Aalto University
School of Chemical
Engineering

Aino Anttila

MACHINE LEARNING MODELS IN STEADY-STATE PROCESS SIMULATIONS

Master's Programme in Chemical, Biochemical and Materials Engineering

Major in Chemical and Process Engineering

Master's thesis for the degree of Master of Science in Technology

Submitted for inspection, Espoo, 29.5.2023.

Supervisor	Professor Ville Alopaeus
Instructors	D.Sc. (Tech.) Susanna Kuitunen
	M.Sc. (Tech.) Artturi Mäntynokka

Aalto University
School of Chemical Engineering

Aalto University, PL 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Aino Anttila	
Title of thesis Machine Learning Models in Steady-State Process Simulations	
Programme Master's Programme in Chemical, Biochemical and Materials Engineering	
Major Chemical and Process Engineering (CHEM3043)	
Thesis supervisor Prof. Ville Alopaeus	
Thesis advisors D.Sc. (Tech.) Susanna Kuitunen, M.Sc. (Tech.) Artturi Mäntynokka	
Collaborative partner Neste Oyj, Engineering Solutions	
Date 29.5.2023 **Number of pages** 103 + 33	**Language** English

Abstract

Process modelling has conventionally relied on first principles methods representing thermodynamic systems and unit operations with a mathematical model. Since the mathematical models are solved iteratively, large simulation flowsheets often have very long computational times, or even fail to converge if certain complex thermodynamic and mass transfer models are used. In recent years, machine learning (ML) methods have shown potential in shortening the computational times and decreasing the number of simulation runs needed. Therefore, the aim of this thesis was to study the opportunities, challenges, and other important factors to consider related to applying ML models in steady-state process simulations.

In the literature part of this thesis, a general overview on ML and the most important thermodynamic and physical constraints for ML models in process simulations were presented. Moreover, factors such as training data quality and quantity, computational and training times as well as opportunities in modelling systems that are difficult to model with first principles methods alone were recognized as important aspects and potential benefits and challenges in ML process modelling. Amine absorption is a reactive unit operation with rate-controlled reactions that conventionally requires a complex model, where convergence issues hinder its use as a part of larger simulation flowsheets. In the applied part of this thesis, ML tools provided by Aspen Plus were used to create neural network and linear models of amine absorption with the goal of shortening computational times and increasing model robustness while maintaining accuracy. Additionally, the effect of different ML model properties as well as training data quantity and wideness were studied. Training data for the models was generated by simulating an amine absorber with a rate-based column model using electrolyte thermodynamics.

The first principles driven, artificial intelligence driven, and reduced order models were able to accurately and with increased model robustness model the absorption column in interpolated validation cases while the computational time per case was reduced to less than half of the base model's computational time. Extrapolating the models, however, is not recommended based on these results, and model training data should be wide enough to cover all necessary use cases. The most accurate models were obtained with training datasets of 25 000 or more datapoints, which gives an indication to the order of magnitude needed in this type of modelling.

Keywords machine learning, hybrid modelling, neural network, training data

Aalto University
School of Chemical Engineering

Aalto University, PL 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Tekijä Aino Anttila

Työn nimi Koneoppimismallit stationaaritilaisissa prosessisimulaatioissa

Koulutusohjelma Master's Programme in Chemical, Biochemical and Materials Engineering

Pääaine Chemical and Process Engineering (CHEM3043)

Vastuuopettaja/valvoja Prof. Ville Alopaeus

Työn ohjaajat TkT Susanna Kuitunen, DI Artturi Mäntynokka

Yhteistyötaho Neste Oyj, Engineering Solutions

Päivämäärä 29.5.2023 **Sivumäärä** 103 + 33 **Kieli** Englanti

Tiivistelmä

Kemiallisten prosessien mallinnuksessa on perinteisesti käytetty laskennallisia menetelmiä erilaisten yksikköoperaatio- ja termodynaamisten mallien ratkaisemiseen. Koska laskennalliset mallit ratkaistaan iteratiivisesti, kookkaiden prosessisimulaatioiden ratkaisuajat ovat usein pitkiä, eivätkä tiettyjä monimutkaisia termodynaamisia- ja aineensiirtomalleja sisältävät simulaatiot välttämättä konvergoi lainkaan. Lähivuosina koneoppimismetodeilla on osoitettu olevan valmiuksia lyhentää simulaatioiden ratkaisuaikoja ja vähentää tarvittujen ajojen määrää. Tämän työn tarkoituksena oli tutkia koneoppimismallien käytön tuomia mahdollisuuksia, haasteita ja muita huomioonotettavia aspekteja stationaaritilaisissa prosessisimulaatioissa.

Työn kirjallisuusosassa tehtiin yleiskatsaus koneoppimismalleihin sekä tärkeisiin prosessisimulaatiosovelluksissa huomioon otettaviin termodynaamisiin ja fyysisiin rajoitteisiin. Lisäksi kirjallisuuskatsaus osoitti tärkeiksi tekijöiksi koneoppimismallien kehityksessä koulutusdatan laadun ja määrän sekä mallien koulutukseen ja valmiiden mallien ratkaisuun kuluvan ajan, sekä mahdollisuudet mallintaa systeemejä joiden mallinnus perinteisin menetelmin on haastavaa. Amiiniabsorptio on reaktiivinen yksikköoperaatio, jonka nopeusperustaista mallia ei kompleksisuutensa ja konvergointivaikeuksiensa takia voida juuri käyttää osana suurempia prosessisimulaatioita. Siispä tämän työn soveltavassa osassa luotiin neuroverkko- sekä lineaarisia koneoppimismalleja Aspen Plus -työkaluilla. Koneoppimismallien tavoitteena oli absorptiokolonnin tarkka mallinnus lyhyemmillä ratkaisuajoilla sekä konvergenssiongelmia välttäen. Myös koneoppimismallien ominaisuuksien, koulutusdatan määrän ja laajuuden vaikutuksia koneoppimismallien ennustuskykyyn tutkittiin. Koulutusdata luotiin simuloimalla nopeusperusteista amiiniabsorberikolonnia elektrolyyttisellä termodynamiikkamallilla.

Työssä luodut mallit pystyivät ennustamaan absorptiokolonnia tarkasti interpoloiduissa validointiajoissa ilman konvergenssiongelmia. Mallien ratkaisuaika oli alle puolet nopeusperustaisen kolonnimallin ratkaisuajasta. Mallien ekstrapolointia sen sijaan ei työn tulosten perusteella suositella, vaan kaikki käyttötapaukset on tarpeen sisällyttää koulutusdataan. Tarkimmin kolonnin toimintaa ennustavat mallit luotiin vähintään 25 000 datapistettä sisältävällä koulutusdatalla, mikä antaa osviittaa kyseisenkaltaisissa sovelluksissa tarvittavasta datan määrästä.

Avainsanat koneoppiminen, hybridimallinnus, neuroverkko, koulutusdata

Preface

This master's thesis was written to complete the degree of Master of Science (Tech.) in the Chemical and Process Engineering major of Master's Programme in Chemical, Biochemical and Materials Engineering. The thesis work was conducted at Neste Oyj, Engineering Solutions, between November 2022 and May 2023.

I would like to thank Leena Perttu-Roiha for this opportunity to dive into the world of machine learning and process simulations. The thesis process has been extremely interesting and rewarding, and I have learned a great deal of new valuable skills. My sincere thanks go also to my thesis advisors Susanna Kuitunen and Artturi Mäntynokka, who have been ready to provide their advice, feedback, and encouragement at all times during the thesis process. From Aalto University, I would like to thank Professor Ville Alopaeus for his valuable feedback and input during the process. I extend my thanks to Anders Riska for guiding me in the beginning of my journey at Neste during the past summers.

As this thesis concludes my years in Aalto University, I wish to also thank my friends for all the great memories during the studies and free time. I am beyond grateful for having had you all by my side through thick and thin, from pandemic to projects. Lastly, a sincere thank you to all my loved ones (you know who you are). Thank you for your endless support in all my endeavours.

In Espoo, 29th of May 2023

Aino Anttila

Table of Contents

List of appendices

Nomenclature

Abbreviations

ANN	artificial neural network
ASNN	algorithmically structured artificial neural network
ASW	Aspen Simulation Workbook
CH_4	methane
CO_2	carbon dioxide
DDB	Dortmund Data Bank
ELEC-NRTL	Electrolytic Nonrandom Two-Liquid theory
FCC	fluid catalytic cracking
FP	first principles
GPR	Gaussian process regression
H_2O	water
H_2S	hydrogen sulfide
LHV	lower heating value
LLE	liquid-liquid equilibrium
LR	learning rate
MAE	mean average error
MAPE	mean average percentage error
MCM	matrix completion method
MDEA	methyl-diethanolamine
ML	machine learning
MSE	mean squared error
NN	neural network
NRTL	Nonrandom Two-Liquid Theory
ODE	ordinary differential equation
PDE	partial differential equation
RK	Redlich-Kwong
ROM	reduced order model
SLE	solid-liquid equilibrium
SS	steady state
SSE	sum of squared error
SVM	support vector machine
SVR	support vector regression
UNI-QUAC	Universal Quasi-chemical Theory
VLE	vapor-liquid equilibrium

Notation

a_i	stoichiometric coefficient of component *i* in a reaction
C	number of components
C_i	concentration of component *i*
Γ	ion charge in relation to hydrogen atom
E_a	activation energy
E_C	cumulative error
E_I	internal energy (J)
E_K	kinetic energy (J)
F	number of degrees of freedom
H	enthalpy $\left(\frac{\mathrm{kJ}}{\mathrm{mol}}\right)$
k	number of training examples
k_r	pre-exponential factor
k_t	number of test cases
N	number of components in a reaction
n	amount of component
n_T	temperature exponent
P	number of phases
p	pressure (kPa)
Q_r	heat input rate $\left(\frac{\mathrm{J}}{\mathrm{s}}\right)$
R	gas constant $\left(\frac{\mathrm{J}}{\mathrm{mol\ K}}\right)$
r	number of independent chemical reactions
T	temperature (K)
t	time (s)
W_r	rate of work done into a system $\left(\frac{J}{s}\right)$
y	gas phase mole fraction
Y_{actual}	value of training example
Y_{mean}	mean value of training examples
$Y_{predicted}$	model output value
x	liquid phase mole fraction
$\underline{\omega}$	weight parameter

Subscripts

i	component number

1 Introduction

Process modelling has largely relied on first principles methods developed with applied mathematical methods such as linear algebra, ordinary differential equations (ODEs), and partial differential equations (PDEs), since the 1950s, when the methods were first introduced (Venkatasubramanian, 2019). First principles models represent thermodynamic systems and unit operations with a mathematical model, which is solved to study the processes (Foo et al., 2017). Machine learning, on the other hand, is a data-driven modelling method that extracts relationships from vast amounts of data (Nasiriany et al., 2019). While the first machine learning methods were introduced to chemical engineering applications already in the 1980s, only the most recent wave of excitement towards machine learning has shown potential for significant improvements in process modelling (Lee et al., 2018). Commercial simulation software providers have also reacted to the interest towards machine learning by offering machine learning tools within the simulation software. For example, Aspen Plus offers different Hybrid Modelling tools allowing the implementation of neural networks and linear machine learning models in process simulation flowsheets (AspenTech, 2020a).

As studies on implementing machine learning in process modelling applications have been published frequently in the recent years, machine learning has shown opportunities in decreasing computational time or the number of simulation runs needed in complex process simulations (Heese et al., 2020). Since large simulations often take a significant amount of time to run, decreasing the computational time is of particular interest in modelling. Moreover, machine learning models may help improve the modelling of such component properties and unit operations that are difficult to model with first principles methods alone (AspenTech, 2020a). Particularly accurate and robust machine learning models have been developed when the models are constrained to obey thermodynamic and physical laws, or first principles methods have been combined with machine learning methods (Carranza-Abaid and Jakobsen, 2022).

Although there are vast amounts of plant, experimental, and thermodynamic process data available, challenges in machine learning model training may arise with data quality and scope, which both affect the performance of a machine learning model (Ding et al., 2021). However, simulated data has also been shown useful in training machine learning models, which may help model developers to obtain better quality and larger scope data (Heese et al., 2020). Challenges in applying machine learning methods to process modelling include the difficulty of human interpretation of the models, which may negatively affect the credibility of the models (Dobbelaere et al., 2021).

If the potential benefits of machine learning can be exploited in process modelling, advantages in more efficient modelling could be achieved. Therefore, the aim of this thesis is to determine, whether currently available machine learning models and methods can be useful in steady-state process simulation applications, and if so, the most suitable ways to utilize them. More precisely, the potential of utilizing neural network models and linear regression type machine learning models as a part of Aspen Plus simulations shall be studied in practice.

The work in this thesis is divided into a literature and an applied part. The literature part covers the basic aspects of machine learning and some of the most relevant machine learning methods regarding process modelling in Section 2. Secondly, important thermodynamic and physical constraints, which should be considered when machine learning models are used in process modelling, are discussed in Section 3. Finally, Section 4 covers the basic working principles, suitable applications, as well as challenges and opportunities regarding the utilization of machine learning models in process simulations. In the applied part, training data developing machine learning models are first generated with Aspen Plus. Data is generated using a rate-based column modelling an amine absorber. Machine learning models are trained and evaluated with the simulated data using the Hybrid Modelling features in Aspen Plus. The model performance and suitability for modelling applications is evaluated by, for example, validating against the first principles model and assessing the model's extrapolation abilities. Recommendations for future work are presented.

LITERATURE PART

2 Machine learning

The term machine learning (abbreviated as ML for the rest of this thesis) refers to algorithms used to extract relationships from data (Nasiriany et al., 2019). These relationships are obtained by training ML models with different types of data, such as images, tabulated data, sensor, or experimental data. The data may be either labeled with the desired output or unlabeled, when the model looks for similarities or anomalies within the data (Mowbray et al., 2022). While the enthusiasm around ML may suggest that ML could achieve extraordinary things and replace human knowledge in some applications by utilizing advanced algorithms, ML models are often built using rather well-known algorithms, such as linear or logistic regression, and some methods have been used already for a long time (Venkatasubramanian, 2019).

A general ML development loop starts with the choice of model architecture followed by model training with training data, validation to assess the model performance, and finally the model is adjusted to enhance its performance (Carranza-Abaid et al., 2020). Model adjustments may include changing the model size, modifying input features, and modifying the model architecture. In addition, training data may be added or augmented by creating more variance to the training data by reducing training data regularization (Ng, 2018).

The main ML categories relevant in the field of process engineering are presented in Section 2.1, based on the type of data used in model training. Simple examples for utilization of different models are included, and more detailed examples of process simulation applications are presented later in Section 4. Additionally, neural networks, support vector machines, and decision trees are introduced in Sections 2.2 through 2.4 as the most relevant ML model types in regards of the process industry. General methods to assess and compare ML model performance are discussed in Section 2.5.

2.1 Main categories of machine learning

Machine learning methods can be divided into categories based on the data that is used in model training. In process engineering applications, the most used ML categories are supervised learning, unsupervised learning, and reinforcement learning. In supervised learning models, the training data is handled, and the model training and testing data are set up by the developer of the model (Wuest et al., 2016). Supervised models are trained with data where the desired output is given to the model by labeling the data (Mowbray et al., 2022). Unsupervised learning models are trained with unlabeled data, with the intention of finding similarities and thus clusters or anomalies within the data (Nasiriany et al., 2019). In cases where ML is applied in sequential decision-making under uncertainty, reinforcement learning models are used (Mowbray et al., 2022).

Supervised learning models are built by selecting a suitable learning algorithm, such as linear regression, artificial neural network (ANN), or decision tree, for the learning task. The model is trained using a labeled training dataset with known outputs, and finally the model is evaluated using validation and/or testing datasets (Ng, 2018). According to the evaluation of the model performance, the model's parameters may be adjusted to enhance the model performance, or if necessary, the whole modelling process may require starting over and choosing a different algorithm in cases of very poor model performance (Wuest et al., 2016). Depending on the problem and the used learning algorithm, the model performance may also be enhanced by introducing more training data (Ng, 2018).

Some of the most widely known applications of supervised learning are email spam filtering and online advertising, where the user gives the model the desired outputs: whether emails from certain senders are considered spam or not (Mashaleh et al., 2022), or predicting the potential buyer behavior in online retail (Parihar and Yadav, 2022). Supervised learning is also perhaps the most commonly used ML type in manufacturing industries, since often there are vast amounts of labelled data available (Wuest et al., 2016). In process engineering, supervised learning can be used to predict parameters such as, mass-transfer coefficients in chemical reactor design, based on, for example, non-linear regression (Mowbray et al., 2022).

When a model is identifying patterns or groups within the input data without being given the desired output, the model is considered an unsupervised learner (Zhou, 2021). General applications where unsupervised learning are utilized are clustering, anomaly detection, and dimensionality reduction (Nasiriany et al., 2019). For example, algorithms suggesting similar articles for the reader in search engines utilize clustering by recognizing similarities in pieces of text as well as similarities in the users (Bouras and Tsogkas, 2014). Unsupervised learning has not been as widely used in the manufacturing industries since the available process data is often labeled, however unsupervised learning may benefit process industry in cases where expert data or feedback is not available, or the goal is to identify outliers in process data (Wuest et al., 2016). An example of an unsupervised model in the process industry is a control chart, where control measurements are categorized into either in-control or out-of-control. There, the categorization is often conducted by determining the distance of the measurement from the statistical model (Mowbray et al., 2022). Process data is usually high-dimensional, and unsupervised learning can be used to reduce the dimensionality to simplify the model while maintaining information (Heese et al., 2020).

The third main category of ML used in process engineering applications is reinforcement learning. The absence of labeled data separates reinforcement learning from supervised learning, while a reinforcement signal separates it from unsupervised learning (Wuest et al., 2016). Reinforcement learning requires a defined reward function to minimize the present error in the output as well as the past and immediate future error (Mowbray et al., 2022). A reinforcement learning model learns which actions generate the best results, i.e., which action results in the best numerical reinforcement signal, by trying different actions and adjusting according to the signal, instead of being given the best results in the training data (Zhou, 2021). Reinforcement learning can be particularly useful in the field of process control, for example, in PID controllers where the parameters can be tuned by the controller or other tuning methods and heuristics (Mowbray et al., 2022).

In the next subsections, some of the most used ML models are introduced. As already discussed, the models may be implemented in either supervised, unsupervised, or reinforcement learning, depending on the type of data used in model training. While neural networks, for example, are often implemented as supervised learning, applications for unsupervised learning neural networks do exist (Wuest et al., 2016). The same applies for other ML algorithms, and usually models can be applied in both supervised and unsupervised learning. Figure 1 illustrates how different ML models may be applied as supervised, unsupervised and reinforcement learning. The figure introduces a number of different machine learning methods, however only neural networks, decision trees and support vector machines are described in the following subsections due to their relevancy in process engineering applications.

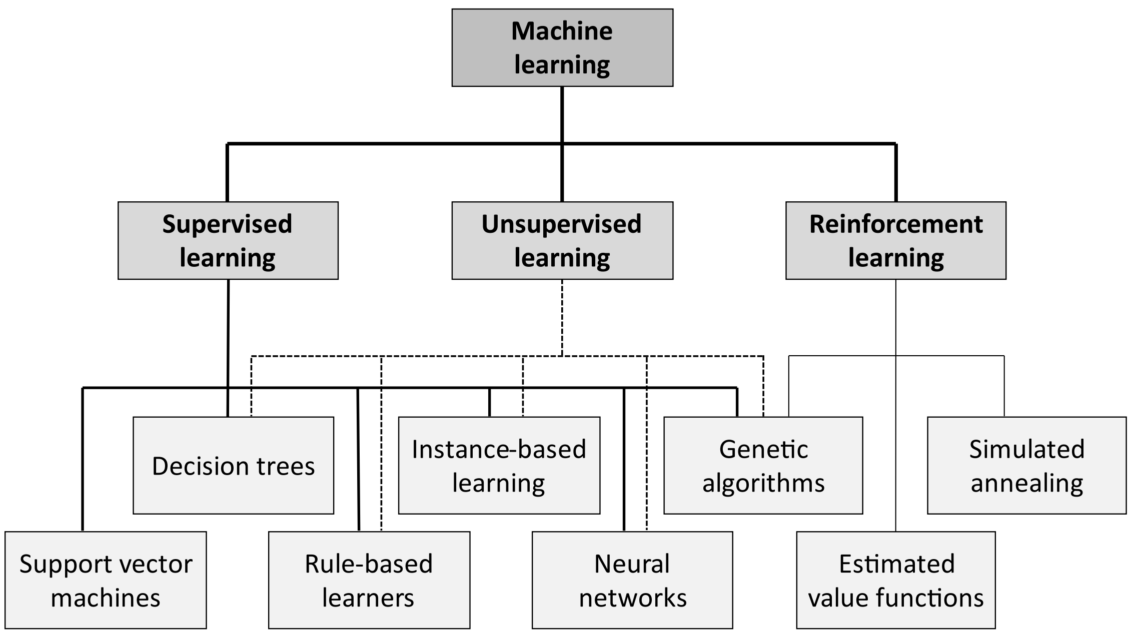


Figure 1. Different machine learning models for supervised, unsupervised and reinforcement learning. Adapted from Wuest et al. (2016).

2.2 Neural networks

Originally, neural networks were developed with the aim at mimicking the human neurons (Macukow, 2016). Neural networks consist of nodes (also called neurons or units), which are organized into layers with a defined number of inputs and outputs, with the first layer being the input layer, last layer the output layer and all other layers in between being hidden layers. Deep learning neural networks consist of more than one hidden layer (Nasiriany et al., 2019). The data input is transferred to the neural network from the input layer, and the optional hidden layers provide a prediction of the outputs as a function of the inputs (Carranza-Abaid and Jakobsen, 2022). The nodes compute weighed sums of their inputs (weighed by weight parameters $\underline{\omega}$) and apply an activation function (a nonlinear or linear function) to compute the output of the layer, which is further passed to the next layer as input (Nasiriany et al., 2019). The mathematical form of a neural network is a set of matrices, where the inputs and weight parameters can be represented as column vectors, and the activation function is a function of the column vectors (Rebala et al., 2019). Similar matrix formulation is done for each node.

The performance of the neural network is monitored and enhanced by calculating and minimizing the loss function, which is defined depending on the problem and the selected learning algorithm. The weight parameters are adjusted according to the model performance and passed back to the neural network as inputs in back-propagation to minimize the loss function (Carranza-Abaid and Jakobsen, 2022). The general structure of neural networks is presented in Figure 2.

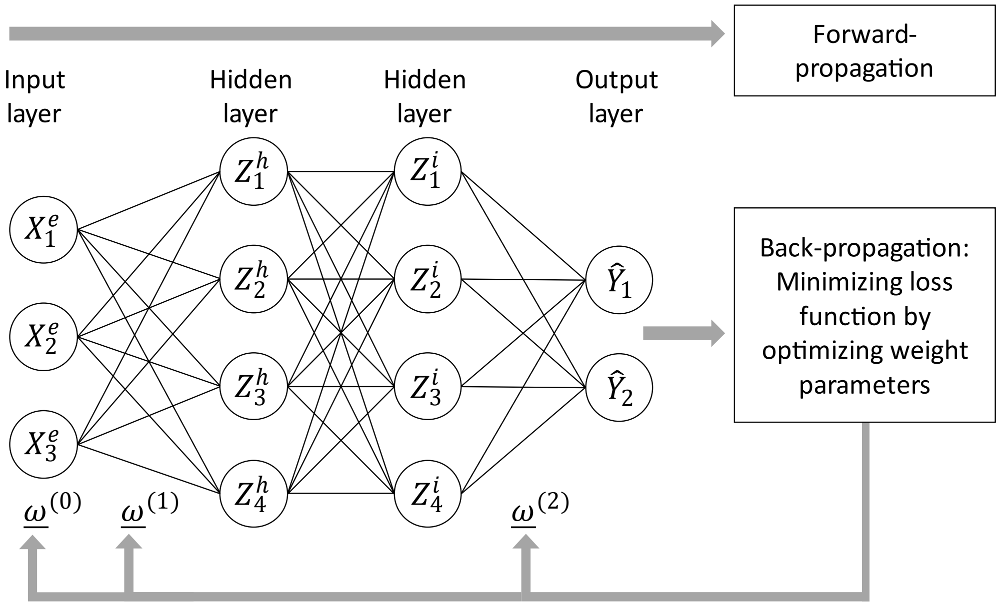


Figure 2. Neural network structure. The number of nodes and layers is exemplary and may vary according to the application. Adapted from Carranza-Abaid et al. (2020) and Zhou (2021).

While neural networks can be constructed rather easily with, for example, Python's NumPy libraries, other well-known libraries equipped with simple functions for layer creation, defining activation, and loss function include PyTorch and Tensorflow (den Bakker, 2017). Neural networks constructed in a matrix form may be introduced to a simulation software with the respective programming language, such as Fortran for Aspen Plus or Aspen HYSYS models. Additionally, neural network ML models can be implemented directly in some process simulation software with ML applications or add-ins included in the software.

2.3 Support vector machines

Wuest et al. (2016) described support vector machines (SVMs) as a ML method suitable for two-group classification problems and stated that their models work well with high-dimensional data. The basic working principle of SVMs is based on the model representing a decision boundary by using a subset of training examples as support vectors (Zhou, 2021). SVMs are especially useful when there is no need to model a probability distribution, but only divide data points into suitable classes hence a need for only an accurate decision boundary (Nasiriany et al., 2019). In process engineering, SVMs have been used to model, for example, the standard enthalpy of formation for hydrocarbons with support vector regression (Yalamanchi et al., 2019).

The term support vector refers to a margin with such a distance to the decision boundary (hyperplane), that there is at least one point on both sides of the boundary with that distance to the boundary, with these support vectors holding the margin planes (Nasiriany et al., 2019). Figure 3 illustrates the support vectors and their distance to the hyperplane. The margin may be defined either as hard-margin or soft-margin, with the former not allowing any points to violate the margin, and latter allowing some points to outlie it (Nasiriany et al., 2019). SVM models look rather similar to linear regression, however linear regression algorithm minimizes the error between the predicted and actual values, whereas SVM defines the allowable error margin, within which the model error does not count into the total loss (Zhou, 2021).

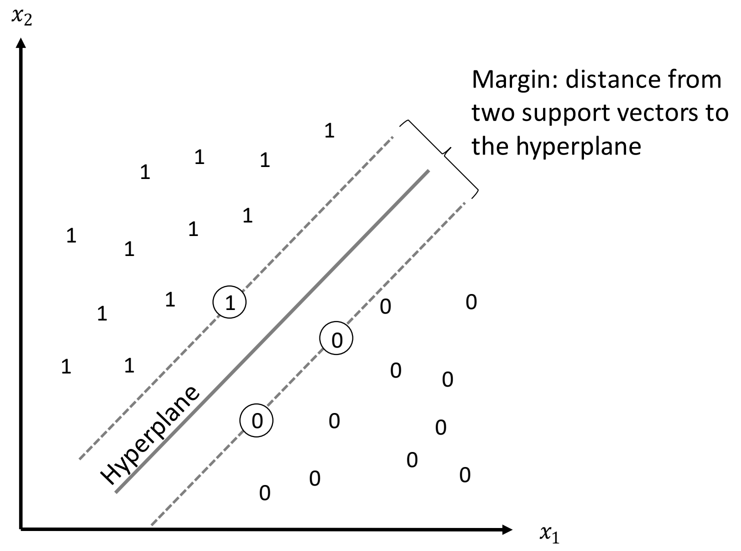


Figure 3. Support vectors (circled), hyperplane, and margin. Adapted from Zhou (2021).

2.4 Decision trees

Decision trees are used as classification algorithms, containing a root node leading to decision nodes based on the decision on going either left or right in the tree (Yu et al., 2022b). The decision nodes may lead to the next decision nodes, and finally, the outputs of a decision tree are defined in the leaf nodes of the tree (Zhou, 2021). The algorithm predicts classes of the inputs on each decision node based on if-then rules on tabulated data. As a simple example, the training data may contain a table of cats and dogs with different features. If the animal does not have pointy ears, it can be classified as a dog at the root node. Animals with pointy ears proceed to the next decision node, where the decision may be made on, for example, whether the animal barks (dog), or if has whiskers (cat). Eventually, all animals are classified to be either cats or dogs. The data can also be classified into more than two classes. Figure 4 illustrates the basic structure of decision trees.

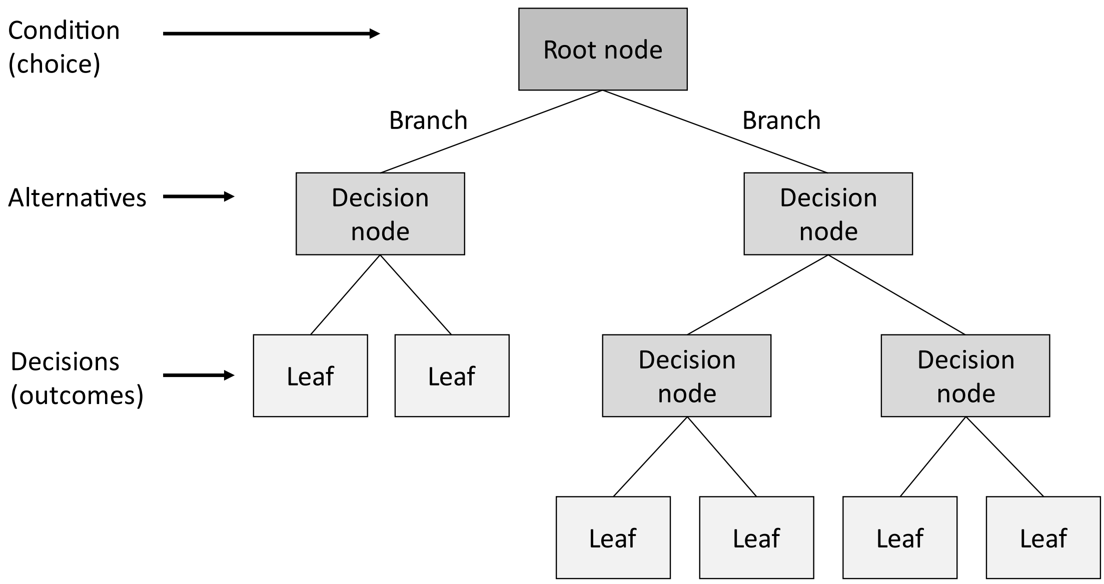


Figure 4. Decision tree structure with an exemplary number of decision nodes and leaf nodes (decisions). Adapted from Yu et al. (2022) and Nasiriany et al. (2019).

The features which the node splits the inputs based on are chosen in such a way that the inputs to the next decision node or leaf node contain as few classes of inputs as possible. In the cat-dog -example, if it is possible to make the decision so that the inputs to one of the next nodes only contain dogs or cats, that kind of features should be used. The splitting should stop when a node's inputs consist of only one class of inputs, the splitting would result in the tree exceeding a predefined maximum depth, or if the improvements in modelling error would be below a predefined threshold (Yu et al., 2022b). Multiple decision trees may be arranged into decision forests or random forests based on what is modelled (Theodoridis, 2015). Trees and tree ensembles are usually rather fast to train, and especially small trees may be human interpretable, which encourages their utilization in environments where the culture may be against models with black-box nature (Mowbray et al., 2022).

2.5 Model performance assessment

Multiple ways to assess the model performance have been introduced in literature. Training dataset is often split into training, testing, and validation datasets, so that the latter datasets can be used for model performance evaluation. The data split should be executed so that it conserves the original data distribution, to avoid introducing additional bias to the data (Zhou, 2021). Another method to split the data is cross-validation, where the training data is randomly split to two sets: training and validation datasets. Testing the model on a fresh dataset provides useful information on how to adjust the model (Rebala et al., 2019). To compensate for bias in small datasets or situations where there is no effective way of splitting the training dataset, bootstrap sampling can be utilized. There, random samples are copied from the original dataset *D* to a new dataset *D'*, which is then used for training, and the model is tested with dataset *D'/D* (Zhou, 2021). In essence, cross-validation uses a fresh dataset for validation, whereas in bootstrap sampling, the model may have seen some of the validation data points already during training.

During model training, the performance of ML models is measured by calculating the cost (also referred to as loss) function describing the difference between the model's output and the desired output, or distance of the model output from a statistical model (Mowbray et al., 2022). In training, the models aim at minimizing the cost algorithm, whose form depends on the used learning algorithm (Rebala et al., 2019). After each round of training, the model parameters are updated with a step size determined by the predefined learning rate, whose size should be optimized to avoid slow convergence (too small learning rate) or fluctuations causing a growing total cost (too large learning rate) (Zhou, 2021).

The accuracy or different models may be compared with different tools of retrospective evaluation. For example, sum of squared error, as shown in Equation 2.1, provides simple information of the total error during model training (Hwangbo et al., 2020). Kartal and Özveren (2020) as well as Zhu et al. (2020) assessed their ML models by determining the mean squared error (MSE) as shown in Equation 2.2 and coefficient of determination (R^2) as shown in Equation 2.3. A smaller MSE and R^2 closer to unity indicate better model performance than a larger MSE and smaller R^2. These tools are most used for different regression problems (Bender et al., 2022).

$$SSE = \sum_{i=1}^{k} \left(Y_{predicted} - Y_{actual}\right)^2 \tag{2.1}$$

$$MSE = \frac{1}{k} \sum_{i=1}^{k} \left(Y_{predicted} - Y_{actual}\right)^2 \tag{2.2}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{k} \left(Y_{predicted} - Y_{actual}\right)^2}{\sum_{i=1}^{k} \left(Y_{predicted} - Y_{mean}\right)^2} \tag{2.3}$$

where k is the number of training examples, $Y_{predicted}$ is the model output value, Y_{actual} is the value for that specific training example, and Y_{mean} is the mean value of the training examples (Kartal and Özveren, 2020).

It should be noted here that these indicators for model accuracy present the accuracy compared to the testing data used, and if the testing (and before that, training) data is of poor quality, also the ML model outputs are of poor quality even though the model accuracy might be high (Dobbelaere et al., 2021). Some public benchmark datasets have been developed for ML model performance assessment for specific applications such as molecular design, bioactivity prediction and property prediction for quantum chemical and physical properties (Bender et al., 2022). However, model evaluation with them must be done within rather specific model properties, such as how the model matches training data distribution. Therefore, these benchmark datasets should not be used to describe the general real-world performance of ML models. Although these benchmark datasets exist, none have been found to be particularly applicable to be used in the applied part of this thesis, nor is finding one a priority. Moreover, when considering the use of ML models in process simulations, it seems necessary to not only look at the model accuracy, but also qualitatively evaluate the user-friendliness of the developed model and the time required in data gathering and conditioning as well as optimizing the ML model structure.

3 Constraints in phase equilibrium and unit operations

Regarding the usage of ML models in engineering applications, and process engineering applications more specifically, it has been recognized that there is a risk that the models produce obvious or irrelevant correlations, or alternatively, results that disobey physical and thermodynamic laws (Mowbray et al., 2022). In order to model chemical processes accurately with relevant and sensible results, certain laws must be considered in the construction of ML models for process engineering applications. This section introduces the relevant thermodynamic laws governing phase equilibria as well as rules relevant to unit operation models that should be included in ML model designs when used to describe processes.

3.1 Thermodynamic laws in phase equilibria

In their work regarding surrogate modelling of vapor-liquid equilibrium (VLE), Carranza-Abaid et al. (2020) discussed the importance of proper analysis of thermodynamic variables. The thermodynamic laws presented in this section must be implemented in ML models used for modelling processes including phase change or when modelling VLE. Perhaps most generally, all systems modelled should obey the laws of thermodynamics. According to the first law of thermodynamics, energy cannot be created or destroyed, when considering the energy of both the system and its surroundings (Engel and Reid, 2014). The second law of thermodynamics introduces entropy in the form of Gibbs phase rule. The number of degrees of freedom – in embodiment, the number of independent variables – can be obtained with the Gibbs phase rule (Theodore and Ricci, 2010). To estimate thermodynamic quantities of a system accurately while designing a ML model, the Gibbs phase rule must be followed as shown in Equation 3.1

$$F = 2 - P + C - r \tag{3.1}$$

where F is the number of degrees of freedom, P is the number of phases, C presents the number of components, and r is the number of independent chemical reactions in the system (Theodore and Ricci, 2010).

According to Carranza-Abaid et al. (2020), physical constraints in phase equilibrium include that the component fractions in both vapor and liquid phases must sum up to 1, in embodiment, Equations 3.2 and 3.3 as presented below must hold.

$$\sum_i y_i = 1 \tag{3.2}$$

$$\sum_i x_i = 1 \tag{3.3}$$

where y_i is the fraction of component i in the vapor phase, and x_i is the fraction of component i in the liquid phase.

The correlation for enthalpy of phase change must be constrained when modelling systems including phase change. For example, carbon dioxide absorption systems often require solvent regeneration, where the duty depends on the enthalpy of phase change of the gas (Carranza-Abaid et al., 2020). However, phase change enthalpy of permanent gases cannot be measured in their pure state, which is why the enthalpy is described as a function of the solvent to which a gas is absorbed. In cases where experimental data on the system's enthalpy of phase change is not available, enthalpy of phase change may be obtained from the VLE data with the van't Hoff equation derived from the Gibbs-Helmholtz relation as shown in Equation 3.4

$$\frac{\delta \ln p_i}{\delta \frac{1}{T}} = -\frac{\Delta H_i}{R} \tag{3.4}$$

where p_i is the partial pressure of component i in the system (kPa), T is the temperature (K), ΔH_i is the enthalpy of absorption of gas i $\left(\frac{\mathrm{J}}{\mathrm{mol}}\right)$, and R is the gas constant $\left(\frac{\mathrm{J}}{\mathrm{mol\ K}}\right)$.

In electrolytic systems, an additional constraint is that overall local charges must sum up to 0 leaving no charge at thermodynamic equilibrium (Carranza-Abaid et al., 2020). Electro-neutrality holds when Equation 3.5 holds.

$$\sum_i n_i \Gamma_i = 0 \qquad 3.5$$

where n_i is the amount of component i in an electrolytic system and Γ_i is the ion charge in relation to a hydrogen ion.

Another relevant constraint for gas-liquid processes, such as absorption, mentioned by Carranza-Abaid et al. (2020) is the gas component concentrations in the liquid phase are described with non-thermodynamic variables, such as gas component loadings and weight percentages in a liquid solvent. The gas component loading should be equal to the relation of the apparent molar fractions of the gas component and the liquid solvent.

3.2 Physical constraints in unit operations

When modelling unit operations, the constraints discussed in this section must hold true when implementing the model. Perhaps most importantly, the mass and enthalpy balances of chemical processes must always hold. The general mass balance (Sinnott, 2005) is shown in Equation 3.6

$$IN + GENERATION = OUT + CONSUMPTION + ACCUMULATION \qquad 3.6$$

where IN and $GENERATION$ are the material input and generation in the process, while OUT, $CONSUMPTION$, and $ACCUMULATION$ are the material out of the system as well as consumed and accumulated during the process, all presented as amount of moles or units of mass per unit of time. In steady-state process with no chemical reaction, material generated, consumed and accumulated in the process are zero, and the equation reduces to $IN = OUT$ (Sinnott, 2005).

Similar equation as 3.6 holds for enthalpy balances at any process step, based on the first law of thermodynamics. In the enthalpy balance, the factors present the energy input, generation by an exothermic reaction, output, consumption by an endothermic reaction, and accumulation, with relevant units of energy. In a steady-state process with no chemical reaction, accumulation, generation, consumption of energy as well as the accumulation of mass are all zero. Differential energy balance, on the other hand, can be written as shown in Equation 3.7

$$\frac{\delta(E_I + E_K)}{\delta t} = (E_I + E_K)_{influx} + Q_r + W_r \tag{3.7}$$

where t is time (s), E_I is the internal energy (J) and E_K is the kinetic energy of the fluid in the system (J), Q_r is the heat input rate $\left(\frac{\mathrm{J}}{\mathrm{s}}\right)$, and W_r is the rate of work done into the system $\left(\frac{\mathrm{J}}{\mathrm{s}}\right)$ (Ravi et al., 2017).

Aside the balance equations, a process model should be constrained so that it does not produce fitting data by producing unphysical values, for example, negative mass flows or pressures, as output results. Additionally, output temperatures should be constrained to not have values below absolute zero, i.e., 0 K, or otherwise unphysical temperatures.

4 Machine learning models in steady-state process simulations

Traditionally, process modelling has been a way of utilizing different tools to perform tasks all the way from equipment sizing to plantwide simulations with different software, such as Aspen Plus, Aspen HYSYS, ProTreat, ChemCAD etc. (Hall, 2018). Steady-state simulations – also known as process flowsheeting – calculate heat and mass balances, equipment sizes, and cost estimations for unit operations and plants by representing the process with a mathematical model and solving it to study the behavior of the process (Foo et al., 2017). Some simulation software also enhance the models by incorporating vast amounts of physical and thermodynamic property data (Hall, 2018). Traditional process models must solve the predefined mathematical model for each iteration point to define the component properties and model outputs for each point, whereas when ML models are used in simulations, the simulation model is trained with, for example, previously generated simulation data, plant data, or VLE data (Foo et al. (2017), Zhu et al. (2020), Jirasek et al. (2020)). Training allows process simulations with an incorporated ML model to predict model output results without the need to solve the complex mathematical model equations for each iteration. As a result, using ML models in steady-state process simulations may result in, for example, shorter simulation times (Iftakher et al., 2022). Combining traditional thermodynamic models with ML models has also resulted in more accurate prediction of phase equilibrium (Del-Mazo-Alvarado and Bonilla-Petriciolet, 2022). These opportunities form the motivation to study the utilization of ML models in simulation purposes in this thesis.

Several ways of utilizing machine learning in process simulation applications have been introduced in the recent years. For example, by embedding a unit operation ML model into a simulation, the model predicts outputs during running the simulation (Kartal and Özveren, 2020). ML models may also be used parallel to traditional models, possibly enhancing the modelling by separately predicting outputs with a ML model and comparing the output results into simulation results generated without machine learning (Carranza-Abaid et al., 2020). Additionally, ML methods may be used to perform sensitivity analyses for simulations, and finally reduce the number of simulations required (Jones et al., 2019).

The following sections discuss the fundamentals of utilizing ML models in the different process simulation applications. Basic working principles when utilizing ML in steady-state process simulations are discussed in Section 4.1. Opportunities and challenges regarding ML model utilization in the different applications are discussed in Section 4.2. Additionally, potential applications in the chemical field are discussed in Section 4.3 along with a review on the opportunities and challenges related to the utilization of ML models in process engineering.

4.1 Basic working principles

The working principles and workflows presented in this section are most applicable to supervised learning applications, although by using different data and ML algorithms, also unsupervised learning models could be utilized. Regardless of the ML type used, the process of model development generally starts with the selection of suitable data for training the model (Kartal and Özveren, 2020). The steps following data selection depend on whether the ML model is embedded in a process simulation predicting simulation outputs, or as its own model outside the simulation environment. A generalized workflow for a ML model embedded in a simulation software is shown in Figure 5, where an ML algorithm, such as an ANN, can be included in the simulation flowsheet.

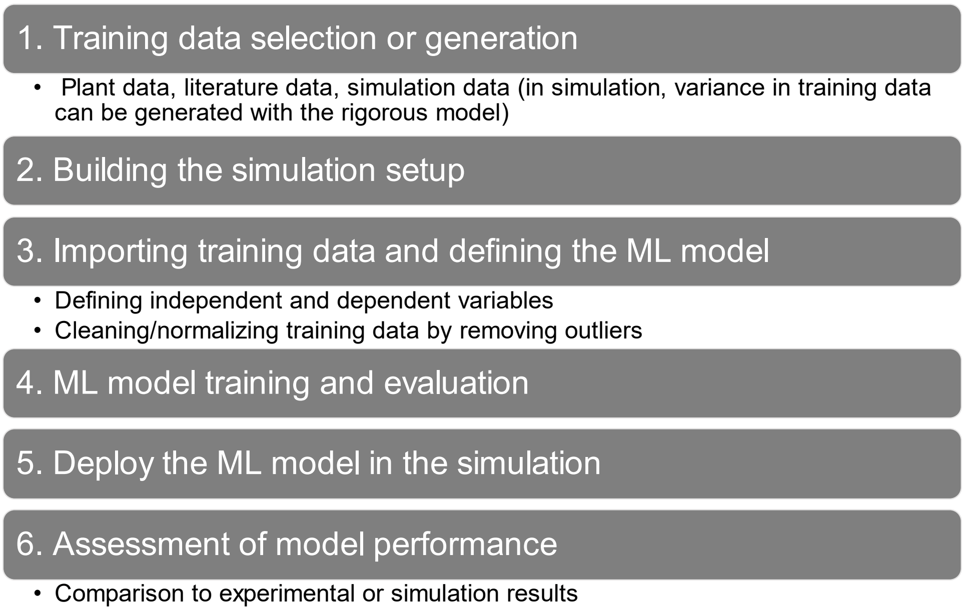


Figure 5. Workflow for applying ML models by embedding into process simulation software. Adapted from Kartal and Özveren (2020) and Beck and Munoz (2020).

If suitable data with sufficient variance can be found in plant data or literature, it may be timesaving compared to generating training data by simulating the process. However, by simulating the data, adequate variance can be generated, as variance may be lacking in plant data (Thebelt et al., 2022). Next, the simulation setup should be built including defining the component properties and building the simulation flowsheet. Depending on the simulation software, the training data should then be imported, and the ML model and its properties defined. For example, in Aspen Plus Hybrid Modelling, a neural network is created, training data is evaluated, the model is trained and potentially adjusted (AspenTech, 2022b). Finally, when sufficient model accuracy is reached, the neural network is deployed in the simulation flowsheet where it predicts the desired outputs (Beck and Munoz, 2020). Alternatively, the ML model could be created and trained outside of the simulation software and programmed into the user model of the simulation to be deployed as a unit operation model.

The workflow seems rather similar when using the ML model outside the simulation program, apart from the obvious difference in the environment for the ML model deployment. On one hand, the workflow presented above seems a bit more complex due to the need of building or programming the ML model into the simulation program. On the other hand, deploying the ML model outside of the simulation program requires the usage of a separate software for the simulation (if the ML model results are compared to simulation results), ML model programming and deployment, as well as data storage in Excel or other tabulated form. Workflow based on the work by Carranza-Abaid et al. (2020) is shown in Figure 6.

1. Training data selection or generation

- Plant data, literature data, simulation data (in simulation, variance in training data can be generated with the rigorous model)

2. Definition of the ML model architecture

- Definition of independent variables
- Training algorithm, sampling method, preprocessing function, loss function, accuracy target
- Cleaning/normalizing training data by removing outliers

3. ML model training and evaluation

4. Assessment of model performance

- Comparison to experimental or simulation results

Figure 6. Workflow for simulating processes with ML models without embedding the ML model into a simulation software.

This kind of ML models are often used as surrogate models. In process modelling, models mimicking complex systems or processes otherwise difficult, expensive, or impossible to model are referred to as surrogate models – also known as black-box models as they map output data to input data when their true relationship is unknown (Williams and Cremaschi, 2019). ML models have been introduced as a method to develop surrogate models with good accuracy, possibly shorter computational times and consistency with thermodynamic laws, while being a rather easy solution to implement (Carranza-Abaid et al., 2020).

4.2 Opportunities and challenges of machine learning in process simulations

Machine learning not only poses a vast variety of great opportunities to enhance the work of process modelling engineers, but also challenges may arise. Originally, the lack of extensive data prevented effective training of ML models, and consequently, the rise of ML models in the process industry was hindered by the inability to process the excessive amounts of data obtained from the operations (Venkatasubramanian, 2019). However, both data and computers capable of processing the data have become available in the latest decades, which allows the training and developing of ML models, but different challenges have been recognized (Mowbray et al., 2022). Nevertheless, these challenges are unlikely to overshadow all the positive effects attainable with ML models.

The current opportunities and challenges recognized regarding applying ML models in process simulations include, on one hand, improvements in computational time, but on the other hand, possibly time-consuming training of ML models. Some challenges may arise with data quality and quantity, interpretability, and the phenomenon of overfitting. However, if relevant thermodynamic and physical constraints are considered, ML modelling poses opportunities in improving the modelling of component properties and unit operations. These factors are discussed in subsections 4.2.1 through 4.2.5.

4.2.1 Computational and training time

One of the factors creating a lot of interest in ML within the process industry is the improvement in computational time, since a trained ML model may predict outputs without solving a complex mathematical process model an iteratively. Hence, improved computational time is perhaps the most often stated strength that has been found when applying ML models in process simulation applications. For example, Kartal and Özveren (2020) found that neural networks are able of calculating properties for complex systems with large databases faster than first principles simulation programs. The prediction of thermo-mechanical pulp refining process variables has been done accurately with the help of ML methods, while previously the computational times without the utilization of ML methods have been too high to implement all process factors in the simulation (Talebjedi et al., 2021). Dobbelaere et al. (2021) stated that in comparison to empirical models, ML models trained with fundamental model results as training data are more accurate yet still able to avoid the computational costs of fundamental models. In addition, not only the simulation itself may be made faster, but also the computational time required for sensitivity analysis has been reduced with polynomial chaos expansion in the work by Jones et al. (2019). The possibility to reduce calculation time has also been recognized in material technology, and more precisely in the simulation of polymer structure formation (Cassola et al., 2022).

While the model and sensitivity analysis execution speed may be considerably improved with ML models, the model training and model structure optimization may be time-consuming (Wuest et al., 2016). Depending on the training data, the model output may not be accurate from the get-go, hence the model or data may need adjustments and multiple rounds of retraining (Neuneier and Zimmermann, 2012). Therefore, the efficiency of model training is heavily affected by the quantity and quality of data available, which shall be discussed next.

4.2.2 Data quantity and quality

Although ML poses an opportunity to utilize the vast amounts of process- and simulated data generated during decades of process engineering, some issues regarding training data quality and quantity must be addressed. Rather self-evidently, the reliability of the training data should be assessed while choosing the suitable dataset, since large systematic errors in the training data surely results in inaccurate model outputs (Dobbelaere et al., 2021). Also, while the amount of available plant data from chemical processes is vast, plant data is sometimes not very information-rich due to the steady-state nature of some of the large processes (Mowbray et al., 2022). For ML models to predict processes accurately, as much as hundreds of thousands of training data points may be needed with sufficient variability within the data, since models trained with insufficient variability are unable of producing accurate outputs (Dobbelaere et al., 2021). Although variance could be generated with experimental data, it would not only be very expensive, but also time-consuming and thus not exactly feasible for training data generation (Linzner and Koeppl, 2021). However, Heese et al. (2020) concluded that simulated process data (i.e., data based on first principles models) can well be used in training both supervised and unsupervised ML models. Another useful means of gathering training data is to utilize process data from large databases, potentially various of them (Dobbelaere et al., 2021).

Too small datasets with noisy data points may result in a phenomenon where noise in the training data affects the model performance negatively, and the model is said to overfit the data (Iftakher et al., 2022). Therefore, training data quality and quantity also affect the potential model performance as the risk of overfitting depends upon it (Dobbelaere et al., 2021). Overfitting and some means to avoid it are further discussed below.

4.2.3 Overfitting and underfitting

If a model is overfitted, the model outputs fit training data well, but considerable oscillations occur between the data points (Bu and Zhang, 2020). The risk of overfitting occurs especially when training models with small datasets of potentially noisy data (Wuest et al., 2016). Overfitting may also occur when too many parameters (i.e., features whose value controls the learning process) are introduced into the model (Nasiriany et al., 2019). An overfitted model likely results in irrelevant outputs, making the model less useful in practical applications (Bu and Zhang, 2020). Figure 7 visualizes a simple model, where higher number of parameters has resulted in poor model performance on the testing dataset.

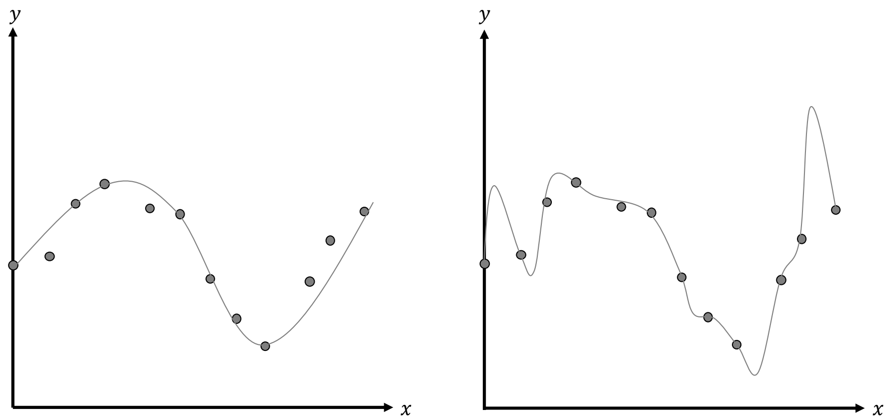


Figure 7. On the left, a model output that fits training data well. On the right, a model that fits the testing data points well, but severe oscillation occurs between data points. Adapted from Bu and Zhang (2020).

While increasing the amount of training data points decreases the effect of outliers in the training data, the outliers may also be removed before model training, as already discussed within the data quality aspect (Dobbelaere et al., 2021). Another general method to avoid overfitting of a model is to implement a criterion that seizes model training based on the observed change in performance on the training dataset (Dobbelaere et al., 2021). Bu and Zhang (2020) used a regularization method that implemented a penalty function for the model ensuring that the model complexity was kept on a desired level. Different dimensionality reduction techniques may also assist in avoiding the overfitting issue (Salam et al., 2021).

The opposite phenomenon of overfitting is underfitting, meaning that the general properties of training data is not learnt well. In contrast to overfitting, underfitting is often rather easily overcome, since it is due to the poor learning ability of a model, which can be enhanced, for example, by adjusting the model architecture to include more training epochs or branches (Zhou, 2021). Since it can generally be avoided, underfitting is not considered much of a threat or a challenge in ML, whereas overfitting as a fundamental difficulty requires actions to be alleviated or overcome.

4.2.4 Improved modelling of properties and unit operations

Apart from the calculation speed, ML models pose the opportunity to model, for example, processes and equipment types, that otherwise cannot be modelled with first principles models alone (AspenTech, 2020a). As an example, an ANN model trained with an adequate amount of data describing different feedstock types (different biomasses in the particular case), the model is able to predict properties for completely new feedstocks that lack experimental data (Kartal and Özveren, 2020). Reynel-Àvila et al. (2019) were able to predict binary interaction parameters for liquid-liquid systems present in biofuels production with ANN-enhanced Nonrandom Two Liquid Theory (NRTL) model. In their work, the predicted parameters had considerably higher accuracy to the experimental data than the parameters produced with conventional NRTL. These kinds of opportunities to model less known or more sustainable feedstock mixtures are of particular importance due to the pressure to develop towards a more circular economy and study the utilization of different new feedstocks.

Where more traditional processes are considered, a fluid catalytic cracking (FCC) unit is generally seen as a complex and non-linear process to model, although especially the theoretical behavior of the unit can and has been modelled. ML models, however, pose an opportunity to model, for example, the catalyst saturation levels in the FCC unit (Steurtewagen and Van den Poel, 2020). In their work, Steurtewagen and Van den Poel (2020) were also able to simulate the control of the feedstock composition for optimal catalyst saturation. There was a predictable relationship between the catalyst saturation level and the fraction of recycled heavy residue from downstream process in the feed, and the stance of an input valve controlling the feed composition was simulated.

4.2.5 Interpretability

Where challenges of ML are considered, most authors mention the difficulty in human interpretation of ML models. Especially ANNs are perceived as black-box models, since the model hyperparameters, such as the number of nodes in an ANN, cannot really be deduced to a physical meaning (Dobbelaere et al., 2021). The lack of interpretability may negatively affect the credibility of the models among other users (Pan et al., 2022).

Additionally, if other model users than the developer do not have a straightforward understanding of the model, maintenance of the model may prove very difficult if the developer is not available at some point during the model's lifecycle (Mowbray et al., 2022). Carranza-Abaid et al. (2020) recognized that the lack of interpretability has so far caused skepticism towards ML models, which may hinder the implementation of ML methods and ANNs in particular.

However, to tackle the issue of interpretability, some methods to increase the model interpretability have been introduced. Components of black-box models may be analyzed by visualizing them as feature maps of neural networks, or in case of random forests, the minimal depth distribution of the forest may be used to analyze the structure of the trees (Molnar et al., 2021). Moreover, combining first principles with the ML model structure not only enhances the ML model accuracy and leads to more sensible results, it can also help improve model interpretability (Carranza-Abaid and Jakobsen (2022), Bikmukhametov and Jäschke (2020)). First principles can be introduced to ML models in multiple different ways, depending on the most useful way of utilization for the modelling application. After comparing five methods to combine first principles and ML models, Bikmukhametov and Jäschke (2020) concluded that the highest accuracy and improved interpretability can be obtained with feature engineering, or creating linear meta-models of the feature engineered models. In feature engineering, physically meaningful features were created and used in model training instead of raw measurements, which resulted in a simpler ML model. The linear meta-models produced a weighted sum of the predictions by multiple feature engineered models to describe the system in question. The model accuracy was increased by tuning the weights of the sub-models with linear regression methods.

4.2.6 Constraints and output quality

As already discussed in Section 3, ML models are prone to producing obvious or misleading correlations in engineering applications if the models are not constrained to obey physical and thermodynamical laws. Without the applications of relevant constraints, ML models have more degrees of freedom to fit data, but the lack of thermodynamical consistency restrains them from being regarded as a relevant method to model process thermodynamics (Carranza-Abaid et al., 2023). The same condition applies for physical constraints as well when modelling unit operations. Therefore, the relevant thermodynamic laws governing phase equilibrium as well as the physical constraints regarding unit operation modelling covered in Sections 3.1 and 3.2 should be carefully implemented in ML models to obtain relevant model outputs.

The risk of irrelevant correlations going unnoticed is present especially when data scientists are modelling engineering processes, hence it has been proven more efficient to train engineers on data science and ML rather than training data scientists on engineering topics (Mowbray et al., 2022). Nevertheless, the training of engineers on data science and ML tools must be sufficient to avoid misuse of ML tools due to lack of understanding the tools (Dobbelaere et al., 2021). Especially in the presence of the previously discussed interpretability issues regarding deep learning methods, adequate understanding of the used methods should be ensured.

4.3 Relevant applications in process simulations

The number of publications regarding the use of machine learning to model different steady-state processes has increased in the latest years, as the excitement around ML has again grown within the process industry. To map the potential of ML models in steady-state process simulations, literature on different ML models of steady-state processes is reviewed in this section. The review is divided into subsections discussing unit operation models and thermodynamic system models, and the subsections discuss the different data types that have been used in training the models in literature.

4.3.1 Unit operation models

When using ML to model unit operations, the model usually acts as a surrogate model of said unit operation. As discussed in Section 4.2.2, simulated and thus first principles - based process data may be readily applied to both supervised and unsupervised machine learning models (Heese et al., 2020). Although process data obtained from plants is sometimes poor in information due to its little variability, studies on interesting applications of ML models trained with plant data have been published in recent years. Hybrid modelling refers to combining mechanistic models with ML models, where advantage lies in simplifying the model by replacing nonlinear parts of the process with ML models, reducing the computational time of the whole model (Carranza-Abaid et al., 2023).

Models based on conventional first principles steady-state simulation results have been reported by, for example, Heese et al. (2020), who trained ML models on simulation data obtained from simple distillation flowsheets: a single equilibrium stage, a distillation column with 20 stages, as well as a pressure swing distillation. It was stated that as the number of equilibrium stages increased, the dimensionality of the obtained simulation datasets increased rapidly, and thus they used unsupervised learning, namely, dimensionality reduction, to simplify the data. Consequently, the labelled simulation data obtained from the flowsheets was used to create surrogate models of the flowsheets by training supervised learning models utilizing Gaussian process regression (GPR) (Heese et al., 2020). GPR is a nonparametric ML framework that produces a probability distribution and the uncertainties of the predicted values (Wang, 2022). The surrogate model performance was estimated with the coefficient of determination (R^2), and by studying the performance of the three flowsheet surrogates as a function of training dataset size, it was concluded that larger datasets predicted the unit operations more accurately, and the accuracy of the simpler flowsheet surrogate model (single stage) was higher than the more complex one (pressure swing). The surrogate models predicted the outputs of the unit operations with R^2 close to unity with the larger datasets, and the authors concluded that ML tools provided important knowledge on process data complexity, and stated the importance of further studying of the application (Heese et al., 2020).

Where unit operation models trained with plant process data are concerned, sensory data from an FCC unit has been utilized to predict the catalyst saturation level (Steurtewagen and Van den Poel, 2020). In their work, Steurtewagen and Van den Poel (2020) reduced the dimensionality of the plant data, which contained pressure, temperature, flow rate, and gas concentration measurements as well as laboratory data on catalyst saturation levels, by principal component analysis. Consequently, supervised learning was applied in the form of a random forest, common linear regression as well as eXtreme Gradient Boosting (XGBoost) machine, whose outputs and performances were then compared. The random forest predicted catalyst saturation the most accurately. Liquid feed to the unit consisted of oil from vacuum distillation unit and recycled heavy residue from downstream of the FCC unit, and higher amount of the heavy residue in the feed resulted in faster catalyst saturation. Based on the causation between the feedstock composition and catalyst saturation level, the authors were able to produce a simulation to optimize the feedstock mixture for optimal catalyst saturation. In the simulation, the stance of an input valve controlling the feedstock composition was modelled (Steurtewagen and Van den Poel, 2020).

Also neural networks and support vector machines have been utilized in plant data-based ML models. For example, Hwangbo et al. (2020) predicted the characteristics of nitrous oxide emissions from a wastewater treatment plant based on long-term plant data. The data included influent and air flow rates, concentrations of ammonia, nitrate, and dissolved oxygen, temperature, and the concentration of nitrous oxide (N_2O) in the liquid phase. The model performances were assessed by calculating R^2s and MSEs for the different model candidates to estimate the accuracy of liquid N_2O concentration prediction. The models utilizing GPR produced highly accurate outputs, and the study provided the authors new insight on the process, as temperature, ammonia and nitrate concentrations were recognized as the most important emission predictors (Hwangbo et al., 2020).

Hybrid modelling has been introduced already in the previous wave of neural network enthusiasm in the early 2000s, when the first principles equations to predict pulp properties were integrated in a neural network with decent accuracy (Aguiar and Filho, 2001). More recently, hybrid models created by Carranza-Abaid et al. (2022) included ANNs and algorithmically structured neural networks (ASNNs) to predict the outputs of a flash drum. The unit operation was modelled with a fully connected neural network as a surrogate for the iterative calculations providing the solution for the mass balances in the flash evaporation. Three hybrid morel structures were introduced: in a serial structure a neural network and the mechanistic model worked in a sequence, where inputs were first fed to the ANN, whose outputs and inputs were consequently fed to the mechanistic model containing the first principles equations. In a parallel structure, the ANN and mechanistic model independently processed the input data, and their outputs were summed. Finally, an integrated model where the first principles equations were included in the architecture of the ASNN. Including the first principles and thus the physical and thermodynamic constraints into the neural network made the interpretation of the otherwise black-box model easier and sped up the training process while producing more accurate results (Carranza-Abaid and Jakobsen, 2022).

To provide an overview of different applications of unit operation ML models, the aforementioned applications and some others are presented in Table 1. Generally, plant data and simulation data have both been used in model training with encouraging results. Therefore, it seems that there are suitable applications for different data despite all of them have their weaknesses, namely little variance in plant data in cases of highly steady-state processes, and potentially time-consuming generation for simulated data (Thebelt et al., 2022). Within the different learning algorithms, studies on different neural networks are perhaps the most frequently published, however multiple successful SVM applications were found as well.

In Table 1, the method used to assess the model performance from the articles are presented with the value obtained for the accuracy. Indicators such as determination coefficient and absolute average relative deviation were included in the table instead of indicators such as mean squared error, since the errors predicted in the applications may differ from each other with multiple orders of magnitude. Even so, the performance indicators presented for different processes in the different articles are not exactly comparable to each other. Thus, they should be considered more of a rough overall indicator showing that all in all, results with rather high accuracy have been obtained in the presented studies. Moreover, a verbal conclusion presented in the respective article is presented for each application. The verbal conclusions further indicate the importance of including first principles/physical constraints in the ML models for improved interpretability and sensible results.

Table 1. Collection of different unit operation ML model applications.

Application, reference	Learning method	Data source	Model performance	Conclusion
Simulations of single equilibrium stage, distillation column, pressure swing distillation (Heese et al., 2020)	Supervised: GPR Unsupervised: Dimensionality Reduction	Simulated data	R^2_{GPR} up to 1	Accurate modelling results: further studies and inter-disciplinary cooperation was recommended
Prediction of FCC catalyst saturation level (Steurtewagen and Van den Poel, 2020)	Supervised: Random forest Lin. regr. XGBoost	Plant data	$R^2_{RF} = 0.78$ $R^2_{LR} = 0.56$ $R^2_{XGB} = 0.75$	An ML based feedstock composition optimization pipeline was proposed
Prediction of wastewater treatment plant NO_x emissions (Hwangbo et al., 2020)	Supervised: ANN	Plant data	$R^2 = 0.98$	Promising and accurate modelling results, further studies planned and recommended
Simulations of flash drum outputs (Carranza-Abaid and Jakobsen, 2022)	Supervised: ANN ASNN	Simulated data	$AARD^*_{ANN} = 0.03 - 0.07$ $AARD^*_{ASNN} = 0.009 - 0.04$	Combining first principles and ANNs method produced accurate results with better interpretability than just ANNs
Prediction of multiphase flowrate in a petroleum production system (Bikmukhametov and Jäschke, 2020)	Supervised: KNN XGBoost	Plant data	$RMSE_{NNs} = 0.03 - 0.05$ $RMSE_{XGB} = 0.04 - 0.05$	Combining first principles and ML method produced accurate results while maintaining interpretability
Prediction of H_2 production rate, CO_2 emissions and production cost in methane steam reforming unit (Byun et al., 2021)	Supervised: SVM GPR Decision tree regression	Simulated data	$RMSE_{SVM} = 0.01 - 0.08$ $RMSE_{GPR} = 0.01 - 0.05$ $RMSE_{DTR} = 0.002 - 0.05$	Most influential factors to the H_2 plant performance were recognized with the ML model
Prediction of syngas lower heating value (LHV) (Kartal and Özveren, 2020)	Supervised: ANN	Simulated data	$R^2 \geq 0.99$	ANN model could estimate LHV for new biomasses

*Absolute average relative deviation, small AARD indicates better model performance (Heese et al., 2020).

Commercial simulator providers have also started to provide their own solutions for ML applications within process simulations. For example, Aspen Plus and Aspen HYSYS offer a separately licensed tool for integrating and training ML models as a part of the first principles models, which according to the software provider's pamphlet, enables modelling systems, such as batch processes, fluidized bed processes, bio-process reactors and fermenters as well as complex refining units, which are difficult to model with only first principles (AspenTech, 2020a). The AspenTech AI training tool is marketed as a tool that can increase simulation speed with more accurate results than either first principles or ML models alone, without requiring significant prior expertise in machine learning from the user. The ML models are trained with either data generated from numerous simulations or plant data, constrained to obey thermodynamic and physical laws, and deployed as a part of the simulation flowsheet (AspenTech, 2020a).

4.3.2 Thermodynamic models

Thermodynamic models simulate pure component properties, such as enthalpy of formation, as well as phase equilibrium behavior, i.e., vapor-liquid, liquid-liquid, or solid-liquid equilibrium (VLE, LLE, or SLE) of mixtures. Activity coefficient models, such as NRTL, Wilson, or Universal Quasi-chemical Theory (UNIQUAC), model phase equilibrium by determining liquid phase activity coefficients, which describe the non-ideality of a multi-component liquid mixture (Farajnezhad et al., 2016). The models include binary interaction parameters, which are fitted to experimental data to predict interactions in the liquid phase. Pure component properties are modelled with the model's pure component parameters and correlations. Therefore, with correct binary interaction parameters and pure component properties, the phase behavior of the mixture can be predicted correctly. Predictive activity coefficient models, such as UNIFAC, predict binary interaction parameters based on the molecular structure of a component (Yu et al., 2022a). When predicting the aforementioned properties with ML models, training may be executed with phase equilibrium data from different databases (i.e., experimental data), or data generated with simulations. Experimental data is often obtained from literature, however in some cases, data may be available from experiments performed by the model developer (Lopez-Zamora et al., 2021).

Del-Mazo-Alvarado and Bonilla-Petriciolet (2022) enhanced the NRTL model with ANNs to predict binary interaction parameters. Five binary systems, including methanol + cyclohexane and water + ethyl acetate among others, were studied with different NRTL objective functions, with and without ANNs. When ANNs were used, they predicted the binary interaction parameters for LLE and VLE modelling with state variables temperature and pressure as input variables. The phase equilibrium obtained with the predicted binary interaction parameters was fit to LLE data as a function of temperature. The most accurate model outputs were obtained with ANN -enhanced NRTL models (Del-Mazo-Alvarado and Bonilla-Petriciolet, 2022). The LLE modelling of a methanol and cyclohexane system with the aforementioned ANN-enhanced NRTL model and normal NRTL model are presented in Figure 8. It can be seen that the model enhanced with an ANN performed better than the normal NRTL model especially in temperatures higher than 290 K. Due to the encouraging results, the authors recommended further studying of the concept with more sophisticated ML methods and different local composition models for more accurate thermodynamic framework.

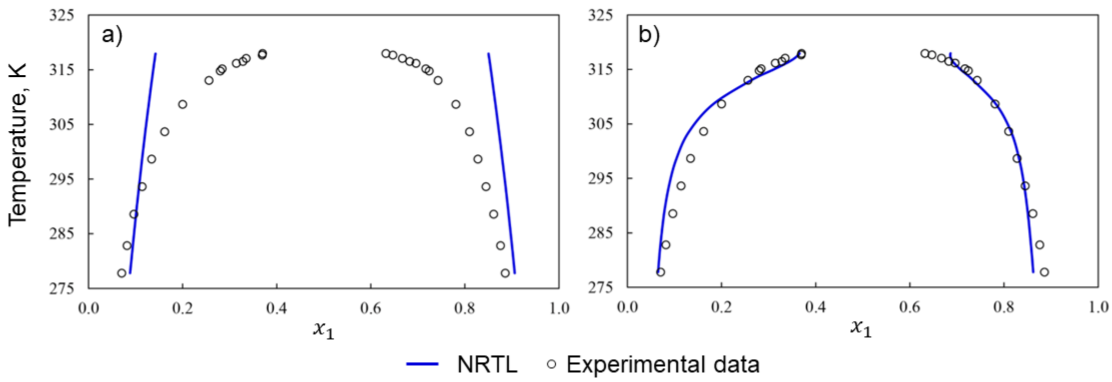


Figure 8. LLE modelling of methanol and cyclohexane. a) NRTL with integrative objective function. b) NRTL with integrative objective function, ANNs and phase stability criterion. Adapted from Del-Mazo-Alvarado and Bonilla-Petriciolet (2022).

ANN application on NRTL model has also been used to enhance phase equilibrium prediction for systems present in biofuels production, including ethanol, glycerol, and different ethyl esters, in the study by Reynel-Ávila et al. (2019). In this study, ANNs were used to predict the binary interaction parameters with temperature and fluid composition as inputs. When the phase equilibrium predicted with the ANN -enhanced NRTL model was fit to experimental LLE data, it was concluded that the enhanced model had better LLE data fitting capabilities compared to the original NRTL model. The authors concluded that the model could be utilized in process simulations for improved LLE estimation.

Jirasek et al. (2020) developed a matrix completion method (MCM) to predict activity coefficients for binary mixtures. In the matrix completion problem, the training data (experimental values from Dortmund Data Bank (DDB)) formed a matrix, where columns and rows corresponded to solutes and solvents, and the matrix entries were the values of the activity coefficients for the systems. The MCM algorithm predicted the missing matrix entries, i.e., the activity coefficients for the missing binary mixtures. To evaluate the MCM model performance, the predicted activity coefficients were compared to such experimental values from DDB that were previously left out of the training dataset. The model results were compared to UNIFAC-Dortmund model, and it was concluded that the MCM model was able to predict activity coefficients with high accuracy for a larger number of predictions than the UNIFAC-Dortmund model. MCM has previously been more known from being used in content recommendation in streaming services such as Netflix, however in this study, its utilization in chemical engineering was deemed simple and useful even with sparse data (Jirasek et al., 2020).

In the publication by Yalamanchi et al. (2019), support vector regression (SVR) and ANNs were trained with large literature datasets to predict the standard enthalpy of formation of alkanes, alkenes, and alkynes. In this study, SVR model performed better based on computational cost and more extensive hyperparameter tuning (Yalamanchi et al., 2019). According to the authors, both types of ML models performed better than traditional group additivity models, but room for improvement was left particularly when ML models were trained with smaller, extreme-valued datasets.

Carranza-Abaid et al. (2020) modelled the VLE behavior of the ternary system of carbon dioxide, methane, and water with success in the accuracy of predicted carbon dioxide fraction in the vapor phase. The model was thermodynamically constrained to produce relevant outputs, and the model was said to be easily extensible to other thermodynamic variables, such as activity coefficients or heat capacities. A slightly different approach to utilizing ML models in thermodynamic system predictions was introduced, when classification algorithms were developed to predict the number of phases in naphtha/water blends (Lopez-Zamora et al., 2021).

An overview of studies on thermodynamic ML models, including the ones presented above, is presented in Table 2 similarly as presented previously for unit operation models. Literature data including experimental data on VLE, LLE and other properties seem to be most commonly used training data, but also simulated data has been used. Overall, promising results have been obtained with thermodynamic ML models as well, and authors generally encourage further studies in utilizing ML methods in process simulations. An abundance of studies on both thermodynamic and unit operation models have been published in the past five years or so as the interest towards ML has increased. The studies are mostly optimistic that ML methods shall be helpful and can potentially enhance process simulations in the future. Nevertheless, also the downsides of ML methods, namely issues with quality data acquisition, time-consuming model training and adjustment, and poor interpretability are broadly recognized. Nevertheless, also traditional thermodynamic models often predict component properties with different regression algorithms, which apart from linear regression are not much easier to interpret.

Table 2. Collection of different thermodynamic ML model applications.

Application, reference	Learning method	Data source	Model performance	Conclusion
Prediction of binary interaction parameters with ANN enhanced NRTL (Del-Mazo-Alvarado and Bonilla-Petriciolet, 2022)	Supervised: ANNs	VLE and LLE data	MME* 13 % for best-performing configuration	NRTL calculations were improved, further study of ML models recommended
Prediction of binary interaction parameters with SPT-NRTL (Winter et al., 2022)	Supervised: ANNs: SPT-NRTL	Simulated data, VLE + P_{vap} data	Mean absolute error between 0.1 and 0.5	Near-experimental accuracy
Enhancing phase equilibrium calculations for ternary systems (Reynel-Ávila et al., 2019)	Supervised: ANNs	LLE data	$R^2 \geq 0.99$	ANNs-NRTL more robust than NRTL, study of other mixtures recommended
Prediction of activity coefficients (Jirasek et al., 2020)	Supervised: MCM	Literature data	Absolute error below 0.1 for 50 % of the data	Promising results, further research recommended
Prediction of standard enthalpy of formation of hydrocarbons (Yalamanchi et al., 2019)	Supervised: SVR ANN	Literature data	$R_{SVR}^2 = 0.99$ $R_{ANN}^2 = 0.99$	Better estimations possible with ML, extendable to other applications
Prediction of VLE behavior of a ternary system (Carranza-Abaid et al., 2020)	Supervised: ANNs	Simulated data	MAARD** 0.49-8.03 %	Accurate, consistent, and fast alternative for surrogate modelling
Prediction of the number of phases in water-naphtha blends (Lopez-Zamora et al., 2021)	Supervised: KNN*** SVM	Exp. data	KNN precision up to 99 % SVM precision up to 98 %	SVM model enhanced phase number prediction
Prediction of melting points of ionic liquids (Wang et al., 2021)	Supervised: SVM	Literature data	$R^2 = 0.85$	SVM model was validated reliable, problems with data quantity and interpretability

*Mean modelling error, small MME indicates better model performance (Del-Mazo-Alvarado and Bonilla-Petriciolet, 2022).
**Model average absolute relative difference, small MAARD indicates better model performance (Carranza-Abaid et al., 2020).
***K-nearest neighbor algorithm

APPLIED PART

The aim of the applied part of this thesis is developing, training, and assessing the performance of machine learning models of amine absorption. Absorption is first modelled in Aspen Plus with a rate-based, reactive column model. Training data is then generated for ML models by running sample cases with Aspen Plus Multi-Case tool. Different ML tools provided by Aspen Plus, including First Principles (FP) Driven Hybrid Modelling, Reduced Order Modelling (ROM) and AI Driven Hybrid Modelling tools are used to create ML models, or in this aspect hybrid models, of an amine absorber column. The hybrid models created with the tools are referred to as FP driven models, AI driven models and ROMs. With FP driven models, the effect of the size and narrowness of the training data on the hybrid neural network model performance are studied. The models are validated by running a set of test cases on the hybrid models and comparing the resulting component flows and stream temperatures on the original rate-based column's results. In ROM and AI driven modelling, constrained neural networks and linear regression model are created, respectively Training datasets of similar size and wideness that seemed to provide the best-performing FP driven model are used. The best FP driven model, ROM and AI driven model are validated with a wider set of test cases to study the extrapolation abilities of the models.

In the applied part, amine absorption process and the methods used in data generation and conventional as well as hybrid modelling are first presented. In the results section, the accuracy of the developed hybrid models when interpolating and extrapolating the models is discussed. Discussion also includes the amount of training data, since collecting or generating the data is often time-consuming, and handling large datasets may be tedious. When creating neural networks for FP driven models, the sufficient model complexity is evaluated and discussed. Additionally, the advantages and disadvantages of hybrid modelling compared to conventional modelling approaches are evaluated. Another important discussion point is what kind of background knowledge is necessary for effectively working with the ML tools provided by Aspen Plus, since they do not require programming knowledge but allow editing the ML model properties.

5 Amine absorption process

Amine absorption is a gas-liquid separation process used to treat sour gas in oil refineries, and more recently also flue gases to capture carbon dioxide. Different amines, such as methyl diethanolamine (MDEA), diethanolamine (DEA), and diisopropylamine (DIPA) are used to remove primarily hydrogen sulfide (H_2S) from sour gas in gas sweetening processes, and carbon dioxide (CO_2) from flue gas in carbon capture processes. In the absorption processes, the aqueous amine solution reacts with the removable gases and absorbs them, after which the rich amine solution is regenerated by stripping. Regenerated lean amine and a stream of makeup amine are fed back to the absorber column. (Stewart, 2011) The process flowsheet of amine absorption process is shown in Figure 9.

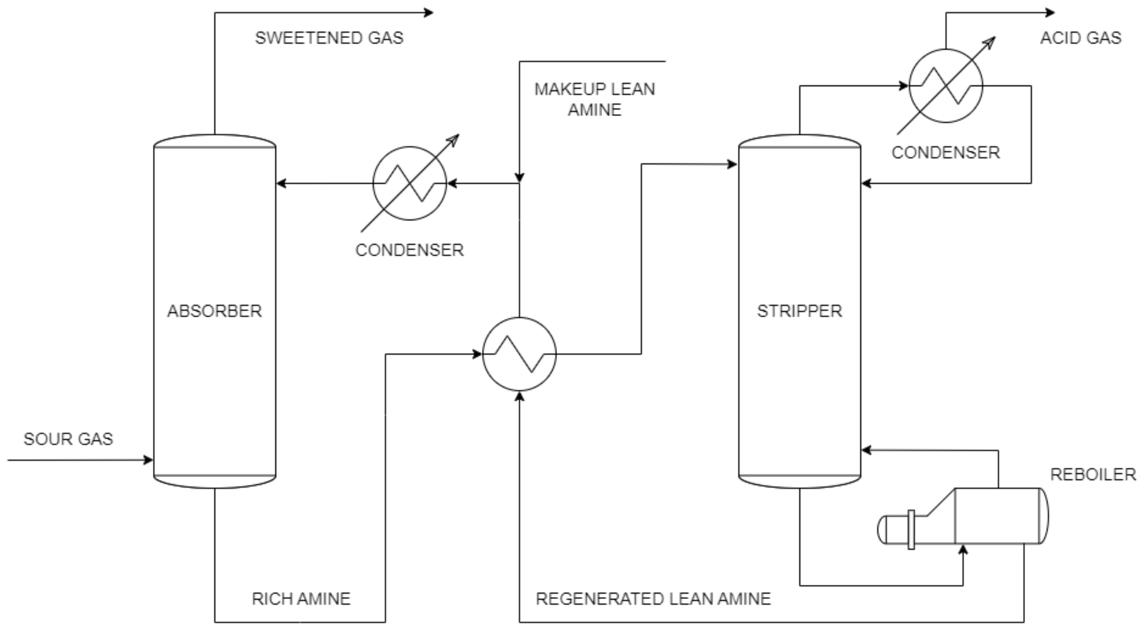


Figure 9. Amine absorption and rich amine regeneration process flowsheet. Adapted from Mudhasakul et al. (2013).

The applied part of this thesis focuses on modelling MDEA absorption, which is a process most often used in fossil refineries to remove H_2S from sour gas in the presence of CO_2 to meet gas sales regulations, which limit the H_2S content in sold gas to 4 ppm by volume (Stewart, 2011). Part of the CO_2 is also absorbed in the process, and with modified amine concentrations and flows, most of the CO_2 could be removed as well. However, in the simulations used in this thesis, MDEA removes most of the H_2S, and approximately half of the CO_2.

In the applied part of this thesis, the possible benefits of utilizing machine learning in simulating an MDEA absorber column are studied. Amine absorption is a reactive unit operation with rate-controlled reactions. Thus, it can be simulated with a rate-based column model using an electrolyte thermodynamic model, such as ELECNRTL (Zhang and Chen, 2010). Due to the inclusion of complex thermodynamics, reactions, and mass transfer models, the rate-based simulation of amine absorption may require a long calculation time and have difficulties in reaching convergence. These issues hinder its use as a part of a larger flowsheet simulation. Therefore, it is of interest to study whether more effective modelling can be achieved with machine learning methods.

6 Methods

The work related to the applied part of this thesis is executed with different tools related to Aspen Plus V14. Aspen Plus is first used to configure the base simulation of the MDEA absorption, as described in Section 6.1. Next, ML model training data is generated with Aspen Plus Multi-Case tool by varying the incoming streams to the base models, which is described in Section 6.2.

In this work, the Aspen Plus First Principles Driven Hybrid Modelling, AI Driven Hybrid Modelling and Reduced Order Modelling tools are used to model MDEA absorption. The FP driven models are created in Aspen Plus (V14), and AI driven models and ROMs in Aspen AI Model Builder (AIMB). According to AspenTech (2020a), the AI driven models are intended for situations where there is an abundance of plant data but no model, and the goal is to develop a model rapidly. ROMs aim at simplifying complicated simulations with long computational times by replacing unit operation models with machine learning models. In FP driven modelling, a neural network is constructed with chosen simulation input variables as its outputs, which are trained so that the model's results obey training data. Configuring machine learning models with Aspen Plus Hybrid Modelling tools is described in sections 6.3 through 6.5. Finally, the methods to assess and validate the developed hybrid models are presented in Section 6.6.

6.1 Amine absorber simulation setup in Aspen Plus

The simulation of amine absorption for training data generation is based on the *ELECNRTL_Rate_Based_MDEA_model.bkp* -absorption model configured by AspenTech (2020b). The absorber is modelled with the Rate-Based Distillation model of the RadFrac column model in Aspen Plus V14 with electrolyte NRTL (ELECNRTL) as the thermodynamic model for liquid properties, and Redlich-Kwong (RK) equation of state for vapor properties. Suitable electrolyte reaction equations and their rate constants are used to solve the sour gas component reaction rates in the column. The simulation file includes data, such as binary and ternary VLE data, excess enthalpy data, heat capacity data as well as absorption heat data, collected by AspenTech (2020). The data is used for the regression of Henry's constant parameters, NRTL interaction parameters, and interaction energy parameters.

The simulation flowsheet is configured with two incoming streams. Amine regeneration and recycling is not included in the base model. The original model by AspenTech is using the true components calculation approach, which reports all ions in the liquid streams. To produce simulated training data with no ions in the product streams, the flowsheet in this case is calculated with apparent components, and only the RadFrac column is calculated with true components as simulation approach. Heaters with no temperature or pressure change are inserted to feed and product streams as DUMMY1-3 to produce apparent component streams. The flowsheet of the model can be seen in Figure 10. In the flowsheet, the stream LEANIN is the lean amine stream into the absorber, and the stream GASIN is the sour gas stream into the absorber. Streams GASOUTT and RICHOUTT present the cleaned gas and rich amine streams with true components, whereas GASOUTA and RICHOUTA present the same streams with apparent components.

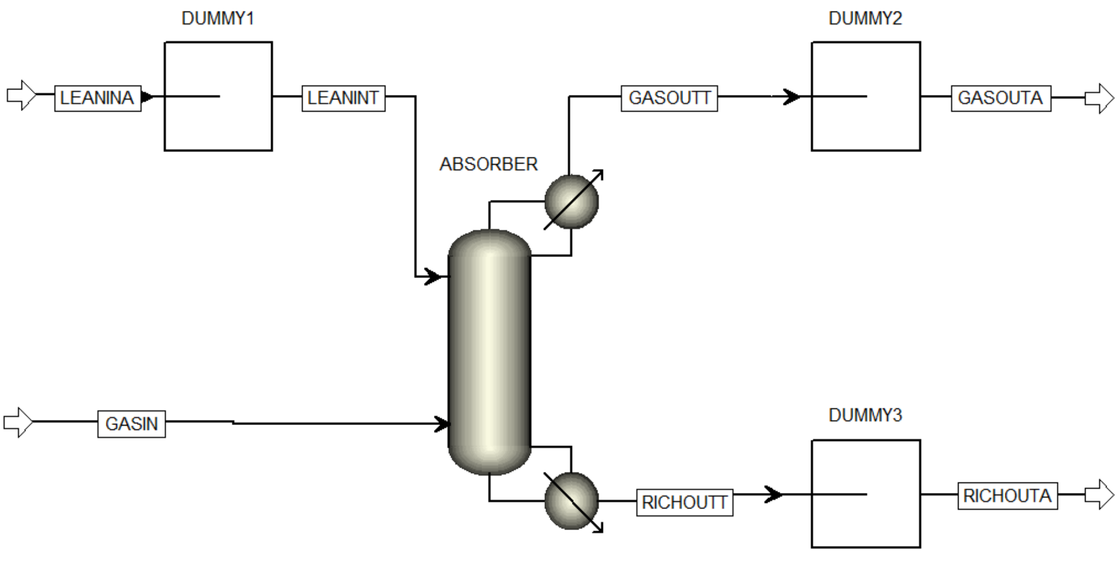


Figure 10. Flowsheet of the amine absorber simulation for training data generation.

The column properties mostly follow the properties set in AspenTech's example model. However, column internals and some calculation options are varied for the purposes of this thesis. Both trays and packings are used in absorber columns, as discussed by e.g., Mudhasakul et al. (2013) and Stewart (2011). In this work, Intalox Metal Tower Packings (IMTP) was chosen as internals for the base model. Since IMTP packings were chosen as column internals instead of the default trays, the height of the column was adjusted to 7.5 meters. Mass transfer coefficient and interfacial area correlations were changed from the default method Onda-68 to Hanley-IMTP 2010 method specifically intended for solving the column model with IMTP packings. The design mode for calculating the column diameter was used instead of defining the column diameter for the model, so that stage flooding and drying up could be avoided in hybrid model validation cases. Table 3 presents the properties of the RadFrac column.

Table 3. Column properties and calculation methods in the rate-based column model.

Property	Value
Calculation type	Rate-Based
Theoretical stages	21
Condenser & reboiler	None & none
Feed streams	LEANIN On-Stage 1 GASIN On-Stage 21
Thermodynamic model	ELECNRTL for liquid properties RK for vapor properties
Simulation approach	True components
Column internals	IMTP packings in 1 section, rating mode
Column diameter	Design mode to calculate column diameter Base stage 21 Base flood 0.8
Section packed height	7.5 m
Liquid phase film resistance model	Discretize film (5 discretization points)
Vapor phase film resistance model	Consider film
Mass transfer coefficient method	Hanley-Im10
Heat transfer coefficient method	Chilton and Colburn
Interfacial area method	Hanley-Im10
Maximum number of iterations	200

In the base case of the simulation, the same incoming stream compositions and properties as well as the reaction chemistry as in AspenTech's example model (2020b) are used. The stream GASIN presents the incoming sour gas stream, and the stream LEANIN presents the incoming amine stream, as shown in the flowsheet in Figure 10. The stream details for the base simulations are shown in Table 4, including the true component and apparent component streams.

Table 4. Composition and properties of the incoming streams. In the product streams, T stands for True components and A stands for Apparent components.

Component/ Property	GASIN A	LEANIN A	LEANIN T	GASOUT T	GASOUT A	RICHOUT T	RICHOUT A
MDEA (mol/s)	0	17.75	17.75	2.45E-04	2.45E-04	9.00	17.75
H_2O (mol/s)	0.40	238.26	238.26	1.77	1.77	228.17	236.89
CO_2 (mol/s)	15.83	0.10	0.10	7.19	7.19	0.03	8.75
H_2S (mol/s)	2.56E-02	1.58E-03	1.58E-03	4.79E-04	4.79E-04	4.96E-04	2.67E-02
H_3O^+ (mol/s)	0	0	0	0	0	1.98E-08	0
OH^- (mol/s)	0	0	0	0	0	2.29E-04	0
HCO_3^- (mol/s)	0	0	0	0	0	8.71	0
CO_3^{-2} (mol/s)	0	0	0	0	0	3.76E-03	0
HS^- (mol/s)	0	0	0	0	0	2.62E-02	0
S^{-2} (mol/s)	0	0	0	0	0	3.42E-06	0
$MDEAH^+$ (mol/s)	0	0	0	0	0	8.75	0
CH_4 (mol/s)	440.03	0	0	439.83	439.83	0.20	0.20
T (°C)	32.22	43.89	43.89	54.31	54.31	48.52	48.52
p (bar)	55.16	55	55	55	55	55	55

The base model's chemistry includes a set of equilibrium reactions as the global chemistry (MDEA) to model the electrolyte solution chemistry. A set of kinetic and equilibrium reactions are included as the absorber's reaction chemistry (MDEA-REA), modelling the CO_2 removal with kinetic reactions and H_2S dissolution with equilibrium reactions.

6.2 Data generation with Aspen Plus Multi-Case

Aspen Plus Multi-Case is used to generate training data for the hybrid models. For FP driven and AI driven models, data is generated with Case Study tool designed for a what-if analysis of steady-state responses to changing input variables. For ROMs, the Reduced Order Model tool is used to generate data. The tools allow swift mapping of independent (input) and dependent (output) variables and using multiple laptop processor cores to run multiple simulations at the same time. (AspenTech, 2022a) Step-by-step workflow on the use of Aspen Plus Multi-Case can be found in Appendix I.

In data generation with the Case Study projects, the goal is to collect such data that contains the component specific molar flows of the incoming streams as independent variables, and component specific molar flows and temperatures of the product streams as dependent variables. The independent and dependent variables of the datasets are presented in Table 5. In the input streams, methane, CO_2 and H_2S flows are varied only in the input gas stream, although also the lean amine stream contains small amounts of CO_2 and H_2S in the base case. Possible changes in amine regeneration influencing the lean amine composition are not considered in data generation. Moreover, the MDEA and water flows are varied in the lean amine stream in such limits that obey general limits for MDEA concentration in the amine solution found in literature: in the base case, the lean amine stream consisted of ca. 30 % of MDEA by weight, and solution strengths in amine absorption normally range up to 50 % amine by weight (Stewart, 2011). Therefore, such limits are set to the water and MDEA ranges that the incoming amine solution strength varies from 30 to 50 wt.%. Dependent variable data is collected from the apparent component gas and rich amine product streams. By varying the incoming stream component specific flows with Aspen Plus Multi-Case, variance is created to the dependent variables. The data can then be used to train the properties of simpler unit operations in Aspen Plus Hybrid Modelling.

Table 5. Independent and dependent variables of the datasets.

Independent variables		Dependent variables	
$CH_{4\ GASIN}$	(mol/s)	$T_{GASOUTA}$	(K)
$CO_{2\ GASIN}$	(mol/s)	$T_{RICHOUTA}$	(K)
H_2S_{GASIN}	(mol/s)	$CH_{4\ GASOUTA}$	(mol/s)
$MDEA_{LEANINA}$	(mol/s)	$CO_{2\ GASOUTA}$	(mol/s)
$H_2O_{LEANINA}$	(mol/s)	$H_2S_{GASOUTA}$	(mol/s)
		$MDEA_{GASOUTA}$	(mol/s)
		$H_2O_{GASOUTA}$	(mol/s)
		$CH_{4\ RICHOUTA}$	(mol/s)
		$CO_{2\ RICHOUTA}$	(mol/s)
		$H_2S_{RICHOUTA}$	(mol/s)
		$MDEA_{RICHOUTA}$	(mol/s)
		$H_2O_{RICHOUTA}$	(mol/s)

Different datasets are generated for different purposes. First, the number of steps in the independent variable ranges is varied to study the effect of the size of the dataset on ML model training and performance. The effect of the dataset size is studied by generating datasets with similar wideness: gas component input flows are varied from 50 to 120 % of the base case flows, MDEA flow is varied from 90 to 120 % and water flow from 60 to 105 % of the base case flows. The studied ranges for water and MDEA in the lean amine stream are different from the studied ranges of CO_2, H_2S and methane, since the amine solution strength is limited to vary from 30 to 50 wt.%. The studied steps in the independent variable ranges are shown in Table 8.

Table 6. Studied numbers of steps in the independent variable ranges.

Dataset no.	**Studied no. of independent variable steps**		**Dataset size**
	$CO_{2\ GASIN}$, H_2S_{GASIN}, $MDEA_{LEANIN}$	$CH_{4\ GASIN}$, H_2O_{LEANIN}	
1	3	3	243
2	5	5	3125
3	7	5	8575
4	10	5	25 000
5	15	5	84 375

Moreover, the wideness of the generated data is varied to study how narrower training data affects model training and performance. The datasets with different wideness are created to all have the same size, which is chosen when studying the effect of dataset size. The studied dataset properties and their ranges are presented in Table 7.

Table 7. Studied independent variable ranges compared to the base case flows.

Dataset no.	**Studied independent variable ranges** (100% = base case flows)		
	$CO_{2\ GASIN}$, H_2S_{GASIN}, $CH_{4\ GASIN}$	$MDEA_{LEANIN}$	H_2O_{LEANIN}
4	50 – 120 %	90 – 120 %	60 – 105 %
6	70 – 115 %	92.5 – 115 %	70 – 104 %
7	80 – 110 %	95 – 110 %	80 – 103 %
8	90 – 105 %	97.5 – 105 %	90 – 102 %

Out of the previously described datasets, the one resulting in the best-performing FP driven model is also used in AI driven modelling, but the component flow data is converted to total flows and mole fractions for the Model Builder. For ROMs, training data is generated with a Multi-Case Reduced Order Model tool instead of the Case Study project. The ROM tool records data of input and output flow pressures, temperatures, total mass or mole flows, and compositions as mass or mole fractions (mole flows and fractions in this work). The total flows and fractions were chosen so that the training data covered similar component flows as in the other datasets. The ROM tool in Multi-Case uses a Sobol algorithm to generate a balanced dataset with data points within balanced distances from each other. The number of data points is chosen to be the same as in the training dataset that was used to train the best-performing FP driven model.

6.3 Aspen Plus First Principles Driven Hybrid Modelling

The purpose of the FP driven modelling is to tune simulations with plant data, so that they predict the process behavior more precisely even if the complete phenomenon behind the plant behavior may not be known (AspenTech, 2020a). The FP driven hybrid modelling tool creates a neural network with simulation input variables as outputs. As the NN is trained with plant data, it adjusts the model parameters so that the simulation results are tuned towards training data. However, training data used in this thesis is simulated data instead of plant data, as explained in Section 6.2. The intention of using simulated data is to tune simpler unit operation models with data generated with a more complex unit operation and thermodynamic model, with the target of numerical stability and shorter calculation times while maintaining similar accuracy. FP driven models are trained with different NN configurations to study the effect of NN properties to training and performance, and to find the optimal NN configuration for the type of data used. Additionally, FP driven models are trained with the different datasets generated to study the effect of dataset properties to model training and performance. Workflow for setting up a FP driven model can be found in Appendix II.

The flowsheet used in FP driven hybrid modelling consists of a separator unit, where the user sets component-wise split fractions between product streams. The initial values for the split fractions are estimated based on the base model’s results. The separator’s outlet streams enter heaters with initial guess temperatures as inputs, similarly estimated based on the base model. The separator + heaters flowsheet before NN deployment is shown in Figure 11. In the figure, inlet streams LEANIN and GASIN are the amine and sour gas streams into the separator unit. GASOUT and RICHOUT are the cleaned gas and rich amine streams out of the separator with the correct component fractions. HOTGAS and HOTRICH are the product streams in the corrected temperature.

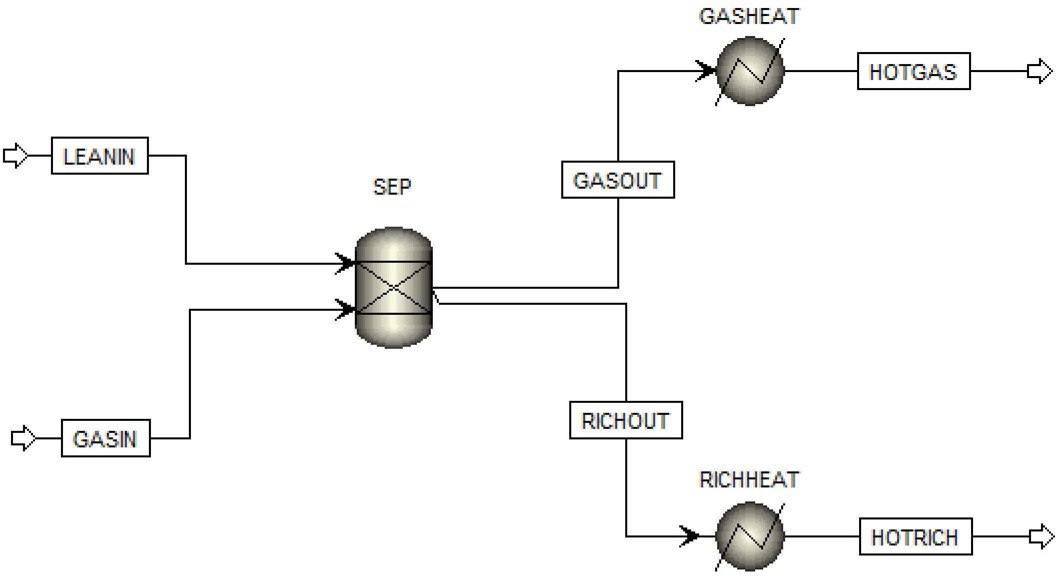


Figure 11. Simulation flowsheet for the FP driven model before NN deployment.

Simulated data is exported from Aspen Plus Multi-Case tool, and unconverged runs are filtered out of the data to avoid outliers. The data is converted for the FP driven modelling tool with an .xlsm -template provided by Aspen Plus, and the data is imported to Aspen Plus. Raw data is inspected visually for outliers. Since the data in this case is simulated and unconverged runs are removed from the data before importing, there are no clear outliers, and a data conditioning run is done without specifying limits for the variables.

The Training mode in the Build Model page is used to build and train the NN. The NN outputs in the hybrid model are the split fractions of the separator as well as temperatures set in the heaters. The model parameters trained as NN outputs are presented in Table 10. Model parameters mapped as independent and dependent variables are shown in Table 11.

Table 8. Model parameters mapped as NN outputs.

Simulation variable mapped as NN output	Range	Initial value
Split fraction of CH_4 into GASOUT stream	0-1	0.9
Split fraction of CO_2 into GASOUT stream	0-1	0.9
Split fraction of H_2O into GASOUT stream	0-1	0.05
Split fraction of H_2S into GASOUT stream	0-1	0.1
Split fraction of MDEA into GASOUT stream	0-1	0.1
Set temperature of the gas product heater	30 – 80 °C	44 °C
Set temperature of the rich amine heater	20 – 70 °C	35 °C

Table 9. Mapped training data and model parameters. I/D in the table present whether the variable is mapped as independent (I) or dependent (D) variable. The product streams are referred to as HOTGAS and HOTRICH similarly as in Figure 11.

Variable in data	Model parameter	Unit	I/D
$CH_{4,GASIN}$ to column	$CH_{4,GASIN}$ to splitting unit	mol/s	I
$CO_{2,GASIN}$ to column	$CO_{2,GASIN}$ to splitting unit	mol/s	I
H_2S_{GASIN} to column	H_2S_{GASIN} to splitting unit	mol/s	I
H_2O_{LEANIN} to column	H_2O_{LEANIN} to splitting unit	mol/s	I
$MDEA_{LEANIN}$ to column	$MDEA_{LEANIN}$ to splitting unit	mol/s	I
$T_{GASOUTA}$	T_{HOTGAS}	K	D
$T_{RICHOUTA}$	$T_{HOTRICH}$	K	D
$CH_{4,GASOUTA}$	$CH_{4,HOTGAS}$	mol/s	D
$CO_{2,GASOUTA}$	$CO_{2,HOTGAS}$	mol/s	D
$H_2S_{GASOUTA}$	H_2S_{HOTGAS}	mol/s	D
$H_2O_{GASOUTA}$	H_2O_{HOTGAS}	mol/s	D
$MDEA_{GASOUTA}$	$MDEA_{HOTGAS}$	mol/s	D
$CH_{4,RICHOUTA}$	$CH_{4,HOTRICH}$	mol/s	D
$CO_{2,RICHOUTA}$	$CO_{2,HOTRICH}$	mol/s	D
$H_2S_{RICHOUTA}$	$H_2S_{HOTRICH}$	mol/s	D
$H_2O_{RICHOUTA}$	$H_2O_{HOTRICH}$	mol/s	D
$MDEA_{RICHOUTA}$	$MDEA_{HOTRICH}$	mol/s	D

Different NN configurations are studied to evaluate the effect of NN properties to model training and performance. There seems to be no peer-reviewed numerical rules to determine the suitable NN configuration. Some internet discussions and blogs claim that the number of hidden neurons should not exceed the number of input layer neurons, or the combined number of input and output layer neurons, etc. (StackExchange.com, 2011, Krishnan, 2021). According to some discussions, one hidden layer is sufficient for nearly all problems, whereas others are sure that increasing the number of hidden layers will enhance the model trainability. Furthermore, it is often stated that there is no shortcut to finding the optimal NN configuration, but rather it should be determined with a systematic iteration process, which is done in this case as well.

In this work, the studied NN properties are learning rate (LR), hidden layer activation function, and the number of hidden layers and neurons in the NN. Default settings for loss function (Huber-Delta) and cost function minimization method (stochastic gradient descent with momentum) are used. In stochastic gradient descent (SGD) method with momentum, the NN weight parameters ω are updated with the slope of cost function J with respect to ω_j, multiplied with learning rate α. Slope of the cost function is calculated based on only one stochastically chosen data point from the dataset. The momentum considers the previous update of ω_j multiplied with momentum parameter γ (value set to 0.9 by default). Equation 6.1 displays the update of ω_j for an iteration, Equation 6.2 shows how the update is added to the weight parameter, and Equation 6.3 shows the two equations combined. In Aspen Plus, the iteration with weight parameters is continued until convergence, i.e., until the cost function value does not get smaller than 0.99-fold during 10 training epochs (default settings that were kept constant). SGD with momentum method was sourced from Rebala et al. (2019), since Aspen Plus documentation did not include information on the method.

$$\Delta\omega_j = \alpha \frac{\delta J}{\delta \omega_j} + \gamma \Delta\omega_{j-1} \qquad 6.1$$

$$\omega_j = \omega_j - \Delta\omega_j \qquad 6.2$$

$$\omega_j = \omega_j - \alpha \frac{\delta J}{\delta \omega_j} - \gamma \Delta \omega_{j-1} \tag{6.3}$$

where ω_j is the weight parameter at iteration j, $\frac{\delta J}{\delta \omega_j}$ is the slope of cost function J. Calculation of J is shown in Equation 6.4.

$$J = \frac{1}{N} \sum_{j=1}^{N} L_\delta(Y_{j,actual}, Y_{j,predicted}) \tag{6.4}$$

where N is the number of data points in a training epoch, and $L_\delta(Y_{j,actual}, Y_{j,predicted})$ is the loss for one datapoint according to the Huber-Delta loss function shown in Equation 6.5. The Huber-Delta loss function calculates the loss as mean squared error (MSE) if the difference between actual and predicted outputs is smaller than the delta parameter δ (value of δ is set to 1 as is by default in Aspen Plus), and otherwise as mean absolute error (MAE).

$$L_\delta(y_{j,actual}, y_{j,predicted}) = \begin{cases} \frac{1}{2}\left(Y_{j,actual} - Y_{j,predicted}\right)^2 \text{ for } \left|Y_{j,actual} - Y_{j,predicted}\right| \leq \delta \\ \delta\left|Y_{j,actual} - Y_{j,predicted}\right| - \frac{1}{2}\delta^2 \text{ otherwise} \end{cases} \tag{6.5}$$

Optimal NN configuration is iterated with a mid-sized dataset with ca. 8500 data points. First, the optimal learning rate (LR) is determined with a NN consisting of one hidden layer with 12 neurons. The hidden layer activation function is SeLu, while the output layer activation function is Sigmoid, which are both defaults. After finding the optimal LR, the optimal number of hidden layers and neurons as well as hidden layer activation function are determined for the 8500-row dataset. The configuration is confirmed with smaller and larger datasets of ca. 3000 and 25 000 data rows. When optimizing the NN configuration, the used datasets are such that the input stream gas component flows vary from 50 to 120 % of the base case flows. The studied and constant NN properties are shown in Table 12.

Table 10. Studied neural network configurations.

Neural network property	Studied range
Learning rate	0.00005 – 0.001
Hidden layers	1 – 3
Neurons in hidden layer(s)	8 – 18
Activation function	SeLu ReLu Leaky ReLu Sigmoid Linear
Fit type (loss function, minimized during training)	Huber-Delta ($\delta = 1$)
Loss function minimization method	Stochastic gradient descent with momentum
Momentum parameter	0.9
Output layer activation function	Sigmoid
Training set/testing set ratio	0.8
Finish training if error does not improve by X during Y iterations	X = 0.99 Y = 10
No. of training epochs	100

Once a satisfactory NN configuration is found, it is deployed to the simulation flowsheet. Figure 12 shows the simulation flowsheet after NN deployment. In the flowsheet, TO stands for training object, which is the NN receiving the feed stream component flows as inputs and giving the separator unit's split fractions and product heater temperatures as outputs. The black lines describe streams, and the blue lines describe simulation variables used or predicted by the NN.

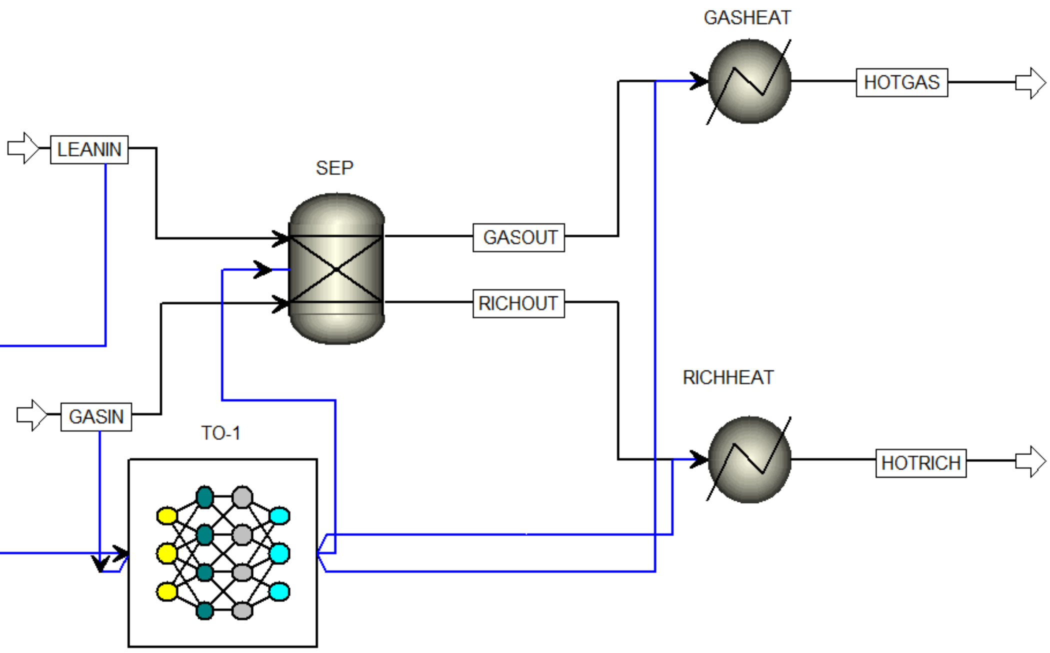


Figure 12. Simulation flowsheet after deploying the neural network.

6.4 Aspen Plus AI Driven Hybrid Modelling

The AI driven modelling tool in the Aspen AI Model Builder is intended for situations where there is a large amount of plant data, but no simulation to begin with, and the goal is to create a simulation swiftly (AspenTech, 2020a). In this work however, simulated data was used to train the AI driven model so that the model could be compared to the FP driven model. Out of the base model parameters, the apparent component stream temperatures, pressures, total mole flows, and component mole fractions are mapped as model parameters for the input and output flows created for the AI driven model.

In the AI driven modelling tool, there are three different linear methods as options for the machine learning method. Several options are available for transformations, three of which are studied in this work. Component balances and mass balances are chosen as constraints for the model. The properties of the AI driven model are shown in Table 13.

Table 11. Studied AI driven model properties.

AI driven model property	Studied range
Machine learning method	Lasso CV Lasso CV, Auto Reluctant Approach Lasso Lasso, Auto Reluctant Approach
Base transformations	None 2nd order 3rd order

After the AI driven model is created, the user can observe the component-wise or average coefficients of determination (R^2s) reported by Aspen Plus (calculation principle shown in Equation 2.3) and export the hybrid model if satisfactory results are obtained. The model is exported as a .ATHM file, which can be imported into an Aspen Plus simulation as a hybrid model and inserted into a flowsheet, as shown in Figure 13. The model cannot be edited in Aspen Plus, and thus re-training and importing a new version of the model is required if changes are needed.

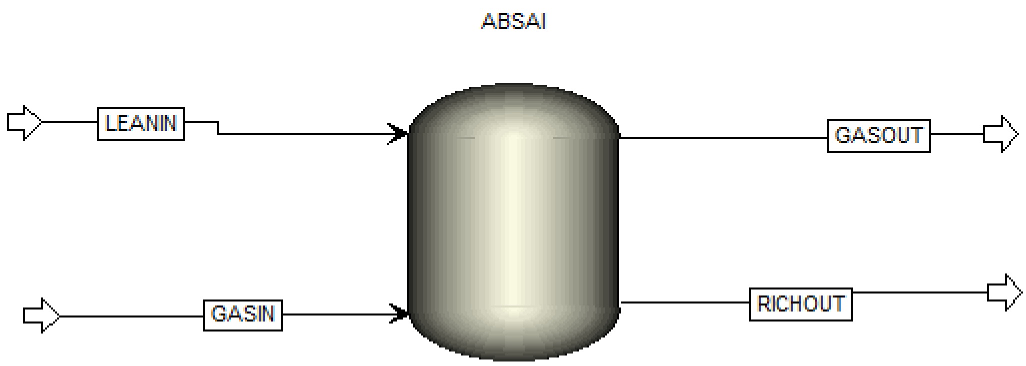


Figure 13. The AI driven model imported to a simulation flowsheet, with corresponding streams linked into it.

6.5 Aspen Plus Reduced Order Modelling

In the ROM tool in the Aspen AI Model Builder, training data generated with a Multi-Case Reduced Order Project is imported to AIMB, and the tool automatically maps the model parameters in the data to the model parameters of the ROM. No data cleaning is needed for the reduced order data, since unconverged cases are again filtered out of the data before importing. When building the model, options for ML method include the linear Lasso CV, Reluctant, and Auto-Reluctant approaches in addition to the constrained neural network, but out of interest to compare the ROM results to FP driven model results, only the constrained neural network is used in this work. No options or information on the cost function minimization method is given in the AIMB or in the Model Builder Help page. Interestingly, the options for hidden layer activation function are slightly different in the ROM tool than in FP driven modelling. The studied reduced order model properties are shown in Table 12.

Table 12. Studied reduced order model properties.

Reduced order model property	**Studied range**
Machine learning method	Constrained Neural Network
Hidden layer activation function	Tanh (hyperbolic tangent activation function) Swish (Sigmoid-weighted Linear unit activation function) Sigmoid Linear
Hidden layers	2-3
Neurons per layer	16-18

When a suitable configuration for the model is found, it is exported as a .ATHM file and can be imported into a simulation flowsheet in a similar manner as the AI driven model. Also the ROMs can only be edited and trained in AI Model Builder. If changes are made to the model, a new version of the model is imported to Aspen Plus.

6.6 Model assessment methods

When iterating the optimal NN configuration, the overall validation error of the FP driven model at the end of training is monitored. Validation error is calculated and reported by Aspen Plus. Additionally, the trainability of different dependent variables is studied by monitoring component-wise coefficient of determination (R^2) in each product stream. The values for R^2 are calculated by Aspen Plus, however Equation 2.3 shows the calculation principle. Component-wise and average R^2s are also calculated by AIMB when creating AI driven models and ROMs.

When studying the effects of dataset size and narrowness on hybrid model performance, the hybrid models are validated against the base model by running a specified set of test cases with Aspen Simulation Workbook (ASW). The same test case set is used in validating all FP driven hybrid models. Additionally, the best-performing FP driven model as well as the ROM and AI driven model are validated with a test set using wider extrapolation. In the test case set, input molar flows are varied one by one while keeping other flows at base case values, so that they extrapolate the training data inputs even in the widest datasets. When one input flow is varied, the others are kept in the base case value. The input molar flow ranges of the test set are shown in Table 13, as percentages compared to the base case molar flows.

Table 13. Varied input component flows in the test set. 100% = base case molar flows.

Input variable	Studied range on all models	Extended extrapolation range
$CO_{2\ GASIN}$ (mol/s) H_2S_{GASIN} (mol/s) $CH_{4\ GASIN}$ (mol/s)	45-150 %	30-250 %
$MDEA_{LEANIN}$ (mol/s)	85-120 %	30-250 %
H_2O_{LEANIN} (mol/s)	55-120%	30-250 %

Hybrid model and ELECNRTL model results are then compared graphically and by calculating the mean absolute error (MAE) and mean absolute percentage error

(MAPE) of the output component flows and temperatures as well as a composite error (E_C) of the flows and temperatures over the whole test case set. Calculation of MAE, MAPE and E_C are shown in equations 6.6 through 6.8.

$$MAE = \frac{1}{k}\sum_{i=1}^{k} \left|Y_{predicted} - Y_{actual}\right| \qquad 6.6$$

$$MAPE = \frac{1}{k}\sum_{i=1}^{k} \frac{\left|Y_{predicted} - Y_{actual}\right|}{Y_{actual}} \cdot 100\ \% \qquad 6.7$$

$$E_C = \sum_{i=1}^{k} \left|Y_{predicted} - Y_{base\ model}\right| \qquad 6.8$$

where k is the number of test cases, $Y_{predicted}$ is the component flow or stream temperature predicted by the hybrid model, and Y_{actual} is the component flow or stream temperature calculated by the base model with ELECNRTL thermodynamic package and the rigorous absorber column model.

While the hybrid models may offer advantage in shorter calculation times, the data generation and model training can be time-consuming. Thus, when generating training data and training the hybrid models, the time consumed in data generation and training are monitored. The time consumption and the possible advantages or disadvantages brought by the hybrid models are discussed.

7 Results and discussion

This section presents the results for the studies conducted with the different training datasets and machine learning tools provided by AspenTech presented in Section 6. The aim of the applied part was to study the effect of different NN properties as well as training dataset size and wideness on model training and performance. First, results from searching for the optimal settings and configurations for the hybrid models created with FP driven modelling, AI driven modelling and ROM tools are presented in sections 7.1 through 7.3. Trainability of the different dependent variables is discussed in Section 7.4. Validation results for the FP driven models with different training datasets are presented in Section 7.5. The effects of the absorber column diameter (calculated with the design mode in Aspen Plus) on the absorption were not drastic, and results for the diameter can be found in Appendix V. It was also of interest to study the extrapolation abilities of the different models, and thus the best-performing FP driven model, AI driven model and ROM were validated with a wider set of test cases. Results for the extrapolation are shown in Section 7.6. Time consumption on training data generation and hybrid model training, as well as comparison of computational times between the base model and the hybrid models are discussed in sections 7.7 and 7.8, respectively. The usability of the different hybrid models regarding deployment in larger simulation flowsheets, the required ML knowledge for effective hybrid modelling as well as other aspects are discussed in Section 7.9.

7.1 Optimal neural network configuration for the FP driven models

An optimal NN configuration was first determined for a mid-sized dataset (no. 3 in Table 8). In the dataset, incoming gaseous component flows were varied one by one from 50 to 120 % of the base case flows, and incoming water and MDEA flows were varied one by one so that the lean amine stream strength was between 30 and 50 % by weight. The optimal learning rate (LR) was first determined, followed by determining the optimal number of hidden layers and neurons using three datasets: first with the dataset with ca. 8500 rows, and then with smaller and larger datasets with ca. 3000 and 25 000 rows.

The effect of LR on the model validation error (VE) and component-wise R^2 was evaluated with a NN with one hidden layer consisting of 12 neurons, and a NN with two hidden layers consisting of 13 neurons each. The optimal LR for the data in question was 0.00005 – 0.0001, while larger LRs resulted in large validation errors. Optimal LR seems to depend on the NN structure, but the lowest validation error resulted from a LR of 0.0001 with the two-layer NN. Therefore, LR of 0.0001 was used in the upcoming tests for NN configuration. Figure 14 shows the validation error reported by Aspen Plus at the end of training for the tested learning rates and NNs.

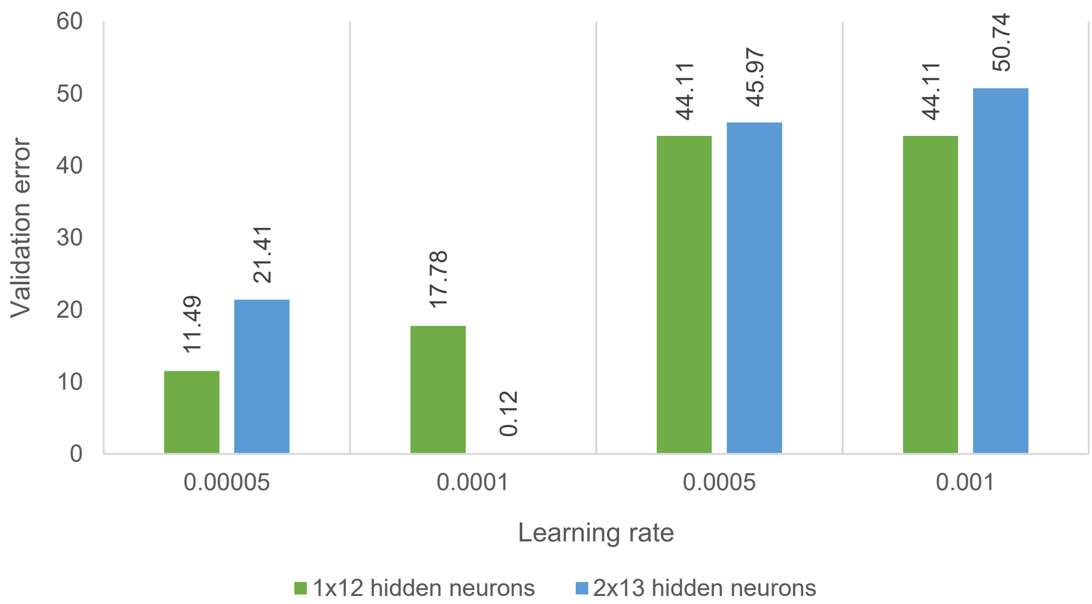


Figure 14. Validation error at the end of hybrid model training with learning rates 0.00005 - 0.001.

Next, the number of neurons and hidden layers was determined with different activation functions. The number of connections between neurons (i.e., weight parameters to be optimized) in the NN were monitored during the iterations. Table 14 presents VE reported by Aspen Plus after hybrid model training for different NN configurations and activation functions. As can be seen, increasing the number of layers and neurons did not directly enhance model trainability. On the contrary, the configuration with 2 hidden layers and 14 neurons per layer showed a sudden peak in VE with SeLu, ReLu and

Leaky ReLu. It is possible that by adjusting the advanced NN settings, better results could have been obtained with the configurations with peaks in VE. However, the testing scope was not extended further. Training models with Sigmoid activation function resulted in rather steady VE with different NN configurations, and no well-trained NN configuration with VE < 1 was found. VE was even higher with Linear activation function. Although it is possible that suitable NN configurations for Sigmoid and Linear activation functions could be found outside the range of configurations tested here, the training results obtained with SeLu, ReLu and Leaky ReLu were satisfactory. Therefore, the testing was limited to the displayed range. The best model training results were obtained with ReLu and Leaky Relu, with NNs containing 2 or 3 hidden layers with 18 neurons each.

Table 14. Validation errors reported by Aspen Plus at the end of hybrid model training for different NN configurations and activation functions.

Layers	**Neurons per layer**	**No. of para-meters**	**Validation error with activation function**				
			SeLu	**ReLu**	**Leaky ReLu**	**Sigmoid**	**Linear**
1	8	192	9.78	9.77	9.76	5.09	16.23
2	8	256	1.37	0.23	0.20	6.54	43.44
2	10	340	0.26	0.12	0.097	6.54	42.35
2	12	432	0.22	0.07	0.092	6.45	70.30
2	14	532	38.39	40.65	40.65	6.54	40.96
2	16	640	0.21	0.069	0.054	6.55	44.61
2	18	756	31.08	0.044	0.041	6.55	42.35
3	14	728	47.00	40.96	44.61	6.55	197 000
3	16	896	Failed	44.61	44.61	6.55	Failed
3	18	1080	Failed	0.037	0.044	6.55	Failed

Some configurations seemingly failed to converge. There, Aspen Plus did not give a message of failed convergence, but only one or a few training epochs were completed, and the hybrid model displayed no results for validation error or component-wise R^2s. When NN training fails to converge, no such point can be found on the learning curve that can be represented as the starting point of getting smaller error. According to some discussions, possible reasons for failed training include too small number of nodes compared to the amount of data. In this case, however, it was usually an increase in the number of nodes/model complexity that caused the training not to converge. Moreover, higher complexity with a certain activation function may also lead to training failure, even though simpler models converged. Using different activation function with more complex models may help with the convergence. (Verma, 2022)

Next, the NN configuration was studied with datasets with ca. 3000 and 25 000 data rows, with ReLu as the hidden layer activation function. Figure 15 shows the validation error at the end of model training with different numbers of parameters in the NN, obtained with the three hybrid models.

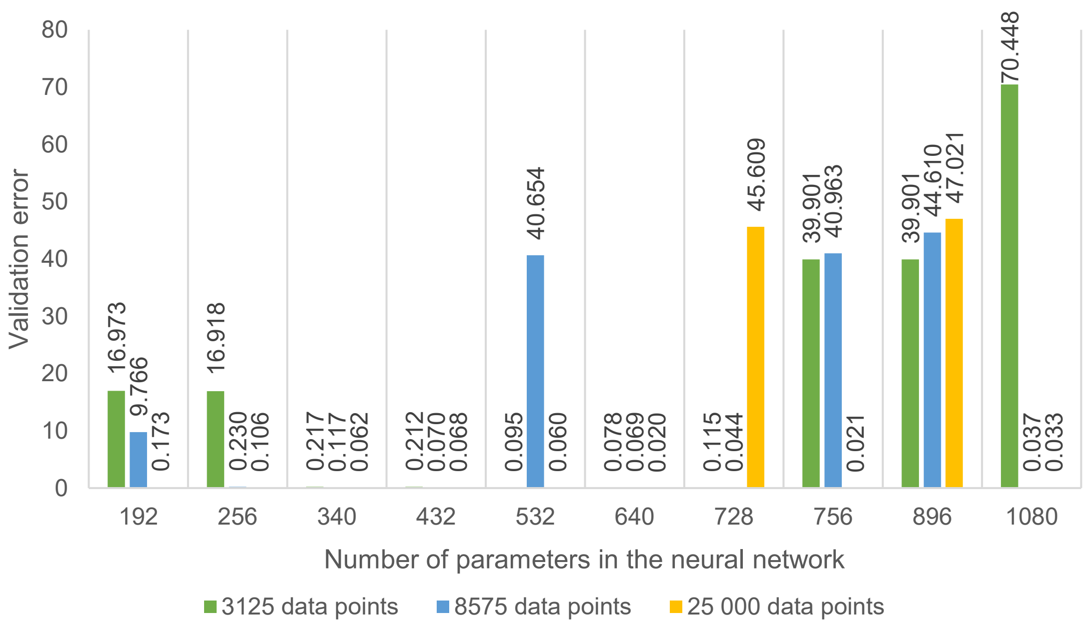


Figure 15. Validation error for obtained with different neural network configurations by hybrid models with different-sized datasets.

From the figure, it seems that simpler models were generally able to train models more accurately. On the other hand, the most complex NN from this testing range resulted in very small validation errors with the two larger training datasets. This observation leaves room for even wider studying of different NN configurations in future works, both within simpler configurations and more complex ones. In this work, however, model complexity was not increased any further, since the few tests done with larger number of hidden layers did not converge.

While the testing did not result in a numerical rule to determine the suitable number of hidden layers and neurons, a set of good configurations were found. The smallest validation errors were obtained with the configurations with two or three hidden layers and 16 or 18 neurons per hidden layer (640, 728, 756, or 1080 model parameters). These configurations were used in the upcoming tests regarding the effect of dataset size and data narrowness, depending on which configuration gave the lowest validation error at the end of model training. It was also briefly tested whether the number of layers and neurons could be changed while conserving the number of parameters. However, similar results were not always obtained with different configurations conserving the parameters, and a definitive numerical rule could not be formed. The NN configurations used for the FP driven models with different training datasets further in this work are shown in Table 15.

Table 15. NN configurations used for FP driven hybrid models with different training datasets.

Data set no.	Data set size	Hidden layers	Neurons per hidden layer	Parameters
1	243	2	18	756
2	3125	2	16	640
3	8500	3	18	1080
4	25 000	2	16	640
5	84 000	2	18	756
6	25 000	2	18	756
7	25 000	2	18	756

7.2 AI driven model configuration

Training results for the different AI driven model configurations are shown in Table 16. Including base transformations, especially 3rd order transformations, enhanced the training accuracy. This leaves room for speculation whether the additional transformations could have resulted in an even better model. However, the results were deemed satisfactory for this work, and the testing scope was limited to the base transformations. For the more accurate models, the Auto-Reluctant approach weakened the accuracy of the trained variables.

Table 16. Tested AI driven model configurations and their average R^2s reported by AspenTech AI Model Builder.

Model no.	Machine learning method	Base transformations	Average R^2
1	Lasso CV, Auto-Reluctant	None	0.946
2	Lasso CV	None	0.946
3	Lasso CV, Auto-Reluctant	2nd order	0.994
4	Lasso CV	2nd order	0.995
5	Lasso CV, Auto-Reluctant	3rd order	0.997
6	Lasso CV	3rd order	0.998

The highest accuracy was obtained with model 6 trained with the Non-Auto-Reluctant Lasso CV method and 3rd order transformations. This model was exported from the AI Model Builder and deployed into its own flowsheet. Validation results for the AI driven model are shown in Section 7.6 together with the validation results for the best-performing FP driven and reduced order models.

7.3 Reduced order model configuration

The ROM training results for the different model configurations are shown in Table 17. Out of the hidden layer activation functions available, the Sigmoid weighted linear activation function (Swish) gave the best training results. For the reduced order dataset with ca. 25 000 data points, the best training results were obtained with two hidden layers, each containing 18 neurons.

Table 17. Tested reduced order model configurations and their average R^2s reported by AspenTech AI Model Builder.

Model no.	Hidden layer activation function	Hidden layers	Neurons per hidden layer	Average R^2
1	Tanh	2	16	0.974
2	Swish	2	16	0.979
3	Sigmoid	2	16	0.963
4	Linear	2	16	0.947
5	Swish	2	18	0.991
6	Swish	3	16	0.969
7	Swish	3	18	0.985

Model 5 was exported from the AI Model Builder and deployed into its own flowsheet. Validation results for the reduced order model are shown in Section 7.6 together with the validation results of the best-performing FP driven and AI driven models.

7.4 Trainability of the dependent variables

To demonstrate which dependent variables were more difficult or easy to fit accurately with the hybrid models, Table 18 shows component-wise R^2s reported by Aspen Plus after FP driven model training. It seems that larger component flows were easier to train, since the accuracy of methane in the gas product as well as water, H_2S and MDEA in the rich amine product was high even in models with high validation error. Very small component flows, such as water, H_2S and MDEA in the gas product as well as methane and CO_2 in the rich amine product were more difficult to train. In this work, the objective function scaling was chosen in Aspen Plus (as is by default), meaning that the error of each dependent variable was scaled with the standard deviation when the loss function was computed. However, no options for scaling the data are given in the Aspen Plus tools. If data scaling is not performed, smaller component flows are less accurately trained, because the relative error in a very small stream quickly becomes very large.

The negative R^2s indicate that the hybrid model training fails completely to follow the trend of the training data, and rather iterates in the wrong direction. Stream temperatures seem slightly easier to train than very small component flows, but not as straightforward as the large component flows. This could be explained by the fact that the large component flows such as methane in the gas product and MDEA in the rich amine product are affected only by the incoming methane and MDEA flow, respectively. Therefore, the training data is clustered in clear, linear points, which are easy for the hybrid model to fit, whereas there is more variance in the temperature data, which is affected by different streams and the reactions in the process.

Another possible reason behind the differing trainability of the dependent variables is that the unconverged cases were filtered out of the training data, which could in theory lead to some imbalance in the data and therefore poorer trainability for some variables. When the raw data was inspected by eye, however, the unconverged cases seemed random, individual unconverged cases rather than large sets of cases due to one particular variable variation. Nevertheless, when generating data and training models it seems important to pay attention to what kind of cases in the training data do not converge and evaluate if it could result in imbalanced data.

When variables could not be trained accurately, the resulting R^2 of a variable was often the same with many or all of poorly trained models. Therefore, it seems that the training proceeds in a similar manner every time. Table 18 displays only results obtained with ReLu activation function, because similar conclusions could be drawn of the other activation functions as well, although the obtained training results were not exactly the same.

Table 18. Component-wise R^2s for neural networks with different configurations (activation function ReLu) trained with dataset 3 (ca. 8000 data points).

Parameters	192	256	340	432	532	640	728	756	896	1080
Validation error	9.77	0.23	0.12	0.07	40.65	0.069	40.96	0.04	44.61	0.04
Gas prod.										
CH_4	1	1	1	1	1	1	1	1	1	1
CO_2	0.985	0.979	0.991	0.996	-17.6	0.995	-10.4	0.997	-17.6	0.997
H_2O	0.928	0.981	0.99	0.994	-7.33	0.994	-7.33	0.996	-7.33	0.996
H_2S	0.951	0.984	0.988	0.989	-0.46	0.992	-0.46	0.991	-0.46	0.994
MDEA	0.893	0.981	0.989	0.993	-2.13	0.993	-2.13	0.996	-2.13	0.996
T	0.889	0.947	0.98	0.989	-26.5	0.991	-26.5	0.995	-26.5	0.994
Rich amine										
CH_4	-25.8	0.977	0.985	0.992	-25.9	0.992	-25.9	0.996	-25.9	0.996
CO_2	0.988	0.982	0.993	0.997	-14.6	0.995	-8.55	0.997	-14.6	0.998
H_2O	1	1	1	1	0.998	1	0.998	1	0.998	1
H_2S	0.994	0.998	0.999	0.999	0.827	0.999	0.827	0.999	0.927	0.999
MDEA	1	1	1	1	1	1	1	1	1	1
T	0.946	0.97	0.976	0.989	0.977	0.987	-6.09	0.993	-6.09	0.993

All product component flows were used as dependent variables in hybrid model training, even though it would be sufficient to train each component in only one of the two product streams. In this case, it would be important to train the smaller component flows to ensure their accurate training, since the relative error in the smaller flows may be large even if the large component flows were trained accurately with small relative errors. This approach was briefly tested, and similar accuracies were observed for the smaller component flows as shown above. However, training times remained the same (more detailed discussion on training times can be found in Section 7.7). Thus, the approach did not offer advantages compared to training all product component flows, and all upcoming tests were conducted using all product component flows as dependent variables so that the obtained results would be in similar form.

7.5 Effect of dataset size and wideness on FP driven model performance

When studying the effect of dataset size on FP driven model performance, datasets with gas component input flows varying from 50 to 120 % of the base case flows were used as model training data. The effect of dataset wideness was studied with 25000-datapoint datasets with gas component input flows of 90-105 %, 80-110 %, 70-115 % and 50-120 % of the gas component base case flows. For the narrowest dataset, a suitable NN configuration was not found despite testing multiple different configurations. While a suitable configuration may have been obtainable with even wider testing range of hidden layers and neurons, the testing was scoped to the range shown in Table 12. Hybrid models were validated in Aspen Simulation Workbook (ASW) with the set of test cases introduced in Section 6.6. To numerically assess the model performances, mean absolute errors (MAEs) and mean absolute percentage errors (MAPEs) were calculated for the product components and stream temperatures. Validation results are summarized in Section 7.5.1. Detailed results and discussion on component flow and stream temperature predictions as functions of the different component input flows are presented in sections 7.5.2 and 7.5.3, respectively.

As discussed in Section 7.4, some of the component flows in the product were easier than others to predict for all models. For example, methane, water, and MDEA flows were rather well predicted by almost all models even when slightly extrapolated. Additionally, varying H_2S input flow had little effect on the other component flows due to its small flow in the base case, and the results were predicted well. More deviation from the base model was seen on H_2S and CO_2 output flows, especially when methane and water input flows were varied. Sections 7.5.2 and 7.5.3 display figures for H_2S, CO_2 and temperatures as functions of methane and water input flows. Additionally, the effect of CO_2 or MDEA input flow variation is shown for some components. The MDEA input flow together with water input flow are key factors that can be adjusted in absorber operation. Also, varying the CO_2 input flow affected the column performance significantly. Other factors to adjust in absorption process operation would be the input stream temperatures and pressures, however they were left out of the scope of this work. The figures for the easily predicted component flows can be found in Appendix VI.

7.5.1 Results summary

MAEs and MAPEs were calculated for all dependent variables with Equation 6.6 and 6.7. MAEs for the component flows and temperatures (component flow absolute errors are opposite numbers for sweet gas and rich amine) of the models with different training datasets are shown in Figure 16, along with MAPEs for sweet gas component flows and temperatures. The information on the datasets on the x-axis is presented in the form *number of data points / input gas component variation range compared to base case*.

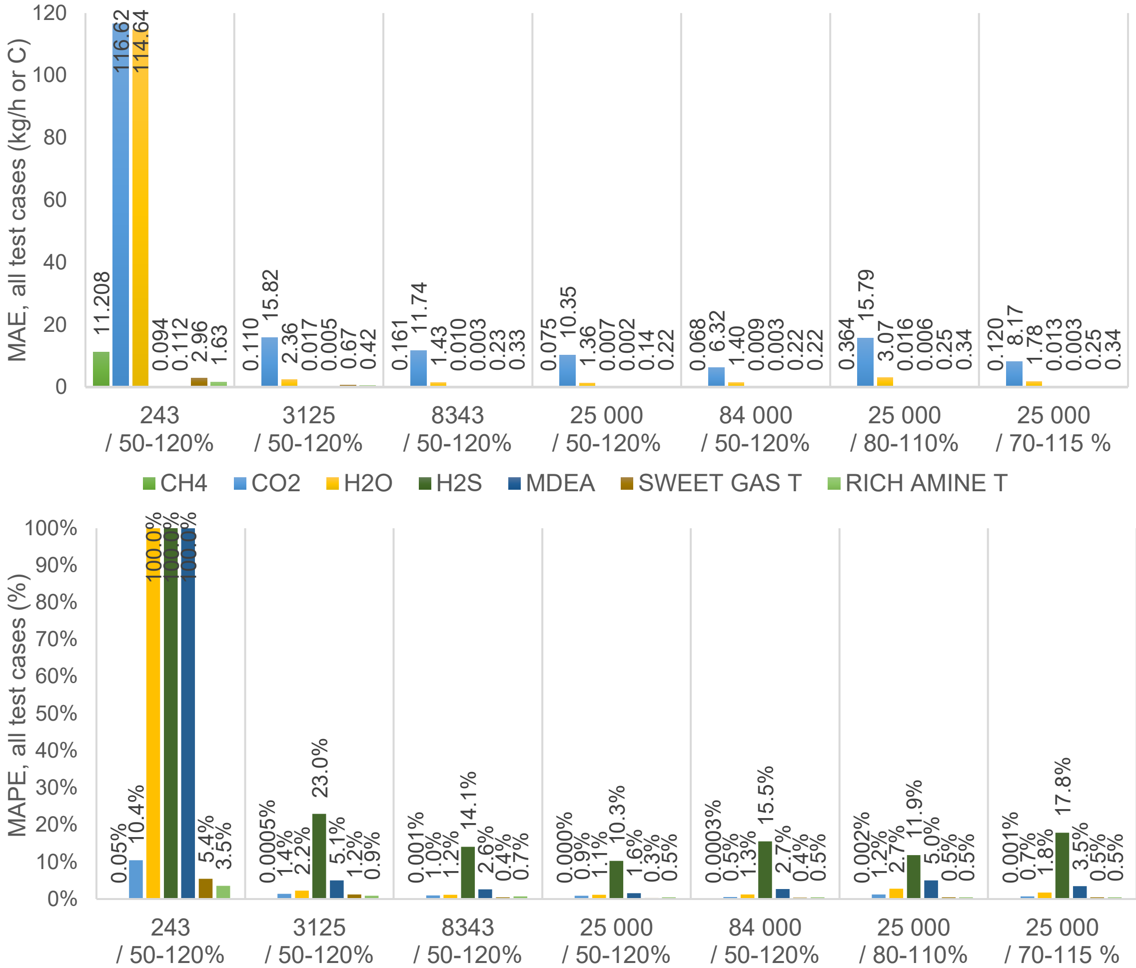


Figure 16. Mean absolute errors (MAEs) and (MAPEs) for component flow and temperature predictions of the different-sized datasets over all test cases. MAPEs are presented for gas component flows.

The figure shows that the models with the widest 25 000-point dataset and 84 000-point dataset predicted the component mass flows and stream temperatures with the smallest MAEs and MAPEs. The model trained with 25 000 data points was more accurate in predicting H_2S flows, whereas the model with the 84 000 -point dataset predicted CO_2 flows slightly more accurately. The error composite over the test case set was 1180 kg/h for 25 000 data point model and 780 kg/h for 84 000 model, and the difference can be mostly explained with the differences in CO_2 flow predictions. The error composites of all stream temperatures were 36.4 °C and 44.4 °C for the 25 000 and 84 000 data point models, respectively. Most MAEs and MAPEs were slightly higher if calculated for the extrapolation areas compared to the entire test case set, especially for the models with narrower training datasets. That is, all models predicted the dependent variables more accurately when modelling inputs were selected from inside the training data range. This matches the often-stated consensus that ML models do not extrapolate well, although some discussions argue that extrapolation of ML models is more common than usually assumed (Yousefzadeh and Cao, 2022). Nonetheless, the weakened accuracy in the extrapolated test cases suggests that including all possible use cases in the training data is a better option than relying on extrapolated models. This would be particularly important in designing or optimizing processes involving hazardous, polluting, or valuable compounds.

Since the MAEs for component flows in sweet gas and rich amine are equal, MAPEs for gas and liquid component flows are different depending on how large the component flows are in sweet gas and rich amine. Therefore, MAEs and MAPEs highlight different components with larger errors. For example, the MAE of CO_2 stands out as a large absolute error compared to the other component flows, but the mass flow is rather large and thus its MAPE stays small. The MAE of H_2S, on the other hand, is very small, but due to its small amount in the sweet gas, its MAPE stands out larger than the other components. Both MAE and MAPE of methane are small for most models despite methane's large flow, since methane does not react in the process and mostly exits the column in one outlet. Therefore, the training data and predicting the methane flow in the products is very linear. Water, whose mass flow is also large, reacts in the absorption process and is split between the outlet streams more evenly, resulting slightly less stable training data and larger MAEs and MAPEs. The model with the smallest dataset resulted in 100 % MAPEs for sweet gas H_2S, water, and MDEA, since all H_2S was predicted to be removed from the gas and no water or MDEA transfer into the gas (more details on the predictions in the next section). For the other models, the errors in sweet gas water and MDEA flows ranged from 1 to 5 %, which is understandable due to their flows being very small in sweet gas. Rich amine MDEA, on the other hand, was very accurately predicted by all models, as MDEA mostly exits the column in the rich amine stream.

7.5.2 Component flow predictions

In figures 17 through 19, sweet gas and rich amine H_2S mass flow predictions are shown as functions of methane, CO_2, and water, respectively. The figures show the input component flows as percentages of the base case flows. The legends indicate the wideness of training data for each input component flow, so that the reader gains insight on where the model was interpolated and extrapolated. For reference, the input component mass flows for the base case (100 % input flow in the figures) and their ranges in these validation cases are shown in Table 19.

Table 19. Component input mass flows in the base case (100 % in the figures).

Component	Sour gas CH_4	Sour gas CO_2	Sour gas H_2S	Lean amine H_2O	Lean amine MDEA
Base case (kg/h)	25 400	2510	3.1	15 450	7560
Validation min (kg/h)	11 440	1130	1.4	8520	6430
Validation max (kg/h)	38 000	3750	4.7	18 590	9080

The figures show that decreased methane input flow allowed for improved H_2S removal, since methane forms most of the sour gas flow and the amine always absorbs small amounts of it. Therefore, the relative amount of lean amine compared to sour gas increases and the liquid phase composition changes, leading to amine becoming in better contact with the H_2S and higher H_2S removal. Some improvement in H_2S removal can also be seen as the methane input flow increases. Nevertheless, the changes in

H_2S flow due to methane input flow were small. Also the absolute differences between the models while varying methane input flows were very small, although the model trained with ca. 3000 data points seemed slightly too optimistic. According to Stewart (2011), the H_2S content in natural gas sales must not exceed 4 ppm by volume (2 ppm for liquid natural gas (LNG) plants). In some of the cases, the model predicted sweet gas H_2S content to be below or close to the 4-ppm limit, although the base model's results already exceeded the limit. In those cases, the results from the models with smaller training datasets may not be usable. The accuracy of the model with the widest 25 000-data point training data, in turn, seems sufficient throughout the validation range and also in the higher extrapolation area.

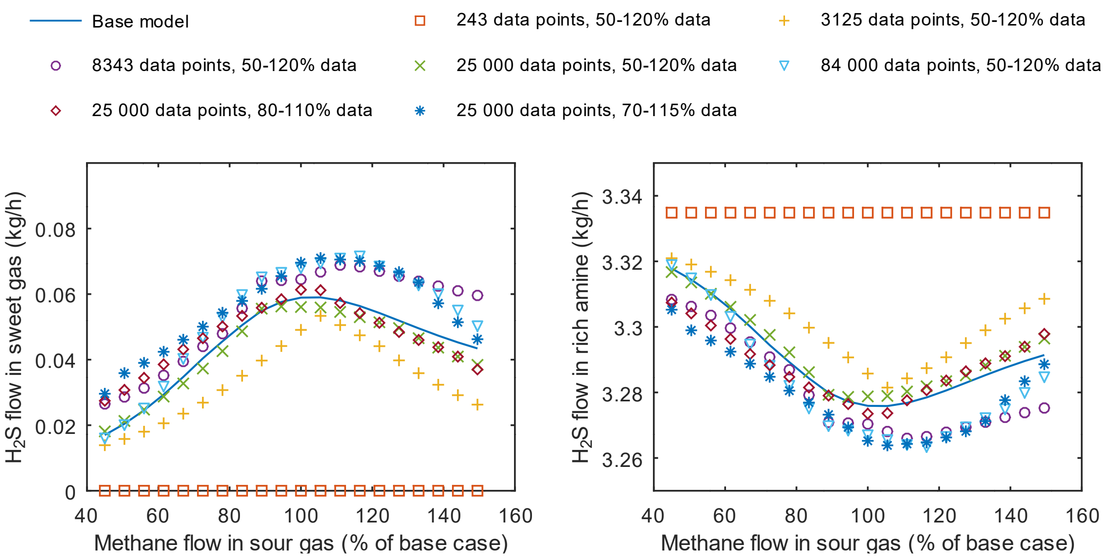


Figure 17. Sweet gas and rich amine H_2S flow predictions as functions of methane input flow.

When lean amine stream's flow and composition were varied, the water input flow affected the H_2S removal a lot more than the MDEA, since H_2S and its intermediate products only react with water in the model. Therefore, decreasing the water flow visibly weakened the H_2S removal, as less water was available to dissolve H_2S, and the reaction equilibrium shifted towards the gaseous reactants. As CO_2 reacts with both water and MDEA, increasing the CO_2 input flow again limited the water molecules available for reactions with H_2S, which decreased the H_2S removal capacity. Most FP driven models seemed to follow the base model's H_2S removal trends relatively well, apart from the model trained with the smallest dataset, which predicted all H_2S to be removed from the sour gas in all test cases. This indicates that the model trained with the smallest dataset failed to correctly detect the relationships within the data that resulted from the reactions between water and H_2S, and the effect of other input component flows to the reactions, making the model unusable in predicting the absorber column.

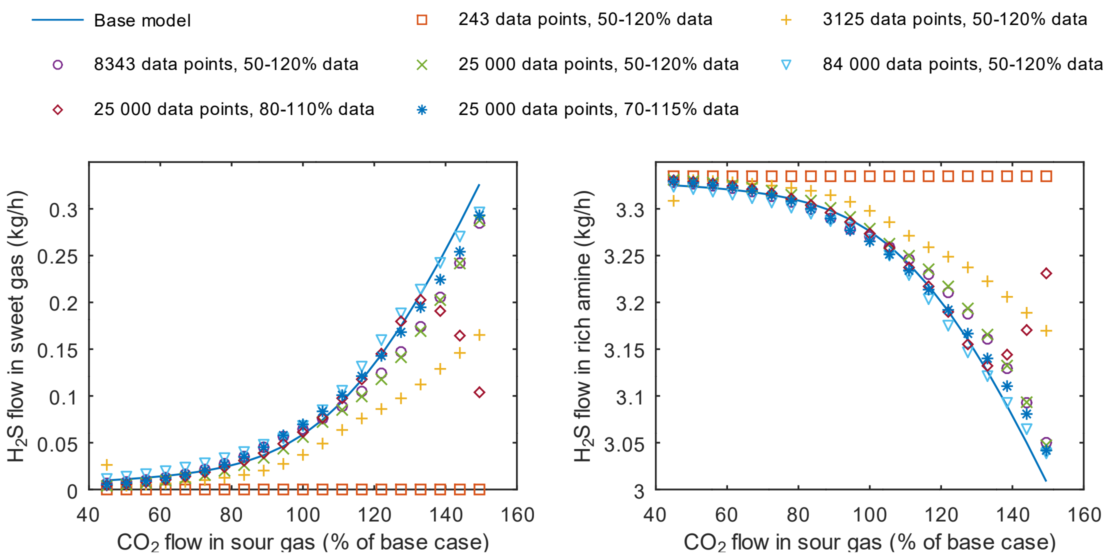


Figure 18. Sweet gas and rich amine H_2S flow predictions as functions of CO_2 input flow.

The effect of water input flow variation was rather accurately predicted by all models with 8000 or more training data (wider data), showing that the models detected the reaction equilibrium of H_2S dissolution to water shifting towards the reactants when the amount of water in the column was decreased. This is encouraging from the operational point of view since the water input flow can be changed to optimize the H_2S removal (Law et al., 2017). Overall, the best hybrid model in terms of predicting H_2S removal was the model with 25 000 training data points. The model trained with the narrowest data, on the other hand, deviates from the base model considerably with the largest CO_2 input flows and smallest water input flows, both towards a more optimistic prediction of CO_2 removal. Since the narrowest model is extrapolated furthest in these cases, it has the least information on the effects of the aforementioned input flows and thus seems to misinterpret the base model's behavior as a wave- or parable-like trends instead of a descending or ascending trend. Nevertheless, even this model is very accurate, actually one of the most accurate models, in the interpolated cases. Interestingly, the narrowest model outperformed the second-narrowest model when predicting the effect of methane input flow on H_2S removal. A more logical result would be that the narrowest training data would be outperformed by the model with slightly wider data. This could be explained simply by a more suitable neural network configuration regarding the H_2S removal having been found for the narrowest dataset.

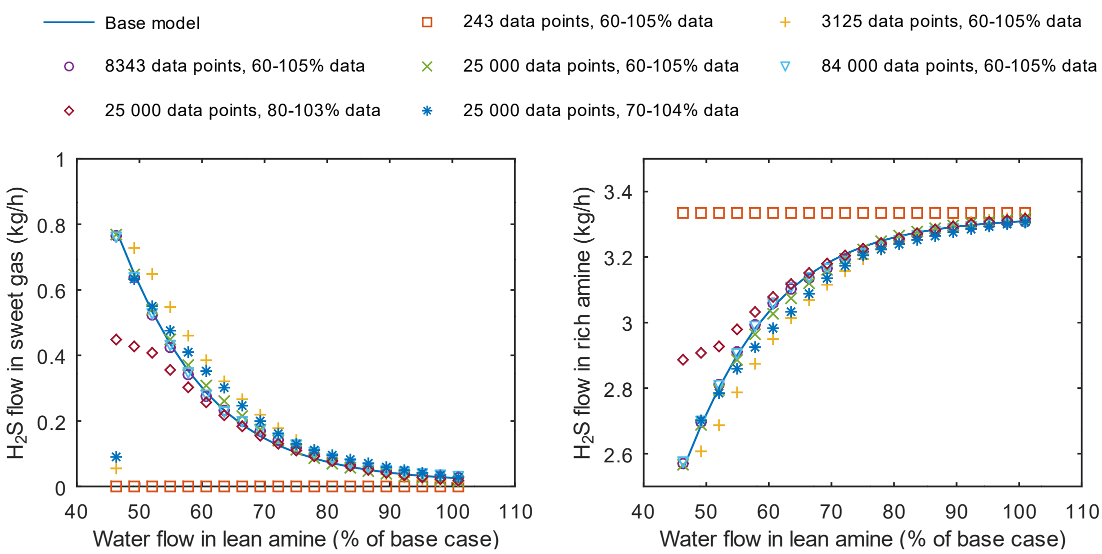


Figure 19. Sweet gas and rich amine H_2S flow predictions as functions of water input flow.

Figures 20 through 22 present the CO_2 removal predictions as functions of methane, water, and MDEA. Similarly as for H_2S, decreased methane input flows allowed for enhanced CO_2 removal as less methane was absorbed into the amine and more amine becomes in contact with the CO_2. As discussed above, CO_2 reacts with water and MDEA in the absorption process, and increasing the amount of MDEA introduced to the process improved the CO_2 removal. Increasing the amount of water into lean amine, on the other hand, slightly reduced the CO_2 removal.

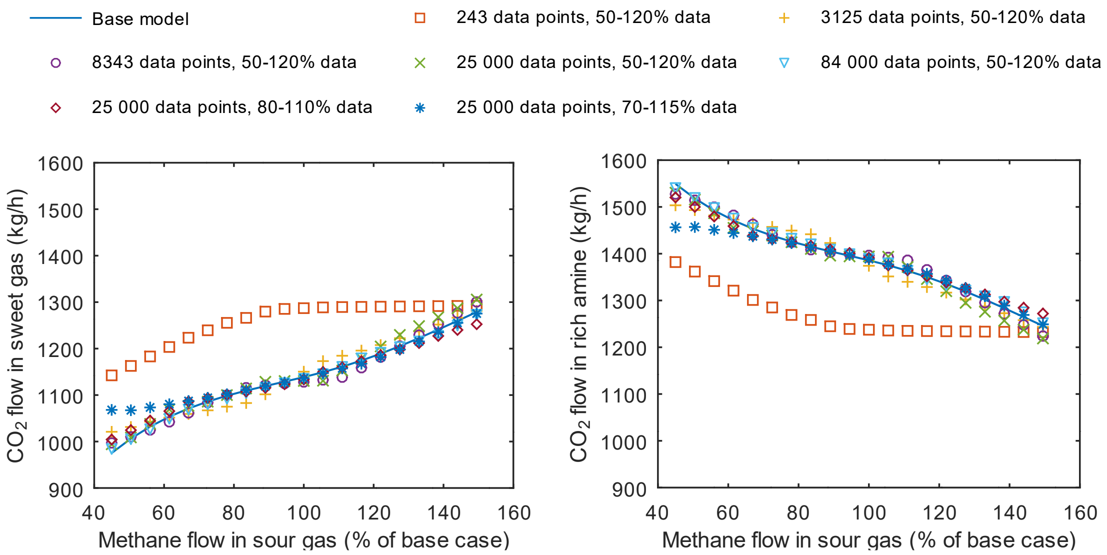


Figure 20. CO_2 removal predictions as functions of methane input flow.

Again, the hybrid model with the smallest dataset was inaccurate in all its predictions. Additionally, the model whose training data contained inputs ranging from 70 to 115% of the base case input flows deviated from the base model results notably with small methane input flows. Interestingly, the same model performed as well or even outperformed the model with the widest training data with large methane input flows and small water input flows. This shows that it would be difficult to choose the best model for extrapolation purposes, since the model performance may vary depending on the inputs. A better option would be to provide the model with adequately wide training data and choose the model with the best validation results in the interpolation area.

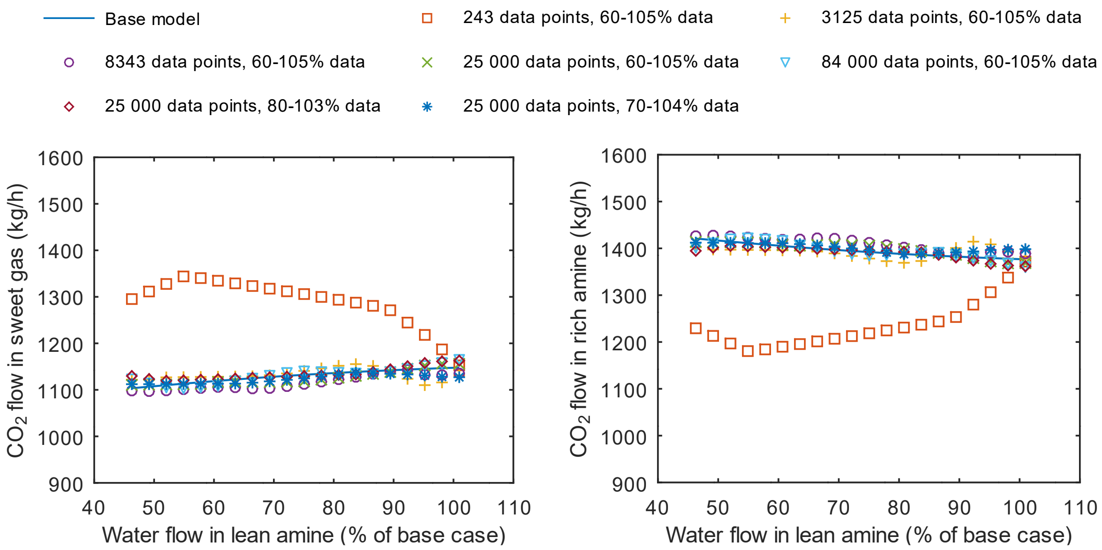


Figure 21. CO_2 removal predictions as functions of water input flows.

As already seen in the bar charts in the previous section, the rest of the other models predicted the CO_2 removal with small relative errors despite the absolute errors being multiple kilograms for CO_2. Particularly CO_2's reactions with MDEA were predicted accurately. Although the relative errors by most of the models were small, there was slight oscillation in the predictions of the effect of methane and water input flows. The oscillation was observed even when in test cases with inputs from inside the training data. The reason behind the phenomena could be overfitting since the training data included only 5 variation points for methane and water input flows in all training datasets in question. Similar oscillation cannot be seen in the prediction as a function of MDEA input flow, which had 7 or more variation points in the training datasets. Therefore, the models seem to have overfitted the relationship between methane and water input flows and sweet gas CO_2 flow. In future work, it would be worth testing to include more variation points to these components as well. Nevertheless, the positive effect cannot be guaranteed beforehand since similar oscillation or possible overfitting was not observed as clearly for other component flows as functions of methane and water inputs.

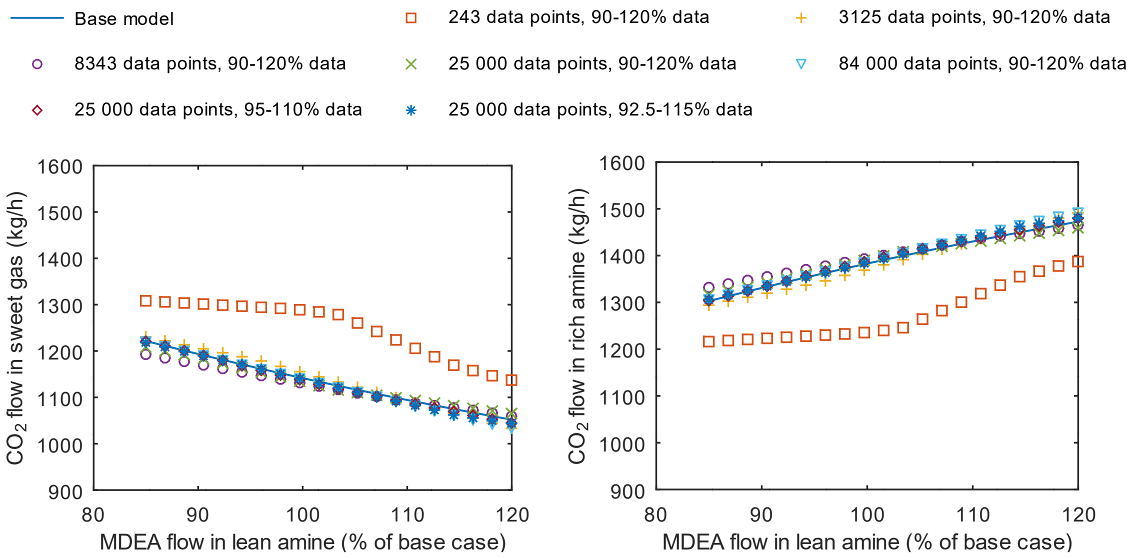


Figure 22. CO_2 removal predictions as functions of MDEA input flows.

According to Stewart (2011), the LNG gas sales limit for CO_2 content is 2-4 mol-%. Although the model with the smallest training dataset is inaccurate in its CO_2 removal predictions, it is more conservative than the base model. The relative differences between the other models and the base model are also small enough that the models did not predict the sweet gas to be below said limit in any case where the base model predicted the sweet gas CO_2 content to be above the limit. All in all, although there is room for improvement in the CO_2 predictions in terms of possibly increasing the number of variation points in the training data for methane and water, the CO_2 removal predictions are satisfactory.

7.5.3 Stream temperature predictions

The incoming stream temperatures were kept constant in the training datasets as well as the set of test cases, but the exothermic CO_2 absorption reactions and the stripping effect of methane in the process affect the output stream temperatures. Therefore, the product stream temperatures were used as dependent variables although the amine regeneration and circulation are not modelled in this thesis. It is important to model the product stream temperatures accurately, so that the modelling and design of the amine regeneration and circulation, as well as the equipment used in handling the sweetened gas, could be done accurately. For example, the rich amine is heated up to ca. 150-200 °C, and inaccurate rich amine temperature prediction would cause errors in estimating the energy consumption of rich amine heater.

Product stream temperature predictions as functions of methane, CO_2, and water are shown in figures 23 through 25. Decreasing the amount of CO_2 in the sour gas decreased the product stream temperatures as less of the exothermic CO_2 absorption occurred. Increasing the methane input flow also decreased the CO_2 absorption, which again decreased the product stream temperatures. Predictions are also shown as functions of water input flow in the lean amine, since smaller water flows resulted in slightly better CO_2 removal and thus had a decreasing effect on the sweet gas temperature. MDEA and H_2S input flow variations, in turn, had a rather small effect on the stream temperatures, and the trends were followed closely by most models. Thus, the figures on the effect of MDEA and H_2S on the temperature predictions can be found in Appendix VI.

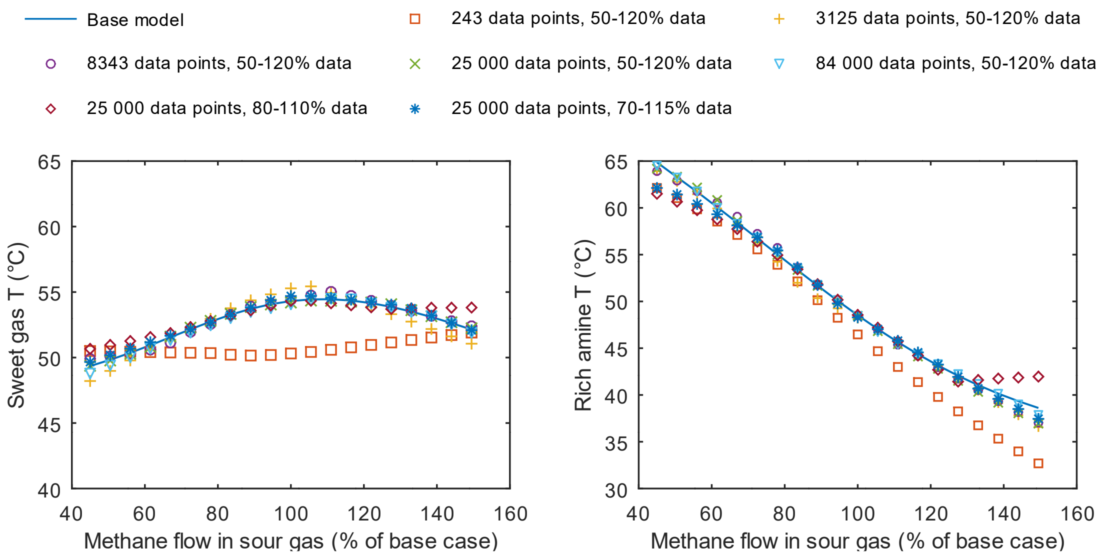


Figure 23. Sweet gas and rich amine temperature predictions as functions of methane input flow.

The product stream temperatures were predicted rather well by most models, and especially the models with wider training data and 8000 or more training data points.

The model with the widest training data of 25 000 data points was the most accurate when predicting the product stream temperatures, whereas the model with the narrowest training data showed a difference of a few degrees Celsius compared to the base model in the lowest and highest methane input flows. Also, the model with the second-widest data started to deviate from the base model's rich amine temperature predictions with the smallest CO_2 flows. As already concluded, the hybrid model reliability is considerably weakened when extrapolated this much, and extrapolation should therefore be avoided.

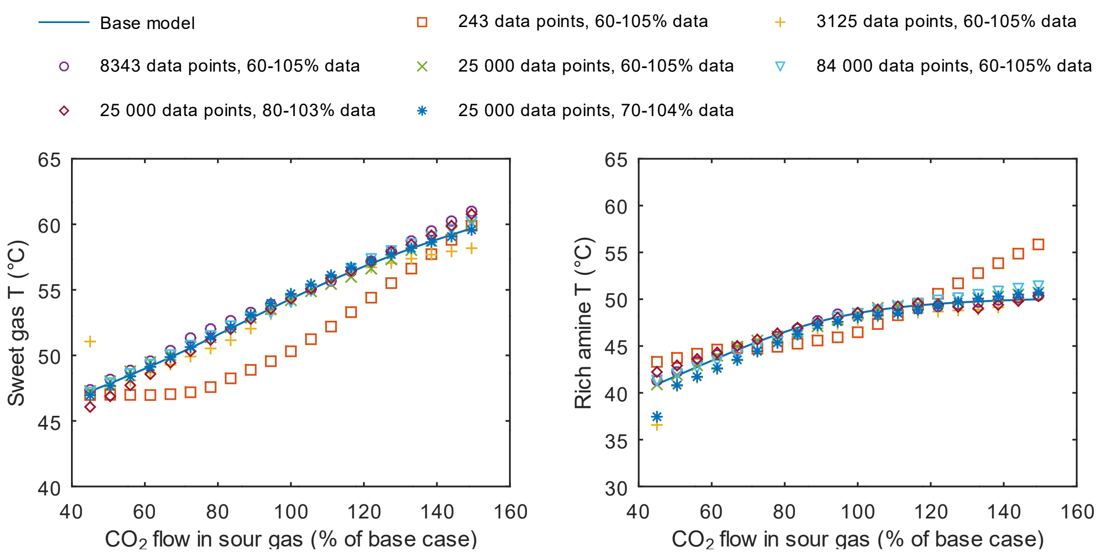


Figure 24. Sweet gas and rich amine temperature predictions as functions of CO_2 flow.

The accurate temperature predictions indicate that the models detected trends in the training data that results from the exothermic reaction, but also from the temperature changes depending on the relative amount of warmer lean amine compared to the cooler sour gas. Also, the effect of increased methane flow leading to increased stripping of water and therefore cooler rich amine was detected by the models. Errors of this magnitude, mostly less than one degree, seem acceptable, as they should not cause errors in the design of the amine system (absorption regeneration and auxiliary equipment). Again, the model with the smallest dataset failed to predict the temperatures accurately for both sweet gas and rich amine temperature predictions, although even that prediction is of the correct magnitude.

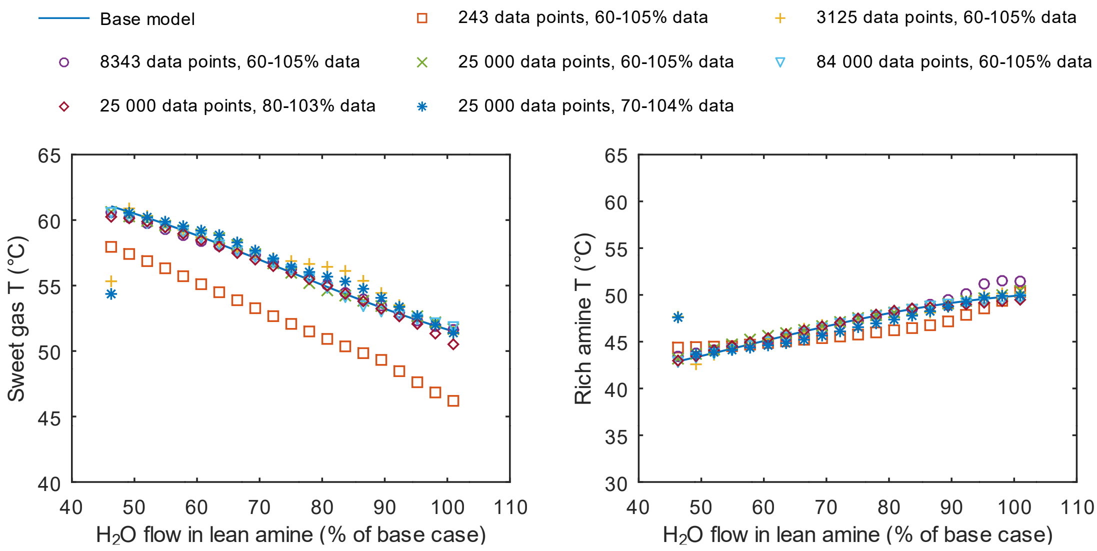


Figure 25. Sweet gas and rich amine temperature predictions as functions of water flow.

7.6 Extrapolation ability of FP driven, AI driven and reduced order models

According to the results presented above, the FP driven model with the widest 25 000-point training dataset (gas component input flow variation from 50 to 120 % of the base case flows) predicted the amine absorption H_2S flows and stream temperatures the most accurately. To further test the model's performance and how extrapolatable the model is, this FP driven model was validated by further extrapolation. The component input flows were varied, one by one, from 30 to 250 % of the base case molar flows for all components. The same wider validation was done to the AI driven and reduced order models created in AspenTech's AI Model Builder. The AI driven model was trained with the same dataset as the FP driven model. The ROM's training dataset was created separately in Aspen Plus Multi-Case using the reduced order project, but the upper and lower limits for the total input flows and mole fractions were set so that the data wideness did not differ from the FP driven model's dataset. The ROM dataset size was also the same as the FP driven model's dataset.

7.6.1 Component flow predictions by the different model types

Results are again shown for H_2S and CO_2 flows, and the rest of the figures can be found in Appendix VII. The figures show the upper and lower limits of the training datasets as vertical lines. In this work, more emphasis was put on optimizing the neural network configuration of the FP driven hybrid model than optimizing the AI driven and reduced order model configuration. Therefore, these results should not be considered as strict comparison of accuracy of the different model types, but rather an experiment on the usability of the models. The H_2S removal predictions with varying water input flows are shown in Figure 26. Decreasing the water input flow to smaller amounts continued to shift the H_2S dissolution reaction equilibrium towards the reactants, and eventually reduced the H_2S removal to ca. one third of the base case removal (97 %). The shift in the equilibrium was accurately predicted by all models. However, the ROM was slightly more conservative than the other models in its predictions with higher water input flows.

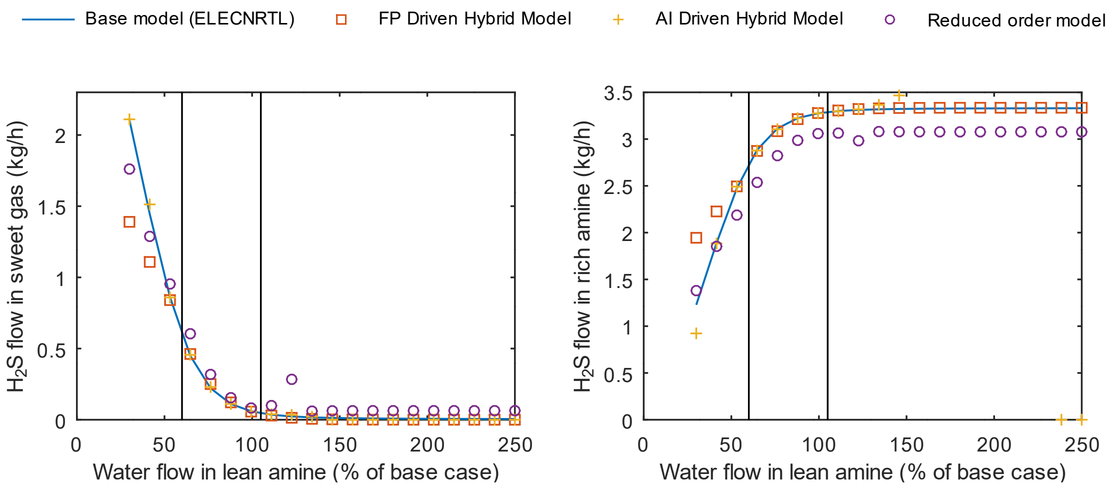


Figure 26. Sweet gas and rich amine H_2S predictions as functions of water input flow.

The effect of methane input flow variations on the H_2S removal predictions is shown in Figure 27. All models predicted the sweet gas H_2S as a function of methane input flow variations accurately in the interpolation area, but the AI driven model and ROM deviated from the base model's result considerably in extrapolation. In very small and very large methane input flows, slight improvements in H_2S removal can be seen, but the AI driven model and ROM predicted the changes to considerably reduce the H_2S removal. Additionally, the ROM gives oddly small results for H_2S flows in rich amine throughout all component variations, and it seems that the model was not very well optimized on the H_2S removal data.

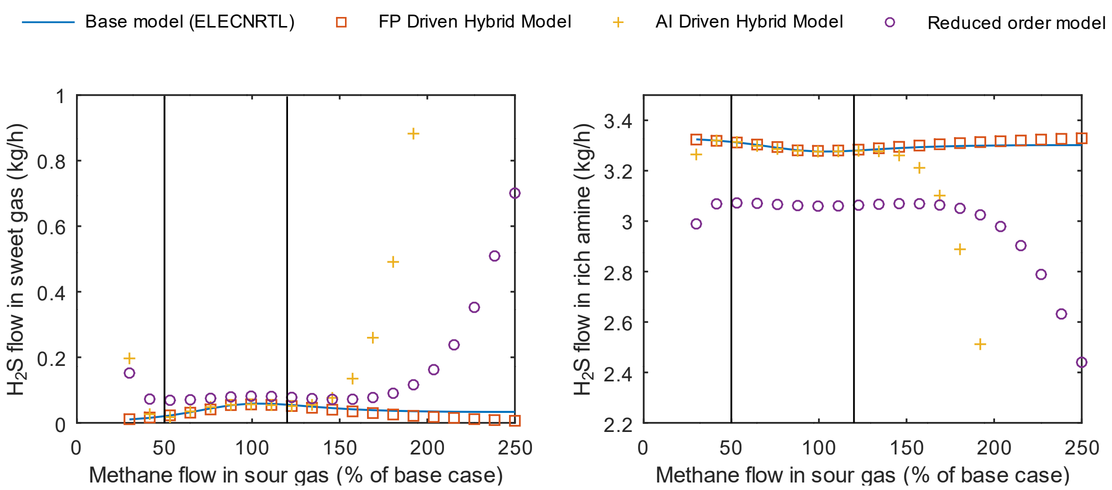


Figure 27. Sweet gas and rich amine H_2S predictions as functions of methane input flow.

Regarding the usability of the models in terms of H_2S removal prediction accuracy, the FP driven hybrid model can withstand some extrapolation, as discussed in Section 7.5.2. However, it gets slightly too optimistic in the largest methane flows: according to the base model, the H_2S flow in the gas product was 0.035 kg/h when methane input flow was 55 000-60 000 kg/h, whereas the FP driven hybrid model predicted a 0.006 kg/h flow of H_2S in the gas product. Although the absolute difference is small, the concentration of H_2S in the hybrid model's gas product flow was ca. 3.4 mg/m^3 (2.4 ppm by volume), whereas the base model's product contained 18.8 mg/m^3 or 13.5 ppm by volume of H_2S. According to the hybrid model, the H_2S content's limit of 4 ppm by volume (Stewart, 2011) is not exceeded although the base model's results clearly exceed the limit. Based on this result only, extrapolating a hybrid model this much is not advisable, particularly in cases where small component flows are very relevant for the model. In embodiment, if a wider set of use cases is needed, also the training data should be wider. This would likely require an increased number of training data points to maintain the accuracy.

The effect of MDEA input flow variation on H_2S removal is shown in Figure 28. Although H_2S only reacts with water in the process and varying the MDEA input flow had otherwise little effect on the H_2S removal, decreasing MDEA flow to very small amounts caused a sharp decrease in the removal. This change in the trend of the removal only occurred at ca. 3000 kg/h or smaller MDEA flows. Therefore, the effect of decreasing MDEA flow in the training data was linear, and the models could not predict the reduced H_2S removal at all.

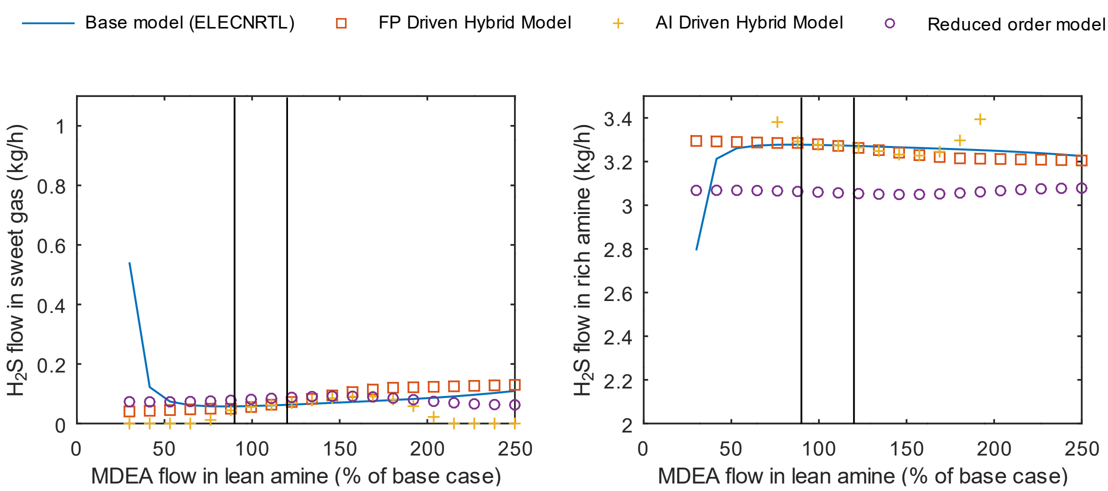


Figure 28. Sweet gas and rich amine H_2S predictions as functions of MDEA input flow.

The CO_2 removal predictions by the different models can be seen as functions of methane, water, and MDEA flow in Figure 29 and Figure 30. As already observed in Section 7.5.2, there was some oscillation in the predictions of the FP driven model when methane and water input flows were varied. Also the ROM, which models the process with a neural network, showed some oscillation, whereas the linear AI driven model seems to have formed more of a parable type trend. While the FP driven model mostly outperforms the other models in all input flow variations, particularly in the interpolation areas, the performance of the AI driven and reduced order models differ depending on the predicted variable. For example, the AI driven model was considerably more accurate in predicting the effect of methane flow on CO_2 removal, but the reduced order model is more accurate in predicting the effect of MDEA flow variations. Overall, it seems that the linear AI driven model was better optimized for CO_2 removal than the neural network ROM, just as it was for H_2S removal.

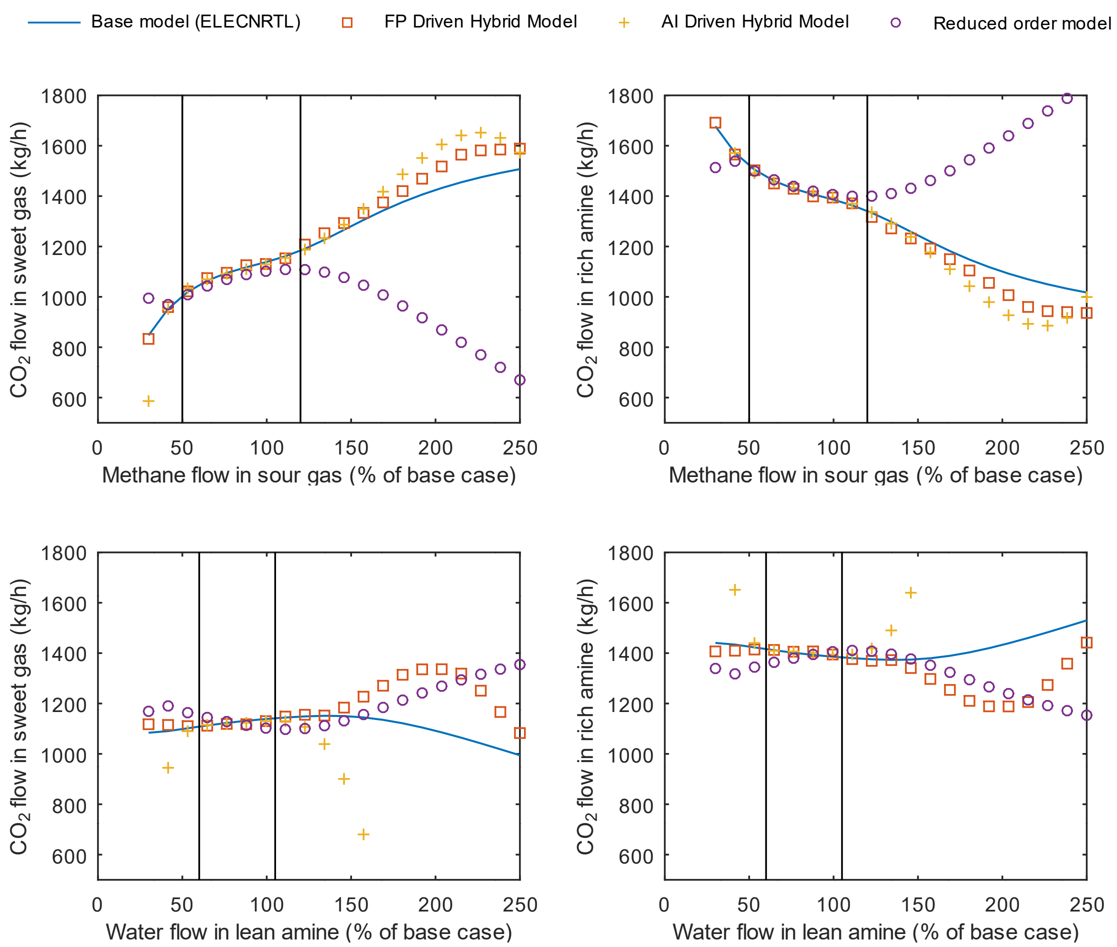


Figure 29. Sweet gas and rich amine CO_2 predictions as functions of methane and water input flows.

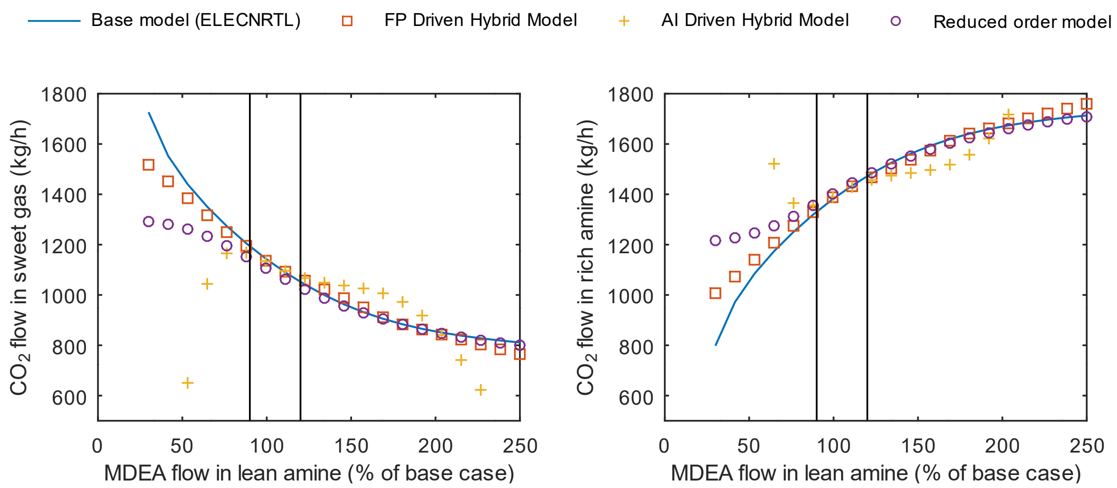


Figure 30. Sweet gas and rich amine CO_2 predictions as functions MDEA input flows.

Overall, the FP driven model mostly outperformed the other two models created in this work clearly, which was predictable due to the efforts put into its optimization. However, the wide extrapolation tests, and particularly the CO_2 removal predictions in far extrapolation, showed that even the otherwise accurate FP driven model deviates from the base model strongly in far enough extrapolation. The deviation was the strongest for CO_2 predictions, as it was already deemed slightly more difficult to train than the other dependent variables, and especially the effect of methane and water input flows caused trouble in the predictions due to their smaller number of deviation points in the data.

7.6.2 Stream temperature predictions by the different model types

Sweet gas and rich amine temperatures as functions of methane and water input flows are shown in Figure 31. The upper and lower limits of the training data are shown as vertical lines. In the interpolation areas, all models predicted the stream temperatures with adequate accuracy, but extrapolation caused all models to deviate strongly from the base model in almost all component flow variations. The linear AI driven model seems to show the largest difference to the base model when extrapolated, whereas the FP driven model and ROM with neural networks showed at least slight following of the base model trend.

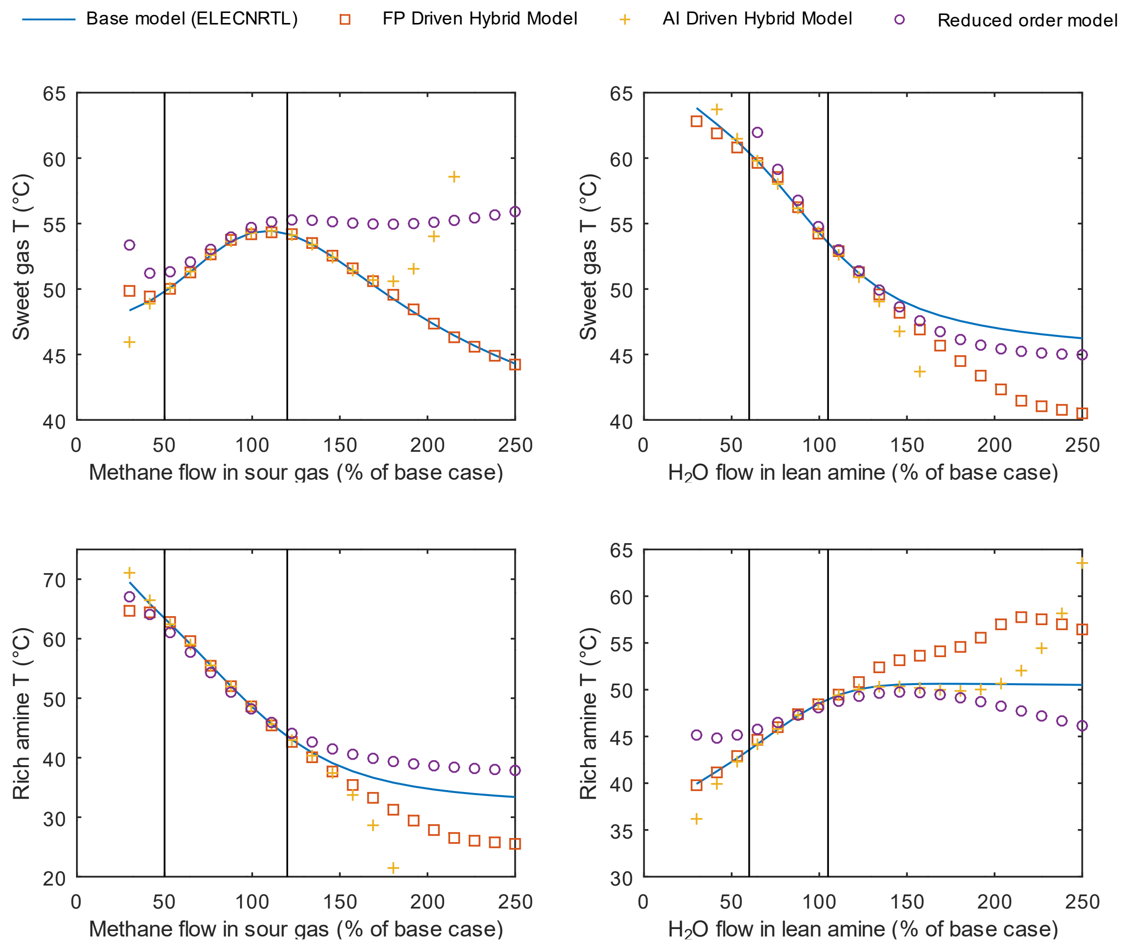


Figure 31. Sweet gas and rich amine temperatures as functions of methane and water input flows. In the model training data, methane and water input flow variations were 50-120% and 60-105% from base case flows, respectively.

While the results were of the correct order of magnitude, the models predicted the temperatures to develop in a similar trend as in the interpolation area, since their training data did not contain information on the temperature development in further extrapolation areas. Nevertheless, the FP driven model predicted the sweet gas temperature accurately in all test cases. This indicates that the model was well optimized for the effect of methane flows to the temperatures and managed to detect the slight decrease in the temperature in the higher end of training data. Out of the three AspenTech tools used in this work, the most accurate model regarding both component flow and stream temperatures was created with the First Principles Driven Hybrid Modeling tool. However, some parameters of the AI driven and reduced order model configurations were left out of the scope of this work, and future studies on those model properties could lead to as high-performing models as the FP driven hybrid model.

7.7 Time consumption on data generation and hybrid model training

Hybrid model training datasets were generated with Aspen's Multi-Case tool. While generating data, the single case calculation time seemed to vary with no apparent reason. Especially when left running overnight for larger datasets, the running would occasionally halt during the night, resulting in very long overall time elapsed in data generation, although the actual calculation time may have been shorter. Table 20 shows that the larger datasets clearly took a longer time to generate. At fastest, Multi-Case was able to run more than 1000 test cases in 10 minutes (when generating dataset 6), but sometimes running the cases was substantially slower.

Table 20. Time elapsed in generating the datasets with Aspen Multi-Case.

Dataset no.	Dataset size	Time elapsed in data generation
1	243	5 min
2	3125	35 min
3	8575	2 h 30 min
4	25 000	5 – 8 h*
5	84375	24 – 40 h*
6	25 000	3 h 40 min
7	25 000	4 h 20 min
8	25 000	5 h 3 min
10 (Reduced Order)	25 000	12 h 15 min

* Multi-Case stopped proceeding during the night and the run had to be paused and continued. Thus, time elapsed in generating datasets 4 and 5 is unclear.

Although the best results in this work were obtained with the dataset with 25 000 data points, it is possible that even more accurate models could be achieved with further optimization of the model with 84 000 training data points. While machine learning can provide benefits to modelling by shortened calculation times of simulations that normally may take several hours to converge, generating datasets with as much as 84 000 or more data points seems rather time-consuming. However, conventional model development may take such a long time that the data generation would not be a limiting factor in the model development process, particularly since data generation is passive work that does not require constant monitoring.

Exemplary times elapsed in hybrid model training for models with different training datasets and NN configurations are shown in Table 21. Also the validation error after model training (FP driven models) or average R^2s (AI driven and ROM) are shown. Generally, the larger the dataset, the longer the hybrid model training took due to the longer time taken by each training epoch to pass. Additionally, the FP driven modelling tool terminates model training if the validation error is not 0.99-fold or smaller within 10 epochs. Therefore, the unsuitable NN configurations which resulted in larger validation errors, were trained in a shorter time than those resulting in smaller validation errors. Model complexity, on the other hand, seemed to have less effect on the training time when the software was able to find a good fit of model on the training data. For less accurate training results though, simpler model was faster to train than a more complex one, as can be seen for the model with ca. 8000 training data points: the NN with 3 hidden layers with 14 neurons each resulted in high validation error and was slightly slower to train than the NN with 1 hidden layer with 8 neurons, whose validation error was smaller. Nevertheless, the iteration process for the suitable NN configuration was the most time-consuming part, and for effective working, the largest datasets (in this case, 25 000 data points and more) don't seem a good starting point for the iteration, when models are trained with the Aspen Plus AI Training tool.

Table 21. Time elapsed in hybrid model training with different NN configurations (activation function ReLu for FP driven models and Swish for reduced order model) and training datasets. AVG R^2 stands for average R^2 in the AI driven model and ROM.

Dataset size	**Hidden layers**	**Neurons per hidden layer**	**Parameters**	**Validation error or AVG R^2 (AI, ROM)**	**Training time (min)**
8343	3	14	728	40.96	12
8343	1	8	192	9.77	10
8343	2	8	256	0.23	12
8343	2	12	432	0.070	25
243	2	18	756	14.39	1
3125	2	18	756	0.115	9
8343	3	18	1080	0.037	30
25 000	2	16	640	0.020	60
84375	2	18	756	0.072	120
25 000 (AI Driven)	Linear Lasso CV method			AVG R^2: 0.997	3
25 000 (ROM)	2	18	756	AVG R^2: 0.991	3

Overall, the iteration process to find any suitable NN configurations with the FP driven modelling tool is rather time-consuming if dozens of configurations need to be tested, but faster iteration can be done with AI driven and reduced order models. The time spent in training was over 15 hours for the model with 8000 training data points (including learning rate tests and NN configuration tests), ca. 1 hour for the dataset with ca. 3000 training data points and ca. 8 hours for the model with ca. 25 000 training data points. The times include only the times elapsed in model training, and not the time spent in altering the NN configuration and recording the results. Therefore, the iteration process should be done as systematically as possible.

7.8 Hybrid model computational time

When running the base model and hybrid models with ASW for validation, the 100 test cases were finished in ca. 13 minutes with the base model, and with ca. 5 minutes with the models deploying AspenTech's machine learning tools. Based on this, the base model computational time was ca. 8 seconds per case, and the FP driven, AI driven, and reduced order model's computational time was 3 seconds per case. Thus, it does seem that the FP driven, AI driven, and reduced order models are lighter to run and offer advantage in computational time compared to the base model. Although the number of parameters in the FP driven neural networks varied, the test case set computational time was roughly the same.

Nevertheless, in the scope of this work, the hybrid models were deployed in a small and simple simulation flowsheet, and the absolute difference between the base model and hybrid model computational times were not large. Therefore, it remains unclear how the computational time would be affected if the hybrid models were deployed as a part of a larger flowsheet that normally takes a long time to converge. The same applies to the training of the first principles driven hybrid models, which may take a longer time in a larger flowsheet. This downside can be avoided by copying the neural network from a simpler flowsheet into the more complex flowsheet, given the target flowsheet contains streams and blocks with identical names as in the simpler flowsheet used to train the hybrid model.

7.9 Comparison of the usability of AspenTech machine learning tools

Since more emphasis was in this work put on optimizing the FP driven model configuration than on optimizing the AI driven and reduced order models, the modelling results cannot be used as a strict comparison of modelling abilities of the tools. However, it is of interest to qualitatively analyze the functionalities of the different tools used. Important aspects regarding modelling and utilizing the hybrid models in larger simulation flowsheets are listed and compared in Table 22.

Table 22. Comparison of different aspects in utilizing the hybrid models.

Modelling aspect	First Principles Driven Hybrid Models	AI Driven Models	Reduced Order Models
Time consumption on generating data	Faster	Faster	Slower
Time consumption on model optimization	Slower	Faster	Faster
Constraints to avoid unphysical model behavior	FP model accounts for balances and physicality of flow, fractions, temperatures, pressures etc.	Optional: Mass- and Component balances	Optional: Component balances, Engineering constraints, User-defined constraints
Accuracy and extrapolation abilities	High accuracy in interpolation, can withstand slight extrapolation (extrapolation not recommended)	Mostly high accuracy in interpolation, further model optimization recommended – poor results in extrapolation	
Importing the model into a larger simulation	NN can be copied and pasted, given streams and blocks with equal names exist in the target flowsheet	Importing via Customize ribbon, user can define streams in the flowsheet as correct streams for the hybrid model	
Model robustness: additional components	User can define the split fraction for additional components, all goes into one stream by default	Additional components evenly distributed to outlet streams if mole frac. in the feed is < 0.0001. Otherwise results in an error.	
Model robustness: validation cases resulting in errors	None	None, if additional components are not added	
Required ML background knowledge	Neural network model with a large number of basic and advanced settings	Linear model with limited number of settings	Neural network model with limited number of settings

As already discussed, time consumption on generating data and optimizing hybrid model configurations may not be the most important factor when choosing a modelling technique. However, there might be different modelling needs in different occasions. For example, an AI driven model might be a useful tool when rough estimates are needed to be made in a short time, whereas the FP driven models might provide more accurate predictions if suitable model configurations are found. The available constraints also encourage the use of FP driven and reduced order models over AI driven models in more accurate modelling.

Deploying the AI driven and reduced order models in larger simulation flowsheets might be slightly easier than copying neural networks from FP driven models. Since the FP models seem primarily intended to be used in their original flowsheet, they are slightly more prone to difficulties when transferring to a new flowsheet due to differences in stream and block names. AI driven and reduced order models have more flexibility in their deployment. On the other hand, the FP driven models created in this work (separator unit with split fractions as neural network outputs) seem to handle additional components better by giving the user a chance to define the split fraction of all additional components. The AI driven and reduced order models result in errors and do not calculate the feed's stream results, if too high mole fractions (>0.0001) of unknown components are present the feeds. That being said, the errors can be avoided by separating the unknown components from the feeds before the hybrid model and inserting them back into the correct product streams.

As mentioned before, the issues with convergence hinder the use of rate-based, reactive absorber models as part of larger simulation flowsheets. The hybrid models offer a great advantage in convergence and thus model robustness since none of the test cases were left unconverged. The FP driven model seems to be even more robust than AI driven models and ROMs, since unknown components do not result in modelling errors, whereas some adjustments are required for the AI driven models and ROMs to work with unknown components present.

While AspenTech's hybrid modelling tools do not require programming knowledge to successfully train models, optimizing the model configuration is easier and more efficient if the user has some knowledge on the basic ML model parameters. Out of the three model types, the FP driven models have the wideset selection of advanced NN properties, whereas the AI driven models offer only a limited number of linear model settings. Nevertheless, as the default settings in the advanced properties resulted in accurate models, basic knowledge seems successful. Additionally, while the FP driven modelling allows adjusting a larger number of settings, it also constraints the model more automatically than the other models and in that sense requires less background information on the constraints required to avoid nonphysical model predictions.

8 Conclusions and suggestions for future work

The aim of this thesis was to study the possibilities in using machine learning methods in process simulation applications. Hybrid modelling tools provided by AspenTech were utilized to create first principles (FP) driven, AI driven, and reduced order models (ROMs) of an amine absorption column removing hydrogen sulfide and carbon dioxide from sour gas. With the first principles driven hybrid models, the effect of neural network properties, dataset size, and dataset wideness were studied with the aim of finding a well-performing neural network configuration and some kind of minimum to the amount and variation of data required in successful model training in this particular application. Additionally, the effect of different properties of a neural network were studied with the first principles driven models. Extrapolation abilities of the three hybrid model types were studied by validating against the base model with a wider set of test cases.

The models created for this work, particularly the FP driven model with the widest training data, were able to accurately predict the component flows and stream temperatures in interpolated test cases while shortening the computational time from base model's 8 seconds to ca. 3 seconds per case. The models also showed increased robustness, as all test cases converged, whereas use of the base model as a part of a larger simulation flowsheet is hindered due to slow or unachieved convergence. Another key finding in these tests was that extrapolating the hybrid models should be avoided, since most model predictions of the key components removed from sour gas in absorption, CO_2 and H_2S, differed significantly from the base model's results in extrapolation. Additionally, the performance of the models may vary between components and the size of component input flows, which would make it difficult to decide on the best model in extrapolated cases.

The studies with neural network configuration, particularly with the number of hidden layers and neurons, showed that it is difficult to determine a definitive, numerical rule for deciding the suitable configuration. Nonetheless, well-performing configurations were found with systematic testing, which was started with a dataset of ca. 8000 data points. Similar approach is recommended for future work as well and training with larger datasets should be started only when there is an approximate of suitable ML model configurations. The tests with dataset size and wideness on the FP driven models showed that larger and wider training data resulted in better-performing models. The most accurate model was trained with 25 000 data points, where the input gas component flows varied from 50 to 120 % of the base case flows. Also the model trained with 84 000 data points was accurate in its predictions, and thus it seems that 25 000 data points is in the lower end of the range for suitable amount of data. In future work, the amount of data could be increased by adding more variation steps for methane and water, which now had only 5 variation steps. This way, the oscillation in CO_2 flow predictions could perhaps be decreased as possible overfitting would be avoided. Additionally, it would be of interest to study the suitable amount of data for systems with more or less variables to model. Regarding suitable training data wideness, the models trained with narrower datasets deviated from the base model earlier than the model trained with the widest dataset, further implying that all possible use cases should be included in the training data.

When comparing best-performing FP driven, AI driven and reduced order models, the FP driven model showed the most accurate predictions, although all models followed the base model trends rather closely in the interpolated cases. When model creating is considered, the data generation for ROMs was slower than data generation for the other models, whereas training the FP driven models was considerably slower than training the other models. In this sense, the strengths of the different model types can be leveraged in different needs, by for example using the fast-to-create AI driven models for quick and rough estimates, and the FP driven and reduced order neural network models for more accurate modelling.

Another factor considered in this work was the background knowledge needed on machine learning methods. Extensive programming or machine learning knowledge is not required to create accurate models with the Aspen Plus hybrid modelling tools, but finding well-performing model configurations can be considerably more effective if the user is familiar with at least the basic properties that can be adjusted in the settings and knows how the different properties may affect the model. The black-box nature of especially the neural network models is emphasized if the user is completely unfamiliar with the properties adjusted. However, default settings are available for all models, and well-performing models can be, and were in this work, created without adjusting the more advanced settings. Additionally, as no programming is required, more time can be spent on adjusting the model settings effectively instead of creating suitable program code for a similar ML model.

All in all, the usability of the hybrid models could be further extended, for example, to sensitivity analysis of an absorber column, if the training data was modified to contain more information on how the key operational factors (input component flows, temperatures, and pressures) of the process affect the products. In this form with the flowsheet consisting of a separator unit and heaters, the FP driven model cannot be used to model the reactions with true components. While simplifying the model with apparent components may be desired, the AI driven models and ROMs could also model the true component reactions. Studying this opportunity was left out of the scope of this thesis, but it would be of interest to study these abilities in future work. Additionally, it would be useful to further examine the hybrid models' performance as a part of a larger simulation flowsheet. Since all models can be relatively easily inserted into another flowsheet, it would be useful to know how the presence of other unit operations and recycle streams affect the hybrid models, particularly when reactions are present in the process. Moreover, the neural network size did not seem to have a notable effect on the hybrid model computational times. Thus, modelling an entire system instead of just a unit operation with a neural network also seems to be an interesting opportunity. However, the data generation could be in this case too time-consuming, particularly if all parameters of the system would be varied in the data. Nevertheless, these aspects would provide more information on how much the hybrid model can enhance the modelling effectiveness by shortening the computational times.

References

Aguiar, H. C. & Filho, R. M. 2001. Neural Network and Hybrid Model: a Discussion About Different Modeling Techniques to Predict Pulping Degree with Industrial Data. *Chemical Engineering Science,* 56**,** 565-570.

AspenTech. 2020a. *Hybrid Modeling: AI and Domain Expertise Combine to Optimize Assets* [Online]. Aspen Plus. Available: https://www.aspentech.com/en/solutions/aspen-hybrid-models [Accessed 4.1. 2023].

AspenTech 2020b. Rate-Based Model of the CO_2 Capture Process by MDEA using Aspen Plus.

AspenTech 2022a. Aspen Multi-Case Help.

AspenTech. 2022b. *Build and Deploy a First Principles Driven Hybrid Model for Hydrogen Production via Methane Steam Reforming* [Online]. Available: https://esupport.aspentech.com/S_Article?id=000099498 [Accessed 5.1. 2023].

Bender, A., Schneider, N., Segler, M., Patrick Walters, W., Engkvist, O. & Rodrigues, T. 2022. Evaluation Guidelines for Machine Learning Tools in the Chemical Sciences. *Nature Reviews Chemistry,* 6**,** 428-442.

Bikmukhametov, T. & Jäschke, J. 2020. Combining Machine Learning and Process Engineering Physics Towards Enhanced Accuracy and Explainability of Data-driven Models. *Computers & Chemical Engineering,* 138.

Bouras, C. & Tsogkas, V. 2014. Improving News Articles Recommendations via User Clustering. *International Journal of Machine Learning and Cybernetics,* 8**,** 223-237.

Bu, C. & Zhang, Z. 2020. Research on Overfitting Problem and Correction in Machine Learning. *Journal of Physics: Conference Series,* 1693.

Byun, M., Lee, H., Choe, C., Cheon, S. & Lim, H. 2021. Machine Learning Based Predictive Model for Methanol Steam Reforming with Technical, Environmental, and Economic Perspectives. *Chemical Engineering Journal,* 426.

Carranza-Abaid, A. & Jakobsen, J. P. 2022. Neural Network Programming: Integrating First Principles into Machine Learning Models. *Computers & Chemical Engineering,* 163.

Carranza-Abaid, A., Svendsen, H. F. & Jakobsen, J. P. 2020. Surrogate Modelling of VLE: Integrating Machine Learning with Thermodynamic Constraints. *Chemical Engineering Science: X,* 8.

Carranza-Abaid, A., Svendsen, H. F. & Jakobsen, J. P. 2023. Thermodynamically Consistent Vapor-liquid Equilibrium Modelling with Artificial Neural Networks. *Fluid Phase Equilibria,* 564.

Cassola, S., Duhovic, M., Schmidt, T. & May, D. 2022. Machine Learning for Polymer Composites Process Simulation – a Review. *Composites Part B: Engineering,* 246.

Del-Mazo-Alvarado, O. & Bonilla-Petriciolet, A. 2022. Assessment of the Simultaneous Regression of Liquid-liquid and Vapor-liquid Equilibria Data of Binary Systems Using NRTL and Artificial Neural Networks *Fluid Phase Equilibria,* 561.

den Bakker, I. 2017. *Python Deep Learning Cookbook: over 75 Practival Recipes on Neural Network Modeling, Reinforcement Learning, and Transfer Learning Using Python,* Birmingham, England, Packt.

Ding, J., Xu, N., Nguyen, M. T., Qiao, Q., Shi, Y., He, Y. & Shao, Q. 2021. Machine Learning for Molecular Thermodynamics. *Chinese Journal of Chemical Engineering,* 31**,** 227-239.

Dobbelaere, M. R., Plehiers, P. P., Van de Vijver, R., Stevens, C. V. & Van Geem, K. M. 2021. Machine Learning in Chemical Engineering: Strengths, Weaknesses, Opportunities, and Threats. *Engineering,* 7**,** 1201-1211.

Engel, T. & Reid, P. 2014. *Physical Chemistry: Pearson New International Edition,* Harlow, Essex, England, Pearson.

Farajnezhad, A., Afshar, O. A., Khansary, M. A., Shirazian, S. & Ghadiri, M. 2016. Correlation of Interaction Parameters in Wilson, NRTL and UNIQUAC Models Using Theoretical Methods. *Fluid Phase Equilibria,* 417**,** 181-186.

Foo, Y., Chemmangattuvalappil, D. C., Ng, N., Elyas, D. K. S., Chen, R., Elms, C.-L., Lee, R. D., Chien, H.-Y., Chong, I.-L., Chong, S. & Hwa, C. 2017. *Chemical Engineering Process Simulation*, Elsevier.

Hall, S. M. 2018. *Rules of Thumb for Chemical Engineers*, Elsevier.

Heese, R., Nies, J. & Bortz, M. 2020. Some Aspects of Combining Data and Models in Process Engineering. *Chemie Ingenieur Technik,* 92**,** 856-866.

Hwangbo, S., Al, R. & Sin, G. 2020. An Integrated Framework for Plant Data-driven Process Modeling Using Deep-learning with Monte-Carlo Simulations. *Computers & Chemical Engineering,* 143.

Iftakher, A., Aras, C. M., Monjur, M. S. & Hasan, M. M. F. 2022. Data‐driven Approximation of Thermodynamic Phase Equilibria. *AIChE Journal,* 68.

Jirasek, F., Alves, R. A. S., Damay, J., Vandermeulen, R. A., Bamler, R., Bortz, M., Mandt, S., Kloft, M. & Hasse, H. 2020. Machine Learning in Thermodynamics: Prediction of Activity Coefficients by Matrix Completion. *J Phys Chem Lett,* 11**,** 981-985.

Jones, M. N., Frutiger, J., Ince, N. G. & Sin, G. 2019. The Monte Carlo Driven and Machine Learning Enhanced Process Simulator. *Computers & Chemical Engineering,* 125, 324-338.

Kartal, F. & Özveren, U. 2020. A Deep Learning Approach for Prediction of Syngas Lower Heating Value from CFB Gasifier in Aspen plus®. *Energy,* 209.

Krishnan, S. 2021. How to Determine the Number of Layers and Neurons in the Hidden Layer? Available from: https://medium.com/geekculture/introduction-to-neural-network-2f8b8221fbd3 [Accessed 17.4. 2023].

Law, L. C., Yusoff Azudin, N. & Abd. Shukor, S. R. 2017. Optimization and Economic Analysis of Amine-based Acid Gas Capture Unit Using Monoethanolamine/Methyl Diethanolamine. *Clean Technologies and Environmental Policy,* 20, 451-461.

Lee, J. H., Shin, J. & Realff, M. J. 2018. Machine Learning: Overview of the Recent Progresses and Implications for the Process Systems Engineering Field. *Computers & Chemical Engineering,* 114, 111-121.

Linzner, D. & Koeppl, H. 2021. Active Learning of Continuous-time Bayesian Networks Through Interventions. *Journal of Statistical Mechanics: Theory and Experiment,* 2021.

Lopez‐Zamora, S., Kong, J., Escobedo, S. & Lasa, H. 2021. Vapour‐liquid‐liquid and Vapour‐liquid equilibrium of Paraffinic Aromatic Synthetic Naphtha/Water Blends: Prediction of the Number of Phases. *The Canadian Journal of Chemical Engineering,* 100, 866-884.

Macukow, B. 2016. Neural Networks - State of Art, Brief History, Basic Models and Architecture. *IFIP International Conference on Computer Information Systems and Industrial Management.* Vilnius, Lithuania: Springer, Cham.

Mashaleh, A. S., Binti Ibrahim, N. F., Al-Betar, M. A., Mustafa, H. M. J. & Yaseen, Q. M. 2022. Detecting Spam Email with Machine Learning Optimized with Harris Hawks optimizer (HHO) Algorithm. *Procedia Computer Science,* 201**,** 659-664.

Molnar, C., Casalicchio, G. & Bischl, B. 2021. Interpretable Machine Learning - A Brief History, State-of-the-Art and Challenges. *ECML PKDD 2020 Workshops.* Cham: Springer International Publishing.

Mowbray, M., Vallerio, M., Perez-Galvan, C., Zhang, D., Del Rio Chanona, A. & Navarro-Brull, F. J. 2022. Industrial Data Science – a Review of Machine Learning Applications for Chemical and Process Industries. *Reaction Chemistry & Engineering,* 7**,** 1471-1509.

Mudhasakul, S., Ku, H.-m. & Douglas, P. L. 2013. A Simulation Model of a CO_2 Absorption Process with Methyldiethanolamine Solvent and Piperazine as an Activator. *International Journal of Greenhouse Gas Control,* 15**,** 134-141.

Nasiriany, S., Thomas, G., Wang, W., Yang, A., Listgarten, J. & Sahai, A. 2019. *A Comprehensive Guide to Machine Learning*, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley.

Neuneier, R. & Zimmermann, H. G. 2012. How to Train Neural Networks. *Neural Networks: Tricks of the trade.* 2nd ed. München, Germany: Springer.

Ng, A. 2018. *Draft Version - Machine Learning Yearning - Technical Strategy for AI Engineers, In the Era of Deep Learning*, DeepLearning.AI.

Pan, I., Mason, L. R. & Matar, O. K. 2022. Data-centric Engineering: Integrating Simulation, Machine Learning and Statistics. Challenges and Opportunities. *Chemical Engineering Science,* 249.

Parihar, V. & Yadav, S. 2022. Comparative Analysis Of Different Machine Learning Algorithms To Predict Online Shoppers' Behaviour. *International Journal of Advanced Networking and Applications,* 13**,** 5169-5182.

Ravi, R., Vinu, R. & Gummadi, S. N. 2017. *Coulson and Richardson's Chemical Engineering, Volume 3A - Chemical and Biochemical Reactors and Reaction Engineering,* Kidlington, Oxford, United Kingdom, Elsevier, Butterworth-Heinemann.

Rebala, G., Ravi, A. & Churiwala, S. 2019. *An Introduction to Machine Learning*, Springer: Cham.

Reynel-Ávila, H. E., Bonilla-Petriciolet, A. & Tapia-Picazo, J. C. 2019. An Artificial Neural Network-based NRTL Model for Simulating Liquid-liquid Equilibria of Systems Present in Biofuels Production. *Fluid Phase Equilibria,* 483**,** 153-164.

Salam, M. A., Taher, A., Samy, M. & Mohamed, K. 2021. The Effect of Different Dimensionality Reduction Techniques on Machine Learning Overfitting Problem. *International Journal of Advanced Computer Science and Applications,* 12.

Sinnott, R. K. 2005. *Coulson & Richardson's Chemical Engineering Vol. 6, Chemical Engineering Design,* Amsterdam, Elsevier Butterworth-Heimann.

StackExchange.com 2011. How to Choose the Number of Hidden Layers and Nodes in a Feedforward Neural Network?

Steurtewagen, B. & Van den Poel, D. 2020. Machine Learning Refinery Sensor Data to Predict Catalyst Saturation Levels. *Computers & Chemical Engineering,* 134.

Stewart, M. 2011. Gas Sweetening and Processing Field Manual. Waltham, MA: Gulf Professional Pub.

Talebjedi, B., Laukkanen, T., Holmberg, H., Vakkilainen, E. & Syri, S. 2021. Energy Simulation and Variable Analysis of Refining Process in Thermo-mechanical Pulp Mill

Using Machine Learning Approach. *Mathematical and Computer Modelling of Dynamical Systems,* 27**,** 562-585.

Thebelt, A., Wiebe, J., Kronqvist, J., Tsay, C. & Misener, R. 2022. Maximizing Information from Chemical Engineering Data sets: Applications to Machine Learning. *Chemical Engineering Science,* 252.

Theodore, L. & Ricci, F. 2010. *Mass Transfer Operations for the Practicing Engineer*, John Wiley & Sons, Inc.

Theodoridis, S. 2015. Machine Learning: A Bayesian and Optimization Perspective. Kent: Elsevier Science & Technology.

Venkatasubramanian, V. 2019. The Promise of Artificial Intelligence in Chemical Engineering: Is It Here, Finally? *AIChE Journal,* 65**,** 466-478.

Verma, Y. 2022. When Does a Neural Network Fail to Converge? Available from: https://analyticsindiamag.com/when-does-a-neural-network-fail-to-converge/ [Accessed 4.14. 2023].

Wang, J. 2022. An Intuitive Tutorial to Gaussian Processes Regression. Available: arXiv:2009.10862v4.

Wang, K., Xu, H., Yang, C. & Qiu, T. 2021. Machine Learning-based Ionic Liquids Design and Process Simulation for CO_2 Separation from Flue Gas. *Green Energy & Environment,* 6**,** 432-443.

Williams, B. A. & Cremaschi, S. 2019. Surrogate Model Selection for Design Space Approximation And Surrogate based Optimization. 47**,** 353-358.

Winter, B., Winter, C., Esper, T., Schilling, J. & Bardow, A. 2022. SPT-NRTL: A Physics-guided Machine Learning Model to Predict Thermodynamically Consistent Activity Coefficients.

Wuest, T., Weimer, D., Irgens, C. & Thoben, K.-D. 2016. Machine Learning in Manufacturing: Advantages, Challenges, and Applications. *Production & Manufacturing Research,* 4**,** 23-45.

Yalamanchi, K. K., van Oudenhoven, V. C. O., Tutino, F., Monge-Palacios, M., Alshehri, A., Gao, X. & Sarathy, S. M. 2019. Machine Learning To Predict Standard Enthalpy of Formation of Hydrocarbons. *J Phys Chem A,* 123**,** 8305-8313.

Yousefzadeh, R. & Cao, X. 2022. To What Extent Should we Trust AI Models When They Extrapolate? *Association for Computing Machinery***,** 17.

Yu, G., Wei, Z., Chen, K., Guo, R. & Lei, Z. 2022a. Predictive Molecular Thermodynamic Models for Ionic Liquids. *AIChE Journal,* 68.

Yu, T., Chen, K., Lin, N., Meštrović, R., Oliveira, T. A., Cen, F. & Yin, H.-M. 2022b. Decision Tree: Basic Theory, Algorithm Formulation and Implementation. 147.

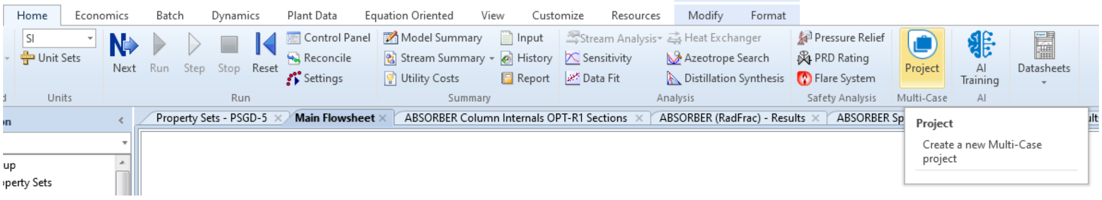
Zhang, Y. & Chen, C.-C. 2010. Thermodynamic Modeling for CO_2 Absorption in Aqueous MDEA Solution with Electrolyte NRTL Model. *Industrial & Engineering Chemistry Research,* 50**,** 163-175.

Zhou, Z.-H. 2021. *Machine Learning*, Springer Singapore.

Zhu, W., Liu, X., Hou, X., Hu, J. & Diao, Z. 2020. Application of Machine Learning to Process Simulation of N-pentane Cracking to Produce Ethylene and Propene. *Chinese Journal of Chemical Engineering,* 28**,** 1832-1839.

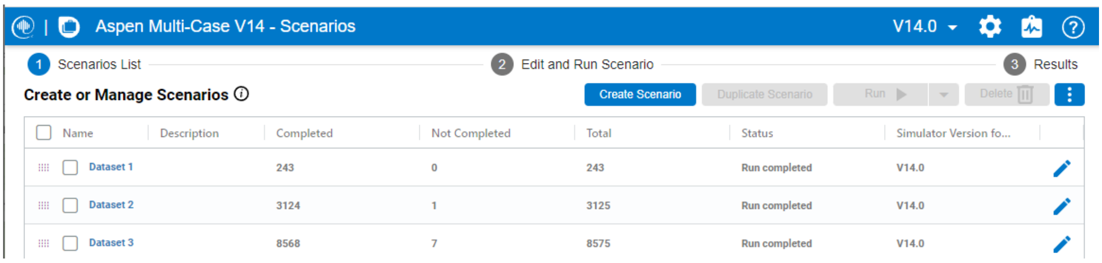
Appendix I – Aspen Plus Multi-Case workflow

The simulation that data is generated with is opened in Aspen Plus. Multi-Case tool is accessed from the Home ribbon. The project launches in the web browser. Multi-Case can also be accessed from the browser's address bar.

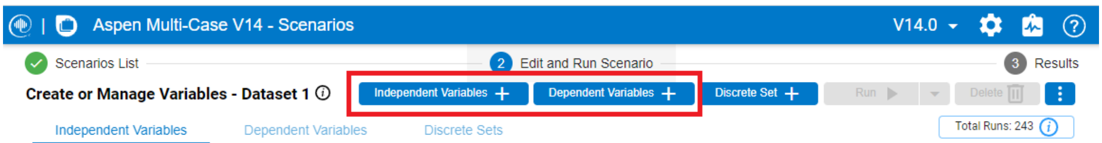


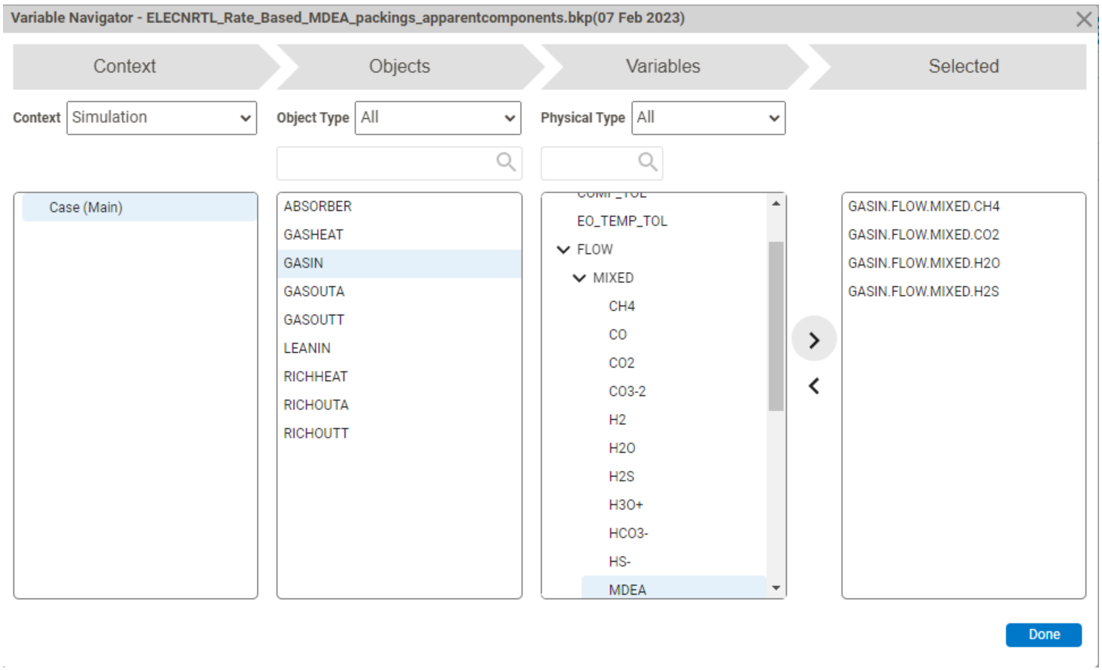
Case study project

A scenario is automatically created and visible in the Scenarios List view. The Edit and Run Scenario view can be opened by double-clicking the scenario.

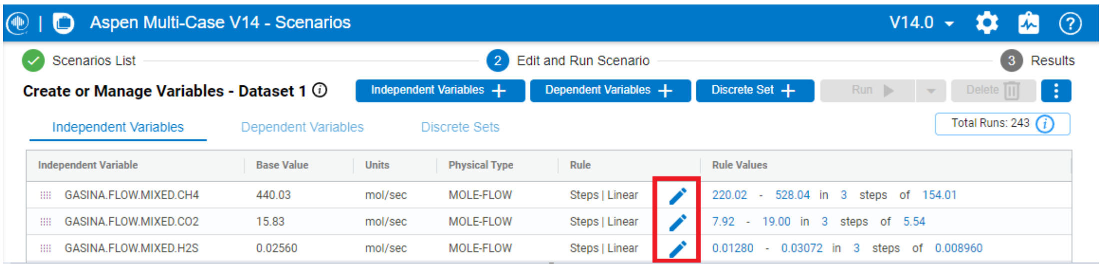


In the Edit Scenario view, the independent and dependent variables are chosen. The model parameters that are varied with Multi-Case are chosen as independent variables with the variable navigator, and the results that are to be recorded are selected as dependent variables.





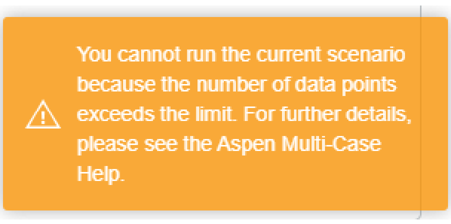
The variable ranges and steps are automatically set, but both can be edited. The variable table shows the base value, units, and variable type from the base simulation.



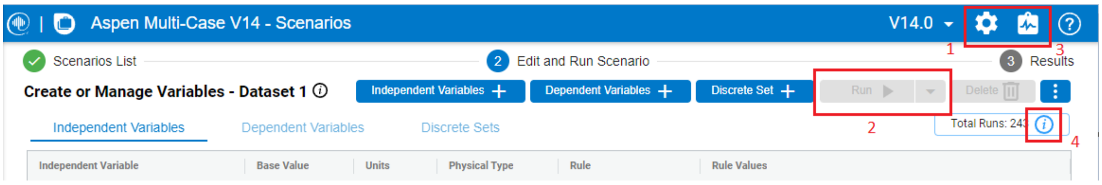
It should be noted here that Multi-Case executes the simulation with all possible combinations of the values of the independent variables, i.e., the simulation is run

$$no.of\ steps_{var\ 1} \cdot no.of\ steps_{var\ 2} \cdot \ldots \cdot no.of\ steps_{var\ n}$$

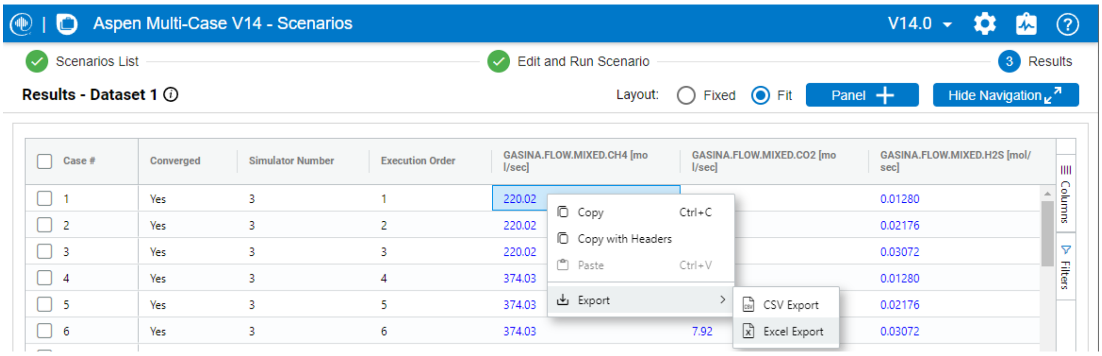
times, when discrete sets are not used (discrete sets were not used in generating data for this thesis, more information on discrete sets can be found in Multi-Case Help). The maximum number of data rows (i.e., simulation runs) is 2 000 000. If that limit is exceeded, following error message shows:



Before running, the maximum number of simulations computed parallelly can be adjusted from the settings (1). While generating data for this thesis, the four cores of the computer were used to compute four simulations parallelly. When the independent and dependent variables have been chosen and the Multi-Case settings are adjusted, the scenario is run (2). While the scenario is running, the Running monitor (3) that shows the number of completed simulations, number of simulations left, number of unconverged simulations as well as time elapsed in running the scenario and the estimated time to completion. When the scenario is completed, the number of converged and unconverged simulations along with the time elapsed in running the scenario can be viewed from (4).



Scenario results table can be viewed from the Results page. The table can be edited and filtered in the Multi-Case tool, and the results table can be exported in different formats by right-clicking any cell in the results table.



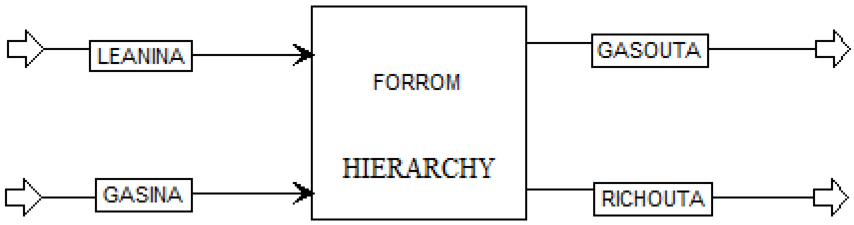
Remarks

If the component specific flows are varied as independent variables without keeping the total molar flow constant, the total flow of the stream must be left empty in the simulation. Otherwise, the total molar flow stays constant despite the component specific molar flows are changing in the input, and strange results are obtained.

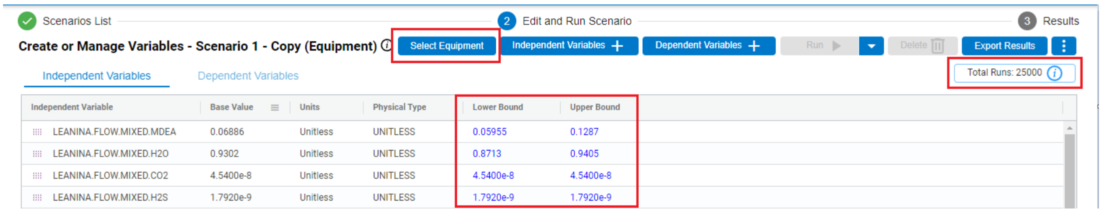
The scenario results are in such units that simulation results are by default in the unit set of the simulation, i.e., the resulting molar flows may be in kmol/sec (SI units) even if property set of kmol/hr etc. has been set in the simulation.

Reduced Order Project

The reduced order project automatically offers independent and dependent variables for the data. For the project to collect the data in a form suitable for the AI Model Builder, the base model must include total mass or molar flows and mass or mole fractions for the components. If the base model contains multiple blocks, all blocks and streams except for the outermost input and output streams must be set into a hierarchy as pictured below. This way, the reduced order project recognizes the correct streams to collect data from.



In Multi-Case, the equipment to be modelled is selected, and independent and dependent variables should be chosen automatically. Additional dependent variables can also be added.



The minimum and maximum values for the independent variables can be edited similarly as in a Case Study project, but instead of adjusting the number of steps for each variable, the total number of cases (data points) created is specified. The data generation is run similarly as a Case Study project, but the results are exported as a .json file. When exporting the results, “Only converged cases” can be selected.

Appendix II – Aspen Plus First Principles Driven Hybrid Modelling workflow

Before proceeding to the First Principles Driven Modelling tool, the data exported from Multi-Case requires some formatting. Aspen's data template requires timestamps for each data row, which are added to the Multi-Case data file (form: mm/dd/yyyy hh:mm) before loading data into the template. Additionally, unconverged runs are filtered out of the data as outliers before importing. The tag names must not contain brackets, and unit of measure is displayed on a separate row.

	A	B	C	F	G
1	Timestamp	Case #	Converged	GASIN.FLOW.MIXED.CH4	GASIN.FLOW.MIXED.CO2
2				mol/sec	mol/sec
3	01/01/2023 0:00	1	Yes	220.02	7.92

Plant Data Template -Excel

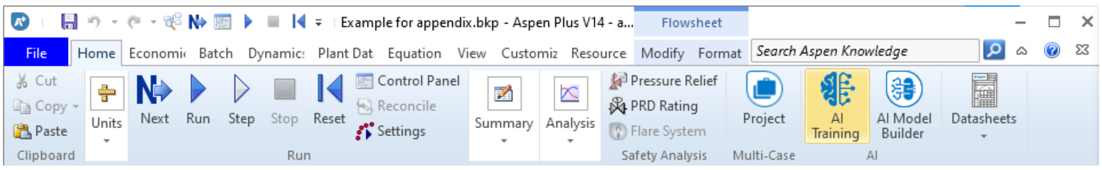
Simulated data is first exported as a .xlsx file from the Aspen Plus Multi-Case tool. The AI Training ribbon in Aspen Plus provides an .xlsm -template for converting the data into a form that is readable for the software. In the LoadData sheet of the template, the cell locations of the first tag name (variable name), unit of measure and timestamp are typed, and the data file is located from three dots.

Location of first tag name	C1
Location of first tag description (Optional)	
Location of first units of measure (Optional)	C4
Location of first dataset's time	A5
Location of input file (*.xlsx)	C:\Temp\Absorption\Data sets\DS 4 f ... hodel builder.xlsx
Input orientation	Timestamps as Rows

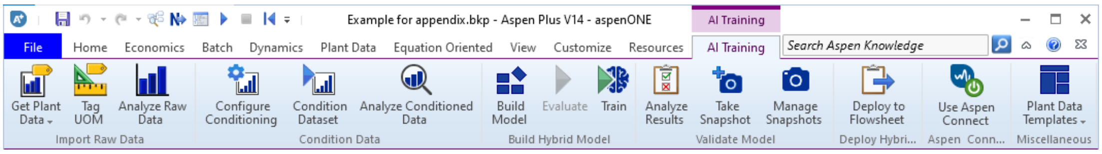
Load Data

AI Training

The First Principles Driven Modelling tool can be accessed from the Home ribbon.

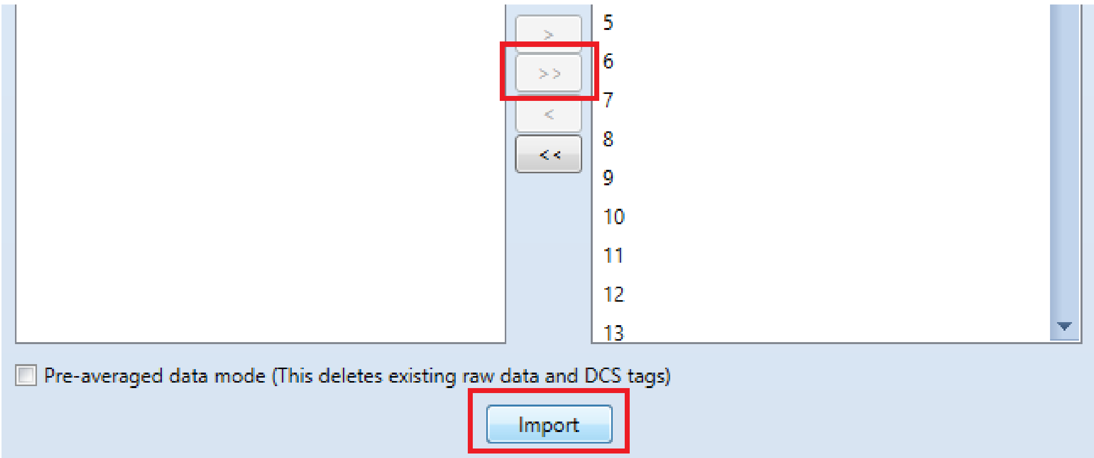


When creating a first principles driven hybrid model, the user proceeds from left to right in the AI Training ribbon. The PlantDataTemplate can be found on the right.



Get Plant Data

After the simulated data has been loaded into Aspen's template, the template is imported to Aspen Plus via Get Plant Data (in this case, Excel template). All data points are selected and imported.



After plant data is successfully imported, the Import Plant Data window is closed and all units of measure are selected and updated.

Analyze Raw Data, Configure Conditioning & Condition dataset

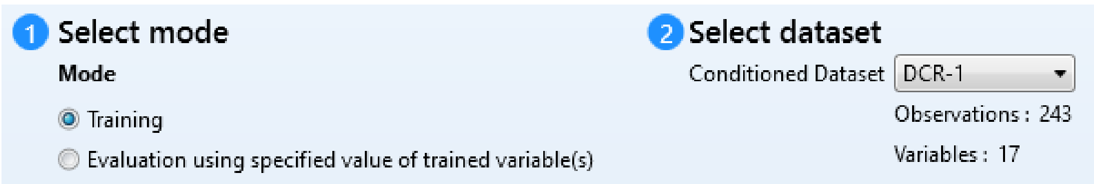
Raw data is inspected by eye for outliers and conditioned with necessary limits for each variable via Condition Plant Data. Since the data in this case is simulated and unconverged runs are removed from the data before importing, there are no clear outliers, and the data conditioning run is done without specifying limits for the variables.

Analyze Conditioned Data

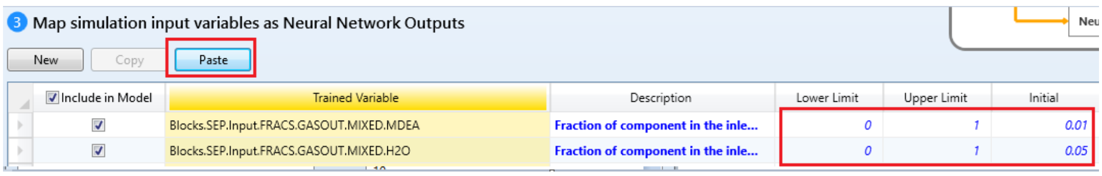
In Analyze Conditioned Data, the imported training data can still be inspected by eye for outliers, and the data conditioning can be run again with new limits for the data if necessary.

Build Model

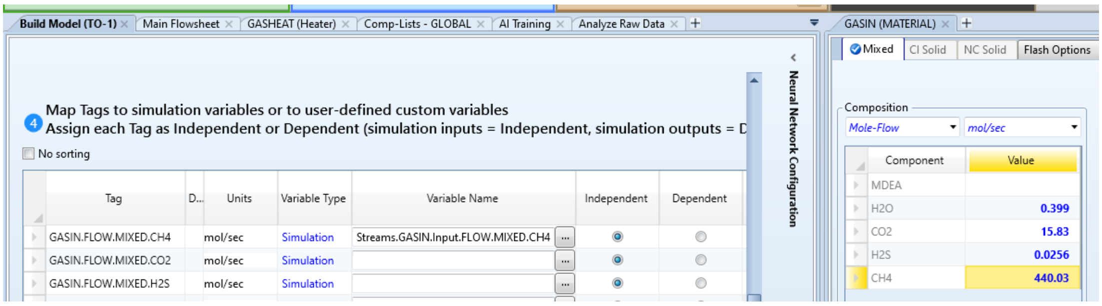
When building the first principles driven hybrid model, the performance of the first principle model can be evaluated with Evaluate mode, where the overall error and component-wise coefficients of determination are calculated. However, in this work simulated data is used to train a simpler unit operation model which would by definition differ significantly from the training data, the evaluation is not considered relevant. Therefore, the hybrid models are trained in the Training mode with the correct conditioned dataset.

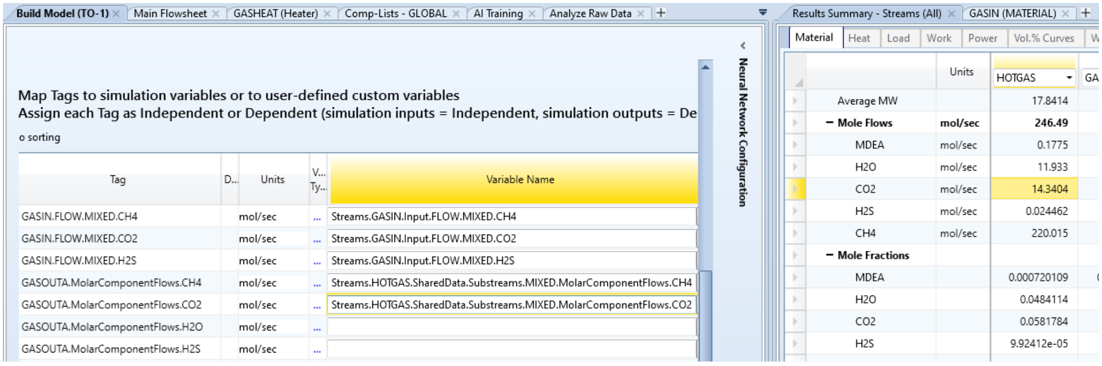


The model parameters to be trained as neural network outputs are copied and pasted (or dragged and dropped) from the stream or block specifications to the “Map simulation input variables as NN outputs” field. Relevant lower and upper limits are set, and initial guess can be edited. The trained hybrid model chooses such values for the NN output variables, that the simulation output variables mapped as dependent variables are predicted according to the training data.

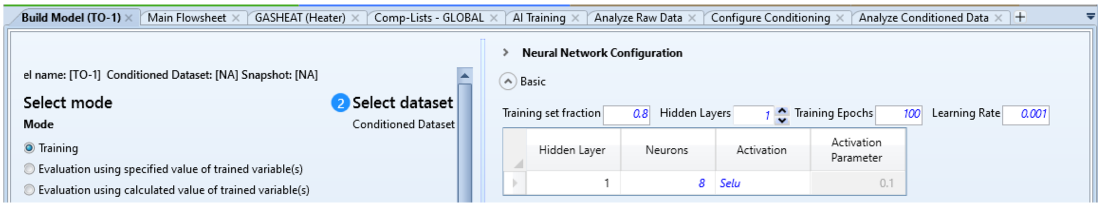


Model parameters and results are mapped as independent or dependent variables in “Map Tags to simulation variables”. The correct model parameters are easiest to select by copying and pasting/dragging and dropping from the stream/block specifications or results (independent variables/inputs must be copied from specifications, dependent variables/outputs from results).



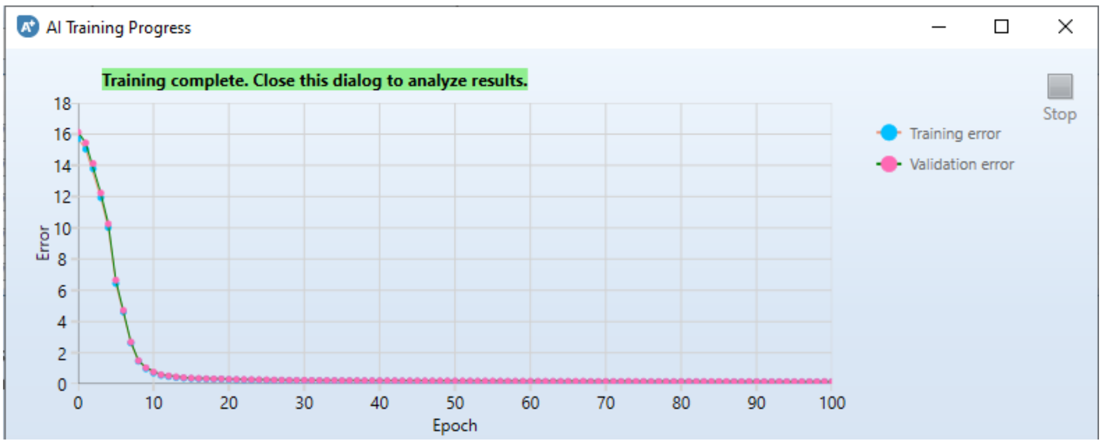


Neural network configuration can be adjusted as wished. More advanced configuration properties are available as well.



Train

When all variables are mapped and neural network configuration is set, the model training run is started. AI Training Progress window appears to allow for monitoring the training.



Analyze Results

Component-wise coefficients of determination, iteration history, variable importance plot and custom plots can be viewed.

Snapshots and Deploying to Flowsheet

To save current model training results, snapshots can be taken. If the neural network is deployed to the flowsheet, a snapshot is created automatically. In manage snapshots, old NN configurations can be restored. Multiple training objects can also be created in the sidebar's AI Training folder, so that multiple NN configurations can exist simultaneously.

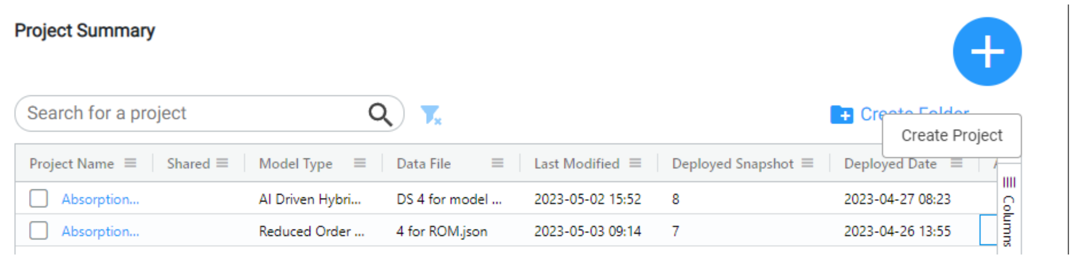
Remarks

To avoid error message "String DateTime not recognized", the user must edit the FirstLastTimeStamps.txt file into the correct DateTime form. The file can be found in the *simulation_name*_PD/RawData folder when using a .bkp file (PD folder is automatically created in the same folder as the simulation). Aspen Plus must be restarted to apply changes in the file.

At the time of publishing this workflow, an emergency patch to avoid the error message automatically is in the works by AspenTech, but not yet available.

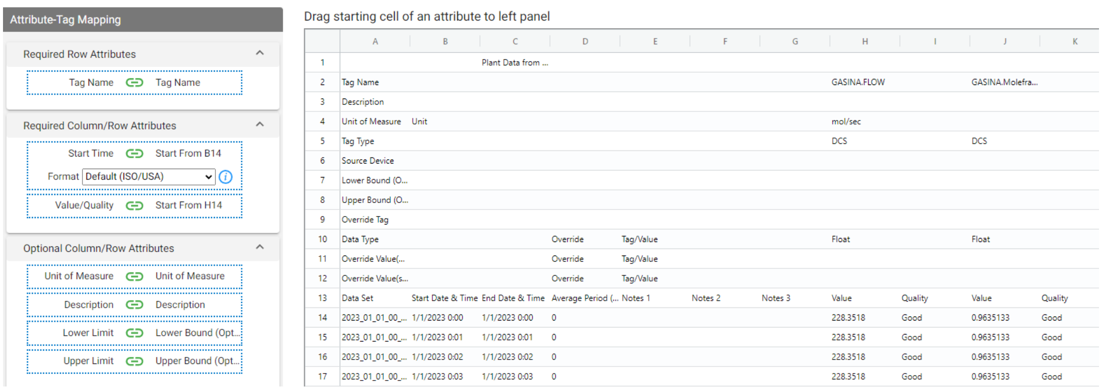
Appendix III – Aspen Plus AI Driven Hybrid Modelling workflow

Aspen AI Model Builder Launcher is used to access AI Model Builder (AIMB) in the web browser. A user account and two-factor authentication using Google Authenticator are required. After login and authentication, the Project summary window appears. A new project is created from the plus sign. The project is named, and the suggested root location for the project is accepted. The purpose for the hybrid model project is chosen, in this case AI Driven Hybrid Models for Engineering.



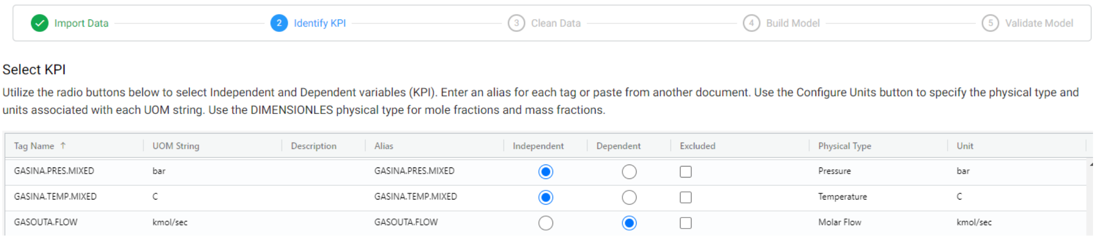
Import data

The data imported to AIMB must contain the input and output stream temperatures, pressures, total molar or mass flows and stream compositions as molar or mass fractions. The data is imported as a .csv file, and correct format can be achieved by saving the PlantDataTemplate.xlsm file introduced in Appendix II as .csv. Alternatively, the .csv template can be downloaded from AIMB. If the imported data is not in the correct format, the Import Wizard can be used to find the correct cells.



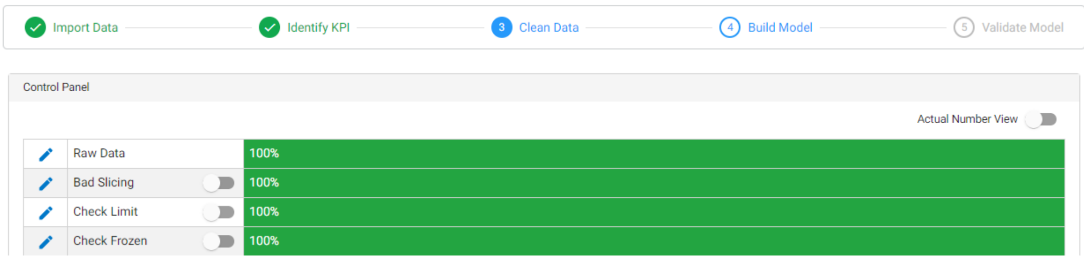
Identify KPI

Model parameters from the data are selected as independent or dependent variables, physical types and units of measure are checked and corrected, if the variables are not automatically mapped correctly. After editing variables, click Update button to save changes.



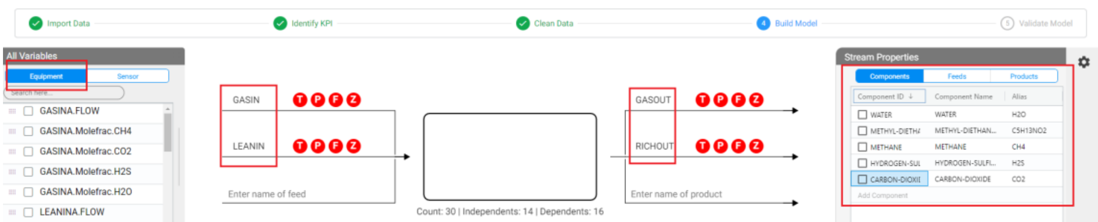
Clean Data

Simulated data should not need cleaning, if unconverged cases have been filtered out. Data cleaning can also be used to find the unconverged cases by turning on Bad Slicing. Other data cleaning methods can also be used, and their settings edited as needed.

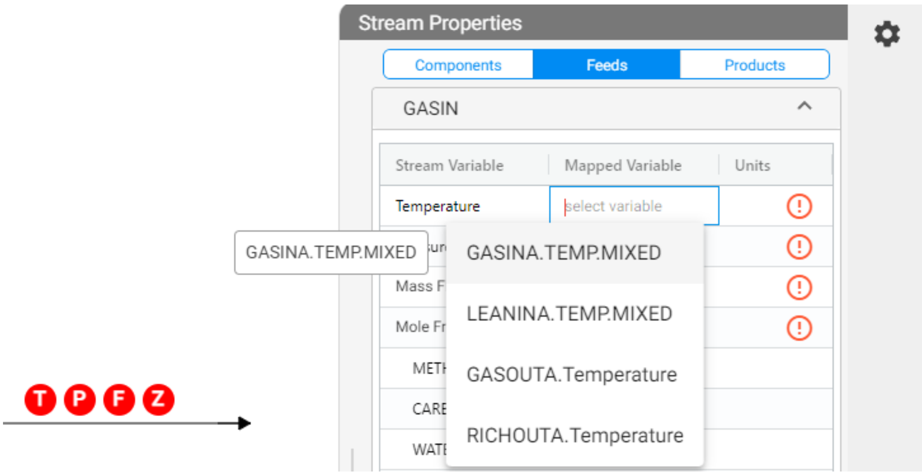


Build Model

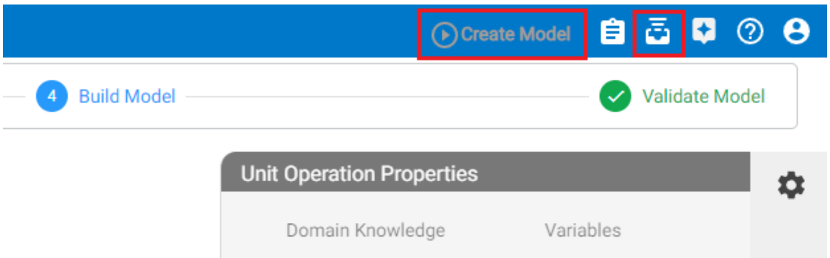
In Build Model page, the correct model is selected (in this work, Equipment model). Feed and product streams are created and named. Stream properties can be edited by clicking the streams. Components present in the process are added, if not automatically recognized.



In stream properties, the correct variables from the data are mapped for the streams, if not done automatically. AIMB suggests variables with the correct physical type.

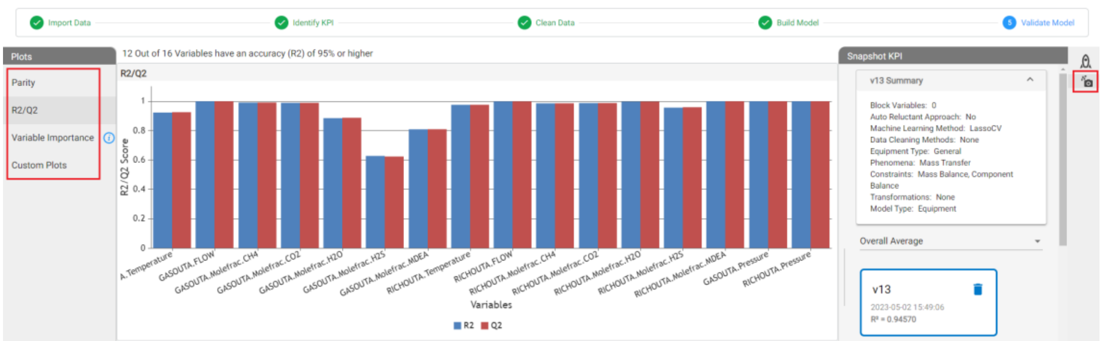


By clicking the white model box in the middle of the screen, the Unit Operation Properties become visible. Constraints, machine learning methods, transformations, etc. can be adjusted. When all variables are mapped and ML method selected, the model is created. While model creating is in process, the progress can be monitored from the Jobs Queue (marked).



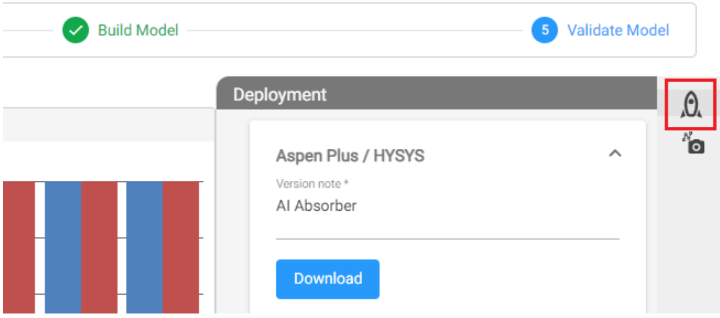
Validate Model

The parity plots (observed vs predicted values for the variables), variable accuracy and predictability (R^2 and Q^2), variable importance plot as well as custom plots can be viewed in Validate Model page. If the model is not accurate enough, the user can return to Build Model page and create a new snapshot of the model. The snapshots can be managed from the Snapshot KPI menu (marked on the right).

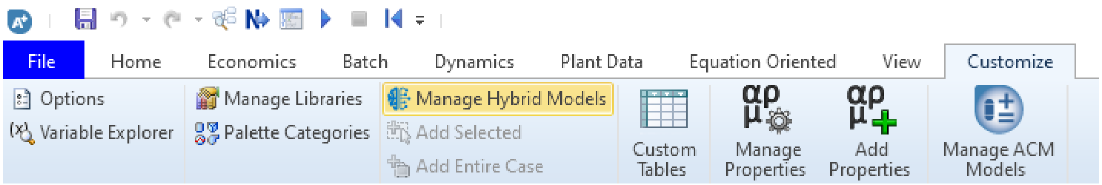


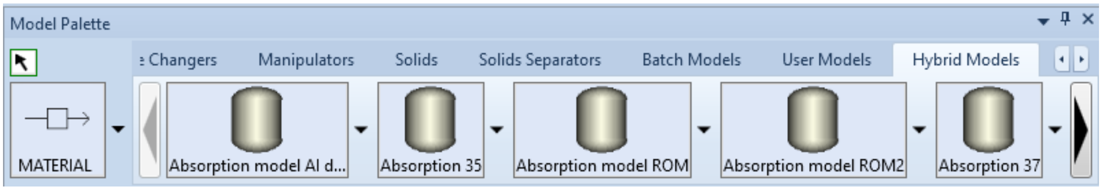
Model Deployment

When a suitable model configuration is found, it can be downloaded as a .ATHM file from the Deployment menu (marked on the right).

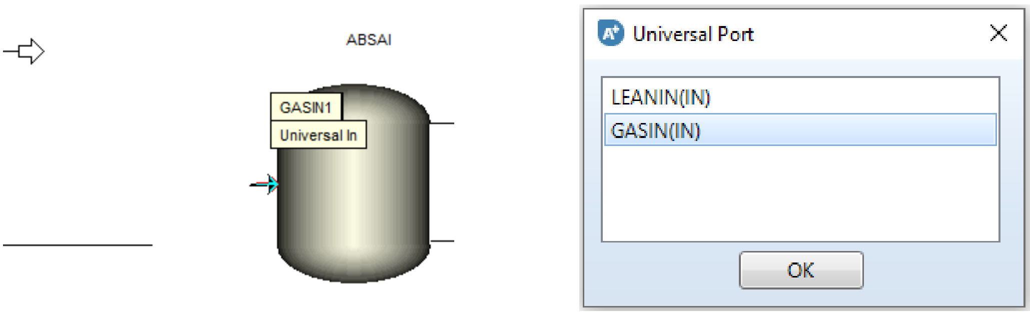


In Aspen Plus, the AI driven model can be imported from Manage Hybrid Models in the Customize ribbon. The model appears in the Hybrid Models of the Model Palette.





When attaching streams to the AI driven model, the user must select which stream of the hybrid model is attached. After streams are specified, the model can be run.



Appendix IV – Aspen Plus Reduced Order Modelling workflow

Aspen AI Model Builder (AIMB) user interface is similar when creating reduced order models and AI driven models, thus pictures of the user interface can be found in Appendix III. The reduced order models in AIMB are created in a similar manner as the AI driven models, but "Reduced Order Equipment Models for Engineering" is chosen as model purpose.

Import Data

The .json file exported from Multi-Case is imported without any additional formatting, given the data includes the input and output stream temperatures, pressures, total molar or mass flows and stream compositions as molar or mass fractions.

Manage Variables

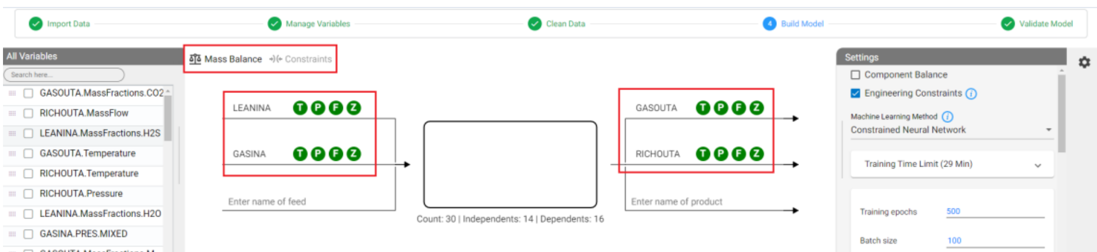
If the training data is formatted correctly, AIMB should recognize the independent and dependent variables as well as the physical types and units of measure automatically. However, variables can still be corrected.

Clean Data

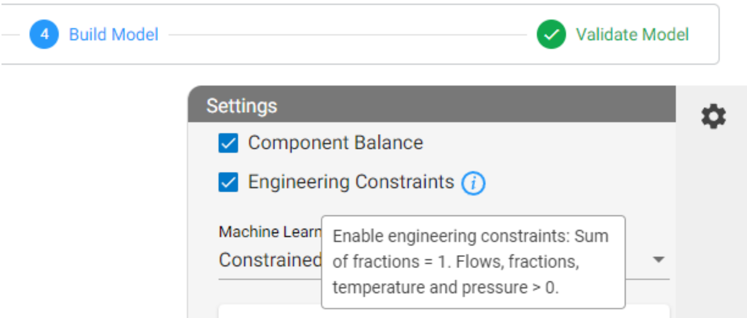
Since reduced order models utilize simulated data created with Multi-Case reduced order project, the data should not contain any outliers and data cleaning is not necessary.

Build Model

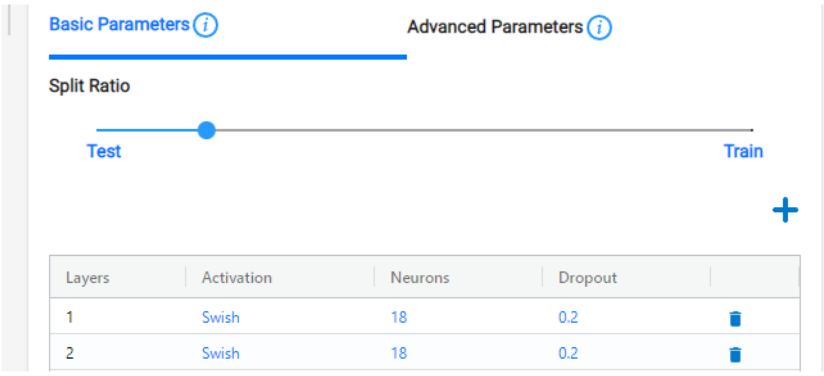
The feed and product streams should be created automatically, given the data is in correct format. Also the variables of the data should be mapped automatically, but corrections can be done manually. The user can toggle between Mass Balance view and Constraints view to adjust the machine learning methods and summation or inequality constraints, respectively. In this work, user-defined constraints are not used.



AIMB reduced order model settings offer Lasso, Lasso CV and Auto-Reluctant linear methods as well as Constrained Neural Network as machine learning methods. In this work, the latter is used. Although component balance is not recommended for use in reactive systems, it is employed in this work since only apparent components are visible in the training data. Engineering constraints are used to force sums of fractions to 1, and keep flows, fractions, temperatures, and pressures physical.



Training time limit is set to suitable time. The number of training epochs and batch size (number of data points used in each epoch) can be adjusted but are left to default values in this work. The neural network settings include training data split ratio, number of hidden layers and neurons in the layers as well as hidden layer activation function as basic parameters. The advanced parameters include EarlyStopping criteria to define how quickly training should stop if defined progress is not achieved. The model is created similarly as the AI driven models.



Validate Model

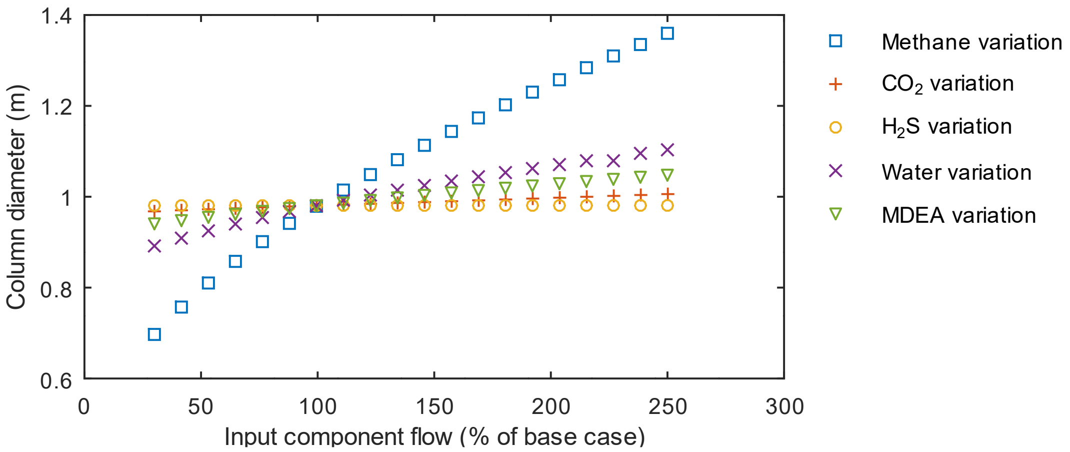
Model validation is similar as in AI driven modelling.

Deployment

The model is downloaded from AIMB and imported to Aspen Plus similarly as the AI driven models. Also attaching streams to the reduced order model works similarly as the AI driven models.

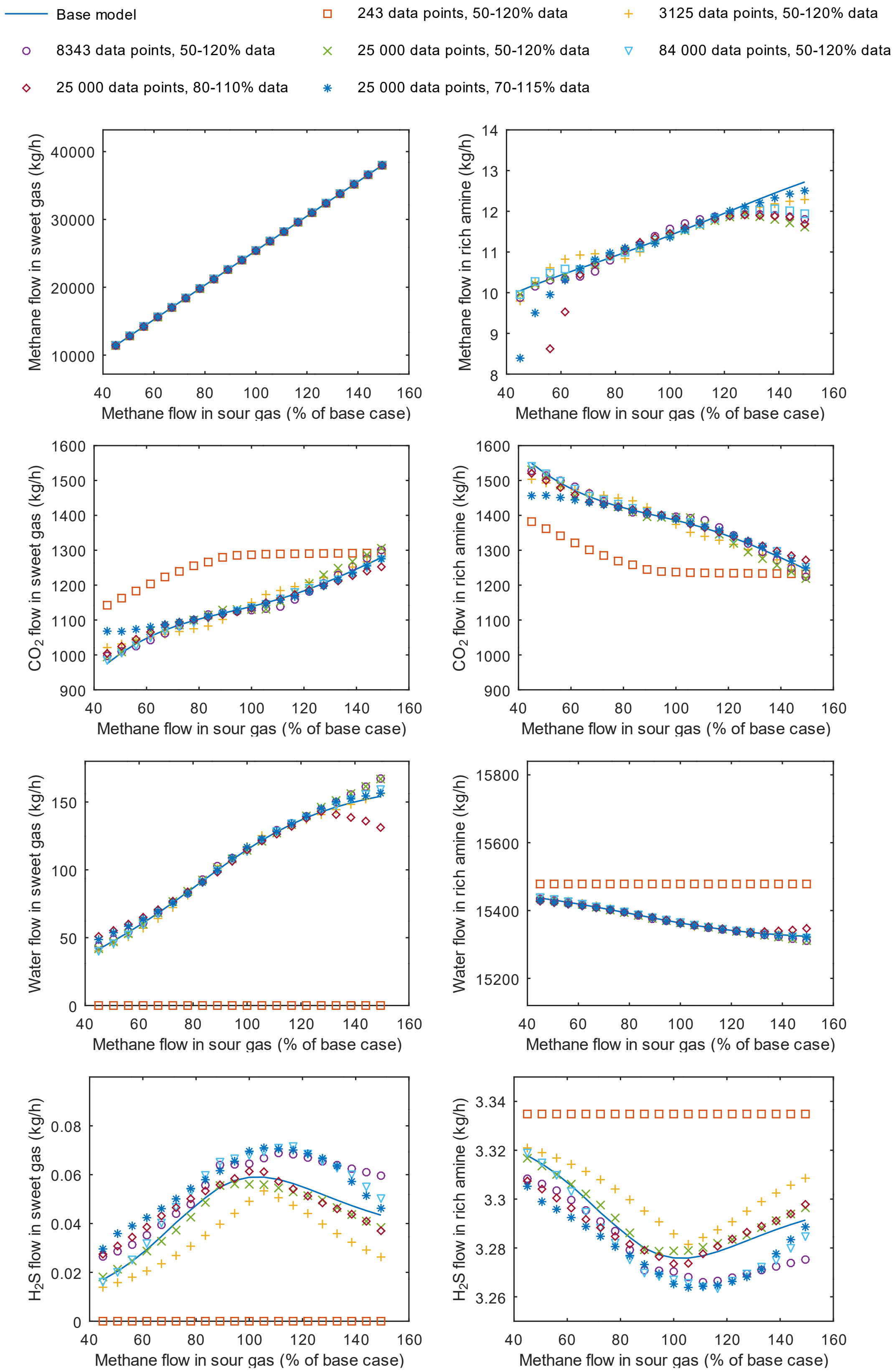
Appendix V – Amine absorber column diameter results

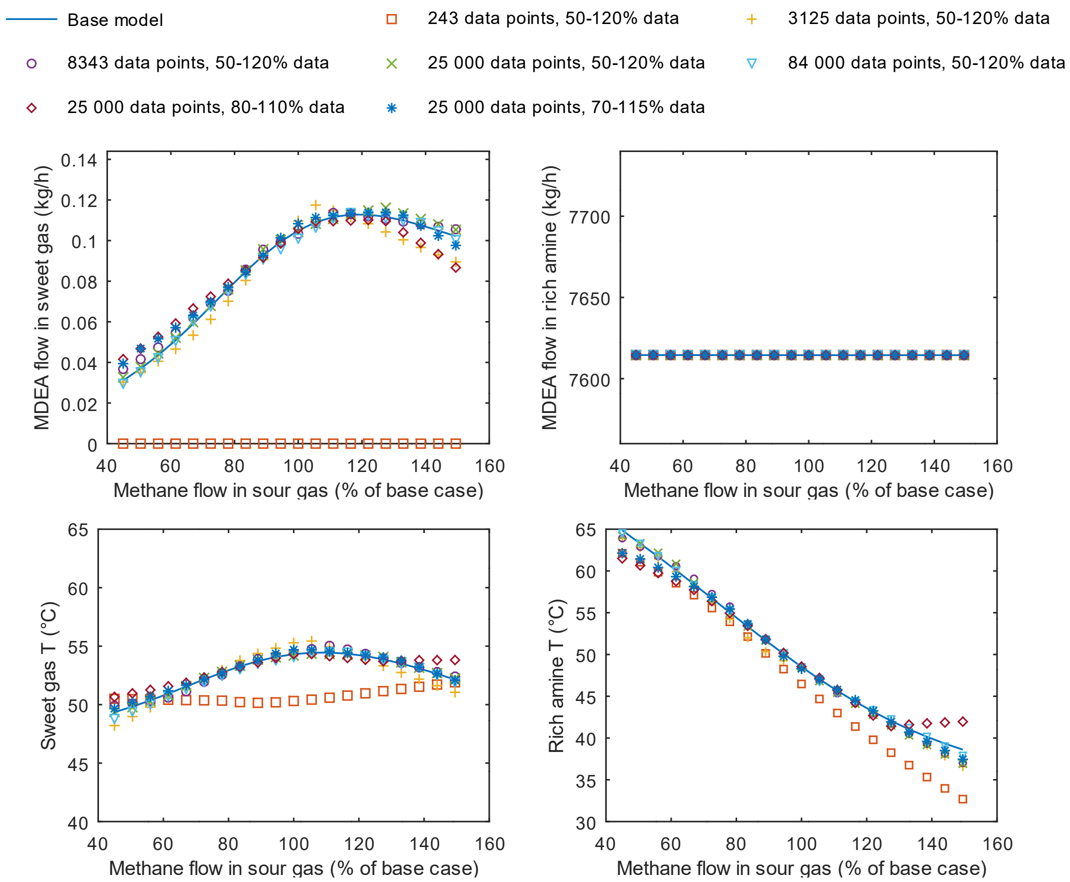
In this work, the design mode was used to calculate the absorber column diameter to avoid stage drying and flooding in the validation cases and training data. Other component flow variations only affected the diameters by a few centimeters, and even methane input flow's effect was ca. 30 cm. The diameter did not drastically affect the column performance. Calculated column diameters as functions of the varied input component flows are shown below.



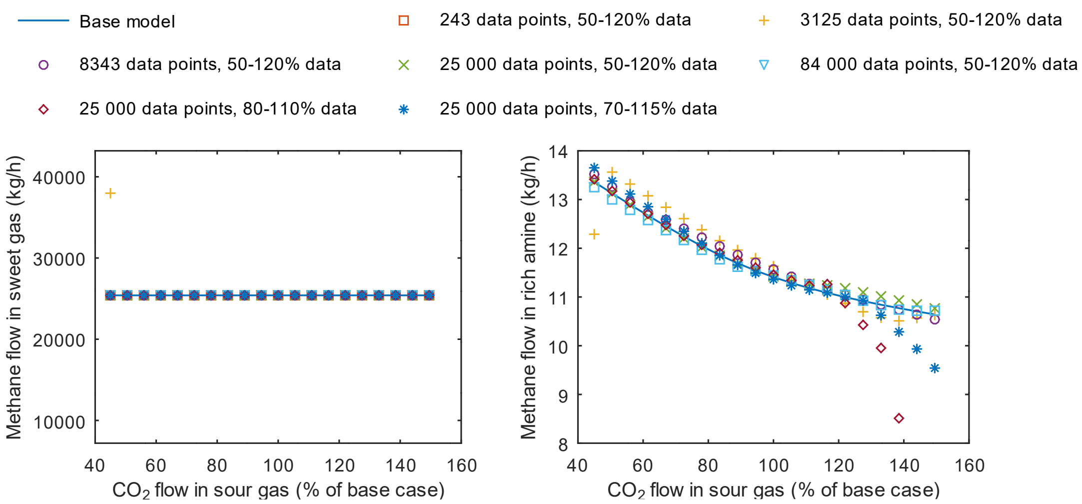
Appendix VI – Additional figures for tests with dataset size and wideness

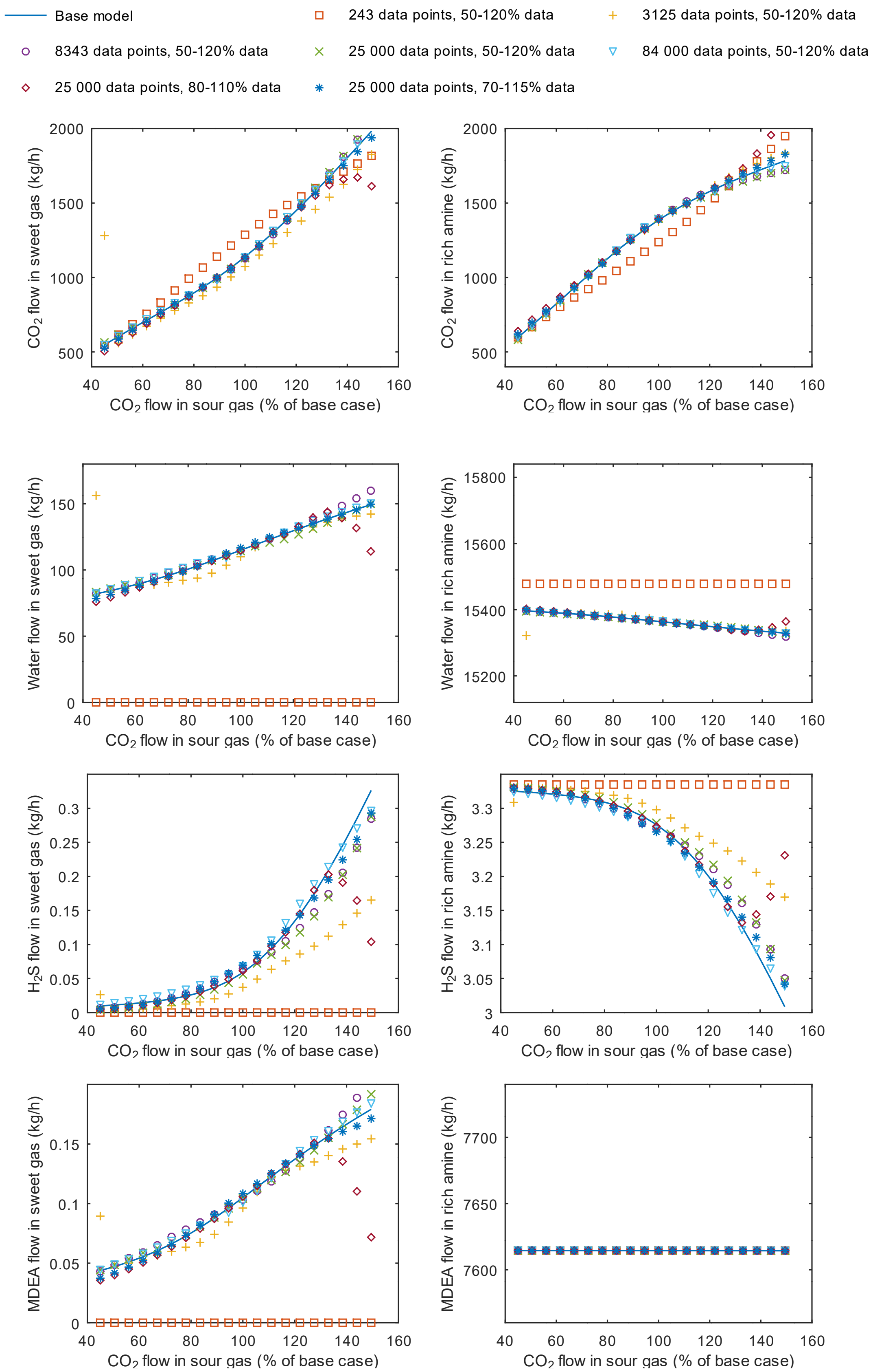
Effect of methane flow variation



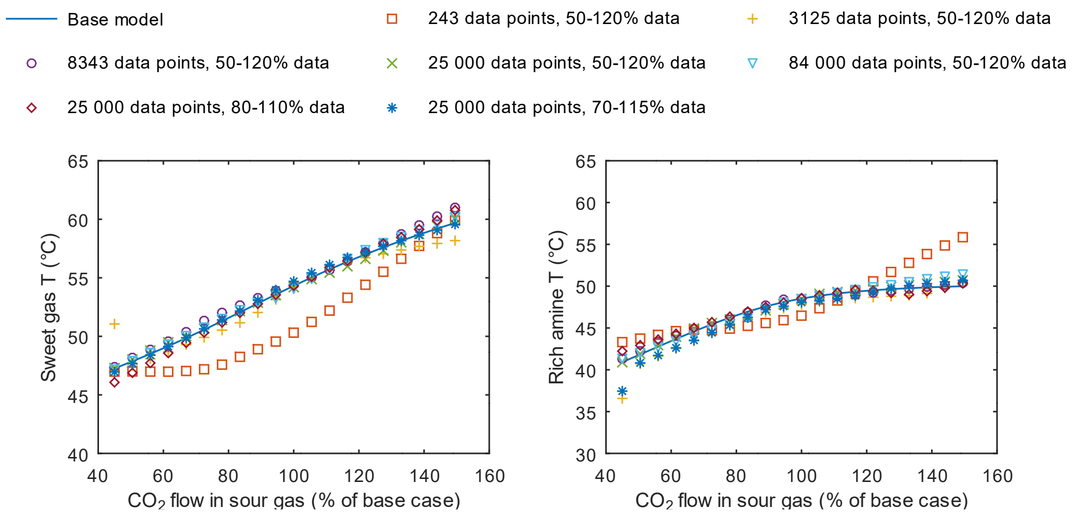


Effect of CO_2 flow variation

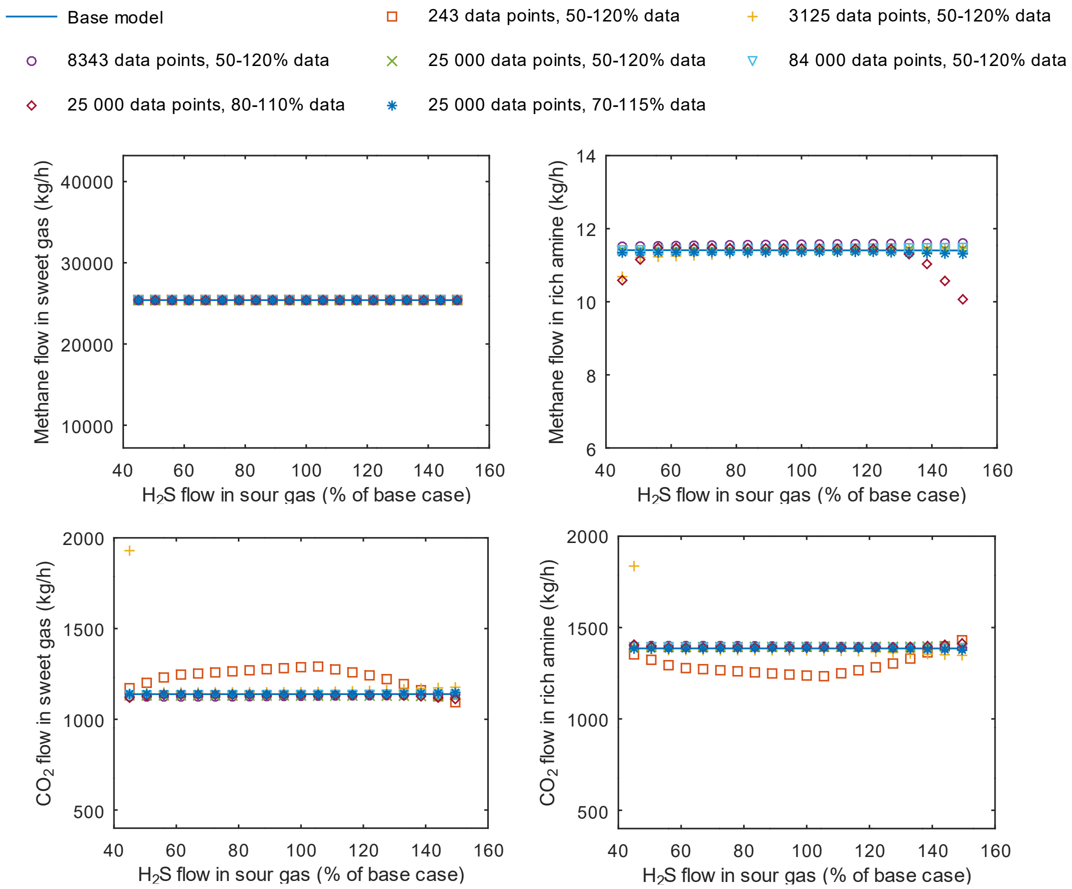


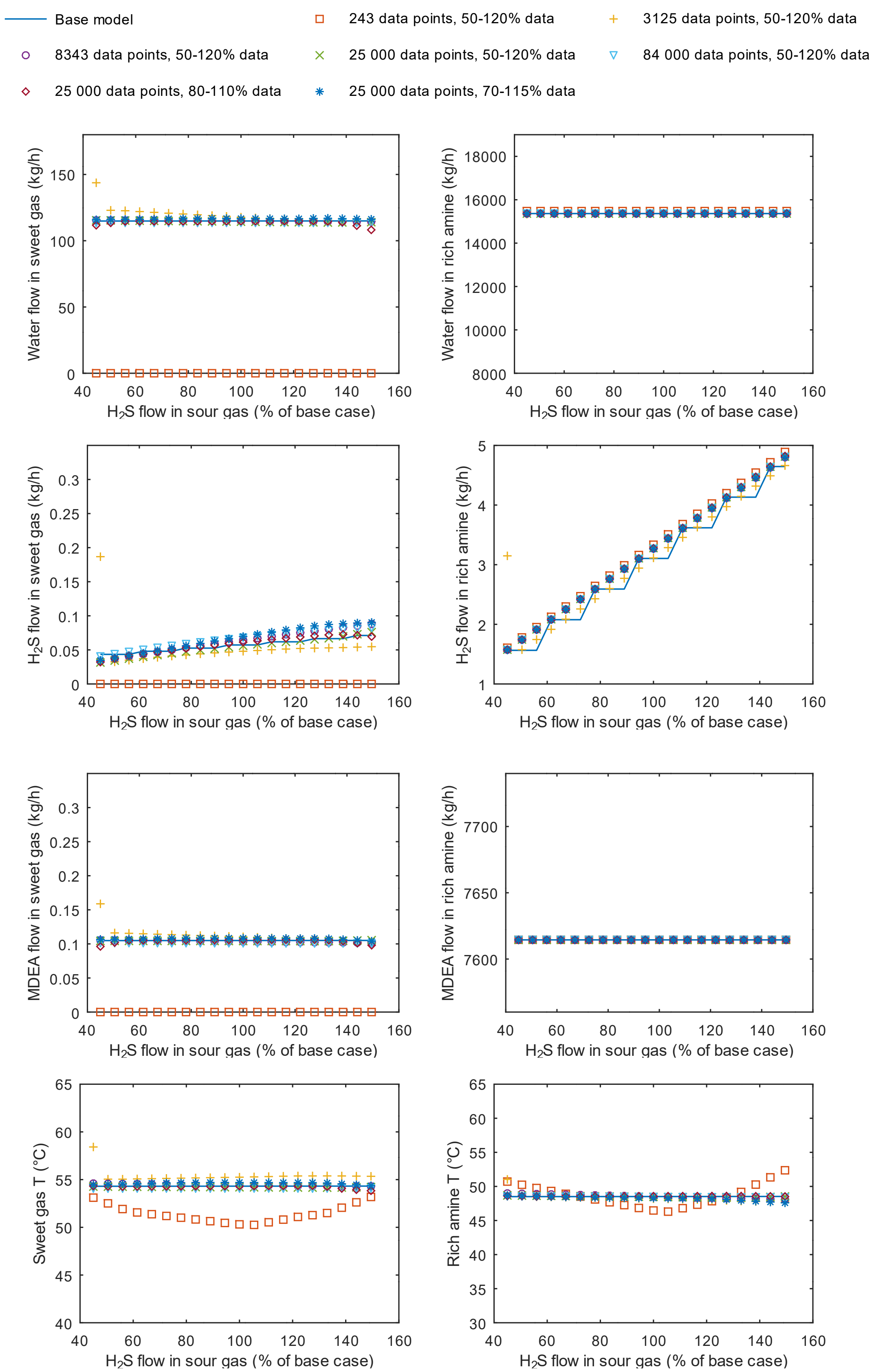


Base model
243 data points, 50-120% data
3125 data points, 50-120% data
8343 data points, 50-120% data
25 000 data points, 50-120% data
84 000 data points, 50-120% data
25 000 data points, 80-110% data
25 000 data points, 70-115% data
CO_2 flow in sweet gas (kg/h)
CO_2 flow in rich amine (kg/h)
Water flow in sweet gas (kg/h)
Water flow in rich amine (kg/h)
H_2S flow in sweet gas (kg/h)
H_2S flow in rich amine (kg/h)
MDEA flow in sweet gas (kg/h)
MDEA flow in rich amine (kg/h)
CO_2 flow in sour gas (% of base case)



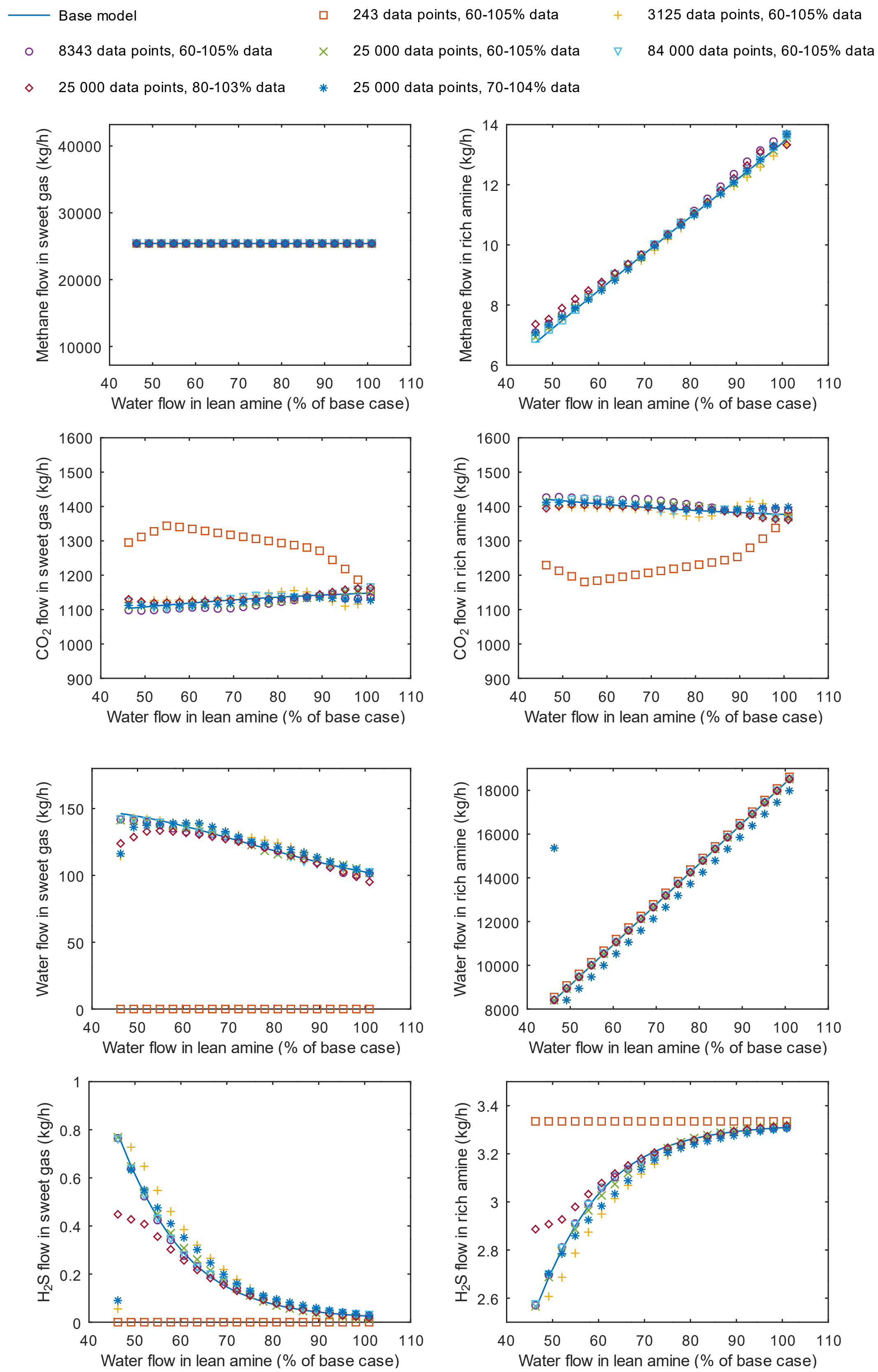
Effect of H_2S flow variation

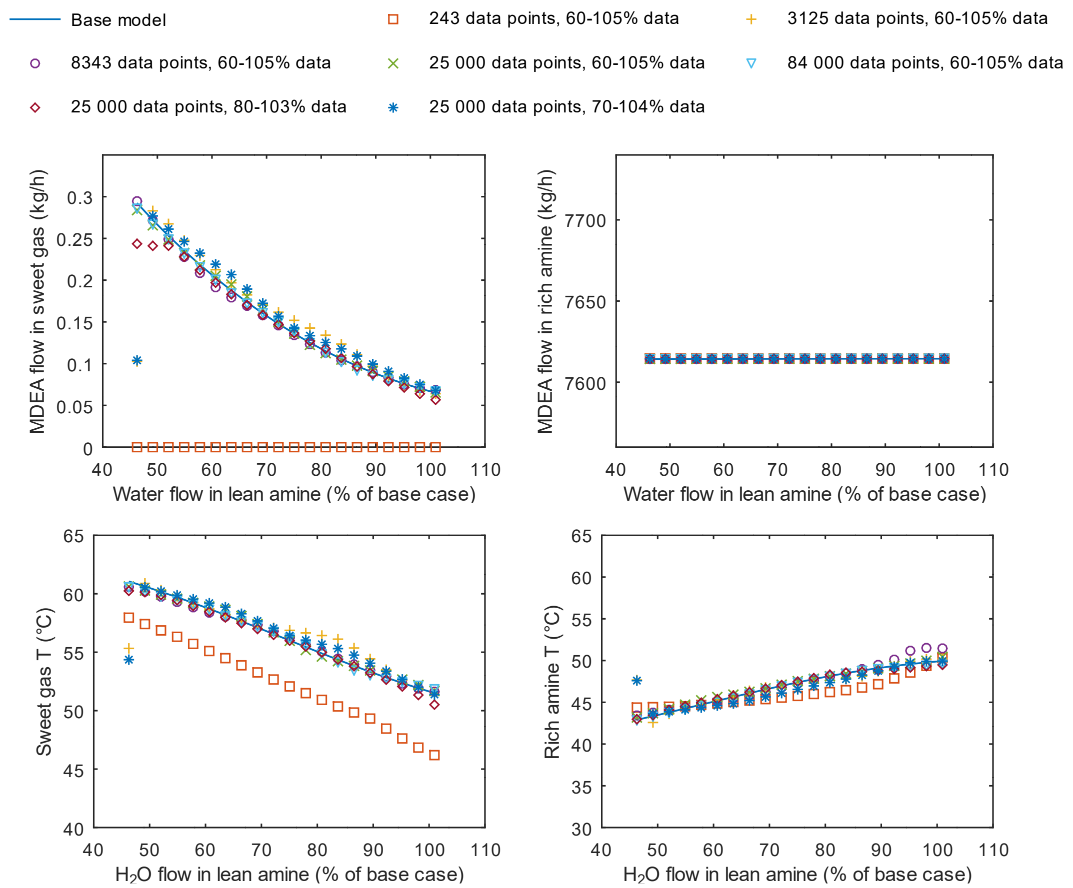




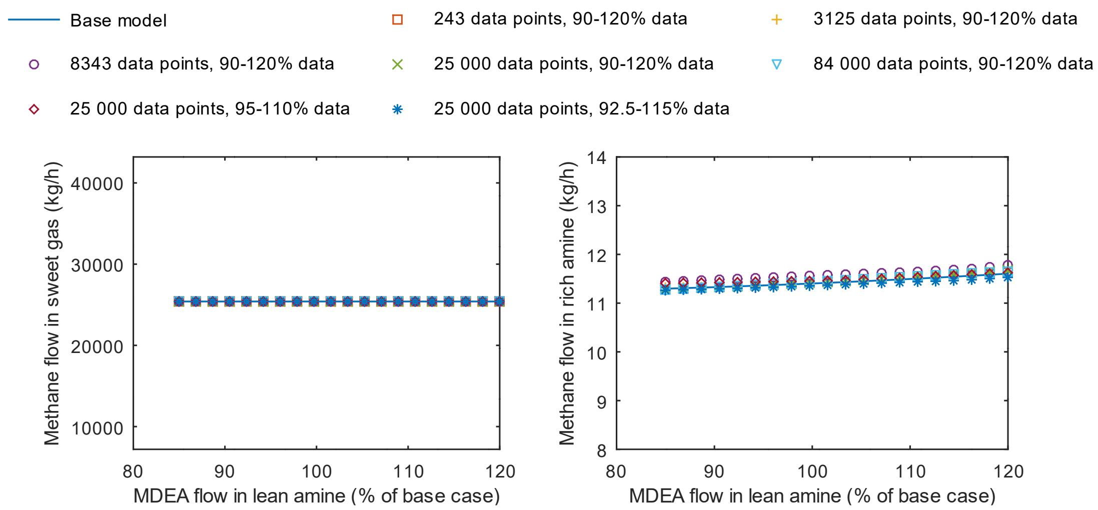
Base model
243 data points, 50-120% data
3125 data points, 50-120% data
8343 data points, 50-120% data
25 000 data points, 50-120% data
84 000 data points, 50-120% data
25 000 data points, 80-110% data
25 000 data points, 70-115% data
Water flow in sweet gas (kg/h)
Water flow in rich amine (kg/h)
H_2S flow in sweet gas (kg/h)
H_2S flow in rich amine (kg/h)
MDEA flow in sweet gas (kg/h)
MDEA flow in rich amine (kg/h)
Sweet gas T (°C)
Rich amine T (°C)
H_2S flow in sour gas (% of base case)

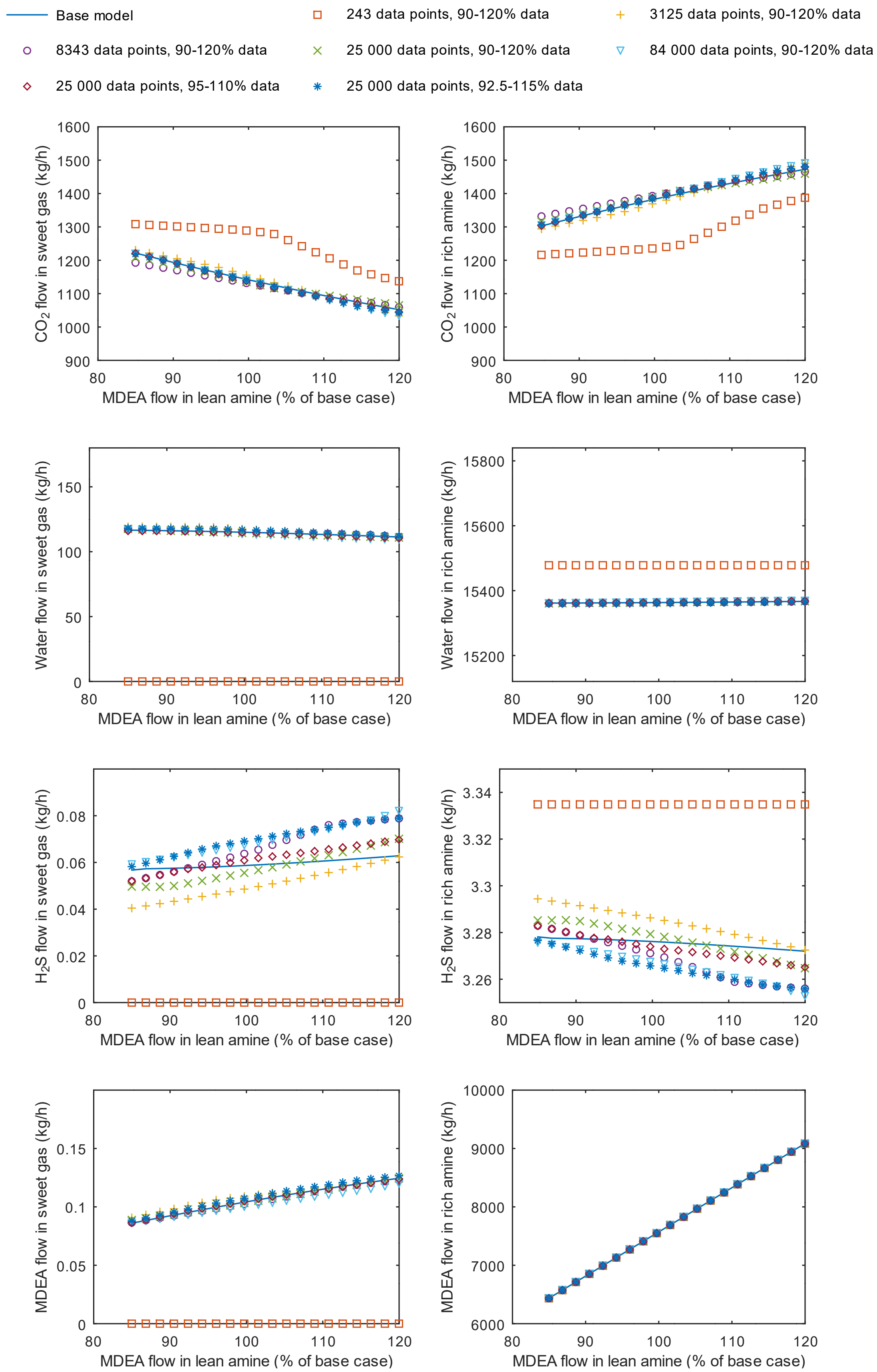
Effect of H_2O flow variation



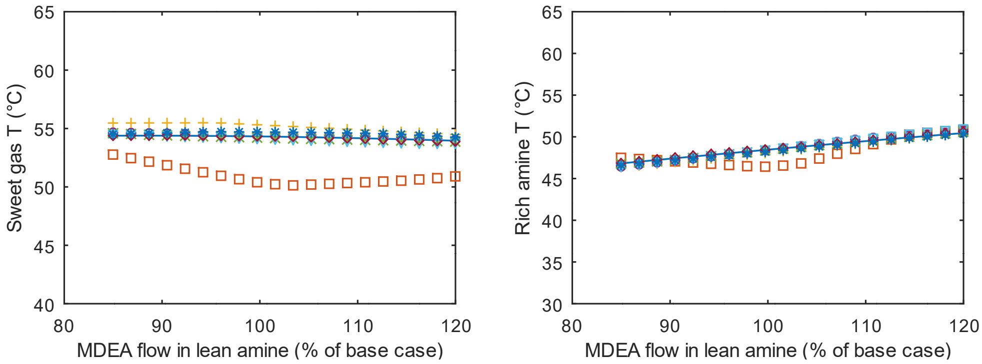


Effect of MDEA flow variation





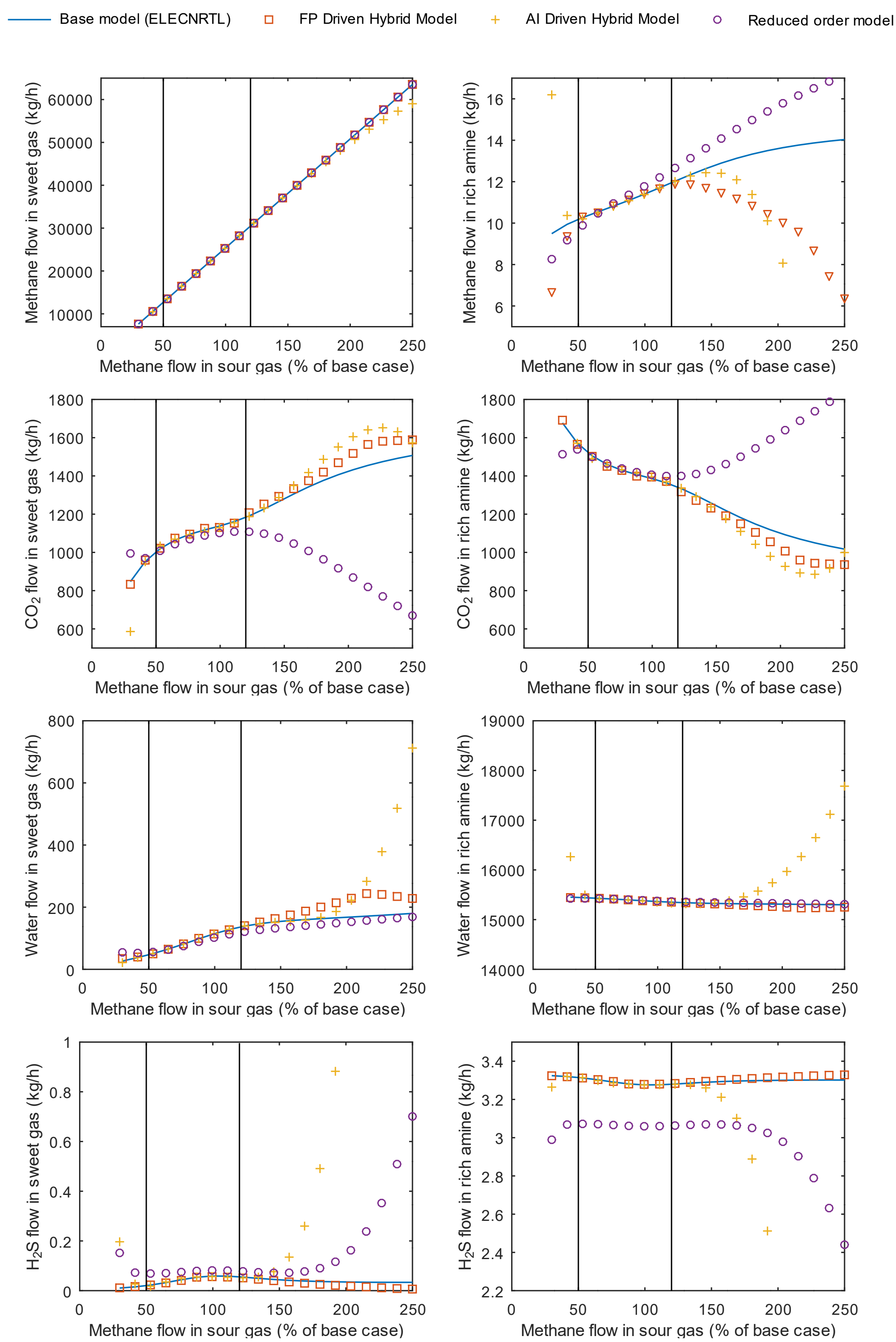
Base model
243 data points, 90-120% data
3125 data points, 90-120% data
8343 data points, 90-120% data
25 000 data points, 90-120% data
84 000 data points, 90-120% data
25 000 data points, 95-110% data
25 000 data points, 92.5-115% data
CO2 flow in sweet gas (kg/h)
CO2 flow in rich amine (kg/h)
Water flow in sweet gas (kg/h)
Water flow in rich amine (kg/h)
H2S flow in sweet gas (kg/h)
H2S flow in rich amine (kg/h)
MDEA flow in sweet gas (kg/h)
MDEA flow in rich amine (kg/h)
MDEA flow in lean amine (% of base case)

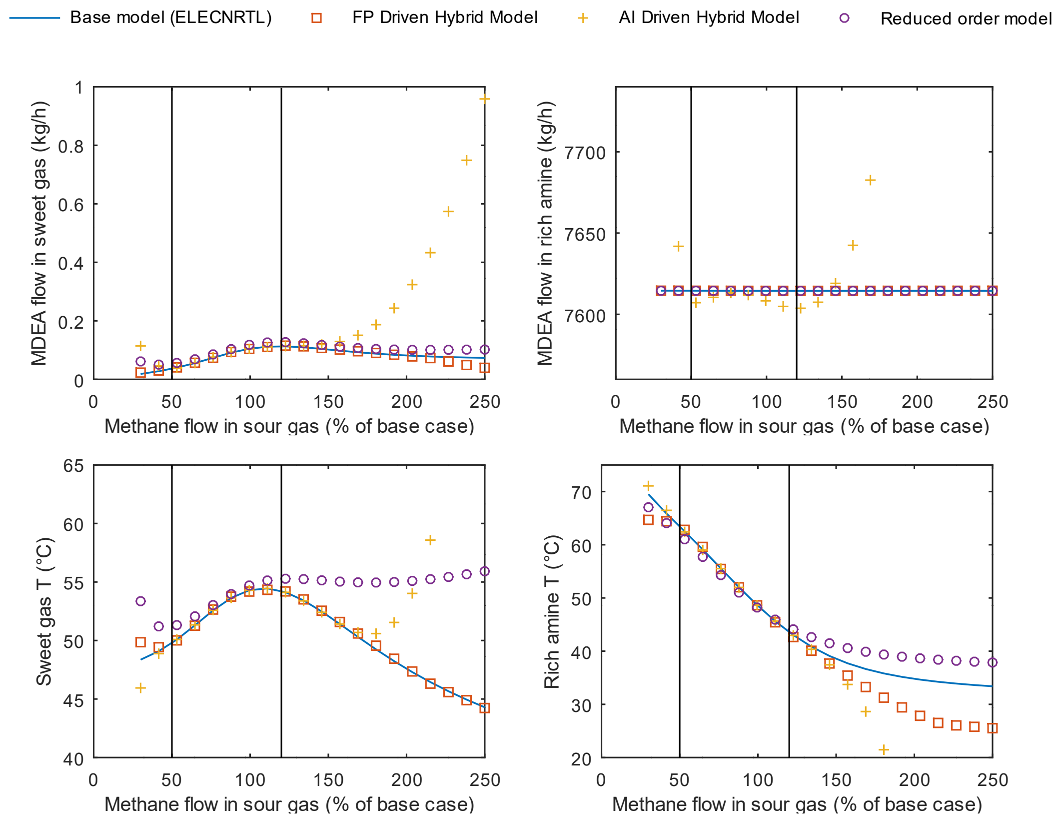


Sweet gas T (°C)
MDEA flow in lean amine (% of base case)
Rich amine T (°C)
MDEA flow in lean amine (% of base case)

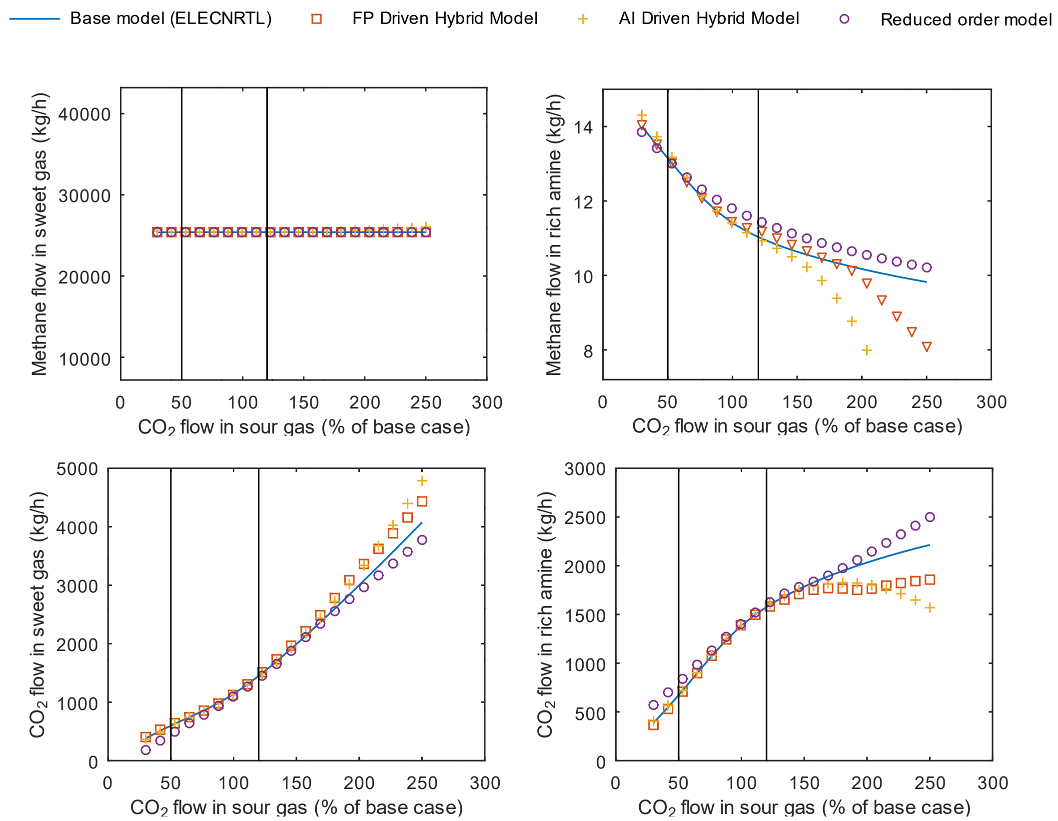
Appendix VII – Additional figures for extrapolation tests with the different models

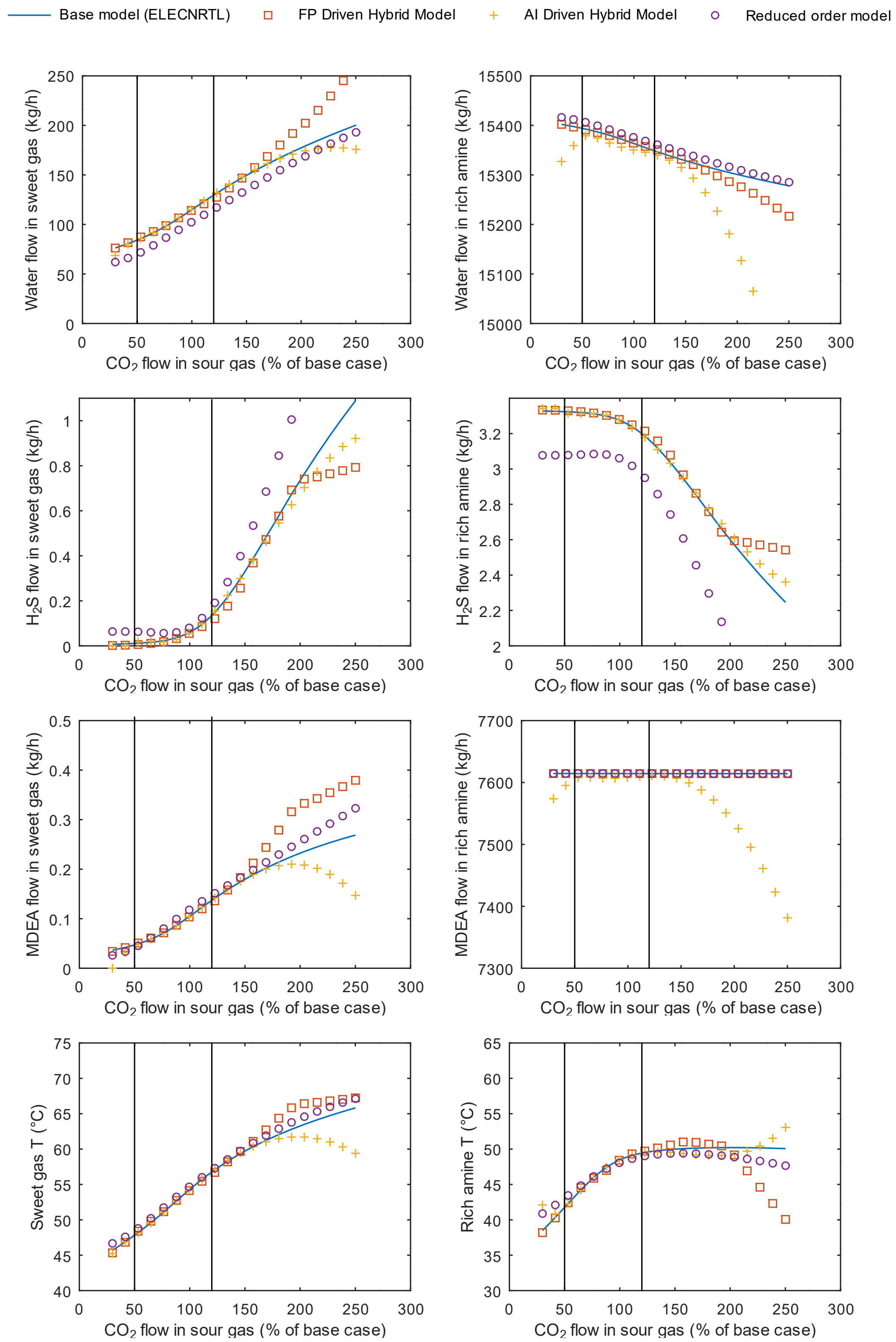
Effect of methane flow variation





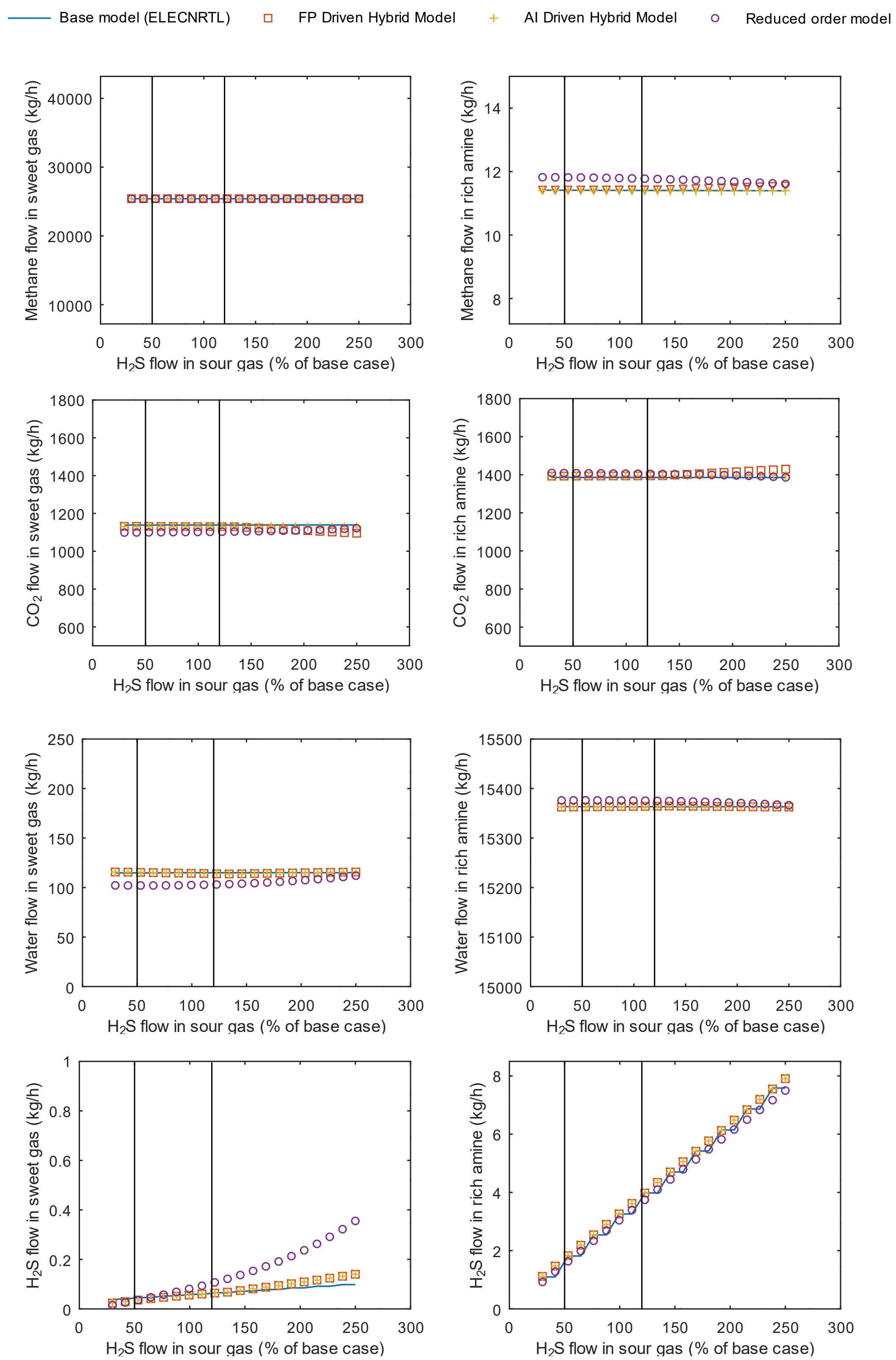
Effect of CO_2 flow variation

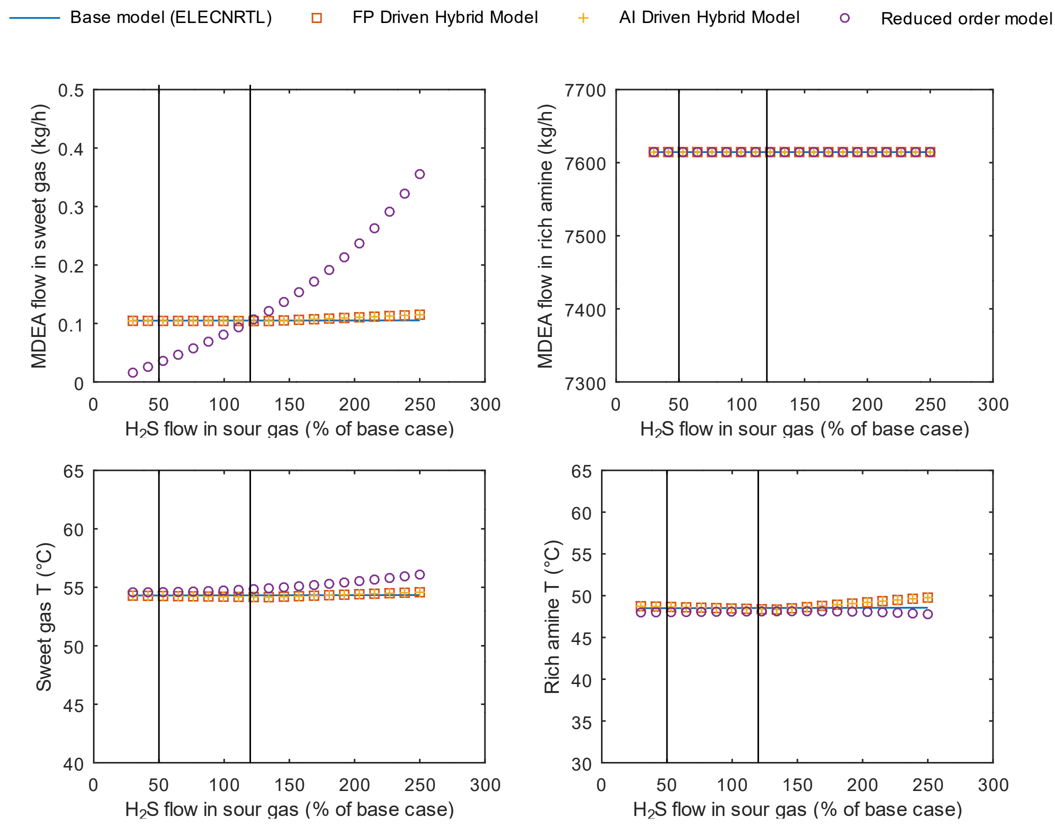




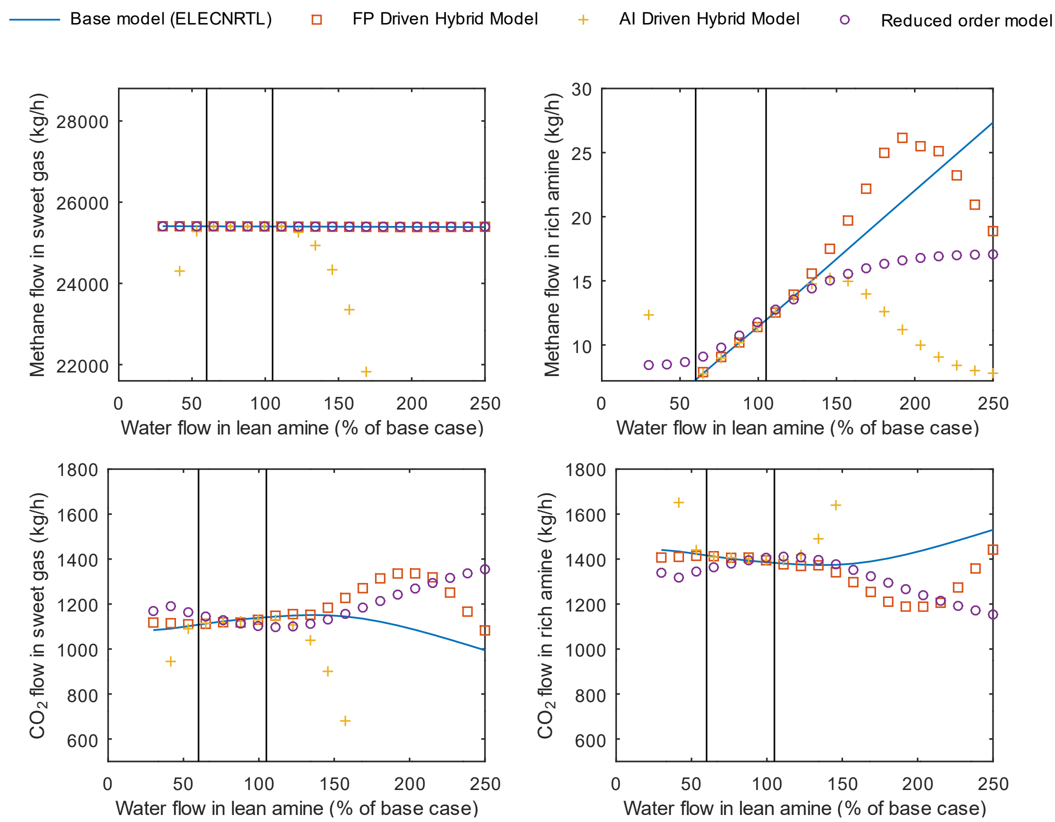
Base model (ELECNRTL)
FP Driven Hybrid Model
AI Driven Hybrid Model
Reduced order model
Water flow in sweet gas (kg/h)
Water flow in rich amine (kg/h)
H2S flow in sweet gas (kg/h)
H2S flow in rich amine (kg/h)
MDEA flow in sweet gas (kg/h)
MDEA flow in rich amine (kg/h)
Sweet gas T (°C)
Rich amine T (°C)
CO2 flow in sour gas (% of base case)

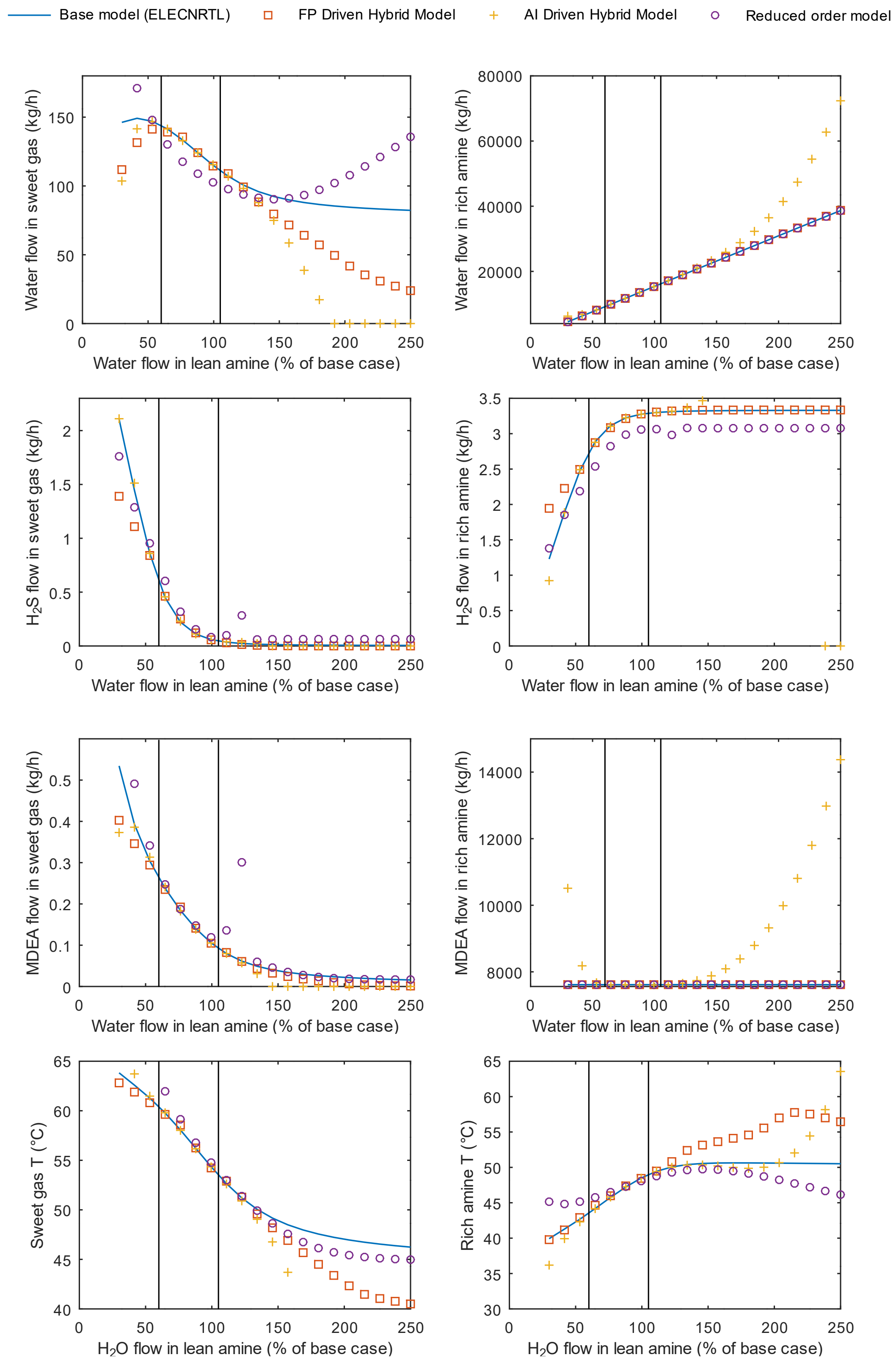
Effect of H_2S flow variation





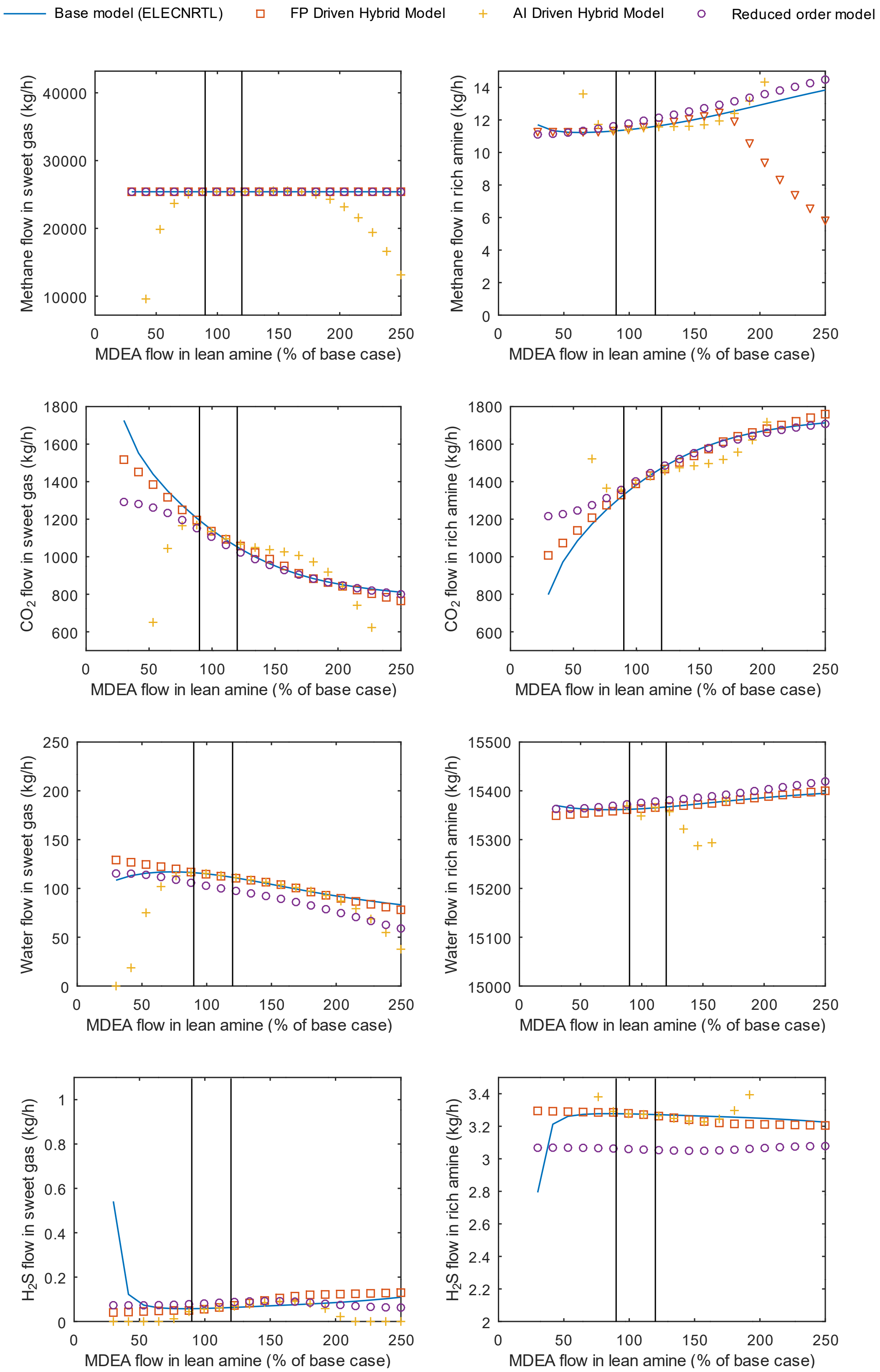
Effect of H_2O flow variation

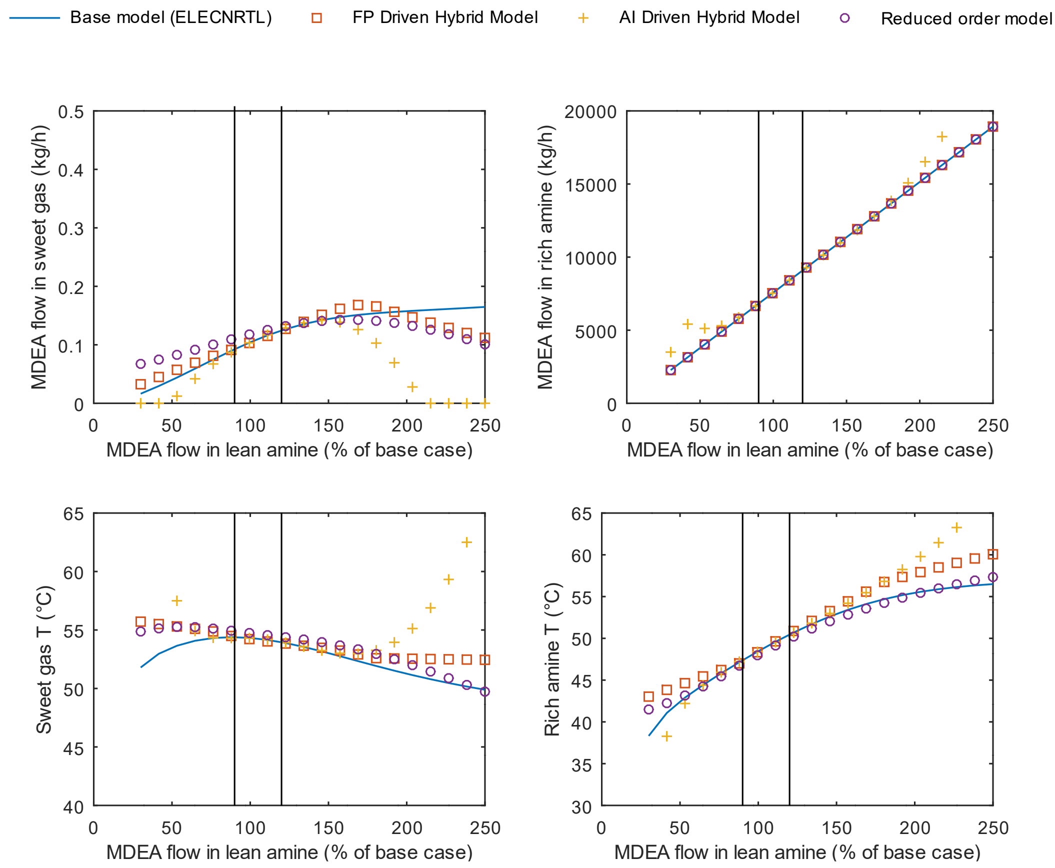




Base model (ELECNRTL)
FP Driven Hybrid Model
AI Driven Hybrid Model
Reduced order model
Water flow in sweet gas (kg/h)
Water flow in lean amine (% of base case)
Water flow in rich amine (kg/h)
Water flow in lean amine (% of base case)
H2S flow in sweet gas (kg/h)
Water flow in lean amine (% of base case)
H2S flow in rich amine (kg/h)
Water flow in lean amine (% of base case)
MDEA flow in sweet gas (kg/h)
Water flow in lean amine (% of base case)
MDEA flow in rich amine (kg/h)
Water flow in lean amine (% of base case)
Sweet gas T (°C)
H2O flow in lean amine (% of base case)
Rich amine T (°C)
H2O flow in lean amine (% of base case)

Effect of MDEA flow variation





Base model (ELECNRTL)
FP Driven Hybrid Model
AI Driven Hybrid Model
Reduced order model
MDEA flow in sweet gas (kg/h)
MDEA flow in lean amine (% of base case)
MDEA flow in rich amine (kg/h)
MDEA flow in lean amine (% of base case)
Sweet gas T (°C)
MDEA flow in lean amine (% of base case)
Rich amine T (°C)
MDEA flow in lean amine (% of base case)