



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Systematic Staging in Chemical Reactor Design

Fischer-Tropsch

**Martin Skjærvø Foss**

Chemical Engineering and Biotechnology

Submission date: June 2013

Supervisor: Magne Hillestad, IKP

Co-supervisor: Paris Klimantos, IKP

Norwegian University of Science and Technology  
Department of Chemical Engineering



# Abstract

Today, crude oil is the main resource for production of liquid fuels. As the demand increases, utilization of alternative resources becomes more and more urgent. Thus, the development of new and continued research on established process technologies is important.

The scope of this Master's thesis has been the catalytic hydrogenation of CO for production of linear, long chained hydrocarbons, known as the Fischer-Tropsch process. The core of a chemical plant is the reactor, thus the ultimate goal is to design the optimal reactor path layout for the process, i.e. find number of reactor stages, cooling media temperature, heat transfer area density, molar feed ratio and the inclusion or omission of separation of products. A kinetic model presented by Todici et al. [1], implemented in MATLAB R2012b, has formed the basis for the calculations.

The simulations confirmed that the performance increases as the degrees of freedom was increased. The supply of additional  $H_2$ , prior to the reactor stages, yielded a major increase in production of key component,  $C_{11+}$ . The optimal design, disregarding any economical aspects, was a three-stage design with a space time,  $\sigma$ , equal to  $0.8\text{ m}^3\text{ s/kg}$ , a molar feed ratio,  $x_{H_2}/x_{CO}$ , equal to 1.5, and separation of water and products between the reactor stages. The design resulted in a production of the key component equal to 12.8wt% of the feed flow. The exclusion of the separation between the reactor stages yielded a decrease of 10.3% in the production.

The utilized method of reactor staging yields an optimal reactor design. The real-life feasibility of the design is still unknown as no economical aspects have been taken into consideration.



# Sammendrag

Råolje er i dag hovedkilden til flytende drivstoff og med verdens økende etterspørsel vil utnyttelse av alternative kilder være uunngåelig. Forskning på eksisterende, og utvikling av ny, prosess teknologi er av denne grunn viktig for å kunne utnytte verdens resurser på best mulig måte.

Reaktoren er kjernen i et prosessanlegg. Det ultimate målet er å designe en optimal prosess ved å bestemme antall reaktortrinn, kjølemediumstemperatur, varmeoverføringsarealtetthet, molart fødestrømsforhold og separasjon av produkter. Masteroppgaven beskriver katalytisk hydrogenering av CO for produksjon av lineære, langkjedede hydrokarboner, kjent som Fisher-Tropsch prosessen. En kinetikkmodell presentert av Todic et al. [1], implementert i MATLAB R2012b, har dannet grunnlaget for beregningene.

Simuleringene bekreftet økt ytelse ved økt antall frihetsgrader. Tilførsel av ekstra  $H_2$  til prosessen ved innløpet til hvert reaktortrinn førte til en markant økning i produksjonen av nøkkelkomponent,  $C_{11+}$ . Ved å utelate økonomiske aspekt ble det optimale designet funnet å bestå av tre reaktortrinn, en oppholdstid,  $\sigma$ , lik  $0.8 \text{ m}^3 \text{ s/kg}$ , et fødestrømsforhold likt 1.5, og separasjon av vann og produkter mellom hvert reaktortrinn. Designet resulterte i en produksjon av nøkkelkomponent lik 12.8 % av fødestrømmen. Ved å fjerne separasjonen mellom trinnene minket produksjonen av nøkkelkomponent med 10.3 %.

Prosessberegningemetoden benyttet i denne masteroppgaven gir et optimalt reaktordesign. Muligheten for å gjennomføre dette reaktordesignet i virkeligheten er ikke kjent da økonomiske beregninger ikke er gjennomført. norsk



# Preface

This Master's thesis is a result of the work within the subject "TKP4900 - Chemical Process Technology, Master Thesis" at the Norwegian University of Science and Technology (NTNU). The work has been completed within the group Environmental Engineering and Reactor Technology at the Department of Chemical Engineering, Faculty of Natural Sciences and Technology, during the spring 2013.

The completion of this Master's thesis would not have been possible without the help, guidance and support from several important people. First of all I would like to thank my supervisor, professor Magne Hillestad. His office door has always been open throughout the work with this thesis. Secondly, I would like to give thanks to my co-supervisor Paris Klimantos. His insight guided me in the right direction.

Thanks goes out to Peter J. B. Lindersen and Ivar M. Jevne for our inspiring discussions and their constructive critique. I will also give thanks to Andrew Fleming for taking time out of his busy schedule to check the English in this thesis.

Finally, I would like to give great thanks to my life partner Pia Odden, for her motivation, moral support and, last but not at least, for proof reading this report. I know it has been tedious work as it is far from her personal interests and academic discipline.

## Declaration of compliance

I hereby declare that this is an independent work in compliance with the exam regulations of the Norwegian University of Science and Technology.

Trondheim, 10th June, 2013

---

Martin S. Foss





# Table of Contents

<b>Abstract</b>	<b>i</b>
<b>Sammendrag</b>	<b>iii</b>
<b>Preface</b>	<b>v</b>
<b>Table of Contents</b>	<b>vii</b>
<b>Nomenclature</b>	<b>xi</b>
<b>List of Figures</b>	<b>xv</b>
<b>List of Tables</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>3</b>
2.1 The Fischer-Tropsch reaction . . . . .	5
2.2 Catalysts . . . . .	5
2.3 Reactor types . . . . .	7
2.3.1 Fixed bed reactors . . . . .	7
2.3.2 Fluidized and slurry bed reactors . . . . .	8
2.3.3 Heat transfer area density . . . . .	11
<b>3 Reactor Path</b>	<b>13</b>
3.1 Reactor volume . . . . .	13
<b>4 Reactor Staging</b>	<b>15</b>
4.1 Plug flow model . . . . .	15
4.2 Completely mixed flow model . . . . .	17
4.3 Composite model . . . . .	18
4.4 Optimization . . . . .	19
4.5 Cost functions . . . . .	20

4.6	Design functions . . . . .	20
4.6.1	Mixing . . . . .	21
4.6.2	Heat transfer area distribution . . . . .	21
4.6.3	Cooling media temperature profile . . . . .	22
4.6.4	Distribution of extra feed . . . . .	23
4.6.5	Catalyst activity . . . . .	23
4.6.6	Separation . . . . .	24
<b>5</b>	<b>Kinetic Model</b>	<b>25</b>
5.1	Lumping of components . . . . .	28
<b>6</b>	<b>Simulations and Raw Data Processing</b>	<b>29</b>
6.1	Input data . . . . .	31
6.2	Raw data processing . . . . .	31
<b>7</b>	<b>Results</b>	<b>35</b>
7.1	Case studies . . . . .	35
<b>8</b>	<b>Discussion</b>	<b>45</b>
<b>9</b>	<b>Conclusions</b>	<b>51</b>
	<b>References</b>	<b>53</b>
	<b>Appendices</b>	
<b>A</b>	<b>Results data</b>	<b>A-1</b>
<b>B</b>	<b>Kinetic Model – Additional Data</b>	<b>B-1</b>
<b>C</b>	<b>Matlab Code</b>	<b>C-1</b>
C.1	fischerTropschTodicLump.m . . . . .	C-4
C.2	fluidpath.m . . . . .	C-8
C.3	integrate.m . . . . .	C-17
C.4	OptimizeDesign.m . . . . .	C-19
C.5	DesignModel.m . . . . .	C-23

## TABLE OF CONTENTS

---

C.6	kinTodicLump.m . . . . .	C-25
C.7	ObjectiveFunction.m . . . . .	C-29
C.8	ModelConstraints.m . . . . .	C-30
C.9	flowPropTPlump.m . . . . .	C-32
C.10	moletomass.m . . . . .	C-33
C.11	masstomole.m . . . . .	C-34
C.12	pathset.m . . . . .	C-34
C.13	AssignVector.m . . . . .	C-35
C.14	AssignObject.m . . . . .	C-36
C.15	Jacobian.m . . . . .	C-38
C.16	intMolarMass.m . . . . .	C-39
C.17	deltaHrx.m . . . . .	C-39
C.18	mid.m . . . . .	C-41
C.19	AssignRate.m . . . . .	C-41
C.20	pathget.m . . . . .	C-41
C.21	MmCpMatrixLump.m . . . . .	C-42
C.22	uniSimPropertiesTodicLump.m . . . . .	C-42
C.23	FTplotting.m . . . . .	C-44
C.24	plotFT.m . . . . .	C-45
C.25	defaultPlotSettings.m . . . . .	C-48
<b>D</b>	<b>Health, Security and Environment</b>	<b>D-1</b>



# Nomenclature

## Latin letters

Symbol	Explanation	Unit
$a$	Heat transfer area density	$\text{m}^2/\text{m}^3$
$A$	Heat transfer area	$\text{m}^2$
$A_i$	Pre-exponential factor	-
$C_p$	Heat capacity	$\text{kJ}/(\text{kg K})$
$C_{p,F}$	Heat capacity	$\text{kJ}/(\text{kg K})$
$E_{a,i}$	Activation energy	$\text{kJ}/\text{mol}$
$\Delta E$	Desorption energy	$\text{kJ}/\text{mol}$
$\Delta H_i$	Heat of reaction	$\text{kJ}/\text{mol}$
$\Delta_r H_j$	Heat of reaction	$\text{kJ}/\text{mol}$
$J$	Cost function	-
$k_i$	Kinetic parameter	-
$K_i$	Kinetic parameter	-
$M$	Molecular weight	$\text{kg}/\text{kmol}$
$n_c$	Number of components	-
$n_{\text{col}}$	Number of internal collocation point	-
$n_s$	Number of reactor stages	-
$P_i$	Partial pressure, $i = \text{CO}, \text{H}_2, \text{H}_2\text{O}$	$\text{MPa}$
$P_{\text{tot}}$	Total pressure	$\text{bar}$
$\mathcal{P}_{C_{11+}}$	Production of key component	-
$\tilde{r}_j$	Reaction rate	$\text{kmol}/(\text{m}^3 \text{s})$
$R$	Universal gas constant	$\text{kJ}/(\text{mol K})$
$\tilde{R}_i$	Component reaction rate	$\text{kg}/(\text{m}^3 \text{s})$
[S]	Fraction of vacant catalyst sites	-
$T$	Temperature	$\text{K}$ or $^\circ\text{C}$
$T_F$	Feed temperature	$\text{K}$ or $^\circ\text{C}$
$T_{\text{max}}$	Max temperature	$^\circ\text{C}$

Continued on next page...

Latin letters continued

Symbol	Explanation	Unit
$T_{\text{ref}}$	Reference temperature	K
$T_W$	Cooling media temperature	K or °C
$u_A$	Design function, catalyst activity	-
$u_F$	Design function, distribution of extra feed	-
$u_H$	Design function, heat transfer area distribution	-
$u_M$	Design function, mixing	-
$u_T$	Design function, cooling media temperature profile	-
$u_S$	Design function, separation	-
$U$	Overall heat transfer coefficient	$\text{kJ}/(\text{m}^2 \text{K})$
$\mathcal{U}$	Solution space	-
$V$	Volume	$\text{m}^3$
$V_R$	Total reactor volume	$\text{m}^3$
$W$	Feed flow	$\text{kg}/\text{s}$
$W_0$	Feed flow	$\text{kg}/\text{s}$
$x_i$	Mole fraction, $i = \text{H}_2, \text{H}_2\text{O}$	-
<b>E</b>	Diagonal matrix	-
<b>E<sub>0</sub></b>	Diagonal matrix	-
<b>f</b>	Model, ODE	-
<b>h</b>	Nonlinear inequality constraints	-
<b>I</b>	Unity matrix	-
<b><math>\tilde{\mathbf{J}}</math></b>	Partial derivative	-
<b>K</b>	Diagonal matrix	-
<b>K<sub>0</sub></b>	Diagonal matrix	-
<b><math>\tilde{\mathbf{R}}</math></b>	Component reaction rate vector	$\text{kg}/(\text{m}^3 \text{s})$
<b><math>\tilde{\mathbf{R}}_0</math></b>	Component reaction rate vector	$\text{kg}/(\text{m}^3 \text{s})$
<b>u</b>	Manipulated variables	-
<b>x</b>	State vector	-
<b>x<sub>0</sub></b>	State vector, initial guess	-
<b>xS</b>	Separation function	-
<b>z</b>	State vector	-

Continued on next page...

**Latin letters continued**

<b>Symbol</b>	<b>Explanation</b>	<b>Unit</b>
$\mathbf{z}_0$	State vector, initial guess	-

**Greek letters**

<b>Symbol</b>	<b>Explanation</b>	<b>Unit</b>
$\alpha$	Feed distribution	kg/(m <sup>3</sup> s)
$\alpha_i$	Chain growth probability	-
$\beta$	Dimensionless heat transfer coefficient	-
$\gamma$	Dimensionless total mass flow	-
$\gamma^{\text{end}}$	Dimensionless total mass flow at outlet	-
$\omega_{\text{C}_{11+}}$	Weight fraction key component	-
$\omega_{\text{C}_{11+}}^{\text{end}}$	Weight fraction key component at outlet	-
$\omega_{F,i}$	Weight fraction extra feed	-
$\omega_i$	Weight fraction	-
$\omega_{i,0}$	Weight fraction feed	-
$\omega_{\text{key}}$	Weight fraction key product	-
$\omega_{\text{CO}}^0$	Weight fraction CO at inlet	-
$\omega_{\text{CO}}^{\text{end}}$	Weight fraction CO at outlet	-
$\rho_{\text{cat}}$	Bulk density catalyst	kg/m <sup>3</sup>
$\rho_{\text{products}}$	Density products	kg/L
$\sigma$	Space time	m <sup>3</sup> s/kg
$\theta$	Dimensionless temperature	-
$\theta_F$	Dimensionless temperature additional feed	-
$\theta_W$	Dimensionless temperature cooling media	-
$\xi$	Dimensionless volume	-
$\Delta\xi$	Dimensionless reactor stage volume	-





# List of Figures

2.1	The main steps in a GTL plant. . . . .	3
2.2	Multi-tubular fixed bed FT reactor. . . . .	8
2.3	Fluidized bed reactors. . . . .	9
3.1	Flowsheet of the reactor path. . . . .	13
4.1	Different mixing design function scenarios. . . . .	21
7.1	One reactor stage. Mixing is optimized. $\sigma = 1.46\text{m}^3\text{s/kg}$ . . . . .	35
7.2	One reactor stage. Mixing, heat transfer area and cooling media temperature are optimized. $\sigma = 1.46\text{m}^3\text{s/kg}$ . . . . .	36
7.3	Three reactor stages. Mixing, heat transfer area and cooling media temperature are optimized. $\sigma = 1.46\text{m}^3\text{s/kg}$ . . . . .	37
7.4	One reactor stage. Mixing, heat transfer area, cooling media temperature and additional feed are optimized. $\sigma = 1.46\text{m}^3\text{s/kg}$ . . . . .	40
7.5	One reactor stage. Mixing, heat transfer area, cooling media temperature and additional feed are optimized. $\sigma = 1\text{m}^3\text{s/kg}$ . . . . .	40
7.6	One reactor stage. Mixing, heat transfer area, cooling media temperature and additional feed are optimized. $\sigma = 0.8\text{m}^3\text{s/kg}$ . . . . .	41
7.7	Three reactor stages. Mixing, heat transfer area, cooling media temperature and additional feed are optimized. $\sigma = 1.46\text{m}^3\text{s/kg}$ . . . . .	41
7.8	Three reactor stages. Mixing, heat transfer area, cooling media temperature and additional feed are optimized. $\sigma = 1\text{m}^3\text{s/kg}$ . . . . .	42
7.9	Three reactor stages. Mixing, heat transfer area, cooling media temperature and additional feed are optimized. $\sigma = 0.8\text{m}^3\text{s/kg}$ . . . . .	42
7.10	Three reactor stages. Mixing, heat transfer area, cooling media temperature and additional feed are optimized. $\sigma = 0.8\text{m}^3\text{s/kg}$ . . . . .	43



# List of Tables

2.1	World reserves of “carbon” relative to oil. . . . .	4
2.2	Approximate and relative price of catalysts, compared to iron. . . . .	6
3.1	Space time vs. reactor volume. . . . .	14
6.1	Simulation overview. . . . .	30
6.2	Physical properties of the component lumps. . . . .	31
7.1	Main results, extracts. . . . .	39
A.1	Main results for all simulations. . . . .	A-2
A.2	Raw data for the design functions. . . . .	A-3
B.1	Values for the parameters in the kinetic model. . . . .	B-1
C.1	Description of the different MATLAB scripts. . . . .	C-1
C.2	Simulation hierarchy for retrieving physical properties. . . . .	C-2
C.3	Simulation hierarchy for plotting the results. . . . .	C-2
C.4	Main simulation hierarchy. . . . .	C-3



# 1 Introduction

The world as we know it today is dependent on fossil fuels. In total, fossil fuels claim a market share of 87% [2] of the total world energy consumption. As the world's energy demand continues to increase and the environmental impacts become more severe, the development of new, and continued research on known, process technologies is vital. Input factors such as raw material and energy should be efficiently utilized to produce valuable products, with the use of minimal equipment volumes, areas and energies. The development of process technologies with enhanced material and energy efficiencies will improve the utilization of the raw material at hand.

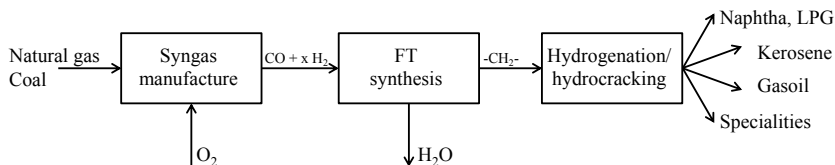
The core of a chemical plant is the reactor. A method for systematic staging of chemical reactors has been applied to the Fischer-Tropsch (FT) process based on a kinetic model published by Todic et al. [1]. Reactants pass through a series of functions or basic operations to form the desired products. The basic operations are represented by design functions on the reactor path. The design functions are fluid mixing (dispersion), distribution of heat transfer area, cooling media temperature, distribution of additional feed points, catalyst dilution distribution and separation of products. The conceptual reactor design problem is solved as an optimal control problem. The realization is a staged process string of multi-functional units where the criterion has been to maximize the production of the desired product in a given volume.

The model has been implemented in MATLAB R2012b. Physical data has been extracted from UniSim Design R400. The model is used to find the optimal mixing configuration, heat transfer area, cooling media temperature distribution and feed distribution which yields a reactor path that produces as heavy hydrocarbons as possible.



## 2 Background – The Fischer-Tropsch Process

The gas-to-liquids (GTL) process leads to the production of liquid hydrocarbons from natural gas via the production of synthesis gas (syngas). The two main production steps in a GTL plant are the syngas production and the FT synthesis, see Figure 2.1. Syngas consists primarily of CO and H<sub>2</sub>. The catalytic hydrogenation of CO which yields H<sub>2</sub>O and a large range of liquid, long chained, linear hydrocarbons is known as the FT process. The long chained products consist primarily of *n*-alkanes (paraffins), 1-alkenes (olefins) and oxygenated hydrocarbons [3], while secondary products include branched hydrocarbons, alcohols, aldehydes and carboxylic acids [4]. The products are used to produce high-quality diesel fuels and gasoline. The process is considered a good option for production of clean transportation fuels and chemicals [5]. It is also of great importance as it makes the production of industrial organic chemicals from simple, inorganic molecules viable.



**Figure 2.1:** The main steps in a GTL plant.

The FT process dates back to the beginning of the 20<sup>th</sup> century [3]. The first experiments with syngas were performed by Sabatire and Senderens in 1902 [3,4,6], who found that the reduction of CO to CH<sub>4</sub> was catalyzed by Ni. The process is named after the two German scientists Fischer and Tropsch, who in 1923 published their work with syngas and alkalized iron catalysts. Their process synthesis was patented in 1925 [6]. The first plant, utilizing the FT process, was constructed by Ruhrchemie in 1933 [3,4]. During the Second World War FT plants were operated in Germany<sup>1</sup> as well as in Japan, Manchuria

<sup>1</sup>20% of the German gasoline demand was produced by coal-to-liquid (CTL) plants [6].

and France. However, after the war, these plants were shut down due to economical reasons.

Today, the main resource for production of liquid fuels and other bulk chemicals is crude oil [3]. As the world's consumption continues to increase it is important to find and utilize other resources. The interest in the FT process has been influenced by scenarios of the supply and demand of the fossil energy throughout the last decade [5, 7]. In 50 to 100 years the world's oil reserves may be depleted [2] and the need for other resources to produce liquid fuels and chemicals will be evident. Natural gas reserves are better distributed around the world compared to oil reserves. Thus, the availability of the former will be less sensitive to disturbances in production rates caused by social, economical and political instability seen around the world today.

In Table 2.1 the world's carbon reserves, relative to oil are presented. Of the four oil substitutes, coal is the fastest growing fossil fuel [2]. Coal has the largest reserves and may be the cheapest to mine, however from an environmental point of view it is not a preferred raw material. Recovery of oil from tar sand and shale oil are both highly energy demanding, in addition these oil types are highly aromatic and thus not suitable for the production of linear hydrocarbons [3]. Biomass is an alternative feedstock which can be utilized to produce syngas as well. Provided that the production does not come into conflict with food production<sup>2</sup>, this is a viable way to produce "green" hydrocarbons in the future.

**Table 2.1:** World reserves of "carbon" relative to oil [3].

Source	Reserves, oil equivalent
Crude oil	1.0
Tar sands	0.7
Shale oil	1.2
Natural gas	1.5
Coal	26

Production of fuels utilizing natural gas and FT synthesis is in direct competition with crude oil processes, i.e. the crude oil price will have a strong effect on the viability of a

<sup>2</sup>First generation bio-diesel has been held responsible for replacing cultivable areas.

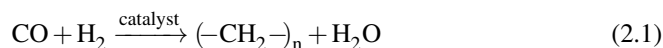


FT plant [3, 7]. Natural gas has to be available at a low cost, such as stranded gas<sup>3</sup> and associated gas<sup>4</sup> which can be used to produce large amounts of “clean” fuel. Gas from remote locations is uneconomical to transport to the market as is. However, by means of FT synthesis it can be made viable [5].

For the overall process, from raw material to products, syngas production is of great importance. The production of syngas typically accounts for 60 to 70% of the capital and running costs of the total plant [7]. However, here the scope is set only to investigate the FT reactor.

## 2.1 The Fischer-Tropsch reaction

The FT reaction, see Equation (2.1), is highly exothermic. Formation of one mole of  $-CH_2-$  releases approximately 160kJ/mol [3, 8]. The large production of heat requires adequate heat transfer from the catalyst particles.



The FT synthesis can be divided into two major technologies, i.e. high temperature Fischer-Tropsch (HTFT) and low temperature Fischer-Tropsch (LTFT). The operation temperatures are approximately 300 to 350°C and 200 to 260°C, respectively [6, 7]. Today, the major focus of attention is directed towards the LTFT processes. LTFT processes primarily yields ultra-clean linear, long chained hydrocarbons (wax), whilst HTFT processes yield lighter, more aromatic products [6, 9].

## 2.2 Catalysts

There are several different catalysts available to the FT process. Catalysts based on Ni, Co, Fe and Ru have the required activity, and can be utilized in commercial plants [3, 7]. The relative price of the different catalysts compared to iron are presented in Table 2.2.

<sup>3</sup>Stranded gas: gas that is far from the market, thus making transport uneconomical.

<sup>4</sup>Associated gas: gas that is co-produced with crude oil and which is currently being flared.

**Table 2.2:** Approximate relative price of catalysts, compared to iron [7].

Catalyst	Relative price
Fe*	1
Ni	250
Co	1 000
Ru	50 000

\* Fe as scrap metal.

Ni is very active. However it has high selectivity towards  $\text{CH}_4$  [7], which is an undesired product. Ru is the most active of the four, but low accessibility and high prices make it unsuitable for commercial applications. Thus, Co and Fe are the only viable catalysts for FT synthesis. The iron catalysts are known to be robust when it comes to syngas composition, but also to have a much shorter lifetime and to produce heavier, olefinic products compared to the cobalt catalyst [9]. The plants operated in Germany during the Second World War utilized a cobalt-based catalyst, whilst the plants in South Africa use catalysts which are iron-based.

In addition to being active, the catalyst has to meet other requirements pertaining to particle size, porosity and strength [3]. In large particles pore diffusion can be an issue. By decreasing the particle size or increasing the pore diameter this issue can be manipulated. The smaller the catalyst particles become in a fixed bed system, the tighter the bed will be packed and the greater the pressure drop over the reactor will be. Catalysts particles that are too small in a two or three phase system will cause a high loss of catalyst when separation is performed. If the catalyst does not meet the requirements on strength it can disintegrate. Too low porosity will result in low conversion. These properties must be taken into consideration and a compromise must be made between porosity and strength.

The different catalysts have different requirements when it comes to the syngas composition. For the cobalt based catalysts the dominant reaction is the FT reaction itself, see Equation (2.1). Thus, a syngas feed with a  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio close to the stoichiometric ratio is used, i.e. approximately 2.15 [7]. In an iron-based catalyst environment the water-gas-shift reaction, see Equation (2.2), competes with the FT reaction. CO is consumed and  $\text{H}_2$  is produced in the reaction, thus the syngas  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio can be

decreased (how low depends on the operation temperature).



Cobalt-based catalysts are more expensive. However the higher activity and the low (negligible) water-gas-shift reaction for these catalysts make them interesting to investigate further. Processes utilizing cobalt-based catalysts are primarily LTFT processes.

## 2.3 Reactor types

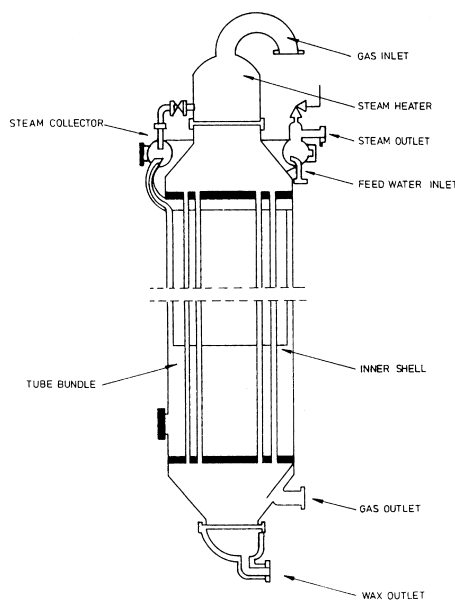
In the commercial history of the FT process three different reactor types have been utilized. These are fixed bed, fluidized bed and slurry bed reactors [9]. The highly exothermic nature of the FT process necessitates sufficient heat removal. Inefficient heat removal can result in an increased deactivation rate of the catalyst, an undesirably high production rate of  $\text{CH}_4$  and in the worst case runaway [7]. Runaway conditions are possible for highly exothermic reactions where the hot spot temperature can raise beyond permissible limits. These conditions, which can damage the equipment, are mainly determined by the temperature sensitivity of the reaction rates, the heat of reaction and the heat transfer potential of the heat exchanger [10].

For the LTFT process a large amount of the products will reside in the liquid phase under operation conditions. Thus, three phases are present. These are gas, liquid and solids (catalyst particles).

### 2.3.1 Fixed bed reactors

In fixed bed reactors the catalyst particles are packed, usually in a vertical cylinder, and the reactants and products pass through the stagnant bed [10]. There are three main reactor designs of fixed bed reactors. These are single-bed, multi-bed and multi-tube units. For highly exothermic reactions single-bed and multi-bed reactors do not offer adequate heat exchange. Thus, multi-tube fixed bed reactors, see Figure 2.2, are preferred. A multi-tube reactor can contain several hundred or thousand small-diameter tubes which are filled with catalyst. The diameter is kept small to avoid excessive temperature and hot spots.

Fixed bed reactors are often fed from the top. In the FT process the liquid products trickle down and out of the catalyst bed, accounting for the separation of liquids from the product stream, as seen in Figure 2.2.

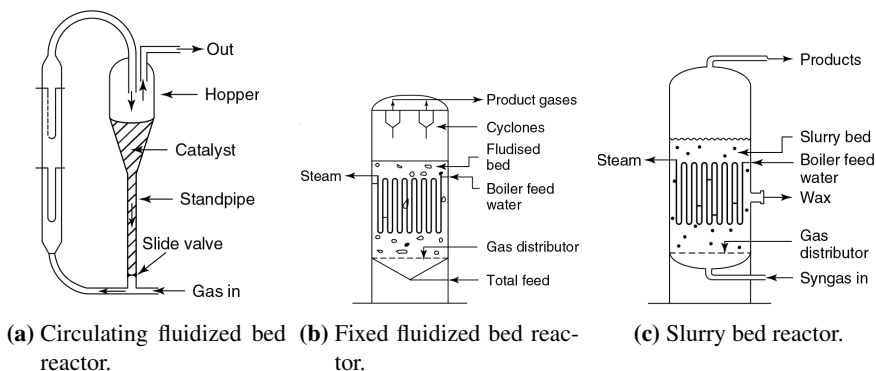


**Figure 2.2:** Multi-tubular fixed bed FT reactor [3].

### 2.3.2 Fluidized and slurry bed reactors

Fluidized reactors are preferred when a high degree of gas to solid contact, large throughput of gas and low pressure drop is required [10]. The syngas is passed through the reactor bed, fluidizing the catalyst. Hence, a close to isothermal reaction zone is obtained [9]. In slurry bed reactors the syngas is fed to the reactor and bubbled through the slurry phase consisting of catalyst particles and molten wax [7, 9]. The products which reside in the gas phase are withdrawn at the top, while the liquid products are continuously separated from the catalyst. Water and other lighter products can be separated from the gas stream after the reactor stage.

In Figure 2.3 two types of fluidized bed reactors and a slurry bed reactor are depicted.



**Figure 2.3:** Fluidized bed reactors [3]. (a) and (b) are two phase systems, while (c) is a three phase system.

*Advantages of fluidized/slurry bed reactors [8, 10]:*

- The ability to withdraw and reintroduce solids continuously.
- Possibility of continuous regeneration of the catalyst particles.
- Close to isothermal conditions, caused by the rapid mixing of solids. Low risk of hot spots, runaway and thermal instability, i.e. well suited for exothermic reactions.
- Low impact of internal and external diffusion phenomena because of the small particle size.
- Heat and mass transfer rates between gas and particles are high, compared with fixed bed reactors.
- The convective heat transfer coefficients at the heat exchanger surface immersed in the bed are high, i.e. internal heat exchangers require relatively small surface areas.

*Disadvantages of fluidized/slurry bed reactors [8, 10]:*

- For the same weight of catalyst, expansion of the bed requires an increase in reactor volume.
- The random movement of the particles causing back-mixing results in an overall

reactor behavior that is closer to an ideal mixed reactor (CSTR) than a plug flow reactor (PFR), which can lead to an increase in the reaction volume and a loss of selectivity.

- The entrainment of solid particles necessitates the installation of a device for separating and recycling fines (wax/catalyst separation).
- Friable solids are pulverized and entrained by the gas and must be replaced.
- Erosion of internals, pipes and vessels from abrasion by particles can be serious.
- Broad residence time distributions of solids due to intense mixing, erosion of the bed internals and attrition of the catalyst particles.
- Broad residence time distributions of the gas due to dispersion and gas bypass in the form of bubbles, especially when operated in the bubbling bed regime.
- Reactor hydrodynamics and modeling are complex. Scale-up and design present serious challenges which limit the use of these reactors to applications that can justify the significant research and development efforts involved.

There are many pros and cons with the different reactor designs. The main advantages of a slurry bed system [3, 7, 9], compared with a multi-tube fixed bed system, are lower investment costs, lower pressure drop, lower catalyst loading (higher activity of smaller catalyst particles yields lower catalyst consumption per tonne of product), close to isothermal conditions, on-line removal/addition of catalyst and a more uniform product distribution. The main disadvantage of a fluidized system is that if any catalyst poison enters the reactor, the total quantity of catalyst will be affected. In a fixed bed system only the top layers will be poisoned. The behavior in the liquid phase, and partly in the gas phase, in a simple slurry bed reactor resembles a CSTR as a result of the back-mixing that the rising gas bubbles introduce to the liquid phase [11, 12]. By the introduction of intermediate baffles this property can be counteracted and a mixing pattern resembling plug-flow can be obtained [11]. In addition, the scale-up of a slurry bed system is more complicated compared to the scale-up of a fixed bed system [9]. None of the reactor systems that are available possess all the optimal features. However, by considering the above mentioned arguments the slurry bed system may be the preferred choice, compared to a fixed bed, for the FT process [5, 13].

### 2.3.3 Heat transfer area density

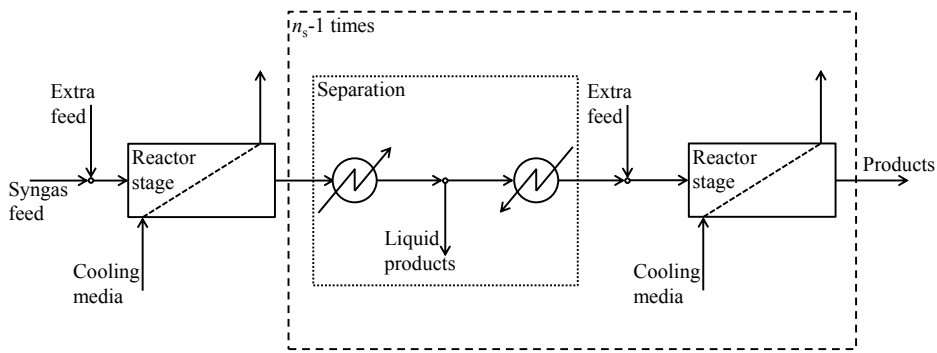
The heat transfer area density,  $a$  [ $\text{m}^2/\text{m}^3$ ], is highly dependent on the reactor system at hand. The geometry of cooling equipment, reactor and the physical properties of fluids and/or solids in the system will all have an effect on the properties of heat transfer. While a multi-tubular fixed bed reactor can have a heat transfer density of at least  $100\text{m}^2/\text{m}^3$  a slurry bed reactor has its maximum in the range  $30$  to  $35\text{m}^2/\text{m}^3$  [11]. Although the heat transfer density is smaller, the overall heat transfer coefficient,  $U$  [ $\text{kJ}/(\text{s}\text{m}^2\text{K})$ ], in a slurry bed is higher compared to a fixed bed reactor.  $U$  is dependent on the physical properties of the fluids in the system, superficial gas velocity, catalyst concentration, geometry of bed internals (e.g. pipe pitches), etc. [12]. The heat transfer, and ancillary assumptions, are elaborated further in Section 4.6.2 (p. 21).





### 3 Reactor Path

A flowsheet of the reactor path is depicted in Figure 3.1. The path consists of a syngas feed which is fed to the first reactor stage. Heat is removed from the reactor stage with cooling equipment. Separation and/or addition of an extra feed is possible before the entrance to the next reactor stage. The number of reactor stages ( $n_s$ ) in series is a design variable. Recycle of the product stream is not included in the reactor path, thus maximal conversion of CO is desired in a single pass.



**Figure 3.1:** Flowsheet of the reactor path.

#### 3.1 Reactor volume

In the simulations space time,  $\sigma$  [ $\text{m}^3 \text{ s/kg}$ ], has been given as one of the design variables.  $\sigma$  is defined in Equation (3.1).

$$\sigma = \frac{V_R}{W_0} \tag{3.1}$$

where  $V_R$  is the reactor volume [ $\text{m}^3$ ] and  $W_0$  is the feed flow [ $\text{kg/s}$ ].

$W_0$  is given by the desired production capacity. Large facilities can have a production in the region of 130000 bbl/d<sup>5</sup> [9]. With a product density,  $\rho_{\text{products}} \approx 0.8 \text{ kg/L}$ , the production is approximately 700 t/h. Results from the performed simulations show that

---

<sup>5</sup> 1 bbl = 159 L

liquid hydrocarbons amount to approximately 27% of the total product stream. As the total mass should be conserved, i.e. mass in equals mass out,  $W_0$  is calculated as approximately 720kg/s. Further, depending on the value of  $\sigma$ ,  $V_R$  can be calculated by rearranging Equation (3.1). In the performed simulations three  $\sigma$ -values were utilized. Table 3.1 presents the different  $\sigma$  and the corresponding total reactor volumes.

**Table 3.1:** Space time vs. reactor volume.

$\sigma$ [m <sup>3</sup> s/kg]	$V_R$ [m <sup>3</sup> ]
1.46	1050
1.0	720
0.8	580

## 4 Reactor Staging

To find the optimal design for the given kinetic model, a method of systematic staging in chemical reactor design presented by Hillestad [14] was utilized. The method involves the construction of a reactor path where reactants go through a series of functions to form the final products. The ideal reactor models (PFR and CSTR), with a homogeneous or pseudo-heterogeneous formulation, form the basis of the method. There may be several phases present, however they are restricted to have the same flow direction, velocity and dispersion.

In the following sections the method is elaborated.

### 4.1 Plug flow model

Steady-state mass balance for component  $i$  and the total mass balance for a PFR model are given in Equations (4.1) and (4.2), respectively.

$$W \frac{d\omega_i}{dV} = \tilde{R}_i + \alpha (\omega_{F,i} - \omega_i) \quad (4.1)$$

$$\frac{dW}{dV} = \alpha \quad (4.2)$$

where  $W$  is the fresh feed [kg/s],  $V$  is the volume [m<sup>3</sup>],  $\tilde{R}_i$  is the reaction rate<sup>6</sup> for component  $i$  [kg/(m<sup>3</sup> s)],  $\alpha$  is the mass flow rate per volume of extra feed [kg/(m<sup>3</sup> s)] and  $\omega_i$  and  $\omega_{F,i}$  are the weight fractions of component  $i$  in the fresh feed and the extra feed, respectively. The reaction rates are given by the kinetic model, presented in Section 5 (p. 25).

By disregarding shaft work, potential and kinetic energy, the steady-state energy balance is given by Equation (4.3).

$$WC_p \frac{dT}{dV} = \sum_j (-\Delta_r H_j) \tilde{r}_j + \alpha C_{p,F} (T_F - T) - Ua(T - T_W) \quad (4.3)$$

---

<sup>6</sup>The reaction rates,  $\tilde{R}_i$ , are on mass basis, i.e. the sum of all the rates will summarize to zero at any point,  $\sum_{i=1}^{n_c} \tilde{R}_i = 0$ .

where  $\Delta_r H_j$  is the heat of reaction  $j$  [kJ/kmol],  $\tilde{r}_j$  is the reaction rate of reaction  $j$  [kmol/(m<sup>3</sup> s)],  $U$  is the overall heat transfer coefficient [kJ/(m<sup>2</sup> s K)],  $a$  is the heat transfer area density along the reactor path [m<sup>2</sup>/m<sup>3</sup>],  $C_p$  and  $C_{p,F}$  are the composition average heat capacity of the reactor medium and the extra feed [kJ/(kg K)], respectively, and  $T$ ,  $T_F$  and  $T_W$  are the temperatures of the reactor medium, the feed and the coolant [K], respectively. By introducing dimensionless temperature,  $\theta = \frac{T - T_{\text{ref}}}{T_{\text{ref}}}$ , Equation (4.3) becomes

$$WC_p \frac{d\theta}{dV} = \sum_j \frac{(-\Delta_r H_j)}{C_p T_{\text{ref}}} \tilde{r}_j + \alpha C_{p,F} (\theta_F - \theta) - Ua (\theta - \theta_W) \quad (4.4)$$

where  $\theta$ ,  $\theta_F$  and  $\theta_W$  are the dimensionless temperatures for fluid, extra feed and cooling media, respectively. By defining a state vector,  $\mathbf{x}$ , consisting of all the mass fractions and the temperature, the mass and temperature equations can be depicted as a system of differential equations in vector form.

$$W \frac{d\mathbf{x}}{dV} = \tilde{\mathbf{R}}(\mathbf{x}) + \alpha \mathbf{K} \cdot (\mathbf{x}_F - \mathbf{x}) - \beta \mathbf{E} \cdot (\mathbf{x} - \mathbf{x}_W) \quad (4.5)$$

Where

$$\mathbf{x} = [\omega_1, \dots, \omega_n, \theta]^\top \quad (4.6)$$

$$\beta = \frac{Ua}{C_{p,\text{ref}}} \quad (4.7)$$

$$\mathbf{K} = \text{diag}\left(1, \dots, 1, \frac{C_{p,F}}{C_p}\right) \quad (4.8)$$

$$\mathbf{E} = \text{diag}\left(0, \dots, 0, \frac{C_{p,\text{ref}}}{C_p}\right) \quad (4.9)$$

$$\tilde{\mathbf{R}} = [\tilde{R}_1, \dots, \tilde{R}_n, \tilde{R}_\theta]^\top, \quad \tilde{R}_\theta = \sum_j \frac{(-\Delta_r H_j)}{C_p T_{\text{ref}}} \tilde{r}_j \quad (4.10)$$

With the introduction of the dimensionless variables  $\xi = V/V_R$  and  $\gamma = W/W_0$  representing the volume and mass flow rate, respectively, together with the space time,  $\sigma = V_R/W_0$ , the plug flow model can be represented as a dimensionless model in the

form.

$$\gamma \frac{d\mathbf{x}}{d\xi} = \sigma \tilde{\mathbf{R}}(\mathbf{x}) + \sigma \alpha \mathbf{K} \cdot (\mathbf{x}_F - \mathbf{x}) - \sigma \beta \mathbf{E} \cdot (\mathbf{x} - \mathbf{x}_W) \quad (4.11)$$

$$\frac{d\gamma}{d\xi} = \sigma \alpha \quad (4.12)$$

## 4.2 Completely mixed flow model

Steady-state mass balance for component  $i$ , the total mass balance and the energy balance for a CSTR model are presented in Equations (4.13) through (4.15), respectively. In the energy balance the dimensionless temperature has been introduced. In the same manner as for the plug flow model, shaft work, potential and kinetic energy have been neglected.

$$W_0(\omega_i - \omega_{i,0}) = V \tilde{R}_i + V \alpha (\omega_{F,i} - \omega_i) \quad (4.13)$$

$$W = W_0 + V \alpha \quad (4.14)$$

$$W_0(\theta - \theta_0) = V \sum_j \frac{(-\Delta_r H_j)}{C_{p,0}} \tilde{r}_j + V \alpha \frac{C_{p,F}}{C_{p,0}} (\theta_F - \theta) - V \frac{Ua}{C_{p,0}} (\theta - \theta_W) \quad (4.15)$$

In the same manner as for the plug flow model, the completely mixed model can be presented as a system of equations in vector form with the state vector,  $\mathbf{x}$ , now in algebraic form.

$$W_0(\mathbf{x} - \mathbf{x}_0) = V \tilde{\mathbf{R}}_0(\mathbf{x}) + V \alpha \mathbf{K}_0(\mathbf{x}_F - \mathbf{x}) - V \beta \mathbf{E}_0(\mathbf{x} - \mathbf{x}_W) \quad (4.16)$$

where  $\beta$ ,  $\mathbf{K}_0$ ,  $\mathbf{E}_0$  and  $\mathbf{R}_0$  are defined in the same manner as Equations (4.7) - (4.10). The subscript zero states that the inlet conditions for the heat capacity is utilized.

By differentiating Equation (4.16) with respect to volume,  $V$ , the completely mixed model can be written more similarly to the plug flow model.

$$W_0 \frac{d\mathbf{x}}{dV} = \tilde{\mathbf{R}}_0(\mathbf{x}) + V \frac{\partial \tilde{\mathbf{R}}_0(\mathbf{x})}{\partial \mathbf{x}} \frac{d\mathbf{x}}{dV} + \alpha \mathbf{K}_0(\mathbf{x}_F - \mathbf{x}) - V \alpha \mathbf{K}_0 \frac{d\mathbf{x}}{dV} - \beta \mathbf{E}_0(\mathbf{x} - \mathbf{x}_W) - V \beta \mathbf{E}_0 \frac{d\mathbf{x}}{dV} \quad (4.17)$$

By manipulating Equation (4.17) it can be presented as

$$[W\mathbf{I} - V\tilde{\mathbf{J}}] \frac{d\mathbf{x}}{dV} = \tilde{\mathbf{R}}_0(\mathbf{x}) + \alpha\mathbf{K}_0(\mathbf{x}_F - \mathbf{x}) - \beta\mathbf{E}_0(\mathbf{x} - \mathbf{x}_W) \quad (4.18)$$

where

$$\tilde{\mathbf{J}} = \frac{\partial \tilde{\mathbf{R}}_0(\mathbf{x})}{\partial \mathbf{x}} + \text{diag}(0, \dots, 0, 1) \left( \alpha \left( 1 - \frac{C_{p,F}}{C_{p,0}} \right) - \beta \frac{C_{p,\text{ref}}}{C_{p,0}} \right) \quad (4.19)$$

With the introduction of  $\sigma$  and the same dimensionless variables,  $\xi$  and  $\gamma$ , as for the plug flow model, the completely mixed model can be presented equivalently to Equation (4.11).

$$[\gamma\mathbf{I} - \xi\sigma\tilde{\mathbf{J}}] \frac{d\mathbf{x}}{d\xi} = \sigma\tilde{\mathbf{R}}_0(\mathbf{x}) + \sigma\alpha\mathbf{K}_0(\mathbf{x}_F - \mathbf{x}) - \sigma\beta\mathbf{E}_0(\mathbf{x} - \mathbf{x}_W) \quad (4.20)$$

### 4.3 Composite model

By defining three dimensionless design functions for mixing,  $u_M$ , distribution of feed,  $u_F$ , and heat transfer area distribution,  $u_H$ , and comparing Equations (4.11) and (4.20) it is evident that these two equations can be merged into a single system of equations. The system is presented in Equation (4.21). With the definition of  $u_F$  the total mass is given by Equation (4.22).

$$[\gamma\mathbf{I} - u_M\sigma\tilde{\mathbf{J}}] \frac{d\mathbf{x}}{d\xi} = \sigma\tilde{\mathbf{R}}(\mathbf{x}) + u_F\mathbf{K}(\mathbf{x}_F - \mathbf{x}) - u_H\mathbf{E}(\mathbf{x} - \mathbf{x}_W) \quad (4.21)$$

$$\frac{d\gamma}{d\xi} = u_F \quad (4.22)$$

In addition to the three design functions describing the mixing, distribution of feed and heat transfer area density, other design functions like catalyst dilution,  $u_A$ , coolant temperature,  $u_T$ , and separation  $u_S$ , can be defined. By utilizing these design functions the optimal reactor path can be determined. All the design functions are elaborated in Section 4.6 (p. 20).

## 4.4 Optimization

The goal of the optimization is to find the optimal reactor path, i.e. find the parameters and their values that yield the best environment for the reacting fluid throughout the path. The problem is written and solved as an optimal control problem, which can be defined as

$$\max_{[\sigma, \mathbf{u}] \in \mathcal{U}} J \quad (4.23)$$

$$\text{s.t.} \quad \frac{d\mathbf{z}}{d\xi} = \mathbf{f}(\mathbf{z}, \mathbf{u}), \quad \mathbf{z}(0) = \mathbf{z}_0 \quad (4.24)$$

where  $J$  is a cost function,  $\sigma$  is the space time [ $\text{m}^3 \text{s}/\text{kg}$ ] and  $\mathbf{u}$  is a vector consisting of the design functions and their boundaries. Thus,  $\mathcal{U}$  is a vector which defines the solution space where  $J$  can be solved. The vector  $\mathbf{z}$  consists of the old state vector,  $\mathbf{x}$ , and the total mass flow, i.e.  $\mathbf{z} = [\mathbf{x}^\top, \gamma]^\top$ . The subscript zero denotes an initial guess. The model,  $\mathbf{f}$ , is a system of ordinary differential equations (ODEs) consisting of Equations (4.21) and (4.22). The size of the system is dependent on the number of components (reacting species and temperature), the number of stages and the number of collocation points within each stage<sup>7</sup>.

In addition to maximize the cost function with the reactor model in mind, additional path constraints can be added,  $\mathbf{h}$ . Constraints like these can be maximum/minimum temperature, maximum/minimum reactor volume, maximum/minimum weight fraction of a component, etc. If present, the constraints are presented as nonlinear inequality constraints as depicted in Equation (4.25).

$$\mathbf{h}(\mathbf{z}, \mathbf{u}) \leq \mathbf{0} \quad (4.25)$$

These constraints will add to the system of ODEs. The total number of equations will increase rapidly as each constraint has to be solved in each collocation point. Thus, if possible, extended use of nonlinear inequality constraints is not recommended.

<sup>7</sup>Size of ODE system: If there are nine reacting species, three reactor stages and the number of internal collocation points are set to 20, the ODE system will consist of  $(9+2) \times 3 \times (20+2) = 726$  equations. All of which are to be solved simultaneously.

## 4.5 Cost functions

The cost function describes the desired goal. An obvious choice of cost function will be to maximize the production of the desired (key) product, i.e. the weight fraction of  $C_{11+}$  should be as high as possible at the end of the reactor path. If no products are separated from the reactor path this would be a suitable objective function. In the case where products are separated from the reactor path, the total production will be an improved objective function. The two objective functions are presented in Equations (4.26) and (4.27), respectively.

$$1. \quad J = \omega_{\text{key}}(1) \quad (4.26)$$

$$2. \quad J = \text{prod}_{\text{key}} \quad (4.27)$$

## 4.6 Design functions

The path optimization consists of a number of design functions.

- $u_M$  – fluid mixing
- $u_H$  – heat transfer area distribution
- $u_T$  – cooling media temperature profile
- $u_F$  – extra feed (distribution)
- $u_A$  – catalyst activity
- $u_S$  – separation

The design functions can be kept constant or optimized. If optimized they are either stage-wise constant or stage-wise linear.

In the following sections these functions will be further elaborated.

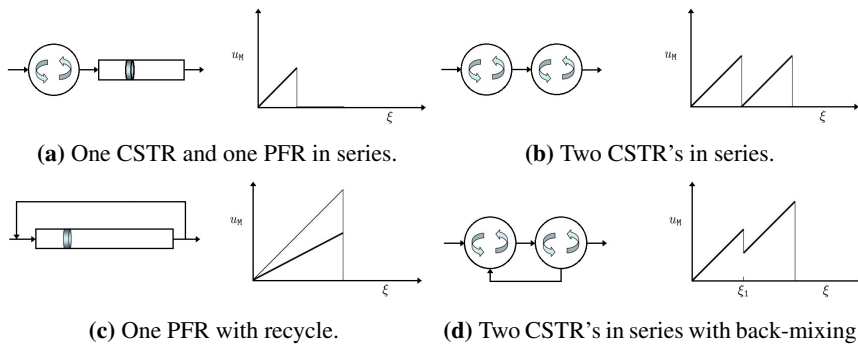


### 4.6.1 Mixing

The design function  $u_M$  describes the mixing in the reactor.

$$u_M = \xi \quad (4.28)$$

Upon integration of Equation (4.21) the choice of  $u_M$  determines the kind of reactor model that is considered. By setting  $u_M$  equal to zero Equation (4.11) is obtained, i.e. the plug flow model. By setting  $u_M$  equal to  $\xi$  the completely mixed model in Equation (4.20) is obtained. It can be shown that for an  $u_M \in (0, \xi)$  the path will consist of a plug flow reactor with recycle. Different designs are depicted in Figure 4.1



**Figure 4.1:** Different mixing design function scenarios.

### 4.6.2 Heat transfer area distribution

The design function  $u_H$  describes the heat transfer area distribution along the reactor path.

$$u_H = \frac{Ua\sigma}{C_p} \quad (4.29)$$

where  $U$  is the overall heat transfer coefficient [ $\text{kJ}/(\text{s m}^2 \text{ K})$ ],  $a$  is the heat transfer density [ $\text{m}^2/\text{m}^3$ ],  $\sigma$  is the space time [ $\text{m}^3 \text{ s}/\text{kg}$ ] and  $C_p$  is the heat capacity of the reactor fluid [ $\text{kJ}/(\text{kg K})$ ].  $a$  is defined as

$$a = \frac{A}{V} \quad (4.30)$$

where  $A$  is the heat transfer area [ $\text{m}^2$ ] and  $V$  is the volume of the reactor stage [ $\text{m}^3$ ].

In the calculations the overall heat transfer coefficient and the fluid heat capacity are assumed to be  $1.5 \text{ kJ}/(\text{s m}^2 \text{ K})$  and  $2.1 \text{ kJ}/(\text{kg K})$ , respectively. They are both assumed to be constant throughout the reactor. The design parameter  $\sigma$  is set for each individual optimization.

The optimization yields a numeric value for  $u_H$  and by rearranging Equation (4.29)  $a$  can be found. Further,  $A$  can be calculated by rearranging Equation (4.30). The heat transfer area is dependent on the reactor design, which is further discussed in Section 2.3 (p. 7).

### 4.6.3 Cooling media temperature profile

The design function  $u_T$  describes the cooling media temperature profile.

$$u_T = \theta_W \quad (4.31)$$

where  $\theta_W$  is the dimensionless temperature, defined as

$$\theta_W = \frac{T_W - T_{\text{ref}}}{T_{\text{ref}}} \quad (4.32)$$

where  $T_W$  and  $T_{\text{ref}}$  are the temperature of the cooling media and a reference temperature [K], respectively.

In the performed simulations the cooling media is assumed to be vaporizing water, i.e. the temperature is constant throughout the reactor volume. As the FT-reaction is highly exothermic the cooling can be utilized to produce steam. It is beneficial to only have a single steam system, with a given temperature and pressure. In respect to this observation, most of the simulations are performed so that the temperature of the cooling media is only optimized in the first stage and kept constant in the subsequent stages.

At a pressure of 20 bar the boiling point of water is approximately  $215^\circ\text{C}$  [15].

#### 4.6.4 Distribution of extra feed

The design function  $u_F$  describes the distribution of extra feed along the reactor path.

$$u_F = \frac{\alpha \sigma}{\Delta \xi} \quad (4.33)$$

where  $\alpha$  is the feed distribution [ $\text{kg}/(\text{m}^3 \text{s})$ ],  $\sigma$  is the space time [ $\text{m}^3 \text{s}/\text{kg}$ ] and  $\Delta \xi$  is the size of the reactor stage.

Each addition of extra feed with an unique composition is associated with an individual design function, regardless of the number of times the feed is added, i.e. addition of pure  $\text{H}_2$  at numerous points in the reactor path yields only one design function, while addition of two extra feeds, with different compositions, yields two design functions, and so forth.

The extra feed can either be point fed prior to the reactor stages or be distributed along the stages. In the performed simulations pure  $\text{H}_2$  has been used as an extra feed, and it has only been added as point feed prior to the stages.

#### 4.6.5 Catalyst activity

The design function  $u_A$  describes the relative catalyst activity, i.e. the proportion of the catalyst that will participate in the reaction.

$$u_A \in [0, 1] \quad (4.34)$$

Fresh catalyst will have an activity of one. Over time, catalyst deactivation due to degradation and impurities will occur. A list of the most common factors involving catalyst deactivation is presented below [3, 8].

- Fouling of the catalyst surface by coke deposits
- Poisons in the gas feed (e.g. sulphur)
- Hydrothermal sintering
- Oxidation of active phase to inactive oxide

On the other hand, continued research and development can increase the activity of the catalyst. The kinetic model does not take into account any deactivation, thus the optimal solution should always be maximum catalyst activity, i.e.  $u_A = 1$ . In an attempt to reduce the number of variables in the performed optimizations the catalyst activity is assumed to be constant and equal to one, throughout the reactor path and over time.

#### 4.6.6 Separation

The design function  $u_S$  describes the separation of components from the reactor path. As for the addition of extra feed, the separation can either be point-wise or distributed along the reactor path. In the performed simulations the separation has been point-wise.

In addition to the hydrocarbons, significant amounts of water are produced in the FT synthesis. To ensure high partial pressures of  $H_2$  and  $CO$ , and consequently high reaction rates, it may be beneficial to separate the products and the water between each stage. In practice the separation process will consist of a cooling, a separation and a reheating process, resembling the process depicted in the “Separation box” in Figure 3.1 (p. 13). The properties of the separation are specified by the equipment at hand. Thus, the design function has not been updated in the optimizations. In the simulations the separation is defined by a single vector,  $\mathbf{xS}$ , consisting of the component fractions that will be separated.

The cooling and reheating of the stream will in practice be heat integrated. Hence, it can be assumed that most of the energy will be conserved. However, as some will be lost the temperature of the stream after the separation is assumed to be  $10^\circ C$  lower than before the separation.

To find the respective separation values a set of streams, with different compositions, were constructed in UniSim Design R400. The streams were cooled to  $40^\circ C$  and the respective liquid fractions were calculated. The resulting separation function is presented in Equation (4.35).

$$\mathbf{xS} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0.7 & 1 \end{bmatrix} \quad (4.35)$$

Thus, 70wt% of the  $C_{5-10}$  lump and 100wt% of the  $H_2O$  and  $C_{11+}$  lump are separated.

## 5 Kinetic Model

The kinetic model adapted is presented by Todic et al. [1]. The model is found to predict the major product distribution characteristics. It is derived with a Langmuir-Hinshelwood-Hougen-Watson (LHHW) approach<sup>8</sup> on a 25% Co/0.48% Re/Al<sub>2</sub>O<sub>3</sub> catalyst in a slurry reactor. In this manner, the rate of hydrocarbon formation is made a function of kinetic constants and the partial pressures of CO and H<sub>2</sub>. The main assumptions of the model are listed below.

- Only one type of FT-synthesis active site is present on the Co catalyst surface.
- The total number of active sites on the catalyst surface is constant.
- The concentrations of surface intermediates and vacant sites are at steady state.
- CH<sub>4</sub> and C<sub>2</sub>H<sub>6</sub> have different formation rate constants than other *n*-paraffins and 1-olefins, respectively.
- Rate constants of chain propagation and hydrogenation to *n*-paraffins and 1-olefins are independent of carbon number (chain length).
- The rate constant of chain desorption to form 1-olefin is exponentially dependent on carbon number.
- Elementary steps for the formation of *n*-paraffins and 1-olefins are rate-determining steps, as is one of the elementary steps involved in chain propagation or monomer formation. All other elementary steps are considered to be quasi-equilibrated.
- Minor FT-synthesis products, e.g. 2-olefins and oxygenates, are disregarded.

The formation rates are defined as functions of the kinetic constants, the partial pressure of H<sub>2</sub> and the surface fraction of various growing chain intermediates, [C<sub>*n*</sub>H<sub>2*n*+1</sub>-S], (see Equations (5.8) through (5.11)). All the model parameters are intrinsic kinetic constants, i.e. the parameters satisfy physiochemical laws in addition to providing a good

---

<sup>8</sup>A LHHW mechanism assumes that all species are adsorbed on the catalyst surface before reacting [16]. The mechanism is constructed by assuming a sequence of reversible, elementary reaction steps [17]. Rate expressions for the different steps are postulated before a step is chosen as rate determining (irreversible). The remaining steps are used to eliminate the catalyst-coverage-dependent terms in the rate expression.

fit to the experimental data.

The chain growth probability,  $\alpha$ , is the key factor connecting the surface fraction of growing chain intermediates to the kinetic constants, partial pressures of CO and H<sub>2</sub> and fraction of vacant sites, [S], on the catalyst surface. It is assumed that CH<sub>4</sub> and C<sub>2</sub>H<sub>6</sub> have different formation rates than the other hydrocarbons. Thus, the  $\alpha$ -values of these components are defined separately. Equations (5.1) through (5.3) present the chain growth probabilities.

$$\alpha_1 = \frac{[\text{CH}_3\text{-S}]}{[\text{H-S}]} = \frac{k_1 P_{\text{CO}}}{k_1 P_{\text{CO}} + k_{5M} P_{\text{H}_2}} \quad (5.1)$$

$$\alpha_2 = \frac{[\text{C}_2\text{H}_5\text{-S}]}{[\text{CH}_3\text{-S}]} = \frac{k_1 P_{\text{CO}}}{k_1 P_{\text{CO}} + k_5 P_{\text{H}_2} + k_{6E} e^{2c}} \quad (5.2)$$

$$\alpha_n = \frac{[\text{C}_n\text{H}_{2n+1}\text{-S}]}{[\text{C}_{n-1}\text{H}_{2n-1}\text{-S}]} = \frac{k_1 P_{\text{CO}}}{k_1 P_{\text{CO}} + k_5 P_{\text{H}_2} + k_{6,0} e^{cn}}, \quad n \geq 3 \quad (5.3)$$

where  $P_{\text{CO}}$  and  $P_{\text{H}_2}$  are partial pressures, the  $k_i$ s are kinetic constants and  $n$  denotes the number of C atoms.

The total number of active sites on the catalyst is assumed constant. Thus, [S] can be calculated by Equation (5.4).

$$[\text{S}] = 1 / \left\{ 1 + \sqrt{K_7 P_{\text{H}_2}} + \sqrt{K_7 P_{\text{H}_2}} \left( 1 + \frac{1}{K_4} + \frac{1}{K_3 K_4 P_{\text{H}_2}} + \frac{1}{K_2 K_3 K_4} \frac{P_{\text{H}_2\text{O}}}{P_{\text{H}_2}^2} \right) \times \left( \alpha_1 + \alpha_1 \alpha_2 + \alpha_1 \alpha_2 \sum_{i=3}^n \prod_{j=3}^i \alpha_j \right) \right\} \quad (5.4)$$

The kinetic constants,  $k_i$  and  $K_i$ , and  $c$  are defined in Equations (5.5) through (5.7), respectively.

$$k_i(T) = A_i \exp\left(-\frac{E_{a,i}}{RT}\right) \quad (5.5)$$

$$K_i(T) = A_i \exp\left(-\frac{\Delta H_i}{RT}\right) \quad (5.6)$$

$$c = -\frac{\Delta E}{RT} \quad (5.7)$$

where  $R$  is the universal gas constant [kJ/(molK)] and  $T$  is the temperature [K]. The respective factors ( $\Delta E$ ,  $A_i$ ,  $E_{a,i}$  and  $\Delta H$ ) are presented in Table B.1 in Appendix B.

From the proposed reaction mechanism by Todici et al. [1], the reaction rates for  $\text{CH}_4$ ,  $\text{C}_2\text{H}_6$ ,  $n$ -paraffins and 1-olefins can be defined as Equations (5.8) through (5.11).

$$\begin{aligned} R_{\text{CH}_4} &= k_{5\text{M}}[\text{CH}_3\text{-S}]P_{\text{H}_2} \\ &= k_{5\text{M}}K_7^{0.5}P_{\text{H}_2}^{1.5}\alpha_1[\text{S}] \end{aligned} \quad (5.8)$$

$$\begin{aligned} R_{\text{C}_2\text{H}_4} &= k_{6\text{E},0}e^{2c}[\text{C}_2\text{H}_4\text{-S}] \\ &= k_{6\text{E},0}e^{2c}\sqrt{K_7P_{\text{H}_2}}\alpha_1\alpha_2[\text{S}] \end{aligned} \quad (5.9)$$

$$\begin{aligned} R_{\text{C}_n\text{H}_{2n+2}} &= k_5[\text{C}_n\text{H}_{2n+1}\text{-S}]P_{\text{H}_2} \\ &= k_5K_7^{0.5}P_{\text{H}_2}^{1.5}\alpha_1\alpha_2\prod_{i=3}^n\alpha_i[\text{S}] \quad n \geq 2 \end{aligned} \quad (5.10)$$

$$\begin{aligned} R_{\text{C}_n\text{H}_{2n}} &= k_{6,0}e^{cn}[\text{C}_n\text{H}_{2n+1}\text{-S}]P_{\text{H}_2} \\ &= k_{6,0}e^{cn}\sqrt{K_7P_{\text{H}_2}}\alpha_1\alpha_2\prod_{i=3}^n\alpha_i[\text{S}] \quad n \geq 3 \end{aligned} \quad (5.11)$$

By coupling Equations (5.8) through (5.11) with expressions for  $\alpha$ , Equations (5.1) through (5.3), and  $[\text{S}]$ , Equation (5.4), the production rates can be explicitly calculated. The consumption rates of CO and  $\text{H}_2$  are calculated by summation of the rates of hydrocarbon consumption, see Equations (5.12) and (5.13), respectively.

$$R_{\text{CO}} = \sum_{n=1}^{15} n(R_{\text{C}_n\text{H}_{2n+2}} + R_{\text{C}_n\text{H}_{2n}}) \quad (5.12)$$

$$R_{\text{H}_2} = \sum_{n=1}^{15} [(2n+1)R_{\text{C}_n\text{H}_{2n+2}} + 2nR_{\text{C}_n\text{H}_{2n}}] \quad (5.13)$$

All rates are given in mol/(g<sub>cat</sub>h). The method at hand utilizes weight basis. Thus the rates must be converted. By multiplying by the bulk catalyst density,  $\rho_{\text{cat}}$ , and the molecular weight,  $M$ , the rates can readily be converted to kg/(m<sup>3</sup>s).

## 5.1 Lumping of components

In the performed simulations hydrocarbons up to and including C<sub>15</sub> have been examined. In addition to *n*-paraffins and 1-olefins the system consists of H<sub>2</sub>, CO, H<sub>2</sub>O and CO<sub>2</sub>. The total number of reacting species is 33. To decrease the number of variables in the simulations some hydrocarbons have been lumped together to create a set of hypothetical components. The resulting components are presented in the following list.

- C<sub>1</sub>
- C<sub>2</sub>, both *n*-paraffin and 1-olefin
- C<sub>3-4</sub>, both *n*-paraffins and 1-olefins
- C<sub>5-10</sub>, both *n*-paraffins and 1-olefins
- C<sub>11+</sub>, both *n*-paraffins and 1-olefins

The lumping decreases the total number of reacting species in the system to nine.



# 6 Simulations and Raw Data Processing

The simulations performed in this study have been executed within MATLAB R2012b. Physical data has been extracted from UniSim Design R400. The MATLAB scripts, together with three simulation hierarchies depicting the simulation flow, are presented in Appendix C. An overview of the simulations is presented in Table 6.1.

In the following, the variables that have been kept constant in the simulations are listed.

- Feed temperature,  $T_F = 200^\circ\text{C}$
- Reference temperature,  $T_{\text{ref}} = 473\text{ K}$
- System pressure,  $P_{\text{tot}} = 20\text{ bar}$
- Catalyst density,  $\rho_{\text{cat}} = 200\text{ kg/m}^3$
- Catalyst activity,  $u_A = 1$
- Molecular weight of lumps (see Table 6.2)
- Heat capacity of lumps (see Table 6.2)
- Number of components,  $n_c = 9$
- Number of internal collocation points,  $n_{\text{col}} = 20$
- Maximum reactor temperature,  $T_{\text{max}} = 250^\circ\text{C}$
- Molar fraction of inert,  $x_{\text{CO}_2} = 0.1$

In addition, for those cases where separation is considered,  $u_S$  is not updated. The separation is set by the  $\mathbf{xS}$ -vector as discussed in Section 4.6.6 (p. 24).

**Table 6.1:** Simulation overview stating number of stages, updated design functions, space time,  $\sigma$ , and cost function utilized in the different cases.

Case	$n_s$	Update*				$\sigma$ [m <sup>3</sup> /kg]	J <sup>†</sup>
		$u_M$	$u_H$	$u_T$	$u_F$		
1	1	All	No	No	No	1.46	1
2	1	All	All	All	No	1.46	1
3	1	All	All	All	No	1.00	1
4	1	All	All	All	No	0.80	1
5	1	All	All	All	All	1.46	1
6	1	All	All	All	All	1.00	1
7	1	All	All	All	All	0.80	1
8	2	All	No	No	No	1.46	1
9	2	All	All	No	No	1.46	1
10	2	All	All	All	No	1.46	1
11	2	All	All	All	No	1.46	1
12	2	All	All	All	No	1.46	1
13	2	All	All	All	No	1.46	1
14	2	All	No	No	2 <sup>nd</sup>	1.46	1
15	3	All	No	No	3 <sup>rd</sup>	1.46	1
16	3	All	No	No	3 <sup>rd</sup>	1.46	1
17	3	All	All	All	No	1.46	1
18	3	All	All	1 <sup>st</sup>	2 <sup>nd</sup> , 3 <sup>rd</sup>	1.46	2
19	3	All	All	1 <sup>st</sup>	2 <sup>nd</sup> , 3 <sup>rd</sup>	1.46	2
20	3	All	All	1 <sup>st</sup>	All	1.46	2
21	3	All	All	1 <sup>st</sup>	All	1.00	2
22	3	All	All	1 <sup>st</sup>	All	0.80	2
23	3	All	All	1 <sup>st</sup>	All	0.80	1

\* If updated,  $u_H$  and  $u_T$  are updated stage-wise constant,  $u_F$  is introduced as point feed prior to the reactor stage and  $u_M$  is updated stage-wise linear.

† 1:  $J = \omega_{\text{key}}(1)$ , (Equation (4.26))

2:  $J = \text{prod}_{\text{key}}$ , (Equation (4.27))

## 6.1 Input data

Physical data, i.e. molecular weight and heat capacity, was extracted from UniSim Design R400. The molecular weight,  $M$ , [kg/mol] of the hydrocarbon lumps was calculated by assuming equimolar amounts of chemical species, i.e. the  $C_{3-4}$ -lump was assumed to consist of 25% of each of  $C_3H_8$ ,  $C_3H_6$ ,  $C_4H_{10}$  and  $C_4H_8$ . To minimize numerical noise in the optimization the molecular weight was rounded to integers. The heat capacity,  $C_p$ , [kJ/kg] was calculated at the feed temperature and pressure, and was assumed constant. As for the molecular weight, the lumps were assumed to consist of equimolar amounts of chemical species when  $C_p$  was calculated. Both the molecular weights and heat capacities are presented in Table 6.2.

**Table 6.2:** Physical properties of the component lumps.

<b>Component</b>	$M$ [kg/kmol]	$C_p$ [kJ/(kg K)]
CO	28	1.07
CO <sub>2</sub>	44	1.02
H <sub>2</sub>	2	14.34
H <sub>2</sub> O	18	4.89
C <sub>1</sub>	16	2.83
C <sub>2</sub>	29	2.36
C <sub>3-4</sub>	50	2.44
C <sub>5-10</sub>	106	3.10
C <sub>11+</sub>	183	2.79

## 6.2 Raw data processing

The design functions are dimensionless. As a consequence they have to a greater or lesser degree been processed in order to present the results with a more physical significance.

The mixing properties are presented as either PFR or CSTR. The heat transfer area distribution is presented as the heat transfer density,  $a$  [m<sup>2</sup>/m<sup>3</sup>], which is found by

rearranging Equation (4.29).

$$a = \frac{u_H C_p}{U \sigma} \quad (6.1)$$

The cooling media temperature profile is presented in degrees Celsius by coupling Equations (4.31) and (4.32), and rearranging for  $T_W$ .

$$T_W = (1 + u_T) T_{\text{ref}} - 273 \quad (6.2)$$

The distribution of extra feed is presented as feed distribution,  $\alpha$  [kg/(m<sup>3</sup>s)], found by rearranging Equation (4.33).  $\alpha$  can be regarded as a fraction of the fresh feed. The feed is kept constant in all simulations, thus the volume is defined by the value of  $\sigma$ . Multiplying  $\alpha$  with the total reactor volume,  $V_R$ , will yield a flow rate in kg/s for the additional feed. However, in this thesis, all data has been chosen to be presented on a fractional basis.

$$\alpha = \frac{u_F}{\sigma} \Delta \xi \quad (6.3)$$

The production of key component,  $\mathcal{P}_{C_{11+}}$ , provides as a measure of the performance of the different simulation cases. The production is calculated by Equation (6.4). The superscript “end” in Equation (6.4) refers either to the end of the reactor path, if the design is one-staged, or to the respective reactor stage, if the design is multi-staged. In multi-staged cases, where separation is included, the total production is found by summation of the contribution from each stage.

$$\mathcal{P}_{C_{11+}} = \omega_{C_{11+}}^{\text{end}} \gamma^{\text{end}} \quad (6.4)$$

where  $\omega_{C_{11+}}$  is the weight fraction of key component and  $\gamma$  is the total weight balance defined as  $\gamma = W/W_0$ <sup>9</sup>. Following from the definition of  $\gamma$ , the production is given as a weight fraction of the fresh feed, i.e. if the production is 10% and  $W_0$  is 100kg/s, 10kg/s of  $C_{11+}$  will be produced. The fresh feed is, as stated above, kept constant in all simulations.

---

<sup>9</sup>Value of  $\gamma$ :

$\gamma = 1$ ; no extra feed and no products are separated (or the same amount fed and separated)

$\gamma > 1$ ; greater amount of additional feed compared to separation

$\gamma < 1$ ; greater amount of separation compared to additional feed

The conversion of CO,  $X_{\text{CO}}$ , is presented on weight basis, i.e.

$$X_{\text{CO}} = 1 - \frac{\omega_{\text{CO}}^{\text{end}}}{\omega_{\text{CO}}^0} \gamma^{\text{end}} \quad (6.5)$$

where  $\omega_{\text{CO}}^{\text{end}}$  and  $\omega_{\text{CO}}^0$  is the weight fraction of CO at the beginning and end of the reactor path, respectively.  $\gamma^{\text{end}}$  is the total mass balance at the end of the reactor path.

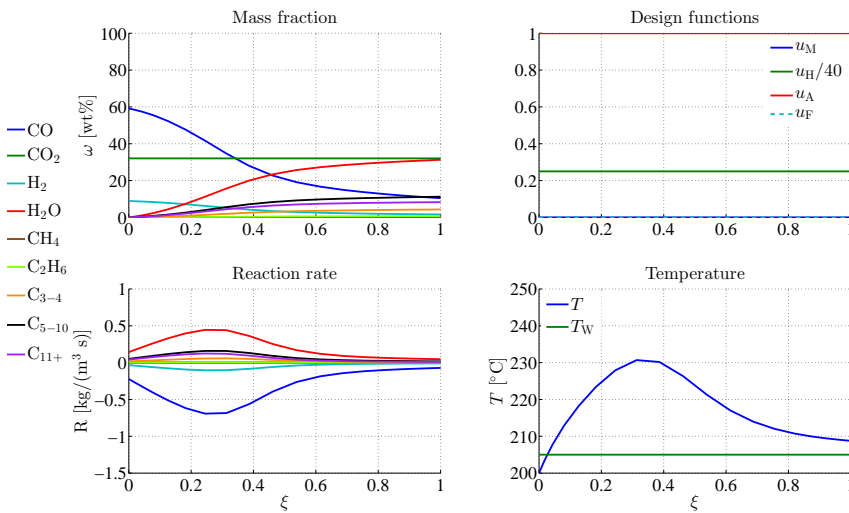


# 7 Results

In total, 23 different cases have been simulated. An overview of the results is presented in a set of tables in Appendix A. Most of the optimization results are in a dimensionless form. In order to derive the physical significance of the results, both raw and processed data are presented in the appendix. In the following section some extracts are given and the cases which are considered of highest importance and interest are presented.

## 7.1 Case studies

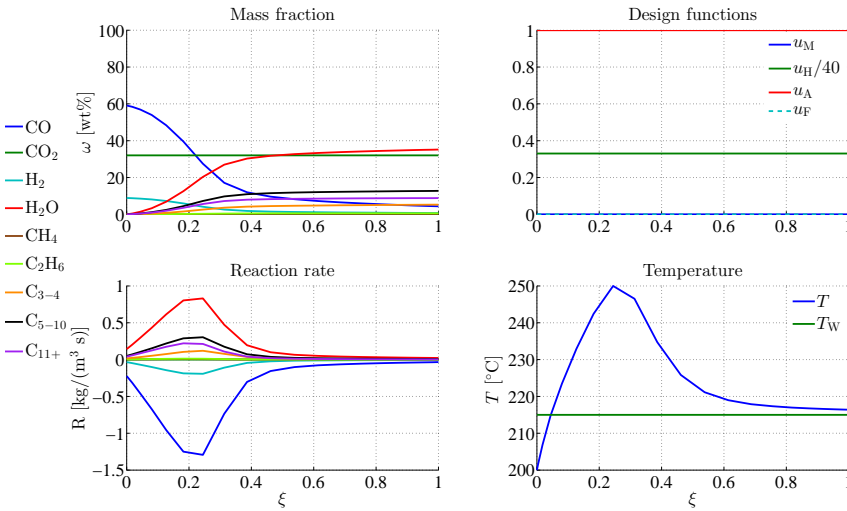
The first case performed (Case 1) was the simplest possible design, i.e. one stage with no addition of extra  $H_2$  feed. Thus, the molar feed ratio,  $x_{H_2}/x_{CO}$ , was set equal to 2.1, i.e. approximately the stoichiometric ratio. Only  $u_M$  was optimized, while  $\sigma$  was set equal to  $1.46\text{ m}^3\text{ s/kg}$ ,  $u_H$  equal to 10 and  $T_W$  equal to  $205^\circ\text{C}$ . The resulting design is depicted in Figure 7.1. The design resulted in a  $X_{CO}$  equal to 82.21 %, a  $J$  equal to  $-0.0828$  and a  $\mathcal{P}_{C_{11+}}$  equal to 8.28 %.



**Figure 7.1:** Case 1: One reactor stage. Mixing is optimized.  $\sigma = 1.46\text{ m}^3\text{ s/kg}$ .

The design in Case 1 was slightly changed to include two reactor stages (Case 8). All other variables were kept alike. This setup yielded, as expected, the same results as Case 1.

To find a more optimal design for the one-stage system both  $u_H$  and  $u_T$  was added to the optimization (Case 2). The  $u_H$  design function was allowed to vary within 5 to 30 and  $T_W$  within 195 to 215 °C.  $x_{H_2}/x_{CO}$  and  $\sigma$  were kept constant at 2.1 and 1.46 m<sup>3</sup> s/kg, respectively. The resulting design is depicted in Figure 7.2. The optimization yielded an  $u_H$  equal to 13.2, which corresponds to an area density of 12.7 m<sup>2</sup>/m<sup>3</sup> (if  $U$  is assumed to be 1.5 kJ m<sup>2</sup>/(s K)), and a  $T_W$  equal to 215 °C. The design resulted in an increase in  $X_{CO}$  to 92.59%, a  $J$  equal to  $-0.0883$  and a  $\mathcal{P}_{C_{11+}}$  equal to 8.83%.

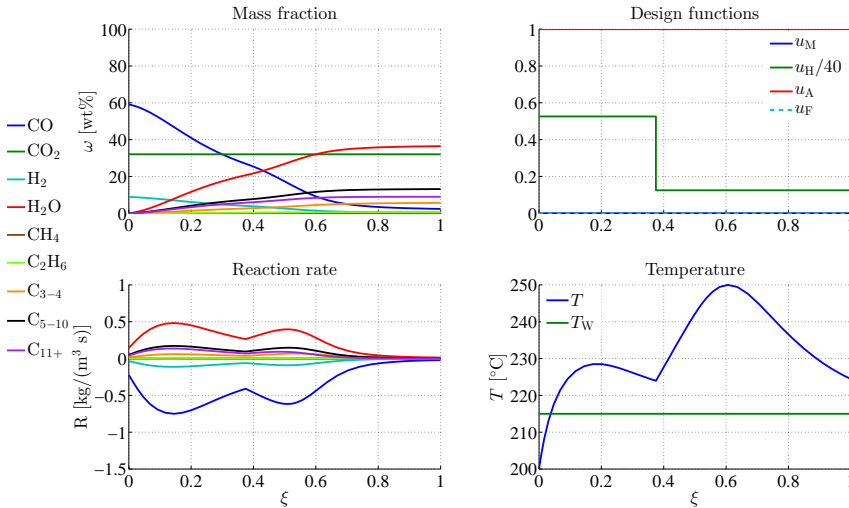


**Figure 7.2:** Case 2: One reactor stage. Mixing, heat transfer area and cooling media are temperature optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]$  °C and  $\sigma = 1.46$  m<sup>3</sup> s/kg.

A corresponding case (Case 17), i.e. optimizing  $u_M$ ,  $u_H$  and  $u_T$ , with three reactor stages was constructed. The design functions,  $u_H$  and  $u_T$ , and the design variables,  $x_{H_2}/x_{CO}$  and  $\sigma$ , were optimized within the same range and set to the same values as for the one-stage case. I.e.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]$  °C,  $x_{H_2}/x_{CO} = 2.1$  and  $\sigma = 1.46$  m<sup>3</sup> s/kg. The resulting design is depicted in Figure 7.3. The optimization yielded an  $u_H$  equal to 21 in the first stage and 5 in the second and third stages, which corresponds to an area



density of  $20.2$  and  $4.8 \text{ m}^2/\text{m}^3$ , respectively (if  $U$  is assumed to be  $1.5 \text{ kJ m}^2/(\text{s K})$ ). The optimal  $T_W$  was found to be  $215^\circ\text{C}$ . The size of the three reactor stages was found to be  $0.38$ ,  $0.14$  and  $0.48\%$  of the total volume, respectively. The design resulted in a  $X_{\text{CO}}$  equal to  $95.99\%$ , a  $J$  equal to  $-0.0899$  and a  $\mathcal{P}_{\text{C}_{11+}}$  equal to  $8.99\%$ .



**Figure 7.3:** Case 17: Three reactor stages. Mixing, heat transfer area and cooling media temperature are optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]^\circ\text{C}$  and  $\sigma = 1.46 \text{ m}^3 \text{ s/kg}$ .

To find the optimal  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio in the feed a total of seven cases, with both one and three reactor stages, were constructed. In these simulations the  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio in the fresh feed was set to  $0.7$  and additional  $\text{H}_2$  was fed both between the reactor stages and prior to the first stage (see Figure 3.1, p. 13).  $u_H$  was allowed to vary within  $5$  to  $30$  and  $T_W$  within  $195$  to  $215^\circ\text{C}$ . The cases of interest are Cases  $5$  through  $7$  and  $20$  through  $23$ . The main results from these cases are both presented in Table 7.1 and depicted in Figures 7.4 through 7.10.

The results show fairly similar optimal  $x_{\text{H}_2}/x_{\text{CO}}$ -ratios for the cases with the same number of stages, i.e. approximately  $2.0$  and  $1.5$  for the one-stage and three-stage cases, respectively. Case  $23$  is not considered in this particular analysis, due to lack of separation from the reactor path.

Cases 5 through 7 and 20 through 22 were also utilized to investigate the effect  $\sigma$  has on the process. As shown in Table 7.1 the  $\mathcal{P}_{C_{11+}}$  decreases from 11.29 to 10.64% for the one-stage cases when  $\sigma$  is lowered from 1.46 to 0.8 m<sup>3</sup> s/kg. In the corresponding three-stage cases the production decreases from 13.08 to 12.75%.

In Case 23 all simulation parameters were kept similar as in Case 22, except for the separation which was not included. Thus, all products are kept inside the reactor-path. The resulting design is depicted in Figure 7.10. The optimization yielded an  $u_H$  equal to 18.3, 21.3 and 7.2 in the three respective reactor stages, which corresponds to an area density of 32.0, 37.2 and 12.6 m<sup>2</sup>/m<sup>3</sup>, respectively (if  $U$  is assumed to be 1.5 kJ m<sup>2</sup>/(sK)). The optimal  $T_W$  was found to be 215 °C. The size of the three reactor stages was found to be 0.29, 0.31 and 0.40% of the total reactor volume, respectively. The design resulted in a  $X_{CO}$  equal to 91.58%, a  $J$  equal to -0.1068 and a  $\mathcal{P}_{C_{11+}}$  equal to 11.44%.

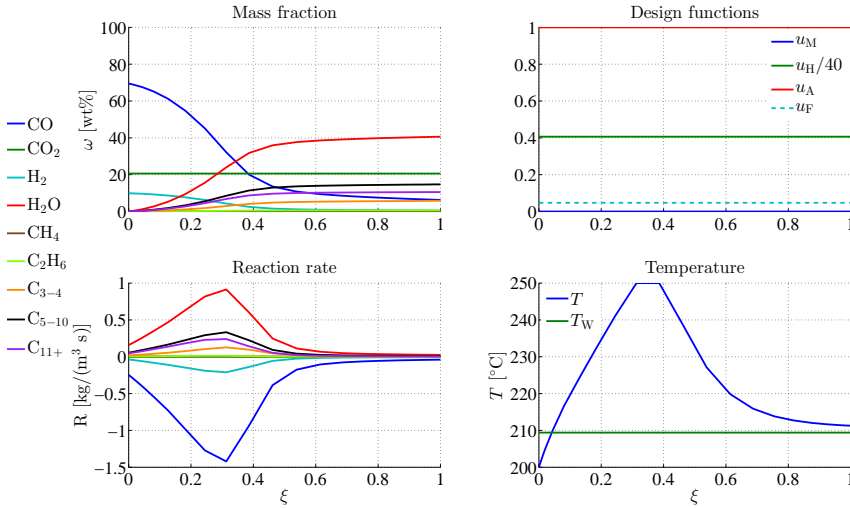
In all the performed cases the optimal mixing properties were found to be plug-flow. Three simulations were performed, Cases 10 through 12, where the mixing properties initially were set to CSTR. These cases produced a PFR solution as well (see Table A.2, p. A-3).

**Table 7.1:** Main results for one-stage Cases 5 through 7 and three-stage Cases 20 through 22.

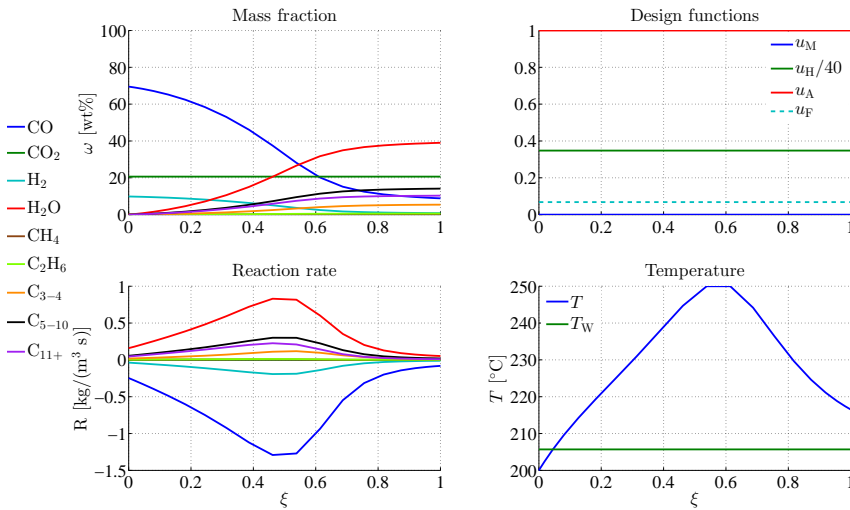
Case	$\sigma$ [m <sup>3</sup> s/kg]	$\frac{x_{\text{H}_2}}{x_{\text{CO}}}$	$X_{\text{CO}}$ [%]	$J$	$\gamma$	$\mathcal{P}_{C_{1+}}$ [%]	$\Delta\xi^\dagger$	$T_w$ [°C]	$a^{*,\dagger}$ [m <sup>2</sup> /m <sup>3</sup> ]	$\alpha^\dagger$ [kg/(m <sup>3</sup> s)]
5	1.46	2.00	90.34	-0.1056	1.07	11.29	1.00	209.4	15.6	0.033
6	1.00	1.99	86.41	-0.1030	1.07	11.01	1.00	205.7	19.5	0.068
7	0.80	1.97	82.11	-0.0997	1.07	10.64	1.00	205.4	24.1	0.105
20	1.46	1.41	97.87	-0.1308	0.47	13.08	0.40/0.34/0.26	211.2	28.9/22.7/8.6	0.018/0.011/0.006
21	1.00	1.47	97.36	-0.1289	0.46	12.89	0.43/0.31/0.26	215.0	39.9/30.0/15.6	0.041/0.023/0.010
22	0.80	1.52	97.08	-0.1275	0.45	12.75	0.43/0.30/0.27	215.0	39.7/37.1/20.3	0.068/0.034/0.013
23	0.80	1.32	91.58	-0.1068	1.07	11.44	0.29/0.31/0.40	215.0	32.0/37.2/12.6	0.051/0.051/0.010

\*  $a$  is calculated with  $U = 1.5 \text{ kJ}/(\text{s m}^2 \text{ K})$  and  $C_p = 2.1 \text{ kJ}/\text{kg}$ .

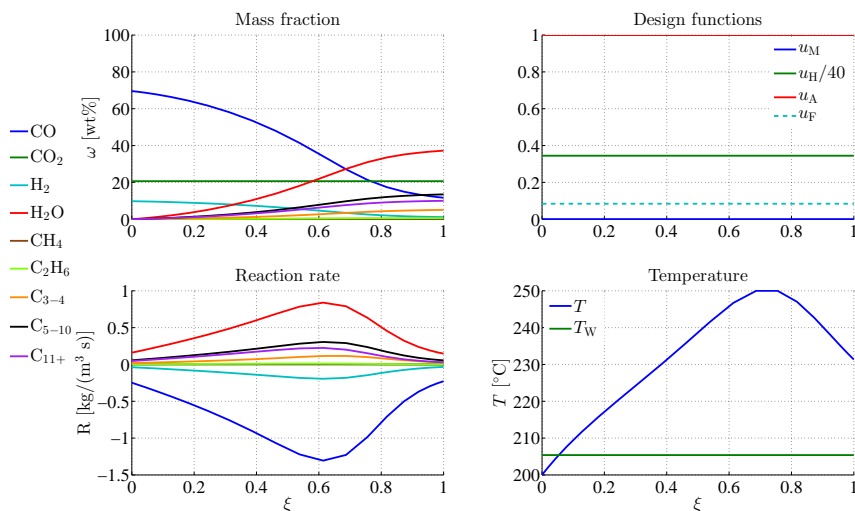
† For the cases with three reactor stages the “/” denotes a stage border.



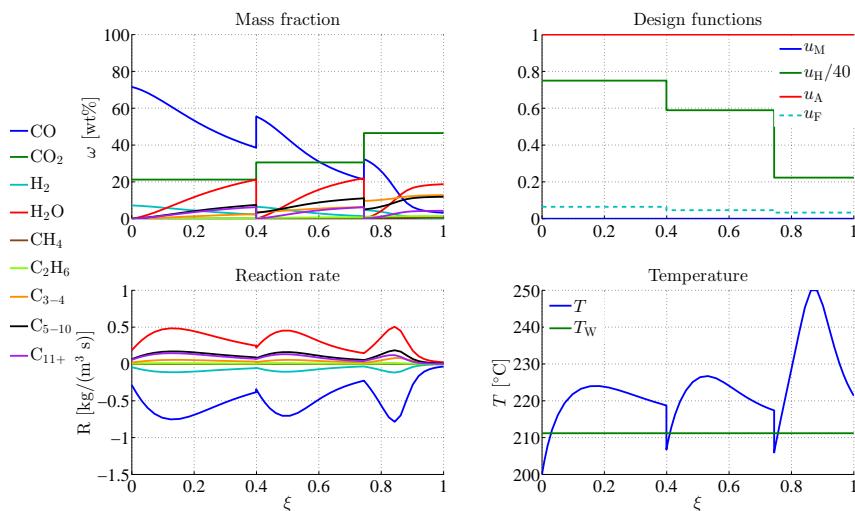
**Figure 7.4:** Case 5: One reactor stage. Mixing, heat transfer area, cooling media temperature and additional feed are optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]^\circ\text{C}$ ,  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio in fresh feed is set to 0.7 and  $\sigma = 1.46 \text{ m}^3 \text{ s/kg}$ .



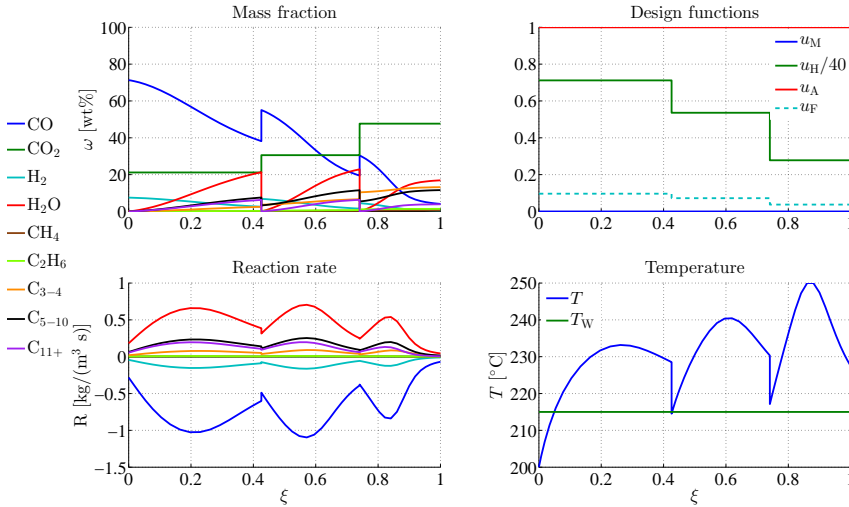
**Figure 7.5:** Case 6: One reactor stage. Mixing, heat transfer area, cooling media temperature and additional feed are optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]^\circ\text{C}$ ,  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio in fresh feed is set to 0.7 and  $\sigma = 1 \text{ m}^3 \text{ s/kg}$ .



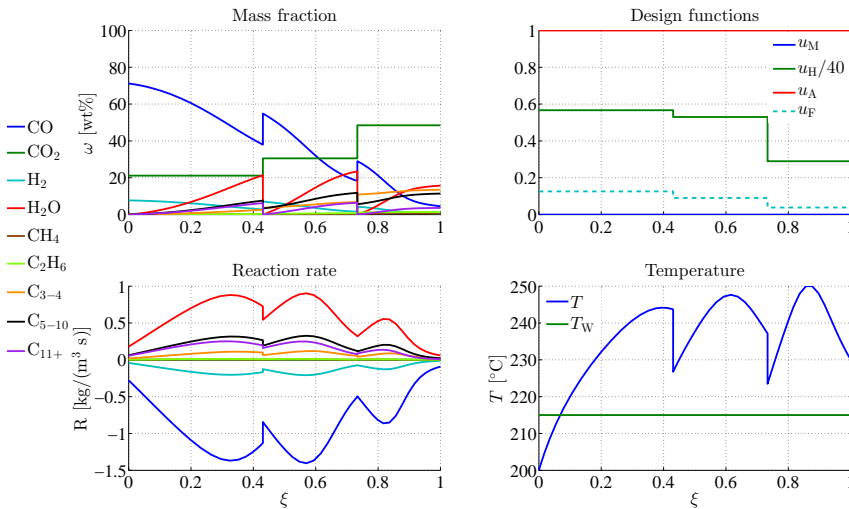
**Figure 7.6:** Case 7: One reactor stage. Mixing, heat transfer area, cooling media temperature and additional feed are optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]$ °C,  $x_{H_2}/x_{CO}$ -ratio in fresh feed is set to 0.7 and  $\sigma = 0.8$  m<sup>3</sup> s/kg.



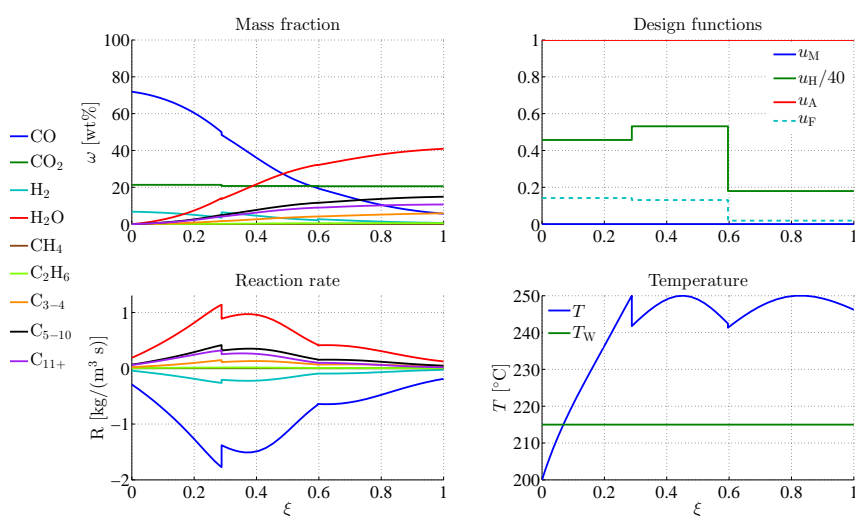
**Figure 7.7:** Case 20: Three reactor stages with separation. Mixing, heat transfer area, cooling media temperature and additional feed are optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]$ °C,  $x_S = [0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0.7 \ 1]$ ,  $x_{H_2}/x_{CO}$ -ratio in fresh feed is set to 0.7 and  $\sigma = 1.46$  m<sup>3</sup> s/kg.



**Figure 7.8:** Case 21: Three reactor stages with separation. Mixing, heat transfer area, cooling media temperature and additional feed are optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]^\circ\text{C}$ ,  $x_S = [0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0.7 \ 1]$ ,  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio in fresh feed is set to 0.7 and  $\sigma = 1 \text{ m}^3 \text{ s/kg}$ .



**Figure 7.9:** Case 22: Three reactor stages with separation. Mixing, heat transfer area, cooling media temperature and additional feed are optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]^\circ\text{C}$ ,  $x_S = [0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0.7 \ 1]$ ,  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio in fresh feed is set to 0.7 and  $\sigma = 0.8 \text{ m}^3 \text{ s/kg}$ .



**Figure 7.10:** Case 23: Three reactor stages. Mixing, heat transfer area, cooling media temperature and additional feed are optimized.  $u_H \in [5, 30]$ ,  $T_W \in [195, 215]^\circ\text{C}$ ,  $x_{\text{H}_2}/x_{\text{CO}}$ -ratio in fresh feed is set to 0.7 and  $\sigma = 0.8 \text{ m}^3/\text{kg}$ .





## 8 Discussion

The implemented FT-kinetics presented by Todic et al. have not been previously utilized with the applied method of systematic reactor staging. The model calculates individual reaction rates for the different  $n$ -paraffins and 1-olefins, thus there is no need to define a paraffin-to-olefin ratio, which has been the case in earlier models. Though this property is an improvement, it can not be fully utilized in view of the component lumping introduced to the system. The lumping only accounts for the number of carbon atoms in the chemical species, but does not distinguish between paraffins and olefins. However, the simplification was necessary as it lowered the number of reacting species from 33 to 9, thus decreasing the total number of optimization variables. As a consequence, the run-time of the simulations was dramatically reduced. Still, with the given number of variables, some of the simulations went on for several days before a solution was obtained. If the individual chemical species, with their respective rates and weight fractions, are to be assessed more powerful computational resources are required.

In both the kinetic model by Todic et al., and the performed simulations, hydrocarbons from  $C_1$  to  $C_{15}$  have been investigated. The kinetic model was able to predict all of the major product distribution characteristics for the given hydrocarbon range, and it should be valid for extrapolation. If a larger range of hydrocarbons is to be investigated, e.g. up to and including  $C_{30}$ , the demand for computational power will increase even further.

The reaction rates are dependent on the partial pressures of  $H_2$ , CO and  $H_2O$ . In the implementation of the kinetics a rough assumption was made that a sharp split between gas and liquid phases at carbon number five exists, i.e. components with carbon number five and greater are assumed to be in the liquid phase and do not affect the partial pressures. If some of the heavier hydrocarbons reside in the gas phase the partial pressure of the reactants would decrease, and consequently the reaction rates would decrease.

Two different cost functions have been applied in the simulation, i.e. maximization of the weight fraction of key component at the end of the reactor path and maximization of the production of key component. The former is utilized for cases without separation, while the latter is utilized when separation is included. As they can not be readily compared, the production of key component is employed as a basis of performance for, and as a

variable for comparison between, the different designs.

Throughout the simulations, the reactor design is assumed to be a slurry bed, mainly due to the low catalyst bulk density which was set to  $200\text{kg/m}^3$ . All simulations yielded an ideal plug-flow mixing design. This result implies sufficient heat transfer area and that there is no excessive heat generation in the reactors. In a real-life slurry bed reactor the mixing would resemble the mixing in a CSTR due to the back mixing of the slurry phase. However, it has been shown that properties close to plug-flow properties can be obtained by introduction of intermediate baffles. If a PFR is to form the basis for the simulations the catalyst bulk density should be much higher, i.e. in the size-range  $1000$  to  $1500\text{kg/m}^3$ . This would result in a much more reactive reactor and the optimal mixing pattern would most probably change to counteract the excessive heat generation.

Another challenge when working with slurry beds is the separation of products. In the simulations the separation was regarded as a “black box” where the product stream is cooled, liquid products are separated and the stream is reheated before entering the next reactor stage. This is a valid assumption if for example a multi-tubular reactor is applied. In a slurry bed the separation will be more complex. The heaviest products will reside in the liquid phase in both reactor systems. In a multi-tubular reactor the liquids trickle down and out of the bed, while in a slurry bed a separation between the liquids and the catalyst particles is required. The lighter hydrocarbons and  $\text{H}_2\text{O}$  follow the gas stream, together with unreacted reactants and inerts, and are separated by condensation in the same way as described in the “black box”.

The separation was assumed to be perfect for both the  $\text{C}_{11+}$ -lump and  $\text{H}_2\text{O}$ , while it was set to 0.7 for the  $\text{C}_{5-10}$ -lump, i.e. 70 wt% of these products are withdrawn from the gas stream before it enters the next reactor stage. This assumption is based on several calculations in UniSim Design R400. The molar fraction of chemical species in a lump will have a substantial effect on the fraction which resides in the liquid phase, especially for the  $\text{C}_{5-10}$ -lump at the given temperature and pressure. This is the reason why the lump is not completely separated.

All simulations, where separation were considered (Figures 7.7 - 7.9, p. 41 - 42), exhibit a pronounced temperature drop between the reactor stages. The drop is due to the addition of “cold”  $\text{H}_2$  with a temperature of  $200^\circ\text{C}$  and the “hard-coded”  $10^\circ\text{C}$  decrease

in temperature caused by the separation process. In the separation “box” in Figure 3.1 (p. 13) the product stream is cooled to 40°C, the liquids are separated and the stream is reheated before entering the next reactor stage. In a real-life design the process would be heat integrated. However some energy will be lost, hence the 10°C temperature decrease.

In the designs where separation is included the weight fraction of the inert, CO<sub>2</sub>, increases throughout the reactor path towards 50 wt% at the end. If additional stages are introduced and more products are separated, the fraction would increase even further, causing decreased partial pressures of H<sub>2</sub> and CO, and lower reaction rates.

The initial simulation, Case 1, with a single reactor stage, yielded a production of key component equal to 8.3%. All design variables, except for the mixing properties, were set. The simple design was retained in Case 2, where the heat transfer area density and the cooling media temperature were optimized as well. The design resulted in a 6.6% increase in production, yielding a production of key component equal to 8.8%. Thus, the performance was improved when several design variables were introduced to the optimization.

By adding two reactor stages, Case 17, the production was increased by 1.8% compared to Case 2. All three designs have a molar feed ratio set equal to 2.1. Case 2 and 17 have similar optimization boundaries for heat transfer area density and cooling media temperature. The latter was found equal to 215°C in both cases, which is at the upper boundary ensuring as high a temperature as possible in the reactor. Continued reaction in the two last reactor stages in Case 17 is achieved by decreasing the heat transfer area. It is adjusted to ensure high temperature simultaneously as the maximum temperature constraint of 250°C is not violated (see Figure 7.2, p. 36 and Figure 7.3, p. 37). In the three-stage case the second and third stages are identical, i.e. in practice the design consists of two stages where the first and second stage account for 38 and 62% of the total reactor volume, respectively. The minor increase in production might not justify the more complex design when economical aspects are considered.

In both cases discussed above the molar feed ratio was set to 2.1. To investigate the optimal feed ratio seven cases were constructed where an extra H<sub>2</sub>-feed was introduced prior to the first reactor stage, as depicted in Figure 3.1 (p. 13). Of the seven cases

there were three one-stage and four three-stage designs. As expected, the results show fairly similar optimal feed ratios for the cases with the same number of stages. All the one-stage designs, Cases 5 through 7, yielded an optimal feed ratio of approximately 2.0, which is close to the stoichiometric ratio 2.1. For three of the three-stage designs, Cases 20 through 22, where products and  $H_2O$  are separated between each stage, the optimal feed ratio was found to be approximately 1.5. The lower feed ratio, compared to the one-stage designs, is due to the supply of additional  $H_2$  between the subsequent stages. The distribution of additional  $H_2$  to the reactor path showed a 45.5% increase in production of key component, when comparing Cases 17 and 20.

The last case which included optimization of the feed ratio was Case 23 (see Figure 7.10, p. 43). In this case no products were separated from the reactor path. The optimal feed ratio was found to be 1.32 which is the lowest of the seven cases. The low feed ratio may be caused by the lack of separation. Compared to Case 22, Case 23 has a lower heat transfer area in the first stage, which results in a steeper temperature profile and higher reaction rates. Case 23 is the only case where the heat transfer area is larger in the second stage compared to the first stage. More  $H_2$  is also added between the first and second reactor stages, compared to case 22.

Cases 5 though 7 and 20 through 23 were used as a basis to investigate the dependency of  $\sigma$  as well. From Table 7.1 (p. 39) it can be seen that the production of  $C_{11+}$  decreases by 5.8% for the one-stage cases when  $\sigma$  is reduced from 1.46 to 0.8 m<sup>3</sup> s/kg. For the corresponding three-stage cases the production decreases by 2.5%. The minor reduction in production when the reactor volume is almost halved may suggest that  $\sigma$  equal to 0.8 m<sup>3</sup> s/kg is a better design from an economical point of view, when investment and operating costs are considered. In future work the optimal value of  $\sigma$  may be investigated by use of a cost function that encompasses, and try to minimize,  $\sigma$ . In addition, the cost function must include a constraint on the conversion of CO per pass to prevent the total reactor volume to approach zero.

By increasing the number of stages from one to three the production of  $C_{11+}$  increases by 15.9, 17.2 and 19.8% when  $\sigma$  is equal to 1.46, 1.00 and 0.80 m<sup>3</sup> s/kg, respectively. The largest increase is found in the case where  $\sigma$  is 0.8 m<sup>3</sup> s/kg. The large increase in production, when comparing the three- and one-stage designs, can justify the construction of one of the more complex three-stage designs.

As discussed in the previous paragraph, the largest increase in production was found in Case 22 (see Figure 7.9, p. 42), when comparing the one and three-stage designs. This case, which includes separation between the reactor stages, yielded a production of key component equal to 12.8%. To investigate the benefits of separation Case 23 was constructed. Here, all simulation variables were kept alike, apart from separation. By retaining all the products within the reactor path, Case 23 yielded a production of 11.4%, i.e. a decrease of 10.3% compared to Case 22. The minor reduction suggests that the economical aspects of including separation should be investigated in more detail before a distinct conclusion can be drawn. However, it should be noted that in a slurry bed reactor, separation of the heaviest components from the slurry must be made. Therefore, it is the separation between the stages that is of interest in future investigations.

All simulations yielded an optimal cooling water temperature in the range 205 to 215 °C. To be able to have vaporizing water at this temperature the cooling water utility system has to be at a pressure of approximately 20 bar. This is equivalent to the pressure in the reactor, and no complications associated with pressure difference between the reactor and the utility system are expected.

A recycle is not included in the reactor path, so a high conversion of CO per pass is desirable. All simulations yielded a conversion in the range of 82.1 to 97.7%, suggesting that the requirement of a recycle is not paramount.

As stated in Section 6.1 (p. 31) the molecular weights and heat capacities of the lumps are calculated with the assumption that the lumps are composed of equimolar amounts of the respective chemical species. In addition, the molecular weights are rounded off to integers and the heat capacities are calculated at feed temperature and pressure. These assumptions are rather rough and the real values will differ. Though not investigated, these assumptions are not expected to have a significant effect on the end results.

The feed temperature has been 200 °C in all simulations. In future work the temperature should be investigated to find the optimal feed temperature. The same applies to the total pressure which has been kept constant for all the performed simulations. The kinetic model is validated in the pressure range 15 to 25 bar. In the performed simulations a total pressure of 20 bar is chosen, which is well within the bounds. Further, the total pressure is assumed constant throughout the reactor path. In reality there would be a

pressure drop. The size of this drop is dependent on the reactor system. In a close-packed multi-tubular reactor the pressure drop will be larger compared to a slurry bed reactor. If the pressure drop is implemented in the model it may cause reduced reaction rates, decreased conversion of CO and thus, lower production of key component per pass.

## 9 Conclusions

The objective of this Master's thesis has been to find the optimal reactor path layout for the Fischer-Tropsch process, i.e. the number of reactor stages, the desired cooling media temperature, the heat transfer area density, the molar feed ratio and the inclusion or emission of separation. As expected, the performed simulations show an increased production of key component when the degrees of freedom are increased, i.e. several optimization variables yield increased performance.

A close to halved total reactor volume showed only a small decrease in production of key component for cases with both one and three reactor stages. Thus a space-time equal to  $0.8\text{m}^3\text{s/kg}$  may be better from an economical point of view, compared to higher values for  $\sigma$ .

Increasing the number of reactor stages from one to three, including separation, yielded a 19.8% increase in production of key component (for  $\sigma = 0.8\text{m}^3\text{s/kg}$ ). The large increase in production may justify the construction of the more complex three-stage design.

Introduction of additional  $\text{H}_2$  yielded an optimal molar feed ratio,  $x_{\text{H}_2}/x_{\text{CO}}$ , of approximately 2.0 for the one-stage designs. For the three-stage designs the ratio was found to be approximately 1.5 and 1.3, for the cases with and without separation, respectively. A 45.5% increase in production of key component was found when two three-staged designs, with and without optimization of the feed ratio, were compared.

The optimal design was found to have three reactor stages, separation, a  $\sigma$  equal to  $0.8\text{m}^3\text{s/kg}$  and a molar feed ratio,  $x_{\text{H}_2}/x_{\text{CO}}$ , equal to 1.52. The design showed a production of key component equal to 11.4%.

A three-staged design without separation showed a 10.3% reduction in production compared to a corresponding case with separation. Thus, economical calculations should be considered before drawing conclusions with regard to construction of a plant with or without separation.

The simulations yield a conceptual, optimal design. The real-life feasibility of the design is still unknown as no economical aspects have been taken into consideration.





# References

- [1] B. Todic, T. Bhatelia, G. F. Froment, W. Ma, G. Jacobs, B. H. Davis, and D. B. Bukur, “Kinetic Model of Fischer-Tropsch Synthesis in a Slurry Reactor on CO–Re/Al<sub>2</sub>O<sub>3</sub> Catalyst,” Industrial & Engineering Chemistry Research, vol. 52, pp. 669–679, 2013.
- [2] British Petroleum, “BP Statistical Review of World Energy June 2012,” 2012.
- [3] M. E. Dry, “The Fischer-Tropsch (FT) Synthesis Process,” in Handbook of Heterogeneous Catalysis, ch. 13.15, pp. 2965–2994, Wiley-VCH Verlag, 2008.
- [4] C. Masters, “The Fischer-Tropsch Reaction,” Advances in Organometallic Chemistry, vol. 17, pp. 61–103, 1979.
- [5] H. Schulz, “Short history and present trends of Fischer-Tropsch synthesis,” Applied Catalysis A: General, vol. 186, pp. 3–12, 1999.
- [6] T. J. Remans, G. Jenzer, and A. Hoek, “Gas-to-Liquids,” in Handbook of Heterogeneous Catalysis, ch. 13.16, pp. 2994–3010, Wiley-VCH Verlag, 2008.
- [7] M. E. Dry, “The Fischer-Tropsch process: 1950-2000,” Catalysis Today, vol. 71, no. 3-4, pp. 227–241, 2002.
- [8] J. A. Moulijn, M. Makkee, and A. Van Diepen, Chemical Process Technology. Chichester: John Wiley & Sons Ltd, 2001.
- [9] K. R. Radtke, M. Heinritz-Adrian, and C. Marsico, “New Wave of Coal-to-Liquids,” International Journal for Electricity and Heat Generation, vol. 86, no. 5, pp. 78–84, 2006.
- [10] H. A. Jakobsen, Chemical Reactor Modeling. Springer, 2008.
- [11] C. Maretto and R. Krishna, “Design and optimization of a multi-stage bubble column slurry reactor for Fischer-Tropsch synthesis,” Catalysis Today, vol. 66, pp. 241–248, 2001.
- [12] S. C. Saxena, “Bubble Column Reactors and Fischer-Tropsch Synthesis,” Catalysis Reviews: Science and Engineering, vol. 37, no. 2, pp. 227–309, 1995.

- [13] C. Maretto and R. Krishna, "Modeling of a bubble column slurry reactor for Fischer-Tropsch synthesis," Catalysis Today, vol. 52, pp. 279–289, 1999.
- [14] M. Hillestad, "Systematic staging in chemical reactor design," Chemical Engineering Science, vol. 65, pp. 3301–3312, 2010.
- [15] S. Skogestad, "Appendix F - Data," in Prosessteknikk, pp. 349–356, Tapir Akademisk Forlag, 2 ed., 2003.
- [16] I. Chorkendorff and J. W. Niemantsverdriet, Concepts of Modern Catalysis and Kinetics. Weinheim: WILEY-VCH, 2007.
- [17] H. S. Fogler, Elements of Chemical Reaction Engineering. New Jersey: Person Educations, Inc, 2009.

# **A Results data**

In this appendix raw-data and processed data for all the simulations is presented.

The main results are presented in Table A.1. Raw data for the design functions are presented in Table A.2.

Table A.1: Main results for all simulations.

Case	$n_s$	$\sigma$ [m <sup>3</sup> s/kg]	$\frac{x_{H_2}}{x_{CO}}$	$X_{CO}$ [%]	$J$	$\gamma$	$\mathcal{P}_{C_{1+}}$ [%]	$\Delta\xi^*$	$T_w$ [°C]	$a^{*,\S}$ [m <sup>2</sup> /m <sup>3</sup> ]	$\alpha^*$ [kg/(m <sup>3</sup> s)]
1	1	1.46	2.10 <sup>†</sup>	82.32	-0.0828	1.00	8.28	1.00	205.0	9.6	
2	1	1.46	2.10 <sup>†</sup>	92.59	-0.0883	1.00	8.83	1.00	215.0	12.7	
3	1	1.00	2.10 <sup>†</sup>	88.82	-0.0863	1.00	8.63	1.00	215.0	18.6	
4	1	0.80	2.10 <sup>†</sup>	86.50	-0.0847	1.00	8.47	1.00	213.8	21.8	
5	1	1.46	2.00 <sup>‡</sup>	90.34	-0.1056	1.07	11.29	1.00	209.4	15.6	0.033
6	1	1.00	1.99 <sup>‡</sup>	86.41	-0.1030	1.07	11.01	1.00	205.7	19.5	0.068
7	1	0.80	1.97 <sup>‡</sup>	82.11	-0.0997	1.07	10.64	1.00	205.4	24.1	0.105
8	2	1.46	2.10 <sup>†</sup>	82.32	-0.0828	1.00	8.28	0.50/0.50	205	9.6/9.6	
9	2	1.46	2.10 <sup>†</sup>	94.94	-0.0894	1.00	8.94	0.37/0.63	205	9.5/4.8	
10	2	1.46	2.10 <sup>†</sup>	96.01	-0.0898	1.00	8.98	0.40/0.60	208.0/215.0	11.5/4.8	
11	2	1.46	2.10 <sup>†</sup>	96.01	-0.0898	1.00	8.98	0.40/0.60	208.0/215.0	11.5/4.8	
12	2	1.46	2.10 <sup>†</sup>	96.08	-0.0899	1.00	8.99	0.37/0.63	215.0	19.2/4.8	
13	2	1.46	2.10 <sup>†</sup>	96.01	-0.0899	1.00	8.99	0.37/0.63	215.0	20.1/4.8	
14	2	1.46	1.57 <sup>†</sup>	96.24	-0.0980	1.02	10.01	0.35/0.65	215.0	19.2/4.8	0.000/0.010
15	3	1.46	1.20 <sup>†</sup>	95.51	-0.1006	1.04	10.48	0.44/0.27/0.29	215.0	20.2/4.8/4.8	0.000/0.015/0.004
16	3	1.46	1.25 <sup>†</sup>	95.34	-0.1005	1.04	10.42	0.44/0.32/0.24	215.0	20.2/4.8/4.8	0.000/0.017/0.001
17	3	1.46	2.10 <sup>†</sup>	95.99	-0.0899	1.00	8.99	0.38/0.14/0.48	215.0	20.2/4.8/4.8	
18	3	1.46	1.25 <sup>†</sup>	97.65	-0.1172	0.52	11.72	0.35/0.37/0.28	211.4	27.3/20.1/6.9	0.000/0.013/0.006
19	3	1.46	1.25 <sup>†</sup>	97.47	-0.1172	0.50	11.72	0.38/0.36/0.26	211.1	28.9/18.8/7.1	0.000/0.013/0.006
20	3	1.46	1.41 <sup>‡</sup>	97.87	-0.1308	0.47	13.08	0.40/0.34/0.26	211.2	28.9/22.7/8.6	0.018/0.011/0.006
21	3	1.00	1.47 <sup>‡</sup>	97.36	-0.1289	0.46	12.89	0.43/0.31/0.26	215.0	39.9/30.0/15.6	0.041/0.023/0.010
22	3	0.80	1.52 <sup>‡</sup>	97.08	-0.1275	0.45	12.75	0.43/0.30/0.27	215.0	39.7/37.1/20.3	0.068/0.034/0.013
23	3	0.80	1.32 <sup>‡</sup>	91.58	-0.1068	1.07	11.44	0.29/0.31/0.40	215.0	32.0/37.2/12.6	0.051/0.051/0.010

\* For the cases with two or three reactor stages the “/” denotes a stage border.

† Set

‡ Optimized

§  $a$  is calculated with  $U = 1.5 \text{ kJ}/(\text{s m}^2 \text{ K})$  and  $C_p = 2.1 \text{ kJ}/\text{kg}$ .

**Table A.2:** Raw data for the design functions.

Case	$u_M^*$	$u_T^\dagger$			$u_H^\dagger$			$u_F^\dagger$		
		1	2	3	1	2	3	1	2	3
1	P	0.0106			10.0					
2	P	0.0317			13.2					
3	P	0.0317			13.3					
4	P	0.0291			12.4					
5	P	0.0199			16.3			0.0473		
6	P	0.0120			13.9			0.0681		
7	P	0.0114			13.8			0.0840		
8	P-P	0.0106	0.0106		10.0	10.0				
9	P-P	0.0106	0.0106		9.9	5.0				
10	P-P	0.0170	0.0317		12.0	5.0				
11	P-P	0.0170	0.0170		12.0	5.0				
12	P-P	0.0317	0.0317		20.0	5.0				
13	P-P	0.0317	0.0317		21.0	5.0				
14	P-P	0.0317	0.0317		20.0	5.0			0.0226	
15	P-P-P	0.0317	0.0317	0.0317	21.0	5.0	5.0		0.0811	0.0214
16	P-P-P	0.0317	0.0317	0.0317	21.0	5.0	5.0		0.0760	0.0065
17	P-P-P	0.0317	0.0317	0.0317	21.0	5.0	5.0			
18	P-P-P	0.0241	0.0241	0.0241	28.4	20.9	7.1			
19	P-P-P	0.0235	0.0235	0.0235	30.0	19.6	7.4			
20	P-P-P	0.0237	0.0237	0.0237	30.0	23.6	8.9		0.0506	0.0312
21	P-P-P	0.0317	0.0317	0.0317	28.5	21.4	11.1		0.5290	0.0325
22	P-P-P	0.0317	0.0317	0.0317	22.7	21.2	11.6		0.0647	0.0332
23	P-P-P	0.0317	0.0317	0.0317	18.8	21.3	7.2		0.0964	0.0372
									0.1257	0.0905
									0.1420	0.0188

\* P = plug flow

† 1, 2 and 3 denotes the reactor stages.



## B Kinetic Model – Additional Data

In this appendix data for the kinetic model is presented.

**Table B.1:** Values for the parameters in the kinetic model.

Parameter	Value	Unit
$A_1$	$1.83 \times 10^{10}$	mol/(g <sub>cat</sub> h MPa)
$A_2$	5.08	-
$A_3$	$2.44 \times 10^1$	MPa <sup>-1</sup>
$A_4$	2.90	-
$A_5$	$4.49 \times 10^5$	mol/(g <sub>cat</sub> h MPa)
$A_{5M}$	$8.43 \times 10^5$	mol/(g <sub>cat</sub> h MPa)
$A_6$	$7.47 \times 10^8$	mol/(g <sub>cat</sub> h)
$A_{6E}$	$7.03 \times 10^8$	mol/(g <sub>cat</sub> h)
$A_7$	$1.00 \times 10^{-3}$	MPa <sup>-1</sup>
$E_1$	100.4	kJ/mol
$E_5$	72.4	kJ/mol
$E_{5M}$	63.0	kJ/mol
$E_6^0$	97.2	kJ/mol
$E_{6E}^0$	108.8	kJ/mol
$\Delta E$	1.12	kJ/mol <sub>CH<sub>2</sub></sub>
$\Delta H_2$	8.68	kJ/mol
$\Delta H_3$	9.44	kJ/mol
$\Delta H_4$	7.9	kJ/mol
$\Delta H_7$	-25.0	kJ/mol





# C Matlab Code

In this Appendix the Matlab scripts used in the thesis are presented. A short description of each of the scripts is presented in Table C.1. In addition, three simulation hierarchies are presented in Tables C.2 through C.4.

**Table C.1:** Description of the different MATLAB scripts.

File name	Description
fischerTropschTodicLump.m	Main file
fluidpath.m	System definition
integrate.m	Integration over reactor volume
OptimizeDesign.m	Optimization function
DesignModel.m	ODE describing the change of states along the reactor path
kinTodicLump.m	Kinetic model
ObjectiveFunction.m	Objective function (that is minimized)
ModelConstraints.m	Model Constraints
flowPropTPlump.m	Extracting flow properties from pre-stored matrices
moletomass.m	Converting from mole to mass fraction
masstomole.m	Converting from mass to mole fraction
pathset.m	Set objects in "fluidpath.m"
AssignVector.m	Store values
AssignObject.m	Store values
Jacobian.m	Calculation of the Jacobian
intMolarMass.m	Calculating molecular weight in whole numbers
deltaHrx.m	Calculating the heat of reaction
mid.m	Calculation of the mean over the reactor path
AssignRate.m	Calculation of reaction rates
pathget.m	Get properties from "fluidpath.m"
MmCpMatrixLump.m	Main file; calculating flow properties
uniSimPropertiesTodicLump.m	Extracting flow properties from UniSim Design R400
FTplotting.m	Main file for plotting results
plotFT.m	Constructing plots
defaultPlotSettings.m	Defining default plot settings

**Table C.2:** Simulation hierarchy for retrieving physical properties.

---

<b>Level</b>		
<b>1</b>	<b>2</b>	<b>3</b>
MmCpMatrixLump.m	uniSimPropertiesTodicLump.m	FlowPropertiesTodicLump.usc*

---

\* UniSim Design R400 file.

**Table C.3:** Simulation hierarchy for plotting the results.

---

<b>Level</b>		
<b>1</b>	<b>2</b>	<b>3</b>
FTplotting.m	plotFT.m	defaultPlotSettings.m

---

Table C.4: Main simulation hierarchy.

	Level						
1	2	3	4	5	6	7	
	flowPropTPump.m moletoMass.m massToMole.m fluidPath.m pathSet.m						
	integrate.m	DesignModel.m	Jacobian.m kinTodielLump.m	kinTodielLump.m massToMole.m heltallMolarMass.m deltaHrx.m	kinTodielLump.m	massToMole.m heltallMolarMass.m deltaHrx.m	
		kinTodielLump.m	massToMole.m heltallMolarMass.m deltaHrx.m				
		AssignVector.m					
		ObjectiveFunction.m	AssignObject.m	AssignRate.m	kinTodielLump.m	massToMole.m heltallMolarMass.m deltaHrx.m	
			mid.m	pathGet.m			
			AssignObject.m	AssignRate.m	kinTodielLump.m	massToMole.m heltallMolarMass.m deltaHrx.m	
	OptimizeDesign.m	ModelConstraints.m	DesignModel.m	Jacobian.m	kinTodielLump.m	massToMole.m heltallMolarMass.m deltaHrx.m	
		AssignObject.m	AssignRate.m	kinTodielLump.m	massToMole.m heltallMolarMass.m deltaHrx.m		

fischerTropSchTodielLump.m

**C.1 fischerTropschTodicLump.m**

```

1 %% Fischer-Tropsch, reactor staging
2 clear all
3 close all
4 clc
5
6 %% Local variables
7 c = ...
      str2mat('CO','CO2','H2','H2O','CH4','C2H6','C3-4','C5-10','C11+');
8
9 Tf = 200;                %% Temperature feed [C]
10 Tw = 205;               %% Temperature cooling [C]
11 Tref = 200 + 273;       %% Reference temperature
12 pTot = 2e6;             %% Pressure [Pa]
13 bulkCatDensity = 200e03; %% Catalyst density [g/m^3]
14
15 % [M, cpnom, liqFrac] = uniSimPropertiesTodicLump(Tf,pTot);
16 [M, cpnom, liqFrac] = flowPropTPlump(Tf,pTot);
17 M = round(M);
18 %           1     2     3     4     5     6     7     8     9     10
19 s = strvcat('CO','CO2','H2','H2O','C1','C2','C34','C510','C11','T');
20 for i=1:size(s,1)
21     eval([s(i,:), '=int8(' ,num2str(i), ');']);
22 end
23 %%
24 ns = 1;                  %% Number of stages
25 nc = length(M);         %% Number of components
26 ncol = 20;              %% Number of internal collocation points
27
28 % Mole fractions fresh feed
29 yF0 = zeros(nc,1);
30 yF0(CO2) = 0.10;
31 yF0(CO) = 0.29;
32 yF0(H2) = 0.61;
33
34 Mav = yF0'*M;
35
36 % Mass fractions fresh feed
37 yF0m = moletomass(yF0,M);
38

```

```

39 % Mole fractions extra feed
40 yF = zeros(nc,1);
41 yF(H2) = 1;
42
43 % Mass fractions extra feed
44 yFm = moletomass(yF,M);
45
46 %%
47 % VR = 400;           %% Reactor volume [m3]
48 % F0 = 5000;         %% Feed flow [mol/s]
49 % W0 = F0/1000*Mav; %% Feed flow [kg/s]
50 % sigma = VR/W0;    %% Space time [m3s/kg]
51 sigma = 1.4556;
52 uHi = 20;           %% Heat transfer area, design function
53
54
55 %% Create a object a of class fluidpath
56 a = fluidpath(ns,nc,ncol);
57 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
58 ul = ones(1,2*ns);
59 a = pathset(a, 'T0', Tf);           %% Feed temperature
60 a = pathset(a, 'Tcool0', Tw);      %% Inital cooling water ...
    temperature
61 a = pathset(a, 'Tref', Tref);       %% Reference temperature
62 a = pathset(a, 'pTot', pTot);       %% Total pressure
63 a = pathset(a, 'M', M');            %% Molar mass [kg/kmol]
64 %           TolCon TolFun TolX
65 a = pathset(a, 'tol', [1e-8 1e-8 1e-6 1e-6 1e-6]); %% Tolerances
66 a = pathset(a, 'cp', cpnom);        %% Heat capacities ...
    [kJ/(kg*K)]
67 a = pathset(a, 'liqFrac', liqFrac');
68 a = pathset(a, 'kinetics', 'kinTodicLump'); %% Kinetic function
69 a = pathset(a, 'xx0', [yF0m; (Tf + 273 - Tref)/Tref], 1); %% State vector
70 a = pathset(a, 'xxF', [yFm; (Tf + 273 - Tref)/Tref], 1); %% State vector
71 a = pathset(a, 'xxFF', [yF0m; (Tf + 273 - Tref)/Tref], 1); %% State vector
72 a = pathset(a, 'xS', [0 0 0 1 0 0 0 0.7 1]', 1); %% Separation function
73 a = pathset(a, 'key', C11);         %% Key component
74 a = pathset(a, 'sigma0', sigma);    %% Inital sigma
75
76 %%
77 % Parameterization of design functions
78 a = pathset(a, 'update_sigma', 0);   %% Space time

```

```

79 a = pathset(a, 'update_uH', repmat([1 -1], 1, ns)); %% Heat ...
    exchanger area
80 a = pathset(a, 'update_uT', repmat([1 -1], 1, ns)); %% Temp ...
    cooling media
81 a = pathset(a, 'update_uF', repmat([0 0], 1, ns)); %% Extra feed
82 a = pathset(a, 'update_uFF', repmat([0 0], 1, ns)); %% 2nd extra feed
83 a = pathset(a, 'update_uA', repmat([0 0], 1, ns)); %% Catalyst ...
    activity
84 a = pathset(a, 'update_uS', repmat([0 0], 1, ns)); %% Separation
85 a = pathset(a, 'uFp', repmat(1, 1, ns)); %% If 1: point feed ...
    1st extra feed
86 a = pathset(a, 'uFFp', repmat(1, 1, ns)); %% If 1: point feed ...
    2nd extra feed
87 a = pathset(a, 'uSp', repmat(0, 1, ns));
88
89 %%
90 % Initial values and boundaries of design functions
91 % Space time
92 a = pathset(a, 'sigma', sigma);
93 a = pathset(a, 'sigma_max', 5);
94 a = pathset(a, 'sigma_min', 0);
95
96 % Heat transfer area distribution
97 a = pathset(a, 'uH', u1*uHi/40);
98 a = pathset(a, 'uH_min', u1*5/40);
99 a = pathset(a, 'uH_max', u1*30/40);
100
101 % Temperature cooling media
102 a = pathset(a, 'uT', u1*(Tw+273-Tref)/Tref);
103 a = pathset(a, 'uT_min', u1*(Tw-10+273-Tref)/Tref);
104 a = pathset(a, 'uT_max', u1*(Tw+10+273-Tref)/Tref);
105
106 % Catalyst activity
107 a = pathset(a, 'uA', u1);
108 a = pathset(a, 'uA_min', u1*0.1);
109 a = pathset(a, 'uA_max', u1*1);
110
111 % 1st extra feed
112 a = pathset(a, 'uF', u1*0);
113 a = pathset(a, 'uF_min', u1*0);
114 a = pathset(a, 'uF_max', u1*1);
115

```

```
116 % 2nd extra feed
117 a = pathset(a, 'uFF', u1*0);
118 a = pathset(a, 'uFF_min', u1*0);
119 a = pathset(a, 'uFF_max', u1*10);
120
121 % Separation
122 a = pathset(a, 'uS', u1*0);
123 a = pathset(a, 'uS_min', u1*0);
124 a = pathset(a, 'uS_max', u1*1);
125
126 % Objective function
127 a = pathset(a, 'Jcrit', 1);
128
129 %% Integration
130 tic
131 a = integrate(a);
132 toc
133 % CO conversion
134 y1 = masstomole(a.Z(end,1:nc)', M);
135 X_CO1 = (yF0(CO) - y1(CO))/yF0(CO); %% Molar basis
136 MX_CO1 = 1 - a.Z(end,1)/a.Z(1,1)*a.Z(end,end); %% Weight basis
137
138 %% Optimization
139 [a, b] = OptimizeDesign(a, 5);
140
141 % CO conversion
142 y = masstomole(a.Z(end,1:nc)', M);
143 X_CO = (yF0(CO) - y(CO))/yF0(CO); %% Molar basis
144 MX_CO = 1 - a.Z(end,1)/a.Z(1,1)*a.Z(end,end); %% Weight basis
145 fprintf('CO conversion before optimization = %0.2f\n', MX_CO1)
146 fprintf('CO conversion after optimization = %0.2f\n', MX_CO)
147 plotFT(a);
148
149 save lstage_uM_uH_uT.mat %% Save results
```

**C.2 fluidpath.m**

```

1 function a=fluidpath(ns,nc,ncol)
2 % FLUIDPATH is the constructor of the class FLUIDPATH and
3 % initiates the structure. A FLUIDPATH object defines a
4 % the Process Path Environment.
5 % Input
6 %   ns       : Number of stages
7 %   nc       : Number of components (species)
8 %   ncol     : Number of internal collocation points in each stage.
9 % Output
10 %   a       : An initialized object of class FLUIDPATH
11 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
12 a.ns = ns;           % Number of stages
13 a.nc = nc;           % Number of components
14 a.ncol = ncol;       % Number of internal collocation points
15 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
16 % The other
17 a.T0 = 273+200;      % Temperature of fresh feed
18 a.Tcool0 = 273+200;  % Temperature of cooling media (initial)
19 a.Tref = 273+250;    % Reference temperature [K]
20 a.pTot = 2.0e6;      % Total pressure [Pa]
21 a.kinetics = str2func('kinTodicLump'); % Kinetic function
22 a.fluid = 'micro';
23
24 a.M = zeros(1,nc);   % Molecular weight kg/kmol
25 a.cp = zeros(1,nc);  % Heat capacity kJ/kg-K
26 a.liqFrac = zeros(1,nc); % Liquid fraction, given temperature ...
    and pressure
27
28 a.nxx = nc + 1;      % Length of vector xx
29 a.nz = a.nxx + 1;    % Length of z
30 %
31 % Collocation
32 [r, A, B, q] = colloc( ncol ,1 ,1);
33 a.A = A;             %% First derivative
34 a.roots = r;         %% Collocation points (roots)
35 a.qweights = q;     %% Collocation weights
36 %
37 a.key = nc;          % Default value
38 a.ndof = 0;          % Number of DOF (degrees of freedom)

```



```

39
40 a.sigma0 = 0;           % Space time (inital)
41 a.sigma = 0;           % VRoverF0 is space time [m3*s/kmol]
42 a.sigma_max = 100.;    % VRoverF0 is space time [m3*s/kmol]
43 a.sigma_min = 0;       % VRoverF0 is space time [m3*s/kmol]
44 a.update_sigma = 1;
45
46 a.algorithm = 2;        % Algorithm=0; manual
47 a.xx0 = zeros(a.nxx,1); % Initial state
48 a.xx = a.xx0;          % Current state
49 a.xxF = a.xx0;
50 a.xxFF = a.xx0;        % Composition of the side second stream
51 a.xS = zeros(nc,1);
52
53 a.J(1:20) = 0;
54 a.Jcrit = 1;
55 %a.ineqcon=[];
56 % tolerances for
57 %      TolCon  TolFun  TolX   niu   niu
58 a.tol = [1.e-7  1.e-9  1.e-9  1.e-6  1.e-9];
59
60 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
61 % Initialize design functions
62 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
63 %
64 a.dxi(1:ns) = 1.0/ns;    % Equally distributed regions initially
65 a.update_dxi(1:ns) = 1;
66 for i=1:ns+1, a.xi(i) = sum(a.dxi(1:i-1)); end
67 % M
68 a.uM(1:2*ns) = 0;        % Mixing regions are initially set ...
   to PFR
69 a.update_uM(1:2*ns) = 1;
70 a.update_uM(1) = 0;
71 % H
72 a.uH(1:ns) = 0;         % Heat exchange area distribution
73 a.uH_min(1:ns) = 0;
74 a.uH_max(1:ns) = 0
75 a.update_uH(1:ns) = 1;
76 a.iuH = [];
77 a.luH = 0;
78 a.puH = 0;
79 % F

```

```

80 a.neF = 1;           % Number of external feeds with different composition
81 a.uF(1:2*ns)       = 0.0;    %% Extra feeds are set to zero initailly
82 a.uFp(1:2*ns)     = 0;      %% Default feed is distributed
83 a.uF0(1:2*ns)     = a.uF(1:2*ns);
84 a.uF1(1:2*ns)     = 0.0;
85 a.uF_min(1:2*ns)  = 0.0;    %% Extra feeds are set to zero initailly
86 a.uF_max(1:2*ns)  = 0.0;    %% Extra feeds are set to zero initailly
87 a.update_uF(1:2*ns) = 1;
88 a.iuF              = [];
89 a.luF              = 0;
90 a.puF              = 0;
91
92 % FF second side stream
93 % FF
94 a.neFF=1;          % Number of external feeds with different composition
95 a.uFF(1:2*ns)     = 0.0;    %% Extra feeds are set to zero initailly
96 a.uFFp(1:2*ns)   = 0;      %% Default feed is distributed
97 a.uFF0(1:2*ns)   = a.uF(1:2*ns);
98 a.uFF1(1:2*ns)   = 0.0;
99 a.uFF_min(1:2*ns) = 0.0;    %% Extra feeds are set to zero initailly
100 a.uFF_max(1:2*ns) = 0.0;   %% Extra feeds are set to zero initailly
101 a.update_uFF(1:2*ns) = 1;
102 a.iuFF            = [];
103 a.luFF            = 0;
104 a.puFF            = 0;
105
106 % Separation
107 a.uS(1:2*ns)      = 0.0;    %% Extra feeds are set to zero initailly
108 a.uSp(1:2*ns)    = 0;      %% Default feed is distributed
109 a.uS_min(1:2*ns) = 0.0;    %% Extra feeds are set to zero initailly
110 a.uS_max(1:2*ns) = 1;      %% Extra feeds are set to zero initailly
111 a.update_uS(1:2*ns) = 0;
112 a.iuS            = [];
113 a.luS            = 0;
114 a.puS            = 0;
115
116 % T
117 a.uT(1:ns)       = 0;      %% Cooling/Heating temperature
118 a.uT_min(1:ns)   = 0;      %% Cooling/Heating temperature
119 a.uT_max(1:ns)   = 0;      %% Cooling/Heating temperature
120 a.update_uT(1:ns) = 1;
121 a.iuT            = [];

```

```

122 a.luT           = 0;
123 a.puT           = 0;
124
125 % A
126 a.uA(1:ns)      = 1.0;           %% Catalyst activity over the ...
    entire volume
127 a.uA_min(1:ns) = 0.0;           %% Catalyst activity over the ...
    entire volume
128 a.uA_max(1:ns) = 1.0;           %% Catalyst activity over the ...
    entire volume
129 a.update_uA(1:ns) = 1;
130 a.iuA           = [];
131 a.luA           = 0;
132 a.puA           = 0;
133
134 a.Z = zeros((a.ncol+2)*ns,a.nz);
135 a.x = zeros((a.ncol+2)*ns,1);
136 a.R = zeros((a.ncol+2)*ns,a.nxx); %% component reaction rates + RT
137 a.alpha = zeros((a.ncol+2)*ns,15);
138
139 %a=class(a,'fluidpath');
140
141 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
142 function [r, A, B, q]=colloc(n,left,right)
143 % colloc: Calculate collocation weights
144 % [r, A, B, q] = colloc( n [, 'left'] [, 'right'])
145 % Inputs:
146 % n - Number of interior node points
147 % 'left' - Include left boundary
148 % 'right' - Include right boundary also
149 % Outputs:
150 % r - Vector of roots
151 % A - Matrix of first derivative weights
152 % B - Matrix of second derivative weights
153 % q - Quadrature weights.
154 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
155 % Copyright (C) 1996, 1997 John W. Eaton
156 %
157 % This program is free software; you can redistribute it and/or ...
    modify
158 % it under the terms of the GNU General Public License as ...
    published by

```

```

159 % the Free Software Foundation; either version 2, or (at your option)
160 % any later version.
161 %
162 % This program is distributed in the hope that it will be useful, but
163 % WITHOUT ANY WARRANTY; without even the implied warranty of
164 % MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
165 % General Public License for more details.
166 %
167 % You should have received a copy of the GNU General Public License
168 % along with Octave; see the file COPYING. If not, write to the Free
169 % Software Foundation, 59 Temple Place - Suite 330, Boston, MA
170 % 02111-1307, USA.
171 %
172 % Adapted from Octave's colloc.cc by Steve Swinnea.
173 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
174
175 n0 = 0 ; n1 = 0;
176 if (nargin > 1)
177     if (strcmp(left,'left') | strcmp(left,'l') )
178         n0 = 1;
179     elseif (left == 0 | left == 1 )
180         n0 = left;
181     else
182         error('Second argument should be the string left or l')
183     end
184 end
185 if (nargin > 2)
186     if (strcmp(right,'right') | strcmp(right,'r') )
187         n1 = 1;
188     elseif ( right == 1 | right == 0 )
189         n1 = right;
190     else
191         error('Third argument should be the string right or r')
192     end
193 end
194
195 [dif1,dif2,dif3,r]=jcobi(n,n0,n1,0,0);
196 q = dfopr(n,n0,n1,0,3,dif1,dif2,dif3,r);
197 for i=1:(n+n0+n1)
198     vect = dfopr(n,n0,n1,i,1,dif1,dif2,dif3,r);
199     A(i,:) = vect';
200 end

```

```

201 for i=1:(n+n0+n1)
202     vect = dfopr(n,n0,n1,i,2,dif1,dif2,dif3,r);
203     B(i,:) = vect';
204 end
205
206 %%%%%% jcobi %%%%%%
207 function [dif1,dif2,dif3,root]=jcobi(n,n0,n1,alpha,beta)
208 if (n0 ~= 0) & (n0 ~= 1)
209     error('** VILERR : Illegal value % N0 ');
210 end
211 if (n1 ~= 0) & (n1 ~= 1)
212     error('** VILERR : Illegal value for N1 ');
213 end
214 if (n+n0+n1 < 1)
215     error('** VILERR : Number of interpolation points less than 1');
216 end
217 %
218 % --- FIRST EVALUATION OF COEFFICIENTS IN RECURSION FORMULAS.
219 % --- RECURSION COEFFICIENTS ARE STORED IN DIF1 AND DIF2.
220 %
221 nt = n+n0+n1;
222 dif1=zeros(nt,1);
223 dif2=zeros(nt,1);
224 dif3=zeros(nt,1);
225 root=zeros(nt,1);
226 ab = alpha+beta;
227 ad = beta-alpha;
228 ap = beta*alpha;
229 dif1(1) = (ad/(ab+2)+1)/2;
230 dif2(1) = 0;
231
232 if (n >= 2)
233     for i=2:n
234         z1 = i-1;
235         z = ab + 2*z1;
236         dif1(i) = (ab*ad/z/(z+2)+1)/2;
237         if (i == 2)
238             dif2(i) = (ab+ap+z1)/z/z/(z+1);
239         else
240             z = z*z;
241             y = z1*(ab+z1);
242             y = y*(ap+y);

```

```
243         dif2(i) = y/z/(z-1);
244     end
245 end
246 end
247 %
248 % — ROOT DETERMINATION BY NEWTON METHOD WITH SUPPRESSION OF
249 % — PREVIOUSLY DETERMINED ROOTS
250 %
251 x = 0;
252 for i=1:n
253     z = 1;
254     while ( abs(z) > 1e-9 )
255         xd = 0;
256         xn = 1;
257         xd1 = 0;
258         xn1 = 0;
259         for j=1:n
260             xp = (dif1(j)-x)*xn - dif2(j)*xd;
261             xp1 = (dif1(j)-x)*xn1 - dif2(j)*xd1 - xn;
262             xd = xn;
263             xd1 = xn1;
264             xn = xp;
265             xn1 = xp1;
266         end
267         zc = 1;
268         z = xn/xn1;
269         if ( i ~= 1 )
270             for j = 2:i
271                 zc = zc - z/(x-root(j-1));
272             end
273         end
274         z = z/zc;
275         x = x-z;
276     end
277     root(i) = x;
278     x = x +.0001;
279 end
280 %
281 % — ADD INTERPOLATION POINTS AT X = 0 AND/OR X = 1
282 %
283 if (n0 ~= 0)
284     root = [ 0 ; root(1:nt-1) ];
```

```

285 end
286 if (n1 == 1)
287     root(nt) = 1;
288 end
289 [dif1 dif2 dif3] = dif( root );
290
291 %%%% dfopr %%%%
292 function vect = dfopr(n,n0,n1,i,id,dif1,dif2,dif3,root)
293 nt = n+n0+n1;
294 vect = zeros(nt,1);
295 if (n0 ~= 0) & (n0 ~= 1)
296     error('** VILERR : Illegal value % N0 ');
297 end
298 if (n1 ~= 0) & (n1 ~= 1)
299     error('** VILERR : Illegal value for N1 ');
300 end
301 if (nt < 1)
302     error('** VILERR : Number of interpolation points less than 1');
303 end
304 if (id ~= 1 & id ~= 2 & id ~= 3 )
305     error('** VILERR : Illegal ID in DFOPR ');
306 end
307 if ( id ~= 3 )
308     if ( i < 1 )
309         error('** VILERR : Index less than zero in DFOPR ');
310     end
311     if ( i > nt )
312         error('** VILERR : Index greater than NTOTAL in DFOPR ');
313     end
314 end
315 %
316 % --- EVALUATE DISCRETIZATION MATRICES AND GAUSSIAN QUADRATURE
317 % --- WEIGHTS.  QUADRATURE WEIGHTS ARE NORMALIZED TO SUM TO ONE.
318 %
319 if ( id ~= 3 )
320     for j = 1:nt
321         if (j == i)
322             if (id == 1)
323                 vect(i) = dif2(i)/dif1(i)/2;
324             else
325                 vect(i) = dif3(i)/dif1(i)/3;
326             end

```

```

327         else
328             y = root(i)-root(j);
329             vect(j) = dif1(i)/dif1(j)/y;
330             if (id == 2 )
331                 vect(j)=vect(j)*(dif2(i)/dif1(i)-2/y);
332             end
333         end
334     end
335 else
336     y=0;
337     for j = 1:nt
338         x = root(j);
339         ax = x*(1-x);
340         if (n0 == 0)
341             ax = ax/x/x;
342         end
343         if (n1 == 0)
344             ax = ax/(1-x)/(1-x);
345         end
346         vect(j) = ax/dif1(j)^2;
347         y = y + vect(j);
348     end
349     vect = vect/y;
350 end
351
352 %%%% dif %%%%
353 function [dif1,dif2,dif3] = dif( root )
354 nt = length( root );
355 dif1 = zeros(nt,1);
356 dif2 = zeros(nt,1);
357 dif3 = zeros(nt,1);
358 if ( nt < 1 )
359     error('** VILERR : Number of interpolation points less than 1');
360 end
361 for i = 1:nt
362     x = root(i);
363     dif1(i) = 1;
364     dif2(i) = 0;
365     dif3(i) = 0;
366     for j = 1:nt
367         if ( j ~= i)
368             y = x - root(j);

```



```

369         dif3(i) = y*dif3(i) + 3*dif2(i);
370         dif2(i) = y*dif2(i) + 2*dif1(i);
371         dif1(i) = y*dif1(i);
372     end
373 end
374 end

```

### C.3 integrate.m

```

1 function a=integrate(a)
2 % Function INTEGRATE
3 % Initial values
4 %tic
5 %zz0=[a.xx0; 1; 1];
6 zz0 = [a.xx0; 1];
7 xi0 = 0.0;
8 Z = []; xi = []; R = []; alpha = [];
9 %options=odeset('RelTol',1e-12,'AbsTol',1.e-12);
10 % % options=odeset('RelTol',1e-6,'AbsTol',1.e-6);
11 %options=odeset('RelTol',1e-5,'AbsTol',1.e-6);
12 for ir=1:a.ns
13     xil = xi0 + a.dxi(ir);
14     xispan = [xi0 xil];
15     xispan = a.roots*a.dxi(ir) + xi0; % Use values only on the roots
16     a.uF0(ir) = a.uF(ir); % Default is distributed feed.
17     if a.uFp(ir) == 1 || a.uFFp(ir)==1 % If point feed
18         a.uF1(ir)=(a.uF(2*ir-1)+a.uF(2*ir))*0.5*a.dxi(ir)*a.sigma; ...
19         % Interpolate
20         a.uF1(ir)=(a.uFF(2*ir-1)+a.uFF(2*ir))*0.5*a.dxi(ir)*a.sigma;
21         a.uF0(ir)=0.0; a.uFF0(ir)=0;
22         nc = a.nc;
23         y = zz0(1:nc); %@mhi 050608
24         yF = a.xxF(1:nc); % Composition of 1st side stream
25         yFF = a.xxFF(1:nc); % Composition of 2nd side stream
26         TF = a.xxF(a.nxx); % Temperature of the 1st ...
27         % side stream
28         TFF = a.xxFF(a.nxx); % Temperature of 2nd side stream
29         ga = zz0(a.nxx+1);
30         T = zz0(a.nxx); %% [K]
31         cp = a.cp*y; %@mhi 120608
32         cpF = a.cp*yF; %@mhi 120608

```

```

31     cpFF = a.cp*yFF;
32     gat = ga + a.uF1(ir)+a.uFF1(ir);
33     yt = (a.uF1(ir)*yF+a.uFF1(ir)*yFF + ga*y)/gat;
34     Tt = (a.uF1(ir)*cpF*TF+a.uFF1(ir)*cpFF+ga*cp*T)/...
35         (ga*cp+a.uF1(ir)*cpF+a.uFF1(ir)*cpFF);
36     zz0 = [yt;Tt;gat];
37     end
38     if a.usp==1 % This should be place after each integration
39         kappa = (a.uS(2*ir-1)+a.uS(2*ir))*0.5; %
40         ga = zz0(a.nxx+1);
41         y = zz0(1:a.nc);
42         sum1 = sum(a.xS.*y);
43         yt = (1-kappa*a.xS).*y./(1-kappa*sum1);
44         Tt = 10./a.Tref;
45         gat = ga-kappa*ga*sum1;
46         zz0 =[yt;Tt;gat];
47     end
48     [x,Zi]= ode15s(@ (x,zz) ...
49         DesignModel(x,zz,a,ir,xi0),xispan,zz0);%options);
50     % [x,Zi]= ode23s(@ (x,zz) ...
51         DesignModel(x,zz,a,ir,xi0),xispan,zz0,options);
52     xi0 = xil;
53     uA = a.uA(ir);
54     for i=1:length(x)
55         xx = Zi(i,1:a.nxx)';
56         cp = a.cp*xx(1:a.nc);
57         [Ri, alphai] = feval(a.kinetics,xx,uA,cp,a.M',a.Tref,a.pTot);
58         R = [R;Ri' ];
59         alpha = [alpha; alphai'];
60     end
61     zz0 = Zi(end,:)';
62     Z=[Z;Zi]; xi=[xi;x];
63     end
64     a.Z=Z;
65     a.x=xi;
66     %a.xx=zz0(1:a.nxx);
67     a.xx=zz0; %endret 6Aug07
68     a.R=R;
69     a.alpha = alpha;

```

## C.4 OptimizeDesign.m

```

1 function [a bb] = OptimizeDesign(a,niter)
2 %function [A b uu]=OptimizeDesign(a)
3 % OPTIMIZEDDESIGN is a function that optimize the design defined ...
   in an
4 % object of class fluidpath.
5 % The vector uu is assigned a fixed structure.
6 % If elements in uu are not to be optimized but fixed as given in a,
7 % they are fixed by upper and lower bound conditions.
8 %
9 % uu=[uM(2) uM3' uM3 uM4' uM4                2*ns-1    1..2*ns-1
10 %     [dxi(1)..dxi(ns)                        ns        2*ns..3*ns-1
11 %     [VRoverF0                                1          3*ns
12 %     [uH(1) uH(2) ...uH(nH)                  ns        3*ns+1..4*ns
13 %     [uT(1) uT(2) ...uT(nT)                  ns        4*ns+1..5*ns
14 %     [uA(1) uA(2) ...uA(ns)                  ns        5*ns+1..6*ns
15 %     [uF(1) uF(2) ...uF(ns)                  ns        6*ns+1..7*ns
16 %
17 % A*uu<=b
18 % Aeq*uu=beq
19 %
20 ns = a.ns;                %% Number of stages
21 n = 3*ns-1;              %% Least number of optimization variables
22
23 nV = a.update_sigma;     %% Is 0 or 1
24 n = n+nV;
25
26 a.puH = n;               %% Heat exchange
27 a.iuH = find(a.update_uH==1);
28 a.luH = length(a.iuH);
29 n = n + a.luH;
30
31 a.puT = n;               %% Temperature cooling media
32 a.iuT = find(a.update_uT==1);
33 a.luT = length(a.iuT);
34 n = n + a.luT;
35
36 a.puA = n;               %% Catalyst distribution
37 a.iuA = find(a.update_uA==1);
38 a.luA = length(a.iuA);

```

```

39 n = n + a.luA;
40
41 a.puF = n;                %% Feed distribution
42 a.iuF = find(a.update_uF==1);
43 a.luF = length(a.iuF);
44 n = n + a.luF;
45
46 a.puFF = n;              %% Second feed distribution
47 a.iuFF = find(a.update_uFF==1);
48 a.luFF = length(a.iuFF);
49 n = n + a.luFF;
50
51 a.puS = n;              %% Separation
52 a.iuS = find(a.update_uS==1);
53 a.luS = length(a.iuS);
54 n = n + a.luS;
55
56 a.ndof = n;            %% Degrees of freedom
57 n = n + (a.ncol+2)*a.ns+a.nz;
58
59 %n =n+a.nxx; %mhi 31Augu07
60 %a.key=n;
61
62 % n is total number of variables to be optimized
63 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
64 % Set linear inequality constraints on uM
65 % A*uM<=b
66 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
67 %A = zeros(2*ns-1,n);b=zeros(2*ns-1,1);
68 A = zeros(3*ns-2,n); b = zeros(3*ns-2,1);
69 for i=1:2*ns-1, A(i,i) = 1.0; end
70 for i=1:2*ns-2, A(i+1,i)=-1.0; end
71 %for i=1:ns, b(2*i-1)=a.dxi(i); end
72 %for i=1:ns, A(2*i-1,2*ns+i)=-1.0; end
73 for i=1:ns, A(2*i-1,2*ns+i-1)=-1.0; end %@
74 for i=1:ns-1, A(2*ns-1+i,2*i)=1; end
75 for i=1:ns-1, A(2*ns-1+i,2*i+1)=-1; end
76 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
77 % Set linear equality constraints on dxi sum(dxi(i))=1
78 % and 1.0*u=a.u where u is constant
79 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80 Aeq = zeros(1,n);beq=zeros(1,1);

```

```

81 %Aeq(1,2*ns+1:3*ns)=1.0; beq(1)=1.0;
82 Aeq(1,2*ns:3*ns-1)=1.0; beq(1)=1.0; %@
83
84 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
85 % Set lower and upper bounds on uu
86 % uu=[uM1 uM1' uM2 uM2' uM3          2*ns-1      1..2*ns-1
87 %     [dxi(1)..dxi(ns)              ns          2*ns..3*ns-1
88 %     [VRoverF0                      1          3*ns
89 %     [uH(1) uH(2) ...uH(ns)         ns          3*ns+1..4*ns
90 %     [uT(1) uT(2) ...uT(ns)         ns          4*ns+1..5*ns
91 %     [uA(1) uA(2) ...uA(ns)         ns          5*ns+1..6*ns
92 %     [uF(1) uF(2) ...uF(ns)         ns          6*ns+1..7*ns
93 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
94 lb=zeros(n,1);
95 ub=zeros(n,1);
96 lb(1:n) = 0.0;           % No design function can be negative
97 ub(1:n) = inf;          % Need to be modified
98 lb(2*ns:3*ns-1)=1/a.ns*0.1; % Minimum dxi
99 ub(2*ns:3*ns-1)=1.0;    % dxi is max 1.0
100 if nV==1,
101     lb(3*ns) = a.sigma_min; % Minimum VRoverF0
102     ub(3*ns) = a.sigma_max; % Maximum VRoverF0
103 end
104 if a.luH>0,
105     lb(a.puH+1:a.puH+a.luH)=a.uH_min(a.iuH); % Min heat ...
106         transfer area
107     ub(a.puH+1:a.puH+a.luH)=a.uH_max(a.iuH); % Max heat ...
108         transfer area
109 end
110 if a.luT>0,
111     lb(a.puT+1:a.puT+a.luT)=a.uT_min(a.iuT); % Min coolant ...
112         temperature
113     ub(a.puT+1:a.puT+a.luT)=a.uT_max(a.iuT); % Max coolant ...
114         temperature
115 end
116 if a.luA>0,
117     lb(a.puA+1:a.puA+a.luA)=a.uA_min(a.iuA); % Min catalyst ...
118         activity
119     ub(a.puA+1:a.puA+a.luA)=a.uA_max(a.iuA); % Max catalyst ...
120         activity
121 end
122 if a.luF>0,

```

```

117     lb(a.puF+1:a.puF+a.luF)=a.uF_min(a.iuF);      % Min extra feed
118     ub(a.puF+1:a.puF+a.luF)=a.uF_max(a.iuF);      % Max extra feed
119 end
120
121 if a.luFF>0,
122     lb(a.puFF+1:a.puFF+a.luFF)=a.uFF_min(a.iuFF); % Min extra feed
123     ub(a.puFF+1:a.puFF+a.luFF)=a.uFF_max(a.iuFF); % Max extra feed
124 end
125
126 if a.luS>0,
127     lb(a.puS+1:a.puS+a.luS)=a.uS_min(a.iuS);      % Min extra feed
128     ub(a.puS+1:a.puS+a.luS)=a.uS_max(a.iuS);      % Max extra feed
129 end
130 % No constraints on Z.
131 lb(a.ndof+1:n)=-inf;
132 ub(a.ndof+1:n)= inf;
133 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
134 % Optimize the design
135 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
136 uu=AssignVector(a);
137 %uu(1:a.ndof)=uu(1:a.ndof)+rand((a.ndof),1);
138 uu0=uu;
139 %bb=uu;
140 %     'HessUpdate','bfgs',...
141 %     'LevenbergMarquardt','on',...
142 %return
143 %'TolFun',1.e-5,'TolX',1.e-5,'Display','iter','MaxIter',niter);
144 options = optimset('Algorithm','active-set','TolCon',a.tol(1),...
145     'TolFun',a.tol(2),'Display','iter','HessUpdate','bfgs',...
146     'MaxFunEval',100000000,'MaxIter',niter,'UseParallel','always');
147 [uu,fval,exitflag,output,lambda,grad,hessian] = ...
148     fmincon(@(uu) ...
149         ObjectiveFunction(uu,a),uu0,A,b,Aeq,beq,lb,ub,...
150         @(uu) ModelConstraints(uu,a),options);
151 display(exitflag)
152 disp(A*uu-b);
153 disp(Aeq*uu-beq);
154
155 a=AssignObject(a,uu);
156 bb.uu=uu;
157 bb.fval=fval;
158 bb.exitflag=exitflag;

```

```

158 bb.output=output;
159 bb.lambda=lambda;
160 bb.grad=grad;
161 bb.hessian=hessian;
162 return

```

## C.5 DesignModel.m

```

1 function [f] = DesignModel(xi,zz, a,ir,xi0)
2 % DESIGNMODEL is the ODE describing the change of states along ...
   the path
3 %
4 % uu,xi0,dxi,nc,VRoF0)
5 %
6 % y =zz(1:nc)
7 % T =zz(nc+1)
8 % ga=zz(nc+2)
9 %
10 % uM=uu(1)-uu(2) linear
11 % uH=uu(3)
12 % uF=uu(4)
13 % uT=uu(5)
14 %
15 % R = [Ry; RT]
16 % Ry(1:nc) = mol/(m3*s)
17 % RT = mol*K/(m3*s)
18
19 %
20 % Assign the design functions
21 %
22 %xi0=sum(a.dxi(1:ir-1));
23 uM = a.uM(2*ir-1) + (xi-xi0)/a.dxi(ir)*(a.uM(2*ir)-a.uM(2*ir-1)); ...
   % For piecewise - linear
24 uH = a.sigma*40*(a.uH(2*ir-1) + ...
   (xi-xi0)/a.dxi(ir)*(a.uH(2*ir) - a.uH(2*ir-1)));
25 uF = a.uF0(ir)*a.sigma*(a.uF(2*ir-1) + ...
   (xi-xi0)/a.dxi(ir)*(a.uF(2*ir) - a.uF(2*ir-1)));
26 uFF = a.uF0(ir)*a.sigma*(a.uFF(2*ir-1) + ...
   (xi-xi0)/a.dxi(ir)*(a.uFF(2*ir) - a.uFF(2*ir-1)));
27 uA = a.uA(2*ir-1) + (xi-xi0)/a.dxi(ir) * ( ...
   a.uA(2*ir)-a.uA(2*ir-1) );

```

```

31 %uS=a.uS(2*ir-1) + (xi-xi0)/a.dxi(ir) * ( a.uS(2*ir)-a.uS(2*ir-1) );
32 uT = a.uT(2*ir-1) + (xi-xi0)/a.dxi(ir) * ( ...
      a.uT(2*ir)-a.uT(2*ir-1) );
33
34 %uH = a.uH(ir) *a.sigma*40;      %kg/(s m3) a.uH(ir) is scaled ...
      07-10-2009-mhi
35 %uH=uH*a.sigma*40;
36 % uF = a.uF0(ir)*a.sigma;      %kg/(s m3) If point feed uF0 has been ...
      set to
37
38                                     %zero before integration and in
39                                     %ModelConstrain
39 % uFF= a.uFF0(ir)*a.sigma;
40 % uT = a.uT(ir);                % (Tw-Tref)/Tref
41 % uA = a.uA(ir);
42
43 xx = zz(1:a.nxx);
44 ga = zz(a.nxx+1);               %ga=F/F0           %@mhi 050608
45 nc = a.nc;
46
47 cp = a.cp*xx(1:nc);            %@mhi 120608
48 cpF = a.cp*a.xx(1:nc);        %@mhi 120608
49 cpFF = a.cp*a.xxFF(1:nc);
50 cpR = a.cp*a.xx0(1:nc);
51 R = feval(a.kinetics,xx,uA,cp,a.M',a.Tref,a.pTot);
52 Rt = a.sigma*R;
53 Rt(1:nc) = Rt(1:nc) + uF*      (a.xx(1:nc) - xx(1:nc));
54 Rt(1:nc) = Rt(1:nc) + uFF*    (a.xxFF(1:nc) - xx(1:nc));
55 Rt(a.nxx) = Rt(a.nxx) + uF*cpF/cp*(a.xx(a.nxx) - xx(a.nxx)) + ...
      uFF*cpFF/cp*(a.xxFF(a.nxx) - xx(a.nxx));
56 Rt(a.nxx) = Rt(a.nxx) - uH*cpR/cp*(xx(a.nxx)-uT);
57 %
58 %if uM>0,
59     J = Jacobian(a,xx,uA,cp,a.M');
60     %J= Jgraaf2(xx,uA,cp,a.M',a.Tref);
61     %if xi>0.5&xi<0.6, disp(xi); disp(Ja); disp(J); end
62     Jt = a.sigma*J;
63     Jt(a.nxx,a.nxx) = Jt(a.nxx,a.nxx) - uH*cpR/cp + uF*(1-cpF/cp) ...
      + uFF*(1-cpFF/cp);
64     J = (ga*eye(a.nxx)-uM*Jt);
65     f = J\Rt;
66 %else
67 %     f = Rt/ga;

```



```

68 %end
69 dgadxi = uF + uFF;           % uF=sigma*alpha @mhi 050608
70 f = [f; dgadxi];

```

## C.6 kinTodicLump.m

```

1 function [R, alpha, varargout] = kinTodicLump(xx,uA,Cp,M,Tref,pTot)
2 %-----%
3 % Name           : kinTodicLump.m
4 % Author        : Martin S. Foss
5 % Date          : Spring 2013
6 % Input:
7 %   xx(1)       : Mass fraction of CO
8 %   xx(2)       : Mass fraction of CO2
9 %   xx(3)       : Mass fraction of H2
10 %  xx(4)       : Mass fraction of H2O
11 %  xx(5)       : Mass fraction of C1
12 %  xx(6)       : Mass fraction of C2
13 %  xx(7)       : Mass fraction of C3-4
14 %  xx(8)       : Mass fraction of C5-10
15 %  xx(9)       : Mass fraction of C11+
16 %  xx(10)      : Temperature (T-Tref)/Tref [-], T = [K]
17 %  uA          : Relative activity (catalyst dilution)
18 %  cp          : Average cp of composition [kJ/(kg*K)]
19 %  M(1:6)      : Molecular weight [kg/kmol]
20 %  Tref        : Reference temperature [K]
21 %  pTot        : Total pressure [Pa]
22 % Output:
23 %  R(i)        : Component reaction rates [kg/(m3*s)]
24 %  alpha       : Growth probability
25 %
26 % Reactions taking place:
27 %   iCO + (2i + 1)H2 = Ci + iH2O, i = 1..15 (Paraffins)
28 %   iCO + (2i)H2 = Ci + iH2O, i = 2..15 (Olefins)
29 %-----%
30
31 Rgas = 8.314472e-03;           %% Gas constant [kJ/K*mol]
32 p = pTot/1e06;                 %% Pressure [MPa]
33 T = Tref*(xx(end) + 1);        %% Temperature [K]
34
35 %% Parameters

```

```

36 A1 = 1.83e10;      %% [mol/(gcat*h*MPa)]
37 A2 = 5.08e00;    %% [-]
38 A3 = 2.44e01;    %% [MPa^-1]
39 A4 = 2.90e00;    %% [-]
40 A5 = 4.49e05;    %% [mol/(gcat*h*MPa)]
41 A6 = 7.47e08;    %% [mol/(gcat*h)]
42 A7 = 1.00e-03;   %% [MPa^-1]
43 A5M = 8.43e05;   %% [mol/(gcat*h*MPa)]
44 A6E = 7.03e08;   %% [mol/(gcat*h)]
45 E1 = 100.4;      %% [kJ/mol]
46 E5 = 72.4;       %% [kJ/mol]
47 E5M = 63.0;      %% [kJ/mol]
48 E60 = 97.2;      %% [kJ/mol]
49 E6E0 = 108.8;    %% [kJ/mol]
50 deltaH2 = 8.68;  %% [kJ/mol]
51 deltaH3 = 9.44;  %% [kJ/mol]
52 deltaH4 = 7.9;   %% [kJ/mol]
53 deltaH7 = -25.0; %% [kJ/mol]
54 deltaE = 1.12;   %% [kJ/(mol_CH2)]
55
56 %% Kinetic constants
57 k1 = A1*exp(-E1/(Rgas*T));    %% [mol/(gcat*h*MPa)]
58 K2 = A2*exp(-deltaH2/(Rgas*T)); %% [-]
59 K3 = A3*exp(-deltaH3/(Rgas*T)); %% [MPa^-1]
60 K4 = A4*exp(-deltaH4/(Rgas*T)); %% [-]
61 k5M = A5M*exp(-E5M/(Rgas*T)); %% [mol/(gcat*h*MPa)]
62 k5 = A5*exp(-E5/(Rgas*T));    %% [mol/(gcat*h*MPa)]
63 k6E = A6E*exp(-E6E0/(Rgas*T)); %% [mol/(gcat*h)]
64 k60 = A6*exp(-E60/(Rgas*T));  %% [mol/(gcat*h)]
65 K7 = A7*exp(-deltaH7/(Rgas*T)); %% [Mpa^-1]
66
67 %%
68 %Assume sharp split: all components with carbon number greater ...
    than 5 in liquid phase
69 gasPhase = xx(1:7);
70 molarGas = M(1:7);
71
72 omegaGas = gasPhase./sum(gasPhase);
73 moleFracGas = masstomole(omegaGas,molarGas);
74
75 partialP = moleFracGas*p;      %% [MPa]
76 pCO = partialP(1);           %% [MPa]

```

```

77 pH2 = partialP(3);           %% [MPa]
78 pH2O = partialP(4);         %% [MPa]
79
80 c = -deltaE/(Rgas*T);
81
82 alpha = zeros(15,1);
83 alpha(1) = k1*pCO/(k1*pCO + k5M*pH2);
84 alpha(2) = k1*pCO/(k1*pCO + k5*pH2 + k6E*exp(2*c));
85
86 for i=3:15
87     alpha(i) = k1*pCO/(k1*pCO + k5*pH2 + k60*exp(i*c));
88 end
89
90 alphaSum = 0;
91 for i=3:15
92     alphaSum = alphaSum + prod(alpha(3:i));
93 end
94
95 S = 1/(1 + sqrt(K7*pH2) + sqrt(K7*pH2)*(1 + 1/K4 + 1/(K3*K4*pH2) ...
96     + pH2O/(K2*K3*K4*pH2^2))* (alpha(1) + prod(alpha(1:2)) + ...
97     prod(alpha(1:2))*alphaSum));    %% Fraction of vacant sites
98
99 rCH4 = k5M*K7^0.5*pH2^1.5*alpha(1)*S;           %% ...
100     [mol/(gcat*h)]
101
102 rC2H4 = k6E*exp(2*c)*sqrt(K7*pH2)*prod(alpha(1:2))*S;    %% ...
103     [mol/(gcat*h)]
104
105 rParaffin = zeros(15,1);
106 rParaffin(1) = rCH4;           %% [mol/(gcat*h)]
107
108 rOlefin = zeros(15,1);
109 rOlefin(2) = rC2H4;           %% [mol/(gcat*h)]
110
111 for i=2:15
112     rParaffin(i) = ...
113         k5*K7^0.5*pH2^1.5*prod(alpha(1:2))*prod(alpha(3:i));
114 end
115
116 for i=3:15
117     rOlefin(i) = k60*exp(c*i)*sqrt(K7*pH2)*prod(alpha(1:2))* ...
118         prod(alpha(3:i));
119 end

```

```

116 bulkCatDensity = 200e03;    %% [g/m^3]
117 kmole = 1000;              %% [mol/kmol]
118 sperh = 3600;              %% [s/h]
119
120 rParaffin = rParaffin*bulkCatDensity*kmole^(-1)*sperh^(-1); %% ...
    [kmol/(m^3*s)]
121 rOlefin = rOlefin*bulkCatDensity*kmole^(-1)*sperh^(-1);    %% ...
    [kmol/(m^3*s)]
122
123 rCO = 0;
124 rH2 = 0;
125 for i = 1:15
126     rCO = rCO + i*(rParaffin(i) + rOlefin(i));
127     rH2 = rH2 + ((2*i + 1)*rParaffin(i) + 2*i*rOlefin(i));
128 end
129
130 %% Lumping components:
131 rC1 = rParaffin(1);
132 rC2 = rParaffin(2) + rOlefin(2);
133 rC34 = sum(rParaffin(3:4)) + sum(rOlefin(3:4));
134 rC510 = sum(rParaffin(5:10)) + sum(rOlefin(5:10));
135 rC11 = sum(rParaffin(11:end)) + sum(rOlefin(11:end));
136
137 Mwhole = intMolarMass;      %% Molecular weight, whole numbers ...
    (C1 - C15)
138 Mpar = Mwhole(5:19);        %% Molecular weight paraffins
139 Mole = [0; Mwhole(20:end)]; %% Molecular weight olefins
140 M2 = (rParaffin(2)*Mpar(2) + rOlefin(2)*Mole(2))/ ...
    (rC2);
141
142 M34 = (rParaffin(3:4)'*Mpar(3:4) + rOlefin(3:4)'*Mole(3:4))/(rC34);
143 M510 = (rParaffin(5:10)'*Mpar(5:10) + ...
    rOlefin(5:10)'*Mole(5:10))/(rC510);
144 M11 = (rParaffin(11:end)'*Mpar(11:end) + ...
    rOlefin(11:end)'*Mole(11:end))/(rC11);
145
146 Mtemp = [M(1:5);
147          M2;
148          M34;
149          M510;
150          M11];
151
152 r = [-rCO;          %% CO

```

```

153     0;           %% CO2
154     -rH2;       %% H2
155     rCO;        %% H2O
156     rC1         %% C1
157     rC2;        %% C2
158     rC34;       %% C3-4
159     rC510;      %% C5-10
160     rC11];      %% C11+ [kmol/(m^3*s)]
161
162     g = uA*r.*Mtemp;           %% [kg/(m^3*s)]
163     massBalance = sum(g);
164     if massBalance > 0.0000001
165         error('error in mass balance')
166     end
167     rTemp = [-rCO;           %% CO      [kmol/(m^3*s)]
168             0;              %% CO2
169             -rH2;           %% H2
170             rCO;           %% H2O
171             rParaffin;      %% CnH (2n+2)
172             rOlefin(2:15)]; %% CnH (2n)
173
174     deltaH1 = deltaHrx(rTemp); %% [kJ/(m^3*s)]
175     hh = -deltaH1/(Cp*Tref);    %% [kg/(m^3*s)]
176     R = [g; hh];              %% [kg/(m^3*s)]
177     return

```

## C.7 ObjectiveFunction.m

```

1  function [fval] = ObjectiveFunction(uu,a)
2  % OBJECTIVEFUNCTION is the function that is minimized by the ...
   optimization
3  % program.
4  %
5  a = AssignObject(a,uu);
6  %end_point= a.ns*(a.ncol+2);
7  J(1) = a.Z(end,a.key);           % Mass fraction at outlet
8  prod = 0;
9  tnc = a.ncol+2;
10 for i=1:a.ns;
11     prod = prod + (a.Z(i+tnc, a.key) - a.Z(tnc*(i-1)+1,a.key)) * ...
        a.Z(i+tnc,a.nz);

```

```

12 end
13 J(2) = prod;      % Mass production
14 J(3) = prod./a.sigma;
15 uHmean = mid(a, 'uH')*40;
16 J(4) = J(3) / (1+0.01 *uHmean*a.sigma);
17 J(5) = J(3) / (1+0.008*uHmean*a.sigma);
18 J(6) = J(3) / (1+0.006*uHmean*a.sigma);
19 J(7) = J(3) / (1+0.004*uHmean*a.sigma);
20 J(8) = exp(J(2));
21 J(9) = exp(J(3));
22 %a=pathset(a, 'J', J);
23 %a.Jcrit=J;
24 uAmean = mid(a, 'uA');
25 J(9) = J(1) / (1+0.008*(uHmean+2*uAmean) *a.sigma);
26
27 fval = -J(a.Jcrit);

```

## C.8 ModelConstraints.m

```

1 function [c, ceq] = ModelConstraints(uu, a)
2 a = AssignObject(a, uu);
3 Z = a.Z;
4 Zout = a.Z;
5 n = (a.ncol + 2)*a.ns;
6 F = zeros(n, a.nz);
7 zz0 = [a.xx0; 1];
8 xi0 = 0;
9 nc = a.nc;
10 cpF = a.cp*a.xxF(1:nc);
11 cpFF = a.cp*a.xxFF(1:nc);
12 del = zeros(a.nz, 1);
13 ee = zeros(a.nz, 1);
14 %%
15 for s=1:a.ns
16     p = (s-1)*(a.ncol+2);
17     p2 = (s)*(a.ncol+2);
18     if p==0,
19         zB = zz0;
20     else
21         zB = Zout(p, :)';
22     end

```

```

23     gammaB = zB(a.nz);
24     omegaB = zB(1:nc);
25     thetaB = zB(nc+1);
26     if a.uFp(s)==1 || a.uFFp(s)==1
27         a.uF0(s) = 0;
28         a.uFF0(s) = 0;
29         delgamma0 = (a.uF(2*s-1) + a.uF(2*s))*0.5*a.dxi(s)*a.sigma;
30         delgamma1 = (a.uFF(2*s-1) + a.uFF(2*s))*0.5*a.dxi(s)*a.sigma;
31         if delgamma0~=0 || delgamma1~=0
32             delgamma = delgamma0 + delgamma1;
33             omega = delgamma0/delgamma*a.xx(1:nc) + ...
34                 delgamma1/delgamma*a.xx(1:nc);
35             cpFm = a.cp*omega;
36             theta = cpF*delgamma0/(cpF*delgamma0 + ...
37                 cpFF*delgamma1)*a.xx(nc+1);
38             theta = theta + cpFF*delgamma1/(cpF*delgamma0 + ...
39                 cpFF*delgamma1)*a.xx(nc+1);
40             delomega = delgamma/(gammaB + delgamma)*(omega - omegaB);
41             cpB = a.cp*omegaB;
42             kapa = cpFm/cpB;
43             deltheta = delgamma*kapa/(gammaB + delgamma*kapa)* ...
44                 (theta - thetaB);
45         else
46             delomega = zeros(nc,1);
47             deltheta = 0.;
48             delgamma = 0.;
49         end
50         del = [delomega; deltheta; delgamma];
51     else
52         a.uF0(s) = a.uF(s);
53         a.uFFu(s) = a.uFF(s);
54     end
55     if a.uSp(s)==1
56         kappa = (a.uS(2*s-1) + a.uS(2*s))*0.5;
57         gamma = Zout(p2,nc+2);
58         omega = Zout(p2, 1:nc);
59         sum1 = sum(a.xS.*omega');
60         delgamma = kappa*gamma*sum1;
61         delomega = zeros(nc,1);
62         if delgamma~=0
63             delomega = kappa*(a.xS - sum1)./(1.0 - ...
64                 kappa*sum1).*omega';

```

```

60         deltheta = 10./a.Tref;
61     else
62         delomega = 0.0;
63         deltheta = 0.0;
64     end
65     ee(1:nc) = delomega;
66     ee(nc+1) = deltheta;
67     ee(nc+2) = delgamma;
68 end
69 Zout(p2,:) = Zout(p2,:) - ee';
70 Zout(p2,1:nc) = Zout(p2,1:nc)./sum(Zout(p2,1:nc));
71 %
72 F(p+1,:) = Z(p+1,:) - zB' - del';
73 jj = p+1:p+a.ncol+2;
74 for i=2:a.ncol+2
75     xi = xi0 + a.roots(i)*a.dxi(s);
76     zz = Z(p+i,:)';
77     f = DesignModel(xi,zz, a,s,xi0);
78     for c=1:a.nz
79         F(p+i,c) = a.A(i,:)*Z(jj,c) - a.dxi(s)*f(c);
80     end
81 end
82 xi0 = xi0+a.dxi(s);
83 end
84 ceq = reshape(F,n*a.nz,1);
85 % Application specific constraints
86 c1 = [Z(:,nc+1) - (250 + 273 - a.Tref)/a.Tref]; % Temperature(xi) ...
87     < 250
88 c = [c1];
89 return

```

## C.9 flowPropTPlump.m

```

1 function [M, cpnom, liqFrac] = flowPropTPlump(T,P)
2 % Function returning molecular weight, heat capacity and liquid ...
3   fraction
4   % for the different components (lumps) at given temperature and ...
5   pressure
6
7 % Written by: Martin S. Foss, Spring 2013
8 %-----%

```



```

7 Tvec = [200 205 210 215 220 225 230];    %% [C]
8 Pvec = [1.5e6 2.0e6 2.5e6 2.7e6];        %% [Pa]
9
10 load MolarMassLump.txt
11 load CpMatLump.txt
12 load liqFracMatLump.txt
13
14 nP = length(Pvec);
15
16 i = find(Tvec == T);
17 j = find(Pvec == P);
18
19 Tcheck = isempty(i);
20 Pcheck = isempty(j);
21
22 if Tcheck == 1
23     error('There are no Cp-values for the given T. Change the ...
24           temperature')
25 elseif Pcheck == 1
26     error('There are no Cp-values for the given P. Change the ...
27           pressure')
28 end
29
30 k = nP*(i-1) + j;
31
32 cpnom = CpMatLump(:,k);
33 liqFrac = liqFracMatLump(:,k);
34 M = MolarMassLump;
35 return

```

## C.10 moletomass.m

```

1 function w = moletomass(x,M)
2 % Function converting mole fraction to mass fraction
3 % Inputs:
4 %   x - Mole fraction [-]
5 %   M - Molar mass vector [kg/kmole]
6 % Output:
7 %   w - Mass fraction [-]
8 % Written by: Martin S. Foss, Spring 2013
9 %-----%

```

```

10 Mav = x'*M;
11 w = x.*M/Mav;
12 return

```

## C.11 *masstomole.m*

```

1 function x = masstomole(w,M)
2 % Function converting mass fraction to mole fraction
3 % Inputs:
4 %   w - Mass fraction [-]
5 %   M - Molar mass vector [kg/kmole]
6 % Output:
7 %   x - Mole fraction [-]
8 % Written by: Martin S. Foss, Spring 2013
9 %-----%
10 n = w./M;
11 ntot = sum(n);
12 x = n/ntot;
13 return

```

## C.12 *pathset.m*

```

1 function a=pathset(a,pn,pv,c)
2 % Overload function SET
3 % FLUIDPATH/PATHSET set properties of the a FLUIDPATH object
4 %   This file is an internal help function to set object properties.
5 if nargin == 3,
6     c=0;
7 end
8 if isnumeric(pv)
9     if size(pv,2)>1,
10         s=['a.',pn,'=[',mat2str(pv),'];'];
11         eval(s)
12         return
13     end
14     if length(pv)>1,
15         pv=reshape(pv,1,length(pv)); % Reshape to a row vector
16         if c==1,
17             s=['a.',pn,'=transpose([' ,num2str(pv),']);'];
18         else

```

```

19         s=['a.',pn,'=[',num2str(pv),'];'];
20     end
21     else
22         s=['a.',pn,'=',num2str(pv),'];'];
23     end
24     eval(s)
25     return
26 else
27     if pn=='kinetics',
28         a.kinetics=str2func(pv);
29     return
30     end
31 end
32 warning('the property name or value are wrong')
33 return

```

### C.13 AssignVector.m

```

1 function uu = AssignVector(a)
2 % ASSIGNVECTOR stores values in the object 'a'
3 % into a vector uu
4 % uu=[uM(2) uM3' uM3 uM4' uM4          2*ns-1    1..2*ns-1
5 %     [VRoverF0                        1          2*ns
6 %     [Dxi(1)..Dxi(ns)                  ns         2*ns+1..3*ns
7 %     [uH(1) uH(2) ...uH(ns)             ns         3*ns+1..4*ns
8 %     [uT(1) uT(2) ...uT(ns)             ns         4*ns+1..5*ns
9 %     [uA(1) uA(2) ...uA(ns)             ns         5*ns+1..6*ns
10 %    [uF(1) uF(2) ...uF(ns)             ns         6*ns+1..7*ns
11 ns = a.ns;
12 %n=3*ns-1 + a.update_sigma + a.luH + a.luT + a.luA + a.luF;
13 n = a.ndof + (a.ncol+2)*a.ns+a.nz;
14 uu = zeros(n,1);
15 uu(1:2*ns-1) = a.uM(2:2*ns);
16 uu(2*ns:3*ns-1) = a.dxi(1:ns);
17
18 if a.update_sigma==1,
19     uu(3*ns) = a.sigma;
20 end
21 if a.luH>0,
22     uu(a.puH+1:a.puH+a.luH) = a.uH(a.iuH);
23 end

```

```

24 if a.luT>0,
25     uu(a.puT+1:a.puT+a.luT) = a.uT(a.iuT);
26 end
27 if a.luA>0,
28     uu(a.puA+1:a.puA+a.luA) = a.uA(a.iuA);
29 end
30 if a.luF>0,
31     uu(a.puF+1:a.puF+a.luF) = a.uF(a.iuF);
32 end
33 if a.luFF>0,
34     uu(a.puFF+1:a.puFF+a.luFF) = a.uFF(a.iuFF);
35 end
36 if a.luS>0,
37     uu(a.puS+1:a.puS+a.luS) = a.uS(a.iuS);
38 end
39
40 z = reshape(a.Z, (a.ncol+2)*a.ns*a.nz, 1);
41 %uu=[uu; z];
42 uu(a.ndof+1:a.ndof+(a.ncol+2)*a.ns*a.nz) = z;

```

## C.14 AssignObject.m

```

1 function a = AssignObject(a,uu)
2 % ASSIGNOBJECT is a function
3 % uu=[uM(2) uM3' uM3 uM4' uM4          2*ns-1    1..2*ns-1
4 %     [dxi(1)dxi(2) ..dxi(ns)         ns        2*ns..3*ns-1
5 %     [VRoverF0                        1          3*ns
6 %     [uH(1) uH(2) ...uH(ns)           ns        3*ns+1..4*ns
7 %     [uT(1) uT(2) ...uT(ns)           ns        4*ns+1..5*ns
8 %     [uA(1) uA(2) ...uA(ns)           ns        5*ns+1..6*ns
9 %     [uF(1) uF(2) ...uF(ns)           ns        6*ns+1..7*ns
10 ns                = a.ns;
11 a.uM(1)            = 0.;
12 a.uM(2:2*ns)      = uu(1:2*ns-1);
13 a.dxi(1:ns)       = uu(2*ns:3*ns-1);
14 for i=1:ns+1,
15     a.xi(i) = sum(a.dxi(1:i-1));
16 end
17 for i=1:ns
18     for j=1:a.ncol+2
19         k = (i-1)*(a.ncol+2)+j;

```

```
20     a.x(k) = a.dxi(i)*a.roots(j)+a.xi(i);
21     end
22 end
23 if a.update_sigma ==1,
24     a.sigma = uu(3*ns);
25 end
26 % uH
27 if a.luH>0,
28     a.uH(a.iuH) = uu(a.puH+1:a.puH+a.luH);
29 end
30 for i=1:2*ns
31     if a.update_uH(i)==-1,
32         a.uH(i) = a.uH(i-1);
33     end
34 end
35 % uT
36 if a.luT>0,
37     a.uT(a.iuT) = uu(a.puT+1:a.puT+a.luT);
38 end
39 for i=1:2*ns
40     if a.update_uT(i)==-1,
41         a.uT(i) = a.uT(i-1);
42     end
43 end
44 % uA
45 if a.luA>0,
46     a.uA(a.iuA) = uu(a.puA+1:a.puA+a.luA);
47 end
48 for i=1:2*ns
49     if a.update_uA(i)==-1,
50         a.uA(i) = a.uA(i-1);
51     end
52 end
53 % uF
54 if a.luF>0,
55     a.uF(a.iuF) = uu(a.puF+1:a.puF+a.luF);
56 end
57 for i=1:2*ns
58     if a.update_uF(i)==-1,
59         a.uF(i) = a.uF(i-1);
60     end
61 end
```

```

62 % uFF
63 if a.luFF>0,
64     a.uFF(a.iuFF) = uu(a.puFF+1:a.puFF+a.luFF);
65 end
66 for i=1:2*ns
67     if a.update_uFF(i)==-1,
68         a.uFF(i) = a.uFF(i-1);
69     end
70 end
71 if a.luS>0,
72     a.uS(a.iuS) = uu(a.puS+1:a.puS+a.luS);
73 end
74 for i=1:2*ns
75     if a.update_uS(i)==-1,
76         a.uS(i) = a.uS(i-1);
77     end
78 end
79
80 %r=a.key-a.nxx;
81 %a.xx(1:a.nc)=uu(r+1:r+a.nc); %mhi 31Aug07
82 %a.xx(a.nxx) =uu(a.key); %ga
83
84 n = (a.ncol+2)*a.ns;
85 z = uu(a.ndof+1:a.ndof+n*a.nz); %ndof is the degree of freedom
86 a.Z = reshape(z,n,a.nz);
87 a = AssignRate(a);

```

## C.15 **Jacobian.m**

```

1 function J = Jacobian(a,xx,aeta,cp,M)
2 % Function J=Jacobian(a,xx,aeta,cp)
3 %
4 % Function for numerical differentiation of R
5 %
6 nxx = a.nxx;
7 h = (1+abs(xx))*1.e-8; % Perturbation size for each variable
8 J = zeros(nxx,nxx);
9 Rnom = feval(a.kinetics,xx,aeta,cp,M,a.Tref,a.pTot);
10 xp = xx;
11 for jj=1:nxx
12     xp(jj) = xx(jj)+h(jj);

```

```

13     Rp = feval(a.kinetics, xp, aeta, cp, M, a.Tref, a.pTot);
14     J(:, jj) = (Rp - Rnom) / h(jj);
15     xp(jj) = xx(jj);
16 end

```

## C.16 intMolarMass.m

```

1 function MolarMass = intMolarMass()
2 % Calculating molecular weight for CO, CO2, H2, H2O and paraffins and
3 % olefins from C1 to C15 in whole numbers
4
5 %Written by: Martin S. Foss, Spring 2013
6 %-----%
7 massH = 1; massC = 12; massCO = 28; massCO2 = 44; massH2 = 2; ...
8     massH2O = 18;
9 massParaffins = []; massOlefins = [];
10
11 for i = 1:15
12     massParaffins(i,1) = i*massC + (2*i+2)*massH;
13     massOlefins(i,1) = i*massC + 2*i*massH;
14 end
15 MolarMass = [massCO;
16             massCO2;
17             massH2;
18             massH2O;
19             massParaffins;
20             massOlefins(2:end)];
21 return

```

## C.17 deltaHrx.m

```

1 function deltaH = deltaHrx(r)
2 % Input
3 %   R - rate [kmole/(m^3*s)]
4 % Output
5 %   deltaH - Heat of reaction []
6 % Written by: Martin S. Foss, Spring 2013
7 %-----%
8 % Heat of reactions C1 - C5 (paraffins and olefins)

```

```
9 CH4 = -206124.000000000; % [kJ/kmol]
10 C2H6 = -173593.000000000; % [kJ/kmol]
11 C3H8 = -165854.000000000; % [kJ/kmol]
12 C4H10 = -162771.500000000; % [kJ/kmol]
13 C5H12 = -162142.000000000; % [kJ/kmol]
14 C6H14 = -159105.666666667; % [kJ/kmol]
15 C7H16 = -158065.428571429; % [kJ/kmol]
16 C8H18 = -157297.750000000; % [kJ/kmol]
17 C9H20 = -156689.555555556; % [kJ/kmol]
18 C10H22 = -156203.000000000; % [kJ/kmol]
19 C11H24 = -155814.000000000; % [kJ/kmol]
20 C12H26 = -155481.500000000; % [kJ/kmol]
21 C13H28 = -155200.153846154; % [kJ/kmol]
22 C14H30 = -154959.000000000; % [kJ/kmol]
23 C15H32 = -154756.666666667; % [kJ/kmol]
24
25 C2H4 = -105059.099609375; % [kJ/kmol]
26 C3H6 = -124414.333333333; % [kJ/kmol]
27 C4H8 = -135451.250000000; % [kJ/kmol]
28 C5H10 = -137579.800000000; % [kJ/kmol]
29 C6H12 = -140212.000000000; % [kJ/kmol]
30 C7H14 = -140129.428571429; % [kJ/kmol]
31 C8H16 = -145307.749023438; % [kJ/kmol]
32 C9H18 = -142734.000000000; % [kJ/kmol]
33 C10H20 = -144404.000000000; % [kJ/kmol]
34 C11H22 = -144396.727272727; % [kJ/kmol]
35 C12H24 = -145015.666666667; % [kJ/kmol]
36 C13H26 = -145531.692307692; % [kJ/kmol]
37 C14H28 = -145988.285714286; % [kJ/kmol]
38 C15H30 = -146384.000000000; % [kJ/kmol]
39
40 H = [CH4 C2H6 C3H8 C4H10 C5H12 C6H14 C7H16 C8H18 C9H20 C10H22 ...
      C11H24...
41      C12H26 C13H28 C14H30 C15H32 C2H4 C3H6 C4H8 C5H10 C6H12 C7H14 ...
      C8H16...
42      C9H18 C10H20 C11H22 C12H24 C13H26 C14H28 C15H30]; % [kJ/kmol]
43
44 deltaH = H*r(5:end);
45 return
```



## C.18 mid.m

```
1 function mn = mid(a, str)
2 %
3 % Calcualtes the mean over the path
4 %
5 sum = 0;
6 dxi = pathget(a, 'dxi');
7 %if sum(dxi)
8 uX = pathget(a, str);
9 for i=1:a.ns
10     sum = sum+dxi(i)*0.5*(uX(2*i-1)+uX(2*i));
11 end
12 mn = sum;
```

## C.19 AssignRate.m

```
1 function a = AssignRate(a)
2 for s=1:a.ns
3     uA = a.uA(s);
4     for j=1:a.ncol+2
5         k = (s-1)*(a.ncol+2)+j;
6         xx = a.Z(k,1:a.nxx)';
7         cp = a.cp*xx(1:a.nc);
8         [Ri, alphai] = feval(a.kinetics, xx, uA, cp, a.M', a.Tref, a.pTot);
9         a.R(k,:) = Ri';
10        a.alpha(k,:) = alphai';
11    end
12 end
```

## C.20 pathget.m

```
1 function pv = get(a, pn)
2 % Overload function PHASEPATH\GET
3 % FLUIDPATH/GET get properties of the a FLUIDPATH object
4 % This file is an internal helper function to get object ...
   properties.
5 s = ['pv=a.', pn, ''];
6 eval(s);
7 return
```

## C.21 MmCpMatrixLump.m

```

1 %% Calculation of molecular weight, Cp and liquid fraction of lumps
2 %% at different temperatures and pressures
3 % Produces 'MolarMassLump', 'CpMatLump.txt' and 'liqFracMatLump.txt'
4
5 % Written by: Martin S. Foss, Spring 2013
6 %-----%
7 clear all
8 clc
9
10 Tvec = [200 205 210 215 220 225 230]; %% [C]
11 Pvec = [1.5e6 2.0e6 2.5e6 2.7e6]; %% [Pa]
12 CpMat = [];
13 liqFracMat = [];
14
15 k = 1;
16 for i=1:length(Tvec)
17     for j = 1:length(Pvec)
18         [M, cpnom, liqFrac] = ...
19             uniSimPropertiesTodicLump(Tvec(i),Pvec(j));
20         CpMat(:,k) = cpnom; %% [kJ/(kg*K)]
21         liqFracMat(:,k) = liqFrac;
22         k = k + 1;
23     end
24 end
25 dlmwrite('CpMatLump.txt',CpMat,'delimiter','\t','precision', '%.16f')
26 dlmwrite('liqFracMatLump.txt',liqFracMat,'delimiter','\t', ...
27         'precision', '%.16f')
28 dlmwrite('MolarMassLump.txt',M,'precision', '%.16f')
29 % type MolarMassLump.txt
30 % type CpMatLump.txt

```

## C.22 uniSimPropertiesTodicLump.m

```

1 function [M, Cp, liqFrac] = uniSimPropertiesTodicLump(T,P)
2 % Function extracting the heat capacity for the components in ...
3 % comstring
4 % from FlowPropertiesTodicLump.usc (Unisim file) at temperature T ...
5 % and pressure P.

```

```
4 % compString = ...
   {'CO','CO2','H2','H2O','C1','C2','C3-4','C5-10','C11+'};
5 % Input:
6 %   T - Temperature [C]
7 %   P - Pressure [Pa]
8
9 % Output:
10 %   Cp - Heat capacity [kJ/(kg*K)]
11
12 % Assumed weight fraction in lumps:
13 %   Equal amounts (mole fraction) of each component
14
15 % Written by: Martin S. Foss, Spring 2013
16 %-----%
17 serv = actxserver('UniSimDesign.Application');
18 usdCase = serv.ActiveDocument;
19
20 sheet = usdCase.Flowsheet;
21 stream = sheet.MaterialStreams;
22
23 compString = {'CO','CO2','H2','H2O','C1','C2','C3-4','C5-10','C11+'};
24 Cp = zeros(length(compString),1);
25 M = zeros(length(compString),1);
26 liqFrac = zeros(length(compString),1);
27
28 for i = 1:length(compString)
29     stream.Item(compString{i}).Pressure.Value = P/1000; %[kPa]
30     stream.Item(compString{i}).Temperature.Value = T; %[C]
31
32     Cp(i,1) = stream.Item(compString{i}).MassHeatCapacityValue;
33     M(i,1) = stream.Item(compString{i}).MolecularWeightValue;
34     liqFrac(i,1) = stream.Item(compString{i}).LiquidFractionValue;
35 end
36 return
```

## C.23 FTplotting.m

```
1 %% Plotting of data
2 clear all
3 fileName = {'1stage_uM.mat'
4             '1stage_uM_uH_uT.mat'
5             '1stage_uM_uH_uT_sigma1.mat'
6             '1stage_uM_uH_uT_sigma08.mat'
7             '1stage_uM_uH_uT_uF_H2CO07.mat'
8             '1stage_uM_uH_uT_uF_sigma1_H2CO07.mat'
9             '1stage_uM_uH_uT_uF_sigma08_H2CO07.mat'
10            '2stage_uM.mat'
11            '2stage_uM_uH.mat'
12            '2stage_uM_uH_uT.mat'
13            '2stage_uM_uH_uT_cstr.mat'
14            '2stage_uM_uH_uT_cstr2.mat'
15            '2stage_uM_uH_uT_cstr3.mat'
16            '2stage_uM_uF_uHuTfixed_H2CO157_2.mat'
17            '3stage_uM_uF_uHuTfixed_H2CO120_2.mat'
18            '3stage_uM_uF_uHuTfixed_H2CO125_2.mat'
19            '3stage_uM_uH_uT.mat'
20            '3stage_uM_uH_uT1_uF_uS07_H2CO125.mat'
21            '3stage_uM_uH_uT1_uF_uS1_H2CO125.mat'
22            '3stage_uM_uH_uT_uF1st_uS07_H2CO07.mat'
23            '3stage_uM_uH_uT_uF1st_uS07_sigma1_H2CO07.mat'
24            '3stage_uM_uH_uT_uF1st_uS07_sigma08_H2CO07.mat'
25            '3stage_uM_uH_uT_uF_sigma08_H2CO07.mat'};
26
27 figureName = {'Case1:1stage_uM'
28              'Case2:1stage_uM_uH_uT'
29              'Case3:1stage_uM_uH_uT_sigma1'
30              'Case4:1stage_uM_uH_uT_sigma08'
31              'Case5:1stage_uM_uH_uT_uF_H2CO07'
32              'Case6:1stage_uM_uH_uT_uF_sigma1_H2CO07'
33              'Case7:1stage_uM_uH_uT_uF_sigma08_H2CO07'
34              'Case8:2stage_uM'
35              'Case9:2stage_uM_uH'
36              'Case10:2stage_uM_uH_uT'
37              'Case11:2stage_uM_uH_uT_cstr'
38              'Case12:2stage_uM_uH_uT_cstr2'
39              'Case13:2stage_uM_uH_uT_cstr3'}
```

```

40         'Case14:2stage_uM_uF_uHuTfixed_H2CO157_2'
41         'Case15:3stage_uM_uF_uHuTfixed_H2CO120_2'
42         'Case16:3stage_uM_uF_uHuTfixed_H2CO125_2'
43         'Case17:3stage_uM_uH_uT'
44         'Case18:3stage_uM_uH_uT1_uF_uS07_H2CO125'
45         'Case19:3stage_uM_uH_uT1_uF_uS1_H2CO125'
46         'Case20:3stage_uM_uH_uT_uF1st_uS07_H2CO07'
47         'Case21:3stage_uM_uH_uT_uF1st_uS07_sigma1_H2CO07'
48         'Case22:3stage_uM_uH_uT_uF1st_uS07_sigma08_H2CO07'
49         'Case23:3stage_uM_uH_uT_uF_simga08_H2CO07'};
50
51 n = length(fileName);
52 for k=1:n
53     clear a
54     load(fileName{k});
55     h = plotFT(a);
56     set(gcf, 'Name', figureName{k});
57 end

```

## C.24 plotFT.m

```

1 function err = plotFT(a)
2 defaultPlotSettings
3 figure()
4 err=0;
5 clf;
6 xi = a.xi;
7 ns = a.ns;
8 nc = a.nc;
9 x = [xi(1)];
10 for i=2:ns,
11     x = [x xi(i) xi(i)];
12 end
13 x = [x xi(ns+1)];
14 uM = a.uM;
15 uF = a.uF;
16 uF = (uF);
17 uH = a.uH;
18 uH = (uH);
19 uT = a.uT;
20 uT = (uT); uT = (1+uT)*a.Tref-273;

```

```

21 uA = a.uA;
22 uA = (uA);
23 uF = a.uF;
24 uF = (uF);
25
26 xc = a.x;
27 Z = a.Z;
28 R = a.R;
29
30 colorSet5 = [
31     0     0  1.0000
32     0  0.5000     0
33     0  0.7500  0.7500
34  1.0000     0     0
35  0.5451  0.2706  0.0745
36  0.5     1     0
37  1.0000  0.5490     0
38     0     0     0
39  0.6275  0.1255  0.9412];
40
41 %%
42 subplot(2,2,1)
43     set(gca, 'ColorOrder', colorSet5);
44     hold on
45     p1 = plot(xc, Z(:,1:nc)*100);
46     grid
47     axis([0 1 0 100])
48     tit = title('Mass fraction');
49     set(tit, 'interpreter', 'latex', 'VerticalAlignment', ...
50         'baseline');
51     leg = legend('CO', 'CO$_2$, 'H$_2$, 'H$_2$O', ...
52         'CH$_4$, 'C$_2$H$_6$, 'C$_{3-4}$', 'C$_{5-10}$', ...
53         'C$_{11+}$');
54     set(leg, 'interpreter', 'latex')
55     ylab = ylabel('$\omega$ [wt%]');
56     set(ylab, 'interpreter', 'latex');
57
58 subplot(2,2,3)
59     set(gca, 'ColorOrder', colorSet5);
60     hold on
61     p3 = plot(xc, R(:,1:nc));
62     grid

```

```

60         axis([0 1 -1.5 1])
61         ylab = ylabel('R [kg/(m$^3$ s)]');
62         set(ylab, 'interpreter', 'latex');
63         tit = title('Reaction rate');
64         set(tit, 'interpreter', 'latex', 'VerticalAlignment', ...
              'baseline');
65         xlab = xlabel('$\xi$');
66         set(xlab, 'interpreter', 'latex');
67
68 subplot(2,2,2)
69         p2 = plot(x, uM, '-.x, uH, '-.x, uA, '-.x, uF, '---');
70         grid
71         xlim([0 1])
72         tit = title('Design functions');
73         set(tit, 'interpreter', 'latex', 'VerticalAlignment', ...
              'baseline');
74         leg = legend('$u_{\mathrm{M}}$', '$u_{\mathrm{H}}/40$', ...
                      '$u_{\mathrm{A}}$', '$u_{\mathrm{F}}$');
75         set(leg, 'interpreter', 'latex');
76         box off
77
78 subplot(2,2,4)
79         p4 = plot(xc, (Z(:,nc+1)+1)*a.Tref-273, x, uT);
80         grid
81         axis([0 1 200 250])
82         ylab = ylabel(['$T$ $^\circ\text{C}$']);
83         set(ylab, 'interpreter', 'latex');
84         tit = title('Temperature');
85         set(tit, 'interpreter', 'latex', 'VerticalAlignment', ...
              'baseline');
86         leg = legend('$T$', '$T_{\mathrm{W}}$');
87         set(leg, 'interpreter', 'latex');
88         xlab = xlabel('$\xi$');
89         set(xlab, 'interpreter', 'latex');
90         box off

```



**C.25 defaultPlotSettings.m**

```
1 %% Script defining plot settings
2 % Default axis fonts
3 set(0, 'DefaultAxesFontName', 'Times New Roman')
4 set(0, 'DefaultAxesFontSize', 24)
5
6 % Default text fonts
7 set(0, 'DefaultTextFontName', 'Times New Roman')
8 set(0, 'DefaultTextFontSize', 24)
9
10 % Default line width
11 set(0, 'DefaultLineLineWidth', 2.5)
```



## **D Health, Security and Environment**

In this appendix the health, security and environmental aspects of the work performed with the thesis is assessed. In the following pages the pre-study risk assessment is attached. This work included “Mapping of high risk activity” and “Risk assessment”, respectively. There were not found any high risk activities associated with the work.


NTNU	Kartlegging av risikofylt aktivitet				Utbetalt av	Nummer	Dato
					HMS-avd.	HMSRV/2601	22.03.2011
HMS					Godkjent av	Side	Erstatter
	Rektor	1 av 1	01.12.2008				

Dato: 14.01.2013

Enhet: Institutt for Kjemisk Prosess teknologi/Department of Chemical Engineering  
 Deltakere ved kartleggingen (m/ funksjon): Martin S. Foss

Kort beskrivelse av hovedaktivitet/hovedprosess: Programmering og simulering ved bruk av datamaskin

ID nr.	Aktivitet/prosess	Ansvarlig	Eksisterende dokumentasjon	Eksisterende sikringstiltak	Lov, forskrift o.l.	Kommentar
	Programmering of simulering	Magne Hillestad	-	-	-	Ingen risikofylt aktivitet

NTNU		Risikovurdering		utarbeidet av		Nummer		Dato	
				HMS-avd.		HMSRV2003		04.02.2011	
HMS/IKS				godkjent av		1 av 2		Erstatter	
				Rektor				9.2.2010	



Dato: 14.01.2013


Enhet: Institutt for Kjemisk Prosess teknologi/Department of Chemical Engineering  
 Linjeleder: Øyvind Gregersen  
 Deltakere ved risikovurderingen (m/ funksjon): Martin S. Foss

ID nr.	Aktivitet fra kartleggings-skjemaet	Mulig uønsket hendelse/belastning	Vurdering av sannsynlighet (1-5)	Vurdering av konsekvens:				Risiko-verdi	Kommentarer/status Forslag til tiltak
				Menneske (A-E)	Ytre miljø (A-E)	Øk/ materiell (A-E)	Om-dømme (A-E)		
	Arbeid med PC; programmering og simulering	Rygg- og nakkeproblemer, underarmsplager	1	A	A	A	A	1	Hyppige pauser. Dette arbeid vil i svært liten grad være skadelig for studenten.

**Sannsynlighet**  
 1. Svært liten  
 2. Liten  
 3. Middels  
 4. Stor  
 5. Svært stor

**Konsekvens**  
 A. Svært liten  
 B. Liten  
 C. Moderat  
 D. Alvorlig  
 E. Svært alvorlig

**Risikoverdi (beregnes hver for seg):**  
 Menneske = Sannsynlighet x Konsekvens  
 Ytre miljø = Sannsynlighet x Konsekvens  
 Økonomi/materiell = Sannsynlighet x Konsekvens  
 Omdømme = Sannsynlighet x Konsekvens

NTNU		Risikovurdering		Utarbeidet av	Nummer	Dato
				HMS-avd.	HMSRV/2603	04.02.2011
HMS/KS				godkjent av		Erstatter
				Rektor	2 av 2	9.2.2010



**Sannsynlighet vurderes etter følgende kriterier:**

Svært liten 1	Liten 2	Middels 3	Stor 4	Svært stor 5
1 gang pr. 50 år eller sjeldnere	1 gang pr. 10 år eller sjeldnere	1 gang pr. år eller sjeldnere	1 gang pr. måned eller sjeldnere	Sjker, ukentlig

**Konsekvens vurderes etter følgende kriterier:**

Gradering	Menneske	Ytre miljø Vann, jord og luft	Øk/materiell	Omdømme
<b>E</b> Svært Alvorlig	Død	Svært langvarig og ikke reversibel skade	Drifts- eller aktivitetstans > 1 år.	Troverdighet og respekt betydelig og varig svekket
<b>D</b> Alvorlig	Alvorlig personskade. Mulig utærnet.	Langvarig skade. Lang resitusjonstid	Driftstans > ½ år Aktivitetstans i opp til 1 år	Troverdighet og respekt betydelig svekket
<b>C</b> Moderat	Alvorlig personskade.	Mindre skade og lang resitusjonstid	Drifts- eller aktivitetstans < 1 mnd	Troverdighet og respekt svekket
<b>B</b> Liten	Skade som krever medisinsk behandling	Mindre skade og kort resitusjonstid	Drifts- eller aktivitetstans < 1 uke	Negativ påvirkning på troverdighet og respekt
<b>A</b> Svært liten	Skade som krever førstehjelp	Ubetydelig skade og kort resitusjonstid	Drifts- eller aktivitetstans < 1 dag	Liten påvirkning på troverdighet og respekt

**Risikoverdi = Sannsynlighet x Konsekvens**

Beregn risikoverdi for Menneske. Enheten vurderer selv om de i tillegg vil beregne risikoverdi for Ytre miljø, Økonomi/materiell og Omdømme. I så fall beregnes disse hver for seg.

**Til kolonnen "Kommentarer/status, forslag til forebyggende og korrigerende tiltak":**

Tiltak kan påvirke både sannsynlighet og konsekvens. Prioriter tiltak som kan forhindre at hendelsen inntreffer, dvs. sannsynlighetsreducerende tiltak foran skjerpet beredskap, dvs. konsekvensreducerende tiltak.