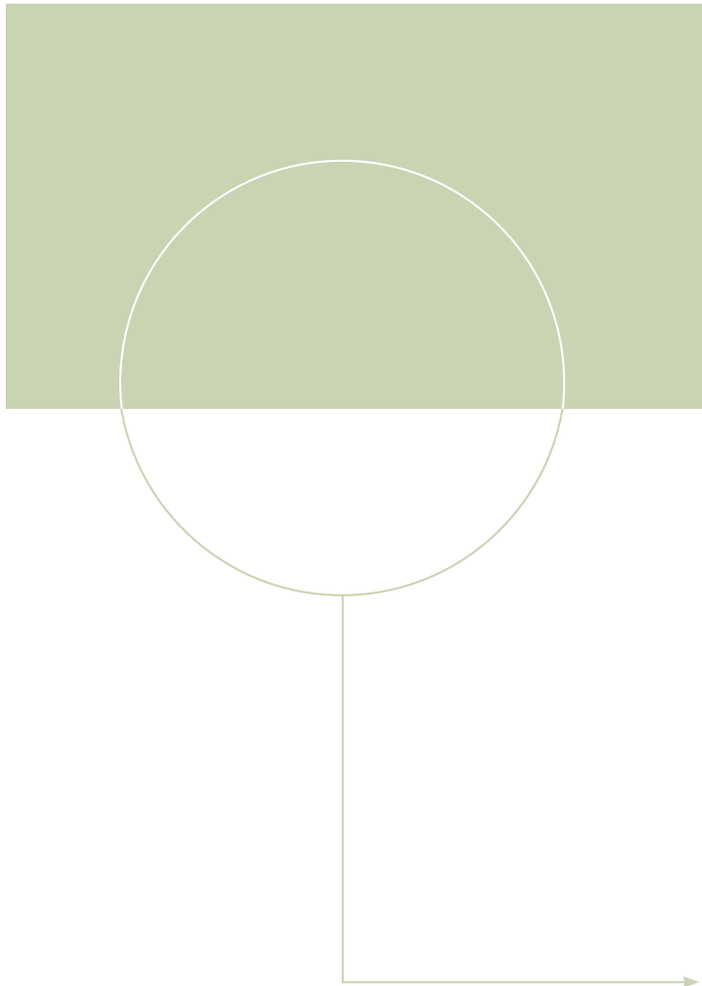


Master's thesis

2022

Anders Bendiksen

NTNU
Norwegian University of
Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

Anders Bendiksen

Creating a machine learning based surrogate model for modelling of a carbon capture plant

Creating a neural network for case studies of carbon capture plants

January 2022

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Creating a machine learning based surrogate model for modelling of a carbon capture plant

Creating a neural network for case studies of carbon capture plants

Anders Bendiksen

Chemical engineering and biotechnology
Submission date: January 2022
Supervisor: Hanna Knuutila
Co-supervisor: Andressa Nakao & Hallvard F. Svendsen

Norwegian University of Science and Technology
Department of Chemical Engineering

Abstract

When conducting case studies of carbon capture plants, the current solutions that exist are both tedious and time consuming. In order to speed up this process, a neural network model can be trained on data produced by a conventional simulation tool, and then in turn perform the case study. The neural network model was created using CO2Sim and keras/tensorflow and had a mean average percentage error of 3.8103%, and returned both plant specifications and economic data that are within the bounds of what one would expect when compared to existing literature.

Sammendrag

Dagens metoder for å gjøre en parameterstudie av et karbonfangstanlegg er ofte en treg prosess som krever mye innsats og manuell input. For å gjøre denne prosessen mindre arbeidskrevende ble et neuralt nettverk konstruert i keras/tensorflow og trent på data produsert av CO2Sim. Modellen hadde en MAPE på 3.8103%, og både responsvariablene og den økonomiske analysen gav verdier som var innenfor det man skulle forvente når man sammenlignet med verdier fra eksisterende forskning.

Acknowledgments

I would like to thank my supervisor for the invaluable aid and insights given throughout the course of this master thesis. Furthermore, the endless patience and helpfulness of my co-supervisors Andressa Nakao and Hallvard Svendsen proved absolutely indispensable. I would moreover like to thank my friends and family for their ever-present warmth and friendship which has helped me through both this work and Covid-19.

List of Figures

List of Tables

List of symbols

Symbol	Unit	Description
α	-	CO_2 Loading
Z	m	Column packing height
G_m	mol/m^2	Molar gas flow per area unit
L_m	mol/m^2	Molar liquid flow per area unit
a	m^2/m^3	interfacial area per volume unit
P	bar	Pressure
C_t	mol/L	total molar concentration
y_i	-	vapour molar fraction
x_i	-	liquid molar fraction
y_e	-	vapour equilibrium molar fraction
x_e	-	liquid equilibrium molar fraction
Q	kJ/h	Heat rate/duty
U	W/m^2K	Total heat transfer coefficient
A	m^2	Area
ΔT_{lm}	K	log mean temperature
h_o	W/m^2K	Outside film fluid coefficient
h_i	W/m^2K	Inside film fluid coefficient
$h_o d$	W/m^2K	Outside dirt coefficient
$h_i d$	W/m^2K	Inside dirt coefficient
k_w	W/mK	Thermal conductivity tube material
d_i	m	Tube inside diameter
d_o	m	Tube outside diameter
H_c	-	Correction factor condensate
K_L	W/mK	Condensate thermal conductivity
ρ_i	kg/m^3	Density in phase i
μ_L	$Pa \cdot s$	dynamic viscosity of liquid
Γ_h	m^3/m	Tube loading
g	m^2/s	gravitational acceleration

Symbol	Unit	Description
$\dot{m}$	kg/h	mass flow
H	m	Pressure head
η	-	compressor efficiency rating
f_{er}	-	erection factor
f_p	-	piping factor
f_i	-	instrumentation factor
f_{el}	-	electricity factor
f_c	-	civil construction factor
f_s	-	structures & buildings factor
f_l	-	lagging & paint factor
f_m	-	material factor
σ	-	Activation function
$R(\theta)$	-	Error function
λ	-	tuning parametre
$J(\theta)$	-	regularization penalty term
W	-	regularization term
N	-	number of observations
y_i	-	actual value of y_i
$\hat{y}_i$	-	predicted value of y_i
∇f	-	gradient of function f
γ	-	learning rate
Ω	-	Big-O notation
s	-	sample standard deviation
μ	-	expected value

List of abbreviations

Symbol	Description
MEA	Monoethylene Amine
ReLU	Rectified Linear Unit
L1	L1 regularization
L2	L2 regularization
MSE	Mean Square Error
MAPE	Mean Average Percentage Error
ADAM	ADaptive MOmentum
NG	Natural Gas
V/L	Vapour/liquid mass flow ratio
Capex	Capital expenditures
Opex	Operating expenditures

Contents

1 Introduction

According to NASA, 2015 and 2020[1] are tied for the warmest years on record, with the last seven years being the seven warmest years ever recorded. If drastic measures are not taken, the 1.5 degree goal outlined in the Paris Agreement seems more and more like optimistic fiction. The capture of CO_2 via aqueous amines currently respresents one of the more mature ways to remove carbon from circulation.

When conducting case studies of different carbon capture plants, the current way of doing it involves using simulation programmes like OGT's Protreat or Aspen Plus. Often these programmes only allow you to study the effect of changing a single parametre, making a comprehensive case study a time consuming and tedious process. In order to streamline this process, a surrogate model of a carbon capture plant was created using a neural network. This surrogate model takes a .csv file with the inputs and returns another .csv file with the responses, greatly reducing the time it would normally take to conduct a similar case study in a traditional process simulation tool. The neural network model should also be used to perform a cost analysis on the resulting carbon capture plant, in order to better understand how the different factors affect the final capital and operating expenditure.

2 Theory

2.1 CO2 absorption

While direct-from-air carbon capture has been talked about and even trialled[2], the most efficient way of controlling the amount of CO_2 in the atmosphere is by not emitting it at all. Naturally this is not always possible, and removing CO_2 from flue gas can allow us to cut the carbon emissions drastically in a feasible way. First of all, the flue gas usually has a higher partial pressure of CO_2, which makes the capture process more efficient[3]. And secondly, the post combustion capture process allows us to concentrate the economic investment where it will make the biggest impact, i.e. at the point of emission.

CO_2 is absorbed in the aptly named absorber column, usually by aqueous amines. The advantage of aqueous amines, is that the process can be easily reversed, most often by either shifting the temperature, or the pressure. What this means in practice is that the CO_2 absorption itself releases energy, whilst the CO_2 desorption requires energy. The regenerated solution is then returned to the absorption column where the process is repeated. There is however an energy penalty related to regenerating the aqueous solution, usually somewhere in the ballpark of 30% of the energy originally produced by the power plant[4]. Of course this figure can be reduced by rigorous experimentation and improvement through innovation. Another great advantage that the amine scrubbing method has is that it can more readily be retrofitted to an already existing plant[5].

In figure 1 a simplified flow diagramme of the absorption process can be seen. At first, the CO_2-rich flue gas is fed into the bottom of the absorber column. Here it meets the lean amine absorbent, usually sprayed from the top of the column through a packing to maximize the surface interface area. In the stripper column, the rich amine solution is sprayed from the top of the column, where it meets heated steam from the reboiler, this then reverses the reaction and releases the CO_2. Which is subsequently removed from the top of the column. The condenser makes sure that the aqueous amine solution is returned to the column. In the bottom of the stripping column, the reboiler adds the heat necessary for the reversion reaction to occur. The lean aqueous amine solution is also removed from the bottom of the column, and a heat exchanger is used to minimize the heat wastage by heating the rich amine solution coming from the absorber column. The lean amine is then pumped back into the absorption column.

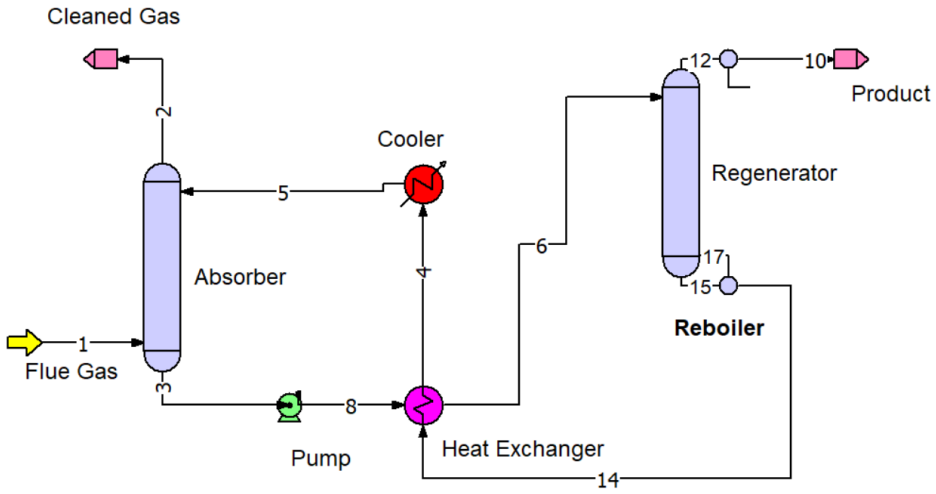


Figure 1: A process flow diagramme of an CO_2 absorption model

There are two main mechanisms[6] for aqueous absorption of CO_2, physical and chemical. Physical absorption is when the CO_2 is dissolved whithin the solvent itself, and chemical absorption is when chemical agents are added to the solvent that react with the absorbed CO_2 and thus removing it from circulation and further facilitating physical absorption of CO_2. For the purposes of this work, the system uses a combination of both physical and chemical absorption.

2.1.1 Monoethylene amine

MEA or monoethylene amine is an organic compound with the chemical formula of $HOCH_2CH_2NH_2$. Normally, it is mixed with water to form aqueous MEA. Aqueous MEA is today the most common way of scrubbing CO_2 from flue gas[3]. Bihong et al. 2015[7] concludes that the zwitterion mechanism is the most probable path of CO_2 absorption into aqueous MEA. The Zwitterion mechanism first reacts amines and cO_2 to form zwitterions, then the intermediary reacts with a base to form carbamate. The reactions are given in equation 1 and 2.

$$CO_2 + RNH_2 <=> RNH_2^+COO^- \tag{1}$$

$$RNH_2^+COO^- + RNH_2 <=> RNHCOO^- + RNH_3^+ \quad (2)$$

The aqeuous MEA is capable of absorbing a large amount of CO_2 before becoming saturated. The most common way of quantifying the amount of CO_2 in the aqueous amine is through the loading α.

$$\alpha = \frac{\text{mole of } CO_2 \text{ carrying species}}{\text{mole of } NH_2 \text{ carrying species}} \quad (3)$$

2.2 Design of process equipment

2.2.1 Absorption columns

Columns are used for a multitude of different purposes, among them are distillation, liquid-liquid extraction and gas absorption. Of these only absorption is of any relevance to the capture of CO_2. In this thesis, only packed columns will be considered.

In an absorption column, the gas and liquid travel counter-flow through the column. The flue gas enters sideways at the lower end of the column, and travels upward towards the top of the column. The amine wash enters the column from the top-side of the column and travels downwards through the packing. The packing serves mainly three purposes[8]:

- Provide a large interfacial area between the gas and liquid
- Ensure low resistance to fluid flow
- Promote uniform fluid distribution across both the packing surface and column cross-section

This then allows the mass exchange between the gas and the liquid to reach it's fullest potential. Structured packings are the most common packing for gas absorption[9], they consist of bundles of corrugated metal, plastic or ceramic sheets that may or may not be perforated.

When designing absorption columns, there are a few main considerations; the type of the packing and the height and width of the column. The dimensions of

the column, or rather the dimensions of the packed part of the column, is the main deciding factor for how much mass transfer can take place..

When modelling gas absorption in a column, certain assumptions are often made to simplify the process. One of the more common simplifications is to assume that the gas flow rate through the column does not change as the solute is absorbed by the liquid phase. This assumption holds true if the concentration of the solute in the gas phase is low enough, say 10%[8]. When this is true, one can combine the equations 4 and 5 in order to find the necessary height of the packing.

$$Z = \frac{G_m}{K_G a C_t} \int_{y_2}^{y_1} \frac{dy}{y - y_e} \tag{4}$$

$$Z = \frac{L_m}{K_L a C_t} \int_{x_2}^{x_1} \frac{dx}{x_e - x} \tag{5}$$

- Z = the required height of the packing
- G_m = molar gas flow rate per unit cross-sectional area
- L_m = molar liquid flow rate per unit cross-sectional area
- a = interfacial area per unit volume
- P = pressure
- C_t = total molar concentration
- y_1 & y_2 = the gaseous molar fractions of the solute in the bottom and top of the column
- x_1 & x_2 = the liquid molar fractions of the solute in the bottom and top of the column
- y_e = equilibrium concentration of the solute on the gas phase
- x_e = equilibrium concentration of the solute on the liquid phase

If the assumption of similar gas flow throughout the column does not hold because of a too large concentration of solute, the column must be broken up into sections that individually have approximately the same gas flow rate. Desorption is the same process as absorption only in reverse, and the same design methods apply.

2.2.2 Heat exchangers

The general equation for heat transfer across a surface[8] is:

$$Q = UA\Delta T_m \tag{6}$$

Q being the heat transfer rate, U the overall heat transfer coefficient and T_m being the temperature difference, the driving force. The primary objective when designing heat exchangers is to determine the area needed to facilitate the necessary heat transfer. For heat exchange across a typical heat exchanger can be expressed as:

$$\frac{1}{U_O} = \frac{1}{h_O} + \frac{1}{h_{od}} + \frac{d_o ln(\frac{d_o}{d_i})}{2k_w} + \frac{d_o}{d_i} \cdot \frac{1}{h_{id}} + \frac{d_o}{d_i} \cdot \frac{1}{h_i} \tag{7}$$

The variables are:

- U = the overall heat transfer coefficient $[Wm^{-2}K^{-1}]$
- h_o = outside fluid film coefficient
- h_i = inside fluid film coefficient
- h_{od} = outside dirt/fouling coefficient
- h_{id} = inside dirt/ fouling coefficient
- k_w = thermal conductivity of the tube wall material
- d_i = tube inside diameter
- d_o = tube outside diameter

There are however ways top determine the overall heat transfer coefficient empirically, so that one does not need to individually model the fouling. Values can either be found through experiments, or taken from already existing literature.

The most common and most inexpensive type of heat exchanger is the shell and tube exchanger. It consists, as the name implies, of a shell, and a tube. In essence it is a bundle of tubes enclosed by a shell, with both shell and tube each being connected to an inlet and an outlet. The advantages are numerous and include[8]: well established construction techniques, easy maintenance and high surface area

to volume ratio.

When sizing and estimating cost of heat exchangers, the area needed to facilitate the heat transfer is the single most important factor. When finding the driving force (temperature) for a common counter-current heat exchanger, the logarithmic mean temperature[8] is used as a stand in. The equation given in equation 9 relates the heat Q to the other factors used to design the heat exchangers.

$$T_m = \frac{(T_1 - t_2) - (T_2 - t_1)}{ln(\frac{T_1 - t_2}{T_2 - t_1})} \tag{8}$$

$$Q = UAT_{lm} \tag{9}$$

Reboilers are simply heat exchangers that use a hot medium to warm another fluid. Therefore the design considerations are quite the same as when designing ordinary shell & tube heat exchangers. The only difference is that the hot flow is usually steam, and the inlet and outlet temperatures are usually known.

When designing and sizing condensers, the most common way of doing so is as a common shell & tube heat exchanger[8]. One notable difference is that a wider baffle spacing is used to accomodate the condensed liquid, with typically $l_B = D_s$.[8] Because of the heat transfer coefficient of the condensate would generally be lower than that of the material of the tubes, a correction factor h_c is generally used. When the condensing flow is inside the tubes of the heat exchanger

$$(H_c)_s = 0.76K_L \cdot [\frac{\rho_L(\rho_L - \rho_V)g}{\mu_L \Gamma_h}] \tag{10}$$

In equation 10, K_L is the condensate thermal conductivity, rho_L and ρ_V are the densities of the liquid and vapour flow, μ_L is the condensate viscosity, g is the gravitational acceleration constant and Γ_h is the tube loading, or the condensate flow per unit length of tube. As might be expected, when creating a surrogate model based on another model, the information necessary to calculate the exact correction factor may not necessarily be availible. And for this purpose Sinnot & Towler[8] gives us the approximate value of 0.8 that may be used in a pinch.

2.2.3 Compressors

When designing and sizing compressors, the most important feature is the power required for the compression work. Also, every pump or compressor has a efficiency rating that relates the power input to the actual compression work done by the compressor. When in doubt, Sinnot & Towler[8] tells us to use an effiiency rating of 0.7. The equation 11 taken from Geankoplis[10] relates the flow rate (Q), fluid density (ρ), the pressure head (H) and efficiency (η) to the power required (P).

$$P = \frac{\dot{m}\rho g H}{3.6 \cdot 10^3 \eta} \tag{11}$$

The most common type of compressor is the single stage, centrifugal pump[8]. Other pump types are used of there are special considerations, like a high pressure head, or small flow rates of additives.

2.3 Cost estimation

Chemical plants mostly exist to turn a profit, or in the case of carbon capture, to be as economical as possible. In order to ensure this, a reliable cost estimate is necessary. Estimating the cost of complex process equipment is an art to itself, and there are multiple resources and engineers who have dedicated their professional lives to the betterment of this discipline. For the purposes of this thesis, cost functions from Sinnot & Towler[8] were used to estimate the purchasing cost of the equipment. The cost curves are given in table 1 and are on a US Gulf Coast basis in January 2010 US dollars. With the exception of the pressure vessels and the structured packing all the equipment is given as the cost of purchasing in carbon steel.

$$C_e = a + b \cdot S^n \tag{12}$$

The cost estimates given above in table 1 are only for the equipment itself. Therefore there are multiple factors that need to be accounted for to ensure a correct price estimate. In order to ascertain that the estimates are consistent the same source is used for all steps of the cost estimation, with the notable exception of inflation and currency conversion data. This is because there are newer and more up to date sources available, for instance from the Bureau of Labor Statistics and

Items	Sizing Unit	a	b	n
Centrifugal Compressor	kW	580000	20000	0.6
Pressure Vessel 304SS Vertical	kg	17400	79	0.85
Pressure Vessel 304SS Horizontal	kg	12800	73	0.85
Mellapak 250y 304SS	m^3	0	7600	1
Heat Exchangers Shell&Tube	m^2	28000	54	1.2
Reboiler	m^2	29000	400	0.9

Table 1: The cost curves of the relevant process equipment

the SSB.

The purchase cost of individual pieces of equipment are only a small part of the total cost of the plant. In addition to the equipment itself, several other factors also play a role in determining the total cost.

- Equipment erection.
- Piping, including isolation and painting.
- Electrical, power and lighting.
- Instruments and process control systems.
- Provisions for utilities like steam and cooling water.
- Site preparation.

In addition to these costs, one must also consider the rising cost of everything, the price of the equipment in stainless steel instead of carbon steel and the added cost of building the plant in another part of the world that is not the US Gulf coast.

To account for inflation, Sinnot & Towler gives us the following expression to calculate the added cost.

$$\text{Cost in year B} = \text{Cost in year A} \cdot \frac{\text{Cost index year A}}{\text{Cost index year B}} \tag{13}$$

In order for the final prices to be correct, the equipment purchasing cost must take into account the material. The cost curves from Sinnot & Towler gives the price in carbon steel. This is done because carbon steel is generally the least

Material	Cost Factor
304 stainless steel	1.3
316 stainless steel	1.3
321 stainless steel	1.5
cast steel	1.1

Table 2: The material costs of certain alloys compared to carbon steel

Major equipment, total purchase cost	Factor
Equipment erection, f_{er}	0.3
Piping, f_p	0.8
Instrumentation & control, f_i	0.3
Electrical, f_{el}	0.2
Civil, f_c	0.3
Structures & buildings f_s	0.2
Lagging & paint f_l	0.1

Table 3: The installation factors

expensive option, but one must also take into account that carbon steel is often not the optimal choice of material. And stainless steel is often better because of it's inert nature. The cost factors relative to carbon steel are given in table 2.

The typical installation factors for a fluid-fluid processing plants are given in table 3 and are from Sinnot & Towler[8].

When estimating the cost, the most common error is overestimating the cost of installation by applying the above factors to the cost of the purchased equipment in stainless steel. They should be multiplied by the purchasing cost in carbon steel, and to account for this, one should use the expression in equation 14. M is the number of equipment pieces, $C_{e,i,A}$ is the purchase cost of equipment i in

Additional costs	Factor
Offsite	0.3
Design & engineering	0.3
Contingency	0.1

Table 4: The cost factor for building out the plant site

carbon steel and the rest are the cost factors given in table 3.

$$C = \Sigma_{i=1}^{i=M} C_{e,i,A}[(1+f_p)f_m + (f_{er}+f_{el}+f_i+f_c+f_s+f_l)] \tag{14}$$

$$C = \Sigma_{i=1}^{i=M} C_{e,i,A}[(1+f_p) + (f_{er}+f_{el}+f_i+f_c+f_s+f_l)/f_m] \tag{15}$$

If the purchased equipment already is priced in the correct alloy (not carbon steel), then equation 15 should be used instead.

2.4 Statistical modelling

Statistical modelling is the use of mathematical and statistical methods and models in order to classify or predict results. For this work, a neural network model will be used.

2.5 Neural Networks

Neural networks are often surrounded in a mystic aura, however they are nothing more than non-linear statistical models. They can be used for both regression and classification, making them extremely versatile. Which no doubt has contributed greatly to their current popularity. They derive their name from the fact that they are connected in a similar manner to neuron cells in a human brain[11].

In figure 2 a simplified model of a neural network can be seen. It has a single hidden layer, consisting of three neurons. It also has three inputs and a single response. The blue circles marked ”1” are the bias nodes. A single neuron works by taking in an input, multiplying it by a neuron weight, applying an activation function to this and then adds a bias term (the nodes marked ”1” in figure 2). The total sum of this is then passed to the next layer in the network. The role of the bias is equivalent to that of the intercept in a linear expression, and must not be confused with the model bias that is discussed in section 2.5.2. It also ensures that all of the neurons have something to pass on to the next layer and thus avoiding so called ”dead neurons”.

The process done by a single perceptron is given in equation 16.

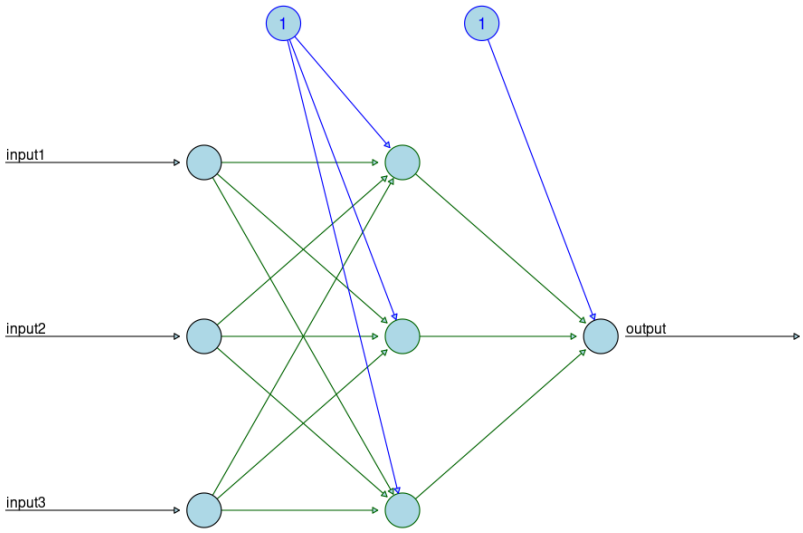


Figure 2: A simple neural network with three inputs, one response and a single hidden layer with three perceptrons.

$$Output = Activation\,Function(\Sigma(Weights \cdot inputs) + bias) \tag{16}$$

The neural network is a so called black box model, and as such there is no way to deduce the physical meaning of the weights[12]. This means that while a neural network may approximate a model very accurately, one must always be careful to ensure that the model returns good data

2.5.1 Activation functions

As mentioned in the segment above, activation functions are an integral part of neural networks. Activation functions are usually expressed as σ. If one wanted to model linear trends, the function $f(x) = ax + b$ could be used as the neuron activation. This would turn the model into a linear regression model, this is of course not mathematically interesting, but it does serve as an example of how activation functions work. For real-world scenarios, the norm is to use non-linear functions, with the sigmoid function, given in equation 17, rectified linear unit or ReLU, given in equation 18 and the hyperbolic tangent, given in equation 19

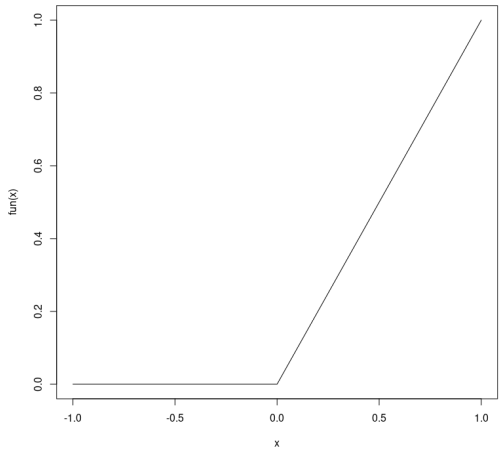


Figure 3: A plot of the rectified linear unit

being amongst the most common ones[13].

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{17}$$

$$\sigma(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases} \tag{18}$$

$$\sigma(x) = \frac{e^{2x}-1}{e^{2x}+1} \tag{19}$$

In figure 3 a plot of the ReLU activation function can be seen. From prior experience the ReLU function is almost always the correct answer when it comes to regression. It's great versatility comes from the fact that it does not activate all the functions at once, allowing a ReLU based model to approximate non-linear trends. And from the fact that it is so simple, keeping the model from becoming too complex and ensuring a fast training period.

2.5.2 Bias-variance trade-off

Overfitting is a common problem in statistical modelling. When a model has been overfitted, it will most likely result in the model performing very well on the data

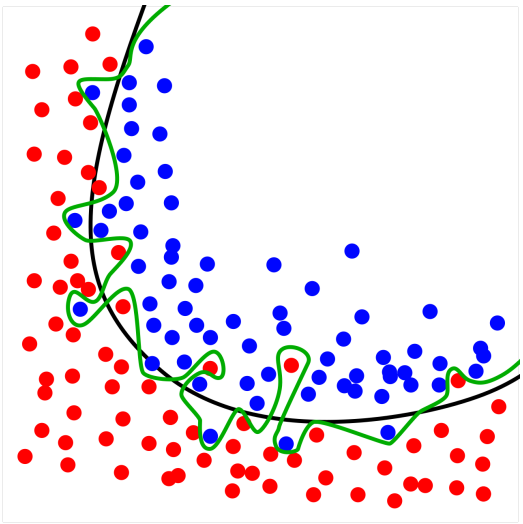


Figure 4: An illustration of overfitting. Picture by Ignacio Icke, distributed under a CC BY-SA 4.0 license.

it has been trained on, but performing deficiently when presented with new data. Therefore, one must deliberately make it so that the model performs poorer on the training data in order for it not to also learn the noise present in every dataset. In figure 4, the green model has been overfitted to the training data. The black model, while undoubtedly performing poorer on the training data has better understood where the true boundary between the blue and red dots is. And would presumably perform better on new data. Overfitting was a constant problem plagueing neural networks back in their infancy, and one of the early ways of combatting it was stopping the training process early[11], and simply praying that this was enough. Luckily, newer and more sophisticated methods have been developed since.

The two most common ways of avoiding overfitting today are through regularization and dropout. Dropout is essentially randomly choosing a set percentage of the neurons in a layer, and then setting their output to zero and thus stopping the weight and bias from being updated. The other way is through regularization, this method works by adding a penalty term to the cost function of the neuron output. This is expressed in equation 20, where $R(\theta)$ is the original cost function, and $\lambda J(\theta)$ is the penalty term. λ is a tuning parameter and $J(\theta)$ is the expression from equation 22 or 21.

$$R(\theta) + \lambda J(\theta) \tag{20}$$

$$J(\theta) = \Sigma_i |W|_i \tag{21}$$

$$J(\theta) = \Sigma_i W_i^2 \tag{22}$$

The two most common ways to regularize are L1 and L2 regularization. They are quite similar, with the difference being that L1 adds a single W-term to the error function as shown in equation 21, and L2 adds a W^2-term as shown in equation 22. They differ in that L1 will estimate around the median of the data, while L2 will estimate around the mean of the data[13]. And this can greatly affect the accuracy of the final model, depending on how the data is distributed.

2.5.3 Training neural networks

The most common, and often the best method of training a neural network is by simple backpropagation. Backpropagation, which is shorthand for backwards error propagation works by matching known response variables to the input variables, and then computing the sum of an error function. The weights and biases are then updated along the gradient of this error function, and then the process is repeated until the desired accuracy is reached. The cost function is often the mean square error (MSE), given in equation 23, or the mean absolute percentage error (MAPE), as given in equation 24. The advantage that the MAPE may have over the MSE is that it allows the model error to be trained evenly across all parametres, and not potentially minimizing the error of one variable to the detriment of others.

$$MSE = \frac{1}{N}\Sigma_{i=1}^{K}(y_i - \hat{y}_i)^2 \tag{23}$$

$$MAPE = \frac{1}{N}\Sigma_{i=1}^{K}\frac{|(y_i - \hat{y}_i)|}{y_i} \tag{24}$$

In order to find the gradient of the loss function, an optimization algorithm must be chosen. The most natural choice is the gradient descent. Gradient descent finds

the gradient by following the gradient of the cost function down the steepest slope, and in this way finding the minimum of the cost function. In order to limit the rate of change of the weights, one can implement a constraint called the learning rate. A too high learning rate will make the weights fluctuate and produce an unstable model. While a too low learning rate will maybe get stuck in a local minimum. Gradient descent[11] can be expressed as done in equation 25.

In order to update the weights and biases of the model correctly, one must find the gradient of the error function. The gradient can of course be found by using the Nabla-operator, as shown in equation 25. Here, b represents the next position, a represents the current position, γ is the learning rate, whilst the $\nabla f(a)$ is the direction of descent.

$$b = a - \gamma \nabla f(a) \tag{25}$$

The normal way of choosing an optimization algorithm is to read the latest peer reviewed works, and use the same as them. Cross-referencing different methods to ensure that the algorithm is actually the best performing. For this thesis, the adaptive momentum optimizer[14], or ADAM will be used. ADAM is a very new algorithm, being first introduced in 2015 by Kingma & Ba[14]. This algorithm builds upon the gradient descent given in equation 25, but also incorporates adaptive learning rate and momentum. Adaptive learning rate works by adjusting the learning rate proportionally to the steepness of the gradient, and momentum adds a temporal element that hastens the convergence along the steepest gradient descent. A good analogy is that of a bowling ball rolling downwards guided by gravity. More information about ADAM kan be found in Kingma & Ba 2015[14].

3 Simulation of CO_2 capture plant

CO2Sim is a rigorous rate-based model developed and validated against data obtained from the Tiller carbon capture pilot project[15]. The data used to train the model was created in CO2Sim and the data exported as excel .xls files. The model is fairly standard when considering CO_2 absorption plants. With one reboiler, one stripper and a heat exchanger to maximise the heat recovery. This allows the model to simulate a wide variety of different plants, but it also restricts the applicability of the model to this single set-up. The data was then imported into python and a neural network was constructed using the tensorflow library. Cost analysis was also performed in python.

The input and response variables are given in tables 5 and 6. When choosing both input and response variables, great care must be taken so that the model returns actually useful information. For instance, the input data should be information that is generally known beforehand, like data pertaining to the flue gas and the desired capture rate. And the response data should then as far as possible describe the system given the input conditions. The more response variables you want the model to return, the more data you have to use for training it. If not, the model may not fully understand all the trends and return less accurate data.

Input variables
Column height [m]
Capture rate [%]
Mole fraction CO_2
V/L ratio

Table 5: The input variables of the model

3.1 Assumptions and background for model

In order to simplify the model and make it actually possible to create the training and validation data within a reasonable time frame, the flue gas flow was assumed constant at 253 $kmolh^{-1}$. The reason for this specific choice was mostly arbitrary, with 253 $kmolh^{-1}$ being the required gas flow rate for the absorber column flow speed to reach 2 m/s halfway through the column, with a column diameter of 1m.

Responses
Reboiler duty [kJ/h]
Solvent lean loading
Solvent rich loading
Sweet gas flow rate
CO_2 fraction (top of column)
Mole fraction H_2O Sweet gas
Mole fraction N_2 Sweet gas
Capture rate CO_2 [kg/h]
Condenser duty [kJ/h]
Delta Temperature rich flow [K]

Table 6: The response variables of the model

It was then assumed that scaling the flue gas flow rate and system by a factor would allow us to approximate many more plants, given that they also have a column gas flow rate of 2m/s. Other assumptions, such as the source of flue gas only affects the amount of carbon, and not the content of other components such as sulphur. This thesis is after all studying the viability of using neural nets to model carbon capture plants, and not an in-depth study of how MEA degrades.

3.1.1 Creating the training and validation data

As stated above, the data was created using CO2Sim with the MEA Astarita thermodynamic package. The data was then imported into R in order to combine and tidy up everything as CO2Sim returned many hundred .xls files in order to build the entire dataset. Whilst describing and specifying the entire CO2Sim model may not be an easy task, but a valiant effort will nonetheless be made. In table 7 the different units in the simulation model are listed, and a note about their specification, design or purpose is included. The heat coefficient values are from Aromada et al[16]. The way the whole system is interconnected is shown in figure 5 and the total number of datapoints in the training dataset was 1532.

The dataset was created along the limits given in table 8, with a concerted effort being made in distributing the data evenly between the extremities. This should allow the neural network to better deliver consistent results. It is however important to remember that the results are greatly affected by the choice of thermodynamic

Unit	Note
Absorption column	defined by model input
Desorption column	0.7 of absorption column
Column packing	Mellapak 250y
Compressor	compresses output to 8 bar
Heat Exchanger	approach temperature = 10K
-	U = 0.73 kW/m^2K
Post HEx cooler	duty specified by CO2Sim
-	U = 0.73 kW/m^2K
Reboiler	duty specified by CO2Sim
-	Pressure = 1.9 bar
-	U = 1.2 kW/m^2K
Condenser	duty specified by CO2Sim
-	U = 1 kW/m^2K
Aqueous MEA	12% mol
Con01	Used by CO2Sim to enforce a constant mass flow in the loop
Flash01/Flash02	Completes the system and ensures stability
Mix01/Mix02	Connects streams

Table 7: An overview of the different units in the CO2Sim model

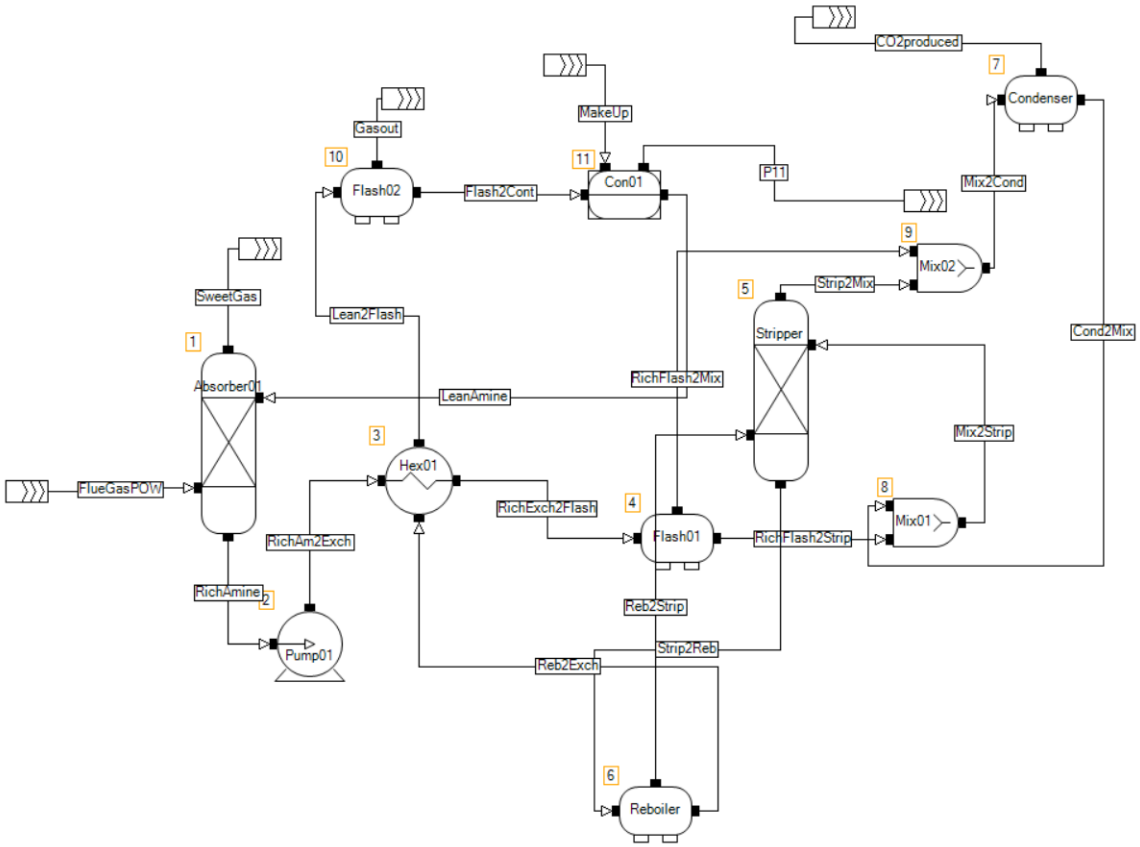


Figure 5: The model as it was created in CO2Sim

package, and that another package may yield different results. Also, the model is guaranteed to work on data within the bounds specified in table 8, and that the use of data outside of these bounds may return inconsistent and erroneous results.

CO_2 percentage in flue gas	3-15 [%]
Column height	10-20 [m]
V/L ratio	1.4-3.7
CO_2 capture rate	80-95 [%]

Table 8: The limits of the training data set

3.2 Cost analysis

The purpose of this thesis is not to accurately design a carbon capture plant down to the most minute detail, but rather to model trends in how certain parametres affect the cost of capturing CO_2. Taxes are not accounted for in the economic analysis of this model. This is because the plant does not produce anything that makes a profit. Rather, it is more probably that the government would provide tax rebates for CO_2 capture.

The cost analysis of this thesis is based on the theory presented in section 2.3. A short summary is presented below in list form in order to aid comprehension of exactly what factors and parametres are involved.

- Using the cost curves in table 1 to find the purchase cost of the equipment.
- Adjusting the purchase cost of equipment to the correct material using the factors in table 2.
- Using the installation factors given in table 3 to properly account for the erection and installation of the equipment.
- Correct the costs for location and inflation.
- Find the consumption of utilities in order to find the operating expenditures of the plant.

When performing cost analysis, certain parametres are very important to consider that do not fit neatly into the other categories presented earlier. Therefore they are given in table 9. The inflation data for the US dollar is given by the US department of labor, the location factor by Ali et al[17] and the rest by SSB.

Inflation factor USD (2010-2022)	1.26
Inflation factor NOK (2020-2022)	1.039
Location factor Norway	1.26
Long term rent for new loans	2.3%
Project lifetime	25 years
Depreciation model	straight line
Depreciation rate	4%

Table 9: Important parametres for cost analysis

3.2.1 Capital expenditures of the plant

To find the capital expenditures of the plant, the cost curves in table 1 were used to find the basic cost of all the equipment on a US Gulf Coast Jan. 2010 basis. The pressure vessels and structured packing are priced in 304 stainless steel, and the rest in carbon steel. Since the assumption is that the carbon capture is retrofitted to an already existing plant. The extra cost associated with building up a plant site is ignored for the cost analysis part.

When finding a cost estimate for the lean/rich heat exchanger, certain assumptions had to be made in order to simplify the model. To accurately find the log mean temperature difference of the lean/rich heat exchanger, one would need 4 new temperatures (the intlet and outlet for the two streams) added to the responses. When considering the relatively small training dataset, getting 14 response variables might be pushing the limits of the model too far, as the prediction error of the condenser duty of the pure CO_2 simply refused to go below 70% when all 14 variables were used. So instead, the temperature difference of the lean flow was used as a variable in the model. Then the log mean temperature differences of the training dataset were found. The mean of all these values was 10.624, with a standard deviation of 0.322. This is close enough to assume that the log mean temperature difference is constant throughout the whole dataset. The heat required to change the temperature of the lean flow to the required temperature was then combined with the duty of the cooler cooling the lean stream into the heat exchanger to fully specify the system. The heat capacity seems to not change all that much depending on the loading of the CO_2 and can then be assumed constant[18] at 4 kJ/kgK.

3.2.2 Operating expenditures of the plant

When finding the operating expenditures of the plant, the most accurate and sophisticated way is not to use the data from Sinnot & Towler, but rather to find similar processes in Norway and use their data for utilities like high pressure steam, electricity and cooling water. Or to use industry standard tools like Aspens varied portfolio of chemical engineering tools. Needless to say, the purpose of this thesis is not to accurately design a carbon capture plant down to the most minute detail, but rather to model trends in how certain parametres affect the cost of capturing CO_2. As would be expected, the prices for electricity and natural gas to produce steam fluctuate wildly, and perhaps even more so at this very moment. They do however provide us with a good estimate for utility cost estimation that we can then adjust for the change in price. J. Jakobsen et al.[19] gives us utility cost estimates from 2017 for steam produced from natural gas, steam produced from a process and electricity. Everything with a mind on CO_2 capture in Norway.

The utility costs given in table 10 are mostly from Aromada et al. 2020[16], with the notable exception of electricity and gas prices as these were found from respectively SSB and tradingeconomics.com. The price of steam was found from the amount of heat required to turn one tonne of 288.15K water into 523.15K steam. On one hand, producing CO_2 to remove CO_2 may seem counterintuitive, especially since the combustion process that produces the flue gas itself should probably be enough to create enough high pressure steam to power the reboiler. Since the whole purpose of this thesis is to capture CO_2 one can safely assume that there is some form of combustion process happening, hopefully one that produces excess heat that can then produce high pressure steam. The exact make-up (with regards to process/NG) of the steam used to power the reboiler cannot be known until the all the details about the process that produces the CO_2 is known. Until these factors are known however, this work will assume that all the steam comes from a gas fired heater and not the main combustion process.

Since the prices of natural gas and electricity fluctuate wildly and today's prices are not necessarily relevant for the long term analysis of the financial viability of CO_2 capture. For instance, the electricity prices for the 3rd quarter of 2021 were used instead of the presumably abnormal prices we are currently seeing, and likewise, the gas prices are taken as those of 13^{th} of January. A large industrial plant like a power plant or refinery would in any case not be beholden to spot energy prices and would in all likelyhood minimize financial risk by signing long term contracts.

Utility	Price [2022]
Electricity (Q3 2021)	8.1 [$/MWh]
Natural gas (13.01.2022)	77.78 [$/MJ]
Steam (NG)	67.15 [$/tonne]
Steam (process)	0
Cooling water	0.025 [$/$m^3$]
Process water	0.24 [$/$m^3$]
Labour operator	95 375 [$/year]
Labour engineer	185 794 [$/year]
MEA	1798 [$/$m^3$]

Table 10: The cost of utilities

3.3 Conditioning the CO_2 for further transport

Before transporting the gas for either use or sequestration, the gas must be compressed and cooled in order to make it easier to handle. That is unless there is immediate use for the CO_2 at the plant site. This thesis will however assume that the gas needs to be processed for further transportation. As described in Aromada et al. 2020[16], the CO_2 product stream will be compressed from just above atmospheric pressure to 151 bar. As the compression is assumed to be adiabatic, cooling is required for the gas stream not to become too hot for transport. Multiple intercooling loops in between multistage compressors seem to be the best way to achieve a transport ready gas. These compressors and coolers would of course be designed and sized according to the principles of compressors and heat exchangers given in the theory section. Since the main purpose is a financial analysis and not a complete plant design, the finer details will be left for further study, and for cost analysis purposes it will be assumed that the compression can be achieved with three compressors and two cooling loops.

3.4 Neural network model

The model was built using keras/tensorflow with the Python interface. For a neural network with N observations, p predictors, M hidden units and L training epochs, the model the number of operations given in equation 26.

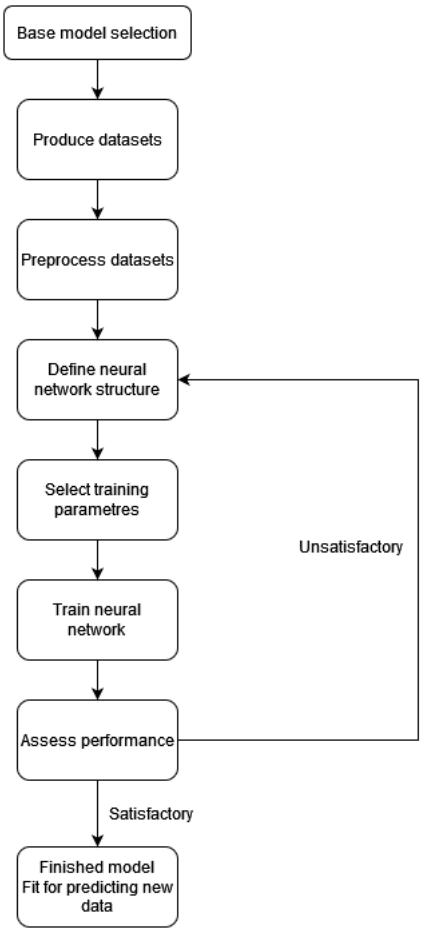


Figure 6: A flow diagramme of the model construction and training process

$$N_{operations} = \Omega(N \cdot p \cdot M \cdot L) \tag{26}$$

One can therefore see that the model has the potential to become rather large, this is however only a problem with super complex models with multiple recursive layers that are used for large data mining operations and image analysis. For the purposes of this thesis, it would be almost impossible to make a large unwieldy model that would take too long to train or propagate. As the main bottleneck would nonetheless be loading in the tensorflow library.

A flowsheet of how the model construction and training process can be seen in

figure 6. First the base model must be selected, then that model must be used to produce the training and validation datasets. Further it is very important that the input and output are the the same respective domains. Therefore they must be normalized/standardized. Next, the basic structure of the neural network model must be preliminarily determined. This neural network must then be trained on the data, and then validated against an independent dataset. If the accuracy is not good enough, the training parametres must be changed until the accuracy is adequate.

3.5 Parametre adjustment

Since statistical modelling and by extension neural networks is a highly empirical field. One cannot know with certainty what parametres are the best choice for the model in advance, and the best way for determining these parametres is by trial and error. Naturally, an exhaustive study of all the possible combinations of parametres would take too long and be outside the scope of this work. As such, certain choices have already been made in the name of keeping things simple. These choices are either explained in this section, or in the results section, and comprise things such as limiting the activation function to only the most commonly used functions or assuming that the model does not need more than 175 neurons per layer.

The best way to find out what parametres fit the model best, is to try everything. And then cross validate against an independent validation data-set. As for the assumption that a shallow neural network is sufficient for approximating this specific model.

3.5.1 Choice of loss function

When choosing a loss function of a neural network, the most common thing to do would be to use the MSE, or mean squared error, as given in equation 23. In this case however, one must also take into account that the inputs and responses often vary greatly in scope, and that while some are fractions between 0 and 1, others are large duties in the scale of multiple million watts. Therefore using the MAPE, or the mean average percentage error as given in equation 24, may counteract some of this error, as the MAPE uses the relative error of every variable and may as such avoid the bias of minimizing the error of large values to the detriment of

smaller ones. As a matter of fact, since the values have been normalized, this may already be enough. Nonetheless, both loss functions will be tested.

3.5.2 Normalizing and/or standardizing the data

When training a neural network, it is very important that the input and output are the the same domains. Data that varies widely in scope would require the neural network itself to compensate and may lead to poorer accuracy. By normalizing or standardizing the data, one can avoid this incurred penalty and hopefully improve the model. Normalization is given by equation 27, the process transforms the variables to lie between 0 and 1 depending on the relative size of the untransformed variable.

$$X_{normalized} = \frac{X - X_{max}}{X_{max} - X_{min}} \tag{27}$$

In equation 28[20], the method for normalization is given. x is the value to be normalized, μ is the average of the variable and s is the sample standard deviation.

$$f(x) = \frac{1}{s\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{s})^2} \tag{28}$$

Both these methods will be used for both the input and response data, and whichever method performs the best will be used.

3.5.3 Epochs and batch size

When training a neural network, one must also specify the number of iterations, or epochs, the algorithm should run. Normally the only real problem that can be encountered is that you train the model for too few epochs, and this concern has to be balanced against the need for a quick training period. When training, the model rarely improved beyond 1200 epochs, and as such, the model was specified to 1500 epochs just to be on the safe side.

A batch is merely a group of observations used together when training an epoch. Having a small batch size is easier computationally, but can introduce erratic and unpredictable patterns into the model. Subsequently, a batch size of 120, an intermediate size was chosen, mainly as a compromise.

3.5.4 Validating the model

When validating the model, an independent dataset was used. The constraints used are the same as for the training dataset, but care was taken so that the input values would be different and the model would have fresh data to benchmark against.

4 Results and discussion

In this section, the results will be presented and discussed. Since there are a great many variables, only the most important variables will be focussed on in the name of presenting a concise and comprehensive argument.

4.1 Choosing model parametres

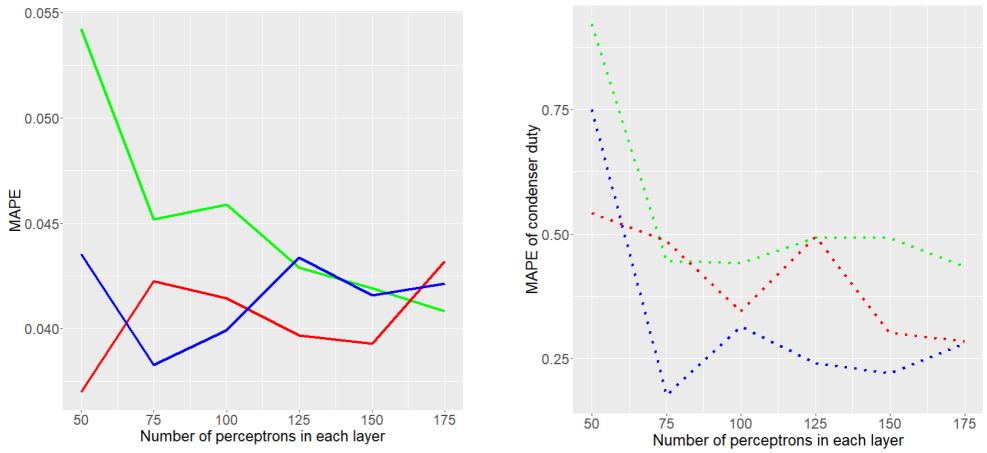
As mentioned earlier, finding the optimal model parametres is a tedious process that is best done by trial and error. The parametres to determine are: number of layers, number of neurons in each layer, the regularization method and the activation function. The results were subsequently plotted to make it easier to comprehend and view. In figure 7a the MAPE of the responses are plotted against the number of perceptrons in each layer, and in figure 7b the relative error of the condenser duty was plotted against the number of perceptrons. The reason that the condenser duty has been singled out in this way can easily be seen by viewing the y-axis of figure 7b, this specific response value proved to be very finicky and great care had to be taken in order for it to return a value with reasonable accuracy. From the total picture provided in figure 7, three layers with 75 perceptrons each then reasonable balances the need for overall accuracy with the special considerations needed for the condenser value response.

The next parametre to determine is the method for regularization and the learning rate. Here a combination of both with a learning rate of 1e-6 is the best option and successfully negates the effects of overtraining.
In table 11 the different MAPE of the selected activation functions are given. The sigmoid and the ReLU are the two best performing, and either would probably do fine. The ReLU does perform slightly better though, and as such is the one chosen for the model.

Activation function	MSE
tanh	0.050216
sigmoid	0.040004
softplus	0.084304
ReLU	0.038148

Table 11: The MSE of the different activation functions tried

When training the model, care was also taken to choose the correct metric of



(a) Relative error

(b) Residual error

Figure 7: The MAPE plotted against the number of perceptrons in one layer, two layers and three layers

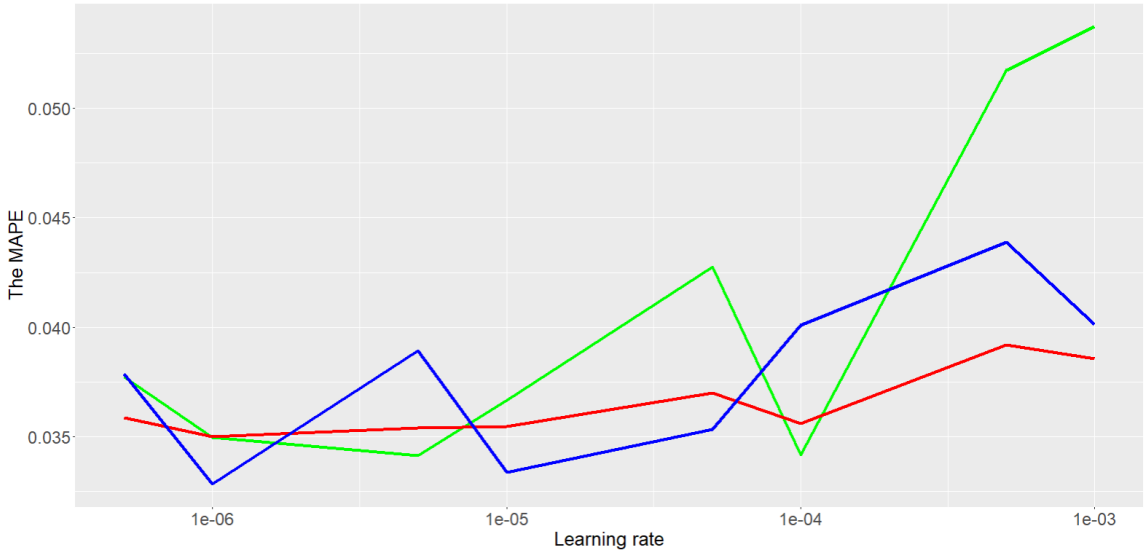


Figure 8: The learning rate plotted against the MAPE for L1, L2 and a combination of both

measuring the error. Using the MAPE often allows the training to minimize the error of all the responses, but this time, using the MAPE for training gave wildly inconsistent and erring results. To the point that taking the average would not accurately represent the chaotic results. Using the MSE as error function however gave very good results, and as such it was selected as the error function of the model.

4.2 Validation errors

In order to validate and assess the accuracy of the model, the MAPE of the different responses of the validation dataset were found. The average MAPE of the training dataset was also found, but since it is not really all that mathematically interesting the focus will instead be on the MAPE of the validation dataset. All these values are given in table 12. As mentioned in section 4.1, the condenser duty did indeed prove to be slightly less accurate when compared to the other values. However, after successfully negating the effects of overtraining by the use of regularization, the MAPE of this response variable is down to a much more managable 7.653%. The other response values are all rather impressive with regards to accuracy. Lastly the average error of the training dataset is just above 0.2%. Which, whilst not a surprise, also speaks to the accuracy of the model.

The fact that regularization had such a profound impact on the accuracy of the model seems to suggest that there was some noise in the data used to train the model. That, or the model itself failed to properly understand all the intricacies of the data, and a little regularization was just what was needed to remove the noise from the model itself. The old adage ”never look a gift horse in the mouth”, may perfectly summarise what should be done, as ascribing physical value to the training parametres of a neural network model, if even possible, rarely accomplishes anything of value.

4.3 Model training

When training the model, each epoch took roughly 500 microseconds to train, with the whole model taking approximately 45 seconds. As expected, the main

Response variable	MAPE
Reboiler duty	0.042175
HEx cooling duty	0.007927
Solvent lean loading	0.021172
Solvent rich loading	0.010085
Sweet gas flow rate	0.000445
CO_2 fraction stripper top	0.021557
Sweet gas composition H_2O	0.045977
Sweet gas composition N_2	0.010485
CO_2 captured	0.003053
Condenser duty	0.07653
ΔT Rich	0.002727
Average	0.038103
Training average	0.002293

Table 12: The different MAPE of the response variables

bottleneck is accessing and retrieving the information and not executing the calculations. Propagating a dataset with 100 observations takes approximately 4 seconds, so it can be stated with confidence that the model is both complex enough to understand the data it has been trained on, but not so complex as to become sluggish. In figure 9 the accuracy and loss of the function has been plotted against the number of epochs trained. Although it seems that the loss function quickly reaches it's minumum, that is not the case as can be seen in the model accuracy plot. Here, one can see that especially the validation accuracy improves up until about epoch 1200.

4.4 Plotting the responses of the model

In order to effectively evaluate if the model is accurate enough or not, just using the mean average percentage error may not provide the entire picture. For this purpose, three small data samples were produced by CO2Sim and then compared to the data predicted by the model. The three first values were fixed, whilst the V/L ratio, because of the nature of how CO2Sim produces it's data was left for the program to decide. The reason this specific data was chosen was simply to provide a varied data pool of different samples.

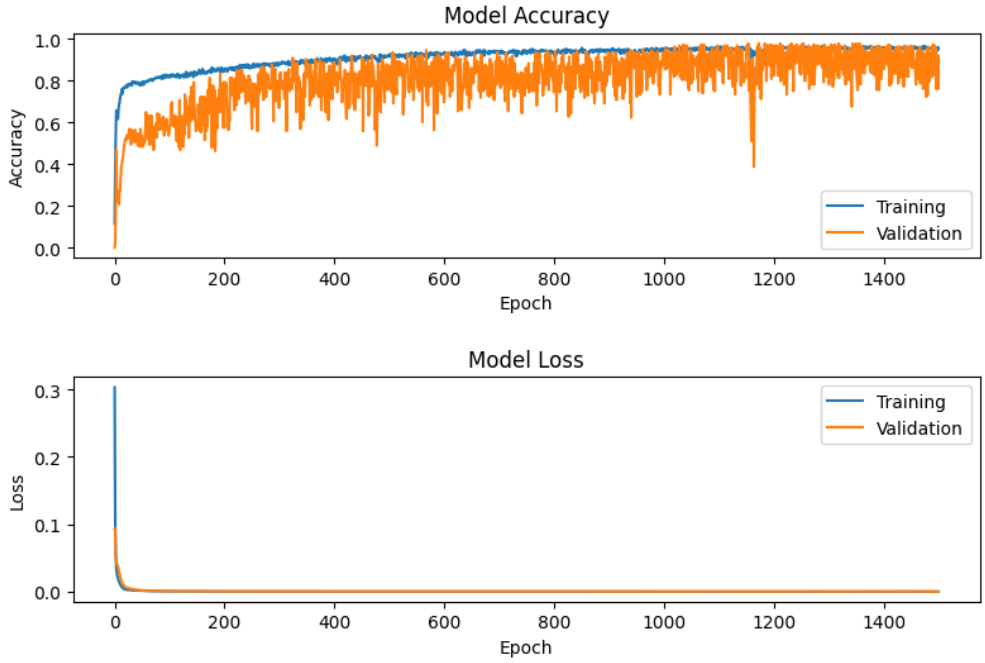


Figure 9: The accuracy and loss of the model plotted against the number of epochs trained

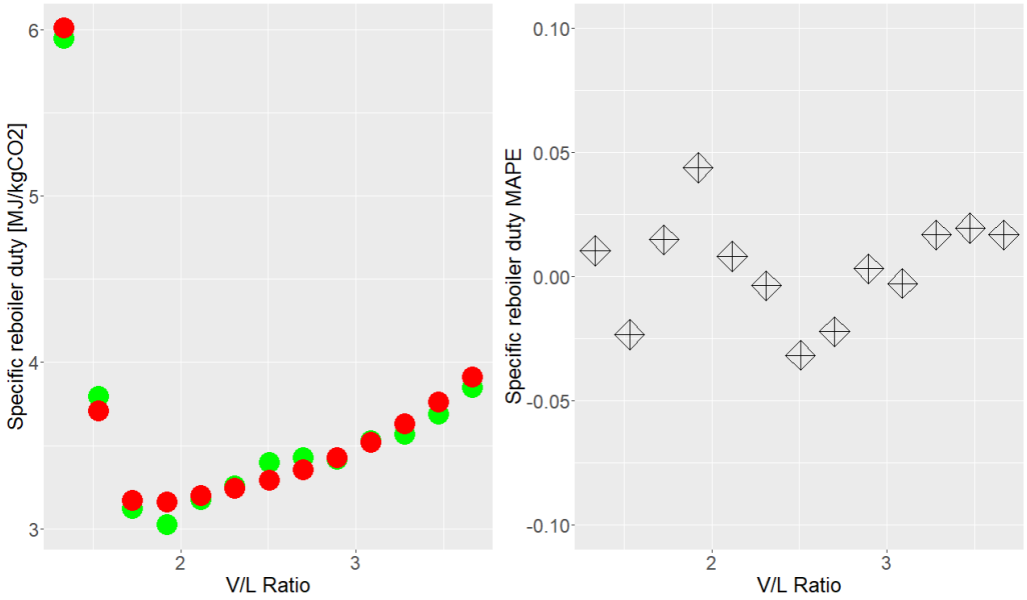


Figure 10: The specific reboiler duty plotted against the V/L ratio for the first variable combination. Predicted & CO2Sim data.

The simplest and most straightforward thing to do would of course be to simply plot the responses against a chosen input parametre, but that would leave out valuable context. So for that purpose, it was decided that the specific reboiler duty in $MJ/kgCO_2$ and the relative residual error of that value would be plotted instead. The choice of specific reboiler duty then also allows us to examine both the accuracy of the reboiler duty and the accuracy of the CO_2 capture rate in $[kg/h]$.

Variable	Value 1	Value 2	Value 3
Column Height	14	16	18
Capture rate	0.8	0.85	0.95
CO2 percentage	10	10	5
V/L ratio	viewable in plot	viewable in plot	viewable in plot

Table 13: The parametres of the inputs used

The specific reboiler duties seem reasonable when comparing them to Sakwattanapong[21] et al. 2005 and Canepa[22] et al.2015 values of respectively 4.7 MJ/kg and 4.2 MJ/kg. The values predicted by the model seem at least reasonable, and the fact that they are slightly different can probably be explained by the

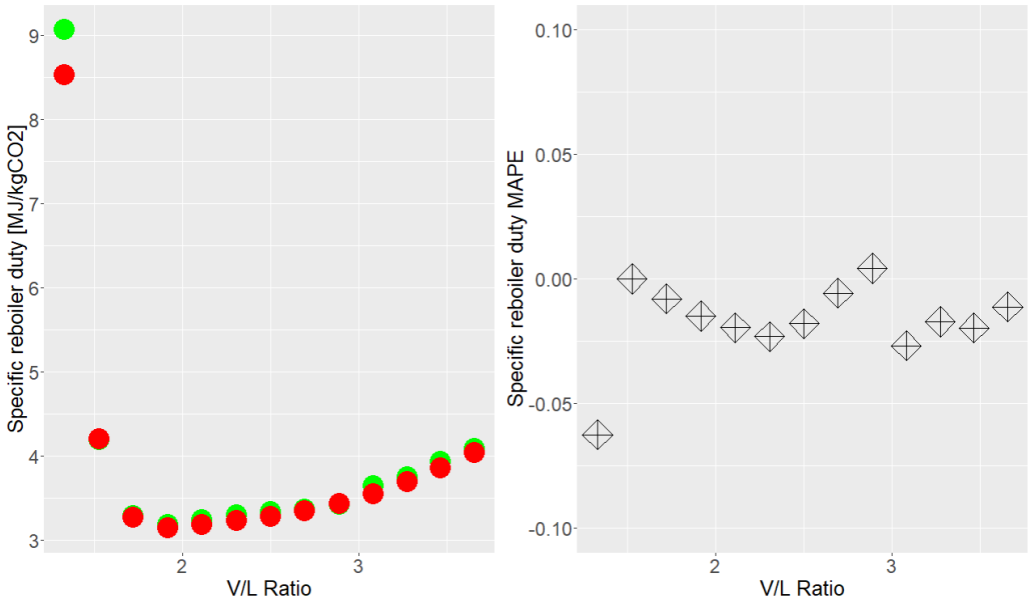


Figure 11: The specific reboiler duty plotted against the V/L ratio for the second variable combination. Predicted & CO2Sim data.

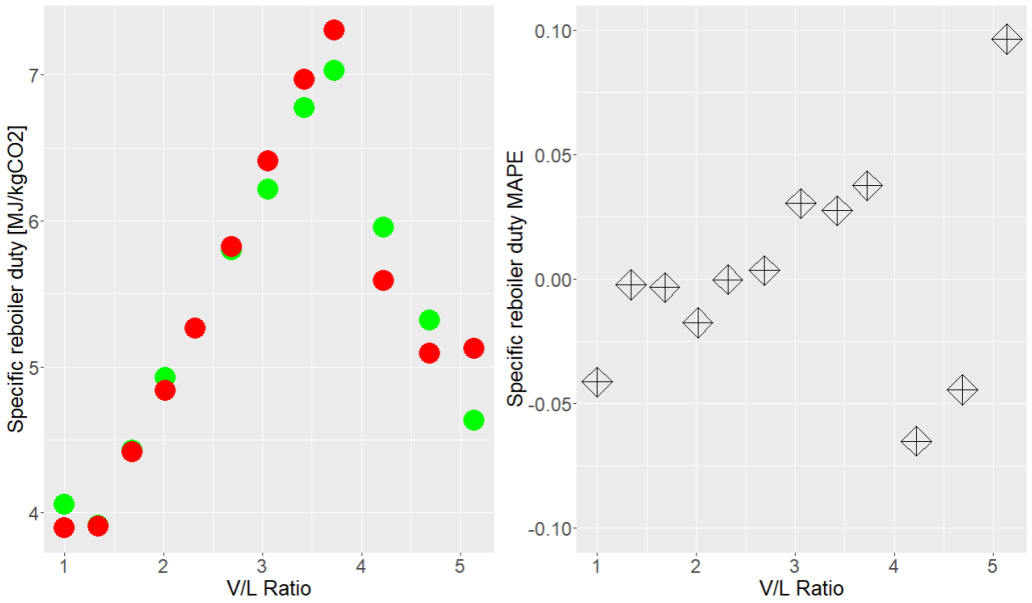


Figure 12: The specific reboiler duty plotted against the V/L ratio for the third variable combination. Predicted & CO2Sim data.

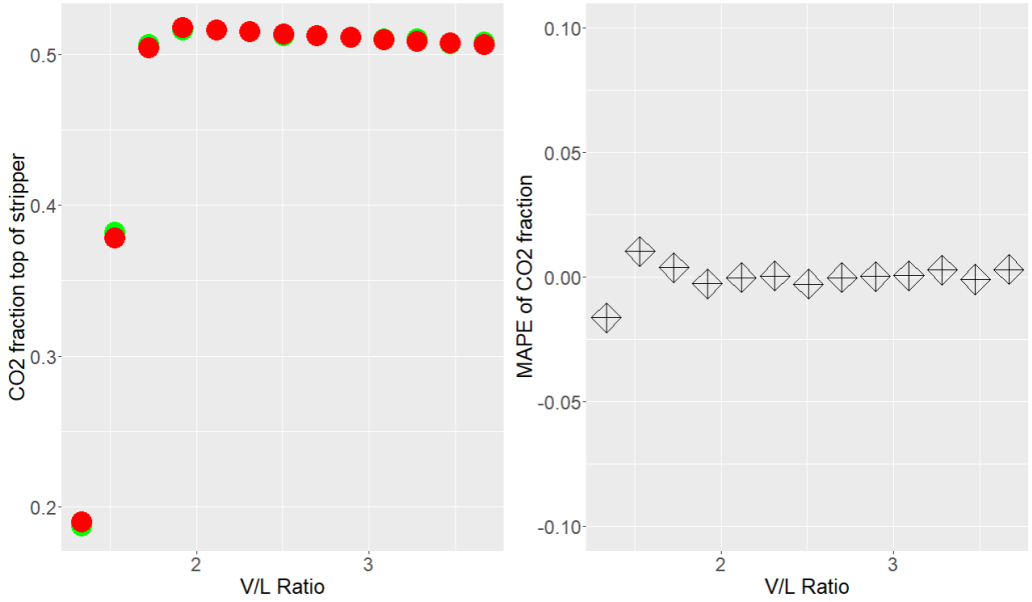


Figure 13: The CO_2 fraction plotted against the V/L ratio for the first variable combination. Predicted & CO2Sim data.

thermodynamic package used by CO2Sim, and differences in plant configuration. Now of course, this is exactly as expected since the whole model is predicated on the fact that CO2Sim is an accurate representation of a carbon capture plant.

Obviously, there are more response variables than just the reboiler duty and the CO_2 capture rate [kg/h]. Since the fraction of CO_2 at the top of the stripper column is also an important factor when it comes to designing both the stripper column and the subsequent condenser. Therefore it was decided that this response value should also be examined closer. These results can be seen in figure 13 to 15. Here one can see that the predicted results generally line up with with the values from CO2Sim, with the largest discrepancy being in figure 15 at about 6%.

From the results plotted in figure 10 to 15 one can already begin to see what plant configurations would work and which do not. At the 5% [mol] inlet, a column height of 18m is not optimal at all. This is probably because of the heat required to maintain the temperature throughout the column, and a shorter column would better serve the requirements of this plant. The heat from the reboiler also evaporates a lot of water compared to the other set-ups which again requires more

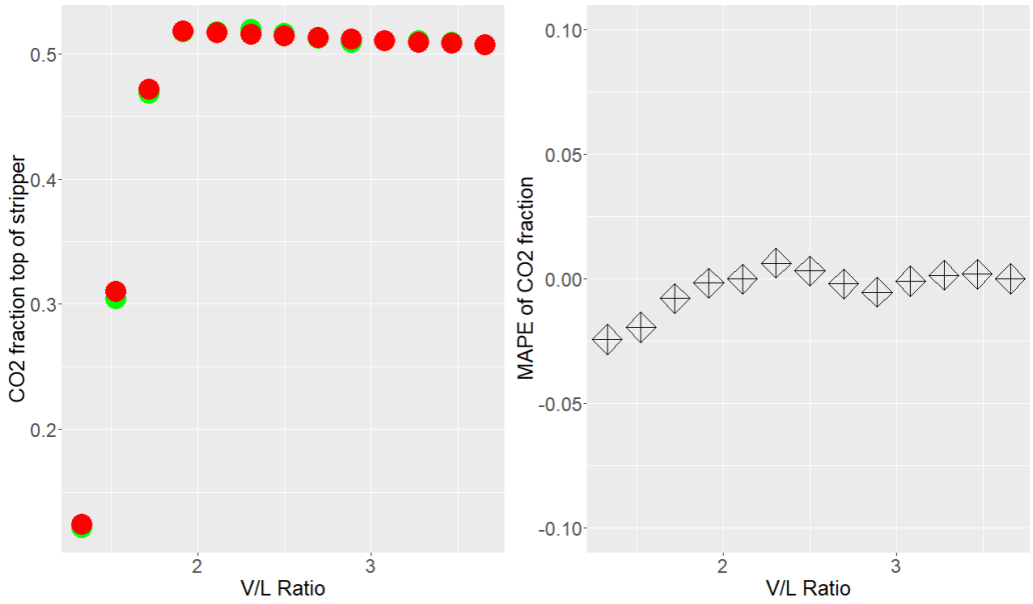


Figure 14: The CO_2 fraction plotted against the V/L ratio for the second variable combination. Predicted & CO2Sim data.

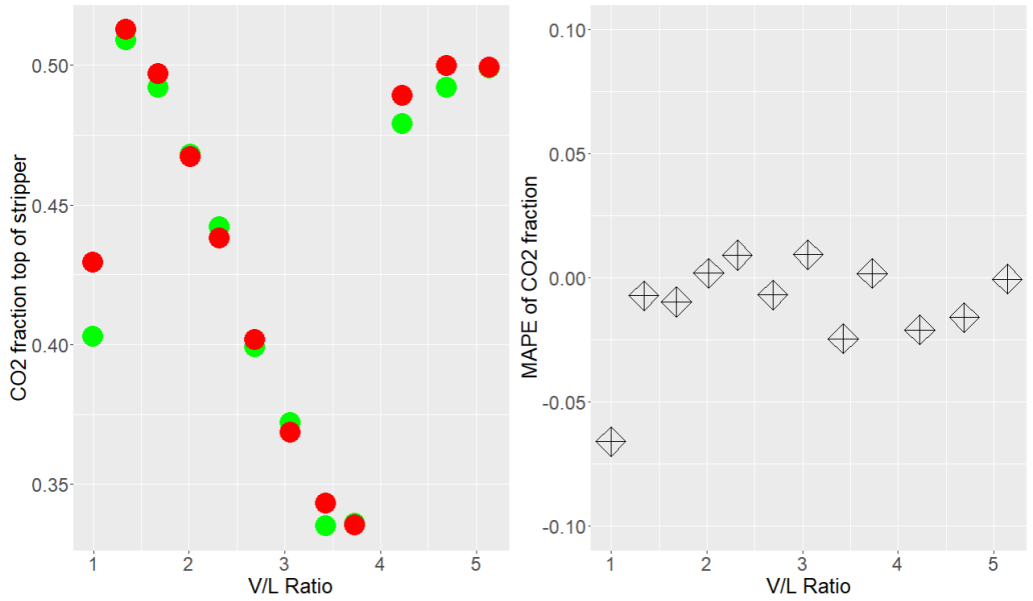


Figure 15: The CO_2 fraction plotted against the V/L ratio for the third variable combination. Predicted & CO2Sim data.

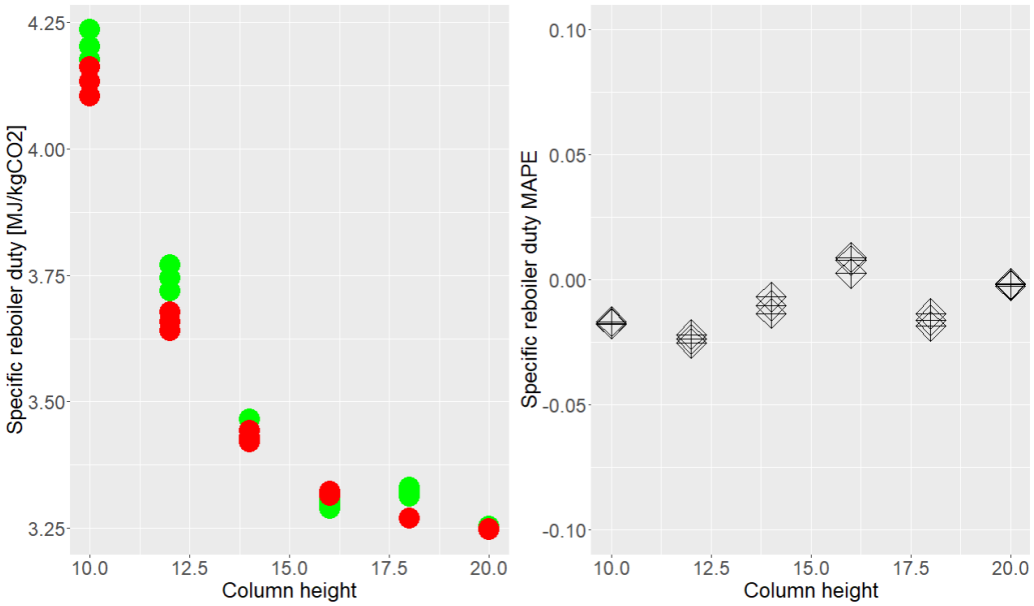


Figure 16: The specific reboiler duty plotted against the column height Predicted & CO2Sim data.

cooling in the condenser, further increasing the cost of carbon capture.

Lastly, the effect the column height has on the specific reboiler duty should also be analyzed. For the purposes of this case study, a new set of limits was found, and then run through the model in order to produce two comparable datasets. These limits are given in table 14

Variable	limits
Column height	10-20
Capture rate	0.85
CO_2 percentage	12.5
V/L ratio	2.3

Table 14: The limits of the column height case study

4.4.1 Plotting the residuals of the validation set

Indeed, there are other values than the ones analyzed in the previous section, but going into as thorough detail with every one of them would perhaps provide too

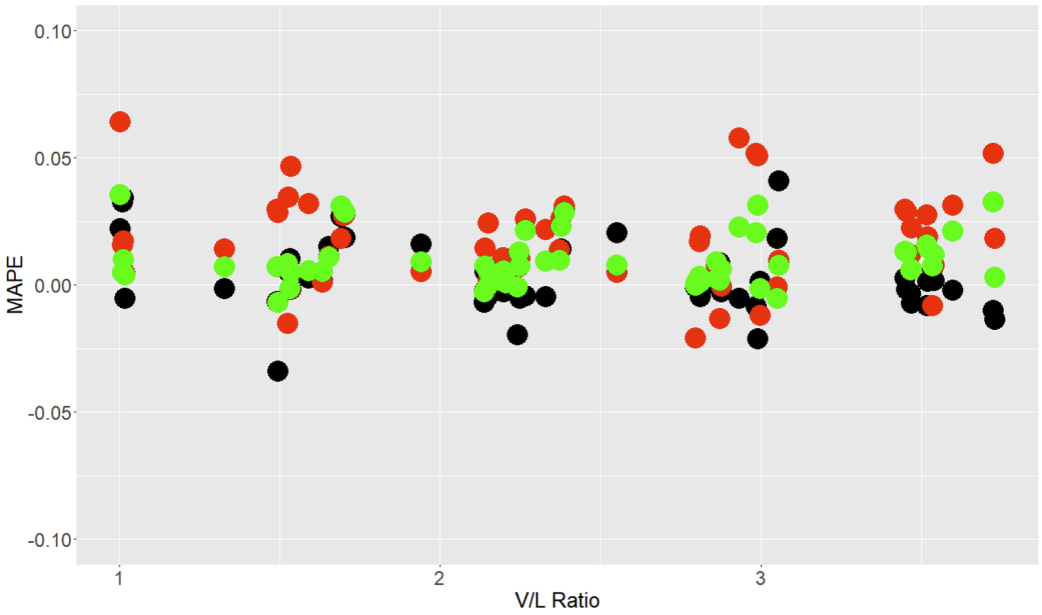


Figure 17: The relative residual error plotted against the V/L ratio for the rich loading & lean loading and **condenser duty**

much detail without actually providing clarity and insight to how the model actually behaves. Therefore, for the sake of expediency and simplicity, the relative residual errors of the condenser duty, the lean loading and lastly the rich loading would be plotted against the V/L ratio. Also, the MAPE of all the variables would also be plotted against the V/L ratio The V/L ratio was chosen simply because simply because it is convenient, and no other special reason. After all, it is the y-axis, not the x-axis that is of interest. The data used was the validation dataset that was used to find the optimal model parametres.

The MAPE for the first three variables can be seen in figure 17. Here, the prediction accuracy is quite good for all three variables, even for the previously troublesome condenser duty behaves very nicely with none of the relative residuals exceeding 7.5% error. This is presumably because the specific model used to predict these values is slightly different than the one used to find the MAPE of every variable in table 12, even though it was trained with the exact same parametres and data. It is very good however, that the relative error rarely exceeds $\pm 0.05\%$.

The MAPE for the rest of the variables can be seen in figure 18. The picture

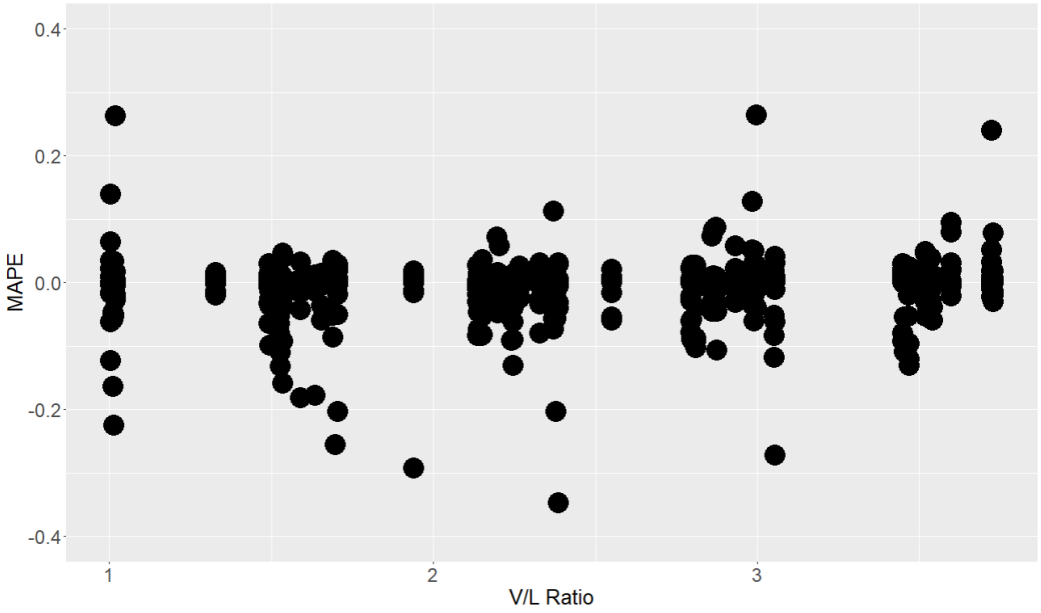


Figure 18: The relative residual error plotted against the V/L ratio for the rest of the responses

this plot paints is not as rosy as the one from figure 17, but it is nonetheless not too bad. The largest error here is about 35%, it is important to remember that it is only one value out of many. The vast majority of the responses are within a 10% margin of error. It is also important to remember that the most important response variables have all been analyzed and found to be very accurate, and that the offending value probably is a one time case from one of the other response variables. Also, when using the data provided in table 12, none of the response variables are repeat offenders when it comes to bad accuracy, and the outlier is most likely not a systematic problem with the model.

4.5 Economic analysis

When analysing the economic viability of the model a coal fired power plant[23] with an output of 250MW was used as the basis for calculations. The power plant has a flue gas flow rate of 354 m^3/s and a molar percentage of CO_2 of 12.5 %. For simplicity's sake, Norwegian prices were used, even though it is utterly improbable that a coal fired power plant be built in Norway. The purpose of this thesis is after all to determine the viability of using neural nets to model carbon capture,

and this hypothetical coal fired plant allows us to view how certain parametres affect the cost of capturing CO_2 in broad trends. Just like in the previous section on plotting the model responses, one must have an input dataset for the model to work on. And in this case study, the inputs are given in table 15. Also, as in the previous section, the cost is given as \$/ton CO_2 in order to give some context to the results.

Variable	Value 1	Value 2	Value 3
Column Height	20	20	10-20
Capture rate	0.9	0.8-0.95	0.9
CO2 percentage	12.5	12.5	12.5
V/L ratio	1.6-3.5	2.3	2.3

Table 15: The parametres of the inputs used

First out is the effect of the V/L ratio has on the cost of CO_2 removal. The cost per tonn of CO_2 removed can be seen in figure 19. As one can see, the V/L ratio does indeed have an effect on the cost, with the most glaring instance being the much higher costs at lower V/L ratios. For the rest of the V/L ratios, the results are reasonable. The results make sense, as with a lower V/L ratio, one would need a much larger difference between lean and rich loading. This requires a lot of energy to achieve. On the other hand, a higher V/L ratio, and thus a lower difference between lean and rich loading requires less energy to regenerate the solution. But sooner or later, the energy advantage is outweighed by the requirements of heating so much flow, and a larger Capex required to accommodate for the increased flow rate. The next variable was the capture rate. These results can be seen in figure 21. Here one can see again that it is the cost of steam that decides single-handedly decides if the configuration is economically feasible or not. There is a slight Capex penalty associated with the higher capture rates, but again, they are all but nullified by the cost of steam. These results also make sense, as there is generally an increased cost associated with upping the capture rate. Both when it comes the the increased reboiler duty required to regenerate the CO_2 from the amine solution and the larger Capex required to handle the increased duty.

Lastly, the effect of the column height on the cost should also be examined. Here, one can see that after a while, the lessened energy requirements that come with a taller column are balanced out by the larger Capex costs of the taller column. This

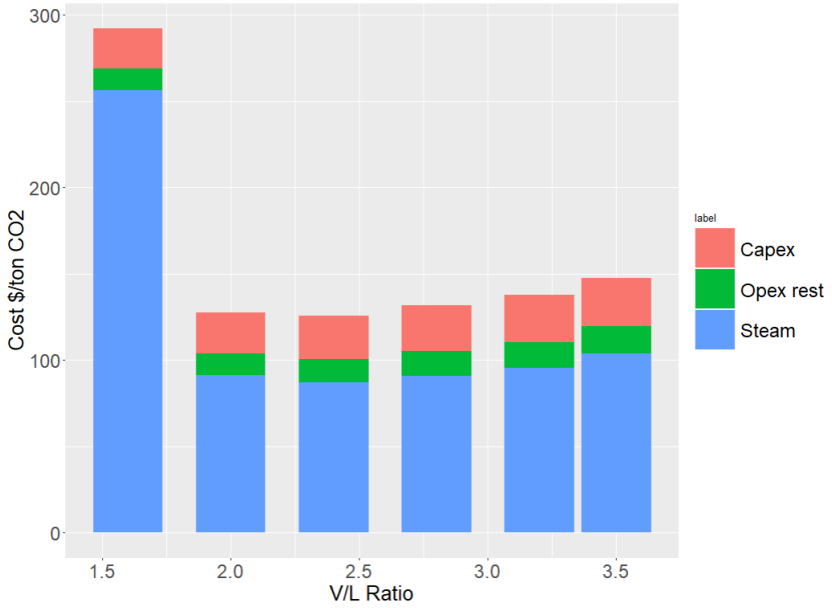


Figure 19: The cost per ton plotted against the V/L ratio

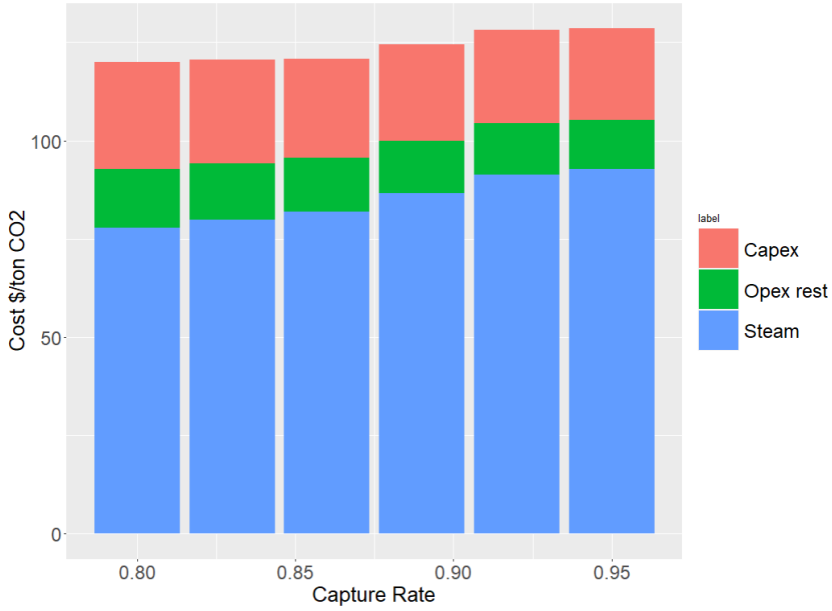


Figure 20: The cost per ton plotted against the capture rate

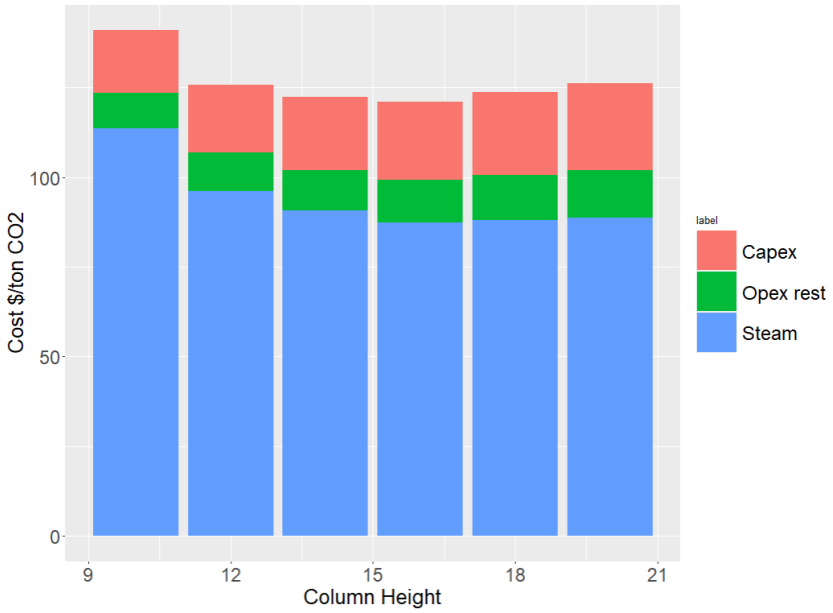


Figure 21: The cost per ton plotted against the column height

is as expected, since sooner or later the height of the column would be too high to absorb the limited amount of CO_2 in the flue gas stream.

When it comes to economic analysis, Li[24] et al.2016 gives a value range of 75.1 $/tonn to 86.4 $/tonn whilst Ali[17] et al.2019 gives us a value range of 50 €/tonn to 128 €/tonn. These literature values might seem at the lower end of what the model is predicting, but now it is important to remember that the abnormally high current prices for natural gas are incorporated into the prices the model is predicting. So when taking this into account, the costs should be quite comparable.

4.6 Further work

The model produced in this thesis should allow us to optimize a carbon capture plant with regards to capital expenditure and cost per tonn CO_2 removed. A method such as response surface method may be used to find the minimum of the cost per tonn price, or for that matter the minimum of a weighted average between the cost per ton and the total capital expenditure. This is important since not everyone can access the credit necessary to build the plant to the specifications

necessary to minimize the cost per ton, and a compromise solution may have to be used instead. Also, using a more sophisticated method for cost analysis would allow the model do more than predict trends in pricing. Namely it would allow the model to be used for accurate cost analysis and thus make the model more useful for equipment design.

Svendsen et al.2013[25] shows that the column diameter can have a significant effect on the operating expenditures of a plant and that depending on the concentration of CO_2 in the flue gas. It could then be interesting to implement the diameter of the two columns into the model as an input variable, and model this aspect of the process too. It should also be noted that an alternative to CO2Sim should be considered, as the single most time consuming part of this thesis was creating the dataset. And producing many multiple amounts of the data used in this thesis would take a prohibitively long time.

5 Conclusion

For this master thesis, a neural network model was created upon the basis of CO2Sim carbon capture simulation software. The neural network was built in Python using the keras/tensorflow package. A cost analysis was also performed on a coal fired power plant with an output of 250 MW in order to examine the effects of the input variables on the capital and operating expenditures.

As demonstrated in section 4, the model performed admirably when it came to the chosen 11 response variables, with an mean average percentage error of 3.8103%. The condenser duty was the variable with the highest MAPE, but this was only 7.653% and perfectly within the range of what should be considered reasonable. Even the cost analysis estimates were mostly within the expected interval as found in literature. The discrepancy in the cost analysis can be explained by the current prices of natural gas as they do somewhat inflate the estimates of the operating expenditures.

The neural network model takes approximately four seconds to propagate a case study file with 100 datapoints, which is a lot faster than the hours it would take compared with a traditional simulation tool. This then means that the surrogate model was both accurate enough, and as fast as one would expect of a neural network model. Thus providing a compelling alternative to the already existing solutions.

References

[1] Gelu-Adrian Chisăliţă and Raluca Moldovan. Energy and environmental strategies in the context of climate change. *Revista Romana de Inginerie Civila*, 13(1):62–71, 2022. doi: 10.1021/acs.est.5b02356. URL http://dx.doi.org/10.37789/rjce.2022.13.1.8. PMID: 26236921.

[2] Mark Jacobson. The health and climate impacts of carbon capture and direct air capture. *Energy Environ*, 12:3567, 2019. ISSN 1383-5866. doi: https://doi.org/10.1039/c9ee02709b. URL https://pubs.rsc.org/en/content/articlepdf/2019/ee/c9ee02709b.

[3] Gary T. Rochelle. Amine scrubbing for co2 capture. *Science*, 325(5948): 1652–1654, 2009. ISSN 0036-8075. doi: 10.1126/science.1176731. URL https://science.sciencemag.org/content/325/5948/1652.

[4] James Yeh, Henry Pennline, and Kevin Resnik. Study of co 2 absorption and desorption in a packed column. *Energy & Fuels - ENERGY FUEL*, 15: 274–278, 03 2001. doi: 10.1021/ef0002389.

[5] M. Wang, A. Lawal, P. Stephenson, J. Sidders, and C. Ramshaw. Post-combustion co2 capture with chemical absorption: A state-of-the-art review. *Chemical Engineering Research and Design*, 89(9):1609–1624, 2011. ISSN 0263-8762. doi: https://doi.org/10.1016/j.cherd.2010.11.005. URL https://www.sciencedirect.com/science/article/pii/S0263876210003345. Special Issue on Carbon Capture & Storage.

[6] Patricia Luis. Use of monoethanolamine (mea) for co2 capture in a global scenario: Consequences and alternatives. *Desalination*, 380:93–99, 2016. ISSN 0011-9164. doi: https://doi.org/10.1016/j.desal.2015.08.004. URL https://www.sciencedirect.com/science/article/pii/S001191641500418X.

[7] Bihong Lv, Bingsong Guo, Zuoming Zhou, and Guohua Jing. Mechanisms of co2 capture into monoethanolamine solution with different co2 loading during the absorption/desorption processes. *Environmental Science & Technology*, 49(17):10728–10735, 2015. doi: 10.1021/acs.est.5b02356. URL https://doi.org/10.1021/acs.est.5b02356. PMID: 26236921.

[8] Ray Sinnott and Gavin Towler. In Ray Sinnott and Gavin Towler, editors, *Chemical Engineering Design (Sixth Edition)*, Chemical Engineering Series, pages 75–140. Butterworth-Heinemann, sixth edition edition, 2020. ISBN

978-0-08-102599-4. doi: https://doi.org/10.1016/B978-0-08-102599-4.00003-5. URL https://www.sciencedirect.com/science/article/pii/B9780081025994000035.

[9] Adisorn Aroonwilas and Amornvadee Veawab. Characterization and comparison of the co2 absorption performance into single and blended alkanolamines in a packed column. *Industrial & Engineering Chemistry Research*, 43(9):2228–2237, 2004. doi: 10.1021/ie0306067. URL https://doi.org/10.1021/ie0306067.

[10] C.J. Geankoplis. *Transport Processes and Separation Process Principles: Includes Unit Operations*. Prentice Hall Professional technical reference. Prentice Hall Professional Technical Reference, 2003. ISBN 9780131013674. URL https://books.google.no/books?id=i9-TQgAACAAJ.

[11] J. H. Friedman, R. Tibshirani, and T. Hastie. *Elements of Statistical Learning*. Springer, 1st edition, 2001.

[12] Griet Heuvelmans, Bart Muys, and Jan Feyen. Regionalisation of the parameters of a hydrological model: Comparison of linear regression models with artificial neural nets. *Journal of Hydrology*, 319(1):245–265, 2006. ISSN 0022-1694. doi: https://doi.org/10.1016/j.jhydrol.2005.07.030. URL https://www.sciencedirect.com/science/article/pii/S0022169405003641.

[13] Ciaburro and Venkateswaran. *Neural Networks in R*. Packt Publishing, 1st edition, 2017.

[14] D. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *ICLR 2015*, 2015.

[15] Finn Andrew Tobiesen, Hallvard F. Svendsen, and Olav Juliussen. Experimental validation of a rigorous absorber model for co2 postcombustion capture. *AIChE Journal*, 53(4):846–865, 2007. doi: https://doi.org/10.1002/aic.11133. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.11133.

[16] Solomon Aforkoghene Aromada, Nils Henrik Eldrup, Fredrik Normann, and Lars Erik Øi. Techno-economic assessment of different heat exchangers for co2 capture. *Energies*, 13(23), 2020. ISSN 1996-1073. doi: 10.3390/en13236315. URL https://www.mdpi.com/1996-1073/13/23/6315.

[17] Hassan Ali, Nils Henrik Eldrup, Fredrik Normann, Ragnhild Skagestad, and Lars Erik Øi. Cost estimation of co2 absorption plants for co2 mitigation – method and assumptions. *International Journal of Greenhouse Gas Control*, 88:10–23, 2019. ISSN 1750-5836. doi: https://doi.org/10.1016/j.ijggc.2019.05.028. URL https://www.sciencedirect.com/science/article/pii/S1750583618309332.

[18] R H Weiland, J C Dingman, and D B Cronin. Heat capacity of aqueous monoethanolamine, diethanolamine, n-methyldiethanolamine, and n-methyldiethanolamine-based blends with carbon dioxide. *Journal of Chemical and Engineering Data*, 42(5), 9 1997. doi: 10.1021/je960314v. URL https://www.osti.gov/biblio/556797.

[19] Jana Jakobsen, Simon Roussanaly, and Rahul Anantharaman. A techno-economic case study of co2 capture, transport and storage chain from a cement plant in norway. *Journal of Cleaner Production*, 144:523–539, 2017. ISSN 0959-6526. doi: https://doi.org/10.1016/j.jclepro.2016.12.120. URL https://www.sciencedirect.com/science/article/pii/S0959652616321837.

[20] Carl Friederich Gauss. *Theoria combinationis observationum erroribus minimis obnoxiae*. 1823.

[21] Roongrat Sakwattanapong, Adisorn Aroonwilas, and Amornvadee Veawab. Behavior of reboiler heat duty for co2 capture plants using regenerable single and blended alkanolamines. *Industrial & Engineering Chemistry Research*, 44(12):4465–4473, 2005. doi: 10.1021/ie050063w. URL https://doi.org/10.1021/ie050063w.

[22] Roberto Canepa and Meihong Wang. Techno-economic analysis of a co2 capture plant integrated with a commercial scale combined cycle gas turbine (ccgt) power plant. *Applied Thermal Engineering*, 74:10–19, 2015. ISSN 1359-4311. doi: https://doi.org/10.1016/j.applthermaleng.2014.01.014. URL https://www.sciencedirect.com/science/article/pii/S1359431114000209. 6th International Conference on Clean Coal Technologies CCT2013.

[23] Chao Wang, A. Frank Seibert, and Gary T. Rochelle. Packing characterization: Absorber economic analysis. *International Journal of Greenhouse Gas Control*, 42:124–131, 2015. ISSN 1750-5836. doi: https://doi.org/10.1016/

j.ijggc.2015.07.027. URL https://www.sciencedirect.com/science/article/pii/S1750583615300347.

[24] Kangkang Li, Wardhaugh Leigh, Paul Feron, Hai Yu, and Moses Tade. Systematic study of aqueous monoethanolamine (mea)-based co2 capture process: Techno-economic assessment of the mea process and its improvements. *Applied Energy*, 165:648–659, 2016. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2015.12.109. URL https://www.sciencedirect.com/science/article/pii/S0306261915016827.

[25] Neda Razi, Hallvard F. Svendsen, and Olav Bolland. Cost and energy sensitivity analysis of absorber design in co2 capture with mea. *International Journal of Greenhouse Gas Control*, 19:331–339, 2013. ISSN 1750-5836. doi: https://doi.org/10.1016/j.ijggc.2013.09.008. URL https://www.sciencedirect.com/science/article/pii/S1750583613003411.

A Appendix

A.1 Training the neural network

This is the code for training the surrogate model. The code takes the training data and the validation data as input. And subsequently saves the neural network as a .h5 file. The code also prints the accuracy and training parametres.

```
#Import libraries
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras import regularizers
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from numpy import loadtxt
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.pipeline import Pipeline

# load dataset for both input var and response var
input_dataframe = loadtxt("combinput_final.csv", delimiter=",
                          ", skiprows=1)
response_dataframe = loadtxt("combresponse_final.csv",
                          delimiter=",", skiprows=1)

#Importing the validation dataset
input_validation = loadtxt("input_validation_final.csv",
                          delimiter=",", skiprows=1)
response_validation = loadtxt("response_validation_final.csv"
                          , delimiter=",", skiprows=1)

#Standardizing the input dataset
scalerNormal = StandardScaler()
scalerNormal.fit(input_dataframe)
input_dataframe = scalerNormal.transform(input_dataframe)

#Standardizing the response dataset
scalerMinMax = MinMaxScaler()
scalerMinMax.fit(response_dataframe)
response_dataframe = scalerMinMax.transform(
                          response_dataframe)

#Transforming the validation data
input_validation = scalerNormal.transform(input_validation)
```

```
response_validation = scalerMinMax.transform(
                                response_validation)

#Defining the keras model
model = Sequential()

model = Sequential()
model.add(Dense(75, activity_regularizer=regularizers.l1_l2(
                                l1=1e-6, l2=1e-6), input_dim=4
                                , activation='relu'))
model.add(Dense(75, activity_regularizer=regularizers.l1_l2(
                                l1=1e-6, l2=1e-6), activation=
                                'relu'))
model.add(Dense(75, activity_regularizer=regularizers.l1_l2(
                                l1=1e-6, l2=1e-6), activation=
                                'relu'))
model.add(Dense(11, activation='linear'))


#Compile the keras model
model.compile(loss='mse', optimizer='adam', metrics=['MSE'])

# fit the keras model on the dataset
history = model.fit(input_dataframe, response_dataframe,
                                epochs=1500, batch_size=150)

# evaluate the keras model
_, accuracy = model.evaluate(input_dataframe,
                                response_dataframe)
print('Accuracy: %.5f' % (accuracy*100))

def individual_error_training(n):
    residuals = []
    difference_array = np.subtract(response_validation[:, n],
                                    response_predicted[:, n])
    absolute_array = abs((difference_array/response_predicted
                                    [:,n]))
    residuals.append(absolute_array)
    return np.mean(residuals)


#Evaluating the validation data
_, accuracy = model.evaluate(input_validation,
                                response_validation)
print('The validation error is: %.5f' % (accuracy*100))
```

```
#Predicting new values from the valdiation data
response_predicted = model.predict(input_validation)

#Rescaling the output
response_predicted = scalerMinMax.inverse_transform(
                                response_predicted)
response_validation = scalerMinMax.inverse_transform(
                                response_validation)

#Finding the MAPE the hard way instead of using a for loop
def individual_error(n):
    residuals = []
    difference_array = np.subtract(response_validation[:, n],
                                   response_predicted[:, n])
    absolute_array = abs((difference_array/response_predicted
                                  [:,n]))
    residuals.append(absolute_array)
    return np.mean(residuals)


for i in range(11):
    print(round(individual_error(i), 5))

import matplotlib.pyplot as plt

#This code snipped prints out the training data
plt.subplot(211)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='lower right')

# summarize history for loss
plt.subplot(212)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Training', 'Validation'], loc='upper right')
plt.tight_layout()
plt.show()
```

```
#This line saves the model for later use
model.save('saved_model/model_co2.h5')
```

A.2 The code used to perform the sizing of the process equipment

A python function was created for every component in the capture plant in order to properly size every piece of equipment.

```
#Creating a function for sizing analysis of each equipment

def cost_analysis_column(input, scaler):
    #The input is the height of the column, also assume that
                                    the inlets and outlets are
                                     at the very ends of the
                                    column
    #Also assume 5/8'' or ca 1.6cm is a good shell thickness
    #remember that the packing is already priced for 304 SS
    #For scaling the stripper column, both inputs should be
                                    scaled by 0.7. For the
                                    diameter, it is merely an
                                    approximation that mostly
                                    holds
    #based on the average gas flow rate from reboiler at the
                                    different v/l ratio
    flow_rate = (7024 * scaler) / 1.12
    radius = ((flow_rate / 3600) / (2 * 3.1415)) ** (1 / 2)
    Shell_volume = 3.1415 * radius ** 2 * input - (3.1415 * (
                                    radius - 0.016) ** 2 * (
                                    input - 0.032))
    Shell_weight = Shell_volume * 7500  # kg/m3
    packing_volume = 3.1415 * (radius - 0.016) ** 2 * (input
                                    - 0.032)
    packing_cost = packing_volume * 7600
    shell_cost = 17400 + 79 * Shell_weight ** 0.85
    total_cost = shell_cost + packing_cost
    return total_cost

def cost_analysis_hex(VLRatio, tempdiff):
    gas_flow = 7600
    spec_heat = 4
    Q = VLRatio*gas_flow*spec_heat*tempdiff
    heat_coeff = 0.500*3600
    log_mean = 10.624
    area = Q/(heat_coeff*log_mean)
    return area

def compressor_analysis(input, scaler):
```

```
    # this function takes the v/l ratio as input, efficiency
                                    is set as 0.7
    fluid_flow = input * scaler * 7600  # kg/h
    power = (fluid_flow * 9.81 * 81.5795) / (3.6 * 10 ** 6 *
                                    0.7)
    return power

def cost_analysis_reboiler(duty, vlratio):
    new_duty = duty
    U = 0.800 * 3600
    spec_heat_steam = 2.000  # kj/kgK
    spec_heat_amine = 4.000  # kj/kgK
    # Finding the necessary flow rate of steam
    flow_heat_amine = 7100 * scaler * vlratio *
                                    spec_heat_amine
    flow_mass_steam = new_duty / (spec_heat_steam * 15)
    temp_in_steam = 525.15
    temp_out_steam = 410
    temp_out_amine = 395
    deltaT_amine = new_duty / flow_heat_amine
    temp_in_amine = temp_out_amine - deltaT_amine
    deltaT1 = temp_in_steam - temp_out_amine
    deltaT2 = temp_out_steam - temp_in_amine
    logT = (deltaT1 - deltaT2) / numpy.log(deltaT1 / deltaT2)
    area = new_duty / (U * logT)
    return numpy.array([area, flow_mass_steam])

def cost_analysis_condenser(duty, co2captured, co2fraction):
    U = 1.000 * 3600
    new_duty =  duty * 1000
    flow_rate_gas = co2captured / co2fraction  # kg/h
    # need to find the molar flow rate of both co2 and h20 in
                                     order to find the amount
                                    of water condensed (by
                                    using vapour pressure of
                                    h2o at 29degrees)
    co2_molarflow = co2captured / 44.01  # kmol/h
    h20_molarflow = (flow_rate_gas - co2captured) / 18.02  #
                                    kmol/h
    # finding amount of water in the gas stream after
                                    condensation
    h2o_post_condensation = co2_molarflow / 0.9276316 -
                                    co2_molarflow
    h2o_condensed = (h20_molarflow - h2o_post_condensation) *
                                     18.02  # kg/h
```

```
    # Thermodynamics of the heat exchanger
    specific_heat_gas = co2fraction * 0.918 + (1 -
                                co2fraction) * 2  # Using
                                spec heat for co2 gas and
                                vapour
    specific_heat_water = 4.18  # kJ/kgK
    enthalpy_phase_change = 2.26 * 1000  # kJ/kg
    heat_condensation = h2o_condensed * enthalpy_phase_change
    heat_tempchange = new_duty - heat_condensation
    # Finding temps
    gas_temp_out = 302.15  # K
    water_temp_in = 288.15  # K
    water_temp_out = 313.15  # k
    # need to find the flow of water necessary
    flow_water = new_duty / (specific_heat_water * 10)
    gas_temp_in = gas_temp_out + numpy.abs(heat_tempchange /
                                (flow_rate_gas *
                                specific_heat_gas))
    deltaT1 = gas_temp_in - water_temp_out
    deltaT2 = gas_temp_out - water_temp_in
    Tlm = (deltaT1 - deltaT2) / numpy.log(deltaT1 / deltaT2)
    area = new_duty / (U * Tlm)
    return numpy.array([area, flow_water])

def HexCooler_analysis(duty, vlratio):
    new_duty = duty *1000
    gas_flow = 7600 #kg/h
    spec_heat = 4 #kJ/kg
    water_temp_in = 15 #C
    water_temp_out = 40 #C
    flow_water = new_duty/(15*4)
    amine_temp_in = 119.36 #C
    amine_temp_out = amine_temp_in - new_duty/(gas_flow*
                                vlratio*spec_heat)
    deltaT1 = amine_temp_in - water_temp_out
    deltaT2 = amine_temp_out - water_temp_in
    logT = (deltaT1-deltaT2)/numpy.log(deltaT1/deltaT2)
    heat_coeff = 0.500 * 3600
    area = abs(new_duty/(heat_coeff*logT))
    return numpy.array([area, flow_water])

def compressor2_analysis(co2captured, scaler):
    # efficiency is set as 0.7
    fluid_flow = co2captured * scaler  # kg/h
    power =  (fluid_flow * 9.81 * 1539.77148) / (3.6 * 10 **
```

```
                                    6 * 0.7) #kW
    return power #kW

def CompressorCooler_analysis(co2captured, power):
    #assume that all the increases in internal energy come
                                    from the work done by the
                                    compressor
    flow_co2_mass = co2captured
    #finding the heat of condensation
    molar_flow = flow_co2_mass/44.01 #kmol/h
    heat_of_condensation = molar_flow * 16.7 * 1000 #kJ/h
    duty = heat_of_condensation + 3600 * power #kJ/h
    spec_heat_co2 = 0.918 #kJ/kgK
    temp_co2_out = 29 #celsius
    temp_water_in = 15 #celsius
    temp_water_out = 25 #celsius
    flow_water = duty/(4.18*10) #kg/h Also assumes that the
                                    power is given in kW and
                                    not kJ/h
    temp_co2_in = temp_co2_out + (duty)/(flow_co2_mass*
                                    spec_heat_co2)
    deltaT1 = temp_co2_in - temp_water_out
    deltaT2 = temp_co2_out - temp_water_in
    heat_coeff = 0.5 * 3600 #kJ/h
    logT = (deltaT1-deltaT2)/numpy.log(deltaT1/deltaT2)
    area = (duty) / (heat_coeff * logT)
    return numpy.array([area, flow_water])

def MEA_amount(scaler, height):
    flow_rate = (7024 * scaler) / 1.12
    radius = ((flow_rate / 3600) / (2 * 3.1415)) ** (1 / 2)
    flow = radius ** 2 * 3.1415 * height
    holdup = (0.05*flow + 0.7*0.05*flow)/30 #divided by 30
                                    because of an assumed
                                    residence time of 2
                                    minutes
    MEA = holdup * 1.4 #add 10\% for every process unit in
                                    the loop
    return MEA
```

A.3 The code used to predict new data and perform the capital expenditure analysis

This code takes the .h5 neural network model, the scaling factor and the functions defined in the code above, and returns the capital expenditure as well as the consumption of all utilities. It then exports this information as .csv files.

```
#Importing the necessary libraries & functions
import tensorflow as tf
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from numpy import loadtxt, ndarray
import numpy as np
from Functions import cost_analysis_hex
from Functions import compressor_analysis
from Functions import compressor2_analysis
from Functions import cost_analysis_column
from Functions import cost_analysis_condenser
from Functions import cost_analysis_reboiler
from Functions import HexCooler_analysis
from Functions import CompressorCooler_analysis
from Functions import MEA_amount


#finding the correct scalers for inputs and responses
input_dataframe = loadtxt("combinput_final.csv", delimiter=",
                          ", skiprows=1)
response_dataframe = loadtxt("combresponse_final.csv",
                          delimiter=",", skiprows=1)
scalerNormal = StandardScaler()
scalerNormal.fit(input_dataframe)
scalerMinMax = MinMaxScaler()
scalerMinMax.fit(response_dataframe)

#Loading the model
new_model = tf.keras.models.load_model('saved_model/model_co2
                          .h5')
#Loading new input
input = loadtxt("input_plotting.csv", delimiter=",", skiprows
                          =1)
#Scaling new input
input_scaled = scalerNormal.transform(input)
#Predicting the responses
responses_scaled = new_model.predict(input_scaled)
#Scaling the responses back
```

```
responses = np.array(scalerMinMax.inverse_transform(
                                responses_scaled))

#exporting the responses
np.savetxt("responses_economics.csv", responses, delimiter=",
                                ")
print(responses)
#Cost analysis section
scaler = 196.36

#Columns
for i in input[:,0]:
    cost_ss_absorber = cost_analysis_column(input[:,0],
                                scaler)
    cost_ss_desorber = cost_analysis_column(0.7*input[:,0],
                                scaler)
#Heat exchangers
for i in responses[:,0]:
    area_hex = cost_analysis_hex(input[:,3], scaler,
                                responses[:,10])
    cost_cs_hex = 28000 + scaler*54*area_hex**1.2
#Compressors
for i in input[:,3]:
    power_compressor1 = compressor_analysis(input[:,3],
                                scaler)
    cost_cs_compressor1 = 580000 + 20000*power_compressor1**0
                                .6
    power_compressor2 = compressor2_analysis(responses[:,7],
                                scaler)
    cost_cs_compressor2 = 580000 + 20000*power_compressor2**0
                                .6
#Hex Cooler
for i in responses[:,1]:
    area_hex2 = HexCooler_analysis(-responses[:, 1], input[:,
                                3])
    cost_cs_hex2 = 28000 + scaler * 54 * area_hex2**1.2
#Reboiler
for i in responses[:,0]:
    area_reboiler = cost_analysis_reboiler(scaler, responses[
                                :,0], input[:,3])
    cost_cs_reboiler = 29000 + scaler * 400*area_reboiler[0,:
                                ]**0.9
#Condenser
for i in responses[:,9]:
    area_condenser = cost_analysis_condenser( -responses[:,9]
```

```
                                    , responses[:,8],
                                    responses[:,5])
    cost_cs_condenser = 28000 + scaler*54*area_condenser[0,:]
                                    **1.2
#Product cooler
for i in responses[:,8]:
    area_CCooler = CompressorCooler_analysis(responses[:,7],
                                    scaler, power_compressor2)
    cost_cs_CCooler = 28000 + scaler * 54*area_CCooler[0,:]
                                    ** 1.2

#Dividing the cost of the different variables into Cost
                                carbon steel, Cost stainless
                                steel, Cost after installation
                                , Cost after location &
                                inflation and operating costs

cost_CS = np.array([cost_cs_hex, cost_cs_compressor1,
                                cost_cs_compressor2,
                                cost_cs_hex2[0,:],
                                cost_cs_reboiler,
                                cost_cs_condenser,
                                cost_cs_CCooler])
cost_SS = np.array([cost_ss_absorber, cost_ss_desorber])

#Adjusting for material cost
cost_CS = np.multiply(cost_CS, 1.3)
cost_SS = (cost_SS, cost_CS)
cost_SS = np.array(cost_SS)

#Taking into account the different installation factor
f_piping = 0.8
f_rest = 1.4/1.3 #This is essentially the rest of the
                                installation factors (
                                excluding the piping)
                                corrected for the cost of
                                stainless steel relative to
                                carbon steel
f_cumulative = 1 + f_piping + f_rest
cost_installed = np.multiply(cost_SS, f_cumulative)

#Adjusting for location data
cost_location = np.multiply(cost_installed, 1.26)

#Adjusting for inflation
```

```
cost_final = np.multiply(cost_location, 1.27)

#Find the consumption of utilities
steam_consumption = np.array([area_reboiler[1,:]])
electricity_consumption = np.array([[power_compressor1], [
                                power_compressor2]])
coolingwater_consumption = ([[area_condenser[1,:]], [
                                area_hex2[1,:]], [area_CCooler
                                [1,:]]])

#Finding the amount of the process fluid, 13% of which is MEA
process_fluid_amount = MEA_amount(scaler, input[:,0])

#Exporting the data for later further cost analysis
np.savetxt("cost_final_columns.csv", cost_final[0], delimiter
                                =",")
np.savetxt("cost_final_rest.csv", cost_final[1], delimiter=",
                                ")
np.savetxt("steam_consumption.csv", steam_consumption,
                                delimiter=",")
np.savetxt("electricity_consumption1.csv",
                                electricity_consumption[0],
                                delimiter=",")
np.savetxt("electricity_consumption2.csv",
                                electricity_consumption[1],
                                delimiter=",")
np.savetxt("coolingwater_consumption.csv",
                                coolingwater_consumption[0],
                                delimiter=",")
np.savetxt("coolingwater_consumption2.csv",
                                coolingwater_consumption[1],
                                delimiter=",")
np.savetxt("coolingwater_consumption3.csv",
                                coolingwater_consumption[2],
                                delimiter=",")
np.savetxt("process_fluid.csv", process_fluid_amount,
                                delimiter=",")
```

A.4 The code used to perform the operating expenditure analysis of the plant

This code imports the .csv files produced above and then performs the cost analysis and plots the results.

```
#Packages
library(ggplot2)
library(patchwork)

#Reading in the data
cooling_water_01 <- as.matrix(read.csv(file="coolingwater_consumption.csv"))
cooling_water_02 <- as.matrix(read.csv(file="coolingwater_consumption2.csv"))
cooling_water_03 <- as.matrix(read.csv(file="coolingwater_consumption3.csv"))
electricity_01 <- as.matrix(read.csv(file="electricity_consumption1.csv"))
electricity_02 <- as.matrix(read.csv(file="electricity_consumption2.csv"))
cost_columns <- as.matrix(read.csv(file="cost_final_columns.csv"))
cost_rest <- as.matrix(read.csv(file="cost_final_rest.csv"))
steam <- as.matrix(read.csv(file="steam_consumption.csv"))
process_fluid <- as.matrix(read.csv(file="process_fluid.csv"))
input <- as.matrix(read.csv(file="input_plotting.csv"))

#The prices of different opex expenditures (From Aromada et al. 2020, or SSB) Values in USD 2020
labour_operator = 95375
labour_engineer = 185794
cooling_water_price = 0.025
process_water_price = 0.24
steam_price = 67.15 #USD/ton, using gas prices and conversion rates 13/01
electricity_price = 0.081 #kwH using industrial prices Norway 3Q 2021
MEA_price = 1798 #m3 density = 1100kg/m3
MEA_degradation_rate = 0.210 #kg/tonne co2 from Moser et al. 2020
process_fluid_cost = t(process_fluid*0.87*process_water_price + process_fluid*0.13*MEA_price)

#Start adding the different cases together
```

```
29
30 #Capex
31 cost <- rbind(cost_columns, cost_rest)
32 cost1 <- sum(cost[,1])
33 cost2 <- sum(cost[,2])
34 cost3 <- sum(cost[,3])
35 cost4 <- sum(cost[,4])
36 cost5 <- sum(cost[,5])
37 cost6 <- sum(cost[,6])
38 cost7 <- sum(cost[,7])
39 cost8 <- sum(cost[,8])
40 cost9 <- sum(cost[,9])
41 cost10 <- sum(cost[,10])
42 cost11 <- sum(cost[,11])
43 cost12 <- sum(cost[,12])
44 cost13 <- sum(cost[,13])
45 cost14 <- sum(cost[,14])
46
47 cost_ISBL <- c(cost1, cost2, cost3, cost4, cost5, cost6, cost7,
       cost8, cost9, cost10, cost11, cost12, cost13, cost14)
48
49 cost_ISBL
50
51 #Opex Based on an assumed 8000 hour run per year
52 cost_electricity <- (electricity_01 + electricity_02) *
       electricity_price
53 cost_steam <- steam/1000 * steam_price
54 cost_cooling_water <- (cooling_water_01*cooling_water_price +
       cooling_water_02*cooling_water_price + cooling_water_03 +
       cooling_water_price)/1000 #m3/h
55 labour_cost = (6*labour_operator + labour_engineer)/8000
56 maintanence = cost_ISBL*0.03/8000 #cost per hour of operation
57 co2_capture_rate = input[,2]*7100*0.18*input[,3]
58 MEA_degradation = (MEA_degradation_rate*co2_capture_rate/1000)/
       1100 * MEA_price #dollars/hour
59 process_fluid_cost = t(process_fluid*0.87*process_water_price +
       process_fluid*0.13*MEA_price)/8000 #Divided by 8000 to find
       hourly rate based on yearly bleed'n feed of process fluid
60
61
62 opex = rbind(cost_electricity, cost_steam, cost_cooling_water,
       process_fluid_cost, labour_cost, maintanence, MEA_degradation
       )
63 opex1 <- sum(opex[,1])
64 opex2 <- sum(opex[,2])
```

```
opex3 <- sum(opex[,3])
opex4 <- sum(opex[,4])
opex5 <- sum(opex[,5])
opex6 <- sum(opex[,6])
opex7 <- sum(opex[,7])
opex8 <- sum(opex[,8])
opex9 <- sum(opex[,9])
opex10 <- sum(opex[,10])
opex11 <- sum(opex[,11])
opex12 <- sum(opex[,12])
opex13 <- sum(opex[,13])
opex14 <- sum(opex[,14])

cost_electricity
cost_steam
cost_cooling_water
labour_cost
maintanence
MEA_degradation


opex_hourly <- c(opex1, opex2, opex3, opex4, opex5, opex6, opex7
    , opex8, opex9, opex10, opex11, opex12, opex13, opex14)
opex_hourly
#Finding the annualized deprecation on an assumed 8000 hour
    yearly operating time frame
rent = 0.023 #% as given as interest on long term fixed loans
    SSB
lifetime = 25
annual_depreciation = 0.04 #This to simulate the
capex_cost_hourly = cost_ISBL*0.063/8000
capex_cost_hourly

totalcost_hourly = opex_hourly+capex_cost_hourly


#Finding the cost per tonne CO2 captured
CC_hourly = co2_capture_rate/1000
price_ton <- totalcost_hourly/(CC_hourly*10)
price_opex <- opex_hourly/CC_hourly
price_capex <- capex_cost_hourly/CC_hourly

price_ton
price_opex
price_capex
```

```
107
108
109 #Plotting the economics
110 p1 <- ggplot() +
111   geom_point(aes(x=input[1:6,1], y=price_ton[1:6]), color = "red
        ", size=10) +
112   xlab("Column Height [m]") +
113   ylab("Cost $/ton CO2") +
114   theme(axis.title = element_text(size = rel(2)),
115         axis.text = element_text(size = 20))
116
117 p2 <- ggplot() +
118   geom_point(aes(x=input[13:16,2], y=price_ton[13:16]), color =
        "red", size=10) +
119   xlab("Capture Rate [%]") +
120   ylab("Cost $/ton CO2") +
121   theme(axis.title = element_text(size = rel(2)),
122         axis.text = element_text(size = 20))
123
124 p3 <- ggplot() +
125   geom_point(aes(x=input[7:12,4], y=price_ton[7:12]), color = "
        red", size=10) +
126   xlab("V/L Ratio") +
127   ylab("Cost $/ton CO2") +
128   theme(axis.title = element_text(size = rel(2)),
129         axis.text = element_text(size = 20))
130
131 p1
132 p2
133 p3
```

The code used to combine and create the training and validation datasets

Below is the code used to combine the .xls files created by CO2Sim into a single file.

```
library(readxl)
library(tidyr)

# Creating the dataset for input

return_inputs <- function(input) {
  #Removing the unecessary first column
  var0 <- input[,-1]
  var1 <- c(var0[2,])
  var2 <- c(var0[19,])
  var3 <- c(var0[17,])
  var4 <- c(var0[44,])
  input <- cbind(var1, var2, var3, var4)
  return(input)
}
#Creating dataset for responses
return_responses <- function(input) {
  #Removing the unecessary first column
  var0 <- input[,-1]
  var1 <- c(var0[30,])  #Reboiler duty
  var2 <- c(var0[47,])  #Hex duty
  var3 <- c(var0[10,])  #Solvent lean loading
  var4 <- c(var0[11,])  #Solvent rich loading
  var5 <- c(var0[43,])  #Sweet gas flow rate
  var6 <- c(var0[29,])  #co2 fraction top of column
  var7 <- c(var0[50,])  #Sweet gas comp h2o
  var8 <- c(var0[51,])  #N2 comp sweet gas
  var9 <- c(var0[20,])  #Capture rate
  var10 <- c(var0[46,]) #Condenser duty
  var11 <- c(var0[8,])  #Solvent lean temp
  var12 <- c(var0[9,])  #Solvent rich temp
  var13 <- c(var0[25,]) #Reboiler temp
  var14 <- c(var0[33,]) #hexrichtempin
  var15 <- c(var0[34,]) #hexrichtempout
  var16 <- c(var0[35,]) #hexleantempin
  var17 <- c(var0[36,]) #hexleantempout
  var18 <- c(var0[48,]) #Sweet gas temp
  response <- cbind(var1, var2, var3, var4, var5, var6, var7,
      var8, var9, var10, var11, var12, var13, var14, var15, var16
```

```
      , var17, var18)
  return(response)
}


#Importing the datasets
my_file01 <- data.frame(read_excel("columnheightplot10.xls"))
my_file02 <- data.frame(read_excel("columnheightplot12.xls"))
my_file03 <- data.frame(read_excel("columnheightplot14.xls"))
my_file04 <- data.frame(read_excel("columnheightplot16.xls"))
my_file05 <- data.frame(read_excel("columnheightplot18.xls"))
my_file06 <- data.frame(read_excel("columnheightplot20.xls"))


#Applying the functions on the inputs and responses
input01 <- return_inputs(my_file01)
response01 <- return_responses(my_file01)
input02 <- return_inputs(my_file02)
response02 <- return_responses(my_file02)
input03 <- return_inputs(my_file03)
response03 <- return_responses(my_file03)
input04 <- return_inputs(my_file04)
response04 <- return_responses(my_file04)
input05 <- return_inputs(my_file05)
response05 <- return_responses(my_file05)
input06 <- return_inputs(my_file06)
response06 <- return_responses(my_file06)

#Combining and writing to csv
input <- rbind(input01, input02, input03, input04, input05)
responses <- rbind(response01, response02, response03,
    response04, response05)
write.csv(input,"input_validation_01.csv", row.names = FALSE)
write.csv(responses,"response_validation_01.csv", row.names =
    FALSE)
```