output behavior of complex systems. In the context of chemical processes, an input could be the number of trays of a distillation column and an output could be the molar fraction of the heavy key component at the bottoms of the same distillation column.

Surrogate models are typically used to reduce the computational expense of complex models or when the relation between input and output is unknown (Williams and Cremaschi 2019). There has been one key factor that made increase the use of surrogate models (especially the ones that use machine learning algorithms) which is the access to large amounts of data. This has allowed machine learning models such as deep learning to perform better than ever before (Ng 2017).

There are many types of surrogate models. Some of the most cited in the literature are *Polynomial Response Surface (PRS)*, which performs well in linear and low-dimensional problems (W. Li, Lin, and Li 2016); *Radial Basis Function*, which is widely used for feasibility analysis (Wang and Ierapetritou 2017); *Support Vector Regression (SVR)*, which is a *Support Vector Machine (SVM)* applied to a regression problem and can be divided into linear SVR and non-linear SVR (Jiang, Zhou, and Shao 2020), *Kriging*, also known as *Gaussian Process Regression* in the field of machine learning, which is used for design and analysis of computer experiments (Bhosekar and Ierapetritou 2018) among other uses; and *Artificial Neural Networks (ANNs)*, which are known to be universal function approximators with low economic computational consumption (Sun and Wang 2019; Pan et al. 2014).

Without loss of generality, ANNs are selected in this project to generate the surrogate model of a process simulation, owing to their wide use as surrogates for process modeling (Meert and Rijckaert 1998), process control (Mujtaba, Aziz, and Hussain 2006) and optimization problems (Nguyen, Nong, and Paustian 2019). ANNs will be further explained in detail in the following section.

4.1. Artificial Neural Networks

Artificial Neural Networks (ANNs) are inspired by the human brain as shown in Figure 4.1.1, where a neuron would be the equivalent to a computational unit. Neurons communicate with one another at synapses (weights W) and take input signals from its dendrites (features x_i) to produce output signals along its axon. For surrogate models, the dendrites would correspond to the decision variables (input features) and the output would be the variables to study, i.e., the output variables. The axon eventually branches out and connects to other dendrites (and neurons) via synapses.

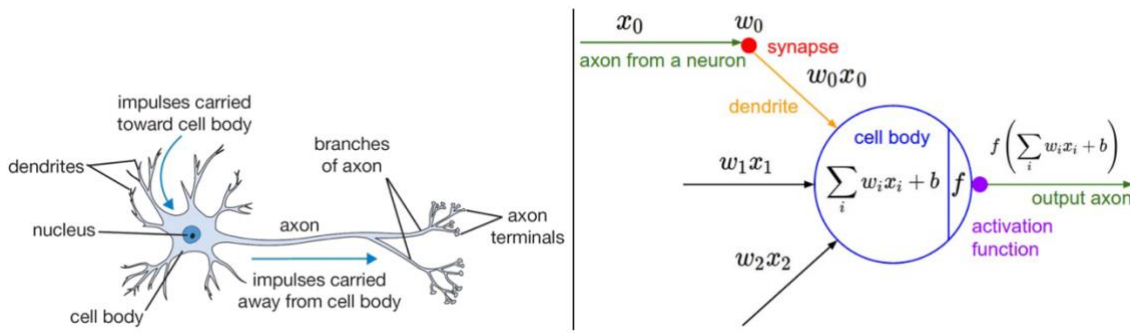


Figure 4.1.1 A cartoon drawing of a biological neuron (left) and its mathematical model (right) (Stanford University 2020)

Activation functions are needed to introduce non-linearity to the so-called hidden layers (see Figure 4.1.2) (Sharma, Sharma, and Athaiya 2020), which allows to solve more complex problems. Following the simile of the biological neuron, activation functions represent the frequency of the spikes along the axon.

There are many different architectures for an ANNs, but they all share a basic structure composed of node layers that contains an input layer, one or more hidden layers and an output layer. A single neuron is equivalent to a linear regression. Figure 4.1.3 depicts a schematic of a neural network layers and a schematic of a single neuron.

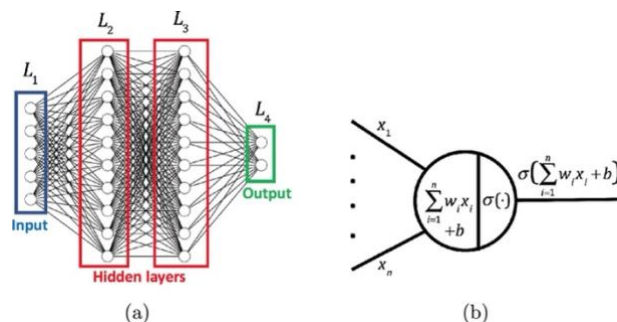


Figure 4.1.2 Schematic of a neural network (NN) (a). Schematic of a single neuron (b) (Tripathy and Bilonis 2018).

The general methodology to build an ANN consists of first defining the ANN structure, meaning the number of input units, which should be equal to the dimension of features x_i ; the number of output units, which should be the number of variables to predict for each example in regression problems and the number of classes in classification problems; the number of hidden units per layer, and the number of total layers. If the number of hidden layers and hidden units is too low, the model will underfit the data, while if it is too large, it will overfit the data. The optimal number depends on the complexity of the specific problem to solve. Usually, more hidden units and layers are used for complex models, although the computational burden also increases. The structure of the ANN in Figure 4.1.2(a) is composed by 5 input units (blue), 2 output units (green), 10 hidden units per layer (red) and 4 total layers (the input layer, two hidden layers and the output layer).

Once the ANN structure is defined, three steps are required to build an ANN, as shown in Figure 4.1.3.

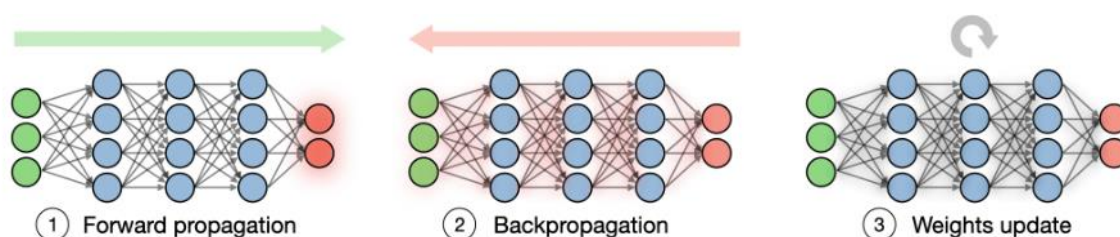


Figure 4.1.3 Schematic of the three steps to build an ANN (Amidi and Amidi 2019).

Firstly, forward propagation is implemented, which consists in taking the training data as input to obtain an output and compare it to the desired output.

To start forward propagation, the input layer is defined as every input unit corresponds to a feature x_i of our data plus a bias unit, which is added to increase the flexibility of the model to fit our data. Bias (b) and weights (W) parameters are randomly initialized before starting calculations. Weights affect the amount of influence a change in the input will have upon the output. When these parameters are set, parameter z , which is the computation of the weighted sum, can be calculated for every hidden unit as shown in Eq. 4.1.

$$z^l = W^l x + b^l \quad \text{Eq. 4.1}$$

Here, l is the index of the layer, W is the weight matrix, and b is the bias vector.

After computing z , there will be a calculation to compute the activation a , which introduces non-linearity to the hidden layers. Most common activation functions are the Sigmoid function and the Rectified Linear Unit (ReLU). Each layer can have a different activation function, e.g., the first hidden layer could use a Sigmoid function while the second hidden layer uses ReLU. Eq. 4.2 shows the sigmoid function of z .

$$a^l = \sigma(z^l) = \frac{1}{1 + e^{-z^l}} \quad \text{Eq. 4.2}$$

For the input layer, the activation function a^0 equals to x_i , and for the output layer, a^L equals to \hat{y} (predicted output). So, parameter z will be computed using the activation function of the last layer, and Eq. 4.1 can be written as shown in Eq. 4.3.

$$z^l = W^l a^{l-1} + b^l \quad \text{Eq. 4.3}$$

For the output layer, to calculate parameter z equation Eq. 4.3 is used, but the bias vector b is zero (Ng 2020).

Finally, after all layers have done those computations, the cost function J , also called loss function $L(\hat{y}, y)$ is calculated to evaluate the performance of the neural network (\hat{y}) by comparing the predicted output to the expected (i.e. the real) output (y). Many functions can be used to compute J depending on the task of the model. For instance, binary classification problems use a binary cross-entropy function, while multiclass problems use a sparse categorical cross-entropy function and

regression problems use the mean squared error. Since on this project the problem is a multivariate regression problem, the mean of squared errors is used, as shown in Eq. 4.4.

$$MSE = \frac{\sum_i (y_i - \hat{y}_i)^2}{N} \quad \text{Eq. 4.4}$$

The third step to build an ANN is to implement backward propagation to obtain the so-called gradients. Backward propagation consists of computing the derivatives of the parameters calculated in forward propagation, to obtain the gradient of the loss function with respect to the parameters. The following equations correspond to the calculations made in backward propagation for each node, starting by the output layer.

$$dz^l = a^l - Y \quad \text{Eq. 4.5}$$

$$dW^l = dz^l a^{(l-1)T} \quad \text{Eq. 4.6}$$

$$db^l = dz^l \quad \text{Eq. 4.7}$$

$$da^l = w^{(l)T} dz^l \quad \text{Eq. 4.8}$$

After the output layers, the parameter z computations are the following for all hidden layers. The other parameters follow the same equations as the output layer.

$$dz^l = da^l g'(z^l) \quad \text{Eq. 4.9}$$

After backpropagation is calculated, the weight parameters are updated (third step in Figure 4.1.3) so the model can fit the data better (i.e., the hypothesis functions learns from the mse and updates its parameters to reduce this error for the given data).

5. Methodology

One of the main objectives of this project is to elaborate a general and systematic methodology to build surrogate models for chemical processes. The flowchart for the general steps to follow is shown in Figure 5.1.

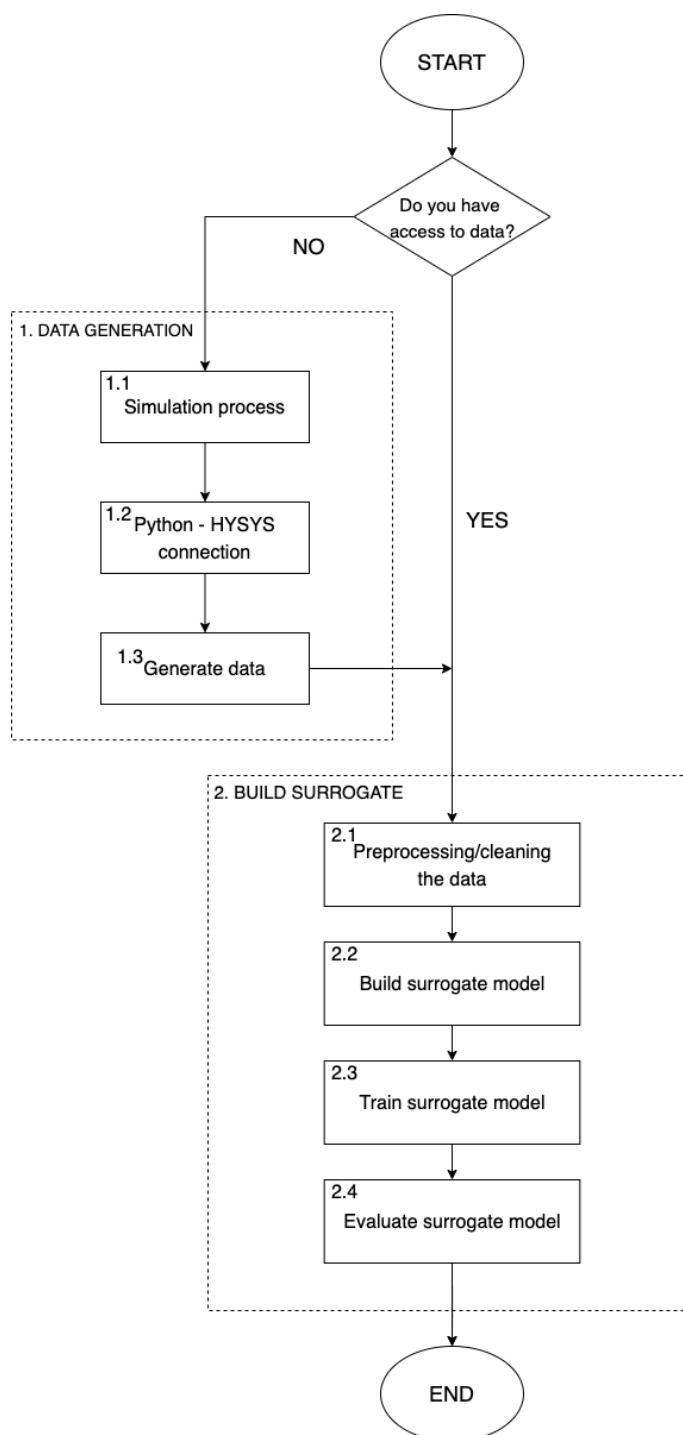


Figure 5.1. Flowchart of the steps to follow to build surrogate models for chemical processes.

The most important thing before starting to develop a surrogate is to have access to large amounts of data to use it to train the surrogate model. If data is unavailable, it can be generated through a simulation process, as shown in the multiple steps of the first big block named “1. DATA GENERATION” in Figure 5.1. Conversely, if data is indeed available, these steps can be skipped and the procedure will then start at the second block of the methodology, called “2. BUILD SURROGATE” in Figure 5.1, which consists of building the surrogate model.

In the worst scenario, data is not available, so it has to be generated by an automated simulation. To this end, one can follow the flowchart depicted in Figure 5.2, which is a more detailed version of block 1.1 in Figure 5.1.

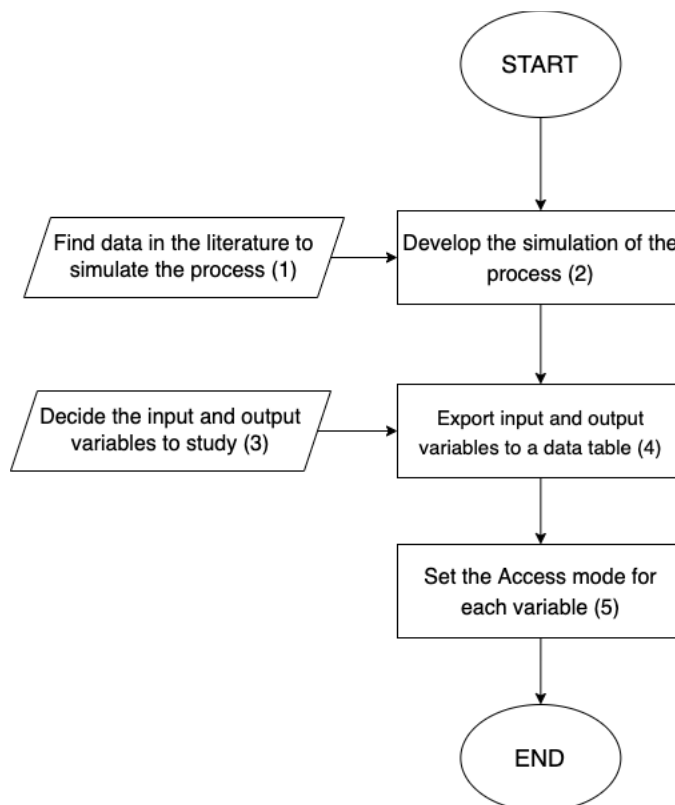


Figure 5.2. Flowchart of the simulation process.

To implement this methodology the next steps must be followed (each number in Figure 5.2 match to the numbers listed below):

1.

Research in the literature for the data needed to elaborate a complete simulation of the desired process. Only one feasible solution that satisfies the requirements of the specific problem to address is needed: additional data can be generated afterwards by altering the values of input variables. This process can be automatized using Python functions, which will send data points (samples) to the input variables and read the outputs from a HYSYS data table. These data will be used to train the surrogate model later.

In this project the data needed to carry out the simulation were extracted from the literature (Somoza et al. 2020; Honus et al. 2016).

2.

Simulate the process using a commercial process simulator. Some of the most common ones have been listed in chapter 6.1. Aspen HYSYS was the software selected for this project, although the general methodology is suitable even if the software used to elaborate the simulation is not Aspen HYSYS.

3.

Decide the input and output variables for the surrogate models. Input variables are typically (some) decision variables of the simulation, while output variables are (some) dependent variables whose values are to be predicted. This step is very specific to the problem addressed. To provide a hint, the input variables chosen for this study were the number of trays, the reflux ratio and the distillate rate for each distillation column studied (T4 and T5, as shown in Table 8.1). Meanwhile, the output variables selected were the output pressure and temperature at the bottoms of T4, the input pressure and temperature of T5, the heat flow at the condenser and at the reboiler of both columns and the mass flow at the distillate of both columns. In addition, bearing in mind the modular surrogate, some additional output data is also collected for the first of the two columns addressed (column T4), in order to address its connection with the column immediately downstream (column T5), namely, the molar fraction of the heavy key and the light key, plus the mass flow of the bottoms stream of column T4 (which happens to be the inlet to column T5). Note that the output and input pressure of the columns are not decision variables here (since the former is the same as the feed of T5, which is set by control valve V3 to 10 bar, and the latter it is set to 1 bar by control valve V4, as explained in chapter 7) and, consequently, should be constant values. This means those variables should not provide any useful information to train the surrogate model, however, they were selected to guarantee that the Python functions work properly (i.e., not affected by this variable) and to provide an example of how to clean the data before using it to train a surrogate model (check step 11). Furthermore, the mass flow at the distillate was taken both as an input and as an output variable. This was done to check that the connection between Python and HYSYS was working properly, i.e., the sampling points were being correctly introduced to the data table, hence it can be considered as a “control variable”.

4.

After building the simulation processes, one needs to create many sample points using the simulation model (i.e., each sample point will be a simulation). To generate all the sample points automatically using Python functions, one must establish a connection between HYSYS and Python so that they can communicate with each other, i.e., send and receive data. To do that, all input and output variables have to be exported into a data table. This requires first to create the data table in Aspen HYSYS. Then, one can right-click onto the desired variable, select “send to”, then select “Data Tables”. The names of all the data tables created so far will be displayed. This process is shown in Figure 5.3. For this case, one data table is enough.

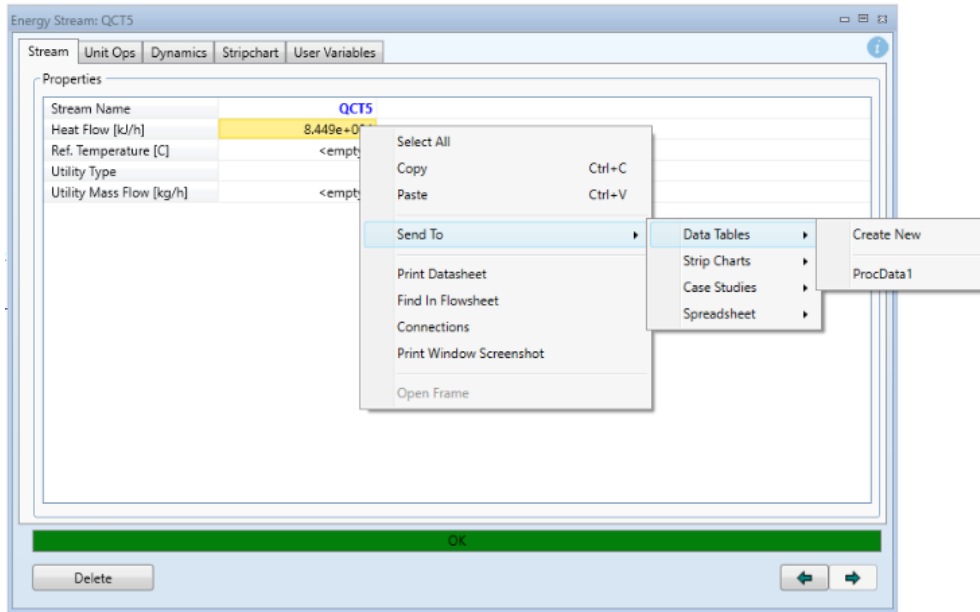


Figure 5.3. Path to follow to send any variable to a data table.

As variables are sent to the data table, they will appear in the same order they have been sent. Although the order of the variables can be changed later in the data table using the arrows at the right-hand side of the window, it is worth noting that independent variables should appear at the top of the table, since this will facilitate coding the functions to send data from Python to HYSYS.

5.

Define the 'Access Mode' for each variable on the data table and introduce a name on the column "Tag". Input variables for the surrogate must be defined as "Read/Write" since they must be able to receive (i.e., write) data from Python functions, and to send (i.e., read) data by Python functions. Meanwhile, output variables for the surrogate should be defined as "Read" because Python functions should not overwrite them.

At this point, the simulation is ready to receive different values for the input variables and propagate them through the simulation in order to generate more samples. Figure 5.4 breaks down the steps to generate these samples and can be understood as a more detailed flowchart for block 1.2 in Figure 5.1 (again, the numbers in the flowchart concur with the listed steps).

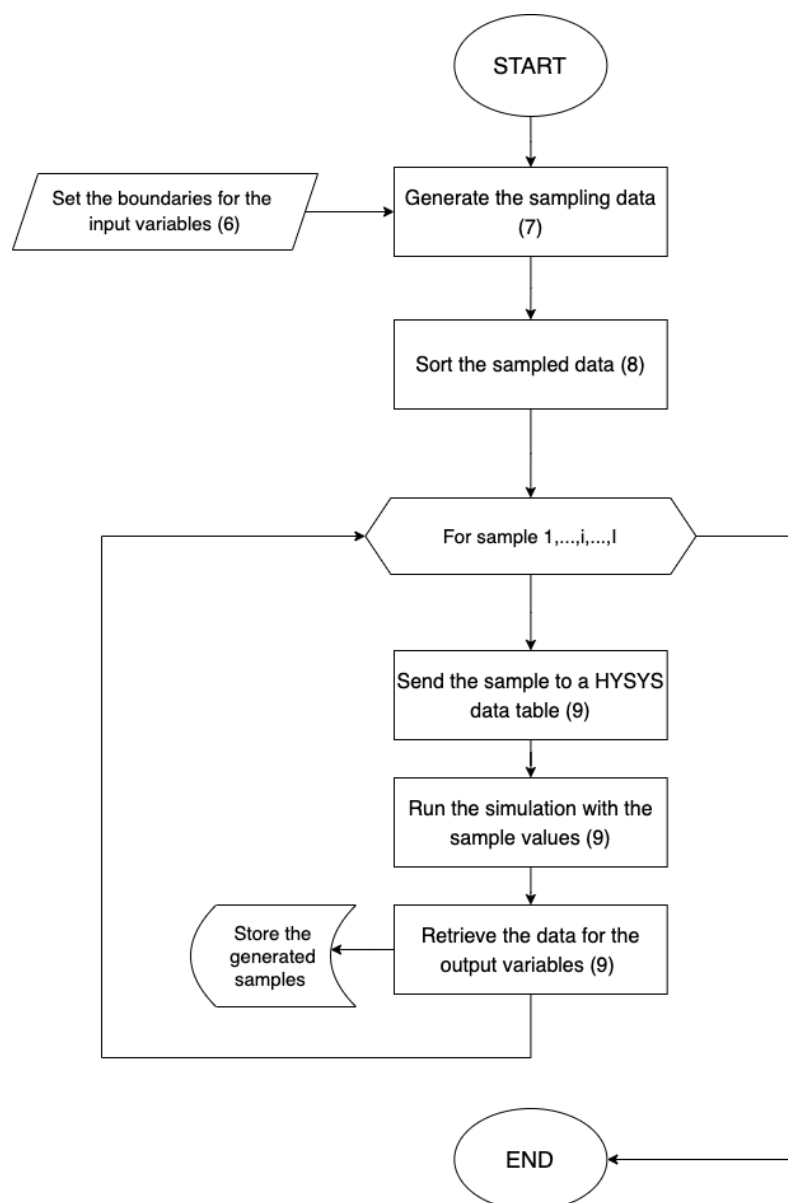


Figure 5.4. Flowchart of the Python-HYSYS connection.

Code the necessary functions to communicate the commercial process simulator with the preferred programming language. The selected programming language for this project is Python, but this methodology is also suitable for other programming languages. The scripts used for the Aspen HYSYS – Python communication are provided in the appendix A2.

For the sake of simplicity and better understanding, the functions used throughout the process to generate the data will be broken down into additional steps. There are six functions that must be defined in order to generate the sampling points and store them:

- i. Define a function to open the Hysys case (.hsc) and, optionally, another function to close it in case something goes wrong.

- ii. Define a function to read the information from a data table and convert it to a Python dictionary.
- iii. Define a function that edits/rewrites the values of your input variables in the HYSYS data table.
- iv. Define a function to sample data. The sampling method used for the case study is the Latin Hypercube Sampling, although other methods could be used. Pseudorandom methods are typically agreed to be more effective than pure random ones.
- v. Define a function to run a script that simulates the click on the 'Reset' button of a column. This function must be defined to reset the calculations in a distillation column in case one iteration (i.e., sample) does not converge.
- vi. Define a function to run a script that simulates the click on the 'Run' button of a column. This function must be defined to start the calculations in any distillation column.

On this project, the first four functions were retrieved from the literature (Martínez 2021), while the latter two were modified to work with more than one distillation columns.

- vii. To write the script that simulates the click on the 'Reset' or 'Run' button, open your case, go to the Aspen HYSYS menu "Customize", click on "Script", choose a directory, and click on the button "Save". Once you have chosen a name and closed the window, everything you do will be recorded and saved into the .scp script that you have just created. Then, double click on a column and click on "Reset" or "Run". Go back to "Script" and click "Stop recording". The actions performed will be automatically transcribed into the .scp script, written in WinWrap Basic v10 programming language (Safari 2022). The following lines can be copied into a .scp script to build the Reset and Run functions as described above:

```
Message "FlowSht.1/UnitOpObject.400(T1)/FlowSht.600" "Reset"
Message "FlowSht.1/UnitOpObject.400(T1)/FlowSht.600" "Run"
```

A line for each column must be copied, replacing only "T1" with the name of the desired distillation column in the simulation.

6.

Define the boundaries of the input variables and the number of samples (i.e., data points). The order of the bounds for the variables must be the same as the order of the variables defined in the data table in the simulation.

- viii. Write the "Tag" names you previously defined in step 5. Remember to use the same order as in the data table.
- ix. Define the highest and lowest value that you want to explore for your input variables. It is important to do a previous study, since there are laws that must be followed, such as

mass and energy conservation, among others, and depending on the decision variables one have chosen, not all the values will be appropriate. Omitting this previous study might result in a large increase of computational time, since the software will report an error and the column will not converge.

- x. Define the total number of samples. Increasing or decreasing this number will affect the computational time. Usually, the more samples one generates, the longer the computational time and the more accurate the resulting surrogate.

7.

Run the function specified in step iv. to generate the sampling data.

8.

Not only the order of the input variables must be studied, but also the order in which the samples are sent to the Aspen HYSYS data table, since both will impact the computation time and the number of unfeasible samples (Martínez 2021). Even though the points are sampled randomly, the order might be critical depending on the input variables. For example, the number of trays is one of the most important variables that defines a distillation column. In Aspen HYSYS, it is convenient to supply points with higher NT to the data table first. This is due the optimal feed automatically setting in HYSYS, once the value of NT is set to a lower value than 5 ($NT < 5$), the optimal feed tray does not go back to a higher tray when NT is modified again.

9.

Create a “for” (i.e., closed) loop for every sample point and run the functions defined before (from i to vii) to send the sample to a HYSYS data table, run the simulation and retrieve the data for the output variables. Figure 5.5 shows a more detailed version of this step, depicting the path that the sample points should follow depending on whether a single or multiple distillation columns are simulated concurrently.

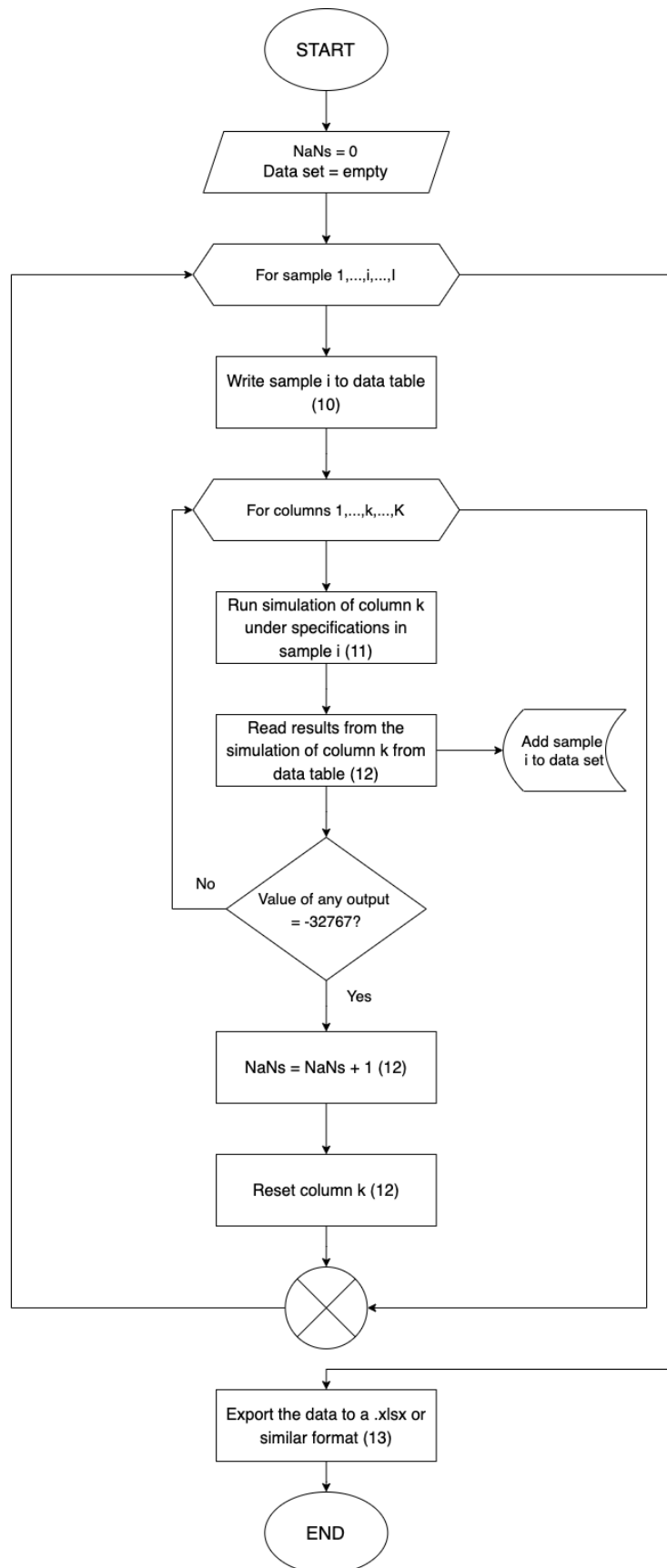


Figure 5.5. Flowchart of the data generation through sampling points.

Since one of the main objectives of this project is to compare a surrogate that encompasses all the simulation process against breaking down the simulation process into smaller parts (e.g., separate process units) to build multiple surrogates for each part, Figure 5.5 represents how the points should be sampled to generate data in a way that is valid for both scenarios.

10.

In this case study, where the behavior of two distillation columns is studied, a sampling point must be written (through automated Python functions) to the HYSYS data table.

11.

Next, the distillation column must be run so it can try to converge, and the results must be read and added to a data set.

12.

If the column has not converged, the Python function will read the value -32767 for the output variables. In this case, the distillation column must be reset. If multiple distillation columns are to be simulated, once one of the columns renders unfeasible, there is no need to explore the remaining columns because they will not converge neither. Therefore, the procedure can directly move to the next sample point.

13.

Export the generated data to a .xlsx format or similar so that it can be read by a programming language or library, such as Pandas. To do so, the “Tag” can be used as the names of the columns (see process viii in Figure 5.4).

To preprocess the data before it can be used to build the surrogate model, several steps need to be done, as depicted in Figure 5.6. Some of these steps requires the use of the tools specified in chapter 6.2, hence, the specific actions performed will be explained in the context of these tools.

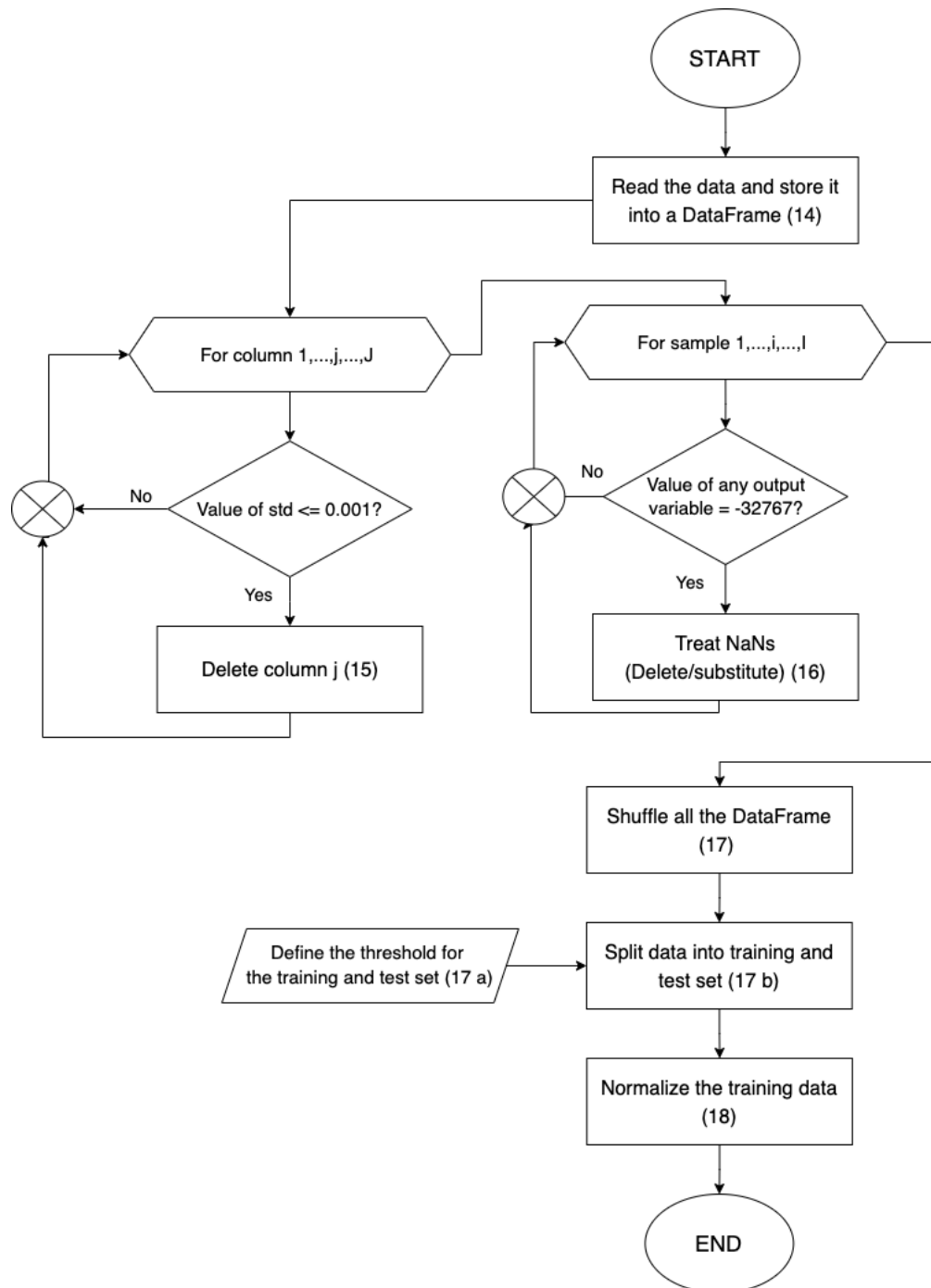


Figure 5.6. Flowchart of the preprocessing of the data.

14.

Import all the libraries that you want to use and store the data from the file generated by the Python code into a Pandas DataFrame (df). To check this step is performed correctly, type “df.head()” to visualize the DataFrame and make sure that the data shown is the same as the one saved in.xlsx or any similar format file.

15.

Inspect the data and look for columns (i.e., variables) that might not contain useful information (i.e., standard deviation is lower than 0.001, which is an arbitrary value that depends on the specific problem) and delete them. This could be an output variable that remains constant through all the examples (e.g., the inlet pressure in this case).

16.

Check for NaNs (Not a Number). When the column has not converged, the values read in HYSYS from Python display the number -32767.0. This is the equivalent of a NaN. There are many ways to treat these examples. In this project, the value of NaNs was relatively low, and it did not affect the final purpose of the study, therefore all rows containing this value were deleted.

17.

Shuffle all the DataFrame so that it can be later split into training and test sets randomly.

- a) Define a threshold (N) for the training and test set
- b) Split the data, selecting the first N points as training set and the remaining ones (all minus N) as test set. Note that this selection will be random owing to the previous step. Typical thresholds range from 70-80 % for the training set and 20-30% for the test set. In this project, a split of 80-20% is used.

18.

Normalize the data. This step can be crucial, especially for ANN because it generally speeds up the learning process and leads to a faster convergence. There are many different methods to normalize the data depending on the type of data. To check that normalization was applied correctly, data can be plotted using the “df.boxplot()” command (see Figure 8.2.2).

After the data is normalized, the artificial neural network can be built following the steps shown in Figure 5.7.

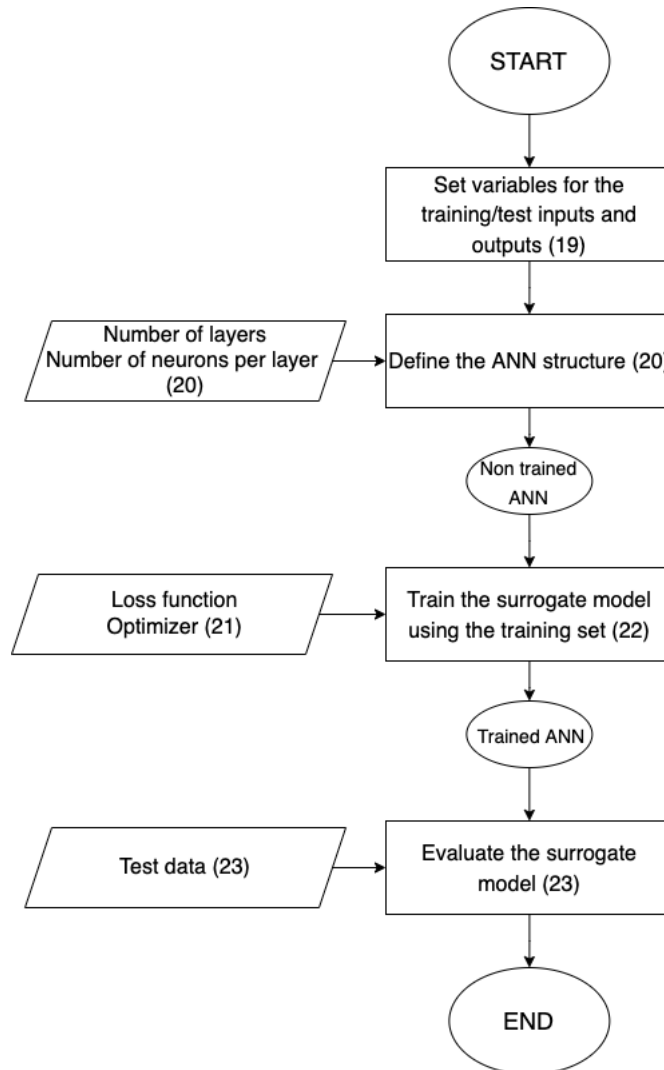


Figure 5.7. Flowchart of the specific steps to build, train and evaluate the surrogate models.

19.

Set 2 multidimensional variables for the training set and 2 multidimensional variables for the test set, one contains the inputs (e.g., the number of trays, the reflux ratio, etc.) and the other containing the outputs (e.g., the condenser duty, the reboiler duty, etc.).

20.

Build the ANN structure. Start by defining the input layer, whose number of input units should match the number of input (i.e., decision) variables. Next, define how many hidden layers the network will have, and how many hidden units each hidden layer will have. Both parameters can be tuned after evaluating the model. Choose which type of activation each hidden layer will have. Finally, define the output unit. If the purpose of the surrogate is to train the model and study the behavior of the process as a whole, only one output layer should be specified, containing as many output units as output variables are desired for the surrogate. If, instead,

the purpose is to train the outputs separately, to independently study disconnected parts of the process, there should be an individual output for each variable one wants to predict.

21.

Compilation is the final step in creating the model. To do so, specify the loss function, the optimizer, and the metrics. Chapter 4.1 provides better understanding of which loss function should be used for each type of problem. The optimizer is a module that optimizes the input weights by comparing the prediction and the loss function. The metrics are used to evaluate the performance for the surrogate model (i.e., how accurate its predictions of the output variables are compared to the simulation).

22.

Train the model using the method `model.fit()`. Specify the previously defined training parameters, the validation split, the number of epochs and the batch size. The last two can be tuned after evaluating the model. Further details on these settings are discussed later in chapter 8.3.

23.

Evaluate the model using the previously defined test set. It is useful to plot the Loss vs Epochs for the training and validation set. By doing this, it is possible to identify if the number of epochs used is correct and if the model is overfit or underfit. Another useful way to evaluate the model is to calculate the final loss and see if the value is low, which means the surrogate model is accurate, or if the value is high, which means the surrogate model is not accurate.

If the model has one output layer but multiple output units (as in this case study, see Figure 8.2.3), it is interesting to evaluate each output separately to see how the neural network is performing for each individual output (e.g., a good average performance may mask poor accuracy in specific outputs).

6. Tools

6.1. Commercial process simulator

To develop any surrogate model, large amounts of data are usually needed beforehand. For artificial neural networks, we need large datasets so that the model learns accurately the relations between the inputs and the outputs. However, since it is not common to find a big dataset for a specific industrial process, a previous simulation of the process is required to generate all the needed data.

There are many powerful simulation tools that can provide the necessary means to develop an accurate simulation of a chemical process. These tools are known as (commercial) process simulators and they contain chemical and physical properties of a large number of components, thermodynamic packages, equipment calculations, optimization and cost analysis tools among others. Some of the most known are listed below.

- Aspen HYSYS
- UniSim
- Aspen PLUS
- CHEMCAD

For this project, the software chosen to simulate the pyrolysis of polyethylene is Aspen HYSYS. This software is developed by AspenTech and it can perform calculations concerning mass and energy balances, vapor-liquid equilibrium, mass and heat transfer and chemical kinetics, amongst others. Although most of the commercial process simulators can perform these calculations as well, Aspen HYSYS was chosen for this project due its capacity to connect relatively easy with external sources such as Python, which will be very useful to automatically generate large amounts of data. An additional reason to choose Aspen HYSYS over alternative software was the previous experience with the tool.

6.2. Programming language and useful libraries

To generate and treat the data for the surrogate models, the Python programming language was used. Python is one of the most commonly used programming languages in the world. What makes it so popular is that it is a free open-source programming language that provides excellent library support and has one of the largest developer communities. Even though other programming languages such as MATLAB were considered, Python was finally chosen due its powerful machine learning libraries and the previous experience on that programming language. The most useful libraries used in this project are listed below.

- NumPy
- Matplotlib
- TensorFlow
- Pandas
- Keras

NumPy was developed for the creation and modification of multidimensional arrays, and it is specialized on numeric calculus and data analysis with large datasets. These features make it an

essential library for the proposed surrogate model, since it is used to create a method to generate a large dataset automatically once a first simulation process with Aspen HYSYS is finished (see step 7 in chapter 5).

Matplotlib is a comprehensive library for creating static, animated and interactive visualizations, which will allow us to represent the data gathered generated for further study.

Pandas is an open source data analysis and manipulation tool that will be utilized to explore, clean, and manage the data generated for the surrogate model, once it is available, in a format that Pandas can read such as .csv (see step 14 in chapter 5).

TensorFlow is an open-sourced end-to-end platform for machine learning. Within this platform we can find useful libraries such as Keras, which is an application programming interface (API) specialized on deep learning, i.e., artificial neural networks. These libraries will allow us to easily implement an ANN without having to specify all the equations stated in section 4.1.

7. Case Study: polyethylene waste pyrolysis

To generate the data needed to train the surrogate models, a process was simulated in Aspen HYSYS. Once a first (converged) solution of this simulation was completed, 10000 sampling points (i.e., simulations) were generated through an automated Python script. Next, the solutions for each sample point were retrieved from a HYSYS data table and stored into a dataset, which was later used for the surrogate models. Note that the process simulation is simplified since the main focus of this work is to develop a systematic framework to generate accurate surrogate models, rather than the simulation itself.

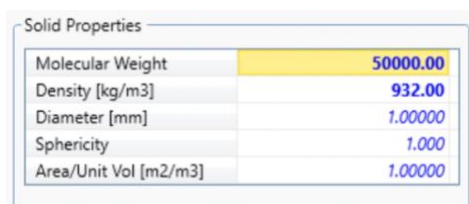
When pyrolyzing polyethylene (PE) at 900°C, it breaks into various smaller molecules, as given by the yields reported in (Honus et al. 2016). These data were used as a basis to develop a simulation process, starting with the pyrolysis reaction. For simplicity, components present at a mass fraction lower than 4% were not considered. The chosen components are listed in Table 7.1.

Table 7.1 Weight composition of PE pyrolysis at 900°C simplified.

Component	Formula	% Wt.
Methane	CH ₄	14.3
Ethylene	C ₂ H ₄	39.9
Ethane	C ₂ H ₆	4.5
Propene	C ₃ H ₆	16.7
1-butene	C ₄ H ₈	12.5
Cyclopentane	C ₅ H ₁₀	4.8

The sum of the weight compositions of the chosen components totals 92.7%, which indicates that although some of the components are neglected, the simulation will maintain an acceptable accuracy. Knowing the outcome composition in advanced, together with the lack of decision variables in the reactor (i.e., no kinetics are provided for this process making the reactor volume trivial, and the data available is valid for 900°C only) makes its presence in the simulation not strictly necessary. Nevertheless, similar works available in the literature still include the reactor as part of the simulation (Fivga and Dimitriou 2018; Somoza et al. 2020), and so was done here for the sake of completeness.

To start the simulation process in Aspen HYSYS, a component list must be created. Since the software does not have PE on its database, a hypothetical solid must be created. The data used from the bibliography (Honus et al. 2016) does not specify which type of PE they are referring to. Without loss of generality, low-density polyethylene (LDPE) was assumed. To create PE in Aspen HYSYS, a molecular weight of 50000g/mol (Miyagawa et al. 2007) and a density of 0.9352 g/cm³ (Serranti and Bonifazi 2019) was specified, as shown in Figure 7.1.

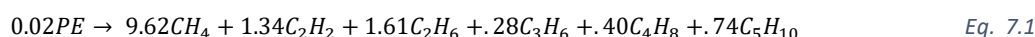


Solid Properties	
Molecular Weight	50000.00
Density [kg/m3]	932.00
Diameter [mm]	1.00000
Sphericity	1.000
Area/Unit Vol [m2/m3]	1.00000

Figure 7.1. Specified properties for PE.

The next step consists in choosing the thermodynamic fluid package, which could be one or multiple, depending on the components and its interactions through the process. Following the literature (Fivga and Dimitriou 2018; Somoza et al. 2020) of simulation processes with similar components, the thermodynamic fluid package PENG-ROBINSON was used.

What comes next is the definition of the reaction. Kinetics in polymers reactions are hard to determine or predict, as every molecule degrades in a different way depending on the temperature and pressure (Speight 2020). In the absence of better data, a conversion reaction that does not depend on temperature was specified. Conversion to olefins at high temperatures can be typically considered of 100% (Somoza et al. 2020). To define a conversion reaction, the stoichiometric coefficients must be specified. Since a simplification of the components was made before, to calculate the pseudo stoichiometric coefficients, the chosen components fractions were normalized, multiplied by 1kg of PE and divided by their respective molecular weight. The global pseudoreaction is represented in Eq. 7.1:



Pseudo stoichiometric coefficients from Eq. 7.1 were introduced for all components except for that of PE, whose coefficient was left empty to use the option “Balance” to calculate the exact pseudo stoichiometric coefficient for that case. This returned a value of 0.02, which is the result of dividing 1000g by its molecular weight of 50000g/mol.

Figure 7.2 represents the process flowsheet of PE pyrolysis to obtain the monomers formed at 900°C.

To obtain a complete working simulation, the data needed to specify the variables so every unit of the process is fully defined, was found in the literature or calculated with approximation techniques.

On this project, the surrogate will not encompass the whole simulation process, but just the two last distillation columns (T4 and T5), leaving the rest for future work. Hence, from this perspective, the simulation can be split into two parts: one part that will be fixed (i.e., constant), whose behavior will not be covered by the surrogate model, and another part that will be simulated many times (i.e., for several values of the decision variables) in order to capture the response of the output variables to different values of input (decision) variables. For the former, the values used will be provided (i.e., they are the same for every simulation), while, for the latter, the range of values used in different simulations will be reported later in Table 8.2.

The process starts by feeding 1000kg/h of PE at 25°C and 1bar (without loss of generality, these values were chosen arbitrarily) which enters to the conversion reactor (i.e., a furnace) that operates at 900°C and 1 bar, where the pyrolysis takes place. The gas leaving the reactor enters a series of three compressors. After each compression, the gas is cooled down to reduce the temperature and the energy consumption of the next compression. This system is completely based on the literature (Somoza et al. 2020), since its design is out of the scope of this project. The gas stream enters the first distillation column at 30 bar and 40°C.

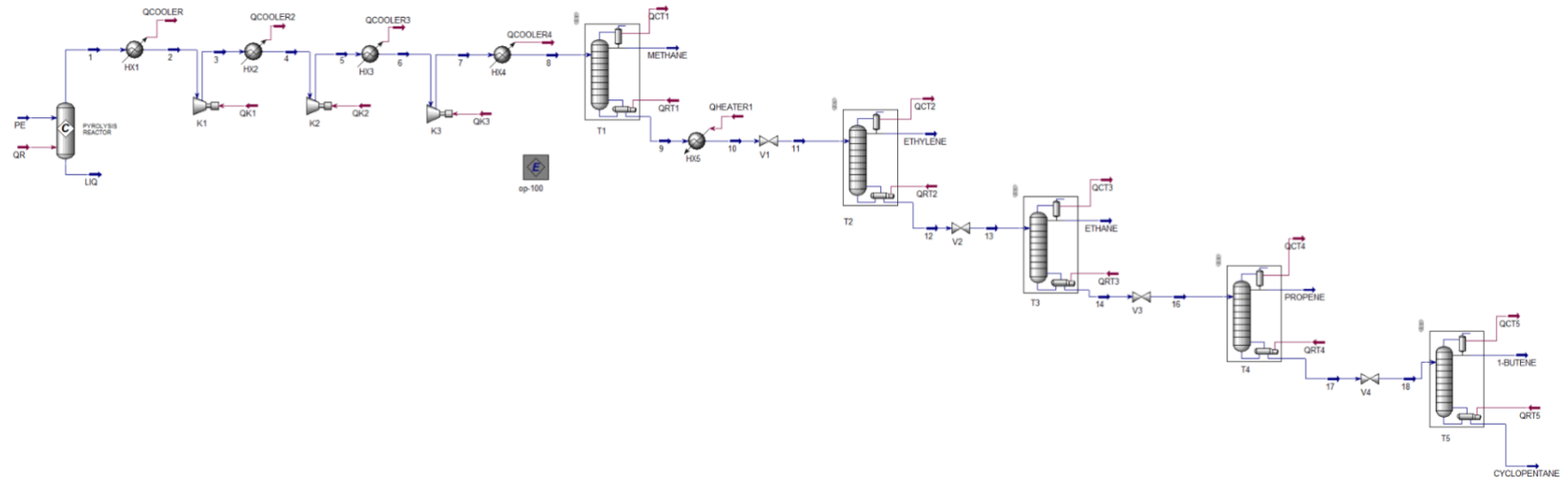


Figure 7.2. Flowsheet for the PE pyrolysis.

For the separation sequence, all components will be separated following the direct sequence heuristic which is used in similar processes. There are 5 distillation columns in the process, and each one recovers a different product at the top: methane, ethylene, ethane, propene and butene respectively. The last column (T5) also recovers cyclopentane at the bottoms. Each bottom stream is fed to the next column, after going through a control valve (V1-V4) that adjusts the pressure to that of the next column. Pressures for the distillation columns are retrieved from the literature (Somoza et al. 2020).

As laws of thermodynamic dictate, mass and energy must be conserved. In distillation processes, equilibrium laws must also be satisfied. At the liquid-vapor equilibrium, the temperature (T), the pressure (P), the liquid molar fraction (x_i) and vapor molar fraction (y_i) are constants. General equilibrium law can be expressed as shown in Eq. 7.2:

$$f(x_i, y_i, T, P) = 0 \quad \text{Eq. 7.2}$$

The number of decision variables that must be specified to define the equilibrium between two phases in systems where there are not chemical reactions is defined by the Gibbs phase rule, shown in equation Eq. 7.3:

$$F + L = C + 2 \quad \text{Eq. 7.3}$$

Where F is the number of phases, L is the number of degrees of freedom and C is the number of chemical components. A distillation process for a mixture of two components is named binary distillation, otherwise is called multicomponent distillation. In a multicomponent distillation each distillation has two components of interest, whose fractional recoveries at the top and bottom of the column can be specified. These components are called key components. The most volatile key is called light key (LK), and, in primary calculations, every other component in the mixture more volatile than the light key (i.e., lighter than light key) is assumed to leave the column by the top. The least volatile key is called heavy key (HK). Opposed to the light key, every component with less volatility than the heavy key (i.e., heavier than heavy key) is typically assumed to leave the column at the bottoms, as a first approximation.

Every distillation column has a total of 5 degrees of freedom. In a “pseudo-binary” system there are multiple components, but only 2 components are considered (HK and LK), as the other ones cannot have any decision variable associated (i.e., they are completely specified by the HK and LK). Additionally, every distillation column has 3 degrees of freedom which are “pseudo-determined” (i.e., related but not restricted) by 2 specifications. In this case, the two specifications selected for each shortcut column were the molar fraction of the HK and the LK at the distillate and at the bottom, respectively.

For distillation columns T1 to T3, which will not be part of the surrogate, the necessary parameters to simulate them with rigorous column model were found using shortcut columns, which are useful to find acceptable approximations. Additional (optional) parameters (e.g., the temperatures at the condenser and the reboiler) were also retrieved from these shortcut columns to help the more rigorous model converge faster (i.e., we provide a reasonable starting point). For this case study, without loss of generality, the purity of each product to recover at the distillate of each column is

at least 99.5% (and the same for cyclopentane, which is recovered at the bottoms of T5). The other 0.5% that will not be recovered corresponds to the molar fractions of the HK and LK at the distillate and bottoms respectively (i.e., the specifications for the shortcut columns). The parameters found with these shortcut columns are shown in Table 7.2.

Table 7.2. Approximated parameters found with shortcuts needed to define the rigorous distillation columns T1, T2, T3.

Parameters	T1	T2	T3
Number of trays	23	83	32
Optimal feed stage	13	53	12
Minimum reflux ratio	3.66	3.47	3.17
Distillate flow (kg/h)	154.00	429.30	50.12
Condenser temperature (°C)	-95.90	-20.68	-15.01
Reboiler temperature (°C)	6.977	61.02	61.16

The shortcuts were later replaced by rigorous distillation columns using these parameters as starting points to find a convergence solution (after that, the reflux ratio and the distillate rate will be replaced as specifications for the ones aforementioned in this chapter (i.e., the molar fraction of the HK and the LK at the distillate and at the bottom, respectively), and a new convergence solution will be found). The minim reflux ratio was multiplied by 1.2, a common value for the optimal reflux ratio where the balance between the cost of the steel and the cost of energy is expected to be near optimal. As we increase the value of the reflux ratio, less trays (less total cost for the steel) and more energy (more total cost for the energy) are needed to achieve the same purity in the products.

It is worth noting that column T1 did not converge using 1.2 times the minimum reflux ratio. After researching in the literature, it was found that the reflux ratio for similar distillation columns (i.e., with similar components) was 15.4 (Somoza et al. 2020), which was significantly higher than the original value tested (4.4). Hence, the reflux ratio was first updated to 15.4, however after specifying only the molar fraction of the HK and the LK at the distillate and at the bottom, respectively and running the column again, the distillation column finally converged at a reflux ratio 11.4.

At this point the “fixed” part of the simulation ends, where no decision variables are present. This “fixed” part can be used as a starting point for increasing the scope of the surrogate model, which at the moment, only covers distillation columns T4 and T5.

To simulate a first converged solution of distillation columns T4 and T5, which will be used to generate more data, the necessary parameters were found using the same method described before, i.e., using shortcut columns to find the approximated parameters to define a rigorous distillation column. These parameters are listed in Table 7.3.

Table 7.3. Approximated parameters found with shortcuts needed to define the rigorous distillation columns T4 and T5.

Parameters	T4	T5
Number of trays	29	17
Optimal feed stage	13	10
Minimum reflux ratio	1.64	0.26
Distillate flow (kg/h)	180.30	134.30
Condenser temperature (°C)	19.79	-6.77
Reboiler temperature (°C)	81.66	46.53

8. Results and discussion

On this section, the part of the simulation process studied on this project, as explained in chapter 7, will be used to generate 10000 sampling points. Next, these data will be preprocessed and used to build the surrogate models that aim to predict the behavior of distillation columns T4 and T5 within the studied range of values (specified in Table 8.2). To this end, a surrogate model that encompasses both distillation columns will be build and studied first (holistic surrogate). Next, a modular surrogate model with the same scope (columns T4 and T5) will be trained and compared with the holistic surrogate in terms of accuracy and computational time. The modular surrogate will contain two independent surrogate models, one for each distillation column, that will be connected through some variables. Specifically, the bridge variables are the mass flow of the stream connecting the two columns and the molar fractions of the light and heavy key components (for column T5) in this stream. In the modular surrogate, these variables will be predicted as outputs by the surrogate that emulates the behavior of T4, and then, will be used as inputs for the surrogate model that emulates the behavior of T5.

8.1. Data generation

The decision variables (inputs) chosen to study the behavior of columns T4 and T5 are the number of trays (NT), the distillate rate (Dflow) and the reflux ratio (RR). As explained earlier in chapter 5, these variables have been sent from the HYSYS simulation to a HYSYS data table named 'ProcData1' to allow connection with Python scripts. The 'Access Mode' for these variables is specified as Read/Write. On the other hand, output variables for each of these columns are, for the distillate stream, the mass flow and the molar fraction of the light key, for the bottoms stream, the mass flow and the molar fraction of the heavy key, for the bottoms stream of T4 and for the inlet stream of T5, the pressure and the temperature respectively, and lastly the heat flows for the condenser and reboiler. These variables are also sent to the same data table and are set to 'Read' as their 'Access Mode'. These input and output variables will be later used to train the surrogate model. Table 8.1 depicts all input and output variables, along with a 'Tag' for each of them.

Table 8.1. Input (in blue) and output (in black) variables information specified at the data table. Values shown correspond to those of the nominal simulation; they will be later replaced by the values of each sample point.

Object	Variable	Value	Units	Tag	Access Mode
Main Tower	Number of Stages	29		NT_T4	Read/Write
T4	Spec Value (Distillate Rate)	180.2	kg/h	D_T4	Read/Write
T4	Spec Value (Reflux Ratio)	1.964		RR_T4	Read/Write
Main Tower	Number of Stages	10		NT_T5	Read/Write
T5	Spec Value (Distillate Rate)	108.029	kg/h	D_T5	Read/Write

T5	Spec Value (Reflux 1.00037 Ratio)	RR_T5	Read/Write
PROPENE	Phase - Mass Flow 180.197 kg/h (Overall)	M_PROPENE	Read
PROPENE	Overall and Phase Comp Mole Frac 0.998611 (Overall-Propene)	PROPENE	Read
QCT4	Heat Flow Condenser 187375 kJ/h	QC_T4	Read
QRT4	Heat Flow Reboiler 176505 kJ/h	QR_T4	Read
17	Phase - Temperature 81.8231 C (Overall)	T_T4	Read
17	Phase - Pressure 10 bar (Overall)	P_T4	Read
CYCLOPENTANE	Phase - Mass Flow 78.364 kg/h (Overall)	M_CYCLOPENTANE	Read
CYCLOPENTANE	Overall and Phase Comp Mole Frac 0.60873 (Overall-Cyclopentane)	CYCLOPENTANE	Read
1-BUTENE	Phase - Mass Flow 108.028 kg/h (Overall)	M_BUTENE	Read
1-BUTENE	Overall and Phase Comp Mole Frac 0.999936 (Overall-1-Butene)	BUTENE	Read
18	Phase - Temperature 3.68248 C (Overall)	T_T5	Read
18	Phase - Pressure 1 bar (Overall)	P_T5	Read
QRT5	Heat Flow Condenser 50746.5 kJ/h	QR_T5	Read
QCT5	Heat Flow Reboiler 84494.9 kJ/h	QC_T5	Read

17	Phase - Mass Flow (Overall)	186.392	kg/h	M_BTTMS_ T4	Read
17	Overall and Phase Comp Mole Frac (Overall-1-Butene)	0.764778		X_BUTENE_ BTTMS_T4	Read
17	Overall and Phase Comp Mole Frac (Overall-Cyclopentane)	0.235211		X_CYCLOPE NTANE_BT TMS_T4	Read

For the decision variables, it is important to note that they have been sorted in a particular way intendedly. The three parameters corresponding to column T4 have been put first. T4 parameters are placed ahead of T5 because the bottoms of T4 are the feed of T5. This will allow us to prevent performing calculations for column T5 when column T4 does not converge, saving time in the process. The order of these three parameters for each column is also important. The number of trays must be the first parameter, followed by the distillation flow and lastly the reflux ratio. This combination will perform faster and will generate less unfeasible samples (Martínez 2021).

To be able to compare a holistic surrogate versus a modular surrogate, the feed of each distillation column must be related to the output from the distillation column upstreams, i.e., when generating data for every module, the feed of the distillation columns must change as it would if a unique surrogate was to be generated. Given the scope of this project, the first distillation column (T4) has a fixed feed, therefore only the next distillation column feed must be changed (T5). On this case study, the sequence of the distillation train is a direct sequence, hence, the bottoms of a distillation column should be related to the feed of the next column. If the sequence was not direct, it is probable that also the top stream of some columns would have to be related to the feed stream of the next one.

The necessary functions to open and close the HYSYS case (.hsc), run and reset each column, and read and write to/from the data table were defined. To generate the sampling data, Latin Hypercube Sample (LHS) was used, which is a more attractive alternative than simple random sampling since it simultaneously stratifies on all input dimensions (Loh 1996). To define the boundaries for the input variables, mass conservation law was considered for the distillate rate, and the values for the number of trays are taken arbitrarily using as starting point the nominal converged simulation. For the reflux ratio, low values (in the range of 1 - 2.5) were chosen because it usually requires less iterations for the column to converge. Table 8.2 shows the boundaries taken for each column.

Table 8.2. Boundaries of the input variables for column T4 and T5.

Column	NT (adimensional)	Dflow (kg/h)	RR (adimensional)
T4	30 - 23	190 - 120	2.5 - 1.5
T5	17 - 10	130 - 70	1.5 - 1

A total of 10000 samples are run, as done in similar papers (van Hoof and Vanschoren 2021). Figure 8.1.1 depicts the sampling map for the Dflow and RR variables for column T4.

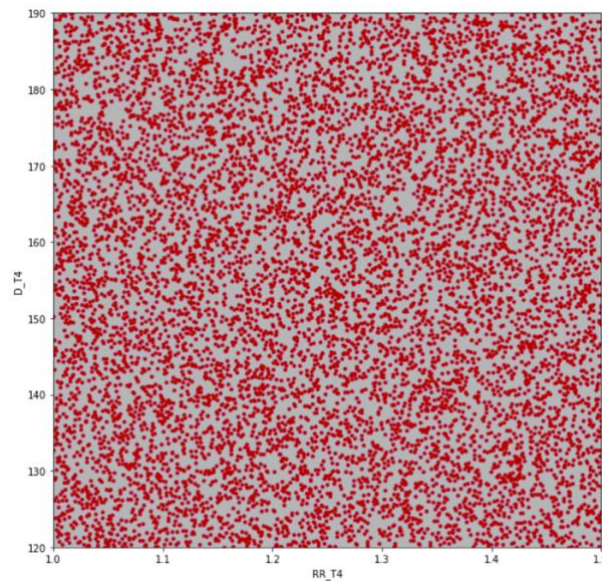


Figure 8.1.1 Sampling map for Dflow and RR variables of column T4 resulting from the LHS method.

All the values are created pseudorandomly and stored in a NumPy array. Sending the samples in a random order will increase the computational time and the number of unfeasible samples (Martínez 2021), since HYSYS distillation columns operate by iteration, and if a reset is not performed, it takes the values of the last iteration as starting point. If a reset were performed, there might be samples where the column does not converge, but it would converge if it started from a closer starting point. Reset is applied only when the column has not converged (see Figure 5.5). To minimize this problem, the parameters from each column were first sorted in decreasing order for the number of trays, then the reflux ratio was sorted also in decreasing order, and finally the distillate rate was sorted in increasing order.

The sampling was also run for column T4 and T5 separately, using the same boundaries for each input variable and the same number of samples as in the holistic sampling (i.e., 10000 points for each column). This experiment aims to explore if a *divide-and-conquer* strategy, breaking the process into smaller modules, could decrease the computational time while maintaining an acceptable number of unfeasible samples and an acceptable accuracy for the resulting surrogate models. In this specific problem, all samples that do not violate the mass balances, should be feasible samples, but HYSYS distillation columns do not always converge with the iterations specified, by default 10000, due to complex calculus. Since the number of unfeasible samples in similar works is not significative (Martínez 2021) and lowering the distillation column iterations can greatly decrease the computational time, a first approach for all simulations was made, setting the value of maximum iterations to 200 for each distillation column. The results for this test are shown in Table 8.3.

Table 8.3. Computational times and unfeasible samples for a distillation train of 2 columns as global and individual.

	Column T4	Column T5	Columns T4+T5
Unfeasible samples (%)	5.9	0.07	6.31
Computational time	2h 14min 14 sec	2h 42min 15sec	7h 19min 2sec

The computational time is lower (48.08%) when sampling each column separately (4h 56min 29sec in total) and has less unfeasible samples (5.97% vs 6.31%), compared to the simultaneous sampling of columns T4 and T5. However, in this test, distillation column T5 had a steady feed when it was sampled separately, i.e., it did not receive the new data generated for every sampling point in distillation column T4, as it did when the sampling was for both units at the same time. Changing the feed composition of an individual column might be more challenging when it is user-specified, as HYSYS opens a window every time a composition value is changed. To confirm the change in the composition, the “ok” button must be pressed. This could be done by a script, similarly to the run and reset scripts for columns. In this case, the function that writes the samples to the data table should be changed, so that every time it writes a composition value in the data table it runs this confirmation script before writing the next value, otherwise it will not work. This was not done on this project due to time limitations.

Since the script to change the compositions was not done, in order to better represent the change in the feed for the modular surrogates, the mass flow at the bottoms of column T4 and the molar fractions of the light and heavy key (of the next column) in that stream were taken as outputs when generating the data for the holistic surrogate. For the modular surrogates that will be developed next, only the data obtained with the global simulation, i.e., simultaneous modifications of decision variables in column T4 plus column T5, will be used to train the surrogates. This means we will not be able to benefit from the time advantage shown in Table 8.3, although, in principle, it would be possible if writing the script described in the previous paragraph. Regardless of this, data will be used as if it had been sampled independently for columns T4 and T5 separately. Hence, data for the three variables of the stream connecting columns T4 and T5, will be considered output variables for the modular surrogate of T4, and as input variables for the modular surrogate of T5. As mentioned earlier in this chapter, if the sequence of the distillation train was not a direct sequence, variables from the top stream of the predecessor distillation column would also be read and used to connect the modular surrogates, in order to better represent the change of the feed of the columns downstream.

The generated data was saved on a .xlsx format so it could be read by the Pandas function ‘pd.read_excel()’.

8.2. Pre-processing and cleaning

Data will likely be imperfect, it may contain redundancies (e.g., if we retrieved as outputs the mass flow at the bottoms of column T4 and the mass flow at the feed of T5), inconsistencies (e.g., if we retrieved the heat flows in kJ/h for column T4 but used different units, as kJ/s for column T5), or missing values (e.g., when a sampling point does not converge) that would make our model not to work properly. In order to enhance performance, data preprocessing must be applied. Figure 8.2.1 shows a box plot (box and whisker plot for each column on the Dataframe, where the box extends from the lower to the upper quartile values of the data, with a line at the median) before any data treatment (only all the values of P_T4 were divided by 10, so it matches better with the scale of the other variables and the plot is more easily readable).

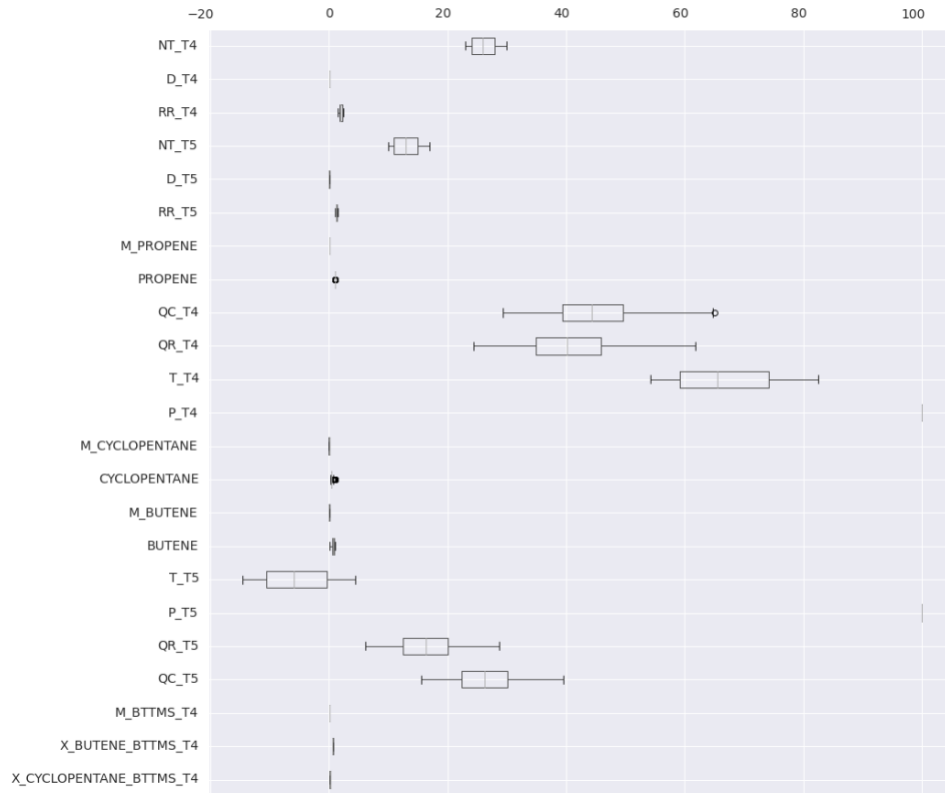


Figure 8.2.1 Box plot of the data before any treatment.

To see which output variables remain constant, the standard deviation of each column (i.e., variable) in Table 8.1 is calculated. If the standard deviation is lower than 0.001 (arbitrary value, might change depending on the problem), as it was for P_T4 and P_T5, we can confirm it is a constant value, hence both columns have been deleted from the DataFrame. Other control variables, as the mass flows at the distillate (M_PROPENE and M_BUTENE in Figure 8.2.1) can also be deleted now.

Next, we must check for unfeasible samples. When a column does not converge, the value read by the Python function is equal to -32767 for any output variable (of the same sampling point). Bearing this in mind, the number of unfeasible samples can be determined by exploring all the DataFrame to see which rows contain a value of -32767. Next, only one column counting how many times this value does appear in a single column of the DataFrame. Next, using Pandas, only one column is needed to delete these NaN. This value can also be counted using Pandas, and it should concur with the value given by the Python function as the data were generated. In this case study, every unfeasible sample (i.e., row) was deleted, because they only represented a 6.31% of the total sample points. In addition, if more data were needed, these could be easily generated.

After the data cleaning, all the DataFrame was shuffled. This step is necessary because when data is split into a training and test set, the function will take the first N samples as training and the rest as a test. Bearing in mind that we initially sorted the data in order to expedite calculations and improve convergence, if data were not shuffled here, the training set will not cover the whole intended spectrum of values as defined by the boundaries of the input variables. Hence, shuffling is done to help the model prevent overfitting and to ensure that batches are representative of the whole data set.

Once shuffle was done, the threshold for the training and test set were defined. Typical thresholds used to split the data do a 70% split for the training set and 30% for the test set (Liu and Cocea 2017) and so is done in this project.

Next, data normalization must be applied to the training set. This step is crucial to obtain good results as well as to significantly fasten calculations (Sola and Sevilla 1997). To normalize the data, the standard scaler technique was employed. This approach standardizes the features by subtracting the mean and scaling to unit variance Eq. 8.1.

$$z = \frac{x_i - \mu}{\sigma} \quad \text{Eq. 8.1}$$

Here, μ is the mean of the training examples and σ is the standard deviation of the training samples. Eq. 8.1 is applied for each variable independently. To check if normalization was applied correctly, the “df.boxplot()” function can be used to plot the data and check the outliers. Figure 8.2.2 shows the boxplot of the data after normalization.

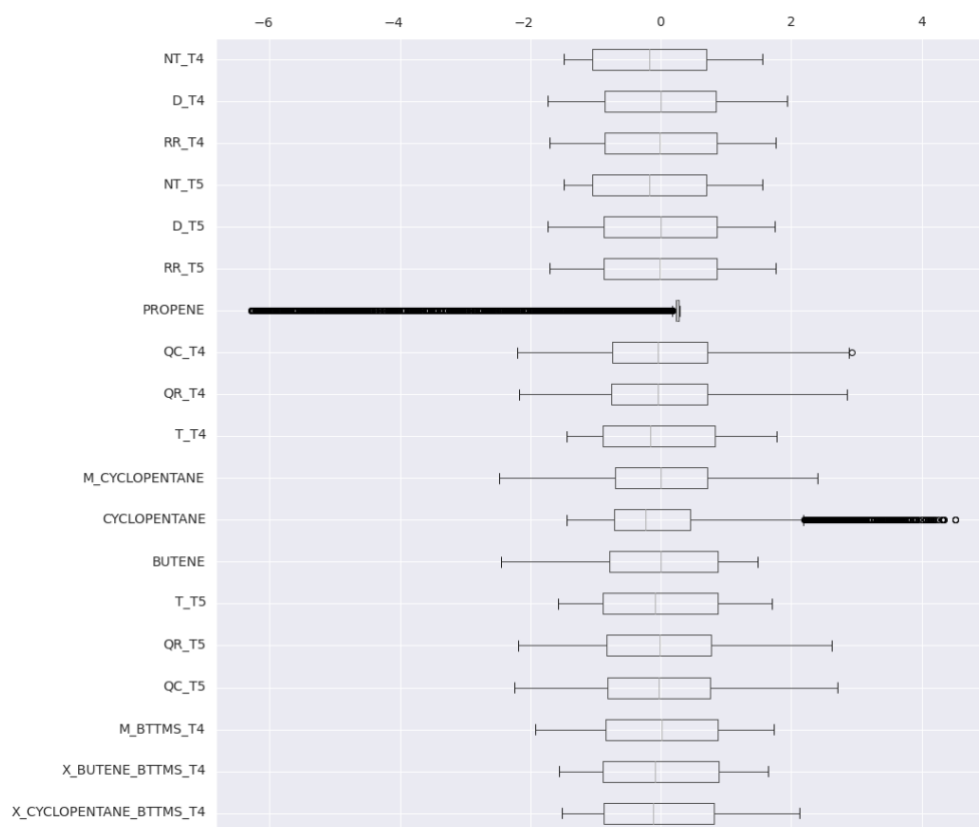


Figure 8.2.2 Box plot of the data after normalization.

The box plot shows that the normalized molar fractions of propene and cyclopentane, at the distillate and bottoms of column T5 respectively, have many outliers. These outliers will not be deleted to explore the capabilities of the surrogate models on predicting these and the other variables correctly (i.e., even if the model does not predict well these two variables, it could still

predict the other variables with an acceptable accuracy for the same range). This will be further studied later on this chapter.

Once the data were ready, parameters x_{train} , x_{test} , y_{train} and y_{test} were defined. These parameters contain the data of the inputs (x) and outputs (y) for the training and test sets, respectively. In the absence of the script that would allow us to change the feed composition of a single column, we use the same dataset to build the global surrogate and the two surrogates in the modular surrogate. The only difference is that the test data (x_{test}) for the modular surrogate that predicts the behavior of T5 will contain three additional inputs (molar fractions of the heavy and light key and mass flow at the bottoms of T4) that will come from the predicted output of the modular surrogate that predicts the behavior of T4. Figure 8.2.3 depicts the different data used for every surrogate model.

Global surrogate

Modular surrogate

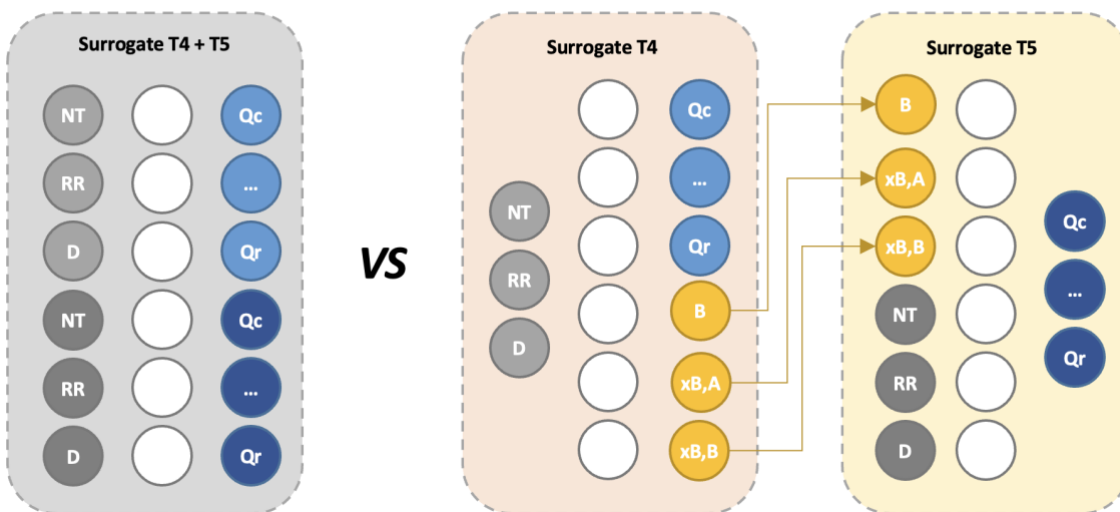


Figure 8.2.3 Schematic of the input data for the global and modular surrogates.

Note that, from a global perspective, the two approaches have the same number of inputs (six, in grey; three for distillation column T4 and three for distillation column T5) and outputs (thirteen, in blue; four for distillation column T4 and six for distillation column T5). In the modular surrogate, column T4 has three additional outputs (yellow), but these are only “internal”, as they are used as inputs for column T5. One would have these kinds of connections for every sequential units in the simulation.

8.3. Building the architecture of the surrogate models

From this point onward, the results will be explained for every approach (global vs modular) separately, since every surrogate model will be built differently and one of the main objectives of this project is to compare their performance.

First of all, the structure of the ANN must be defined. Figure 8.3.1 depicts the ANN structure for the different surrogate models with information about their shape.

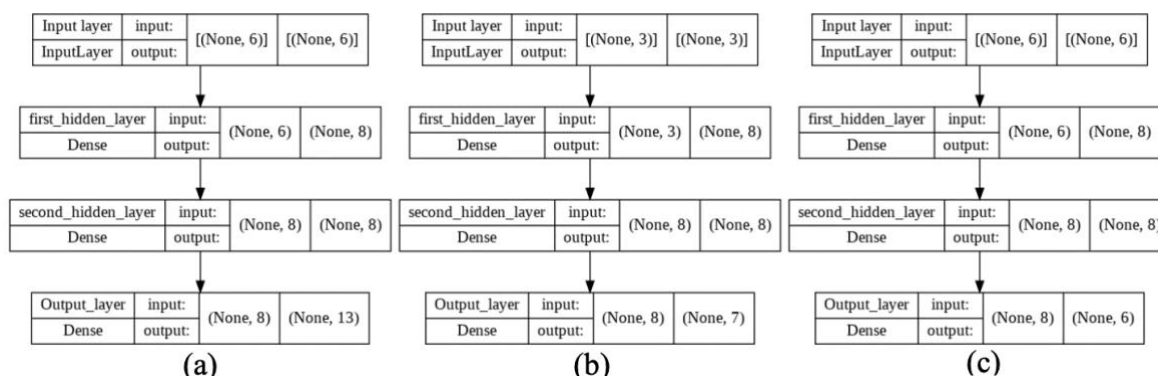


Figure 8.3.1 Surrogate models architectures, (a) global surrogate (b) T4 module surrogate (c) T5 module surrogate

The input layer must have the same input units as the number of input variables, i.e., the same number of variables as in x_{train} and x_{test} . The global surrogate model contains 6 input variables; hence, we need 6 input units (Figure 8.3.1(a)). Conversely, the surrogate model of T4 only needs 3 input units because this surrogate only contains 3 input variables (Figure 8.3.1 (b)); meanwhile, the surrogate model of T5 has 6 input units, since it will take as input variables the 3 output variables from the surrogate model of T4 to define its feed (Figure 8.3.1(c)).

For the hidden layers, a constant value of 2 hidden layers is considered arbitrarily for every surrogate model. These are named `first_hidden_layer` and `second_hidden_layer` in Figure 8.3.1 and, in a first approach, each contains 8 hidden units and the ReLU (Rectified Linear Unit) activation function. The number of hidden units will be further studied later in this chapter.

Finally, the output layer is defined with the same number of output units as the number of output variables, i.e., the same number as variables in y_{train} and y_{test} . This corresponds to 13 output units for the global surrogate, and 7 and 6 output units for the modular surrogates of columns T4 and T5, respectively.

Once the structures are defined, the models must be configured for training, specifying the loss function (i.e., the function used to evaluate the ANN performance, calculating the difference between the predicted output with respect to the real output), the optimizer (i.e., the algorithm that will seek the value of the weights and the learning rate to maximize the fit), and the metrics (i.e., the specific value for the loss function, such as the mean of squared errors). For the loss function, since this is a regression problem, the mean of squared errors was used, as stated in chapter 4.1. For the optimizer, a stochastic gradient descent method based on adaptive estimation of first-order and second-order moments (Adam) was used. The metric chosen to evaluate the model is the mean of the squared errors (mse).

Once the model is compiled, it must be trained specifying the number of epochs (the number of times that the model will work through the entire dataset), the batch size (the number of samples to work through before the models internal parameters are updated), the training data and the validation split. Without loss of generality, the number of epochs is first set to 50, and the batch size is initially set to 300, although the number of epochs will be further studied after training the models. The training data was defined before as x_{train} and y_{train} . The validation split is a fraction of the training data that will be used as validation data to tune the parameters (weights and learning rate) of the ANN. It is important that the validation set and test set are different, since the ANN is tuned with the validation set, hence it is not completely unbiased to these data, whereas the test set is completely unseen data for the ANN. The validation set will not be trained and will evaluate

the loss and the metrics defined at the end of every epoch. The value chosen for the fraction associated to the validation data on these models is 20%, as it is a common value (Vabalas et al. 2019).

Figure 8.3.2 depicts a plot of the loss at every epoch for the training and validation sets of the holistic surrogate model, which is useful to understand how the model is performing.

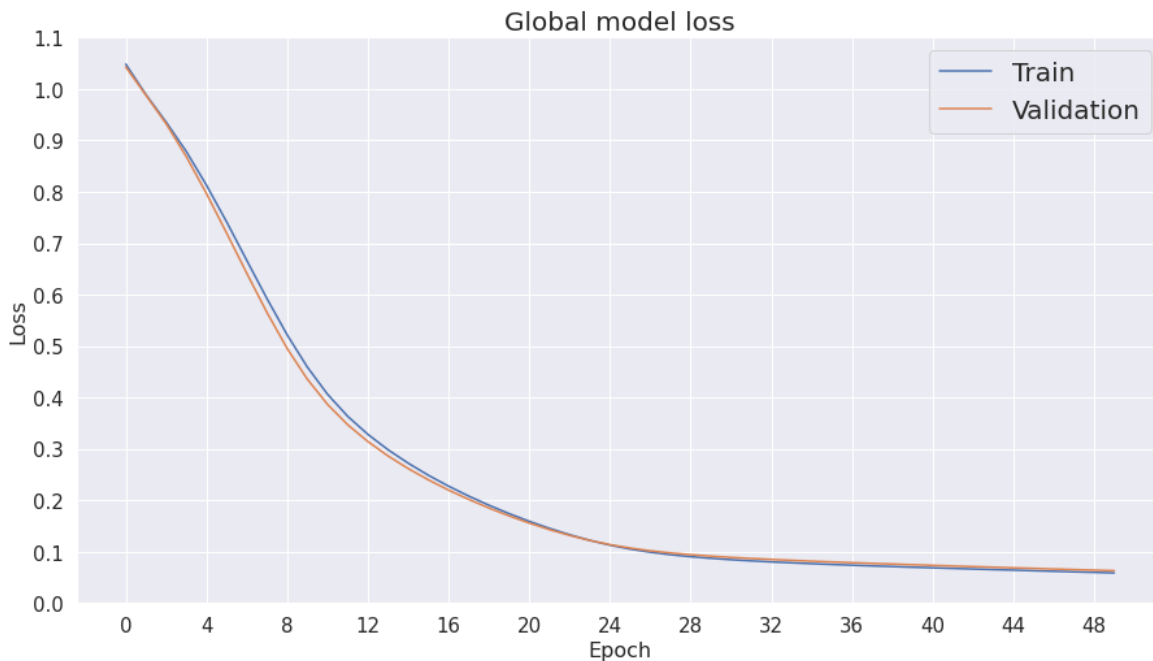


Figure 8.3.2 Model loss at every epoch for training and validation sets of the holistic surrogate model.

As we can see, both curves decrease to a point of stability, with a small gap between them, which means that the model is learning well (Brownlee 2019). The curves start to flatten around 20-24 epochs, so we could reduce the number of epochs way below the initial setup of 50, which would decrease the computational time to train the surrogate.

Figure 8.3.3 and Figure 8.3.4 depict the loss at every epoch for the training and validation sets of the T4 and T5 modular surrogate, respectively.

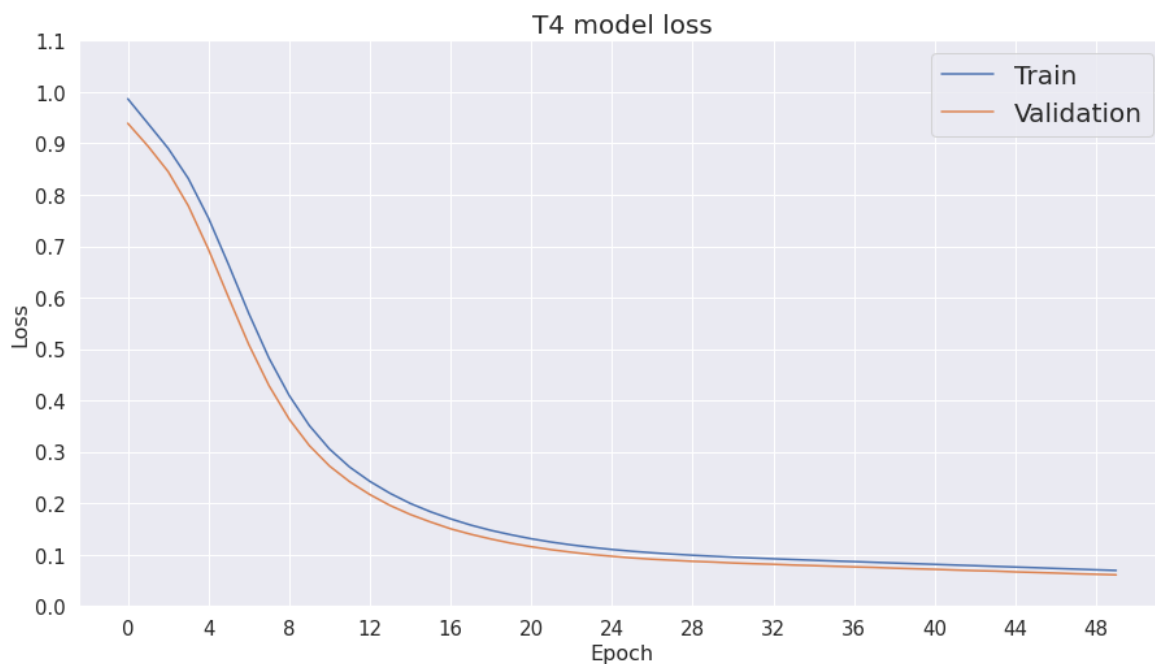


Figure 8.3.3 Model loss at every epoch for training and validation sets of the T4 modular surrogate model.



Figure 8.3.4 Model loss at every epoch for training and validation sets of the T5 modular surrogate model.

In both figures, the training and validation curves decrease to a point of stability with a small gap between them, which indicates a good fit. In this case, the curve in Figure 8.3.3 starts to flatten around 16-20 epochs, while in Figure 8.3.4 it flattens around 20-24 epochs.

Finally, we can evaluate the model with the test set (i.e., validate). To do so, the `model.evaluate()` function was used, introducing `x_test` and `y_test`, previously defined, as arguments. This returns the loss and metrics values for the test set, in this case, the mean of squared errors (mse). Note that these errors correspond to the average among all sample points and outputs for each

surrogate. In Figure 8.3.5, we show these errors for the holistic and the modular surrogates (i.e., 13 output variables estimated from six inputs in both cases).

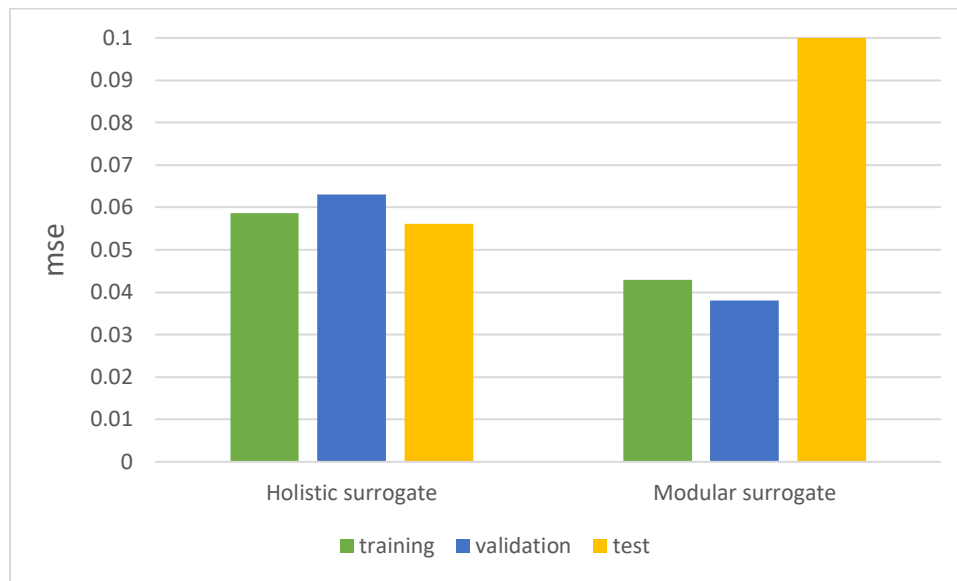


Figure 8.3.5 Mean of the squared errors for the holistic and modular surrogates.

Note that the mse value for the test set of the modular surrogate has an actual value of 0.9412, but the bar was cut so that other bars could become more easily readable (i.e., we zoomed in the y axis). The mse of the holistic surrogate does not show big differences between sets (<0.007), with a low average value. This indicates that the difference between the predicted output and the real output is small and, hence, that the surrogate should predict most output variables with an acceptable accuracy (the individual behavior of the different outputs will be studied later in this chapter, see Figure 8.3.8). On the contrary, the mse of the test set of the modular surrogate differs by almost 25 times from the value of the validation set. To explore the cause of this error, the mse for the sets of the two modules of the surrogate (i.e., the surrogate for column T4 and the surrogate for column T5) will be analyzed individually. Figure 8.3.6 depicts the mse for these surrogates.

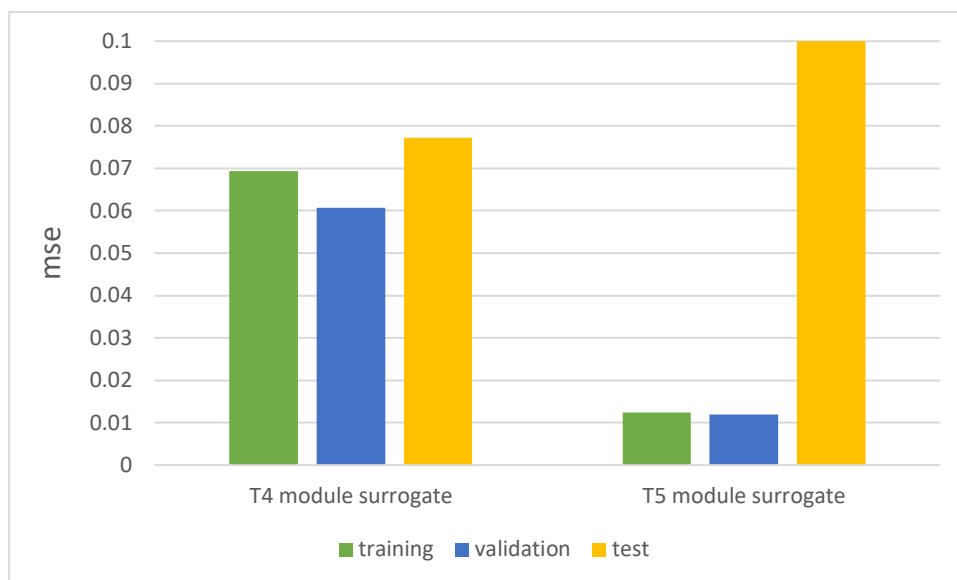


Figure 8.3.6 Mean of the squared errors for the modules of the modular surrogate.

Again, the mse for the test set of the T5 module surrogate was cut at a value of 0.1 to make the other data more easily readable, but its real value is 1.949. As we can see in Figure 8.3.6, the T4 module surrogate has a consistent and low mse in all sets, performing a bit worse than the holistic surrogate (<0.017). Conversely, in the T5 module surrogate, the mse of test set is almost 163 times higher than the mse of the training set and validation sets.

Now that the error for the modular surrogate has been localized on the T5 module surrogate, we explore the potential cause for it. This module seems to perform properly with the training and the validation data (mse < 0.013 in both cases), which suggests that the connection of the two modules might be the source of this error. To investigate this, the same module (T5) will be tested with the original data. Specifically, the three input variables of this module that are obtained from the output of the module T4 (see Figure 8.2.3), will be replaced by real data from the simulation. The results for this test are depicted in Figure 8.3.7, which shows the mse values obtained with the T5 module when the aforementioned three inputs are replaced by the original simulated data instead of using the output from the T4 module.

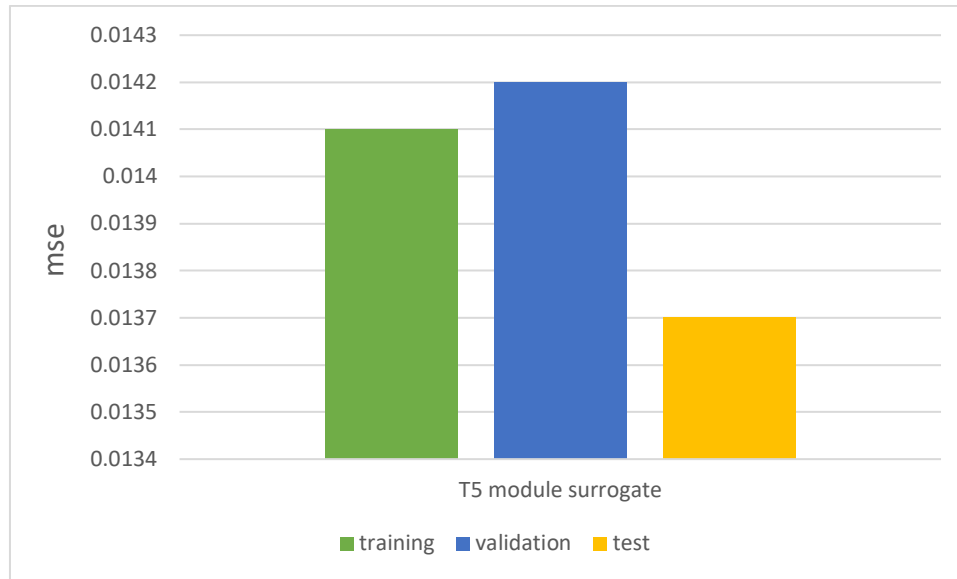


Figure 8.3.7 Mean of the squared errors for T5 module surrogate using the original data as test.

Although the mse test error is the lower across all sets, the maximum difference is lower than 0.0005, the lowest calculated across all surrogates, which indicates that this surrogate has an acceptable accuracy. If the mse value of the test set was significantly lower than the mse value on the training set, it would mean that the surrogate has bias.

These results imply that there is some error in the connection of both modular surrogates, i.e., when we use some outputs from the T4 module as inputs for the T5 module. A further study of the cause of the error is given in appendix A1.

After analyzing both alternatives (holistic vs modular surrogates), the global surrogate has a better performance (less error) for this particular configuration. Hence, the study of hidden input units will be conducted only for the holistic surrogate.

Note that the number of hidden units can be set independently for each hidden layer. Since the holistic surrogate model proved to work well, i.e., it showed a low global mse value with 8 hidden units, adding more hidden units was not considered. Thus, a list of other arbitrary values (≤ 8) was chosen, being these values 3, 6 and 8. To explore all possible combinations of these values between the two hidden layers, a list with all permutations between the 3 values was created. Table 7.1 shows the combinations used in each model.

Table 8.4. Number of hidden units used in every layer per every model.

Model	1	2	3	4	5	6	7	8	9
1 st layer	3	3	3	6	6	6	8	8	8
2 nd layer	3	6	8	3	6	8	3	6	8

In addition, we found out that the mse achieved by the holistic surrogate in every set are very similar and low (Figure 8.3.5), which indicates that the model has been trained correctly and can make acceptable predictions as a whole. However, for a more comprehensive evaluation, each output should be evaluated independently rather than grouped, i.e., the mse should be evaluated

for each individual output on a one-by-one basis using the test set. Hence, the results for the number of hidden units were evaluated comparing the prediction of each output variable given by the surrogate with the respective real value (from the simulation, i.e., y_{test}) for every variable. The results are collected in Figure 8.3.8.

	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Model 9
QC_T4	0,39856801	0,18896657	0,3116325	0,32120149	0,05154005	0,06922582	0,26174337	0,03410069	0,05151099
QR_T4	0,35945166	0,17192256	0,29241081	0,28553598	0,04761586	0,04239282	0,24277471	0,02269086	0,03009485
T_T4	0,7860951	0,09500001	0,03141073	0,07766819	0,02352022	0,0204886	0,05389914	0,01801788	0,02040922
M_CYCLOPENTANE	0,51514778	0,1270945	0,09083994	0,18524191	0,04124794	0,03934246	0,36769973	0,02068889	0,028185
BUTENE	0,19466317	0,06599465	0,08236031	0,17599095	0,0411007	0,04306037	0,09207248	0,02864219	0,05496049
T_T5	0,1465487	0,09472638	0,03406208	0,14377206	0,02237519	0,01258655	0,04268994	0,02125704	0,01854975
QR_T5	0,7214704	0,510993	0,08232226	0,36067848	0,02935966	0,04884307	0,99745144	0,02621293	0,06851489
QC_T5	0,69104482	0,59612471	0,08769777	0,17345651	0,02935707	0,05691372	0,98886994	0,03016101	0,06717101
M_BTTMS_T4	0,13965823	0,10194384	0,03403563	0,12379242	0,02306407	0,01399591	0,04751966	0,04113928	0,01793917
X_BUTENE_BTTMS_T4	0,28769744	0,08095844	0,02606335	0,1254202	0,02554223	0,02232263	0,04390723	0,01307534	0,02289012
X_CYCLOPENTANE_BTTMS_T4	0,43455449	0,09105245	0,03641624	0,0849995	0,02996882	0,02640634	0,04446492	0,0112298	0,01791246
PROPENE	0,98515621	0,85243335	0,81429883	0,78402031	0,82208562	0,61329862	0,81503025	0,76987775	0,73269736
CYCLOPENTANE	0,47997238	0,32562996	0,10934979	0,1114988	0,13868234	0,05727405	0,32792357	0,0881512	0,09326861
Average	0,47231	0,25406	0,15638	0,22718	0,10196	0,08201	0,33277	0,08656	0,09416

Figure 8.3.8 Heatmap of the test mse values for each model and every output variable of the holistic surrogate.

The color scale on the heatmap represents the range of the values (the largest value of the matrix is the upper boundary, and the lowest value of the matrix is the lower boundary, disregarding the column and row the other values will be colored from these values), low mse values are represented in green, large mse values in red, and in the middle mse values are represented in yellow for the “good half” and in orange for the “bad half”.

Figure 8.3.8 shows that propene (distillate) and cyclopentane (bottoms) molar fractions of distillation columns T4 and T5, respectively, are not well represented by any of the different surrogate models, especially the propene molar fraction, for which the lowest mse value achieved is 0.613 (model 6), significantly above the average error of this surrogate (0.082). This was already expected since, as shown in Figure 8.2.2, this variable had many outliers. For a better understanding of the behavior of model 6 predictions, the propene molar fraction values predicted by this model were plotted against the real propene molar fraction values from the test set. The results are shown in Figure 8.3.9.

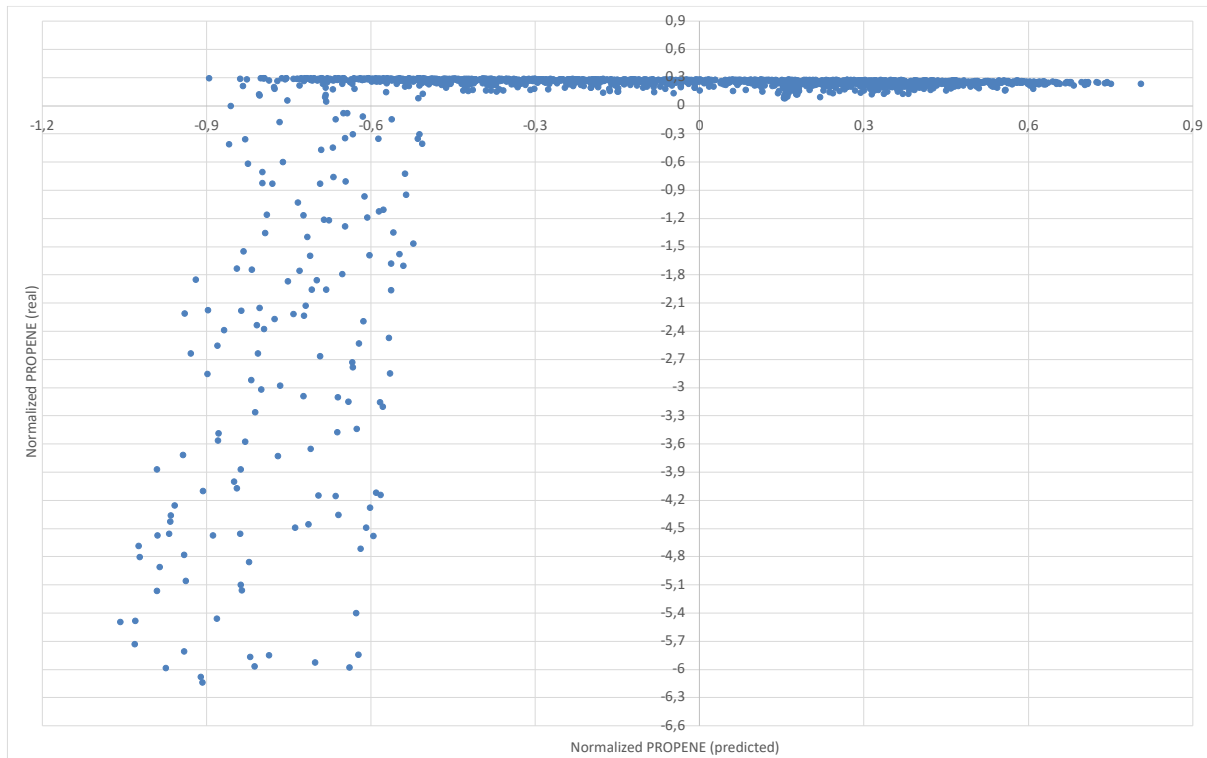


Figure 8.3.9 Real vs predicted values by model 6 of the molar fraction of propene at the distillate of T4

This plot concurs with the data showed in the boxplot (Figure 8.2.2), since both show that most normalized points are close to 0.3. A possible explanation of the high mse for the propene molar fraction is that, since most of the training data (+50%) were found around 0.3, the surrogate might predict 0.3 more than it should (i.e., the surrogate overfits the data for this output variable). There are different ways to treat overfitting, some are listed in chapter 3, but to improve the performance of these surrogates is left for future work.

In any case, results suggest that the remaining variables can be predicted at an acceptable accuracy.

The surrogate models that perform better are those for which (i) the mse average is low and (ii) consistently achieve a low mse value for every output variable. Models with a low average mse but a particularly high mse value in some output variables might not be acceptable for some applications where these latter variables are highly relevant. Bearing this in mind, the models that perform better are models 5, 6, 8 and 9. From these models, 6 and 8 combined perform better than models 5 and 9 in all the variables, hence a more detailed analysis is shown in Figure 8.3.10.

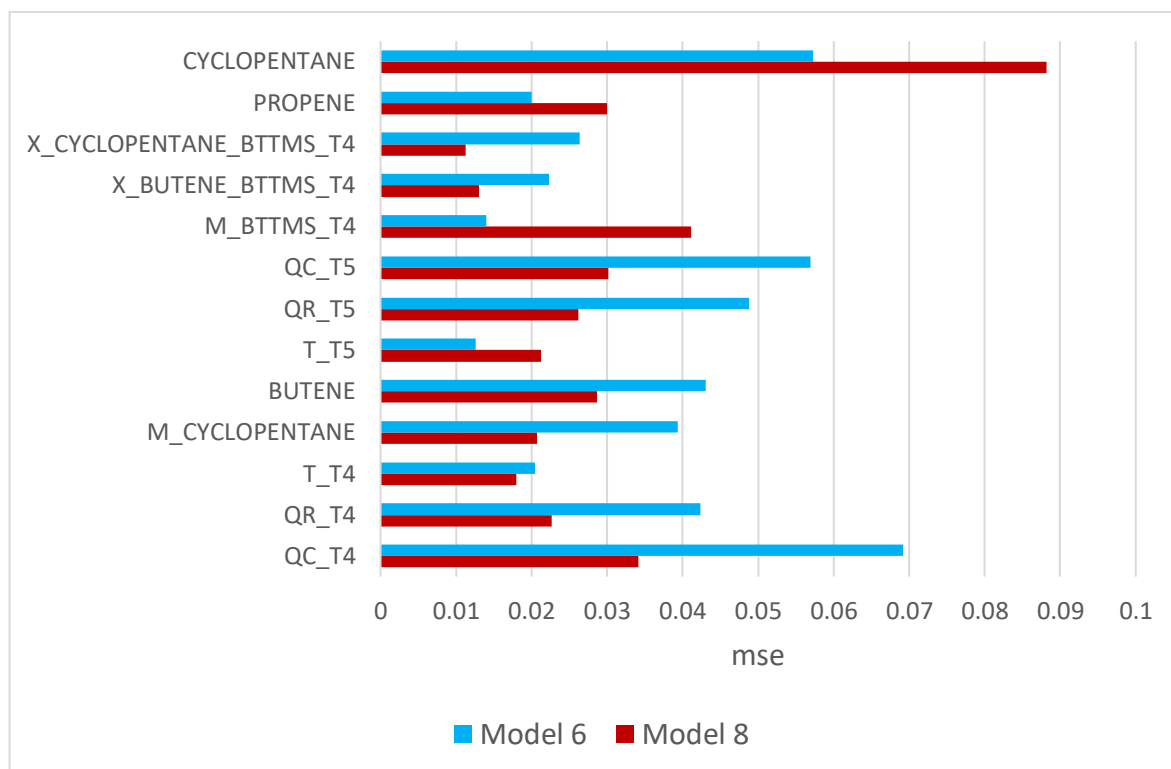


Figure 8.3.10 Mse values comparison for each output variables between models 6 and 8 of the holistic surrogate.

Note that the mse values for cyclopentane were reduced in order to make the data more easily readable, but they should concur with the values specified in Figure 8.3.8.

Model 6 performs better, i.e., predicts more accurately, for the molar fraction of cyclopentane and propene at the bottoms and distillate of column T5, respectively, for the mass flow at the bottoms of column T4 and for the input temperature of T4, whereas model 8 performs better in every other variable (4 vs 9 variables). Although model 8 performs better on most variables, the difference between the errors is also important to put into perspective, since as shown in Figure 8.3.8, model 6 performs better overall (0.08201 vs 0.08656).

The biggest differences are on the heat flows of distillation column T4 and T5 (both for the condenser and reboiler), where model 6 performs significantly worse (almost 2 times higher than model 8). The other big differences between these 2 models are found at the molar fractions of propene and cyclopentane at the distillate and bottoms, respectively, of distillation column T5. In this case, model 6 performs better for both variables (0.613 vs 0.769 for propene and 0.129 vs 0.159 for cyclopentane). There is a tradeoff between model 6 and model 8, so depending on which output variables are more important for the specific problem, one would choose a model over the other.

Lastly, it is worth noting that the output variables predicted by the chosen surrogate (along with other parameters and variables that are not studied on this project) could be used to calculate the economic cost and environmental impact of, in this case, distillation columns T4 and T5.

9. Environmental impact

As stated in chapter 1.2, nowadays one of the most important challenges to tackle is global warming. This has empowered the circular economy movement, which intends to expand the life cycle of a product or material. The case study of this project is a great example, where waste polyethylene is pyrolyzed and separated to obtain various monomers and give it a new life.

The project *per se* has only been fully executed on a digital environment. Therefore, only the energy consumed by the laptop to develop this project will be evaluated in terms of the equivalent CO₂ emissions. The laptop used to develop this process is a MacBook Air Monterey with the apple chip M1 (8 core GPU and 8 core CPU). To compute the energy consumed the power draw by the GPU and the CPU, were found, standing at values of 10W (NotebookCheck 2020) and 13.5W (Ars Technica 2020), respectively. Then, their combined power consumption was multiplied by the number of GPU hours, which was considered the same number of hours dedicated to the project, i.e., 584h (see Economic evaluation) and by a Power Usage Effectiveness of 1.59 (Fu, Hosseini, and Plataniotis 2021), as shown in Eq. 9.1.

$$Pt (Wh) = 1.59 \cdot GPU \text{ hours} \cdot (p_g + p_c) \quad \text{Eq. 9.1}$$

This adds up to a total of 21821.16 Wh.

To convert this number of Watt-hours into CO₂ equivalent emissions, the national average of CO₂ emissions across Spain, published by the CNMC the 20th of April in 2022, is used. This factor stands at 259 g CO_{2eq}/kWh (Gencat 2022), which is equivalent to $2.59 \cdot 10^{-4}$ kg CO_{2eq}/Wh. By multiplying this value by the energy consumed, we obtain a total of 5.65 kg CO_{2eq} emitted during the development of this project.

Conclusions

After developing a systematic methodology to build surrogate models for chemical processes and analyzing two alternative approaches for this (holistic vs modular), the following conclusions have been drawn with respect to the project objectives (see chapter 2.1).

- A methodology to build surrogate models for chemical processes was successfully built, enabling the development of holistic or modular surrogate models.
- The computational time required to generate the data (i.e., computing the simulations) was 48.08% lower when it was collected by modules (4h 56min 29 sec vs 7h 19min 2 sec). However, in the test performed, the feed to distillation column T5 was steady, which should not be allowed if these data were really used to build the corresponding surrogate.
- The accuracy of the modular surrogate was not as good as that of the holistic surrogate (test mse = 0.9412 for the modular surrogate vs 0.0561 for the holistic model). Apparently, this was caused by the lack of accuracy in the prediction of variables connecting the two modules (i.e., module T4 was not predicting these outputs accurately enough). However, when each module was used independently, they both had an acceptable accuracy (test mse of 0.0773 for T4 and 0.0137 for T5, respectively).
- With an architecture of 2 hidden layers, the optimal number of hidden units to use on the holistic surrogate model are 6 and 8 hidden units for the first and second hidden layer, respectively.
- A detailed systematic methodology was successfully described, providing examples for each step (using the case study). In addition, the files and code used as a case study are provided in a GitHub repository to enable the continuation of the proposed work.

On the basis of the conclusions and the results of this project, some future work is proposed below.

- To extend the proposed methodology to build a surrogate model for the whole process simulation.
- To implement and test the proposed methodology for other chemical processes and study its replicability (i.e., potential patterns for the optimal architecture of the ANN).
- To generate the necessary script and modify the functions to enable changes in the compositions of a stream in the HYSYS simulation, so that the feed of a module can be better represented.
- To repeat the study about the computational time needed to get the data for one module (i.e., process unit) when the feed is not steady.

- To improve the accuracy of the modular surrogate by treating the propagation of error or identifying the root of the cause that makes the connection to not work properly.
- To explore different input variables and study the accuracy of the surrogate models with the new input variables.
- To explore the use of increased weights, for variables connecting modules, when computing the loss functions of the different modules in a modular surrogate.
- To study the number of input variables a surrogate need to predict with an acceptable accuracy a constant value of output variables.

Economic evaluation

The economic evaluation of this project encompasses the costs associated to the software and hardware used as well as the personnel costs.

For the hardware costs, the physical material to develop this project has been considered. To calculate the final cost, amortization was applied following Eq. 0.1.

$$\text{Amortization} \left(\frac{\text{€}}{\text{year}} \right) = \frac{\text{Purchase price (€)} - \text{Salvage value (€)}}{\text{Lifespan (year)}} \quad \text{Eq. 0.1}$$

Table 0.1 shows the material used and its correspondent information to compute the final price. The lifespan considered for every hardware component is 4 years.

Table 0.1. Total cost of hardware components.

Hardware	Purchase price [€]	Salvage value [€]	Amortization [€/year]
Computer	1299	600	174.75
Total cost [€]			174.75

The software costs include the commercial process simulator used, Aspen HYSYS, and Microsoft Office tools. As the programming language used was Python, which is an open-source programming language, this cost was not included. Table 0.2 shows the total software costs of the project.

Table 0.2. Total costs for software licenses.

Software	Cost [€]
Microsoft Office 2021	299.00
Total cost [€]	299.00

The Microsoft Office 2021 license price was found in the official site (Microsoft Office 2021), and the Aspen HYSYS license price was not taken into consideration, since the university license was used, which is independent from this project. To convert the price from dollars to euros, the original price was multiplied by 0.94€/\$, as found on a trading currency site (Investing.com 2022) on the 06/06/2022.

Lastly, the personnel costs are summarized on Table 0.3.

Table 0.3. Total costs for personnel.

Personnel	Salary per hour [€/h]	Total hours	Cost [€]
Process Engineer	12.94	584	7556.96
Social security (23.6%)	3.05	584	1783.44
Total cost [€]			9340.40

The costs of the personnel have been calculated assuming that this project was developed by a junior process engineer. The annual gross salary for this position in Barcelona is 26911 €/year (Glassdoor 2022) on 06/06/2022. Considering 40 hours per week and 52 weeks per year, the hourly salary is 12.94 €/h. The hours dedicated to this project have been reckoned considering an average of 4 h per day, starting the 21st of January and finishing the 15th of June. That adds up to 584 h. The cost shown Table 0.3 is the result of multiplying the total number of hours per the hourly salary. The social security in Spain, which is regulated by the government, has a percentage of 23.6% for common contingencies (Gobierno de España 2022).

Additionally, an overhead of 10% has been applied to the total cost to consider other costs such as electricity and internet connection. The total costs of the project are summarized in Table 0.4.

Table 0.4. Total costs of the project.

Associated cost	Total cost [€]
Hardware	174.75
Software	299.00
Personnel	9340.40
Total without overhead	9814.15
Overhead (10%)	981.41
Total with overhead	10795.57

Figure 0.1 represents the cost distribution of the project.

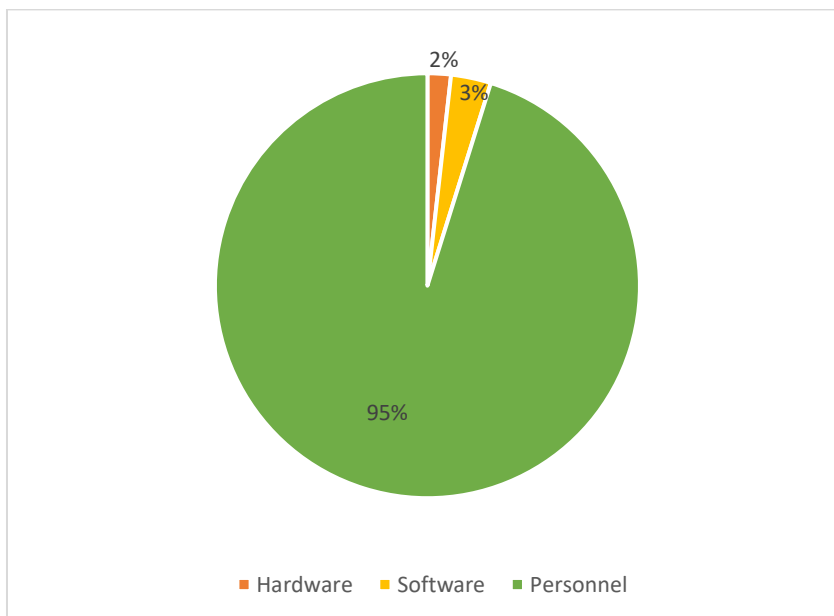


Figure 0.1. Distribution of the total cost of the project (%)

As can be seen in the pie chart, the personnel costs are the most significant item, representing 95% of the total costs of the project, followed by the software costs (3%) and, finally, the hardware costs (2%).

Bibliography

- Amidi, Afshine, and Shervine Amidi. 2019. "CS 230 - Deep Learning Tips and Tricks Cheatsheet." 2019. <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks>.
- Ars Technica. 2020. "MacOS 12 Monterey: The Ars Technica Review." 2020. <https://arstechnica.com/gadgets/2021/10/macOS-12-monterey-the-ars-technica-review/11/>.
- Bhosekar, Atharv, and Marianthi Ierapetritou. 2018. "Advances in Surrogate Based Modeling, Feasibility Analysis, and Optimization: A Review." *Computers & Chemical Engineering* 108 (January): 250–67. <https://doi.org/10.1016/J.COMPCHEMENG.2017.09.017>.
- Brownlee, Jason. 2019. "How to Use Learning Curves to Diagnose Machine Learning Model Performance." 2019. <https://machinelearningmastery.com/learning-curves-for-diagnosing-machine-learning-model-performance/>.
- Calvo-Serrano, Raul, Miao Guo, Carlos Pozo, Ángel Galán-Martín, and Gonzalo Guillén-Gosálbez. 2019. "Biomass Conversion into Fuels, Chemicals, or Electricity? A Network-Based Life Cycle Optimization Approach Applied to the European Union." <https://doi.org/10.1021/acssuschemeng.9b01115>.
- Fivga, Antzela, and Ioanna Dimitriou. 2018. "Pyrolysis of Plastic Waste for Production of Heavy Fuel Substitute: A Techno-Economic Assessment." *Energy* 149 (April): 865–74. <https://doi.org/10.1016/J.ENERGY.2018.02.094>.
- Fu, Andre, Mahdi S Hosseini, and Konstantinos N Plataniotis. 2021. "Reconsidering CO2 Emissions from Computer Vision."
- Gencat. 2022. "Factor de Emisión de La Energía Eléctrica: El Mix Eléctrico. Cambio Climático." 2022. https://canviclimatic.gencat.cat/es/actua/factors_demissio_associats_a_lenergia/.
- Geyer, Roland, Jenna R. Jambeck, and Kara Lavender Law. 2017. "Production, Use, and Fate of All Plastics Ever Made." *Science Advances* 3 (7). <https://doi.org/10.1126/SCIADV.1700782>.
- Glassdoor. 2022. "Glassdoor." 2022. https://www.glassdoor.co.uk/Salaries/barcelona-process-engineer-junior-salary-SRCH_IL.0,9_IM1015_KO10,33.htm?clickSource=searchBtn.
- Gobierno de España. 2022. "Seguridad Social." 2022. <https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537>.
- Honus, Stanislav, Shogo Kumagai, Ondřej Němček, and Toshiaki Yoshioka. 2016. "Replacing Conventional Fuels in USA, Europe, and UK with Plastic Pyrolysis Gases – Part I: Experiments and Graphical Interchangeability Methods." *Energy Conversion and Management* 126 (October): 1118–27. <https://doi.org/10.1016/J.ENCONMAN.2016.08.055>.
- Hoof, Jeroen van, and Joaquin Vanschoren. 2021. "Hyperboost: Hyperparameter Optimization by Gradient Boosting Surrogate Models."
- Investing.com. 2022. "Investing.Com." 2022. <https://es.investing.com/currencies/usd-eur>.

- Jiang, Ping, Qi Zhou, and Xinyu Shao. 2020. "Classic Types of Surrogate Models." *Springer Tracts in Mechanical Engineering*, 7–34. https://doi.org/10.1007/978-981-15-0731-1_2/FIGURES/10.
- Khoo, Hsien H. 2019. "LCA of Plastic Waste Recovery into Recycled Materials, Energy and Fuels in Singapore." *Resources, Conservation and Recycling* 145 (June): 67–77. <https://doi.org/10.1016/J.RESCONREC.2019.02.010>.
- Li, Houqian, Horacio A Aguirre-Villegas, Robert D Allen, Xianglan Bai, Craig H Benson, Gregg T Beckham, Sabrina L Bradshaw, et al. 2021. "Expanding Plastics Recycling Technologies: Chemical Aspects, Technology Status and Challenges."
- Li, Weixuan, Guang Lin, and Bing Li. 2016. "Inverse Regression-Based Uncertainty Quantification Algorithms for High-Dimensional Models: Theory and Practice." <https://www.sciencedirect.com/science/article/pii/S0021999116301851>.
- Li, Y. F., S. H. Ng, M. Xie, and T. N. Goh. 2010. "A Systematic Comparison of Metamodeling Techniques for Simulation Optimization in Decision Support Systems." *Applied Soft Computing* 10 (4): 1257–73. <https://doi.org/10.1016/J.ASOC.2009.11.034>.
- Liu, Han, and Mihaela Cocea. 2017. "Semi-Random Partitioning of Data into Training and Test Sets in Granular Computing Context." *Granular Computing* 2 (4): 357–86. <https://doi.org/10.1007/S41066-017-0049-2/FIGURES/11>.
- Loh, Wei Liem. 1996. "On Latin Hypercube Sampling." *Annals of Statistics* 24 (5): 2058–80. <https://doi.org/10.1214/AOS/1069362310>.
- Ma, Yannan, Jinzu Weng, Zhijiang Shao, Xi Chen, Lingyu Zhu, and Yuhong Zhao. 2015. "Parallel Computation Method for Solving Large Scale Equation-Oriented Models." *Computer Aided Chemical Engineering* 37 (January): 239–44. <https://doi.org/10.1016/B978-0-444-63578-5.50035-9>.
- Martínez, Arnau. 2021. "END-OF-DEGREE PROJECT Degree in Chemical Engineering DEVELOPMENT OF SURROGATE MODELS FOR DISTILLATION TRAINS Report and Appendices."
- Meert, Kürt, and Marcel Rijckaert. 1998. "Intelligent Modelling in the Chemical Process Industry with Neural Networks: A Case Study." *Computers & Chemical Engineering* 22 (SUPPL.1): S587–93. [https://doi.org/10.1016/S0098-1354\(98\)00104-5](https://doi.org/10.1016/S0098-1354(98)00104-5).
- Microsoft Office. 2021. "Microsoft Office 2021." Microsoft Office. 2021. <https://www.microsoft.com/es-es/microsoft-365/get-started-with-office-2021>.
- Mitchell, Tom M. 2006. "The Discipline of Machine Learning."
- Miyagawa, Eiichi, Katsuhisa Tokumitsu, Akira Tanaka, and Koh hei Nitta. 2007. "Mechanical Property and Molecular Weight Distribution Changes with Photo- and Chemical-Degradation on LDPE Films." *Polymer Degradation and Stability* 92 (10): 1948–56. <https://doi.org/10.1016/J.POLYMDEGRADSTAB.2007.05.019>.
- Mujtaba, I. M., N. Aziz, and M. A. Hussain. 2006. "Neural Network Based Modelling and Control in Batch Reactor." *Chemical Engineering Research and Design* 84 (8): 635–44. <https://doi.org/10.1205/CHERD.05096>.
- Naqa, Issam el, and Martin J. Murphy. 2015. "What Is Machine Learning?" *Machine Learning in Radiation Oncology*, 3–11. https://doi.org/10.1007/978-3-319-18305-3_1.
- Ng, Andrew. 2011. "Regularization - The Problem of Overfitting." Machine Learning - Stanford University | Coursera - Week3. 2011. <https://www.coursera.org/learn/machine-learning/home/welcome>.

- . 2017. “Why Is Deep Learning Taking off? | Coursera.” 2017. <https://www.coursera.org/learn/neural-networks-deep-learning/lecture/paGm/why-is-deep-learning-taking-off>.
- . 2020. “Forward Propagation in a Deep Network | Coursera.” 2020. <https://www.coursera.org/learn/neural-networks-deep-learning/lecture/MjzH/forward-propagation-in-a-deep-network>.
- Nguyen, Trung H., Duy Nong, and Keith Paustian. 2019. “Surrogate-Based Multi-Objective Optimization of Management Options for Agricultural Landscapes Using Artificial Neural Networks.” *Ecological Modelling* 400 (May): 1–13. <https://doi.org/10.1016/J.ECOLMODEL.2019.02.018>.
- NotebookCheck. 2020. “Apple M1 8-Core GPU vs NVIDIA GeForce GTX 1080 Ti (Desktop) vs Apple M1 7-Core GPU.” 2020. https://www.notebookcheck.net/M1-8-Core-GPU-vs-GeForce-GTX-1080-Ti-vs-M1-7-Core-GPU_10552_7700_10560.247598.0.html.
- Pan, Indranil, Masoud Babaei, Anna Korre, and Sevkett Durucan. 2014. “Artificial Neural Network Based Surrogate Modelling for Multiobjective Optimisation of Geological CO₂ Storage Operations.” *Energy Procedia* 63: 3483–91. <https://doi.org/10.1016/J.EGYPRO.2014.11.377>.
- Safari, Ayoub. 2022. “Automation of Control Degrees of Freedom in Aspen Hysys.” *IFAC Journal of Systems and Control* 19 (March): 100187. <https://doi.org/10.1016/J.IFACSC.2022.100187>.
- Serranti, Silvia, and Giuseppe Bonifazi. 2019. “Techniques for Separation of Plastic Wastes.” *Use of Recycled Plastics in Eco-Efficient Concrete*, 9–37. <https://doi.org/10.1016/B978-0-08-102676-2.00002-5>.
- Sharma, Siddharth, Simone Sharma, and Anidhya Athaiya. 2020. “ACTIVATION FUNCTIONS IN NEURAL NETWORKS.” *International Journal of Engineering Applied Sciences and Technology* 4: 310–16. <http://www.ijeast.com>.
- Sola, J., and J. Sevilla. 1997. “Importance of Input Data Normalization for the Application of Neural Networks to Complex Industrial Problems.” *IEEE Transactions on Nuclear Science* 44 (3 PART 3): 1464–68. <https://doi.org/10.1109/23.589532>.
- Somoza, Ana, Andres Gonzalez, Carlos Pozo, Moisés Graells, Antonio Espuña, and Gonzalo Guillén. 2020. “Realizing the Potential High Benefits of Circular Economy in the Chemical Industry: Ethylene Monomer Recovery via Polyethylene Pyrolysis.” <https://doi.org/10.1021/acssuschemeng.9b04835>.
- Speight, James G. 2020. “Monomers, Polymers, and Plastics.” *Handbook of Industrial Hydrocarbon Processes*, January, 597–649. <https://doi.org/10.1016/B978-0-12-809923-0.00014-X>.
- Stanford University. 2020. “CS231n Convolutional Neural Networks for Visual Recognition.” 2020. <https://cs231n.github.io/neural-networks-1/#add>.
- Statista. 2022. “Global Plastic Production 1950-2020.” January 12, 2022. <https://www.statista.com/statistics/282732/global-production-of-plastics-since-1950/>.
- Sun, Gang, and Shuyue Wang. 2019. “A Review of the Artificial Neural Network Surrogate Modeling in Aerodynamic Design.” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering* 233 (16): 5863–72. <https://doi.org/10.1177/0954410019864485>.
- Tripathy, Rohit K., and Ilias Bilionis. 2018. “Deep UQ: Learning Deep Neural Network Surrogate Models for High Dimensional Uncertainty Quantification.” *Journal of*

- Computational Physics* 375 (December): 565–88.
<https://doi.org/10.1016/J.JCP.2018.08.036>.
- Vabalas, Andrius, Emma Gowen, Ellen Poliakoff, and Alexander J. Casson. 2019. “Machine Learning Algorithm Validation with a Limited Sample Size.” *PLOS ONE* 14 (11): e0224365. <https://doi.org/10.1371/JOURNAL.PONE.0224365>.
- Wang, Zilong, and Marianthi Ierapetritou. 2017. “A Novel Feasibility Analysis Method for Black-Box Processes Using a Radial Basis Function Adaptive Sampling Approach.” *AIChE Journal* 63 (2): 532–50. <https://doi.org/10.1002/AIC.15362>.
- Williams, B. A., and S. Cremaschi. 2019. “Surrogate Model Selection for Design Space Approximation And Surrogatebased Optimization.” *Computer Aided Chemical Engineering* 47 (January): 353–58. <https://doi.org/10.1016/B978-0-12-818597-1.50056-4>.

Appendix A

A1. Discussion of the modular surrogate error

In the modular surrogate, some outputs from the module surrogate T4 are retrieved and used as inputs for the module surrogate T5, which caused the modular surrogate to have high mse values for the test set (i.e., it does not predict accurately for new unseen data). Hence, to explore the source of the error for the modular surrogate, the values of the mse for module surrogate T4 were evaluated for every output variable independently. The results are shown in Figure 0.2.

	mse
PROPENE	0,4533
QC_T4	0,0074
QR_T4	0,0075
T_T4	0,0119
M_BTTMS_T4	0,0066
X_BUTENE_BTTMS_T4	0,0138
X_CYCLOPENTANE_BTTMS_T4	0,0053

Figure 0.2. Specified properties for PE.

Specifically, the last three parameters are the ones that will be used as inputs for the module surrogate T5. Looking at the mse values, it seems that the module surrogate T4 predicts, at least, 2 times worse the molar fraction of butene, compared to the other two variables. It is possible that this error propagates to the module surrogate T5, which would explain its high mse value. The predicted value for the molar fraction of butene is compared to the corresponding real value in Figure 0.3.

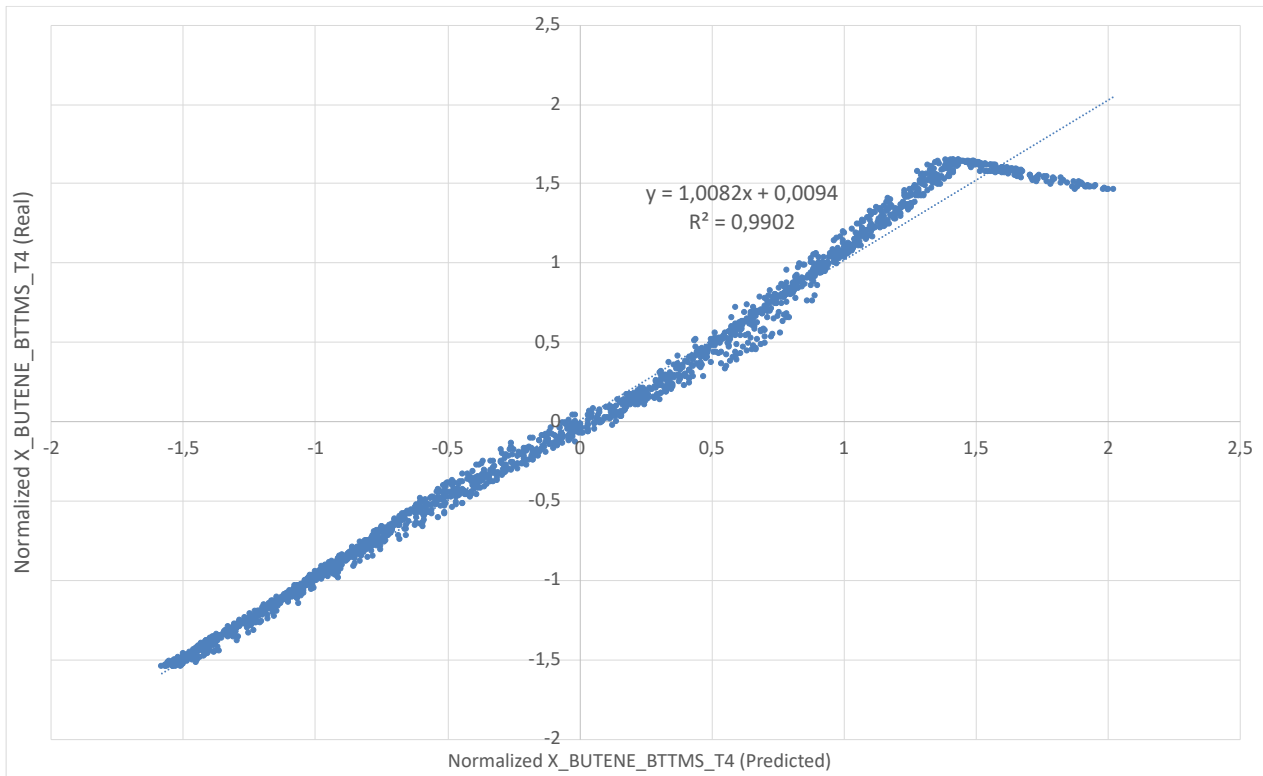


Figure 0.3. Real vs predicted values by module surrogate T4 of the molar fraction of butene at the bottoms of T4

It is now clear that the model predicts well most of the data (the slope is very close to 1, specifically 1.0082) until the normalized molar fraction (predicted) reaches a value of ~ 1.4 , where the straight line starts to flatten and decrease progressively.

Another way to visualize this consists of plotting the absolute error between the predicted values and the real values, as shown in Figure 0.4.

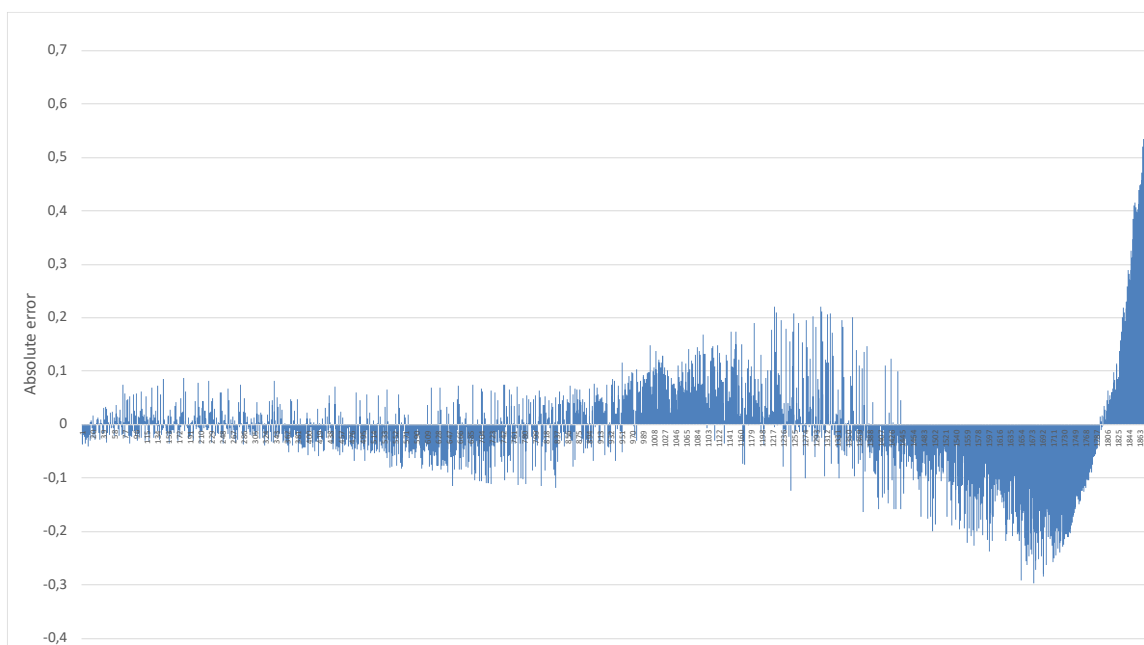


Figure 0.4. Absolute error between the real and the predicted values by module surrogate T4 of the molar fraction of butene at the bottoms of T4

For this plot, the data points were sorted in an ascending order of molar fraction. Hence the figure shows how the surrogate can predict the output with an acceptable accuracy at low values, yet it seems there is a wave-like error, with amplification, which results in larger error for higher molar fractions, as suggested earlier on this chapter. Further studies of how to avoid this error are left for future work.

A2. Files used to build the surrogate models

All the files used to design the simulation process, generate the data (and the data as well) and build the surrogate models are uploaded on the linked GitHub repository.

<https://github.com/guillemrh/TFG>