



University of Cape Town

Department of Chemical Engineering

CHE4045Z

**“JAXED-UP” OPTIMISATION OF THE CARBON DIOXIDE TO
METHANOL PROCESS**

By: Gabriella Heurlin and Anna Midgley



23rd November 2021

Project Number: 34

Process Modelling and Optimisation Group

Supervisor: Klaus Möller

Synopsis

This thesis demonstrated that JAX can be used to improve process optimisation. JAX can compute accurate automatic derivatives of complex process models. It allows for high performance compiled code. Both of which, increase the efficiency of process optimisation. This thesis used the carbon dioxide to methanol process as a case study and showed that a better optimum was found due to the superior optimisation techniques used.

Process models often contain numerical methods that make them more advanced, but their optimisation tricky. Previously, to optimise these complex models, non-derivative methods were typically used because the computation of their derivatives is difficult. However, “black-box” non-derivative methods are much slower and less efficient. When derivative-using optimisation methods were used, then numerical differentiation (ND) was typically employed. However, ND is inaccurate compared to automatic differentiation (AD), and computationally expensive. An alternative was to simplify these models by manipulating them into algebraic equations. This allowed for the use of non-derivative and derivative-using optimisation, with any type of differentiation. However, the major disadvantage to model manipulation, is that it must be manually done, which is both time consuming and hard to implement.

The CO₂ to methanol process is environmentally important. The process is simple, with detailed modelling only needed and performed for its two essential units, the reactor and separator. However, the optimisation problem remained complex. The complexity of the problem stemmed from the exponential nature of the reactor’s kinetic model and its ordinary differential equations that required solving. Further complexity was added by the equations of state required to model non-idealities of components in the separator. This was the perfect type of problem, as modelling was focused in two unit areas, but the overall model was challenging enough to compare the performance of the different optimisation methods.

It was shown that derivative-using optimisation methods took 100 times fewer iterations than non-derivative methods, concluding them as more efficient. Derivative-using methods were further improved by using AD instead of ND. AD outperformed ND on the complex, cost objective function by 6 times. The overall optimisation of the process led to a 47% improvement in the methanol produced, and a 42% improvement in the process profit, compared to that of the base case. The overall conversion achieved was also higher than two previous optimisation studies of this process, thus proving that a better optimal was found. Lastly, JAX and its just-in-time compilation were able to reduce the run time of the reactor and separator model by 1000 times. The ability to reduce computational time means that faster optimisation can be now performed in Python, and that complexities can be added without excessive computational time overheads.

To conclude, this thesis demonstrated that by using JAX, the computational efficiency of Python can be increased, as well as the performance of optimisation methods through the use of AD. This led to superior optimisation because each iteration of the method was calculated faster and more accurately. The result of this was that a better optimum was found, due to the ability to increase the complexity of the optimisation problem. This was successfully demonstrated on the environmentally important process that researchers have been attempting to optimise for years. Ultimately, this better optimisation can be used to improve the design and operation of processes by maximising their overall performance.

Table of Contents

Synopsis	i
Table of Contents	ii
List of Figures	iv
List of Tables	v
Nomenclature	vi
Glossary	vii
Abbreviations	ix
1. Introduction	1
1.1. Background Information	1
1.2. Problem Statement	1
1.3. Scope of Study	1
1.4. Objectives	2
2. Literature Review	3
2.1. Solving the Carbon Conundrum	3
2.2. Simplified Flowsheet of the CO ₂ to Methanol Process	3
2.3. Modelling the Process	4
2.3.1. Reactor Modelling	4
2.3.2. Flash Separator Modelling	5
2.3.3. Recycle Modelling	5
2.4. Variables that Reveal Trade-Offs in the CO ₂ to Methanol Process	6
2.5. Current Optimisation Techniques	7
2.6. Automatic Differentiation	8
2.7. JAX	10
2.8. Numerical Methods	11
3. Methodology	13
3.1. Introduction	13
3.2. Mathematical Model	13
3.2.1. Reactor Model	13
3.2.2. Separator Model	14
3.3. Model Validation	15
3.4. Formulation of Problem	16
3.4.1. Implementation of Model	16
3.4.2. Optimisation Problem – Objective Function, Constraints, and Bounds.	16

3.5. Optimisation Performance Tests	18
3.5.1. Computational Speed	18
3.5.2. Derivative vs Non-Derivative Optimisation.....	19
3.5.3. Ipopt with Numerical Differentiation vs Automatic Differentiation.....	20
3.5.4. Optimal Point Comparisons to Literature	21
4. Results and Discussion	22
4.1. Model Validation.....	22
4.2. Optimisation Performance.....	25
4.2.1. Computational Speed	25
4.2.2. Derivative vs Non-Derivative Optimisation.....	26
4.2.3. Ipopt with Numerical Differentiation vs Automatic Differentiation.....	27
4.2.4. Evaluation of the Optimal Point Found	29
4.3. Issues with JAX.....	31
5. Conclusions and Recommendations	32
5.1. Further Work	33
6. References.....	34
7. Appendices.....	A
7.1. Appendix A - Reactor Modelling.....	A
7.2. Appendix B – Separator Modelling.....	E
7.3. Appendix C – Process Costing.....	I
7.4. Appendix D – Mathematical Formulation.....	L
7.5. Appendix E – Base Case Conditions and Results.....	O
7.6. Appendix F – Reactor Model Validation Additional Graph.....	P
7.7. Appendix G – Computational Efficiency Additional Graphs.....	Q
7.8. Appendix H – Differential Evolution Flow Diagram.....	R
7.9. Appendix I – Signed Ethics Form	S
Plagiarism Declaration.....	U

List of Figures

Figure 1. Simplified PFD of the CO ₂ to methanol process.....	4
Figure 2. Computational graph illustrating AD process from Baydin et al. (2015)	9
Figure 3. Different approaches to computing the derivative of mathematical expressions	10
Figure 4. Illustration of current optimisation space using reactor modelling as an example.....	12
Figure 5. Class structure of coded model.....	16
Figure 6. Component molar flowrates along the reactor length (excluding hydrogen)	22
Figure 7. Temperature profile along the reactor length	23
Figure 8. Comparison of K values obtained from thesis simulation (line) to those from obtained from COCO (dot)	24
Figure 9. Comparison of computational time taken to solve models without just-in-time compilation and with just-in-time compilation	25
Figure 10. Computational time taken by Ipopt method using ND vs. AD.....	28
Figure 11. Optimisation performance for Ipopt with ND vs. AD for the cost objective function.....	28
Figure 12. Component molar flowrates along the reactor length (including hydrogen)	P
Figure 13. Computational time for each run of the separator model	Q
Figure 14. Flow diagram of DE algorithm	R

List of Tables

Table 1. Illustration of expression swell using a logistic map example taken from Baydin et al. (2015)	9
Table 2. Constants used in the reactor mass and energy balance equations	14
Table 3. Summary of numerical methods.....	15
Table 4. Comparison of thesis simulation with industrial data (Chen et al. 2011)	22
Table 5. Summary of comparison of DE and Ipopt.....	26
Table 6. Ipopt with AD and ND comparison for both objective functions	27
Table 7. Comparison of base case and optimal case.....	29
Table 8. Comparison of scaled methanol production to Chen et al. (2011) and Santos, Santos, and Prata (2018)	30
Table 9. Comparison of optimal points found, and optimisation techniques used	30
Table 10. Syngas Feed Composition	A
Table 11. Constant parameters for reactor mass and energy balance equations	B
Table 12. Constants for kinetic model	B
Table 13. Equilibrium constants for reaction kinetics.....	B
Table 14. Constants for the heat capacity and heat of formation calculations.....	C
Table 15. Component Values for Antoine Equation.....	E
Table 16. Critical temperatures, pressures, and acentric factors for each component	G
Table 17. Binary interaction parameters.....	H
Table 18. Typical carbon ratios of the syngas feed seen in literature.....	M
Table 19: Typical reactor pressures seen in literature.....	M
Table 20. Base case parameter conditions	O
Table 21. Base case stream table	O

Nomenclature

Symbol	Description
ρ_{cat}	Density of catalyst bed
a	Activity of catalyst
ε_{cat}	Void fraction of catalyst
U	Overall heat transfer coefficient from reactor tube to shell side
T_{shell}	Temperature of boiling water on the shell side of reactor
C_p	Specific heat at constant pressure
r_i	Rate of reaction for component i
$\Delta H_{f,i}$	Heat of formation of component i
D_{int}	Internal diameter of reactor tube
A_{int}	Cross-sectional area of reactor tube
\dot{f}_t	Total molar flowrate per tube
T	Temperature
x_i	Mole fraction of component i in the liquid phase
y_i	Mole fraction of component i in the vapour phase
z_i	Mole fraction of component i in the separator feed
MW_i	Molecular weight of component i
z	Length of reactor
N_t	Number of tubes
N	Number of species
ϕ_i	Fugacity co-efficient of component i
δ_{ij}	Binary interaction parameter for component i and j
ν_i	Stoichiometric coefficient of component i
n	Number of reactions

Glossary

Term	Description
Automatic Differentiation	The computation of the derivative of a function at a specified value, using the mechanical application of the chain rule.
Differential Evolution	A population-based search algorithm that optimises a problem by iteratively improving candidates with respect to their fitness score (the value of the objective function). The method is based on the evolutionary process.
Discretisation	The process of making continuous functions or models into discrete counterparts.
Objective function	The function, in terms of the decision variables, that is to be maximised or minimised in optimisation. The basis is normally economic. For example, an equation that describes the profits of a process.
Python library	Reusable chunks of Python code that contain functions which can be imported and utilised. An example of which is the NumPy.Roots function which can be imported and implemented from the NumPy library and will output the roots of the inputted polynomial equation.
JAX	JAX is a library created by Deepmind that combines XLA, a domain specific compiler for linear algebra, and Autograd. The library therefore brings high-performance computing together and great automatic differentiation abilities to Python.
Numpy	NumPy is a scientific computing Python library. It can provide a multidimensional array object and an assortment of routines for array operations. An example of which is shape manipulation.
SciPy	SciPy is an easy to use, performant Python library, that contains routines for optimisation, integration, differential equations, and many other types of problems.
NumDifftools	A Python library of tools for differentiation for single and multi-variate problems.
Compiler and Interpreter	Compilers and interpreters are used to convert high level code, written by humans, into machine that can be understood by computers. Interpreters, used by languages such as Python, translate the code one statement at a time. On the other hand, compilers, used by languages such as C++, scan the entire program and translate all of it at once into machine code. Interpreters therefore spend less time analyse the human code and more time executing the code whilst compilers spend more time analysing it and less time executing it.
Python class	Classes are a type of object in Python. They are a means to bundle data and functionality together - like a blueprint for an object. Instances of a class can be made when each class instance has attributes (variables) attached to it for maintaining its state, as well as methods that modify its state.
Just-in-time compilation	Compilation of code during execution of a program at run time rather than before execution. JIT compilers continuously analyse the code that is being executed and locate areas where speedup

	gained from compilation or recompilation are greater than the overheads of compiling that code.
Jax.jit	A function from JAX library that allows you to compile your code using XLA and just-in-time (jit) compilation. XLA is like an engine that can be used to accelerate blocks of code. The first time the code is run, the code is compiled whilst from the second time onwards the optimised version in terms of computational efficiency is run.
Jax.lax.scan	Scans a function over leading array axes while carrying along state. It prevents a jit-compiled loop from being unrolled each time it is stepped through - which would lead to large XLA computations.

Abbreviations

Abbreviation	Description
AD	Automatic differentiation
ND	Numerical differentiation
DE	Differential evolution
jit	Just-in-time
ODE	Ordinary differential equation
PDE	Partial differential equation
EOS	Equation of state
PR	Peng Robinson
SM	Sequential-modular
EO	Equation-oriented
Ipopt	Interior Point Optimisation
XLA	Accelerated Linear Algebra
MB	Mass balance

1. Introduction

1.1. Background Information

This thesis explores the general optimisation space of process modelling. Process models often contain numerical methods that serve various functions. Examples of such functions include the solving of ordinary differential equations (ODEs) or finding the roots to an equation. However, these can make models highly complex and hence, efficient optimisation tricky. Previously, non-derivative methods were typically used to optimise these complex models. However, non-derivative methods are much slower, less efficient, and “black-box” where the relationship between variables is not understood. When derivative-based optimisation methods were used, then numerical differentiation (ND) was typically used. However, ND is inaccurate compared to automatic differentiation, and computationally expensive.

An alternative was to make these models less complex by re-writing them in terms of algebraic equations. This then allowed for all optimisation techniques to be applied (non-derivative or derivative-using, with any type of differentiation used to compute the derivatives). However, the major disadvantage to model manipulation, is that it must be manually done and is therefore, time consuming and harder to implement. It also becomes exponentially harder as processes become more complex. Additionally, specialised numerical libraries are likely to be significantly more advanced than numerical methods coded in by a user.

Recently, AD libraries (such as JAX) have progressed significantly, bolstered by machine learning research. This thesis will aim to prove that optimisation can be performed on complex models that contain numerical methods, and that the more accurate derivative finding method, AD, can be used. It will show the simplicity, accuracy, and efficiency that JAX can bring to flowsheet modelling and optimisation.

1.2. Problem Statement

To show the efficiency of optimisation that uses automatic derivatives supplied by JAX using the CO₂ to methanol process as an example problem, and to find a better optimum for this process.

1.3. Scope of Study

Various assumptions were made to simplify the process model of the CO₂ to methanol flowsheet. Only the essential units, the reactor and separator, as well as the recycle loop, were modelled. It was assumed that the heat exchangers and the recycle compressor operated at their desired conditions, thus leading to a temperature or pressure change as needed, but not requiring modelling of the energy transfers that occurs. The scope of the reactor model did not include added complexities such as intra-particle diffusion limitations to simplify it. It was decided that equations of state (EOS) were within the scope as their use was essential for accurate modelling of the high-pressure separator. Peng Robinson (PR) EOS was used to capture the non-idealities of the components in the separator. Regarding the optimisation problem, the decision variables that were analysed were inlet temperature of reactor, temperature of separator, pressure of separator, recycle ratio, and ratio of carbon monoxide to carbon dioxide of the syngas feed. Other variables, such as the reactor shell temperature or reactor length, could be optimised but were not, because there was limited time for this project.

The optimisation was assumed to be at steady state, and hence, the scope of the process did not include dynamic optimisation. The only two objective functions

considered were to maximise the methanol production rate and to maximise the profit yielded from the process. The costs of the flowsheet incorporated both the capital costs (the catalyst, reactor preheater, reactor product cooler, and compressor costs), and the energy costs (the compressor work, steam used for the reactor preheating, and cooling water needed for the reactor product cooling). The forementioned costs were included as their value depends on the decision variables that were investigated. All other costs, such as reactor cost, were assumed constant and therefore ignored. The total annualized cost (TAC) was then calculated using the total capital cost and assuming a payback period of 3 years. The incomes considered for the flowsheet were the sale of methanol, the high-pressure steam generated in the reactor, and the heating value of the purge stream which can be burned to recover heat.

The modelling of the process and the different optimisation methods were performed in Python and used the JAX library. JAX was the only AD library considered in this thesis. The optimisation methods that were considered, derivative and derivative free, were ones available in Python libraries. The derivative-free method considered was SciPy's differential evolution. The derivative-using method considered was Interior Point Optimisation (Ipopt) (Wächter and Biegler, 2005).

1.4. Objectives

The objectives of the report are to:

1. Model the CO₂ to methanol, high pressure loop, process in Python code.
2. Formulate and implement optimisation of the process model.
3. Show the added computation efficiency that JAX brings to Python for process modelling and optimisation, thus proving its broad use and applicability to chemical engineering.
4. Compare the performance of derivative (Ipopt) and non-derivative (differential evolution) optimisation methods.
5. Compare the performance of derivative methods (Ipopt) when using numerical differentiation versus automatic differentiation.
6. Compare the value of the optimal points found to that of literature for the optimisation of the same process.
7. Map out key problems and potential issues that arise from setting up the model to be compatible with JAX functions.

2. Literature Review

2.1. Solving the Carbon Conundrum

The large energy demand from our fast-growing society has meant the exorbitant burning of fossil fuels and accompanying large emissions of carbon dioxide. The increased amounts of CO₂ in the atmosphere have not only caused a take-over of the natural carbon cycle but have caused adverse environmental changes seen across the globe. Top-priority research has explored powerful tools to help mitigate these adverse effects. Examples of which include renewable energy, and the recycling of CO₂ into value-added chemicals such as methanol (Goeppert *et al.*, 2014).

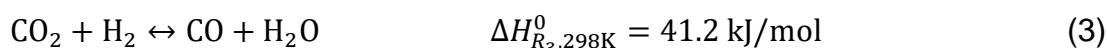
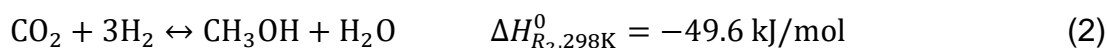
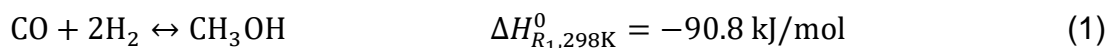
Renewable energy continues to rise with importance and has been receiving increased global attention. However, for renewable energy to fulfil just a slight portion of the exorbitant energy demands from our society, energy storage on excess capacity is required (Goeppert *et al.*, 2014). Energy storage, with its vital “use when needed” quality, is especially useful for the two most promising yet highly variable renewable energies: solar - which relies on clear skies, and wind - which relies on a constant blow of wind. Electrical energy, generated from renewable energy, can be stored as chemical energy (bonds) in compounds such as hydrogen (H₂), or methanol.

H₂ is an excellent fuel that can be produced by the splitting of water. However, its physical and chemical properties have presented some major drawbacks for its use as energy storage. Due to its low volumetric density, H₂ requires compression at high pressures or liquefaction at very low temperatures (Goeppert *et al.*, 2014). This makes the storage, transport, and distribution of H₂, highly problematic and expensive. The storage of energy in liquid form has therefore, been proven to be much less cumbersome and highly preferable. A major alternative candidate is methanol.

Methanol’s use as an energy carrier has been greatly encouraged to date. Moreover, due to its high-octane rating, methanol proves itself as a suitable substitute for gasoline in internal combustion engines (Goeppert *et al.*, 2014), or to be used in direct methanol fuel cells (Goeppert *et al.*, 2014). Although it is currently more economical to produce methanol via fossil fuels, methanol can be made from any carbon-containing feedstock. The more common process consists of the conversion of syngas (containing mainly CO, CO₂, and H₂) to produce methanol via various catalytic systems (Santos, Santos and Prata, 2018). The subsequent combustion of methanol with its derivatives releases more CO₂, which can then be recycled back. This resulting closure of the carbon loop can then act as a powerful solution to the global carbon conundrum (Goeppert *et al.*, 2014). This environmentally important process that converts CO₂ to methanol, is of interest to this thesis.

2.2. Simplified Flowsheet of the CO₂ to Methanol Process

A simplified flowsheet of the methanol synthesis process can be seen in Figure 1. The production of methanol in a catalytic reactor (R-101) includes the hydrogenation of CO and CO₂ (reaction 1-2), as well as the reverse water-gas shift reaction (reaction 3).



The effluent from the reactor is cooled to a temperature that allows for liquid methanol to be separated in a succeeding high-pressure flash separator (T-101). The vapour from the flash gets compressed and recycled back. A purge is included to avoid the accumulation of inerts in the process.

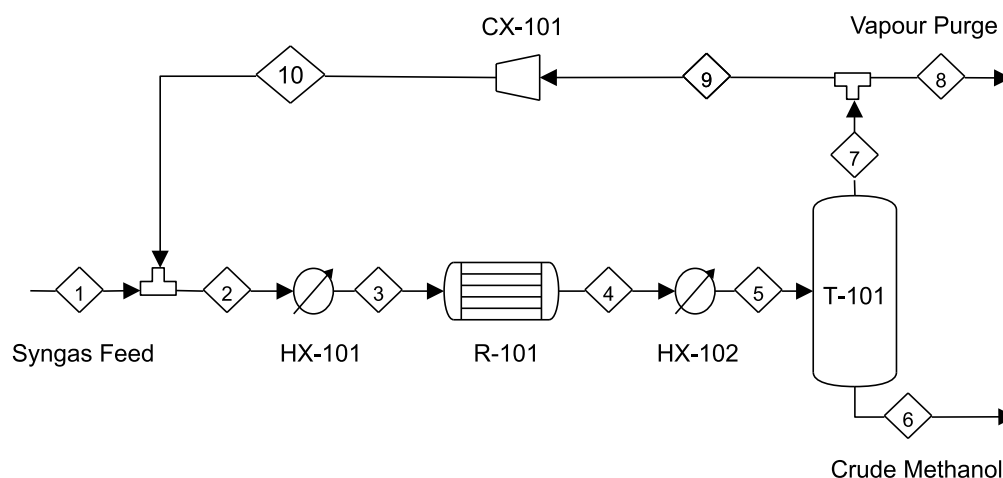


Figure 1. Simplified PFD of the CO₂ to methanol process

The stream numbers in the flowsheet (Figure 1 above) are used consistently throughout this report.

2.3. Modelling the Process

Developments of mathematically rigorous models of the CO₂ to methanol process have been shown by Santos, Santos and Prata (2018) and Abrol and Hilton (2012). For relevance to this thesis, details of the methanol reactor, flash separator, and recycle loop were required. The assumption made by Abrol and Hilton (2012) that the process heat exchangers and recycle compressor were at their desired pressure and temperature values, deemed applicable.

2.3.1. Reactor Modelling

These studies showed that the mass and energy balances across the catalytic plug flow reactor, could be described as a set of differential equations comprising of composition and temperature along the reactor length (Abrol and Hilton, 2012; Santos, Santos and Prata, 2018). Assumptions such as unidimensional flow, negligible axial diffusion, negligible heat conduction, and no accumulation of coolant in the reactor jacket, were made by (Manenti, Cieri and Restelli, 2011) and Chen *et al.* (2011).

Abrol and Hilton (2012) performed a dynamic optimisation on the process. This meant that their mass and energy balance equations were a set of partial differential equations (PDEs). Abrol and Hilton (2012) solved the open-loop dynamics of the reactor by discretizing the PDEs on the nodes of a one-dimensional mesh in the axial z-direction. Backward finite difference approximation was then used to reduce the equations to a set of ODEs, which were then solved using a stiff solver. Discretisation was required by Abrol and Hilton (2012), as their process was dealing with transient conditions. This is typical for control problems when there are time changes, and the process is not at steady state. However, for relevance to this thesis, the CO₂ to methanol process will be modelled at steady state. The PDEs used by Abrol and Hilton (2012) could, however, be converted to steady state differential equations by setting the time differentials equal to zero. Nevertheless, these were not used as a basis for

the reactor modelling as the differential equations by Abrol and Hilton (2012) were written in terms of mass, and rather, a molar basis was more appropriate for steady state process modelling.

Steady state optimisation of the process was shown by Santos, Santos and Prata (2018). They described the reactor by a set of ODEs that were solved using the 4th order Runge-Kutta method. Equations given by Smith (1950) were suggested for use when calculating the heat of formation for compounds: CH₃OH, CO₂, CO, H₂O, and H₂, as well as the specific heats for compounds: CH₃OH, CO₂, CO, H₂O, H₂, CH₄, and N₂.

Kinetic data provided by Bussche and Froment (1996) was suggested as the most accurate for use by both Santos, Santos and Prata (2018) and Abrol and Hilton (2012). The Bussche and Froment (1996) kinetic model was formulated to overcome the limitations shown by Graaf *et al.* (1990) who based their kinetics on all three reactions (1-3). However, it can be shown that only two of the three reactions are independent, and thus, any two reactions can be used for the kinetics. Further limitations shown by Graaf *et al.* (1990) were due to certain intermediate species (such as formyl and methoxy) being predicted simultaneously in their model and occurring in two different reactions at the same time. Although it is extensively used, the Graaf *et al.* (1990) model was, therefore, proven redundant.

2.3.2. Flash Separator Modelling

Santos, Santos, and Prata (2018) modelled the flash separator using a Pressure-Temperature flash algorithm described by Biegler, Grossmann, and Westerberg (1997). The algorithm used known values for temperature, pressure, and inlet stream composition, to calculate the vapour ratio and the gas and liquid outlet compositions. Santos, Santos and Prata (2018) solved the Rachford-Rice equation (see Equation 7 under Heading 3.2.2) using the bisection method, a root finding algorithm.

The flash separator operates at a high temperature and high pressure. Therefore, the non-ideality of the components in the gas phase (CO, H₂, and CO₂) cannot be disregarded (Santos, Santos and Prata, 2018). The liquid phase, comprising of mainly CH₃OH and H₂O, express highly non-ideal behaviours at these temperatures due to their polar molecular structures. Therefore, for a more realistic and accurate prediction of the non-ideal vapour-liquid equilibrium, Santos, Santos and Prata (2018) recommended using the Peng Robinson equation of state (PR-EOS) to calculate the components' fugacity coefficients. However, Santos, Santos and Prata (2018) assumed that the liquid phased contained negligible gases and therefore, they assumed ideality. This assumption made their model simpler and therefore easier to solve.

2.3.3. Recycle Modelling

The vapour product from the separator is recycled back after being compressed. The resulting recycle loop adds complexity to the model. The loop can be solved using the sequential-modular (SM) or equation-oriented (EO) based approach. The SM approach solves each block in the flowsheet in a sequence, where the inlet stream is given, and each block computes its relevant outlet streams. If recycle loops are present, then multiple iterations are required (Aspen Engineering Suite, 2004). Abrol and Hilton (2012) used the SM approach to solve their recycle loop by implementing the Wegstein algorithm. This involved the tearing of the loop, followed by the updating of subsequent values using the algorithm. The interactions were halted when the

maximum allowable iterations were exceeded, or when the tolerance criterion was met. Although the SM approach is simple and effective for many types of simulations, the strategy can be very time-consuming and difficult to solve with the many successive solutions it requires. SM optimisation capabilities are also limited (Kisala *et al.*, 1987). Therefore, it has been agreed that the EO approach has significant advantages over the SM approach, as it is able to gather all the model equations together and solve them simultaneously at the same time (Abrol and Hilton, 2012). Although the number of variables and equations can be very large, the robust EO approach proves very effective for highly heat-integrated processes, highly recycled processes, processes with many design specifications, and process optimisation (Aspen Engineering Suite, 2004).

2.4. Variables that Reveal Trade-Offs in the CO₂ to Methanol Process

Most studies that have performed optimisation on the CO₂ to methanol process, have set the objective function to maximise the methanol production rate. The parameters that are varied within the optimisation problem, stem from those that reveal trade-offs that impact the methanol flowrate. It was seen that the reactor temperature, carbon content of the reactor feed, the CO₂/CO ratio in the syngas feed, and the recycle/purge ratio were important parameters that revealed trade-offs. The reasons for which are discussed in further detail below.

Studies have shown that temperature has severe effects on both the methanol production rate and catalyst deactivation (Kordabadi and Jahanmiri, 2007; Chen *et al.*, 2011; Santos, Santos and Prata, 2018). Methanol production is favoured at lower temperatures; however, higher temperatures increase the rate of the reactions. Nevertheless, at higher temperatures, the risk of catalyst deactivation adversely increases. Thus, proving temperature to be a complicated, yet highly important parameter to consider when optimising.

Lim *et al.* (2010) studied the effect of the CO₂ fraction at the inlet to the reactor. It was seen that an increased amount of CO₂ would lead to an increased production of water via the reverse water gas shift reaction. However, despite the process needing water for catalyst activation, if too much is produced, the rate of catalyst deactivation increases. The carbon content of the reactor feed is also directly proportional to the recycle ratio. A higher recycle ratio means an increased amount of CO₂ at the inlet to the reactor, and thus, an increased production of water (which needs the careful attention). However, an increased carbon content into the reactor also increases the overall methanol yield, which will generate the process more income. Further investigation was shown by Abrol and Hilton (2012), on the additional trade-offs seen by the recycle ratio.

It was seen that a smaller purge ensured inert gas removal and no negative impacts on the carbon conversion within the reactor (Abrol and Hilton, 2012). However, the subsequent, increased recycle was also seen to be undesirable. This was because it increased the operating costs of the recycle compressor, the size of the reactor, and the overall catalyst cost. Despite a decreased carbon-conversion, Abrol and Hilton (2012) mentioned that a decreased recycle showed less water leaving with the crude methanol and therefore eliminated the need for large amounts of water to be removed downstream. Lastly, Abrol and Hilton (2012) also noted that a decreased recycle (i.e.,

a larger purge) could generate the process additional income, as the purge can be combusted and utilized as a fuel gas.

The multiple trade-offs displayed by the recycle ratio, conclude that it is nothing short of significant for optimisation. However, despite extensive research into various optimisation studies for this process, the inclusion of the recycle ratio as an optimisation variable has not yet been seen. One reason for this could stem from the added complexity that the recycle ratio brings to the solving of model, as the problem increases in sensitivity to becoming ill-conditioned.

2.5. Current Optimisation Techniques

Previous studies that investigated the optimisation of the CO₂ to methanol process used differential evolution or genetic algorithm approaches to perform optimisation (Kordabadi and Jahanmiri, 2005; Lim *et al.*, 2010; Arab Aboosadi, Jahanmiri and Rahimpour, 2011; Santos, Santos and Prata, 2018). Both these optimisation methods are derivative-free and form part of the larger class, evolutionary algorithms (EA) (Vikhar, 2017). The EA methods optimise a problem by maintaining a population of candidate solutions. The candidate solutions with the highest fitness scores are then kept and used to create the following generation. This means that the optimisation problem is treated as a black box where the measure of the performance is given for candidate solutions, and the gradient (or derivative) is not needed (Bajaj, Arora and Hasan, 2021). The result of a black box method is lack of an introspection, where the relationships between variables are not understood (Freeman *et al.*, 2021). Additionally, optimisation methods that do not use derivatives are slower, less efficient (Heath, 2002a), and require expensive computation of the fitness score of all candidate solutions. See Appendix H for a flow diagram that aids in the explanation of how EA methods work.

There are various other disadvantages that the EA methods face, namely issues with complexity and convergence to local optima. The first issue is that the algorithm does not scale well with complexity. If there are many parameters that are exposed to mutation, then there is an exponential increase in the search space size. This often requires the problem to be broken into a simpler representation (Vikhar, 2017). The second issue with complexity is how to protect the parts that have evolved to represent good solutions from the mutator, thus, further worsening their fitness score. This is an issue particularly when the fitness score requires the solutions to combine well with other parts. Finally, EAs have the tendency to converge towards local optimum or arbitrary points, rather than to global optimal (Taherdangkoo *et al.*, 2013). Therefore, the method does not trade off short-term fitness for longer-term fitness (Vikhar, 2017).

One of the main advantages of EAs, or black-box methods, is that they only require objective function evaluations, and do not require derivatives (Bajaj, Arora and Hasan, 2021). Derivatives of process models are complicated to calculate and have previously required manipulation of the model into algebraic equations (if automatic differentiation was to be used). It is expected that this is the predominant reason for why optimisation studies of this CO₂ to methanol process, have used evolutionary algorithms.

There is, however, a wide array of efficient optimisation methods that use derivatives. The most basic and oldest (invented in 1847) is the steepest descent method (Heath, 2002b). The method works by taking a step in the direction of the negative of the gradient (or derivative). By doing this, the algorithm iteratively steps along the steepest descent direction, until it converges at the minimum of the function. Many

improvements have been made to this method, but the principle behind it is still used in the more complex methods used today. An example of a recent, very efficient derivative-using method is Interior Point Optimisation (Ipopt). The Ipopt algorithm uses interior point line search, based on filter methods, to find a local solution to large-scale non-linear optimisation problems (Wächter and Biegler, 2006). Both the objective function and equality constraints can be linear or non-linear and convex or concave, but either way, they must be twice differentiable. The algorithm exploits Jacobians and Hessians of both the equality constraints and the objective function. Ipopt’s applicability to process optimisation is well-established and the method is frequently used (Wächter and Biegler, 2005). This is mainly due to its ability to handle large-scale, non-linear optimisation problems which are typical of process optimisation. The method has been implemented in Ipopt code. Cyipopt is a Python wrapped version of it (Aides and Kummerer, 2017). Ipopt will calculate the derivative using numerical differentiation (specifically the Quasi-Newton method) unless the derivative is supplied. If the derivative is supplied, it is usually via automatic differentiation (Wächter and Biegler, 2006). This thesis will use Ipopt as an example of a derivative-using method and will analyse the effects that different differentiation techniques have on its performance.

2.6. Automatic Differentiation

There are four main approaches to computing the derivative of a mathematical expression. These are namely manual, symbolic, numerical, or automatic differentiation. Their differences can be seen in Figure 3.

Manual differentiation is the calculation of the derivative expression by hand and its coding. It is time-consuming and prone to human error, especially when models become more complex. Numerical differentiation is the finite difference approximation of the derivative using values of the original function evaluated at points. It is based on the limit definition of the derivative – see Equation 4 below, where \mathbf{e}_i is the i -th unit vector and h the small step size (Baydin *et al.*, 2015).

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h} \quad (4)$$

The main advantage of manual differentiation is that it is easy to implement. However, its major disadvantages are the introduction of truncation and round-off errors inflicted by the limited precisions of computers. To manage the inaccuracy caused from these errors, careful consideration in selecting the step size is therefore, required (Baydin *et al.*, 2015). Additionally, the method requires $O(n)$ evaluations of the gradient in n dimensions (Baydin *et al.*, 2015). Therefore, the method scales poorly for chemical simulations where the derivative with respect to many parameters is commonly needed.

Symbolic differentiation uses packages such as Mathematica, to produce the exact derivative expression as its output (Baydin *et al.*, 2015). To compute the derivative, the software automatically and mechanistically applies the rules of differentiation to the formulae. It addresses the weaknesses of manual and numerical differentiation; however, it frequently results in “expression swell”. Expression swell is the phenomenon of the exponentially larger representation of a derivative, as opposed to representation of the original function (Baydin *et al.*, 2015). See a further illustration in Table 1 below.

Table 1. Illustration of expression swell using a logistic map example taken from Baydin et al. (2015)

Function: $l_{n+1} = 4l_n(1 - l_n), l_1 = x$

n	$f(x)_n$	$\frac{d}{dx}f(x)_n$	$\frac{d}{dx}f(x)_n$ (Simplified form)
1	x	1	1
2	$4x(1 - x)$	$4(1 - x) - 4x$	$4 - 8x$
3	$16x(1 - x)(1 - 2x)^2$	$16(1 - x)(1 - 2x)^2 - 16x(1 - 2x)^2 - 64x(1 - x)(1 - 2x)$	$16(1 - 10x + 24x^2 - 16x^3)$
4	$64x(1 - x)(1 - 2x)^2(1 - 8x + 8x^2)^2$	$128x(1 - x)(-8 + 16x)(1 - 2x)^2(1 - 8x + 8x^2) + 64(1 - x)(1 - 2x)^2(1 - 8x + 8x^2)^2 - 64x(1 - 2x)^2(1 - 8x + 8x^2)^2 - 256x(1 - x)(1 - 2x)(1 - 8x + 8x^2)^2$	$64(1 - 42x + 504x^2 - 2640x^3 + 7040x^4 - 9984x^5 + 7168x^6 - 2048x^7)$

Symbolic differentiation does not consider the fact that there are commonly many sub-expressions in the different derivative expressions. This leads to inefficiency in its calculation. Symbolic and manual differentiation also require the model to be defined as closed-form expressions. This limits the expressivity of the model as they are then unable to use branches or loops (Baydin et al., 2015).

The fourth technique, auto-differentiation (AD), obtains the exact result of the derivative by breaking down the expression into elementary, arithmetic operations (addition, multiplication, etc) and elementary functions (exp, log, sin, etc). It then applies the chain rule to the derivatives of these operations (Baydin et al., 2015). This procedure is illustrated in Figure 2 below. This can be done to all expressions because it has been shown that all numerical computations are composed of a finite set of elementary operations for which derivatives are known (Verma, 2000; Baydin et al., 2015). Thus, AD frameworks can calculate the derivative of any arbitrary computer code, so long as it is made up of differential operations. AD uses symbolic rules of differentiation on an operation-by-operation basis instead of the entirety of the expression (like what is done in symbolic differentiation leading to expression swell). The process whereby the expression is broken down into elementary parts is illustrated below.

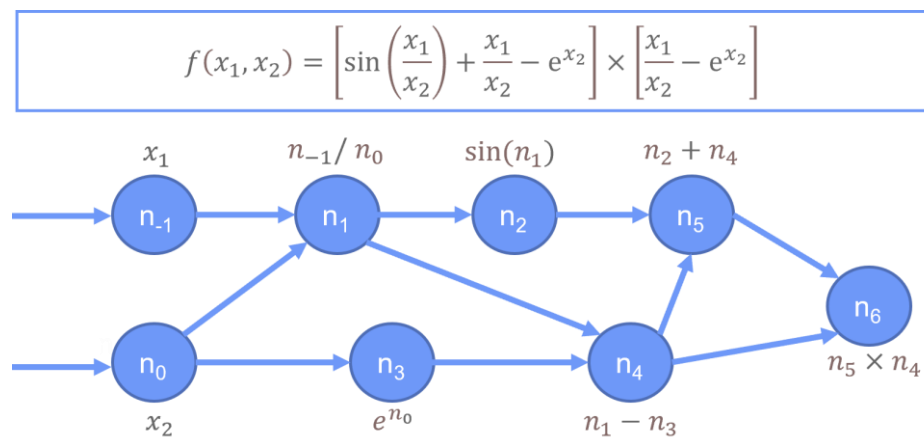


Figure 2. Computational graph illustrating AD process from Baydin et al. (2015)

A final visual representation of each approach described above, is seen in Figure 3 below.

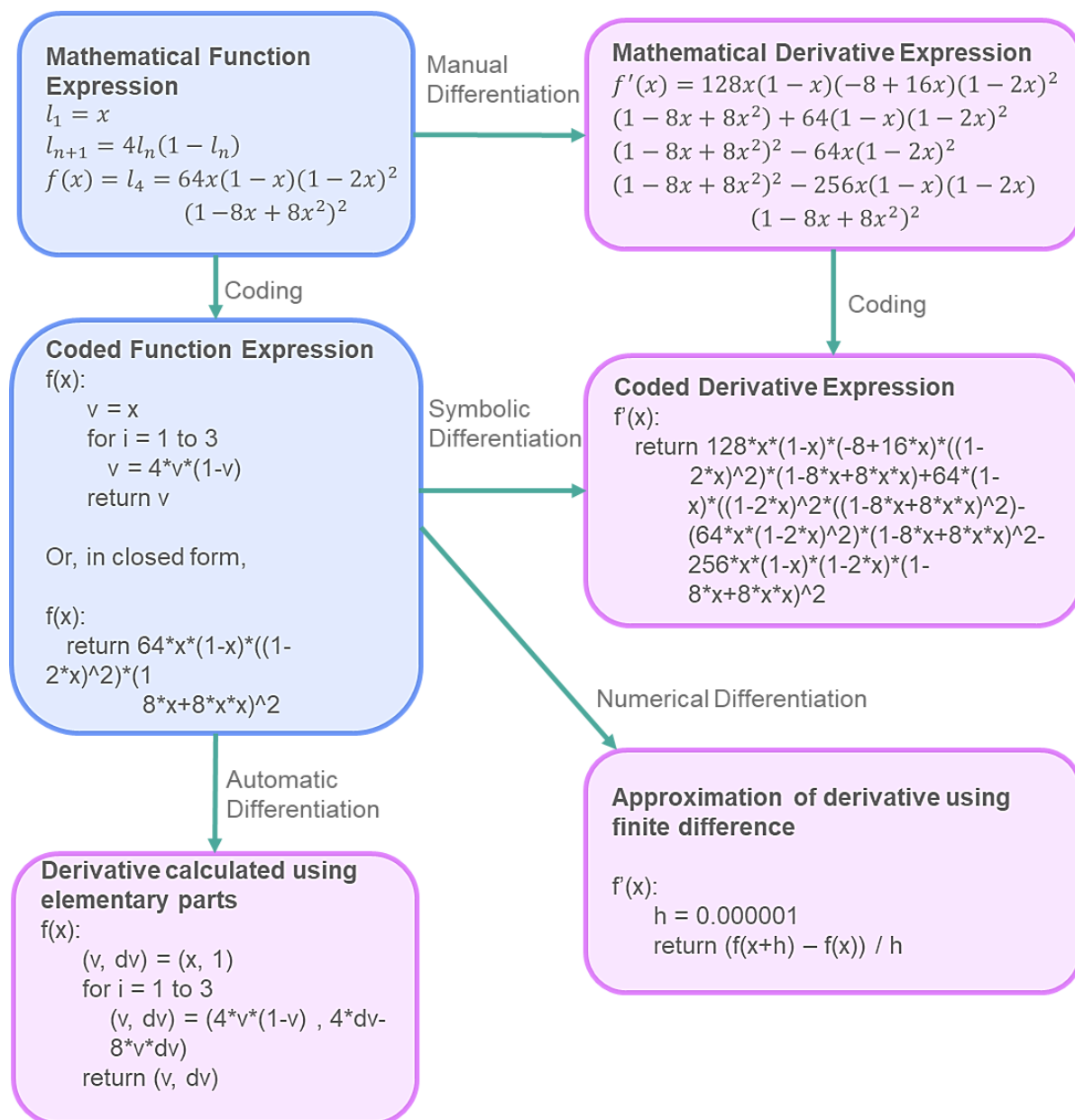


Figure 3. Different approaches to computing the derivative of mathematical expressions

2.7. JAX

Recently, AD libraries have progressed significantly, bolstered by machine learning research (Abbott *et al.*, 2021). Derivatives are ubiquitous in machine learning as most traditional learning algorithms require the calculation of Jacobians or Hessians of the loss function (objective function). JAX, an AD library, was developed by Deepmind (Google_ and combines Autograd with accelerated linear algebra (XLA) (Abbott *et al.*, 2021). XLA is a domain specific compiler that optimises algebraic computations, leading to significant improvements in speed and memory performance. Autograd can automatically differentiate native Python and NumPy code. It can handle loops, recursion, and closures, as well as take higher order derivatives. JAX supports both reverse and forward-mode differentiation.

Deepmind created JAX, which is an easily programmable and highly performant library, to accelerate their machine learning. Two examples of projects where JAX has been used are Brax and JAX M.D. Brax is an open-source library for rigid body simulations. Brax can train reinforcement learning agents that perform locomotion and dexterous manipulation tasks (Freeman *et al.*, 2021). It utilises auto-vectorisation, just-in-time compilation, and auto-differentiation, to simulate simple rigid body physics systems, in thousands of independent environments, across hundreds of accelerators (Freeman *et al.*, 2021). JAX M.D. on the other hand, is a package capable of simulating molecular dynamics. The simulations of the molecular dynamics are themselves, differentiable. This results in entire trajectories that can be differentiated for meta-optimisation (Schoenholz and Cubuk, 2020). JAX now underpins many of Deepmind’s recent publications (Budden and Hessel, 2020).

Automatic differentiation (grad), auto-vectorisation (vmap) and just-in-time compilation (jit) are a few of the various functions within JAX that are useful and applicable to this thesis. Vectorisation allows for functions to be applied to lots of data without the need for batching. This is useful to as optimisation can be done for many different starting points simultaneously which can be used to ensure that the global minimum or maximum is found. Jit compilation allows for programmes to be easily scaled to one or many accelerators, without the coder needing previous experience in high-performance computing. The function is compiled during execution of the program rather than before execution. The jit compiler continuously analyses the code that is being executed. It identifies parts of the code where the speedup gained from compilation (or recompilation) outweighs the cost of compiling the code. JAX can therefore, bring extensive performance and functionality to the ease of development of models in Python code (Abbott *et al.*, 2021).

There are other AD libraries, such as JuliaDiff for Julia, that are similar to JAX in their capabilities (White and Lubin, 2021). The use of such libraries would have been a similar approach to that of the one used in this thesis.

2.8. Numerical Methods

Process models often require numerical methods that serve various functions. Examples of such include the solving ordinary differential equations (ODEs) or finding the roots to an equation. Prior to JAX and other recent AD libraries, most numerical methods were incompatible with the use of AD. Thus, if derivative-using optimisation methods were to be used, then only numerical differentiation (ND) could be used. ND is, however, far more inaccurate, and computationally expensive. The alternative was to instead code the model using algebraic equations, and no numerical methods (Saad, Ali and Ertogral, 2011). This can be further explained using reactor modelling as an example:

Figure 4 below shows the various options regarding how the reactor model can be optimised. Often, reactor modelling contains a set of ODEs that describe its mass and energy balances. Either the ODEs can be solved by discretisation, or by advanced ODE solvers. These more advanced solvers are just a more complex and accurate form of discretisation. However, they are significantly easier to implement as they can be imported from various libraries. The main disadvantage to discretisation is that the model, which is continuous, gets described by a finite set of numbers. This results in a loss of information (Wächter and Biegler, 2006). The following are techniques commonly used in chemical reactor engineering, namely: finite difference, weight residuals, or finite element methods (Jakobsen, 1992). This thesis aims to show that

automatic differentiation can now be done for complex models that include advanced numerical methods, such as ODE solvers. This is due to the progression in AD libraries such as JAX. The result of this, and what this thesis aims to prove, is that now optimisation methods with AD can be performed without manipulation of models into algebraic equations.

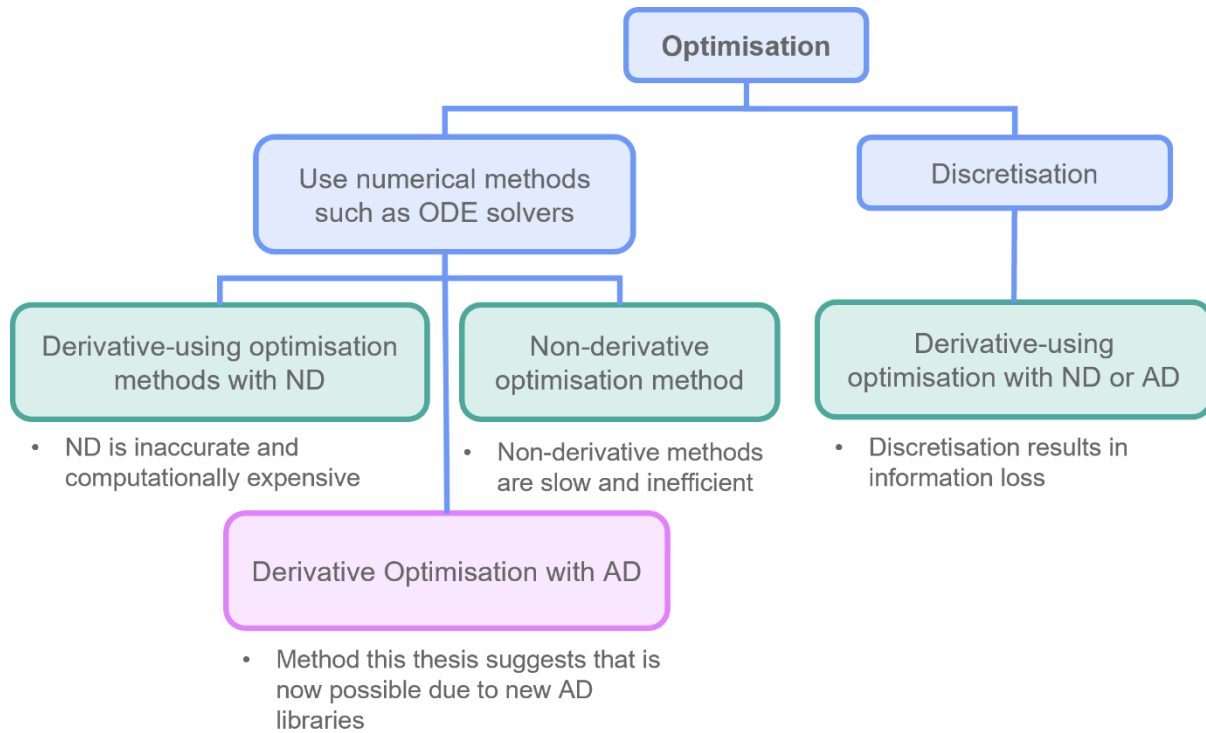


Figure 4. Illustration of current optimisation space using reactor modelling as an example

3. Methodology

3.1. Introduction

The reason why the CO₂ to methanol process was used as the case study, was for three reasons. Firstly, it is an environmentally important process. Secondly, the process is simple, with the detailed modelling only required for two units. However, the optimisation problem remains complex. The complexity of the problem arises from the highly exponential nature of the kinetic model of the reactor, the reactor's ODEs that require solving, and the required use of EOS in the separator model. This was the perfect type of problem as modelling could be focused, but the resulting model was challenging enough to see the difference in optimisation methods. Lastly, previous researchers have optimised the process using black box (or non-derivative methods) due to the forementioned complexities of the process. This allowed for concluding comparisons to be made to previous studies to show whether in fact the methods used in this thesis were able to find a better optimum.

The entirety of the CO₂ to methanol process was modelled in one succinct Python code. This was so that the different optimisation methods that were performed could be fairly compared by applying them to the same problem. The information used for the reactor and separator mathematical models, as well as the various assumptions made, are described below. Following, is an explanation of the model implementation into the Python code. Details on the optimisation problem - including the objective function, constraints, and bounds, are included. The Python code of this thesis is available at the following [Github repository](#).

3.2. Mathematical Model

The structure of the model was simplified into the essential units due to the limited time frame and novel nature of this work. The units that were modelled were the reactor and separator (flash tank). Both the heat exchangers (before and after the reactor), as well as the recycle compressor were not modelled. It was decided to take the assumption by Abrol and Hilton (2012), that the temperature and pressure conditions of these units were at their desired values, thus leading to the temperature or pressure changes as needed, but not requiring modelling of their energy transfers.

3.2.1. Reactor Model

The methanol reactor was based on the commercial type, i.e., a tubular reactor with a packed bed of catalyst and a counter-current coolant jacket (Chen *et al.*, 2011; Manenti, Cieri and Restelli, 2011). The fluid in the jacket was steam which provided the required cooling for the exothermic reactions. Income can be generated from the generation of this high-pressure steam.

The kinetic model used was described by Bussche and Froment 1996, as was recommended by Abrol and Hilton (2012) and Santos, Santos and Prata (2018). Full details of the kinetic equations can be found in Appendix A. The following assumptions were made for the model: unidimensional flow, steady-state operation, negligible axial diffusion, negligible heat conduction, no intra-particle diffusion limitations, no pressure drop across the reactor, and no accumulation of coolant in the jacket (Chen *et al.*, 2011; Manenti, Cieri and Restelli, 2011).. The temperature of the shell (coolant) was specified as a constant variable, with a minimum temperature approach of 10K taken from the lower bound of the changing reactor temperature. The mass and energy balance equations were derived from first principles (Green and Southard, 1997) and

were expressed by the following set of ordinary differential equations (ODEs), with the constants specified in Table 2.

$$\frac{dF_i}{dz} = N_t A_{int} \rho_{cat} (1 - \varepsilon_{cat}) \sum_{i=1}^N r_i v_i \quad (5)$$

$$\frac{\partial T}{\partial z} = \left(\rho_{cat} a \sum_{i=1}^n r_i (-\Delta H_{f,i}) + \frac{4}{D_{int}} U (T_{shell} - T) \right) \left(\frac{N_t A_{int} (1 - \varepsilon_{cat})}{\dot{f}_t c_p} \right) \quad (6)$$

Table 2. Constants used in the reactor mass and energy balance equations

Parameter	Symbol	Value	Units
Cross sectional area of reactor tube	A_{int}	0.00113	m ²
Void fraction of catalyst	ε_{cat}	0.39	
Density of catalyst	ρ_{cat}	1 100	kg/m ³
Activity of catalyst	a	1	
Internal diameter of reactor tube	D_{int}	0.038	m
Overall heat transfer co-efficient	U	631	W/m ² K
Temperature of boiling water on shell side	T_{shell}	511	K
Reactor length	z	7	m
Number of tubes	N_t	6 560	
Number of species	N	7	
Number of reactions	n	3	

The reactor model contains explicit differential equations that were solved using “jax.experimental.odeint” (Deepmind, 2021). This JAX based ODE solver, that JAX can differentiate, uses Dormand-Prince ODE integration with adaptive stepsize which is mixed 4th/5th order explicit Runge-Kutta method (Deepmind, 2021). The method uses six function evaluations to calculate 4th and 5th order accurate solutions (Deepmind, 2021). The difference between these solutions is used as the error estimate, which is then used in the adaptive stepsize calculations. The reactor model is non-stiff as there are no rapid variations in the solutions to the differential equations, thus making it suitable for non-stiff Dormand-Prince (Jakobsen, 1992).

3.2.2. Separator Model

The high-pressure separator was used to separate the product, methanol, from the unreacted gases which get recycled back. Therefore, the liquid stream produced was predominantly methanol and water with a small fraction of dissolved gases. The separator was modelled using the Pressure-Temperature Flash algorithm described by (Biegler, Grossmann, and Westerberg 1997).

The separator operates at very high pressure and temperature conditions, where the compounds deviate from ideality (Santos, Santos, and Prata 2018). Thus, to model the vessel accurately, a non-ideal vapour-liquid equilibrium was assumed. Peng Robinson equation of state (PR-EOS) was recommended for use by Santos, Santos, and Prata (2018) to calculate the components’ fugacity coefficients. The PR-EOS equations found in (Biegler, Grossmann, and Westerberg 1997) and (Poling *et al.*, 2001) were used. The fugacity coefficients were then used to calculate the K values (i.e., the vapour-liquid equilibrium) for each component. The feed composition, flash temperature and flash pressure were known inputs. This then allowed for the final vapour fraction $\left(\frac{F_v}{F}\right)$, i.e., the ratio of vapour flowrate to feed flowrate, to be calculated

by minimising the Rachford-Rice equation below (Equation 7). See Appendix B for a more detailed explanation of the iteration procedure.

$$\sum_{i=1}^N \frac{z_i(K_i-1)}{1+\left(\frac{F_v}{F}\right)(K_i-1)} = 0 \quad (7)$$

Newton’s method was used to find the approximate solution to the above Equation 7. The loop was written in such a way that it was compatible with JAX automatic differentiation and just-in-time compilation. In solving the PR-EOS, NumPy’s polynomial root finder was used to find the roots of the cubic equations for the vapour and liquid phase (Z^V and Z^L). These roots were then used to calculate the fugacity coefficients for each component.

All the numerical methods that were used in the process model are summarised below.

Table 3. Summary of numerical methods

Part of Model	Numerical Method	Package
Set of ODEs describing reactor mass and energy balance	Dormand-Prince Runge-Kutta	JAX.experimental.odeint
Minimisation of Rachford-Rice Equation to calculate vapour fraction of feed	Newton’s method	Package not used, and method coded manually
Finding the roots of the cubic equation in Peng Robinson equations of state	Algorithm uses eigenvalues of the companion matrix to find the roots.	NumPy.roots

3.3. Model Validation

Various comparisons were made to ensure the accuracy and validity of the coded reactor and separator models. Firstly, to check the implementation of the reactor model, the outlet flowrates and temperature were compared to industrial data of the same reactor type, from Chen *et al.* (2011). To ensure the results were comparable and that the comparison was fair, the same reactor operational conditions, feed specifications, shell temperature, assumptions for the heat transfer coefficient, catalyst data, and reactor dimensions made by Chen *et al.* (2011), were specified into the code.

To confirm the validity of the separator model and the implementation of the PR-EOS, the K values for each component were compared to the K values obtained from COCO. The reason why K values were used for model validation, was twofold. Firstly, they are the main parameters on which the separator’s modelling depends. Secondly, the accurate K value calculations require the separator loop to work both in 1) iteratively finding the K values and vapour fraction, and in 2) finding the roots to the cubic equations in the EOS.

To ensure a fair comparison, the same feed was specified for the COCO separator as was for the coded model. The separator in COCO was specified to use PR-EOS. An example of a typical feed was used as a basis for the feed and pressure specifications, to test the K values at.

3.4. Formulation of Problem

3.4.1. Implementation of Model

To implement the above models into Python code, the model was structured and compartmentalised using classes, shown in Figure 5 below. Classes were used as they acted as “blueprints” for the species’ thermodynamic data, reactor model, separator model, and mixer/splitter model. The attributes (variables) of these classes were constants (such as catalyst density), whilst the functions were the equations that describe the model (such as the reactor’s mass balance). The class structure allowed the code to be broken into sections, and then for these sections to share information between themselves. For example, the objective function (under Heading 1. Process Class – Figure 5) requires the solving of the reactor class (Heading 2.1) which contains a mass balance that depends on the specific heat (under Heading 3.1 Species Class), which it then receives from the species’ thermodynamic data class.

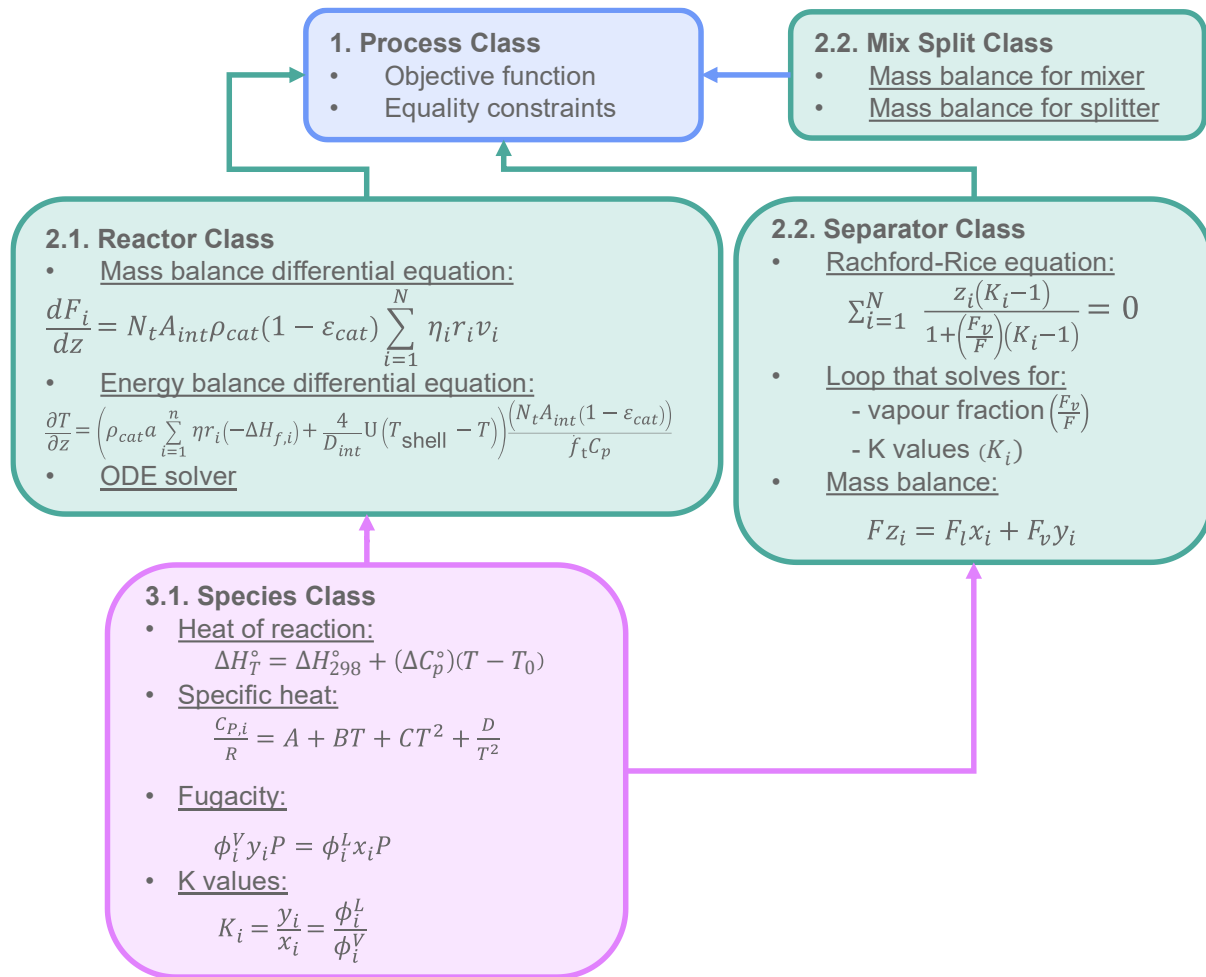


Figure 5. Class structure of coded model

3.4.2. Optimisation Problem – Objective Function, Constraints, and Bounds

The optimisation problem was formulated in terms of two objective functions, constraints, and bounds on the decision variables. These were then kept constant throughout the testing.

The first objective function was chosen to be the maximisation of the methanol flowrate exiting the separator. As the syngas feed flowrate is fixed, this is equivalent to

maximising the carbon efficiency of the process. Methanol production as the objective function was chosen for three reasons. Firstly, it was representative of the overall economic and environmental goal of the process, secondly it was similarly performed by the following papers (Kordabadi and Jahanmiri, 2005, 2007; Lim *et al.*, 2010; Luyben, 2010; Santos, Santos and Prata, 2018) – which would, therefore, make for fair comparisons at the end, and thirdly, it allowed for simplification of the problem where detailed costing was not required.

The second objective function was chosen to be based on economics, which would then maximise the profit of the process. The main reason for looking at cost was that it was a more relevant, relatable, and important description of the process. Therefore, to illustrate the profit, the function was written as the subtraction of the costs, from the income generated. The costs of the flowsheet included both the capital costs (the catalyst, reactor preheater, reactor product cooler, and compressor costs), and the energy costs (the compressor work, steam for the reactor preheating, and cooling water for the reactor product cooling). The total annualized cost (TAC) was then calculated using the total capital cost and assuming a payback period of 3 years. The incomes generated from the flowsheet were the sale of methanol, the high-pressure steam generated in the reactor, and the heating value of the purge stream which can be burned to recover heat. A yearly basis was assumed, and all the costing formulas and figures were obtained from (Luyben, 2010).

The two objective functions differ in complexity. This is useful as it allows for the optimisation method's performance to be compared on easy problem (methanol production) and on a hard problem (cost of process). The cost function is a more complex function as it contains exponential and logarithm functions. The capital costs are an example of an exponential part of the function. The energy costs (that depend on the log mean temperature difference) are an example of a logarithmic part of the function. See Appendix C and D for the detailed mathematical equations of both objective functions.

The objective functions were expressed as equations that depend on decision variables. The values of the decision variables were manipulated in the optimisation process to find the optimal point. The decision variables chosen for optimisation were the reactor inlet temperature, the temperature and pressure of the separator, the recycle ratio, and the ratio of CO to the total carbon fed in the syngas feed. These were chosen as the most important operating parameters that have the highest impact on the amount of methanol produced, and overall cost of the process. Equation 8 and 9 below, define the calculation for the recycle and carbon ratio respectively, with the stream numbers relating to the flowsheet in Figure 1.

$$\text{recycle ratio} = \frac{\text{flowrate of stream recycled}}{\text{total flowrate to splitter}} = \frac{F_9}{F_7} \quad (8)$$

$$\text{carbon ratio} = \frac{\text{syngas CO flowrate}}{\text{syngas CO flowrate} + \text{syngas CO}_2 \text{ flowrate}} = \frac{F_{1,\text{CO}}}{F_{1,\text{CO}} + F_{1,\text{CO}_2}} \quad (9)$$

The constraints of the optimisation problem were the mass and energy balances across the process units. The result of this was that the recycle flowrate was solved for simultaneously with the optimisation.

Various assumptions were made regarding the bounds of the decision variables. The assumptions made for the bounds were based on what was realistically and safely achievable in industry, as well as what ensured that the optimisation problem remained

computationally stable. Heuristics that defined safe operating conditions, were used for the bounds on the reactor temperature, separator temperature, and separator pressure (Turton *et al.*, 2003; Fogler, 2019). The maximum pressure of the separator was set to be 5 bar less than the maximum operating pressure of the reactor. The reason for this was that a flash tank utilises a pressure drop to separate components. Therefore, it must operate at a lower pressure than its feed (which is the reactor effluent). It was assumed that, to avoid inert accumulation, the maximum amount recycled was 99% of the flowrate to the splitter (i.e., a 1% purge). The bounds for the carbon ratio were assumed to be between 0 and 1. Full description of the assumptions made, and bounds used in the optimisation, can be found in Appendix D.

An important bound for ensuring stability of the problem was that of the recycle ratio. The upper bound of the recycle ratio was chosen to ensure that the problem remained stable and did not become ill-conditioned. An ill-conditioned problem has a large change in the answer, for a small change in the input or independent variables (Heath, 2002a). A small change in the recycle ratio when the recycle ratio is large, leads to a large change in the mass balances across the process units. In other words, as the recycle ratio becomes larger, it becomes harder to find the correct value of the streams that ensure that the process balances. This is especially true for the process model at hand. The model is heavily non-linear which makes the problem more sensitive to becoming ill-conditioned. The upper bound was therefore set at 0.8 to ensure that this did not occur. The value (0.8) was found by altering the upper bound of the recycle ratio and choosing the highest value that remained stable when optimisation was run.

The performances of the optimisation methods were compared by looking at their efficiency to find the optimum point and the strength of the optimal point found (i.e., the value of the objective function). The efficiency was determined by the number of iterations, the number of function evaluations, and the number of Jacobian function evaluations (except for differential evolution as it did not utilise differentiation). Additionally, the computational times taken for each method were analysed. For all these metrics, the smallest value was indicative of a more efficient and faster-performing optimisation method.

A base case was used to compare the optimisation results for an analysis of the improvements made. The conditions of the base case were based on literature (Chen *et al.*, 2011; Santos, Santos and Prata, 2018) and full details can be seen in Appendix E. The conditions of the base case were additionally used as the initial guesses for both the decision variables and stream flowrates, that were then supplied to the various optimisation routines.

3.5. Optimisation Performance Tests

Better optimisation can be determined from three main qualities: the ability of the model to accurately reflect the real process, the speed and efficiency of the method in finding the optimal point, and the value of the found optimum. The following sections first discuss the comparisons that were made regarding speed and efficiency of the method. Thereafter, the analysis of value of the found optimum is discussed.

3.5.1. Computational Speed

Python is one of the most widely used coding languages. It is especially useful for process modelling. This is largely due to its extensive library support that includes a number for numerical calculations, such as NumPy. The result of this is that numerical tools, such as polynomial root finders, ODE solvers, and optimisation routines, are

widely available and easy to implement. This also means that model development can be done relatively quickly. Python does, however, use an interpreter and is dynamically typed. The result of this is that the source code is compiled at run time which slows down the speed of computation. Thus, the execution of the process model is slow in Python. For modern process modelling, both simulation and optimisation are required to run as fast as possible, especially as the size and complexity of problems is ever increasing. However, JAX can bring the required computational performance to Python through jit compilation, and thus, addresses its major downfall.

JAX can jit compile Python code using XLA. This results in a dramatic speedup of code execution time. The first time the code is run, JAX compiles the code. From the second run onwards, it only re-compiles the code if the speedup gained is greater than the overheads of compilation (Nain, 2021). In other words, the first run serves as a “warmup”, with the subsequent running the compiled version (Nain, 2021). This leads to more efficient execution of Python functions.

To demonstrate this increased computational efficiency, the computational speeds taken to solve the mass and energy balances across the reactor model, as well as the mass balance across the separator model, were analysed. The solving of these models is complex. For the reactor model, the complexity was due to highly exponential nature of the kinetic equations, and dependence on ODEs. For the separator, the complexity was due to the loop used to calculate the vapour fraction of the feed and the K values, both of which require finding the roots of a cubic equation. JAX’s jit does, however, introduce overhead itself. Therefore, it should only be used if the compiled function is complex, and the code run numerous times. The applicability to process modelling is therefore highly relevant because the models are often complex, as seen in this example, and computed numerous times in an optimisation routine.

The computational time was measured for the model to solve without jit compilation, and then with jit compilation. When the code was jit compiled, the run time was measured, as well as the compile and run time. The run time was measured by timing the code once it had undergone “warmup” or compilation. This was done to illustrate the point that the first time the code is called, it takes longer due to the compilation. From the second time onwards, the run time is much less due to it using the compiled code. The computational time was measured for 100 runs of the code for without compilation, and with compilation. The average of these runs was then compared for each scenario as there is variation in computational time of computers (Nogueira, Matias and Vicente, 2014). The compiled and non-compiled models were solved under the same conditions. This included the computer being plugged in to a power source and operating at a constant temperature – both of which conditions affect computer performance. The test was used as a proof of concept that JAX can increase computational speed. Further work could be done on performing the test on a higher level of accuracy, but this was outside the scope of this thesis.

3.5.2. Derivative vs Non-Derivative Optimisation

As a proof of concept that derivative-using methods have superior performance to non-derivative using methods, the performance of standard Ipopt was compared to differential evolution for a simple version of the process. Differential evolution was investigated as a baseline, representing the current method used by majority of studies on this specific process optimisation. The simple version of the process assumes ideal gases and only optimises reactor temperature, separator temperature, and separator

pressure. A constant feed composition and a constant recycle ratio was specified. The objective function was methanol production in the separator. A simple version of the process was utilised as 1) it was a proof of concept 2) if there were significant differences in performance for a simple problem, then there would be even greater on more complex problem 3) differential evolution was highly unstable on the complex problem and therefore could not be implemented without alteration to the bounds and starting guesses, which would have made the comparisons unfair.

3.5.3. Ipopt with Numerical Differentiation vs Automatic Differentiation

The optimisation performance of Ipopt when the with derivatives supplied were calculated using numerical differentiation (ND), was compared to when the derivatives supplied were calculated using automatic differentiation (AD). The performance metrics were used to analyse the impact of a more accurate derivative calculated by automatic differentiation. In theory, a more accurate derivative, which is used to calculate a more accurate step direction, should result in Ipopt converging on the optimum faster.

Ipopt has an option to supply both the derivatives of the objective function and the derivatives of the constraints. If the derivatives are not supplied, Ipopt uses ND to calculate these values. Optimisation methods that are self-contained, i.e., Ipopt with no information regarding derivatives supplied, are computationally faster than a method that utilises code supplied by the user. This is because when the code is supplied by the user, it must “step outside” its routine. Thus, to compare the computational speed of Ipopt when using ND versus AD, the derivatives were supplied by the user in both cases. This eliminated the difference that is made by the optimisation methods being self-contained or not.

AD was implemented using JAX library which has various functions for calculating gradients, Jacobians, and Hessians of functions. JAX can differentiate through various functions including ODE solvers, loops, and polynomial root finders. Thus, JAX can effectively handle all the complexities of the process model.

ND was implemented using the NumDiff Python Library. NumDiff calculates the numerical derivative in an adaptive manner, coupled with Richardson extrapolation methodology, to provide a maximally accurate output. All the tools that could be combined to increase ND’s performance, were therefore, used. To ensure fair testing, only Python libraries were used to compute the derivatives of the objective function and constraints for both AD and ND. If one had not been implemented using a library, then 1) the computational speeds would be unfairly comparable and 2) the difference in optimisation performance may be due to inefficient coding of the method.

Numerical differentiation requires the complicated choice of the step size to use in its computation. The following Equation 10 can be used to estimate the step size (h) that is small without producing a large roundoff error (Edgar, Himmelblau and Lasdon, 1988).

$$h = \sqrt{\varepsilon}x \quad (10)$$

where ε (machine epsilon) = 2.2×10^{-16}

Using the above equation, it was estimated to be equal to 10^{-6} . This value was kept constant and used throughout the thesis where ND was required. Despite the relationship being known, this thesis did not investigate the impact that the value of

the step size has on the optimiser’s performance. The best value for the step size could, however, be found by performing the optimisation multiple times for different values of the step size, and then choosing the value of the step size that would lead to the most efficient optimisation. The process of finding the best step size is computationally expensive, especially as problems get harder. This is a disadvantage of ND that AD does not face. Therefore, the use of the default estimate of the step size was a fair comparison.

3.5.4. Optimal Point Comparisons to Literature

For proof that the optimisation performed in this thesis achieved an improvement to the optimal points previously found for the process, comparisons were made with literature. Two literature sources, Santos, Santos, and Prata (2018) and Chen *et al.* (2011) were chosen as fair comparisons. The reasons for which are outlined below.

Chen *et al.* (2011) performed optimisation on a commercial Lurgi reactor, which was the same reactor type used as a basis for this thesis. Chen *et al.* (2011) formulated an objective function to maximise the methanol production rate exiting the reactor. Santos, Santos and Prata (2018) on the other hand, performed optimisation on the entire process flowsheet to maximise the methanol production rate exiting the separator. Both Chen *et al.* (2011) and Santos, Santos and Prata (2018) used the same reactor and catalyst specifications, as well as the same reaction kinetics described by Bussche and Froment (1996). This brought ease when implementing their assumptions into the code, as well as made for a fair comparison of the amount of methanol produced after optimisation.

To accurately compare whether this thesis was able to produce a greater amount of methanol, and hence, achieve a better optimisation for the process, the methanol production rate needed to be normalised to the different feeds used in both Chen *et al.* (2011) and Santos, Santos and Prata (2018). Normalisation was done for the final methanol flowrate leaving the process, over the total carbon content fed to the process (i.e., the conversion). For Chen *et al.* (2011), this meant the ratio of the methanol exiting the reactor to the total carbon fed into the reactor (Equation 11 below). For Santos, Santos and Prata (2018), this meant the ratio of the methanol exiting the separator to the total carbon fed into their bio reformer, i.e., biogas (Equation 13). For this thesis simulation, it meant the ratio of the methanol exiting the separator to the total carbon fed in the syngas fed to the process loop (Equation 11).

$$Conversion = \frac{\text{Optimum } F_{CH_3OH} \text{ in separator outlet}}{(F_{CO} + F_{CO_2} + F_{CH_4} + F_{CH_3OH}) \text{ in syngas feed to loop}} \quad (11)$$

$$Conversion \text{ Per Pass} = \frac{\text{Optimum } F_{CH_3OH} \text{ in reactor outlet}}{(F_{CO} + F_{CO_2} + F_{CH_4} + F_{CH_3OH}) \text{ in syngas feed to reactor}} \quad (12)$$

$$Conversion = \frac{\text{Optimum } F_{CH_3OH} \text{ in separator outlet}}{(F_{CO} + F_{CO_2} + F_{CH_4} + F_{CH_3OH}) \text{ in biogas feed}} \quad (13)$$

4. Results and Discussion

4.1. Model Validation

Model validation was done for the reactor and separator models. Column 5 in Table 4 shows that the relative errors between the industrial data and the thesis simulation of the reactor model, were small. This concluded that the reactor was coded and modelled correctly, as the results obtained were like that of a real, industrial reactor.

Table 4. Comparison of thesis simulation with industrial data (Chen et al. 2011)

	Feed Specifications	Outlet Stream (thesis simulation)	Outlet Stream (industry)	Relative Error (%)
Temperature (K)	498	530	528	0.3788
Component's flowrate (kg/h)				
CO	10728	4955.4	4921.0	0.69904
CO ₂	23684	18317	18316	0.00546
CH ₃ OH	756.70	11268	11283	0.13294
H ₂	9586.5	8018.2	8013.7	0.05615
H ₂ O	108.80	2305.7	2309.3	0.15589
CH ₄	4333.1	4333.1	4333.1	0.00000
N ₂	8071.9	8071.9	8071.9	0.00000

For a visual representation of the results from the reactor model, the component's flowrates (Figure 6), and the temperature profile along the reactor length (Figure 7), were plotted. H₂ has a significantly larger flowrate than all the other components. For a closer view of the shape of the components' profiles, H₂ was therefore excluded in Figure 6 below. However, it is included in Figure 12 in Appendix F.

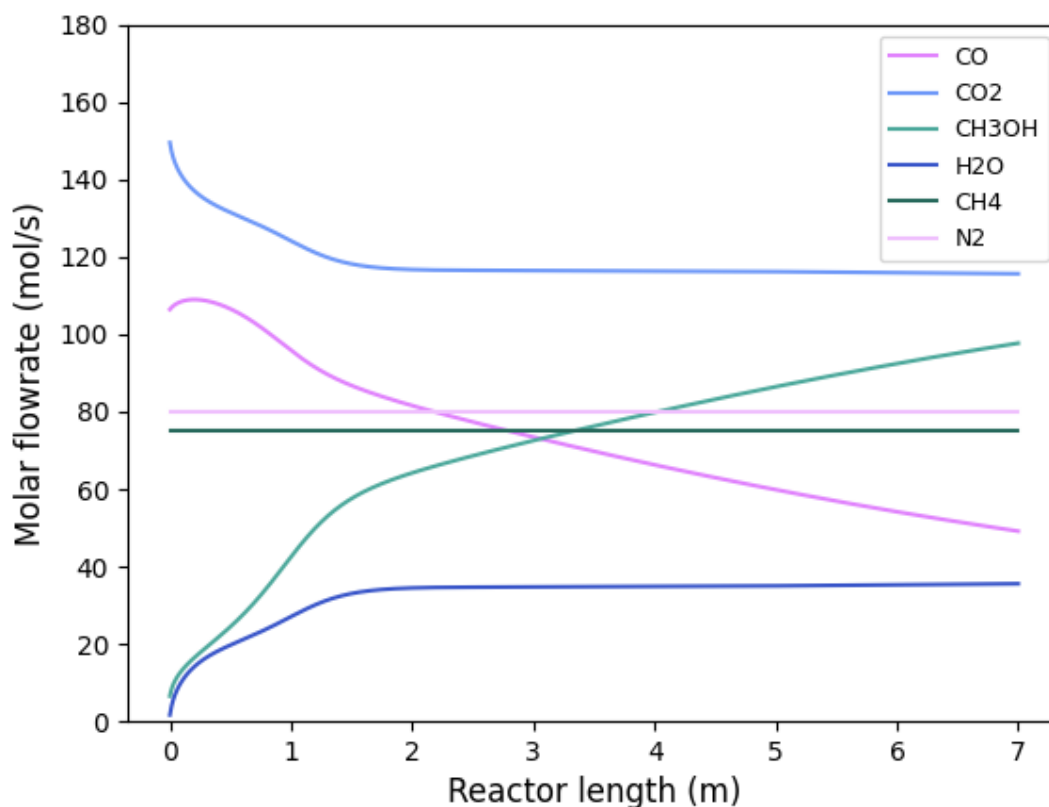


Figure 6. Component molar flowrates along the reactor length (excluding hydrogen)

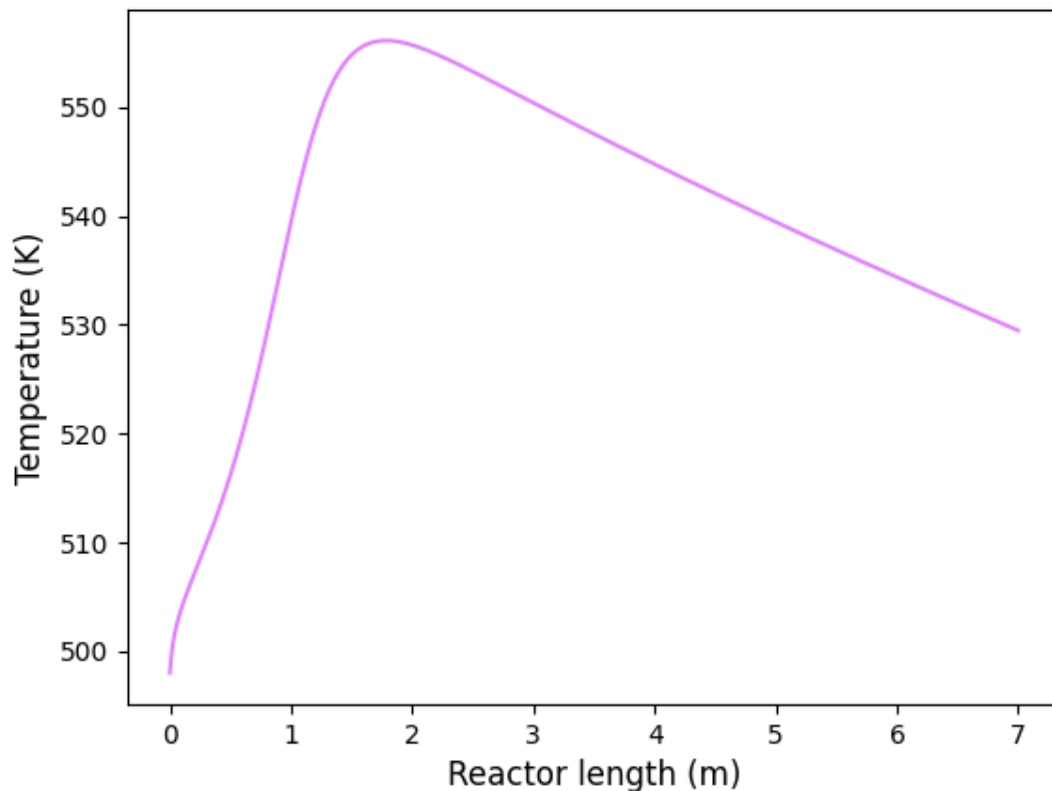


Figure 7. Temperature profile along the reactor length

It was seen from Figure 8 below, that the K values obtained from the separator model were similar to those of COCO, across a range of different temperatures for a constant pressure of 60 bar. This confirmed that the EOS were correctly implemented in the separator model. The differences between the K values were likely due to the usage of marginally different constants that appear in the K value calculation. The constants include the gas constant (R), critical temperature (T_c), critical pressure (P_c), and acentric factors (w) for each component. The K values are sensitive to these values and therefore, the differences in the number of significant figures could have impacted the differences in the final K values.

Flash separation simulation depends predominantly on the K values. Thus, by having K values like COCO, the validity of the separator model was confirmed.

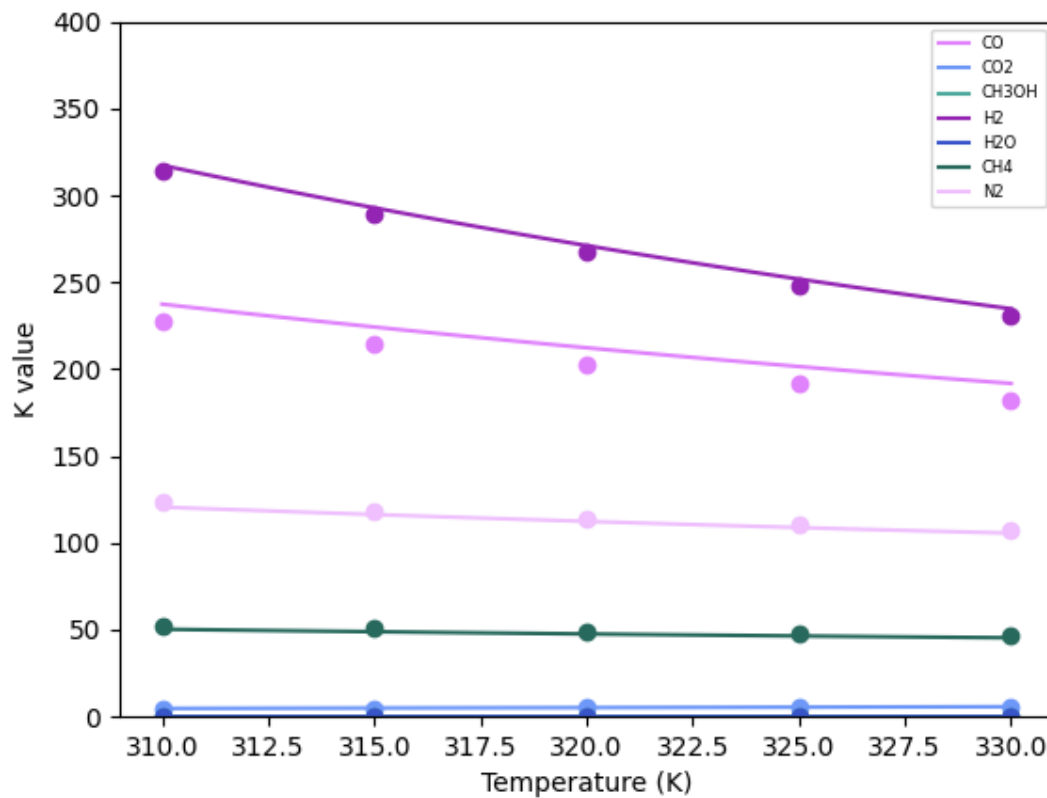


Figure 8. Comparison of *K* values obtained from thesis simulation (line) to those from obtained from COCO (dot)

4.2. Optimisation Performance

4.2.1. Computational Speed

The speed that just-in-time compilation is able bring to process modelling in Python can be seen in the figure below. The run time without compilation, was 1000 times greater than the run time with compilation.

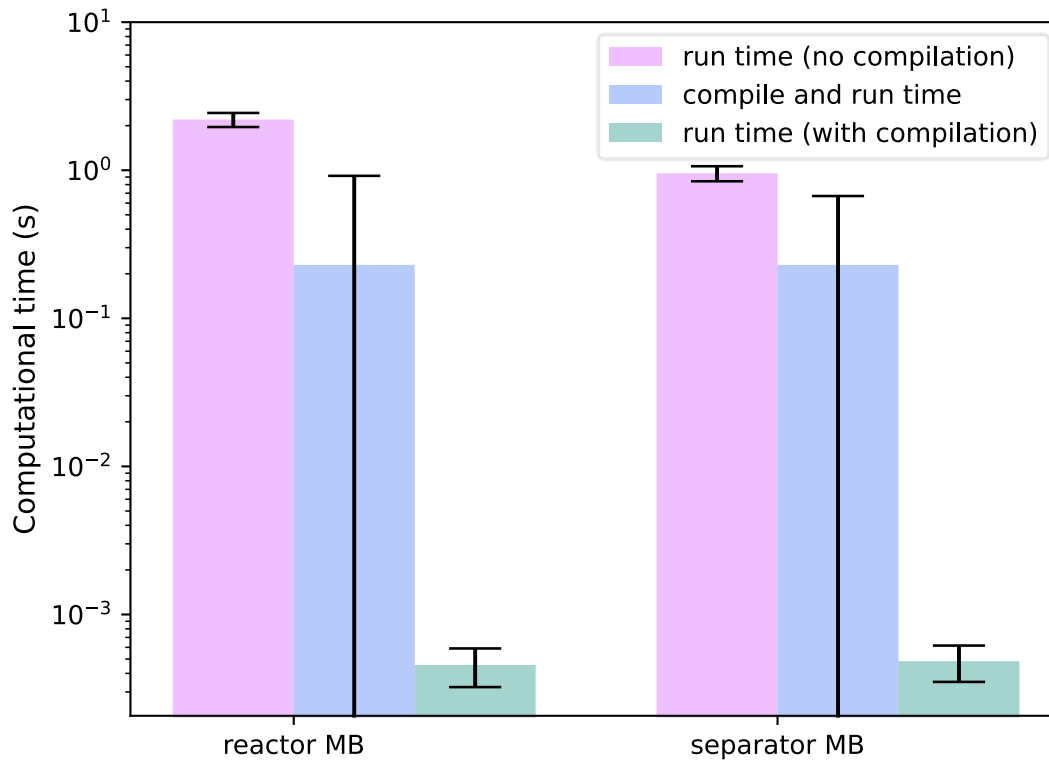


Figure 9. Comparison of computational time taken to solve models without just-in-time compilation and with just-in-time compilation

The reason why there was greater variation in the compile and run time, was due to its first run. The first time the model runs, JAX scans and compiles the function. From then onwards, it only re-compiles the code when the increased computational speed outweighs the overhead of compilation. Thus, the first run always has a significantly greater computational time than the rest of the runs – which is what leads to a greater variation. This was confirmed by the run time of the compiled code (green graph above) that showed less variation, and whose average was smaller than that of the compile and run time (blue graph). To further illustrate this point, the time taken for each run was plotted in Figure 13 in Appendix G.

The separator’s reduced time was in part because of jit compilation and in part because of “scanning” its loop. If a loop is jitted, then JAX “unrolls” the code to figure how best to optimise the running of the code when it compiles it. When it “unrolls” the code it converts the for-loop to long repeated code. JAX, therefore, must compile lots of repeated code. This leads to inefficiency that linearly increases as the loop’s length increases. One way to address this issue is to use `JAX.lax.scan`. It communicates to `JAX.jit`, telling it that it can compile the inside of the loop once, and then re-use it. `JAX.lax.scan` prevents JAX from unrolling the loop for each step and instead compiles the code once.

The impact of an increased computational efficiency to process modelling in Python, is significant. The models in this process were relatively simple compared to the high levels of complexity that can occur when more factors are considered (such as fluid flow inside a reactor). Models can therefore take significantly long to solve. Thus, being able to reduce the computational time 1000 times, means that faster optimisation can be done, or that other complexities can be added without excessive computational time overheads. This thesis therefore suggests that JAX can be used to address the slowness of Python and allow for more efficient modelling and optimisation to be done in the language.

4.2.2. Derivative vs Non-Derivative Optimisation

A summary of the simple optimisation problem used for DE and Ipopt, is shown in Table 5 below. It was seen that Ipopt, the derivative-using method, was significantly more efficient in finding the solution than DE, the non-derivative method. This was seen by the fact that Ipopt with AD required 100th of the number of iterations that DE required. If the maximum number of iterations was set at a higher value, then this would be even greater. Both methods, however, did find the same optimum where methanol production had a maximal flowrate of 129.4 mol/s.

Table 5. Summary of comparison of DE and Ipopt

		Differential Evolution	Ipopt
Use of derivative		No	Yes, AD
Optimiser Performance	Number of iterations	1000 (max)	9
	Number of function evaluations	1001	10
Decision Variable value at optimum	Reactor temperature	501.3 K	501.3 K
	Separator temperature	308 K	308 K
	Separator pressure	75.0 bar	75.0 bar
Objective function value at optimum	Methanol production rate	129.4 mol/s	129.4 mol/s

The results confirmed the expectation that differential evolution, a non-derivative method, is slower than a derivative-using method, Ipopt. DE is a stochastic direct search method that searches for a set of random points around the current point, and then selects the point with the best value of the objective function. It is therefore, not focused on each iteration. Methods that use derivatives ensure that each iteration is stepping towards the best search direction, based on the shape of the local surface of the current point. Thereby ensuring that the optimum is found relatively quickly. DE iterations randomly jump across the solution space. This explains why many iterations are therefore, required to find the optimum.

The main reason that non-derivative methods are used in process optimisation is that they do not require the process to be differentiable. Thus, they can handle complexities such as ODE solvers inside the model. They are however significantly less efficient in finding the optimum - which is clearly shown from the above results.

4.2.3. Ipopt with Numerical Differentiation vs Automatic Differentiation

The performance of Ipopt was analysed for two different objective functions using derivatives supplied externally through numerical differentiation and automatic differentiation. The first objective function was the methanol production flowrate and the second was the cost function of the process. The results are summarised in Table 6 below.

Table 6. Ipopt with AD and ND comparison for both objective functions

	Objective Function			
	Methanol Production		Cost	
Performance	Ipopt with ND	Ipopt with AD	Ipopt with ND	Ipopt with AD
No. iterations	42	17	116	19
No. function evaluations	43	18	369	20
No. Jacobian evaluations	44	19	119	21
Objective function final value	184.5 mol/s	184.5 mol/s	\$115 000 000	\$115 000 000

On the simple objective function (methanol production), the methods performed more similarly. Ipopt with ND was twice as slow as Ipopt with AD. On the complex objective function (cost), Ipopt with AD outperformed Ipopt with ND. Ipopt with AD required 1/6th of the number of iterations and 1/18th of number of function evaluation than Ipopt with ND. When the objective function was simple, then the optimum was much easier to find. However, when the cost function was solved – which was significantly more complex due to the logs and exponential functions it contained, the methods needed much greater accuracy in their step direction. This then led to the difference in performance of Ipopt with AD, compared to Ipopt with ND.

It was also seen from Figure 10 below, that the computational time taken by Ipopt using ND was significantly higher than that for AD, for both objective functions. For the methanol objective function, where both the methods performed more similarly, the time difference was due to Ipopt with AD being more stable. For the cost objective function, the time difference was due to the method requiring fewer iterations, function evaluations, and Jacobian evaluations.

Taking a closer look at the cost objective function comparison, the number of iterations, function evaluations, and Jacobian evaluations were compared in Figure 11 below. The result showed that Ipopt performed significantly better when the derivatives were supplied using AD. The reason was due to the increased accuracy of AD, as the method does not deal with the truncation and round-off errors that are faced by ND.

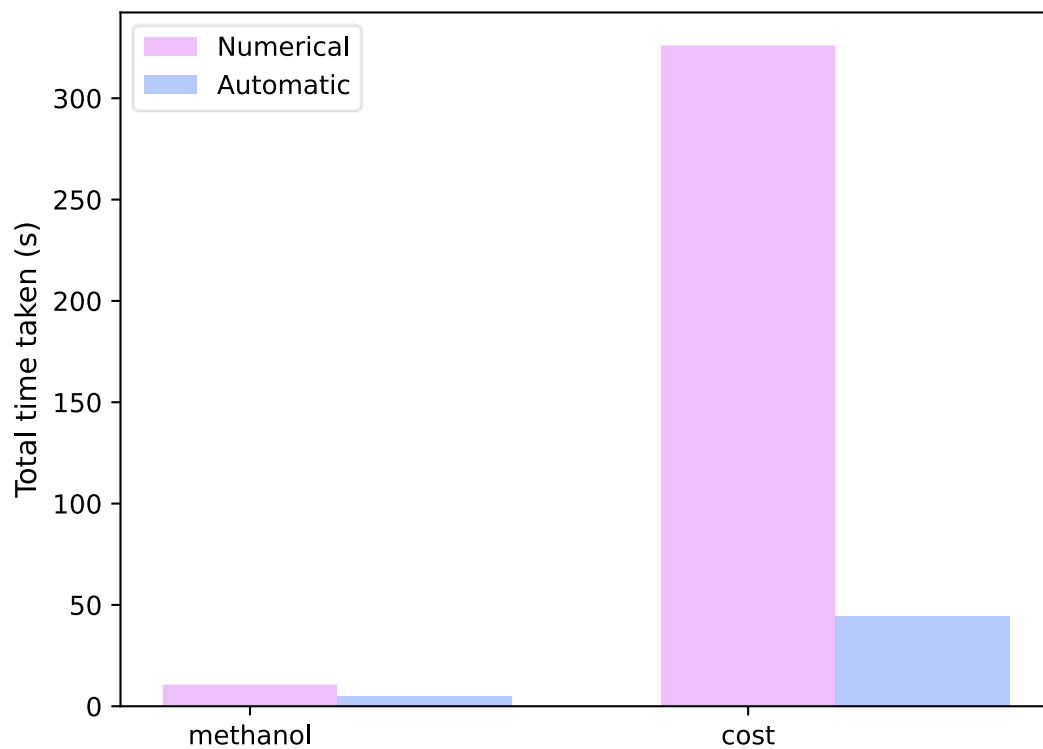


Figure 10. Computational time taken by Ipopt method using ND vs. AD

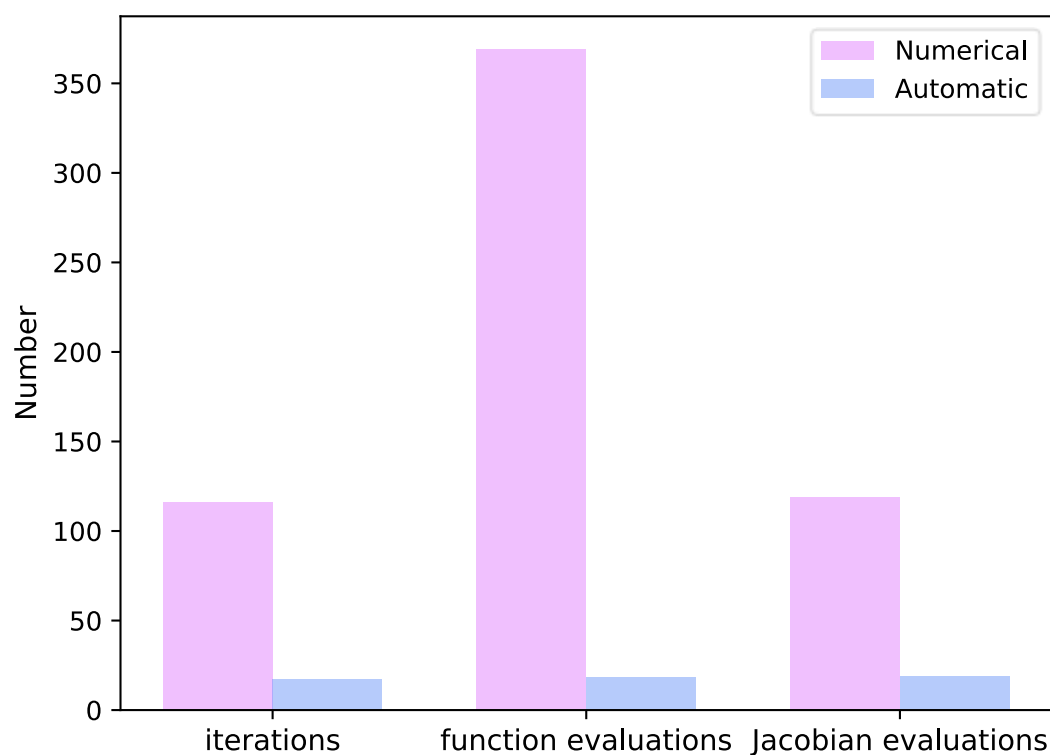


Figure 11. Optimisation performance for Ipopt with ND vs. AD for the cost objective function

The difference in the two types was seen when the iterations reached near the optimal. Ipopt using AD, efficiently converged on the optimal value. However, Ipopt using ND failed to do so. When the methods were far from the optimal point, the step direction could afford to be less accurate. As the methods close in on the optimal, accuracy, however, becomes increasingly more important. Ipopt using ND struggled to take steps to the optimal point when the accuracy of its derivatives was required. The result of this was that it struggled to converge on the optimum, thus requiring more iterations, function evaluations, and Jacobian evaluations.

An analogy can be made to a hiker trying to find the highest point in a landscape. When the hiker is far from the peak, the exact accuracy of their step direction is not required for them to make progress towards nearing the peak. But, as the hiker gets closer to the peak, the accuracy of their step direction is essential to them finding the exact, maximum (highest) point of the peak. If the hiker’s direction is slightly off, the hiker will step past the peak or towards the side of the peak, and not to exactly on the peak.

The data showed how AD can effectively and accurately compute the derivatives of the models in the process. It therefore enabled Ipopt to perform at a higher level, compared to when ND was used. The overall result of this was the proof that optimisation can be performed better and faster.

4.2.4. Evaluation of the Optimal Point Found

The results of the optimisation were compared to the base case as well as to results of optimisation studies done by others. The following Table 7 shows the improvements that were made to the base case through optimisation.

Table 7. Comparison of base case and optimal case

		Base Case	Optimal Case	
Objective Function			Methanol production	Cost
Methanol Production	mol/s	125.8	184.5	184.2
Process Cost	\$/yr	79 990 000	113 200 000	115 000 000
Overall conversion		0.3727	0.5469	0.5458
Reactor inlet temperature	K	500.0	503.3	502.2
Separator temperature	K	308.0	308.0	308.0
Separator pressure	bar	60.0	65.6	75.00
Recycle ratio		0.5000	0.8000	0.8000
Carbon ratio		0.5290	0.6600	0.6122

The methanol production improved by 47% and the cost by 42%. This shows that optimisation was successful and substantial improvements were made to operation of the process.

For reasons discussed under Heading 3.4.4, Chen et al. (2011) and Santos, Santos, and Prata (2018) were used as literature comparisons for the optimal points found by the thesis simulation. The normalisation of the methanol flowrates produced after optimisation, to the total carbon content fed to the process, were calculated using the Equations 8-10 specified under Heading 3.4.4, and compared below:

Table 8. Comparison of scaled methanol production to Chen et al. (2011) and Santos, Santos, and Prata (2018)

	Thesis Simulation	Chen (2011)	Santos (2018)
Conversion Per Pass	0.547	0.286	0.388

The larger ratios yielded by this simulation proved that a larger methanol production rate was achieved, and therefore, the optimisation performed was better than that of the two literature sources. Using this conclusion, the optimal points found for each decision variable were contrasted in Table 9 below. The different optimisation techniques used were also specified.

Table 9. Comparison of optimal points found, and optimisation techniques used

			Thesis Simulation	Santos	Chen
	Optimisation technique		Ipopt supplied with Jacobian calculated using AD.	Genetic algorithm	Sequential Quadratic Programming
Decision Variables	Optimal reactor temperature	K	503	474.5	509
	Optimal reactor pressure	bar	-	80.0	-
	Optimal separator temperature	K	300	298.5	-
	Optimal separator pressure	bar	80.0	72.9	-
	Recycle Ratio		0.80	-	-
	Carbon Ratio		0.72	-	-

The main difference between the optimisation performed by the thesis simulation, compared to that performed by Chen *et al.* (2011), and Santos, Santos and Prata (2018), stemmed not only from the fact that different optimisation methods were used, but that additional decision variables were included in the thesis simulation. The thesis simulation’s inclusion of the recycle ratio and the carbon ratio as additional optimisation variables, would explain the reason for the better optimisation achieved:

1. By optimising the recycle ratio, the amount of methanol produced from the objective function would increase more than if the recycle ratio was fixed. The

exclusion of the recycle ratio as a decision variable could be due to the added complexity the recycle loop brings to the model, making it much harder to solve when optimisation is performed, and hence, more computationally expensive.

2. Additionally, the thesis simulation’s inclusion of the carbon ratio would also allow for an increased methanol production via the various catalytic reactions.

It was found that a high carbon ratio (0.72) was more optimal. This is plausible, because if too much CO₂ is fed in the feed, there is an increased risk of there being too much water produced, and therefore, an increased risk of catalyst inhibition by absorbed water. Additionally, it was seen that a higher conversion and maximum production of methanol occurred when the recycle ratio was at its highest (0.8). A higher recycle ratio, however, brings additional CO₂. Therefore, the optimal higher ratio of CO in the syngas feed made sense as it ensured that the reactor feed contained a reasonable amount of CO₂ – limiting the production of excess water via the reverse water gas shift reaction.

It was found that the optimal temperatures of the reactor inlet and outlet were heavily dependent on the temperature of the shell (coolant). The temperature of the shell was not included as an optimisation variable in this simulation, but rather as a constant due to the limited time frame of this thesis.

4.3. Issues with JAX

A disadvantage with JAX, is that the model must be written in a way that is compatible with it. The constraint that this poses on model formulation, is that the functions must be pure. A pure function is one whose output value follows solely from its input value, without any side-effects. A pure function is like the mathematical idea of a function. For example, the function $\sin(x)$. The function returns the value of sine of x , and will always return the same value, for the same value of x . It does not produce any side-effects which are: setting a global variable or altering data structures, for example. The size of this constraint is dependent on the model at hand. It may result in JAX not being able to be used, or for the coder to have to work around the issue. For example, if the model required a stiff ODE solver, this problem may arise. However, in this thesis this constraint posed by JAX was slight, and it was something that the coder just needed to be aware of while composing the model.

A second constraint is that only JAX-compatible libraries (such as NumPy) can be utilised in the model. The result of this is that fewer libraries can be used and that the coder would need to manually code the function more often, rather than importing it. The reason for this is that JAX is new, and contributions are only made by interested developers when they are needed. There are, however, more libraries available to use with JAX, compared to traditionally used AD software such as GAMS.

5. Conclusions and Recommendations

In this thesis, JAX was used to improve process optimisation. This ultimately allowed for a better optimum to be found for the chosen case study, the CO₂ to methanol process. The improvement in optimisation was not only in the type of optimisation method used, but additionally in the computational speed.

First, the CO₂ to methanol process was developed into Python code to be used for subsequent testing. The process model was effectively created, implemented, and validated, by comparison to industrial data. The optimisation problem was then formulated using two different objective functions, one to maximise the methanol production and the other to maximise the process profit.

The thesis went on to demonstrate that just-in-time compilation can dramatically increase the computational speed of Python by a factor of 1000. This was proven on the specific process models investigated. JAX can, therefore, improve the computational efficiency of previously slow Python. Ultimately leading to the conclusion that chemical engineering processes can now be simulated in a simple-to-write Python code, while obtaining a highly performant, compiled code using JAX.

Two points were made regarding the best type of optimisation method. Firstly, it was demonstrated that optimisation methods that use derivatives were more efficient at finding the optimum than non-derivative methods. This was proven by the significantly fewer iterations that Ipopt required (a derivative method), versus differential evolution, (a non-derivative method). This was shown as a “proof of concept” that direct search methods (non-derivative methods) are not preferably chosen for their speed and efficiency, but rather they are chosen because they do not require the process model to be differentiable. Secondly, this thesis demonstrated that derivative-using optimisation, specifically using derivatives computed by automatic differentiation (AD), was better than using derivatives computed by numerical differentiation (ND). AD computes a more accurate derivative than ND, which results in an increased efficiency in the optimisation routine finding the optimum.

In the concluding parts of this thesis, the optimum that was found was analysed. Through optimisation, the methanol production flowrate and profit of the base case were increased by 47% and 42%, respectively. The overall conversion that was achieved through optimisation, was compared to that of two literature sources. Both sources looked at the same objective function in their optimisation studies for this process. This made the comparison fair. It was seen that this thesis’s simulation achieved a larger overall conversion, and therefore, a conclusion was drawn that better optimisation was performed by this thesis, than that of the other two sources.

To conclude, this thesis demonstrated that by using JAX, the computational efficiency of Python can be increased, as well as the performance of optimisation methods using AD. This led to better optimisation because each iteration of the method was calculated faster and more accurately. The overall result of this was that a better optimum was found, due to the ability to increase the complexity of the optimisation problem. This was successfully demonstrated on the environmentally important case study, that researchers have been attempting to optimise for years. Ultimately, this better optimisation can be used to improve the design and operation of processes, by maximising their performance.

5.1. Further Work

There are two main sections of further work: 1) the expansion of the optimisation done in this thesis (in terms of the CO₂ to methanol process) and 2) the investigation into the use of JAX and AD on harder optimisation problems.

This thesis focused on a simple proof of concept. It made simplifications to the modelling and the number of decision variables that were considered. Thus, there is significant room for further work to extend the scope of this project. The reactor model can be improved by looking at multi-stage reactors. The use of two, or even three stages, is popular in industry. One is typically used for cooling and the other for steam generation which has an associated income. Additionally, because the reactor's operation is highly dependent on heat transfer, further work should be done to model the shell temperature as a quantity that varies across the length of the reactor. This brings with it the option of making the shell temperature a decision variable. This would not only allow for the optimal shell temperature to be found, but also for a better (and more accurate) optimum for the reactor inlet and outlet temperatures.

Additionally, it is recommended to include more decision variables in the process optimisation. This would help truly find the most ideal operating conditions for the CO₂ to methanol process. The most important variable to add would be reactor length. This was assumed constant in this thesis. By fixing the reactor length, a limit is put on the amount of methanol that can be produced for a given feed. Further work should be done to include the reactor length as a variable, as it not only has a substantial effect on the methanol production, but also on the reactor capital cost. Other decision variables to include are the number of tubes in the reactor, the tube diameter, and the reactor pressure.

Further work should also be done on increasing the upper bound of the recycle ratio. To ensure that the problem does not become ill-conditioned, this could be performed iteratively. The optimisation would work by 1) running it using the current upper bound, then 2) the next answer found would be used as the next initial guess when optimisation is run again. This would result in an increase the upper bound of the recycle ratio slightly. The iteration process would continue until the upper bound was either increased to 99% (assuming a minimum of a 1 % purge is required), or until the problem becomes too ill-conditioned.

With regards to the second section of further work, this thesis proved an important concept that JAX and AD can be used to perform better optimisation. This was however done using a fairly simple process compared to the complexity of current industrial processes. It is therefore recommended that work is done to implement the optimisation techniques on harder problems such as mixed-integer and non-linear problems. This would provide further validation on the conclusion of this thesis.

6. References

- Abbott, A. S. *et al.* (2021) ‘Arbitrary-Order Derivatives of Quantum Chemical Methods via Automatic Differentiation’, *J. Phys. Chem. Lett.* 2021, 12, pp. 3232–3239. doi: 10.1021/acs.jpclett.1c00607.
- Abrol, S. and Hilton, C. M. (2012) ‘Modeling, simulation and advanced control of methanol production from variable synthesis gas feed’, *Computers and Chemical Engineering*, 40, pp. 117–131. doi: 10.1016/j.compchemeng.2012.02.005.
- Aides, A. and Kummerer, M. (2017) *Welcome to cyipopt’s documentation! — cyipopt 1.1.0 documentation*. Available at: <https://cyipopt.readthedocs.io/en/stable/> (Accessed: 21 November 2021).
- Arab Aboosadi, Z., Jahanmiri, A. H. and Rahimpour, M. R. (2011) ‘Optimization of tri-reformer reactor to produce synthesis gas for methanol production using differential evolution (DE) method’, *Applied Energy*, 88(8), pp. 2691–2701. doi: 10.1016/j.apenergy.2011.02.017.
- Aspen Engineering Suite (2004) *Getting Started Using Equation Oriented Modeling*. Available at: <https://sites.ualberta.ca/CMENG/che312/F06ChE416/HsysDocs/AspenPlus20041GettingStartedEOModeling>.
- Bajaj, I., Arora, A. and Hasan, M. M. F. (2021) ‘Black-Box Optimization: Methods and Applications’, *Springer Optimization and Its Applications*, 170, pp. 35–65. doi: 10.1007/978-3-030-66515-9_2.
- Baydin, A. G. *et al.* (2015) ‘Automatic differentiation in machine learning: a survey’, *Journal of Machine Learning Research*, 18, pp. 1–43. Available at: <https://arxiv.org/abs/1502.05767v4> (Accessed: 13 September 2021).
- Budden, D. and Hessel, M. (2020) *Using JAX to accelerate our research*. Available at: <https://deepmind.com/blog/article/using-jax-to-accelerate-our-research> (Accessed: 28 October 2021).
- Bussche, K. M. Vanden and Froment, G. F. (1996) ‘A Steady-State Kinetic Model for Methanol Synthesis and the Water Gas Shift Reaction on a Commercial Cu/ZnO/Al₂O₃ Catalyst’, *JOURNAL OF CATALYSIS*, 161, pp. 1–10.
- Chen, L. *et al.* (2011) ‘Optimization of Methanol Yield from a Lurgi Reactor’, *Chemical Engineering and Technology*, 34(5), pp. 817–822. doi: 10.1002/ceat.201000282.
- Deepmind (2021) *JAX reference documentation — JAX documentation*. Available at: <https://jax.readthedocs.io/en/latest/> (Accessed: 13 November 2021).
- Edgar, T., Himmelblau, D. and Lasdon, L. (1988) *Optimization of chemical processes*. McGraw Hill.
- Fogler, H. S. (2019) *Elements of Chemical Reaction Engineering, Series in the Physical and Chemical Engineering Sciences*.
- Freeman, C. D. *et al.* (2021) ‘Brax -- A Differentiable Physics Engine for Large Scale Rigid Body Simulation’, in *NeurIPS 2021 Datasets and Benchmarks Track*. Available at: <https://arxiv.org/abs/2106.13281v1> (Accessed: 13 September 2021).

Goeppert, A. *et al.* (2014) ‘Recycling of carbon dioxide to methanol and derived products-closing the loop’, *Chemical Society Reviews*, 43(23), pp. 7995–8048. doi: 10.1039/c4cs00122b.

Graaf, G. H. *et al.* (1990) *Intra-Particle Diffusion Limitations in Low-Pressure Methanol Synthesis*, *Chemical Engineering Science*.

Green, D. and Southard, M. (eds) (1997) *Perry’s Chemical Engineers’ Handbook*. Ninth. McGraw Hill.

Heath, M. (2002a) *Scientific Computing - An Introductory Survey*. Second. New York: McGraw Hill.

Heath, M. (2002b) *Scientific Computing - An Introductory Survey*. Second. New York: McGraw Hill. Available at: https://vula.uct.ac.za/access/content/group/d5a8660c-7419-4d05-b886-aac0ee4fcbcf/book/Heath2002_Scientific_Computing_An_Introductory_Survey_Second_Edition.pdf (Accessed: 8 October 2021).

Hernández, B. and Martín, M. (2016) ‘Optimal Process Operation for Biogas Reforming to Methanol: Effects of Dry Reforming and Biogas Composition’, *Industrial and Engineering Chemistry Research*, 55(23), pp. 6677–6685. doi: 10.1021/acs.iecr.6b01044.

Jakobsen, H. (1992) *Chemical Reactor Modelling, Concise Encyclopedia of Modelling & Simulation*. doi: 10.1016/b978-0-08-036201-4.50024-2.

Kisala, T. P. *et al.* (1987) ‘Sequential modular and simultaneous modular strategies for process flowsheet optimization’, *Computers & Chemical Engineering*, 11(6), pp. 567–579. doi: 10.1016/0098-1354(87)87003-5.

Kordabadi, H. and Jahanmiri, A. (2005) ‘Optimization of methanol synthesis reactor using genetic algorithms’, *Chemical Engineering Journal*, 108(3), pp. 249–255. doi: 10.1016/j.cej.2005.02.023.

Kordabadi, H. and Jahanmiri, A. (2007) ‘A pseudo-dynamic optimization of a dual-stage methanol synthesis reactor in the face of catalyst deactivation’, *Chemical Engineering and Processing: Process Intensification*, 46(12), pp. 1299–1309. doi: 10.1016/j.cep.2006.10.015.

Lim, H. W. *et al.* (2010) ‘Optimization of methanol synthesis reaction on Cu/ZnO/Al₂O₃/ZrO₂ catalyst using genetic algorithm: Maximization of the synergetic effect by the optimal CO₂ fraction’, *Korean Journal of Chemical Engineering*, 27(6), pp. 1760–1767. doi: 10.1007/s11814-010-0311-7.

Luyben, W. L. (2010) ‘Design and Control of a Methanol Reactor/Column Process’. doi: 10.1021/ie100323d.

Manenti, F., Cieri, S. and Restelli, M. (2011) ‘Considerations on the steady-state modeling of methanol synthesis fixed-bed reactor’, *Chemical Engineering Science*, 66(2), pp. 152–162. doi: 10.1016/J.CES.2010.09.036.

Nain, A. (2021) *TF_JAX_Tutorials - Part 7 (JIT in JAX)* | Kaggle. Available at: <https://www.kaggle.com/aakashnain/tf-jax-tutorials-part-7-jit-in-jax> (Accessed: 18 November 2021).

Nogueira, P. E., Matias, R. and Vicente, E. (2014) ‘An experimental study on execution time variation in computer experiments’, *Proceedings of the ACM Symposium on Applied Computing*, (March), pp. 1529–1534. doi: 10.1145/2554850.2555022.

Poling, B. E. *et al.* (2001) *The Properties of Gases and Liquids*. Fith. McGraw Hill. doi: 10.1036/0070116822.

Saad, S., Ali, A. and Ertogral, K. (2011) ‘Optimization Techniques in Chemical Processes’, *Asian Journal of Engineering, Sciences & Technology*, 1(2).

Santos, R. O. dos, Santos, L. de S. and Prata, D. M. (2018) ‘Simulation and optimization of a methanol synthesis process from different biogas sources’, *Journal of Cleaner Production*, 186, pp. 821–830. doi: 10.1016/j.jclepro.2018.03.108.

Schoenholz, S. S. and Cubuk, E. D. (2020) ‘JAX, M.D. A framework for differentiable physics’, *Advances in Neural Information Processing Systems*, (33).

Smith, J. M. (1950) *Introduction to chemical engineering thermodynamics*, *Journal of Chemical Education*. doi: 10.1021/ed027p584.3.

Taherdangkoo, M. *et al.* (2013) ‘An efficient algorithm for function optimization: Modified stem cells algorithm’, *Central European Journal of Engineering*, 3(1), pp. 36–50. doi: 10.2478/S13531-012-0047-8/MACHINEREADABLECITATION/RIS.

Turton, R. *et al.* (2003) *Analysis, Synthesis, and Design of Chemical Processes*. Fith. Prentice Hall.

Verma, A. (2000) ‘An introduction to automatic differentiation’, *Current Science*, 78(7), pp. 804–807.

Vikhar, P. A. (2017) ‘Evolutionary algorithms: A critical review and its future prospects’, *Proceedings - International Conference on Global Trends in Signal Processing, Information Computing and Communication, ICGTSPICC 2016*, pp. 261–265. doi: 10.1109/ICGTSPICC.2016.7955308.

Wächter, A. and Biegler, L. T. (2005) ‘On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming’, *Mathematical Programming* 2005 106:1, 106(1), pp. 25–57. doi: 10.1007/S10107-004-0559-Y.

Wächter, A. and Biegler, L. T. (2006) ‘Line search filter methods for nonlinear programming: Motivation and global convergence’, *SIAM Journal on Optimization*, 16(1), pp. 1–31. doi: 10.1137/S1052623403426556.

White, L. and Lubin, M. (2021) *JuliaDiff*. Available at: <https://juliadiff.org/> (Accessed: 18 November 2021).

7. Appendices

The following Appendices (A and B) describe the reactor and separator models. For simplification, it was assumed that the output temperature of the heat exchangers and the output pressure of the compressor were at their desired values, instead of modelling them (Abrol and Hilton, 2012). The feed to the process was assumed to be constant except for the flowrates of CO and CO₂. The feed was based off industrial data found in literature for a typical feed composition (Chen *et al.*, 2011). The total carbon flowrate to the process was kept constant but the ratio of CO and CO₂ was varied.

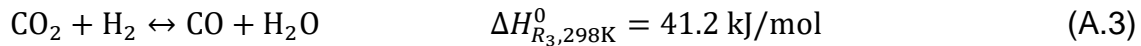
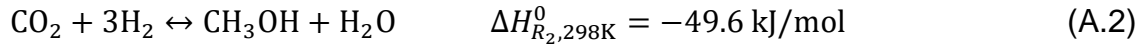
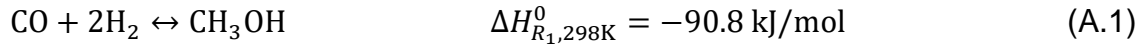
Table 10. Syngas feed composition

	Flowrate (mol/s)
CO and CO ₂	255.8
CH ₃ OH	6.560
H ₂	1321
H ₂ O	1.678
CH ₄	75.03
N ₂	80.04

7.1. Appendix A - Reactor Modelling

The reactor converts carbon monoxide and carbon dioxide to methanol in the presence of a catalyst. The following are the important reactions that take place.

Reactions



The reactor utility comprises of circulating boiling water on its shell side. The purpose of the boiling water is to remove the exothermic heat of the reactions. The following equations describe the mass and energy balances across the reactor assuming steady state. These equations were found in literature (Santos, Santos, and Prata 2018; Abrol and Hilton 2012), however, were corrected and checked by deriving them from first principles (Green and Southard, 1997).

Reactor mass and energy balances

$$\frac{dF_i}{dz} = N_t A_{int} \rho_{cat} (1 - \varepsilon_{cat}) \sum_{i=1}^N r_i v_i \quad (\text{A.4})$$

$$\frac{\partial T}{\partial z} = \left(\rho_{cat} a \sum_{i=1}^n r_i (-\Delta H_{f,i}) + \frac{4}{D_{int}} U (T_{shell} - T) \right) \frac{(N_t A_{int} (1 - \varepsilon_{cat}))}{\dot{f}_t C_p} \quad (\text{A.5})$$

Table 11 consists of the constant parameters needed for the equations above. These were obtained from Santos, Santos, and Prata (2018):

Table 11. Constant parameters for reactor mass and energy balance equations

Parameter	Symbol	Value	Units
Cross sectional area of reactor tube	A_{int}	0.00125	m ²
Void fraction of catalyst	ε_{cat}	0.285	
Density of catalyst	ρ_{cat}	1 190	kg/m ³
Activity of catalyst	a	1	
Internal diameter of reactor tube	D_{int}	0.04	m
Overall heat transfer co-efficient	U	118.44	W/m ² K
Temperature of boiling water on shell side	T_{shell}	490	K
Reactor length	z	7	m
Number of tubes	N_t	1620	
Number of components	N	7	
Number of reactions	n	3	

Kinetics

The kinetics of the reactor were described by the following rate expressions for the two rate-determining steps. The kinetic modelling described by Bussche and Froment (1996) was recommended for use (Santos, Santos, and Prata 2018; Abrol and Hilton 2012).

$$r_{MeOH} = \frac{k_{MeOH} p_{CO_2} p_{H_2} \left[1 - \left(\frac{1}{K_{eq}^1} \right) (p_{H_2O} p_{CH_3OH} / p_{H_2}^3 p_{CO_2}) \right]}{(1 + K_1 (p_{H_2O} / p_{H_2}) + K_2 \sqrt{p_{H_2}} + K_3 p_{H_2O})^3} \quad (A.6)$$

$$r_{RWGS} = \frac{k_{RWGS} p_{CO_2} \left[1 - \left(\frac{1}{K_{eq}^2} \right) (p_{H_2O} p_{CO} / p_{CO_2} p_{H_2}) \right]}{(1 + K_1 (p_{H_2O} / p_{H_2}) + K_2 \sqrt{p_{H_2}} + K_3 p_{H_2O})} \quad (A.7)$$

The value of the rate constants and equilibrium constants were determined using equations A.8 and A.9., with the associated constants in Tables 12 and 13 below (Bussche and Froment 1996).

$$k = A \exp \left(\frac{B}{RT} \right) \quad (A.8)$$

$$K_{eq} = 10^{(A/T - B)} \quad (A.9)$$

Table 12. Constants for kinetic model

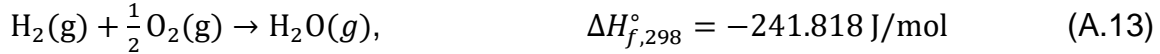
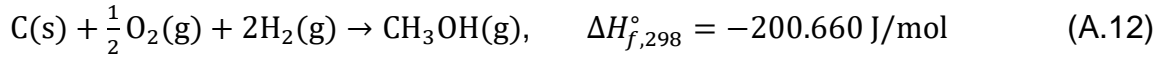
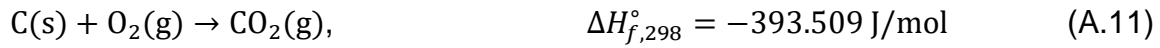
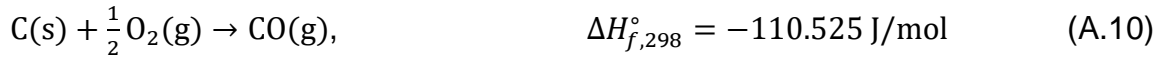
	A	B
k_{MeOH}	1.07	36 696
k_{RWGS}	1.22×10^{10}	-94 765
K_1	3453.38	-
K_2	0.499	17 197
K_3	6.62×10^{-11}	124 119

Table 13. Equilibrium constants for reaction kinetics

	A	B
K_{eq}^1	3 066	10.592
K_{eq}^2	-2 073	-2.029

Heats of formation

The following reactions were used to calculate the heats of formation for the following compounds: CO, CO₂, CH₃OH, H₂O, at varying temperatures (Abrol and Hilton 2012).



First, the standard heat capacities as a function of temperature were set up for compounds CO, CO₂, CH₃OH, H₂O, using the constants obtained from (Smith et al. 2018) in Table 14 below:

$$\langle \Delta C_p^\circ \rangle_H = R \times \left[\Delta A + \frac{\Delta B}{2} T_0 \left(\frac{T}{T_0} + 1 \right) + \frac{\Delta C}{3} T_0^2 \left(\left(\frac{T}{T_0} \right)^2 + \left(\frac{T}{T_0} \right) + 1 \right) + \frac{\Delta D}{TT_0} \right] \quad (\text{A.14})$$

Table 14. Constants for the heat capacity and heat of formation calculations

Ideal Gases							
	Tmax (K)	Cp ^{ig} ₂₉₈ /R	A	10 ³ B	10 ⁶ C	10 ⁻⁵ D	ΔH _f [°] ₂₉₈ (J/mol)
CO	2500	3.507	3.376	0.557		-0.031	-110.525
CO ₂	2000	4.467	5.457	1.045		-1.157	-393.509
CH ₃ OH	1500	5.547	2.211	12.216	-3.450		-200.66
H ₂	3000	3.468	3.249	0.422		0.083	
H ₂ O	2000	4.038	3.47	1.45		0.121	-241.818
CH ₄	1500	4.217	1.702	9.081	-2.164		-74.52
N ₂	2000	3.502	3.28	0.593		0.04	
O ₂	2000	3.535	3.639	0.506		-0.227	
Liquids							
		Cp ^{ig} ₂₉₈ /R	A	10 ³ B	10 ⁶ C	10 ⁻⁵ D	ΔH _f [°] ₂₉₈
CH ₃ OH		9.798	13.431	-51.28	131.13		-238.66
H ₂ O		9.069	8.712	1.25	-0.18		-285.83

Where:

$$T_0 = 298 \text{ K, and}$$

$$\Delta A \equiv \sum_i \nu_i A_i \quad (\text{A.15})$$

The heats of formation for CO, CO₂, CH₃OH, H₂O, were then found using Equation A.16 below, for reactions A.10-A.13.

$$\Delta H_T^\circ = \Delta H_{298}^\circ + (\Delta C_p^\circ)(T - T_0) \quad (\text{A.16})$$

Specific heats at constant pressure

The specific heats for each compound were then set up using the following equations, using the relevant constants in Table 14 above:

$$\frac{C_{P,i}}{R} = A + BT + CT^2 + \frac{D}{T^2} \quad (\text{A.17})$$

$$C_p = \sum_{i=1}^N y_i C_{p,i} \quad (\text{A.18})$$

7.2. Appendix B – Separator Modelling

The reactor product is cooled to a temperature low enough to separate out the liquid methanol product in a high-pressure separator (flash). The second product from the flash is the vapour that gets compressed and recycled back.

The separator was modelled as a pressure-temperature (P-T) flash, with the temperature, pressure, and inlet stream composition known. The vapour-ratio, gas outlet, and liquid outlet were unknown and therefore calculated using the model. The following equations were used to describe the mass balance for each component in the P-T flash:

Mass balance for each component:

$$Fz_i = F_l x_i + F_v y_i \quad (\text{B.1})$$

Where: z_i = mole fraction of component i in the feed to the flash

y_i and x_i = product vapour and liquid fractions of component i from the flash

Components CO, H₂, and CO₂ show non-ideal behaviour due to their critical temperatures being much higher than the reaction temperatures of 200-400°C. CH₃OH and H₂O also express non-ideal behaviours at these temperatures due to their polar molecular structures.

Owing to the novel nature and limited time frame of this thesis, first, all components were modelled as ideal in the separator. Then as time permitted, the added complexities of non-idealities were modelled using the Peng-Robinson equation of state (EOS) (Biegler, Grossmann, and Westerberg 1997).

Modelling an ideal separator:

For an ideal gas, the K-values (i.e., the ideal vapor-liquid equilibrium) were described as the ratio of the saturated pressure for each component, to the total pressure. Antoine Equation was used to find the saturated vapour pressures for each component. The values of A, B, and C were found in Yaws (2009) – see Table 15 below.

$$\log_{10} P_i^{Sat} = A_i - \frac{B_i}{C_i + T} \quad (\text{B.2})$$

With P_i^{Sat} in mmHg and T in °C.

Table 15. Component Values for Antoine Equation

	A	B	C
CO	6.72527	295.228	268.243
CO ₂	7.58828	861.82	271.883
CH ₃ OH	7.9701	1521.23	234
H ₂	6.14858	80.948	277.532
H ₂ O	8.05573	1723.64	233.076
CH ₄	6.84566	435.621	271.361
N ₂	6.72531	285.573	270.087

The K-values were then found:

$$K_i = \frac{p_i^{Sat}}{P} \quad (B.3)$$

Which then allowed for the vapour fraction $\left(\frac{F_v}{F}\right)$ to be calculated using the Rachford-Rice Equation:

$$\sum_{i=1}^N \frac{z_i(K_i-1)}{1+\left(\frac{F_v}{F}\right)(K_i-1)} = 0 \quad (B.4)$$

And hence, for the overall mass balance to be solved, using the definition of the K values as the ratio of the vapour and liquid mole fractions at equilibrium.

$$Fz_i = F_l x_i + F_v y_i$$

$$\text{With: } K_i = \frac{y_i}{x_i}, \text{ and } \therefore y_i = K_i x_i \quad (B.5)$$

$$Fz_i = F_l x_i + F_v K_i x_i$$

$$\therefore x_i = \frac{Fz_i}{F_l + F_v K_i} \quad (B.6)$$

And:

$$\therefore y_i = K_i x_i$$

Modelling a non-ideal separator:

To account for simplicity and accuracy in incorporating the non-idealities, the fugacity coefficients were calculated using the Peng Robinson EOS (Santos, Santos, and Prata 2018). From the modified Raoult's law, the phase equilibrium was defined as follows:

$$f_i^V = f_i^L \quad (B.7)$$

$$f_i^V = \phi_i^V y_i P \quad (B.8)$$

$$f_i^L = \phi_i^L x_i P \quad (B.9)$$

$$\therefore \phi_i^V y_i P = \phi_i^L x_i P \quad (B.10)$$

Where: ϕ_i = fugacity coefficient of each component

The model for the Peng Robinson EOS is as follows (Ch et al. 1973):

$$P = \frac{RT}{V-b} - \frac{a(T)}{(V^2+2bV-b^2)} \quad (B.11)$$

The following cubic equation was used to solve for the compressibility factor, Z, for both the liquid and vapour phases. The largest root found for Z^V , and the smallest root for Z^L , were chosen. The values for Z^V and Z^L were then used in Equations B.28 and B.29 to find the fugacity coefficients:

$$Z^{L^3} - (1 - B^L)Z^{L^2} + (A^L - 3B^{L^2} - 2B^L)Z^L - (A^L B^L - B^{L^2} - B^{L^3}) = 0 \quad (B.12)$$

$$Z^{V^3} - (1 - B^V)Z^{V^2} + (A^V - 3B^{V^2} - 2B^V)Z^V - (A^V B^V - B^{V^2} - B^{V^3}) = 0 \quad (B.13)$$

With:

$$A^L = \frac{a^L P}{R^2 T^2} \quad (\text{B.14})$$

$$A^V = \frac{a^V P}{R^2 T^2} \quad (\text{B.15})$$

$$B^L = \frac{b^L P}{RT} \quad (\text{B.16})$$

$$B^V = \frac{b^V P}{RT} \quad (\text{B.17})$$

The pure component parameters for the PR-EOS are:

$$a_i = \left(0.457235 \frac{R^2 T_{c,i}^2}{P_{c,i}} \right) \alpha_i(T) \quad (\text{B.18})$$

$$\alpha_i(T) = \left[1 + \kappa_i \left(1 - \sqrt{\frac{T}{T_{c,i}}} \right) \right]^2 \quad (\text{B.19})$$

$$\kappa_i = 0.37464 + 1.5422 \cdot \omega_i - 0.26992 \cdot \omega_i^2 \quad (\text{B.20})$$

$$\omega_i = -1 - \log_{10} \left(\frac{p^{\text{vap}}(0.7 \cdot T_c)}{p_c} \right) \quad (\text{B.21})$$

$$b_i = 0.077796 \frac{RT_{c,i}}{P_{c,i}} \quad (\text{B.22})$$

The values of the component's critical temperatures (T_{ci}), pressures (P_{ci}) and the acentric factors (w_i) were found from Poling et al. (2001):

Table 16. Critical temperatures, pressures, and acentric factors for each component

	T_{ci} (K)	P_{ci} (bar)	w_i
CO	132.9	35	0.066
CO ₂	304.1	73.8	0.239
CH ₃ OH	512.6	80.9	0.556
H ₂	33.0	12.9	-0.216
H ₂ O	647.3	221.2	0.344
CH ₄	190.4	46	0.011
N ₂	126.2	33.9	0.039

However, empirical mixing rules were needed to relate pure-component parameters to mixture parameters. The following mixing rules were applied:

$$a^V = \sum_{i=1}^N \sum_{j=1}^N y_i y_j a_{ij} \quad (\text{B.23})$$

$$a^L = \sum_{i=1}^N \sum_{j=1}^N x_i x_j a_{ij} \quad (\text{B.24})$$

With:

$$a_{ij} = (1 - \delta_{ij}) a_i^{1/2} a_j^{1/2} \quad (\text{B.25})$$

The binary interaction parameters (in Table 17 below) for each component (δ_{ij}) were obtained from experimental data in (Green and Southard, 1997) and from COCO:

Table 17. Binary interaction parameters

Compound	CO	CO ₂	CH ₃ OH	H ₂	H ₂ O	CH ₄	N ₂
CO		0	0	0.0919	0	0.03	0.033
CO ₂	0		0.022	-0.1622	0.0063	0.0793	-0.0222
CH ₃ OH	0	0.022		0	-0.0778	0	-0.2141
H ₂	0.0919	-0.1622	0		0	0.0263	0.0711
H ₂ O	0	0.0063	-0.0778	0		0	0
CH ₄	0.03	0.0793	0	0.0263	0		0.0289
N ₂	0.033	-0.0222	-0.2141	0.0711	0	0.0289	

Lastly:

$$b^V = \sum_{i=1}^N y_i b_i \quad (\text{B.26})$$

$$b^L = \sum_{i=1}^N x_i b_i \quad (\text{B.27})$$

Therefore, the fugacity coefficients could be found using:

$$\ln \phi_i^L = \frac{b_i}{b^L} (Z^L - 1) - \ln (Z^L - B^L) - \frac{A^L}{2\sqrt{2}B^L} \left(\frac{2}{a^L} \sum_i x_i (a)_{ij} - \frac{b_i}{b^L} \right) \ln \frac{Z^L + (1+\sqrt{2})B^L}{Z^L + (1-\sqrt{2})B^L} \quad (\text{B.28})$$

$$\ln \phi_i^V = \frac{b_i}{b^V} (Z^V - 1) - \ln (Z^V - B^V) - \frac{A^V}{2\sqrt{2}B^V} \left(\frac{2}{a^V} \sum_i y_i (a)_{ij} - \frac{b_i}{b^V} \right) \ln \frac{Z^V + (1+\sqrt{2})B^V}{Z^V + (1-\sqrt{2})B^V} \quad (\text{B.29})$$

Which then allowed for the K values for each component, i.e., the ratio of the liquid and gas fugacity coefficients, to be found by modifying equation B.7. above.

$$K_i = \frac{y_i}{x_i} = \frac{\phi_i^L}{\phi_i^V} \quad (\text{B.30})$$

And hence, the vapour fraction $\left(\frac{F_v}{F}\right)$ to be calculated using the Rachford-Rice Equation. This allowed for the overall mass balance to be solved.

$$\sum_{i=1}^N \frac{z_i(K_i-1)}{1+\left(\frac{F_v}{F}\right)(K_i-1)} = 0 \quad (\text{B.31})$$

The following was the iteration process that was followed and set up in the code:

1. Guess initial values for K_i
2. Guess $\left(\frac{F_v}{F}\right)$
3. Calculate y_i and x_i using $Fz_i = F_l x_i + F_v y_i$, where $F_l = F - F_v$,

$$\therefore x_i = \frac{Fz_i}{F_l + F_v K_i}$$

$$\therefore y_i = K_i x_i$$
4. Calculate ϕ_i^V and ϕ_i^L using the above steps for Peng Robinson EOS.
5. Find $K_i = \frac{\phi_i^L}{\phi_i^V}$
6. Recalculate $\left(\frac{F_v}{F}\right)$ using $\sum_{i=1}^N \frac{z_i(K_i-1)}{1+\left(\frac{F_v}{F}\right)(K_i-1)} = 0$

7.3. Appendix C – Process Costing

The economics of the process were considered for the use as an objective function. The costs of the flowsheet incorporated both the capital costs (catalyst, reactor, reactor preheater, reactor product cooler, and compressor), and the energy costs (compressor work, steam used for reactor preheating, and cooling water for reactor product cooling). The total annualized cost (TAC) was calculated assuming a payback period of 3 years. The incomes considered for the flowsheet were the sale of methanol, the high-pressure steam generated in the reactor, and the heating value of the purge stream which can be burned to recover heat. A yearly basis was assumed for all costing. All costing formulas and figures were obtained from (Luyben 2010).

The objective function was written to maximise profit. Therefore, the function was written as the subtraction of the expenses (TAC and energy costs) from the income generated.

$$\text{Profit} = \text{income} - \text{expenses} \quad (\text{C.1})$$

$$\begin{aligned} \text{Profit} = & (\text{Methanol income} + \text{Purge income} + \text{HP steam generated from reactor}) - \\ & - (\text{Compressor work}) - (\text{HPS for reactor preheater}) - \\ & (\text{CW for reactor product cooler}) - \left(\frac{\text{Total Capital}}{3} \right) \end{aligned} \quad (\text{C.2})$$

Note: please see Figure 1 (under Heading 2.2 in the main report) for stream numbers.

Where:

$$\text{Methanol income} = \left(F_{\text{methanol},6} \frac{\text{mol}}{\text{s}} \right) \left(\frac{\$0.021}{\text{mol}} \right) \left(\frac{3600 \text{ s}}{\text{hour}} \right) \left(\frac{8760 \text{ hours}}{1 \text{ year}} \right) \quad (\text{C.3})$$

$$\text{Purge income} = \left(F_8 \frac{\text{mol}}{\text{s}} \right) \left(\frac{0.331 \text{ J}}{\text{mol}} \right) \left(\frac{\$6}{\text{GJ}} \right) \left(\frac{1 \text{ GJ}}{10^9 \text{ J}} \right) \left(\frac{3600 \text{ s}}{\text{hour}} \right) \left(\frac{8760 \text{ hours}}{1 \text{ year}} \right) \quad (\text{C.4})$$

$$\begin{aligned} \text{HP generated from reactor} = & \left(F_3 \frac{\text{mol}}{\text{s}} \times C_p \frac{\text{J}}{\text{mol.K}} \times (T_4 - \right. \\ & \left. T_3) \text{ K} \right) \left(\frac{\$6}{\text{GJ}} \right) \left(\frac{1 \text{ GJ}}{10^9 \text{ J}} \right) \left(\frac{3600 \text{ s}}{\text{hour}} \right) \left(\frac{8760 \text{ hours}}{1 \text{ year}} \right) \end{aligned} \quad (\text{C.5})$$

$$\begin{aligned} \text{Compressor Work} = & \left(\frac{F_9 \frac{\text{mol}}{\text{s}} \times R \times T_9 \left[\left(\frac{P_{10}}{P_9} \right)^{0.2908} - 1 \right]}{0.2908} \right) \left(\frac{\$16.8}{\text{GJ}} \times \frac{\text{GJ}}{277778 \text{ Wh}} \right) \left(\frac{8760 \text{ hours}}{1 \text{ year}} \right) \end{aligned} \quad (\text{C.6})$$

$$\begin{aligned} \text{HPS for reactor preheater} = & \left(F_2 \frac{\text{mol}}{\text{s}} \times C_p \frac{\text{J}}{\text{mol.K}} \times \left(T_3 - \left(\frac{F_1}{F_1 + F_{10}} \times T_1 + \frac{F_{10}}{F_1 + F_{10}} \times \right. \right. \right. \\ & \left. \left. T_{10} \right) \right) \right) \left(\frac{\$8.22}{\text{GJ}} \right) \left(\frac{1 \text{ GJ}}{10^9 \text{ J}} \right) \left(\frac{3600 \text{ s}}{\text{hour}} \right) \left(\frac{8760 \text{ hours}}{1 \text{ year}} \right) \end{aligned} \quad (\text{C.7})$$

$$\begin{aligned} \text{CW for reactor product cooler} = & \left(F_4 \frac{\text{mol}}{\text{s}} \times C_p \frac{\text{J}}{\text{mol.K}} \times (T_4 - \right. \\ & \left. T_5) \text{ K} \right) \left(\frac{\$0.44}{\text{GJ}} \right) \left(\frac{1 \text{ GJ}}{10^9 \text{ J}} \right) \left(\frac{3600 \text{ s}}{\text{hour}} \right) \left(\frac{8760 \text{ hours}}{1 \text{ year}} \right) \end{aligned} \quad (\text{C.8})$$

$$\textbf{Total Capital} = \text{Catalyst}^1 + \text{Reactor Preheater}^2 + \text{Reactor Product Cooler}^3 + \text{Compressor}^4 \quad (\text{C.9})$$

Where:

$$\textbf{Catalyst}^1 = \left(\frac{\$10}{kg} \times \rho_{cat} \frac{kg}{m^3} \times A_{int} m^2 \times \text{No. of tubes} \times \text{Length } m \times \varepsilon_{cat} \right) \quad (\text{C.10})$$

The heat exchangers before and after the reactor, as well as the reactor were costed using the capital cost formula for a heat exchanger – see the general equation C.11. below:

$$\textbf{Capital Cost HXR} = 7296(\text{Area})^{0.65} \quad (\text{C.11})$$

$$\textbf{Reactor Preheater}^2 = 7296 \left(\frac{F_2 \times Cp \times \left(\left(\frac{F_1}{F_1 + F_{10}} \times T_1 + \frac{F_{10}}{F_1 + F_{10}} \times T_{10} \right) - T_3 \right)}{UF \frac{\Delta T_i - \Delta T_j}{\ln \frac{\Delta T_i}{\Delta T_j}}} \right)^{0.65} \quad (\text{C.12})$$

Where HPS is supplied at 648 K:

$$\Delta T_i = \text{hot in} - \text{cold out} = 648 \text{ K} - T_3$$

$$\Delta T_j = \text{hot out} - \text{cold in} = 648 \text{ K} - T_2$$

$$U = 850 \frac{W}{m^2 K}$$

$$F = 1$$

$$\textbf{Reactor Product Cooler}^3 = 7296 \left(\frac{F_4 \times Cp \times (T_4 - T_5)}{UF \frac{\Delta T_i - \Delta T_j}{\ln \frac{\Delta T_i}{\Delta T_j}}} \right)^{0.65} \quad (\text{C.13})$$

Where cooling water is supplied at 298 K:

$$\Delta T_i = \text{hot in} - \text{cold out} = T_4 - 313 \text{ K}$$

$$\Delta T_j = \text{hot out} - \text{cold in} = T_5 - 298 \text{ K}$$

$$U = 850 \frac{W}{m^2 K}$$

$$F = 0.9$$

$$\textbf{Compressor}^4 = \frac{(1293)(517.3)(3.11) \left(\frac{hp}{745.7 W} \times \left(\frac{F_{10} \frac{mol}{s} \times R \times T_9 \left[\left(\frac{P_{10}}{P_9} \right)^{0.2908} - 1 \right] \right)}{0.2908} \right)^{0.82}}{280} \quad (\text{C.14})$$

Assumptions made:

- The temperature into the reactor preheater was taken as a weighted average of the temperature of the stream exiting the compressor and the temperature of the syngas.
- The cost of the flash was not included in the capital costs, as the relative change in the cost of the flash when the optimisation variables are altered, would be negligible compared to the effects seen for the change in cost of the compressor and heat exchangers, for example.
- Assumed 8 760 hours in a year.
- The specific heats were calculated using an average temperature in and out of the unit and assuming a constant molar composition as an estimate.
- The temperature out of separator (F₇) was assumed to be equal to the temperature in the recycle (F₉), purge (F₈), and outlet of the compressor (F₁₀) stream.
- Assumed the compressor was 100% efficient.
- It was assumed the temperature of the syngas feed was fed at 523 K taken from (Abrol and Hilton 2012).
- It was assumed that the heat transfer coefficient was constant at 850 W/m²K for each heat exchanger (Green and Southard, 1997).
- Cooling water was supplied at 298 K and HPS was supplied at 648 K.
- All heat exchangers were assumed to be counter current.

7.4. Appendix D – Mathematical Formulation

The following equations describe the optimisation problem. The constraints were the mass balances across each unit in the process, whilst the bounds were the feasible operating conditions.

Objective Function

$$1) \text{ minimise } F_{obj}(x) = -F_{6,MeOH} \quad (D.1)$$

$$2) \text{ maximise } F_{obj}(x) = Profit \quad (D.2)$$

Constraints

$$m_1 + m_{10} - m_2 = 0 \quad (D.3)$$

$$m_2 - m_3 = 0 \quad (D.4)$$

$$m_3 - m_4 = 0 \quad (D.5)$$

$$m_4 - m_5 = 0 \quad (D.6)$$

$$m_5 - m_6 - m_7 = 0 \quad (D.7)$$

$$m_7 - m_8 - m_9 = 0 \quad (D.8)$$

$$m_9 - m_{10} = 0 \quad (D.9)$$

Bounds

The bounds for the decision variables were the minimum and maximum values that they can be, whilst ensuring safe and realistic operation. Discussed below are the values of the bounds and an explanation for each.

The upper bound of the reactor inlet temperature was set to ensure no catalyst deactivation would occur. Catalyst deactivation occurs at temperatures above 583 K. The lower bound was determined using reactor heuristics and a minimum approach temperature (Turton *et al.*, 2003). The reactor is cooled by the boiling of water in its shell. The temperature of its shell, was therefore assumed to stay at a constant temperature of 490 K. To ensure that the minimum approach temperature was not violated, the difference between the reactor inlet temperature at the coolant must be 10 K. The minimum temperature the reactor inlet could therefore be, was 500 K.

$$500 \text{ K} \leq T_{reactor,in} \leq 583 \text{ K}$$

Similar reasoning was used for the bounds of the reactor temperatures. The maximum temperature it can reach must ensure that no catalyst deactivation occurs and that the minimum bound must not violate the minimum approach temperature.

$$500 \text{ K} \leq T_{reactor,out} \leq 583 \text{ K}$$

The bounds for the separator temperatures were made sure to be in the range that ensured flash safety (Turton *et al.*, 2003). They were also the typical operating conditions for a flash separator in this process. The minimum bound was set to ensure that the minimum approach temperature of the heat exchanger before the separator was maintained. The heat exchanger uses cooling water with a temperature in of 298 K. Therefore, using a minimum approach of 10 K, the coldest temperature out for the hot process stream was 308 K.

$$308\text{ K} \leq T_{\text{separator}} \leq 323\text{ K}$$

The maximum pressure of the separator was set to a value below the operating pressure of the reactor. Hernández and Martín (2016) say typical flash pressures are between 50-75 bar. Therefore, with a reactor pressure set at 80 bar from (Santos, Santos, and Prata 2018), the bounds for the separator were set as follows:

$$50\text{ bar} \leq P_{\text{separator}} \leq 75\text{ bar}$$

As discussed under section 3.4.2, the maximum recycle ratio was determined to be the highest value usable, where the problem did not become ill-conditioned.

$$0.0 \leq \text{recycle ratio} \leq 0.80$$

The bounds of the carbon ratio $\left(\frac{CO}{CO+CO_2}\right)$ were between 0 and 1. They were obtained by comparing typical syngas feeds seen in literature (Chen *et al.*, 2011; Abrol and Hilton, 2012; Santos, Santos and Prata, 2018) - see Table 18 below:

Table 18. Typical carbon ratios of the syngas feed seen in literature

	Abrol	Santos	Chen
CO / (CO + CO ₂)	0.899	0.3117	0.2107

$$0.0 \leq \text{carbon ratio} \leq 1.0$$

The bounds of the species flowrates were testing. The minimum flowrate was simple as it could be set at zero (the minimum of any flowrate). The upper bound however was more difficult. If the value was set too high, the problem became unstable. If the value was set to low, the optimisation routines failed to solve the balances across each unit. Additionally, a bound had to be determined for every species in every stream. The highest value the components flowrates would reach, is at the maximum recycle ratio - as then the flowrates exiting the process are less. The upper bound of the components' flowrates were therefore taken to be their flowrates when the recycle ratio was equal to its maximum (0.8) and the rest of variables were taken as that of the base case. These were then multiplied by a safety factor of 100%. The safety factor was to account for changes in the other parameters - such as reactor temperature.

$$0 \frac{\text{mol}}{\text{s}} \leq F_{i,j} \leq 200\% F_{i,j} \text{ when recycle ratio} = 0.8$$

where *i* is species and *j* is stream number

The reactor pressures for methanol synthesis process are typically between 50-100 bar (Kordabadi and Jahanmiri, 2005; Chen *et al.*, 2011; Abrol and Hilton, 2012; Hernández and Martín, 2016; Santos, Santos and Prata, 2018). However, the reactor pressure was not used as a decision variable in the optimisation, but rather, a constant pressure of 80 bar was used (Santos, Santos and Prata, 2018).

Table 19: Typical reactor pressures seen in literature

	Abrol	Santos	Chen	Kordabadi	Hernandez
Reactor Pressure (bar)	51.7	69.7	66.7	75.0	50.0

Initial guess

The initial guess is often called the “starting point”. When the optimisation routines iteratively improve, the initial guess is used to attempt to converge at the optimum. The initial guess that was supplied to all the optimisers, were taken as the values of the base case. The base case conditions and stream table are shown in Appendix E.

7.5. Appendix E – Base Case Conditions and Results

The following conditions were specified for the base case. The values were based on the base case of two literature sources and indicative of standard operating conditions (Chen *et al.*, 2011; Santos, Santos and Prata, 2018). If the base case condition was not given, then the middle of the range of the decision variable was used. The constants used where the same as that for the optimal case.

Table 20. Base case parameter conditions

Parameter		Value
Reactor inlet temperature	K	500
Reactor pressure	bar	80.0
Separator pressure	bar	60.0
Separator temperature	K	308
Recycle ratio		0.500
Carbon ratio		0.529

The stream table for the process operating at the above conditions is shown below. Please see Figure 1 in the main body for clarification on stream numbers. The methanol production was 125.8 mol/s and the cost function had a value of \$79 990 000/yr.

Table 21. Base case stream table

	Stream No.	1	2	3	4	5	6	7	8	9	10
Component Flowrates (mol/s)	CO	106.4	151.6	151.6	90.51	90.51	0.0191	90.50	45.25	45.25	45.25
	CO ₂	149.5	232.5	232.5	168.1	168.1	2.134	166.0	82.99	82.99	82.99
	CH ₃ OH	6.560	12.8	12.83	138.3	138.3	125.8	12.54	6.268	6.268	6.268
	H ₂	1321	2327	2327	2011	2011	0.3371	2011	1005	1005	1005
	H ₂ O	1.678	2.428	2.428	66.80	66.80	65.29	1.501	0.7506	0.7506	0.7506
	CH ₄	75.03	149.9	149.9	149.9	149.9	0.1577	149.7	74.87	74.87	74.87
	N ₂	80.04	160.0	160.0	160.0	160.0	0.0648	160.0	79.98	79.98	79.98

7.6. Appendix F – Reactor Model Validation Additional Graph

Below is the graph of the component flowrates along the reactor length, which was compared to literature (Chen *et al.*, 2011) for model validation. The graph in the main body of the report (Figure 6) excludes the component hydrogen for a better view of the shape of the rest of the components’ profiles – as it can be seen that H₂ has a much larger flowrate than the rest.

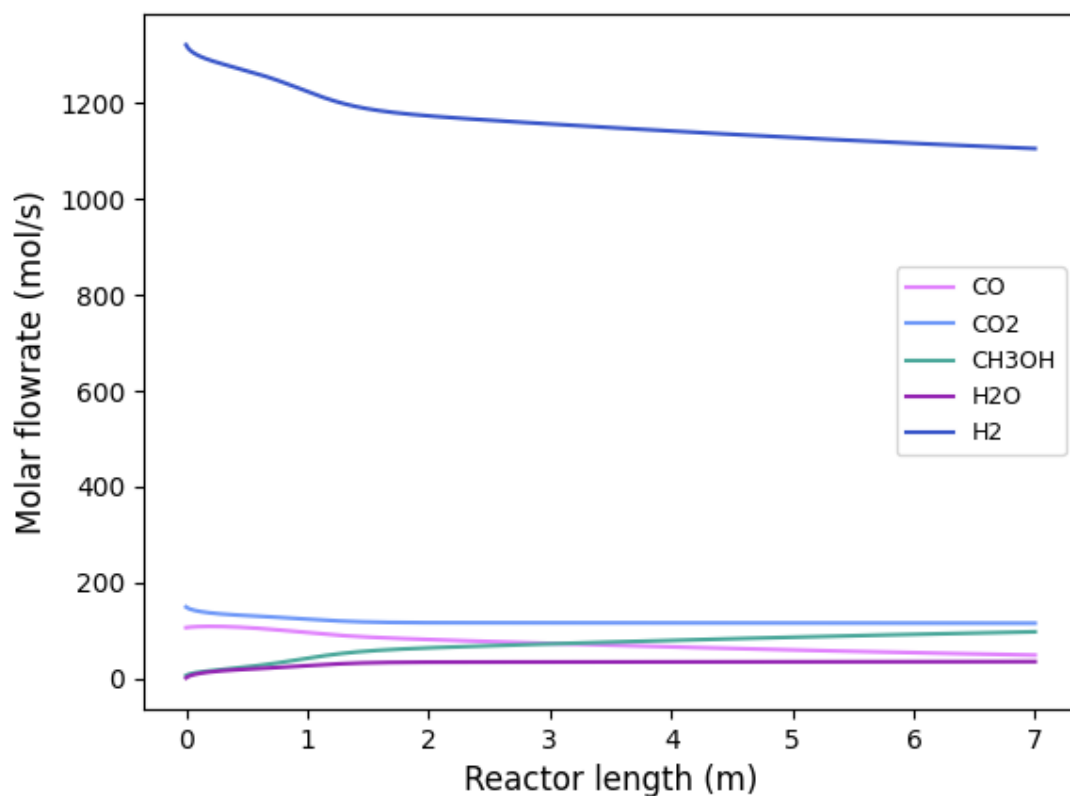


Figure 12. Component molar flowrates along the reactor length (including hydrogen)

7.7. Appendix G – Computational Efficiency Additional Graphs

Figure 13 is meant to aid in the illustration of how jit compilation works. Firstly, it shows how compilation leads to computational efficiency as the compiled code takes considerably less time to solve than the uncompiled code. Secondly, the figure shows how the code is compiled in the first run (see the high value of run 1), with all subsequent runs being executed quickly.

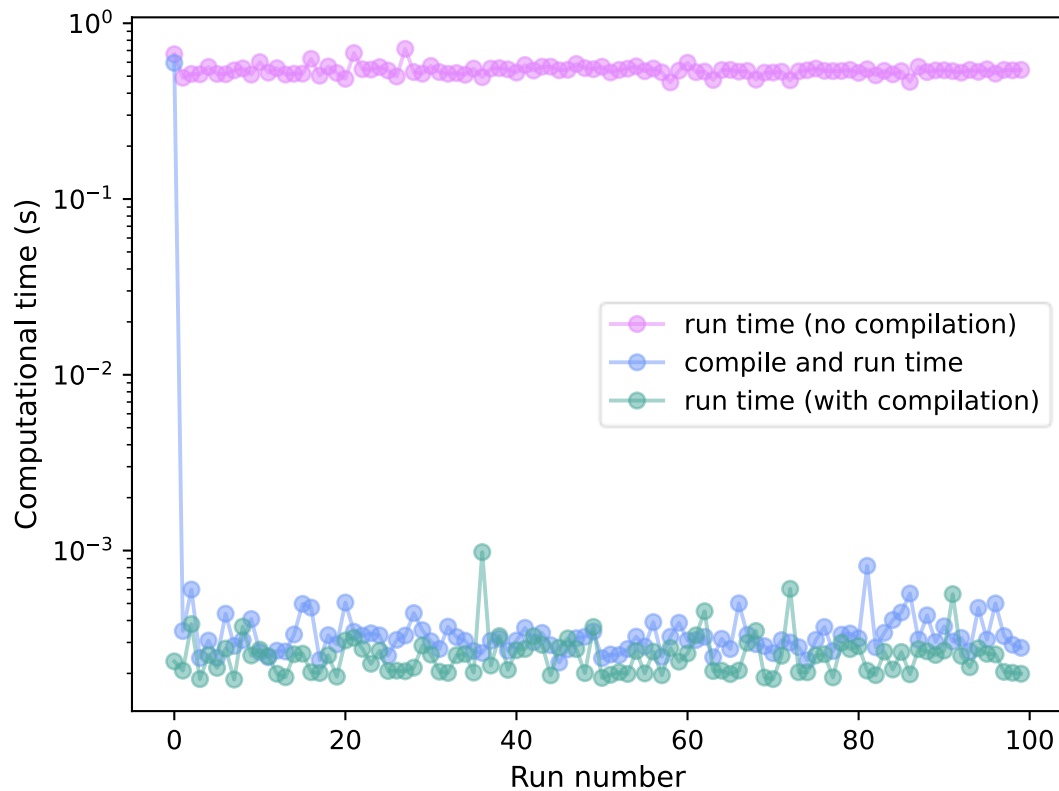


Figure 13. Computational time for each run of the separator model

7.8. Appendix H – Differential Evolution Flow Diagram

The steps of differential evolution are shown in the flow diagram below.

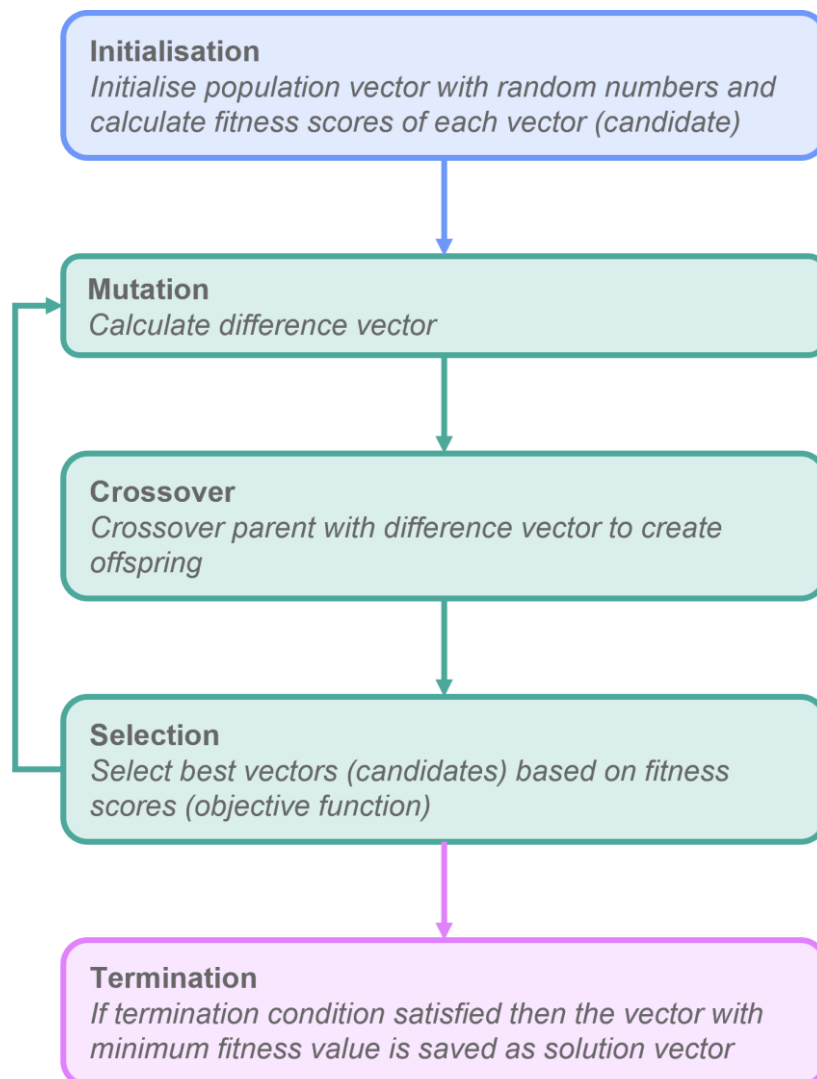


Figure 14. Flow diagram of DE algorithm

7.9. Appendix I – Signed Ethics Form

APPLICATION FORM


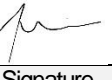

Please Note:

Any person planning to undertake research in the Faculty of Engineering and the Built Environment (EBE) at the University of Cape Town is required to complete this form before collecting or analysing data. The objective of submitting this application prior to embarking on research is to ensure that the highest ethical standards in research, conducted under the auspices of the EBE Faculty, are met. Please ensure that you have read, and understood the EBE Ethics in Research Handbook (available from the UCT EBE, Research Ethics website) prior to completing this application form: <http://www.ebe.uct.ac.za/ebe/research/ethics1>

APPLICANT'S DETAILS		
Name of principal researcher, student or external applicant		Gabriella Heurlin Anna Midgley
Department		Chemical Engineering
Preferred email address of applicant:		HRLGAB001@myuct.ac.za MDGANN002@myuct.ac.za
If Student	Your Degree: e.g., MSc, PhD, etc.	BSc Chemical Engineering
	Credit Value of Research: e.g., 60/120/180/360 etc.	36
	Name of Supervisor (if supervised):	Klaus Möller
If this is a research contract, indicate the source of funding/sponsorship		n.a.
Project Title		JAXed Up Optimisation of the Carbon Dioxide to Methanol Process

I hereby undertake to carry out my research in such a way that:

- there is no apparent legal objection to the nature or the method of research; and
- the research will not compromise staff or students or the other responsibilities of the University;
- the stated objective will be achieved, and the findings will have a high degree of validity;
- limitations and alternative interpretations will be considered;
- the findings could be subject to peer review and publicly available; and
- I will comply with the conventions of copyright and avoid any practice that would constitute plagiarism.

APPLICATION BY	Full name	Signature	Date
Principal Researcher/ Student/External applicant	Gabriella Heurlin Anna Midgley	 	11 Oct 2021
SUPPORTED BY	Full name	Signature	Date
Supervisor (where applicable)	Klaus Möller		11 Oct 2021

APPROVED BY	Full name	Signature	Date
HOD (or delegated nominee) Final authority for all applicants who have answered NO to all questions in Section 1; and for all Undergraduate research (Including Honours).	Click here to enter text.		Click here to enter a date.

<p>Chair: Faculty EIR Committee For applicants other than undergraduate students who have answered YES to any of the questions in Section 1.</p>	<p>Click here to enter text.</p>		<p>Click here to enter a date.</p>
--	--	--	--

Plagiarism Declaration



Department of Chemical Engineering

CHE4045Z

PLAGIARISM DECLARATION

- We know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is our own.
- We have used the Harvard referencing system for citation and referencing.
- In this report, all contributions to, and quotation from, the work(s) of other people have been cited and referenced.
- This report is our own work. We have not allowed and will not allow anyone to copy our work.

PeopleSoft No.	Sign
1649784	
1637209	