NTNU
Norwegian University of
Science and Technology

Optimization based design of an IRCC process with CO2 capture

Erik Lien Johnsen

Industrial Economics and Technology Management

Submission date:	June 2011
Supervisor:	Bjørn Nygreen, IØT
Co-supervisor:	Professor Truls Gundersen (EPT), EPT Researcher Rahul Anantharaman, SINTEF Energy Research

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

NTNU

Det skapende universitet

MASTERKONTRAKT

- uttak av masteroppgave

1. Studentens personalia

Etternavn, fornavn **Johnsen, Erik Lien**	Fødselsdato **07. jul 1986**
E-post **erik.lien.johnsen@gmail.com**	Telefon **99541980**

2. Studieopplysninger

Fakultet **Fakultet for Samfunnsvitenskap og teknologiledelse**	
Institutt **Institutt for industriell økonomi og teknologiledelse**	
Studieprogram **Industriell økonomi og teknologiledelse**	Hovedprofil **Anvendt økonomi og optimering**
E-post **erik.lien.johnsen@gmail.com**	Telefon **99541980**

3. Masteroppgave

Oppstartsdato **17. jan 2011**	Innleveringsfrist **13. jun 2011**
Oppgavens ~~(foreløpige)~~ tittel **Optimization based design of an IRCC process with CO2 capture**	
Oppgavetekst/Problembeskrivelse The purpose of this work is to find a more efficient design of an energy conversion process with CO2 capture by optimizing process integration, and to examine whether an approach with metamodeling is suited for this task. The main focus is on optimizing an IRCC process using simplified relations. Department of Energy and Process Engineering (EPT) – NTNU and SINTEF Energy Research will provide some of the needed technical information about the IRCC process and actual data to use in testing of the chosen approach. Main contents: 1. Describe the design problem. 2. Formulate one or more models for the design problem. 3. Implement the one or more sub model(s) for the problem by use of appropriate software. 4. Test the model(s) with real data. 5. Discuss the applicability and adequacy of the model(s) as decision support tools for the given problem.	
Hovedveileder ved institutt **Professor Nygreen Bjørn**	Biveileder(e) ved institutt **Professor Truls Gundersen (EPT)**
Ekstern bedrift/institusjon **SINTEF Energy Research**	Ekstern veileder ved bedrift/instutisjon **Researcher Rahul Anantharaman**
Merknader **1 uke ekstra p.g.a påske.**	

ELJ

4. Underskrift

Student: Jeg erklærer herved at jeg har satt meg inn i gjeldende bestemmelser for mastergradsstudiet og at jeg oppfyller kravene for adgang til å påbegynne oppgaven, herunder eventuelle praksiskrav.

Partene er gjort kjent med avtalens vilkår, samt kapitlene i studiehåndboken om generelle regler og aktuell studieplan for masterstudiet.

NTNU 13/1-2011

Sted og dato

Student

Hovedveileder

Originalen oppbevares på fakultetet. Kopi av avtalen sendes til instituttet og studenten.

Abstract

To deal with the threat of climate change, many technologies should be investigated, and power generation through an IRCC with CO_2 capture is one alternative. However, capturing CO_2 has a negative effect on the efficiency of the process as it requires a lot of energy. In this work, we try to reduce the energy consumption of an IRCC process with CO_2 capture by developing a tool for finding the optimal process design with extensive heat integration.

The design of an IRCC process involves many parameters which interfere in complex relationships. In this report, an MINLP model is established for optimizing important parameters simultaneously. The model relies on metamodeling based on process simulations in Aspen HYSYS to approximate difficult correlations, combined with a more direct approach for modeling computationally easier parts of the process. A general model for heat recovery targeting is developed for the heat integration optimization, and implemented as a part of the full IRCC optimization model.

The global solver BARON is used for solving the problem, together with a relaxation procedure based on pinch analysis insights, and optimal solutions are usually found within several hours.

The optimized IRCC process reaches a net electric efficiency of 49.97 %, assuming maximum heat integration, with only 1 % of the cooling and heating demands to be covered by utilities. The accuracy of the model is relatively good when compared to process simulations, but a less idealistic version of the IRCC should be designed based on the results to confirm the capability of the model.

Preface

This work is my master thesis in the master programme of industrial economics at the Norwegian University of Science and Technology (NTNU). As my technological specialization is in process engineering, and my economical specialization is in optimization, I have chosen a project in which I can use my knowledge from both fields.

Combining the two disciplines have been rewarding, as large parts of my education have come to use, but it has also been challenging. I have had to balance between different conventions, learn new software and dive deeper into the field of non-linear optimization.

I started this work with a project thesis in the previous semester, and I want to mention that the sections 2.3 and 3.1 and parts of the introduction are modifications of text from this prestudy. Except for these parts, which are included for completeness, and some of the preface, this master thesis is new material.

I would like to thank my supervisor, Professor Bjørn Nygreen at the Department of Industrial Economics and Technology Management, for letting me work with this project, for guidance to the project in general and for sharing his optimization expertise.

I am also thankful to Professor Truls Gundersen at the Department of Energy and Process Engineering for coming up with the project description and initiating the collaboration with Rahul Anantharaman, Bjørn Nygreen, him and me, and for his useful comments and insights.

Finally, I will thank researcher Rahul Anantharaman at SINTEF Energy Research for answering my questions whenever I dropped into his office.

He has contributed to this work by setting up the flow sheet in HYSYS, providing background information and reviewing my work with a critical but always positive vision. It has been encouraging to hear that he intend to publish an article based on my work[1].

Trondheim, 14 June 2010

Erik Lien Johnsen

[1] for the ESCAPE 22 conference

Contents

Chapter 1

Introduction

Climate change due to anthropogenic emissions of greenhouse gases (GHG) is accepted as a serious threat, and the increase of CO_2 in the atmosphere is the main contributor to the increased greenhouse effect. There are many ways to reduce the amount of CO_2 in the atmosphere, and CO_2 capture and storage (CCS) is one way which can be cost effective compared to other techniques (David and Herzog, 2000; Metz et al., 2005). The process of CCS is normally divided into three steps: Capture, transportation and storage. Of these, the capture is the main element of the CCS cost (Gibbins and Chalmers, 2008).

Power generation from fossil fuels constitute a substantial share of the man-made GHG emissions, and most of the large emitters are such power plants, making them well suited for CCS. Capturing CO_2 requires a lot of energy, leading to reduced efficiency and increased cost for the power plant. Thus, an energy-efficient design of the plant with CO_2 capture is important to make CSS more attractive.

The energy demand of a process can be reduced by applying heat integration in an optimal way, but this is a complex task. Mathematical programming (MP) — or optimization — is a tool suited for this exercise, and it has to some extent been successfully applied. However, the approach has usually been to match heat sources and sinks, whose temperatures and duties are given, while better solutions could be obtained by not fixing the temperatures and duties in advance. This complicates the model by introducing the need for non-linear relations, which can lead to computational difficulties.

Optimization can also find the most efficient process design, both with respect to heat integration and for fulfilling the main purpose of the process — power generation. However, a mathematical description of the process may be highly nonlinear and very complex in terms of optimization. A way to reduce these problems is approximating difficult correlations by simpler ones based on data from process simulations, and if the approximation is chosen carefully this approach can be fruitful (Palmer and Realff, 2002b).

The goal of this work is to establish an optimization model for an integrated reforming combined cycle (IRCC) process with CO_2 capture, which can find process designs with optimal integration such that the energy demand of the process is reduced. By optimizing the main process and heat integration simultaneously, better solutions can be found than with a sequential approach. To model such a complex process, it is important to find a good balance between accuracy and solvability by approximating some correlations and making simplifying assumptions, without over-simplifying.

Chapter 2

The IRCC process with CO_2 capture

In this chapter, the integrated reforming combined cycle process with CO_2 capture is described, starting with a short introduction to CO_2 capture, followed by an overview of IRCC design, defining the scope of this work from a process engineering point of view. Then the unit operations and equipment that make up the process are described more in detail. IRCC process designs from the literature are described shortly for comparison in section 5.4.

2.1 CO_2 capture

Carbon capture technologies are divided into three categories, namely pre-combustion, post-combustion and oxy-combustion. The integrated reforming combined cycle is of the pre-combustion type, in which the carbon dioxide, which normally is a product from combustion, is separated from the fuel before the main combustion occurs. The IRCC is one of many ways of generating power from natural gas (NG) while capturing CO_2. A much more thorough description of CCS is found in Bolland (2010).

2.2 IRCC design

Reforming is the process of converting natural gas into syngas, and this takes place in a reformer. Syngas is a mixture of carbon monoxide (CO), hydrogen gas (H_2) and CO_2. For CO_2 capture the syngas is further processed such that the CO and steam (H_2O) is converted to more CO_2 and H_2 in one or several water-gas shift (WGS) reactors. The CO_2 is separated in what we here call the CO_2 capture unit (CCU), which normally is an absorption process. The hydrogen can be used for many purposes, but in the IRCC the hydrogen-rich gas is combusted and the exhaust is expanded in a turbine to generate electricity. The heat in the exhaust is then utilized by a heat recovery steam generator (HRSG).

Nord and Bolland (2011) point out six important aspects of designing an IRCC process with CO_2 capture:

1. Which type of reformer to use, e. g. steam reformer, partial oxidation or autothermal reformer
2. What pressure level to adopt
3. Design of the heat recovery steam generator
4. Level of heat integration
5. Use of membranes in the reforming process, in the water-gas shift section or for CO_2 capture
6. Method for CO_2 capture

Regarding aspect number one, we assume an autothermal reformer to be used. For the last two aspects, membrane technologies are not used, and an amine-based absorption process is assumed for CO_2 capture. Aspects number two, three and four are addressed in this work. The pressure level is one of the parameters we try to optimize. We do also try to optimize many parameters not mentioned by Nord and Bolland (2011), but the aim of this is related to heat integration, where we try to reach a very high level. This also relates to the HRSG, as increased power output from the steam cycles is one of the goals for the heat integration. However, we do not try to optimize the pressure levels of the HRSG

2.3 Unit operations and equipment[1]

Figure 2.3 is a simple flow-sheet of our process, where the type of reformer and CCU is fixed, and some assumptions are made for the HRSG design. The unit operations here are explained in the following subsections. There should also be a number of heat exchangers, compressors and other unit operations, like in the more complete IRCC process depicted in Nord et al. (2009), but we are here only considering the most important ones.

2.3.1 Auto-thermal reformer (ATR)

The reforming of natural gas to syngas can be done in several ways. In the process examined in this report, it is done in an auto-thermal reformer (ATR). "Auto-thermal" refers to that the amount of oxygen is set such that no external heat supply is needed to keep the temperature at a specified level. The reactions taking place in the ATR are

$$CH_4 + \frac{1}{2}O_2 \rightarrow CO + 2H_2 \quad \Delta H = -35.9 \text{ [kJ/mol]} \tag{2.1}$$

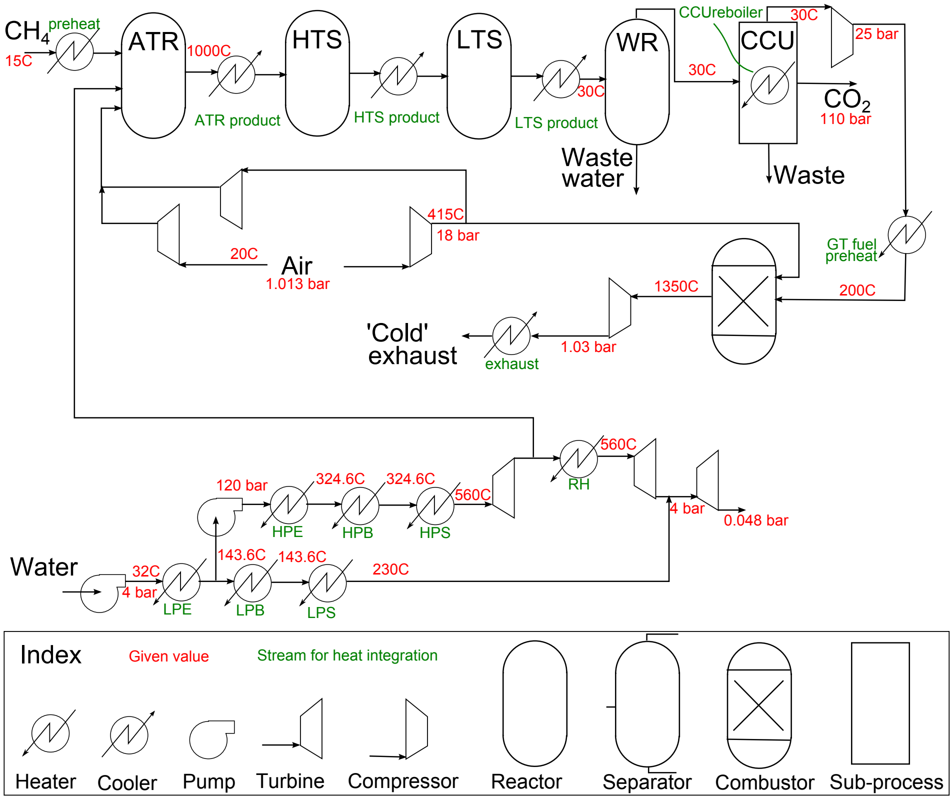
$$CH_4 + H_2O \rightleftharpoons CO + 3H_2 \quad \Delta H = 205.9 \text{ [kJ/mol]} \tag{2.2}$$

$$CO + H_2O \rightleftharpoons CO_2 + H_2 \quad \Delta H = -41.2 \text{ [kJ/mol]}. \tag{2.3}$$

The oxygen added provides heat by reaction (2.1), which is seen by the negative heat of reaction (ΔH). The extent of the other two reactions depends on how much heat is supplied by the reaction with oxygen, temperature conditions and on the amount of steam added, which is referred to by the steam-to-carbon ratio, the molar ratio of steam to methane.

Before the reformer, the natural gas feed is pre-reformed with added hydrogen such that the longer hydrocarbon chains are split into methane, but in this work the natural gas is for simplicity assumed to be pure methane.

[1] This section is to a large extent based on the authors project thesis (Johnsen, 2010).


CH4
preheat
15C
ATR
1000C
ATR product
HTS
HTS product
LTS
LTS product
30C
WR
Waste water
CCUreboiler
30C
30C
CCU
25 bar
CO2
110 bar
Waste
415C
18 bar
20C
Air
1.013 bar
GT fuel preheat
200C
1350C
'Cold' exhaust
exhaust
1.03 bar
560C
RH
120 bar
324.6C
324.6C
560C
HPE
HPB
HPS
4 bar
0.048 bar
Water
32C
4 bar
LPE
143.6C
143.6C
LPB
230C
LPS
Index
Given value
Stream for heat integration
Heater
Cooler
Pump
Turbine
Compressor
Reactor
Separator
Combustor
Sub-process

The oxygen needed in the ATR can be supplied as air or in a relatively pure form. The latter option requires an air separation unit (ASU) upstream of the reformer, while the former results in higher flow rates and larger equipment. In this work an air-blown ATR is used.

2.3.2 WGS reactors

In the WGS reactors, the water-gas-shift reaction (2.3) takes place. The equilibrium of this reaction is shifted more to the right with lower temperatures, thus cooling is applied before each of the reactors. The shift is usually done in two stages, a high temperature shift (HTS) with Fe_3O_4/Cr_2O_3 as catalyst, and a low temperature shift (LTS) with $Cu/ZnO/Al_2O_3$ as catalyst. The first catalyst normally operates at temperatures between 310 and 450 °C, and the one in the LTS at 210–240°C (Rhodes et al., 2002, 1995), but operation slightly outside these ranges should be possible.

The reaction is exothermic, and the coolers can be used to generate steam.

The water left in the stream is taken out in the water removal flash, before the gas enters the CCU.

2.3.3 CCU

In what is here called the carbon capture unit, the CO_2 is separated from the other components in the gas stream. There are many different technologies for capturing CO_2, including membranes, adsorption and distillation, but absorption processes are most widely used, usually with an amine-based solvent. The separation requires a lot of energy, and the higher the capture ratio[2], the higher the energy consumption. An economically acceptable capture ratio is around 85–95% (Bolland, 2010), and in this work 90 %is chosen as a lower limit.

The main consumer of energy in the amine absorption process is the amine reboiler. The temperature of the reboiler is typically around 120 °C,

[2] capture ratio: $\eta_{cap} = \frac{\text{amount of CO}_2\text{ captured}}{\text{amount of CO}_2\text{ formed}}$

and about 1.5 MJ of heat has to be supplied per kg of CO2 captured (Nord et al., 2009).

2.3.4 Combustor

After the CO_2 is captured, the gas consists of mainly H_2, together with N_2 since the reformer is air-blown. This gas is compressed before being fed to the combustor together with air and eventually a diluent. In the combustion chamber, the hydrogen is completely combusted with air as in reaction (2.4). There will also be some CO, CH_4, H_2O and CO_2 left in the syngas, and the former two are combusted completely as in reactions (2.5) and (2.6).

$$H_2 + \frac{1}{2}O_2 \rightarrow H_2O \qquad \Delta H = -241.8 \text{ [kJ/mol]} \tag{2.4}$$

$$CO + \frac{1}{2}O_2 \rightarrow CO_2 \qquad \Delta H = -283.0 \text{ [kJ/mol]} \tag{2.5}$$

$$CH_4 + 2\ O_2 \rightarrow 2\ H_2O + CO_2 \qquad \Delta H = -890.4 \text{ [kJ/mol]} \tag{2.6}$$

The energy released from reactions (2.4)-(2.6) is then utilized in the gas turbine.

2.3.5 Gas turbine

The gas turbine is used to generate power by expanding the combustion product of air and hydrogen-rich fuel, and the power output and outlet temperature may be calculated with the following equations:

$$\dot{W} = \eta\ \dot{m}\ c_p\ (T_{in}(1 - \left(\frac{p_{out}}{p_{in}}\right)^{\frac{\kappa-1}{\kappa}}) \tag{2.7}$$

$$T_{in} - T_{out} = \eta T_{in}(1 - \left(\frac{p_{out}}{p_{in}}\right)^{\frac{\kappa-1}{\kappa}}) \tag{2.8}$$

$\dot{W}$ denotes power, η is the efficiency, c_p is the specific heat capacity, T_{in}, T_{out}, p_{in} and p_{out} are temperatures and pressures at the inlet and outlet of the turbine, and κ is the specific heat ratio.

The large-scale gas-turbines available are designed for natural gas, and using hydrogen instead of NG is not straight-forward, but developing a new, specialized turbine is extremely expensive. One problem is that combustion of hydrogen involves much higher flame temperatures, which lead to high NO_x emissions. This is avoided by diluting the fuel, and as nitrogen is a suitable diluent(Chiesa et al., 2005), using an air-blown ATR is advantageous. However, by diluting the fuel, the volumetric heating value become lower and larger equipment or higher gas velocity is needed to generate the same amount of power as in the case of natural gas.

2.3.6 Air compressors

To make the ATR and combustor operate at the chosen pressures, air must be compressed accordingly. The large main air compressor is used to compress the combustion air to 18 bar, and may also compress all or parts of the air to the ATR, possibly followed by a second air compressor if the ATR pressure is higher than that of the combustor. We have also included the option of using an extra, smaller air compressor for supplying the ATR with compressed air.

The equations for compression work and compressor outlet temperatures are similar to those for turbines, but with inverse efficiencies:

$$\dot{W} = \frac{1}{\eta}\ \dot{m}\ c_p\ (T_{in}(\left(\frac{p_{out}}{p_{in}}\right)^{\frac{\kappa-1}{\kappa}} - 1) \tag{2.9}$$

$$T_{out} - T_{in} = \frac{1}{\eta T_{in}}(\left(\frac{p_{out}}{p_{in}}\right)^{\frac{\kappa-1}{\kappa}} - 1) \tag{2.10}$$

Large-scale air compressors and gas turbines are only available in certain sizes, and thus the process has to be dimensioned to match the size of this equipment. A typical large-scale air compressor can handle 640 kg/s of air, which is the size we assume in this work.

2.3.7 HRSG and steam turbines

Energy left in the flue gas is exploited in a heat recovery steam generator, driving one or several steam turbines. Usually there is an arrangement including high and low pressure steam, and sometimes an intermediate pressure level too. Steam is heated in the economizer part, evaporated in the boiler and then superheated before being expanded in the turbine, and a pump is used to increase pressure. The heating of steam can also be integrated with heat sources in other parts of the process.

HRSG design have a large impact on the efficiency of the IRCC power plant, but as it is a complex task in itself this work is limited by looking at two predefined steam levels, low pressure (LP) steam at 4 bar and high pressure (HP) steam at 120 bar. For both pressure levels, water is pumped to the chosen pressure, then heated up to saturation, boiled, and superheated to 230 and 560 °C. Steam to the ATR is extracted from the HP turbine, and the steam left is then reheated to 560 °C and expanded in an intermediate pressure (IP) turbine down to the LP level, where it is expanded again together with the LP steam down to 0.048 bar.

Chapter 3

The IRCC optimization model

In this chapter, a model for optimizing an IRCC process with CO_2 capture is presented, but first we look at similar work already done by others and provide an overview for the decisions made for modeling this process.

3.1 Related work[1]

Much of the optimization work in process engineering is of the simple type where one variable is analyzed at a time, like the recent study of Emun et al. (2010) on an integrated gasification combined cycle (IGCC) process, which is the coal equivalent of the IRCC, using Aspen Plus for simulation. Their work did not include CO_2 capture, but some heat integration was examined.

Martelli (2010) applies optimization to a greater extent, optimizing the design of heat recovery steam cycles. His model is also applied on an IGCC process with CO_2 capture, and show promising results. The non-linear functions are linearized in the optimization, and temperatures and duties

[1] This section is to a large extent based on the authors project thesis (Johnsen, 2010).

are fixed in the heat integration part.

More high-level integration is studied by Sadhukhan and Zhu (2002), which formulate their model for integrating gasification in an overall refinery as a mixed integer non-linear program. They decompose the problem into several steps, including a “site-level” and a “process-level” optimization problem, and solve them heuristically.

Palmer and Realff (2002a,b) have used non-linear functions from regression approximating difficult correlations in their optimization, and their model is tested on an ammonia synthesis process. The model based on polynomial regression is solved with a steepest ascent method. Their focus is on establishing a model based on very few flow-sheet simulations, for cases where simulation is time-consuming.

For optimizing heat integration simultaneously with process design, Duran and Grossmann (1986) and Grossmann et al. (1998) have developed general models which are discussed in section 3.5.

To summarize, the works dealing with the IRCC process do not apply optimization in any depth, while some more optimization is applied on similar processes. When looking at works not dealing with reforming processes, a wider range of approaches to process optimization is found, and some of the methods in the literature are applied in this work to find more efficient designs of an IRCC process with CO_2 capture.

3.2 Overview of the IRCC optimization model

Making an optimization model for the whole IRCC process is a challenging task. To see how a change in one variable effects other variables, optimization relies on a mathematical description of relationships between variables. The relationships between pressures, temperatures, mass and energy flows and so on in chemical and thermodynamic processes are often complex in themselves. Many of the relationships in the IRCC process are non-linear, and to obtain a satisfactory level of accuracy, we need a non-linear optimization model.

Chemical equilibrium reactions do often require iterative calculations, and to find the enthalpy of steam the usual approach is to interpolate between values in a steam table rather than using analytical equations. Such formulations are incompatible with most optimization software, at least for efficient solving.

In addition to this, the heat integration modeling may require binary variables together with non-linear equations as both flows ($\dot{m}c_p$) and temperatures are taken as variables.

To deal with these factors, different modeling strategies are applied to different parts of the process. For the part of the system involving equilibrium reactions, approximate relationships based on results from process simulations and regression are used, as powerful software is available for simulating the process accurately and fast, and this is presented in section 3.3. The steam turbines are also modeled in a similar manner. Where practically possible, physical relationships from thermodynamics and balance equations are used in the modeling, and this is presented in section 3.4. The heat integration modeling is based on a separate MINLP[2] formulation which is connected to the rest of the model, and this model is explained in section 3.5.1

3.2.1 Symbols and notation

In the preceding chapters, the notation common in process engineering have been used, whereas the notation we will use for optimization modeling is adapted to optimization. Capital letters denote constants, "curly" capital letters denote sets, while lowercase symbols are used for variables. In subscripts, lowercase letters are used for indices, and capital letters or numbers are part of the variable or constant name, or a specific instance of an index. Greek letters are used for constants even in lowercase. A bar above or below the symbol for a variable is used to denote upper and lower bounds, respectively.

The symbols used for the model are presented in the following table, grouped by type and generally listed in the order they appear in the fol-

[2] Mixed-Integer Non-Linear Program (optimization model with both integer variables and non-linearities)

lowing sections.

Indices

i, j - independent variable

d - dependent variable

k, l - component or substance

n - used for the coefficient before the n^{th} power of temperature in (3.12)

e - equipment

s - stream

p - stream (used for the stream making the pinch candidate in that equation)

h - hot stream (stream requiring cooling)

c - cold stream (stream requiring heating)

$in(e)$ - the inlet stream for equipment e (which has only one inlet stream, or for which the inlet streams can be considered one stream)

$out(e)$ - the outlet stream for equipment e (which has only one outlet stream)

Sets

$\mathcal{I}$ - all independent variables

$\mathcal{I}_{1,d}$ - the independent variables (used in regression) for which there are linear terms $\alpha_{d,i}$ for dependent variable d

$\mathcal{I}_{2,d}$ - the independent variables i for which there are bilinear or quadratic terms $\beta_{d,i,j}$ for dependent variable d

$\mathcal{J}_{d,i}$ - the independent variables j for which there are bilinear or quadratic terms $\beta_{d,i,j}$ for dependent variable d and independent variable i

$\mathcal{D}$ - all dependent variables

$\mathcal{D}_F$ - dependent flow variables

$\mathcal{D}_S$ - dependent state variables

$\mathcal{E}_{T0}$ - set of turbines and pumps modeled as mass flow multiplied by a constant

$\mathcal{E}_{T1}$ - set of turbines (and compressors) modeled with a linear function of feed pressure

$\mathcal{E}_{T2}$ - set of turbines and compressors modeled with equations from thermodynamics

$\mathcal{E}_{C2}$ - set of compressors modeled with equations from thermodynamics

$\mathcal{E}_R$ - set of reactors (with known extent of reaction)

$\mathcal{E}_{SP}$ - set of splitters

$\mathcal{E}_M$ - set of mixers

$\mathcal{E}_{SE}$ - set of separators modeled with fixed splitting for each component

$\mathcal{E}_P$ - set of equipment with power output

$\mathcal{E}_W$ - set of equipment requiring work input

$\mathcal{K}_s$ - set of substances in stream s

$\mathcal{K}_{F,s}$ - set of fuel components in stream s
$\mathcal{S}_1$ - set of all process streams
$\mathcal{S}_{IN,e}$ - set of streams flowing into equipment e
$\mathcal{S}_{OUT,e}$ - set of streams flowing out of equipment e

$\mathcal{S}$ - set of streams subject to heat integration
$\mathcal{S}_P$ - set of streams that might possibly make pinch
$\mathcal{H} \subset \mathcal{S}$ - set of hot streams (non-isothermal)
$\mathcal{C} \subset \mathcal{S}$ - set of cold streams (non-isothermal), $\mathcal{C} \cap \mathcal{H} = \emptyset$
$\mathcal{H}_I \subset \mathcal{S}$ - set of hot isothermal streams, $\mathcal{H}_I \cap \mathcal{H} = \emptyset$
$\mathcal{C}_I \subset \mathcal{S}$ - set of cold isothermal streams, $\mathcal{C}_I \cap \mathcal{C} = \emptyset = \mathcal{H}_I \cap \mathcal{C}_I$
$\mathcal{S}_N \subset \mathcal{S}$ - set of non-isothermal streams (hot and cold), $\mathcal{H} \cup \mathcal{C}$
$\mathcal{S}_I \subset \mathcal{S}$ - set of isothermal streams (hot and cold), $\mathcal{H}_\mathcal{I} \cup \mathcal{C}_\mathcal{I}$

$\mathcal{S}_{S,TI}$ - set of tuples (s,p) of heat integration streams s with inlet temperature equal to that of stream p
$\mathcal{S}_{S,TO}$ - set of tuples (s,p) of non-isothermal heat integration streams s with outlet temperature equal to that of stream p
$\mathcal{S}_{S,F1}$ - set of tuples (s,p) of non-isothermal heat integration streams s with heat capacity flow equal to that of stream p, modeled with a constants $C_{P,s,k}$
$\mathcal{S}_{S,F2}$ - set of tuples (s,p) of non-isothermal heat integration streams s with heat capacity flow equal to that of stream p, where $f_{C,p}$ is a variable
$\mathcal{S}_{E,Q}$ - set of tuples (s,e) of isothermal heat integration streams s with heat flow equal to that of equipment e
$\mathcal{S}_{S,M}$ - set of tuples (s,p) of non-isothermal heat integration streams s with the mass flow as isothermal heat integration stream p

Constants

$\alpha_{d,0}$ - constant term from regression
$\alpha_{d,i}$ - regression coefficient for linear term
$\beta_{d,i,j}$ - regression coefficient for bilinear (or quadratic) term $x_i\ x_j$
W_e - work or power output per unit mass flow through equipment e
$A_{W,e}$ - coefficient for pressure in equations for the work or power output of equipment e
$A_{T,e}$ - coefficient for pressure in equations for the outlet temperature of equipment e
$B_{W,e}$ - constant term in equations for the work or power output of equipment e
$B_{T,e}$ - constant term in equations for the outlet temperature of equipment e
$Q_{LHV,k}$ - Lower Heating Value for fuel component k
$Q_{s,k}$ - heat demand resulting from one unit of flow of substance k in stream s

$G_{e,k,l}$ - amount of substance k in the reaction product resulting from one unit of substance l in the inlet stream to reactor e

η_e - efficiency of equipment e

$H_{n,k}$ - coefficient for the nth power of t in the enthalpy equation for substance k

T_s - temperature of stream s

P_s - pressure of stream s

R_k - mass specific gas constant for substance k

$N_{s,k}$ - fraction of component k in stream s

$V_{e,s,k}$ - fraction of component k from the inlet of separator e going to outlet stream s

U - minimum CO_2 capture ratio multiplied by the molar mass of CO_2 divided by the molar mass of methane

$F_{s,k}$ - mass flow of substance k in stream s

$C_{P,s,k}$ - average specific heat capacity for substance k representative for the state of stream s

K_H - unit cost for the hot utility (heating supplied above pinch)

K_C - unit cost for the cold utility (cooling supplied below pinch)

K_s - cost per heat capacity flow of stream s

$T_{ADJ,s}$ - equals HRAT (heat recovery approach temperature) if s $\in \mathcal{C} \cup \mathcal{C}_I$, 0 else

$M_{1,s,p}$, $M_{2,s,p}$ - big M constants

Q_s - specific heat for isothermal stream s

Variables

z - The objective (net plant efficiency based on LHV)

z_2 - The alternative objective (net power output)

x_i, x_j, x_d - variable used in the regression[3]

w_e - work or power output for equipment e

$f_{s,k}$ - mass flow of substance k in stream s

p_s - pressure of stream s

t_s - temperature of stream s

$t_{I,e}$ - ideal temperature out of equipment e

a - exponent for the pressure ratio in (2.8)

q_e - heating demand of equipment e

q_H - hot utility consumption (external heating above pinch)

q_C - cold utility consumption (cooling below pinch)

$t_{IN,s}$ - inlet temperature for stream s

$t_{OUT,s}$ - outlet temperature for stream s

[3] The names x_i are used here for convenience, but these variables will later be referred to by different names which explain their physical interpretation.

$t_{M,s,p}$ - a "‘constructed"’ temperature between $t_{IN,s}$ and $t_{OUT,s}$ used in calculating the heat transfer above pinch candidate p

$f_{C,s}$ - heat capacity flow ($\dot{m}c_p$) for stream s

$f_{Q,s}$ - heat flow for isothermal stream s

$f_{QP,s,p}$ - heating exchanged above pinch if stream p would make the pinch for isothermal stream s

$y_{s,p}$ - binary variable, related to whether $t_{IN,s}$ is above or below $t_{IN,p}$

Named instances of indices/set members

Streams	CC	The carbon capture stream leaving the CCU
Equipment	ATR	The autothermal reformer
	CCU	The carbon capture unit
	GT	The gas turbine
Substances	C_4	Methane
	H_2O	Water/steam
	CO_2	Carbon dioxide

3.2.2 The objective function

We will start the presentation of the model with the objective function. Ideally, one would like an objective function in economic terms, e. g. maximizing the net present value of the whole plant, but there are many reasons for not using such an objective. Cost data for process equipment are generally not publicly available, and there are many factors influencing the cost of the plant, and some of these are not included in our model. In addition, prices for the outputs of the process — power and possibly avoided CO_2 emissions — are not known with certainty. Another reason is that proper modeling of investment costs and net present value would introduce more non-linearities and discrete variables.

While some economical considerations are taken in choosing bounds and parameters for the model, the objective is rather to maximize the net electric efficiency of the process:

$$\text{minimize } z \tag{3.1}$$

$$Q_{LHV,CH_4}\ f_{in(ATR),CH_4}\ z = \sum_{e \in E_P} w_e - \sum_{e \in E_W} w_e \tag{3.2}$$

An alternative is to maximize the power output for the given size of the air compressor. This is achieved by replacing z with z_2 (equation 3.3), which has the advantage of being linear.

$$z_2 = \sum_{e \in I_P} w_e - \sum_{e \in E_W} w_e \tag{3.3}$$

Before the equations are presented in the next sections, it should be noted that all variables are non-negative, and are bounded by upper and lower bounds. The complete optimization model with all constraints is presented in appendix A.

3.3 Metamodeling

Palmer and Realff (2002a,b) propose the metamodeling approach to deal with processes that are too difficult for optimization when modeled directly. The idea is to simulate the process under various conditions, and apply regression on the simulation results to obtain equations correlating the variables of the model. In Palmer and Realff (2002a) the importance of a smart sampling scheme to get as much information as possible from few simulations is highlighted, and polynomial and kriging regression models are investigated. In the second article (Palmer and Realff, 2002b) they present heuristics to solve the optimization problems.

Minimum bias Latin hypercube sampling (MBLHS; or orthogonal sampling) is recommended if the number of simulations is very low. This becomes impractical when one can afford more simulations, as it gets more difficult to find a minimum bias Latin hypercube. A regular Latin hypercube sampling (LHS) scheme is easier to implement, and still a good choice as it ensures a wide distribution of the values for each input variable in the simulation, such that LHS is a good method for varying input variables (McKay et al., 1979).

LHS is done by dividing the range for each input variable into one equally large — or equally probable — interval per simulation to be performed. Then, for each simulation, the value of each input variable is taken from a randomly selected interval under the condition that the interval is not used in any other simulation.

Choosing an appropriate regression model is critical to ensure a good description of reality without forgetting the classical trade-off between reality representation and solution speed (Nygreen et al., 1998, p. 252). The regression functions need to have many of the same characteristics as the actual correlation, but should also be relatively easy in terms of optimization. Kriging regression models may be well suited in describing the relationships between the variables in the IRCC process, but they are more complex and not ideal for global optimization. Polynomial models are more straightforward to use, and, as pointed out in detail in the project thesis (Johnsen, 2010), second order polynomials (3.4) fit the simulation results quite well.

$$x_d = \alpha_{d,0} + \sum_{i \in \mathcal{I}_{1,d}} \alpha_{d,i}\ x_i + \sum_{i \in \mathcal{I}_{2,d}} \sum_{j \in \mathcal{J}_{d,i}} \beta_{d,i,j}\ x_i\ x_j \quad \forall d \in \mathcal{D} \tag{3.4}$$

The ATR and especially the WGS reactors were considered too difficult for direct, physical modeling, and were thereby modeled with the metamodeling approach. The part of the process for which metamodeling was performed is shown in figure 3.1, where variables are indicated with blue letters, and the input variables are furthermore written in bold font.

In our model, the feed flow rate of methane should be a variable, but in the simulations this flow rate was set to a fixed value, as the correlation between variables from the regression and the methane feed flow rate is straight-forward. Energy and mass flow variables are proportional to the methane feed flow rate, while state variables like temperatures and pressures are independent of this. Thus, the resulting equations for our model are as follows:

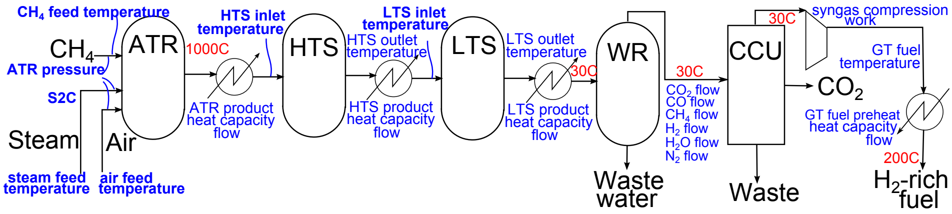


Figure 3.1: Flow-sheet for the metamodeling part of the process

$$x_d - \left(\alpha_{d,0} + \sum_{i \in \mathcal{I}_{1,d}} \alpha_{d,i}\ x_i + \sum_{i \in \mathcal{I}_{2,d}} \sum_{j \in \mathcal{J}_{d,i}} \beta_{d,i,j}\ x_i\ x_j \right) f_{in(ATR),CH_4} \\ = 0 \quad \forall d \in \mathcal{D}_F \tag{3.5}$$

$$x_d - \sum_{i \in \mathcal{I}_{1,d}} \alpha_{d,i}\ x_i - \sum_{i \in \mathcal{I}_{2,d}} \sum_{j \in \mathcal{J}_{d,i}} \beta_{d,i,j}\ x_i\ x_j = \alpha_{d,0} \quad \forall d \in \mathcal{D}_S \tag{3.6}$$

The steam turbines are also modeled with the help of simulations because of the difficulties in calculating steam properties. As the conditions of the low pressure steam turbine are set in advance, only one simulation with a unit mass flow to find the specific power output was needed, and the power output is simply calculated by multiplying this value with the actual mass flow:

$$w_e - W_e\ f_{in(e),H_2O} = 0 \quad \forall e \in \mathcal{E}_{T0} \tag{3.7}$$

For the high and intermediate pressure turbines, the intermediate pressure is a variable, such that the simple approach used for the LP steam turbine does not work. Instead, simulations are done for pressures from 18 to 30 bar to find the power output per unit mass flow and outlet temperatures of the turbines. Because the correlations between pressure and, respectively, power and temperature were not too far from linear, linear regression was chosen. These results are presented graphically in figures 3.2 and 3.3. This give us the following equations:

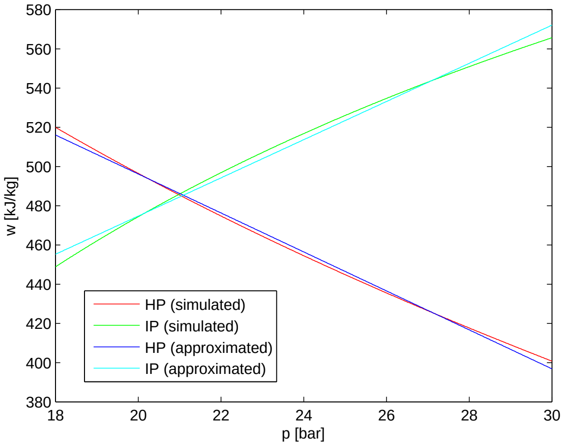


Figure 3.2: Simulated and approximated mass specific power for HP and IP steam turbines

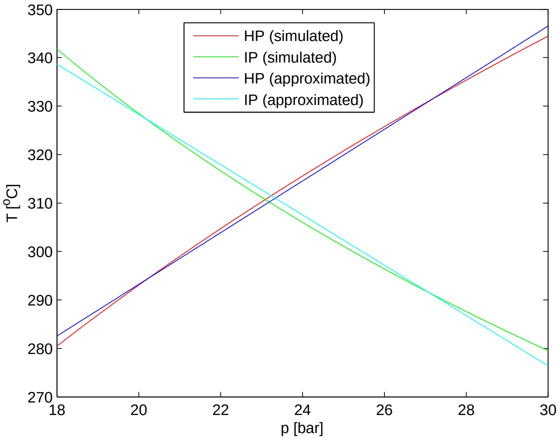


Figure 3.3: Simulated and approximated outlet temperatures for HP and IP steam turbines

$$w_e - (B_{W,e} + A_{W,e}\ p_{in(ATR)})f_{in(e),H_2O} = 0 \qquad \forall e \in \mathcal{E}_{T1} \tag{3.8}$$

$$t_e - (B_{T,e} + A_{T,e}\ p_{in(ATR)}) = 0 \qquad \forall e \in \mathcal{E}_{T1} \tag{3.9}$$

3.4 Physical modeling

Parts of the process can be described quite accurately with analytical equations from thermodynamics and simple balance equations, and it appears appropriate to exploit these direct, physical relationships rather than making more arbitrary approximations for such parts of the process.

The combustor, the gas turbine and the air compressors are modeled in this way, and we start by looking at the combustor, which is a reactor for which the extent of reaction is known, as we have complete combustion. The syngas entering the combustor consists of H_2 and N_2, and some CO_2, CO, CH_4 and H_2O. The fuel components H_2, CO and CH_4 are combusted completely, and air is supplied in excess such that the temperature out of the combustor is 1350 °C. The amount of air is determined by the energy balance of equation (3.10), which ensures that the heat released in reactions (2.4)-(2.6) equals the heat needed to heat the reaction products and the other components including the air to 1350 °C. The lower heating value (LHV; $Q_{LHV,k}$) is used for the energy released in the combustion, while the enthalpy equations (3.12) are used in calculating the heat demands $Q_{s,k}$. The composition of the exhaust is determined by equation (3.11) according to the stoichiometry of the combustion reactions.

$$\sum_{s \in \mathcal{S}_{IN,e}} \sum_{k \in \mathcal{K}_{F,s}} Q_{LHV,k}\ f_{s,k} = \sum_{s \in \mathcal{S}_{IN,e}} \sum_{k \in \mathcal{K}_s} Q_{s,k} f_{s,k} \quad \forall e \in \mathcal{E}_R \tag{3.10}$$

$$f_{out(e),k} = \sum_{s \in \mathcal{S}_{IN,e}} \sum_{l \in \mathcal{K}_s} G_{e,k,l}\ f_{s,l} \quad \forall k \in \mathcal{K}_{out(e)}, e \in \mathcal{E}_R \tag{3.11}$$

Turbines and compressors are essentially working in the same way, and are modeled with the same equations. However, the conditions of and even the flow rate through the main air compressor are set in advance, so these calculations are done before the optimization starts, while the conditions of the other air compressors and the gas turbine are variables in our model. The steam turbines are modeled as described in section 3.3.

Equations (2.7)–(2.10) are the basis for the modeling of turbines and compressors, but the modeling is more complex than it seems. The heat capacities c_p depend on the temperature, and the specific heat ratio κ likewise, as it is a function of heat capacities.

For calculating the average heat capacity over the temperatures in the turbine or compressors, we use the relationship between enthalpy and heat capacity, $\Delta h = c_p \, \Delta T$. Fifth-degree polynomial equations,

$$h_k(t) = \sum_{n=0}^{5} H_{k,n} \, t^n, \tag{3.12}$$

where $h_k(t)$ is the specific enthalpy of substance k at temperature t, are used to calculate the enthalpy at a certain temperature for each component of the stream. By using the ideal[4] outlet temperature as defined in (3.14), we get equation (3.13) for the power output of the turbine. Equation (3.15) give us the actual outlet temperature.

$$w_e - \sum_{k \in K_{in(e)}} \eta_e \; f_{in(e),k} \sum_{n=1}^{5} H_{k,n} ({T_{in(e)}}^n - {t_{I,e}}^n) = 0 \qquad \forall e \in \mathcal{E}_{T2} \tag{3.13}$$

$$t_{I,e} - T_{in(e)} \left(\frac{p_{out(e)}}{P_{in(e)}} \right)^{a_e} = 0 \qquad \forall e \in \mathcal{E}_{T2} \tag{3.14}$$

$$t_{out(e)} - |\eta_e| \; t_{I,e} = (1 - |\eta_e|) \; T_{in(e)} \qquad \forall e \in \mathcal{E}_{T2} \tag{3.15}$$

$$a_e \; w_e - \eta_e \; (T_{in(e)} - t_{I,e}) \sum_{k \in \mathcal{K}_e} R_k \; f_{k,in(e)} = 0 \qquad \forall e \in \mathcal{E}_{T2} \tag{3.16}$$

The exponent a_e in (3.14), corresponding to the expression $\frac{\kappa - 1}{\kappa}$ in equations (2.7) and (2.9), is set according to equation (3.16), which is derived in appendix C.

[4] Ideal as in no irreversibilities or energy loss, i. e. if $\eta = 1$.

The reasoning behind equations (3.13)–(3.16) is based on the case of a turbine, but the same equations are also used for compressors. This is accomplished by setting the efficiency terms η_e, where e is a compressor, equal the inverse of the compressor efficiency, multiplied by minus one to reverse the temperature difference in equation (3.13):

$$\eta_e = -\frac{1}{\eta_{0,e}} \quad \forall e \in \mathcal{E}_{C2},$$

where $\eta_{0,e}$ is the actual efficiency for compressor e. The minus sign should not be used in equation (3.15), and this is ensured by using the absolute value $|\eta_e|$.

This modeling of the gas turbine and air compressors is highly non-linear, and special care is taken to reduce the computational challenges related to these equations with appropriate scaling and bounds, as discussed in section 4.1.2. The inlet temperatures and pressures for the gas turbine and air compressors in our process are predetermined, and as shown with capital letters in the equations, they are fixed parameters in our model. For the gas turbine, even the outlet pressure is set in advance, such that equation (3.13) could be written with $P_{out(GT)}$ instead of $p_{out(GT)}$, and this is taken care of in the implementation. If instead both inlet and outlet conditions were unknown, the same equations could be used with similar adjustments, but that would mean even heavier non-linearities in the optimization model.

Splitting and mixing of compressed air is also modeled physically. Splitting is straight-forward if the stream is considered as made up of one substance, as we only need equation (3.17) to ensure mass balance while the splitting ratios are taken care of by other equations and optimization. If the stream consists of several components, which are modeled as such, an equation like (3.18)[5] is also needed, to ensure that the composition is the same for the outlet streams as for the inlet stream. This equation assumes that the composition of the inlet stream is known, and if that is not true,

[5] It should be noted that this equation is needed for only one $k \in \mathcal{K}_s$ and all $l \in \mathcal{K}_s \setminus k$, and that the sets $\mathcal{K}_s, s \in \mathcal{S}_{OUT,e}$ must be equal to $\mathcal{K}_{in(e)}$.

non-linear equations are needed. However, the splits we consider in the IRCC are modeled without distinguishing different substances, such that only equation (3.17) is needed.

$$f_{in(e),k} = \sum_{s \in \mathcal{S}_{OUT,e}} f_{s,k} \qquad \forall e \in \mathcal{E}_S, k \in \mathcal{K}_{in(e)} \tag{3.17}$$

$$N_{in(e),l} f_{s,k} = N_{in(e),k} f_{s,l} \qquad \forall e \in \mathcal{E}_{S2}, s \in \mathcal{S}_{OUT,e}, k, l \in \mathcal{K}_s \tag{3.18}$$

Mixing does in addition to the mass balance equation (3.19) involve the calculation of the mixing temperature, which is the mass average temperature as in equation (3.20). The mixing pressure is the lowest of the pressures of the streams to be mixed, and this could be modeled using binary variables, but in our case this need not be modeled explicitly as the pressures of the streams to be mixed are known to be equal.

$$f_{out(e),k} = \sum_{s \in \mathcal{S}_{IN,e}} f_{s,k} \qquad \forall e \in \mathcal{E}_M, k \in \mathcal{K}_{out(e)} \tag{3.19}$$

$$t_{out(e)} \sum_{k \in K_{out(e)}} f_{out(e),k} = \sum_{s \in \mathcal{S}_{IN,e}} \sum_{k \in \mathcal{K}_s} t_s \ f_{s,k} \quad \forall e \in \mathcal{E}_M \tag{3.20}$$

The amine absorption process for carbon capture is modeled simply as a separator with fixed ratios of each component in the inlet stream going to each of the three outlet streams, the H_2-rich fuel, the CO_2-rich carbon capture stream (CC) and a waste stream to model the material lost to the amine. In addition, the amine reboiler is modeled as a heater requiring 1.5 MJ of heat at 120 ° C per kg of CO_2 captured with equation (3.22). Work for compressing CO_2 to 110 bar is set at 342 MJ (95 kWh) per kg of CO_2 and modeled with equation (3.23).

$$f_{s,k} = V_{e,s,k} \ f_{in(e),k} \qquad \forall e \in \mathcal{E}_{SE}, s \in \mathcal{S}_{OUT,e}, k \in \mathcal{K}_{in(e)} \tag{3.21}$$

$$q_{CCU} = Q_{CC,CO_2} \ f_{CC,CO_2} \tag{3.22}$$

$$w_{CCU} = W_{CCU} \ f_{CC,CO_2} \tag{3.23}$$

$$f_{CC,CO_2} \geq U \ f_{in(ATR),CH_g} \tag{3.24}$$

It should be noted that the CCU is partly included in the metamodeling part, but as the relations between variables are this simple, regression is only done for the $f_{in(CCU),k}$ variables, and the other variables are calculated with equations (3.21)-(3.23). The last equation for the CCU ensures that the capture ratio is above a given limit, which is set to 90 % in this model.

In addition to what is mentioned so far, some simple modeling is done for unit operations which have a smaller impact on the total energy consumption of the process. Pumps to compress the steam to the given steam levels are modeled as the LP steam turbine, but the minor temperature increase is neglected. Work related to condenser cooling in the steam cycles is set to 0.5 % of the total power output from the steam turbines, which is a commonly applied rule of thumb. Use of cold utilities, that is cooling not covered by heat integration, is included in the model as the work needed to pump cooling water from atmospheric pressure to 2 bar, and the amount of cooling water is determined by assuming that it is heated from 10 to 25 °C, which is realistic in Norway. Use of hot utilities is modeled conservatively by assuming a one-to-one relationship between power and heat, and this is good enough because a demand for external heating in the process is unlikely.

3.5 Heat integration modeling

In the IRCC process, there are several streams to be cooled and heated, and if hot and cold streams[6] are matched appropriately, the energy requirements of the process can be reduced significantly. This is the basis for heat integration.

To model heat integration for the IRCC, we use the concept of the heat recovery pinch (Linnhoff and Hindmarsh, 1983). Complete modeling of heat integration would involve both heat recovery targeting and heat exchanger network design. However, as we aim for simultaneous optimization of the process, including heat integration, the temperatures and flow rates

[6] With respect to heat integration, a cold stream is a stream requiring heating, whereas hot refers to a demand for cooling, regardless of the temperature of the streams.

of the streams to be integrated are not known before optimizing the heat integration, and a full model with heat exchanger network design would become too hard to solve. Thus, our heat integration model is limited to heat recovery targeting, following the principles described in section 5.2 of Gundersen (2000).

Optimization of a process with respect to the heat recovery target is a complex task in itself when temperatures and flow rates are not fixed. The calculation of heat flows involves non-linear terms, as temperature differences are multiplied by specific heat times mass flow rate. The implicit matching of hot and cold streams based on feasible temperature differences, that is the modeling of the pinch, is based on discrete decisions, which can be modeled with binary variables.

Duran and Grossmann (1986) developed a model for simultaneous optimization and heat integration of chemical processes, much like what is the purpose of this work, but with approximating functions instead of binary variables to model kinks (functions with discontinuous derivatives). These approximating functions might be problematic, as the optimization tend to choose solutions where the approximation is least accurate. As the heat integration model is an important part of this work, and as algorithms and hardware have improved since 1986, we want to model the kinks exactly with binary variables. Grossmann et al. (1998) extend the model for isothermal streams, and present a formulation with binary variables, although recommending the use of approximating functions for non-isothermal streams to ease the optimization. This MINLP formulation contain three binary variables (or one special ordered set of type 1 with 3 elements) for each possible combination of pinch candidate and stream. In principle, the inlet temperature of any of the streams may cause pinch, thus the model will in worst case have $3{N_S}^2$ binary variables, where N_S is the number of streams.

In this work, a model with fewer binary variables is presented. Instead of three binary variables per combination of pinch candidate and stream, our model have only one binary variable (or one SOS1 with two members) per such combination. Even though the number of binary variables might be an indication of how hard it is to solve a problem, there are other factors which influence the solvability, and when comparing our model with that of Grossmann et al. (1998) on several test problems, some problems were solved faster with our model, and some were easier for the other model. However, both models found the same optimal solutions. We cannot conclude that our model is better or worse than that of Grossmann et al. (1998), but it is chosen for this work because the author is more familiar with his own formulation, and for the purpose of presenting and testing it. The novel heat integration model is presented in the following section.

3.5.1 The heat integration model

The main principles of the heat integration targeting model developed in this work, are that cold streams above pinch are undesirable and that the pinch can be implicitly defined by a set of constraints (3.25) which ensure that there is at least enough heating (including utilities) to cover the cooling demand above any temperature. The temperature corresponding to the strictest of these constraints is the pinch temperature. This is equivalent to choosing the amount of external heating such that no residuals in the heat cascade are negative, as is the standard procedure for heat recovery targeting based on pinch analysis when the temperatures and flows are known. This is also the principles used by Grossmann et al. (1998).

$$q_H \geq \sum_{c \in C} (t_{OUT,c} - t_{M,c,p}) f_{C,c} - \sum_{h \in H} (t_{IN,h} - t_{M,h,p}) f_{C,h} \forall p \in \mathcal{S}_P \qquad (3.25)$$

When the utility heating demand q_H is defined, the demand for utility cooling q_C is given from the energy balance of (3.26). The objective of the heat integration model, if it is to be run separately, is to minimize the cost of utility consumption and possibly costs for streams, as in (3.27).

$$q_C - q_H = \sum_{s \in \mathcal{S}} (t_{IN,s} - t_{OUT,s}) f_{C,s} \qquad (3.26)$$

$$\min \quad K_H \; q_H + K_C \; q_C + \sum_{s \in S} K_s \; f_{C,s} \qquad (3.27)$$

The novelty of our model is that instead of explicitly distinguishing between the three different cases of a stream being above, across or below a temperature, we do only need to know whether the inlet temperature for hot streams, or the outlet temperature for cold streams, is above or below each possible pinch temperature, and then a constructed intermediate temperature ($t_{M,s,p}$) and the fact that we try to minimize utility consumption take care of the rest. The temperature differences $t_{IN,h} - t_{M,h,p}$ and $t_{OUT,c} - t_{M,c,p}$ of constraints (3.25) accounts for the part of the stream that is above pinch candidate p, and this is ensured by constraints (3.28)–(3.33).

$$
\begin{aligned}
& t_{M,h,p} \geq t_{OUT,h} && \forall h \in \mathcal{H},\ p \in \mathcal{S}_P \quad (3.28)\\
& t_{M,h,p} \geq t_{IN,p} + T_{ADJ,p} - T_{ADJ,h} - M_{1,h,p} y_{h,p} && \forall h \in \mathcal{H},\ p \in \mathcal{S}_P \quad (3.29)\\
& t_{M,h,p} \geq t_{IN,h} - M_{2,h,p}(1 - y_{h,p}) && \forall h \in \mathcal{H},\ p \in \mathcal{S}_P \quad (3.30)\\
& t_{M,c,p} \leq t_{OUT,c} && \forall c \in \mathcal{C},\ p \in \mathcal{S}_P \quad (3.31)\\
& t_{M,c,p} \leq t_{IN,p} + T_{ADJ,p} - T_{ADJ,c} - M_{1,c,p}(1 - y_{c,p}) && \forall c \in \mathcal{C},\ p \in \mathcal{S}_P \quad (3.32)\\
& t_{M,c,p} \leq t_{IN,c} - M_{2,c,p} y_{c,p} && \forall c \in \mathcal{C},\ p \in \mathcal{S}_P \quad (3.33)
\end{aligned}
$$

To explain how these equations work, we start by looking at hot streams, assuming for simplicity that $T_{ADJ,s} = 0\ \forall s$. The temperature difference in equation (3.25), $t_{IN,h} - t_{M,h,p}$, should be $t_{IN,h} - t_{OUT,h}$ if hot stream h is above pinch candidate p, $t_{IN,h} - t_{IN,p}$ if it crosses pinch (and stream p makes the pinch), and 0 if the stream is below pinch candidate p, to make sure that only heat exchanged above pinch candidate p is included.

Heating above pinch is valuable, so in the difference $t_{IN,h} - t_{M,h,p}$, we want $t_{M,h,p}$ to be as low as possible, and this is ensured as Q_H is to be minimized (3.27) and lower $t_{M,h,p}$ in (3.25) yields lower Q_H[7]. Equations (3.28)-(3.33) make sure that $t_{M,h,p}$ is at least as high as $t_{OUT,h}$ and either $t_{IN,h}$ or $t_{IN,p}$.

If the hot stream h is above pinch candidate p, then $t_{OUT,h}$ is the highest, and thus we get $t_{M,h,p} = t_{OUT,h}$. If the stream is below or across pinch candidate p, both $t_{IN,h}$ and $t_{IN,p}$ will be higher than $t_{OUT,h}$, and $t_{M,h,p}$ will equal the lowest of $t_{IN,h}$ and $t_{IN,p}$.

[7] This is only true (and relevant) for the p that actually makes the pinch, and this applies even to the next arguments.

For hot streams across pinch candidate p, $t_{IN,p}$ will be the lowest, and we get the correct contribution above pinch candidate p, $t_{IN,h} - t_{IN,p}$, in equation (3.25). For streams below pinch, $t_{IN,h}$ is lower than $t_{IN,p}$, thus the binary variable $y_{h,p}$ is chosen such that $t_{M,h,p} = t_{IN,h}$, and the temperature difference in equation (3.25) becomes $t_{IN,h} - t_{IN,h}$, which is 0 as it should be.

Equations (3.31)-(3.33) for the cold streams follow the same principles, but with opposite inequalities as cooling above pinch is unfavorable, and with $t_{OUT,c}$ as the higher temperature in the temperature difference in equation (3.25).

This first formulation do not allow for isothermal streams, but this can be done by changing equations (3.25) and (3.26) to (3.34) and (3.35) and adding (3.36)-(3.40):

$$\begin{aligned} q_H \geq \sum_{c \in C} C_{P,c}(t_{OUT,c} - t_{M,c,p}) f_{C,c} + \sum_{c \in \mathcal{C}_I} f_{QP,c,p} \ - \\ \sum_{h \in H} C_{P,h}(t_{IN,h} - t_{M,h,p}) f_{C,h} - \sum_{h \in \mathcal{H}_I} f_{QP,h,p} \end{aligned} \qquad \forall p \in \mathcal{S}_P \tag{3.34}$$

$$q_C - q_H = \sum_{s \in \mathcal{S}_N} (t_{IN,s} - t_{OUT,s}) f_{C,s} + \sum_{h \in \mathcal{H}_I} f_{Q,h} - \sum_{c \in \mathcal{C}_I} f_{Q,c} \tag{3.35}$$

$$t_{IN,h} \geq t_{IN,p} + T_{ADJ,p} - T_{ADJ,h} - M_{1,h,p} y_{h,p} \qquad \forall h \in \mathcal{H}_I, \ p \in \mathcal{S}_P \tag{3.36}$$

$$t_{IN,c} \leq t_{IN,p} + T_{ADJ,p} - T_{ADJ,c} + M_{1,c,p}(1 - y_{c,p}) \forall c \in \mathcal{C}_I, \ p \in \mathcal{S}P \tag{3.37}$$

$$f_{QP,h,p} \leq M_{2,h,p}(1 - y_{h,p}) \qquad \forall h \in \mathcal{H}_I, \ p \in \mathcal{S}_P \tag{3.38}$$

$$f_{QP,h,p} \leq \ f_{Q,h} \qquad \forall h \in \mathcal{H}_I, \ p \in \mathcal{S}_P \tag{3.39}$$

$$f_{QP,c,p} \geq f_{Q,c} - M_{2,c,p} y_{c,p} \qquad \forall c \in \mathcal{C}_I, \ p \in \mathcal{S}_P \tag{3.40}$$

What is left then, is only defining the domains for the variables. In addition to (3.42), there should be upper and lower bounds for $t_{IN,s}$, $t_{OUT,s}$ and $f_{C,s}$, and the difference between $t_{IN,s}$ and $t_{OUT,s}$ will often be constrained.

$$y_{s,p} \in \{0, \ 1\} \qquad \forall s \in \mathcal{S}, \ p \in \mathcal{S}_P \tag{3.41}$$

$$q_H, q_C, t_{IN,s}, t_{OUT,s}, t_{M,s,p}, f_{C,s}, f_{Q,s}, f_{QP,s,p} \geq 0 \tag{3.42}$$

This whole formulation is assuming that the pinch point can only be located at the inlet of a stream. This is true if the heat capacity is constant (not varying with temperature), but heat capacity does in reality vary with temperature, and if the HRAT is low, this may not only result in wrong determination of the pinch point, it may even mean that the heat exchange in the optimized solution is infeasible. The assumption of constant heat capacities is however widely used, partly because heat capacity is almost constant over a the temperature interval of most streams, and with the HRAT chosen at 20 °C it is fairly safe anyway.

Some improvements of the heat integration model are presented in section 4.1.3.

The heat integration do of course depend on the temperatures and flow rates of the overall model, and such connections are either ensured by using the same variable in both parts of the model, or by simple equality constraints such as

$$t_{IN,s} = t_p \qquad \forall(s,p) \in \mathcal{S}_{S,TI} \tag{3.43}$$

$$t_{OUT,s} = t_p \qquad \forall(s,p) \in \mathcal{S}_{S,TO} \tag{3.44}$$

$$f_{C,s} = \sum_{k \in \mathcal{K}_p} C_{P,p,k}\ f_{p,k} \qquad \forall(s,p) \in \mathcal{S}_{S,F1} \tag{3.45}$$

$$f_{C,s} = f_{C,p} \qquad \forall(s,p) \in \mathcal{S}_{S,F2} \tag{3.46}$$

$$f_{Q,s} = q_e \qquad \forall(s,e) \in \mathcal{S}_{E,Q}, \tag{3.47}$$

where p refers to a stream, not necessarily a pinch candidate anymore. Equation 3.45 is an approximation because a constant is used for heat capacity, without knowing the state of stream p. In addition there are some of the heat integration streams that depend on each other, for example is the heating of steam divided into three steps, which of course should have the same underlying mass flow rates and equal temperature at the outlet of the first stream and the inlet of the next. These streams are modeled with constant temperatures, such that there is only a need for connecting the flows:

$$Q_{p,H_2O}\ f_{C,s} = C_{P,s,H_2O} f_{Q,p} \forall(s,p) \in \mathcal{S}_{S,Q} \tag{3.48}$$

In our model, such connections are only made between a isothermal and a non-isothermal stream, but it would of course be possible to model the connection of two non-isothermal streams with a similar equation.

Chapter 4

Implementing the IRCC optimization model

In the preceding chapters, the model is established formally. Additional considerations are made in the actual implementation of the model, both to enhance computational performance and to deal with practical issues such as enabling communication between different software. These measures are explained in this chapter, starting with issues related to optimization and the solver in use, followed by a description of the simulation and then an overview of the architecture of the implementation.

4.1 Global optimization

While linear optimization is extensively applied, and software for effectively solving large-scale linear optimization problems — mainly based on the simplex algorithm, but also interior-point methods — is easily available, solving non-linear optimization problems is not that straight-forward. Convex problems can usually be solved quite easily, but many real-world problems are non-convex. The solution of a non-convex problem do often depend on good bounds on the variables and initial values close to the optimum, as most software can only find local solutions.

Optimization with integer variables — Mixed Integer Programming (MIP) — is also more challenging than continuous linear optimization, but the development of algorithms for solving MIP problems — dominated by the successful branch-and-bound algorithm — have reached farther than for non-linear problems, and MIP is applied extensively. Optimization with both non-linearities and discrete variables — Mixed Integer Non-Linear Programming (MINLP) — is, as one should expect, generally very challenging.

Some efforts have been made to develop global optimization software that can solve a wide range of non-convex optimization problems, and even MINLPs, and the performance of such solvers is improving as better algorithms are developed and computational power is ever increasing. In this work such a solver is used, and thus the solving is not dealt with directly, but some measures are applied to help the software.

4.1.1 BARON[1]

As this work depend heavily on the global optimization software used, it is natural to describe the capabilities of the solver before discussing the efforts made to ease the optimization. BARON, an acronym for the Branch-And-Reduce Optimization Navigator, is a global solver that also use external solver software for solving LP and NLP subproblems.

BARON is capable of solving MINLPs with factorable functions derived recursively from sums and products of a set of basic functions, including a^x, $ln(x)$, x^a and x^y, where the a's are real numbers and x and y are variables. This is referred to as factorable non-linear programming, and is general enough to cover a wide range of optimization problems, including that of this work.

As the name suggests, BARON is based on branching and reduction. Branch and bound is applied with branching on both continuous and integer variables. Branching means that the problem is divided into smaller sub-problems, which together span the original problem, but individually

[1] This section is based on Sahinidis and Tawarmalani (2010) and the more thorough, but older Sahinidis (2000).

are somewhat easier to solve, and bounding refers to finding upper and lower bounds to the sub- and master problems, which are used to decide whether a sub-problem is worth investigating. The subproblems are relaxed[2] with convex underestimators (assuming minimization). The relaxed problem is solved to find lower bounds, and local minimization is applied to find upper bounds. Branching is done for the variable that, according to certain criteria, contribute most to the gap between the relaxation and the actual model, and several approaches are used to determine the value for branching. These are the midpoint of the feasible interval for that variable, the incumbent solution or the solution to the lower bounding problem.

The relaxations in the lower bounding problems are based on underestimators for concave terms. Bilinear terms and strictly concave power terms are underestimated with linear functions that make a straight line between the upper and lower bound for the term, and Sahinidis (2000) point out that, as the relaxed problems are typically constructed such that they are exact at the upper and lower variable bounds, tight variable bounds are important to get tight relaxations.

Reduction refers to measures that reduce the size of the problem. One such measure is optimality-based range reduction, which is a procedure for adding valid inequalities based on the optimal solution of relaxations to decrease the feasible area without cutting off the optimal solution. Feasibility-based range reduction is another such measure, in which variable bounds are tightened by using insights from the constraints in what is called the "Poor man's linear programming" heuristic, to cut off infeasible portions of the solution space.

Most non-linear optimization solvers require user-supplied starting points, but BARON do not need this. BARON do even perform a multi-start local search before starting the main branch and bound algorithm, without any initial values provided by the user.

[2] a relaxed problem — or a relaxation — is a modification of the original problem with less stringent constraints and/or an objective function consistently giving better objective values. All feasible points in the original problem are feasible in the relaxation, and the optimal solution of the relaxation is at least as good as the optimal solution to the original problem, although it is not necessarily feasible in the original problem.

4.1.2 Scaling, variable bounds and reformulation

State-of-the-art solver software will usually do several forms of pre-processing like scaling and reformulation of the problem, but there are good reasons to do this manually before the problem is sent to the solver. Firstly, it is not always obvious what the software does of pre-processing, and — more importantly — the software do not have qualitative insights to the problem.

Scaling is applied to avoid a difference in many orders of magnitude between variables in the model to avoid numerical problems. Manual scaling is recommended by McCarl et al. (2011, chapter 12.2), and scaling is especially important in non-linear optimization. An example from our model is the variable corresponding to the outlet temperature of the gas turbine, which have a upper bound of almost 1000 K. This may not seem dramatically high, but this variable appears to the fifth power in (3.15), resulting in an upper bound for this intermediate expression of 10^{15}, while the coefficient has a value of about 10^{-13}, if no scaling is applied. By scaling the temperature variables such that this upper bound is instead around 1.2, and the related coefficients accordingly, we get $1.2^5 \approx 2.5$, with the values of corresponding coefficients not far from this. In the implementation of our model, applying appropriate scaling of critical variables and coefficients manually have been necessary to make BARON solve the problem.

Providing appropriate variable bounds is not only important in avoiding physically impossible or undesirable solutions, it is also critical in to the process of solving, as mentioned in section 4.1.1 and because the size of the solution space is reduced. A considerable effort is made to find good bounds on critical variables, and both upper and lower bounds are provided for all variables. For non-linear models in general, it is also important to set variable bounds such that division with zero and discontinuities are avoided. However, in our formulation we have used a form of reformulation to avoid having division expressions with variables in the denominator at all. This simple trick is illustrated below:

$$\frac{x}{y} = z, \qquad \overset{reformulation}{\rightarrow} \qquad x = y\ z$$

4.1.3 Improving the heat integration modeling

The heat integration model is in itself relatively heavy, as it contains both binary variables and non-convex constraints, and measures were taken to make it easier to deal with.

The binary variables used in modeling the pinch are complicating the model, and there can be quite many of them. In our model, 14 streams are included in the heat integration part, which would give $14^2 = 196$ binary variables if all streams are considered candidates for making pinch. However, the steam superheaters cannot logically cause pinch because their inlet is at the outlet of boiler, resulting in a kink in the cold composite curve away from the hot curve. There are two superheaters, thus we are left with $14 \cdot 12 = 168$ binary variables.

The model can be improved additionally by utilizing that he binary variables $y_{s,i}$ are related to which of the streams s and i have the highest inlet temperature (adjusted for HRAT). If the lower bound of $t_{IN,s}+T_{ADJ,s}$ is higher than the upper bound of $t_{IN,i} + T_{ADJ,i}$, there is no need for the binary variable $y_{s,i}$. This logic is used to remove variables from the model, and makes it easier for the solver. This is another reason to provide good variable bounds, and for our process, the number of binary variables was reduced to from 168 to 44.

As in any big M formulation, finding the lowest possible — but still big enough — value for the big M constant has a positive effect on the model's solvability. The equations involving big M constants are (3.29), (3.30), (3.32), (3.33), (3.36)–(3.38) and (3.40). Taking (3.29) and (3.30) as an example,

$$\begin{aligned} t_{M,h,i} &\geq t_{IN,i} + T_{ADJ,i} - T_{ADJ,h} - M_{1,h,i}y_{h,i} \\ t_{M,h,i} &\geq t_{IN,h} - M_{2,h,i}(1 - y_{h,i}), \end{aligned}$$

$M_{1,h,i}$ must be as big as the highest possible value of $t_{IN,i}+T_{ADJ,i}-T_{ADJ,h}-t_{M,h,i}$ for which $t_{M,h,i}$ is greater than $t_{IN,h}$. With $\bar{t}$ and $\underline{t}$ denoting upper and lower bounds, respectively, for the variable t, we get the following tight value for $M_{1,h,i}$:

$$M_{1,h,i} = \bar{t}_{IN,i} + T_{ADJ,i} - T_{ADJ,h} - \underline{t}_{IN,h,i} \tag{4.1}$$

Similar reasoning is used in setting the other big M values.

4.1.4 Relaxation based on process insights

BARON could not find the optimal solution to our model, at least not prove optimality without spending more time than tolerable. To improve the solving, a relaxation scheme was developed. As the model was solvable before the heat integration modeling was included, it was natural to try making a relaxation of this part, and the discrete variables were suspected to cause difficulties for BARON.

The $y_{s,i}$ variables are related to pinch candidate streams by their second index, and their purpose is to determine the $t_{M,s,i}$ variables used in constraints (3.25). For a given optimal solution, only one of these equations will be binding, namely the one corresponding to the actual pinch point. As the temperatures and heat flows of the integrated streams are governed by other equations in the model too, one could expect that several of the constraints (3.25) are redundant. Said with other words, there are probably only a few of the pinch candidate streams that are close to make pinch.

If only a subset of the pinch candidates are accounted for in the model, this corresponds to removing some of the constraints (3.25), and all other equations defined over the set of pinch candidates. As constraints are removed, we get a relaxation of the original problem, and the optimum of the problem accounting for fewer of the pinch candidates will be at least as good as the optimum of the original problem. If the optimum of the relaxation is feasible in the original model, i. e. it satisfies the pinch constraints that are removed, it is even the actual optimum. As many of the pinch candidates may be very unlikely, the chance for these removed constraints being fulfilled is possibly quite high if the appropriate pinch candidates are chosen to remain.

Even if the solution of the relaxation is not feasible in the original problem, it is very easy to make it so — it is just to add more external cooling and heating. A feasible solution constructed in this way need not be optimal, but it will at least provide a lower bound (as we are maximizing).

A process engineer with knowledge of the process should be able to detect which streams are likely to make pinch, and the three boilers (LP steam, HP steam and CCU reboiler) were natural suspects. However, it is possible to make the model suggest likely pinch candidates too, and the following iterative approach is suggested:

1. Run the original model for some time, and add the stream p causing pinch in the current solution to the currently empty subset of pinch candidates, $\mathcal{S}_{P*} := \{p\}$.

2. Run the model with $\mathcal{S}_{P*}$ instead of $\mathcal{S}_P$ as pinch candidates to optimality, and post-process the solution to find which stream p would cause the pinch with this solution.

3. If the stream causing pinch in this solution is in the subset of pinch candidates: Stop. The solution is optimal.

4. Else, add the new stream causing pinch (p) to the subset of pinch candidates $\mathcal{S}_{P*} := \mathcal{S}_{P*} \cup \{p\}$, and go to step 2.

This approach depend on the optimal solution of the relaxed problem, and if the optimum of the original problem is not found during the first few iterations, the relaxed problem will become hard to solve, as more pinch candidates — and thereby more constraints with corresponding discrete variables — are added. It is by no means guaranteed to work well under all circumstances, but for the purpose of this work, it has worked well.

Lagrange relaxation has also been considered. The relaxation used here may be interpreted as Lagrange relaxation with all multipliers equal to zero, and as it was successful, other multiplier values have not been investigated.

4.2 Simulation and regression

Process simulation is utilized to provide data for regression in the metamodeling. Software like Aspen Plus, PRO/II and Aspen HYSYS are known to give very accurate results for a wide range of chemical and thermodynamic processes, and in this work Aspen HYSYS (Asp, 2006) is used for process simulation.

5000 simulations were performed with LHS, for values of each input variable within the ranges presented in table 4.1. To avoid bad approximations in the outer parts of the feasible solution space, these ranges are a bit wider than the ranges used in the optimization, but for the ATR pressure, a value higher than 30 bar would result in negative compression further downstream, and to avoid this, while still trying to ensure a good fit near this value, many simulations are run for an ATR pressure of 30 bar.

Table 4.1: Ranges for input variables

Variable	Feasible range	Range tested
ATR pressure	18–30 bar	15.6–30 bar
Steam to carbon ratio	1–2	0.8–2.2
Temperature of methane to the ATR	200–500 °C	140–560 °C
HTS temperature	300–450 °C	270–480 °C
LTS temperature	180–250 °C	166–264 °C
Temperature of air to the ATR	410–530 °C	386–554 °C
Temperature of steam to the ATR	280–345 °C	267–358 °C

The upper five variables in table 4.1 are considered free in the optimization, with ranges based on what might be considered practically possible, while the other two variables do actually depend on variables in the model which are not part of the simulated process, and the ranges for these are based on assumptions for the air compressors and steam turbines.

The regression for each output variable is done only for input variables which theoretically can influence the output variable, and where coefficients with values close to zero have turned up, the regression is performed again, requiring this coefficient to be zero to get rid of such insignificant coefficients. All variables are scaled to the interval $\left\langle -\frac{1}{2}, \frac{1}{2} \right\rangle$ before regression to make the coefficients comparable and avoid numerical trouble. These measures are discussed and explained in Johnsen (2010).

The resulting second order polynomial regression coefficients are indicated in table 4.2, where + means that the coefficient is positive, – means that it is negative, and an empty cell means that the coefficient is zero. Actual regression coefficients are found in appendix B. The output variables are listed in table 4.2, with numbering that is used in the other tables. In

Table 4.2: Output variables

d	Variable	d	Variable
1	Mass flow of CO_2 after LTS	10	Syngas compressor outlet temperature
2	Mass flow of CO after LTS	11	Mass flow of H_2O after WR
3	Mass flow of H_2 after LTS	12	Mass flow of CO after WR
4	HTS product temperature	13	Mass flow of CO_2 after WR
5	LTS product temperature	14	Mass flow of H_2 after WR
6	GT fuel preheating	15	Mass flow of CH_4 after WR
7	HTS product cooling	16	Mass flow of N_2 after WR
8	LTS product cooling	17	GT fuel compression work
9	ATR product cooling		

addition to the simulation described in section 3.3, simulations of a larger part of the process, with values from optimized solutions, are used to verify the accuracy of our optimization model.

4.3 Overview of the implementation

The complete model consists of a process model in HYSYS, and an optimization model written in GAMS (General Algebraic Modeling System; Brooke et al., 2003), which is a high-level modeling language which can call several solvers, including BARON. For efficiently handling input and output data for both HYSYS and GAMS/BARON, in addition to some calculations such as the regression and data processing, MATLAB was chosen as an appropriate tool, with its powerful language, built-in functions and data plotting capabilities. Communication between MATLAB and GAMS is done with the GDXMRW package (Ferris et al., 2010), and the HYSYS simulations are controlled with Hysyslib (Berglihn, 2011) functions.

An overview of the interaction between software is sketched in figure 4.1, which also shows the structure of the implementation. The part above the dashed line is the basis for metamodeling. The LHS sampling scheme is calculated in MATLAB, and then Hysyslib functions are used in setting the values of input variables in the HYSYS process model. When the simulation is done, the results are sent back to MATLAB and stored, and the

Table 4.3: Signs of regression coefficients

d	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
$\alpha_{d,0}$	+	−	+	+		+		−		−	−	−	+	+	−	−	−
$\alpha_{d,1}$	−	+	+				−	−	−		−	+	−	+	+	−	
$\alpha_{d,2}$	−	+	−		+			−				+	−	−			
$\alpha_{d,3}$			−			+				−	−			−	+		−
$\alpha_{d,4}$	−	+	−	+	+			−	+			+	−	−			
$\alpha_{d,5}$	+	−	−	−	−		+	+	+		+	−	+	−	−	+	
$\alpha_{d,6}$	−	+	+				−	−	−		−	+	−	+	+	−	
$\alpha_{d,7}$			+											+		−	
$\beta_{d,1,1}$			+											+	+	−	
$\beta_{d,2,2}$	−	+	−					+				+	−	−			
$\beta_{d,3,3}$						−				+	+				+		+
$\beta_{d,4,4}$																	
$\beta_{d,5,5}$	−	+	−					+				+	−	−	+		
$\beta_{d,6,6}$																	
$\beta_{d,7,7}$																	
$\beta_{d,1,2}$																	
$\beta_{d,1,3}$															+		
$\beta_{d,1,4}$																	
$\beta_{d,1,5}$	+	−	+									−	+	+	−		
$\beta_{d,1,6}$																	
$\beta_{d,1,7}$																	
$\beta_{d,2,3}$																	
$\beta_{d,2,4}$		+	−									+		−			
$\beta_{d,2,5}$	+	−	+		+			−				−	+	+			
$\beta_{d,2,6}$																	
$\beta_{d,2,7}$																	
$\beta_{d,3,4}$																	
$\beta_{d,3,5}$			+											+	−		
$\beta_{d,3,6}$															+		
$\beta_{d,3,7}$																	
$\beta_{d,4,5}$	+	−	+	+	−			−				−	+	+			
$\beta_{d,4,6}$																	
$\beta_{d,4,7}$																	
$\beta_{d,5,6}$	+	−	+									−	+	+	−		
$\beta_{d,5,7}$			+											+		−	
$\beta_{d,6,7}$																	

simulation is repeated for the next sample. After simulating the process for all sets of input data from the sampling scheme, MATLAB performs regression to fit the data to second order polynomials. The regression coefficients are then saved, such that this procedure needs to be done only once unless there are changes in the process simulation model. In the part below the dashed line, some constants are calculated in MATLAB before the GDXMRW package is used to send data to GDX files, which GAMS can read. GAMS is either called through MATLAB or separately, and sends the model with data from the data files to BARON, which solves the model. Solution data are then written to another GDX file, which is read by MATLAB and used to produce figures and tables for displaying the solution.

If using the iterative solution approach of section 4.1.4, this procedure should be repeated until the solution of the relaxed model is feasible in the overall model. As indicated by a dashed arrow, this looping must be done manually.

Finally, the accuracy of the solution is checked by running a HYSYS simulation with values from the optimal solution as input data. The results from this solution is compared with the optimization model results like in table 5.3.

The dashed arrow with input data and parameters indicates that this part of the model could be run again with alternative data, and in section 4.4 we present different scenarios for which the model will be re-run.

4.3.1 Software versions

Software versions used are listed in table 4.3.1. BARON uses CPLEX for linear subproblems, and MINOS for NLPs.

4.3.2 Hardware

The computations are carried out on an Acer Aspire 5741G laptop with an Intel Core i3-350M processor (2.26 GHz, 3MB L3 cache) and 4 GB DDR3 RAM, running Windows 7.

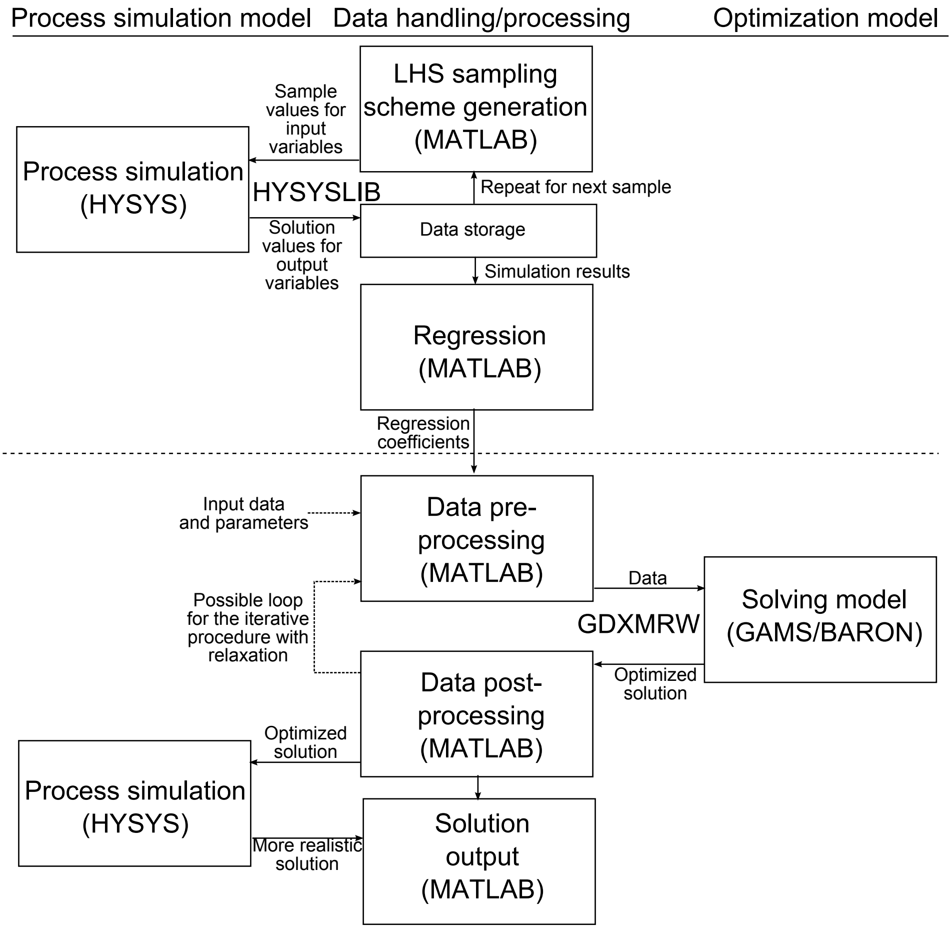

Process simulation model
Data handling/processing
Optimization model
LHS sampling scheme generation (MATLAB)
Sample values for input variables
Process simulation (HYSYS)
HYSYSLIB
Repeat for next sample
Data storage
Solution values for output variables
Simulation results
Regression (MATLAB)
Regression coefficients
Input data and parameters
Data pre-processing (MATLAB)
Data
Solving model (GAMS/BARON)
GDXMRW
Possible loop for the iterative procedure with relaxation
Optimized solution
Data post-processing (MATLAB)
Optimized solution
Process simulation (HYSYS)
More realistic solution
Solution output (MATLAB)

Figure 4.1: Structure of the implementation and software interaction

Table 4.4: Software versions

Software	Version
BARON	9.0.6
ILOG CLEX	12.2.0.1
MINOS	5.51
GAMS	23.6.2
HYSYS	7.2
MATLAB	7.11.0.584 (R2010b)

4.4 Scenarios

To show that the model works for different data and modifications, and to investigate the effect of changing certain parameters, we will run the model for different scenarios. The base case, which receive most of the attention in the Results chapter, is the model run with the data presented so far in this work, with the objective to maximize the electric efficiency of the process. The alternative scenarios are modifications of this, with the changes described in the following. Results for these scenarios are presented in section 5.5.

4.4.1 Maximizing power output

This scenario is run with the alternative objective of (3.3), to see whether the solution is changed if one decides to maximize power output for the given size of the air compressor rather than maximizing the efficiency.

4.4.2 No choice for ATR air compression

In the model there is a choice between using a second compressor after the main air compressor or an extra air compressor, or a combination, to compress the air fed to the ATR. With this scenario, we can see the impact of having to use only the main and second compressors.

4.4.3 Lower turbine inlet temperature

The assumption of a combustion temperature of 1350 °C might be a bit optimistic when turbine cooling is not modeled, as this becomes the turbine inlet temperature, therefore it is interesting to see what happens to the solution if the TIT is set to 1285 °C.

4.4.4 Higher pressure steam cycles

A limitation of the model is that it does not optimize the pressure levels in the steam cycle. In this scenario, the high pressure level is set to 130 bar, and the low pressure level to 8 bar, to see if a different pressure level affects the solution.

4.4.5 Lower limit for CO_2 capture

In this scenario, we investigate the impact of the restriction requiring 90 % of the carbon in the methane fuel being captured as CO_2. The CCU is still modeled in the same way, but the process need not be designed to form that much CO_2 in reformer and the water gas shifts, as the constraint of minimum 90 % capture is relaxed.

Chapter 5

Results

Results for the IRCC optimization model are presented in this chapter, showing both the performance of the resulting IRCC design, and the accuracy of the model. We will compare the optimized process to similar processes in the literature, and go through results for the scenarios described in section 4.4. Finally, the computational performance of the model is analyzed.

5.1 Overall performance

The main results with respect to power are presented in table 5.1, and the optimized values for different variables are shown in the flow-sheet in figure 5.1. The gas turbine power output reported in the table does not include air compression, meaning that the net gas turbine power is 418 MW. Auxiliary power consumption comprises the work for pumping of steam, and pumps for condenser and utility cooling.

It is interesting to note that the ATR pressure is at its upper bound of 30 bar. The pressure do not affect the reactions very much, so this suggests that the lost power generation in the HP turbine for extracting the steam at this relatively high pressure and compressing air accordingly is at least compensated for by avoiding compression of the syngas. The steam to

Table 5.1: Summary of power output and consumption for optimized IRCC

Gas turbine power (gross)	707.04	MW
Main air compressor work	-288.76	MW
Second air compressor work	-0.00	MW
Extra air compressor work	-71.76	MW
HP steam turbine power	74.05	MW
IP steam turbine power	90.55	MW
LP steam turbine power	97.84	MW
CO_2 compression	-19.95	MW
GT fuel compression work	-0.15	MW
Auxiliary power consumption	-4.51	MW
Net power output	584.35	MW
Energy input (CH_4 LHV)	1183.89	MW
Efficiency	49.36	%
Cold utilities (external cooling)	-8.39	MW
Hot utilities (external heating)	-0.00	MW
CO_2 capture ratio	90.00	%
ATR pressure	30.00	bar
Steam to carbon ratio	1.07	

carbon ratio is close to its lower bound to avoid too much steam extraction, but the CO_2 capture constraint keeps it from being lower as more steam drives the reforming and WGS reactions towards more hydrogen and CO_2.

The extra air compressor is chosen for supplying the ATR with air such that the second air compressor is not in use. One interesting detail of the result is that there is no LP steam production, and reasons for this are discussed in 5.2. The distribution of power generation between the gas turbine and the steam turbines is however standard, with 418 MW of net power from the gas turbine and around 260 MW from the steam turbines.

5.2 Heat integration results

To illustrate the heat integration results, we will use composite curves (figure 5.2) and the grand composite curve (figure 5.3), showing the driving forces for heat exchange in the resulting process. The rationale behind such

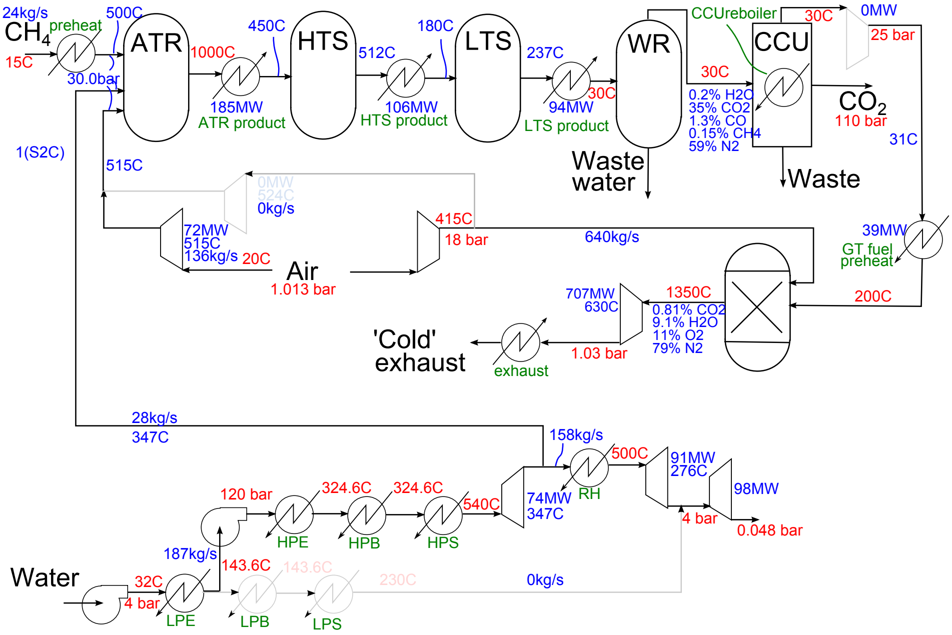


Figure 5.1: Flow-sheet with data for the optimized process

plots are presented in Gundersen (2000, section 5.2.1). It is important to keep in mind that the heat integration results are based on the maximum level of heat recovery, and the consumption of utilities is likely to be higher with a realistic heat exchanger network design.

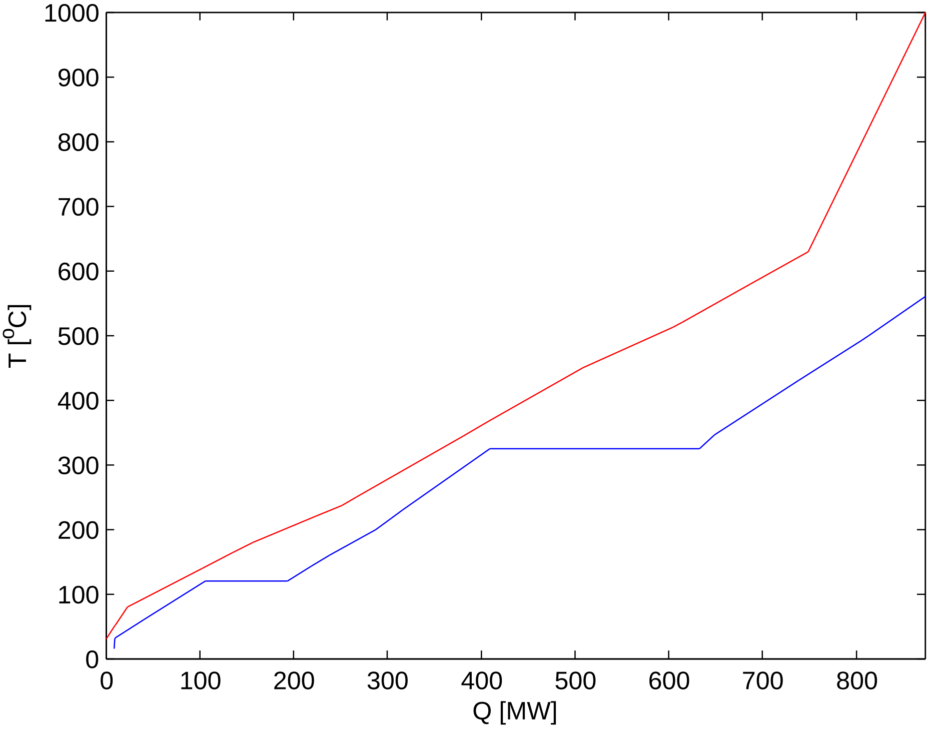


Figure 5.2: Composite curves for optimized IRCC

The composite curves and the grand composite curve do not show the results for individual streams, so the heat integration stream data are presented in table 5.2.

The demand for external cooling in the optimized process is low compared to the total cooling demand of the process, and there is no demand for external heating. The process is however pinched, with the LP econ-

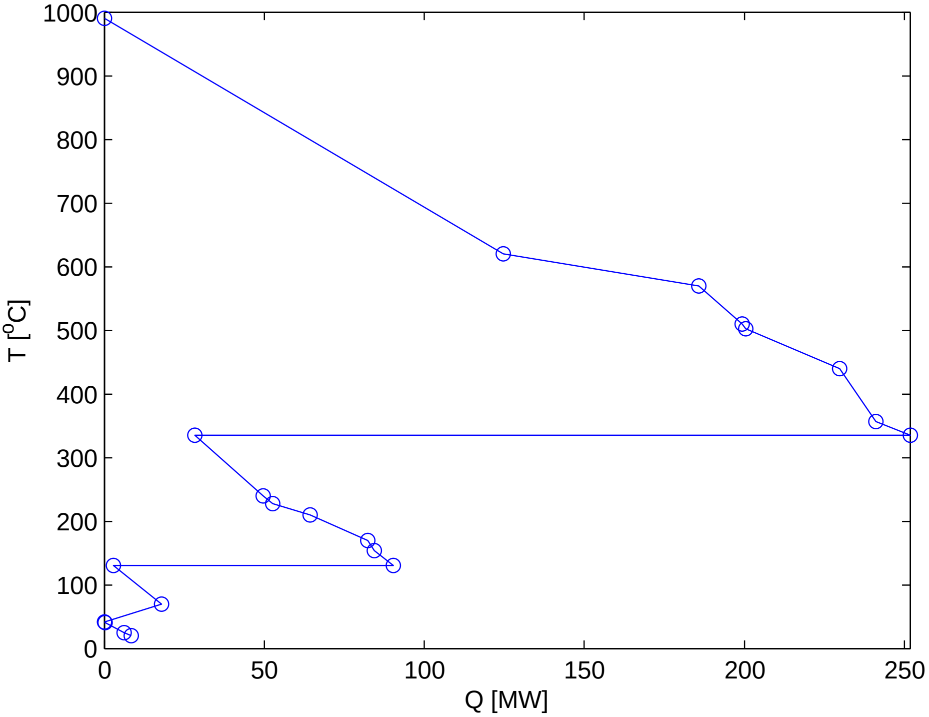


Figure 5.3: Grand composite curve for optimized IRCC

Table 5.2: Heat integration stream data

Stream	t_{IN}	t_{OUT}	heat capacity flow	heat exchanged
ATR product	1000	450	0.3369	185.3
HTS product	512.3	180	0.3192	106.1
LTS product	237.1	30	0.4518	93.56
exhaust	629.9	80	0.8882	488.4
GT fuel preheat	30.64	200	0.2299	-38.94
LPE	32	143.6	0.7867	-87.79
LPB	143.6	143.6	-	-0
LPS	143.6	230	0	-0
HPE	143.6	324.6	0.9073	-164.2
HPB	324.6	324.6	-	-223.5
HPS	324.6	560	0.647	-152.3
IP steam reheat	346.5	560	0.3533	-75.43
CH_4 preheat	15	500	0.07266	-35.24
CCU reboiler	120	120	-	-87.52

omizer being responsible for the pinch. The CCU reboiler and to a lesser extent the HP boiler are also quite close to making pinch, as seen from the parts of the grand composite curve pointing against the temperature axis.

The large “pocket” to the left of the upper part of the grand composite curve indicates that it could be potential for integrating some additional power production, but at such high temperatures it is not very realistic, with the possible exception of choosing a higher pressure level for the HP steam.

In the optimized IRCC there is no LP steam production. This is probably so because the CCU reboiler is demanding heating in the same temperature range as the LP boiler, and because there is enough high temperature heating available to generate HP stream instead. This feature is also advantageous as the process become simpler with only one steam level, and less equipment is needed.

When looking at the temperatures of individual streams, we see that the inlet temperature for the HTS is at its upper bound, while that of the LTS is at its lower bound of 180 °C. High WGS temperatures make heating available at higher temperatures, while low temperatures result in more

CO_2 and hydrogen formed, and the solution ensure good heat integration while respecting the CO_2 capture constraint. The methane feed is preheated to 500 °C, which is the upper bound. There are two possible reasons for this, the first being that with higher inlet temperatures, the ATR needs less air to reach the specified 1000 °C, because air is used to heat the ATR with reaction (2.1), and less air means less air compression work. Secondly, mixing streams of different temperatures leads to exergy losses, and the air, which constitutes most of the flow into the ATR is supplied at a temperature of 514 °C.

The high temperatures of the ATR product cooler are in reality restrictive for which streams it could exchange heat with. The problem is related to the high CO content of the stream together with high temperatures, and is called metal dusting (Grabke et al., 1993), which is a catastrophic form of corrosion. This means that this hot stream should only be mixed with streams that are "cold enough", or boiling streams, and this is not accounted for in the heat integration modeling[1]. However, if we look closer at the solution, we see that metal dusting can be avoided. The three cold streams with temperatures high enough to cause metal dusting are the superheating of HP steam, the reheating of IP steam and the preheating of methane, as their outlet temperatures are above the HP stream boiling temperature. A safe source of high temperature heat is the exhaust stream, and this stream may actually cover the entire heat demand above the temperature of the HP boiler. This is evident in figure 5.4, which shows the composite curves for the cold streams above 324.6 °C against the curve for the exhaust stream. As the ATR product stream is not needed for heating to high temperatures, metal dusting can be prevented in the optimized IRCC design.

[1] A way to deal with this is to model hot streams which may cause metal dusting as isothermal streams at their outlet temperature, such that they cannot be integrated with high temperature streams.

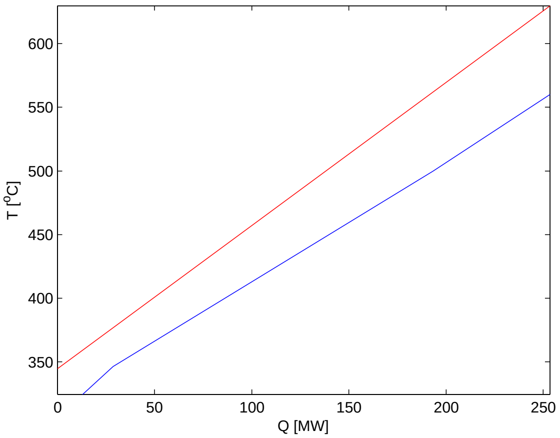


Figure 5.4: Heat exchange to avoid metal dusting

5.3 Accuracy

The model is based on some approximations, and it is important to check how the results of the model fit the reality, or at least with simulations in HYSYS. Table 5.3 shows a comparison between outputs of the model and values for process simulation results from HYSYS based on the optimized solution as inputs. The table includes the majority of the variables in our model, except those that are given as input for the HYSYS simulation.

The deviation is reasonably low for most of the variables, and for what might be the single most important one, the gas turbine (gross) power output, the deviation is only 0.4 %.

There 15 % deviations for the CO flow rates in the syngas, which also leads to a deviation for the CO_2 in the exhaust, are disappointing, even though these are less important numbers for the efficiency. This must mean that the regressed polynomial do not fit as good as it should should for the CO, but as the CO flow rate is low compared to that of CO_2, this do not have much effect on the solution. This deviation give a slight underestimation of the CO_2 capture ratio.

Table 5.3: Model outputs compared with simulation in HYSYS

Variable	unit	Model	HYSYS	Deviation
Gas turbine power	MW	707.041	709.849	-0.396 %
Second aircompressor work	MW	0	-0	- %
Extra air compressor work	MW	71.7603	72.6515	-1.227 %
HP steam turbine power	MW	74.0493	74.7811	-0.979 %
IP steam turbine power	MW	90.5535	89.5368	1.135 %
LP steam turbine power	MW	97.8402	103.435	-5.409 %
GT fuel compression work	MW	0.148669	0	∞
Mass flow of air to combustor	kg/s	640	638.945	0.165 %
Mass flow of CO2 after LTS	kg/s	60.7901	61.0586	-0.440 %
Mass flow of CO after LTS	kg/s	2.22987	1.93889	15.007 %
Mass flow of H2 after LTS	kg/s	7.62393	7.63742	-0.177 %
Mass flow of H2O after WR	kg/s	0.352254	0.349639	0.748 %
Mass flow of CO after WR	kg/s	2.22986	1.93889	15.007 %
Mass flow of CO2 after WR	kg/s	60.7561	61.0135	-0.422 %
Mass flow of H2 after WR	kg/s	7.62391	7.6374	-0.177 %
Mass flow of CH4 after WR	kg/s	0.255106	0.262419	-2.787 %
Mass flow of N2 after WR	kg/s	103.856	103.736	0.116 %
Mass flow of H2O after combustion	kg/s	68.5331	68.6648	-0.192 %
Mass flow of CO2 after combustion	kg/s	6.10469	5.67659	7.541 %
Mass flow of O2 after combustion	kg/s	86.451	86.2423	0.242 %
Mass flow of N2 after combustion	kg/s	594.592	593.657	0.157 %
Gas turbine outlet temperature	°C	629.856	630.372	-0.082 %
Extra air compressor outlet temp.	°C	514.828	516.926	-0.406 %
HP steam turbine outlet temp.	°C	346.516	344.432	0.605 %
IP steam turbine outlet temp.	°C	276.421	279.527	-1.111 %
HTS product temperature	°C	512.313	512.957	-0.125 %
LTS product temperature	°C	237.107	240.921	-1.583 %
GT fuel compressor outlet temperature	°C	30.6394	30	2.131 %
GT fuel preheating	MW	38.9406	38.2443	1.821 %
HTS product cooling	MW	106.09	105.935	0.147 %
LTS product cooling	MW	93.5612	94.4921	-0.985 %
ATR product cooling	MW	185.299	183.595	0.928 %
Exhaust cooling	MW	488.404	488.477	-0.015 %
LP steam superheating	MW	0	0	- %
HP steam boiling	MW	223.523	223.523	-0.000 %
HP steam superheating	MW	152.296	152.296	-0.000 %
IP steam reheating	MW	75.4303	76.9975	-2.035 %
Methane preheating	MW	35.241	35.2993	-0.165 %
Cooling water pump work	MW	0.0176	0.0131	34.131 %

The actual work of the syngas compressor should be zero, while the model reports 0.15 MW. This has only a negligible impact on the efficiency, but it is worth examining how good the modeling of this compressor really is, as this error is large in relative terms. If we take a look at the formula used in calculating the syngas compression work, we will see that this, in relative terms, bad fit is only occurring when the compression work is close to zero. As can be seen from table 4.2, the regression function do only depend on the feed pressure (input variable number 3), and the plot of simulation results versus the fitted curve in figure 5.5 shows that the approximating function actually fits the data quite nicely.

The 5.4 % deviation for the LP steam turbine power output is more serious, as these 5.5 MW would change the plant efficiency with almost 0.5 percentage points. This deviation is caused by some conservative modeling. The inlet temperature to the LP turbine is in reality the mixing temperature of the superheated LP steam, and the steam out of the IP steam turbine, while the calculations are based on the superheated LP steam temperature, which is always the lower of the two. As higher temperature yields higher power output, this is an underestimation, and particularly so for these results, as there is no LP steam production in the optimized process design. However, modeling it more accurately would complicate the model, and as it is believed that this would not change the overall structure of the solution, this underestimation is kept as it is.

Another deviation worth noticing is that of the heat flow rate of the steam reheater. The reheater is modeled with one fixed number for specific heat capacity, while in reality this is a variable of both temperature and pressure, and steam extraction at a high pressure in the solution, the pressure and temperature of the reheater are high, and the number used for heat capacity in the model is too low. However, the deviation is acceptably low, given the savings in model complexity of assuming constant heat capacity.

Finally, the relative deviation for cooling water pump work is large because external cooling is determined by a difference of much higher numbers, like the exhaust cooling of 488 MW, and with this in mind, this is not troubling at all, and the effect on the efficiency is minimal.

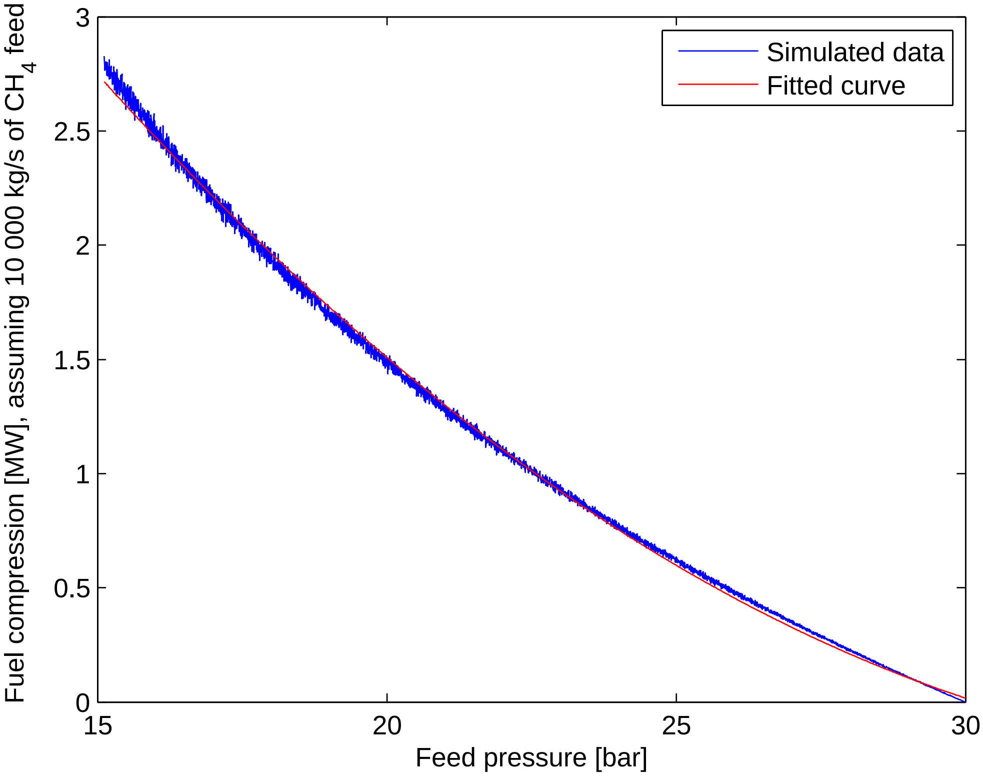


Figure 5.5: Simulation data versus fitted curve for syngas compression

It is worth noting that the efficiency of the optimized process corrected according to the HYSYS simulation is 49.97 %, which is 1.3 % or 0.62 percentage points higher than the efficiency calculated in the optimization model.

5.4 Comparison with other IRCCs

It is always difficult to compare efficiencies of different plants, as there is a wide range of parameters and assumptions that might differ, but we will however look at some related processes from the literature as a basis for comparison. All efficiencies reported in the following are based on LHV.

Nord and Bolland (2011) report an efficiency of 45.3 % for an IRCC plant with CO_2 capture similar to ours, with two steam levels. The focus of their work is on the HRSG design, including heat integration with the WGS coolers, but they do also use supplementary firing for the steam cycles. The reformer is an ATR, while the carbon is capture with a hot potassium carbonate absorption process. The IRCC plant of Nord et al. (2009) is also similar to ours, and have a net plant efficiency of 41.9 %, but their work is more focused on operability. This process do also use autothermal reforming, and an amine absorption process for capture.

Manzolini et al. (2011a,b) present results for another kind of IRCC, which uses sorption enhanced water gas shift (SEWGS) to combine water-gas shift and CO_2 capture. In this process, the reforming is done both with a gas-heated reformer (GHR) and an ATR. With a high level of heat integration, they reach an electric efficiency of 50.9 %, while efficiencies of 49.9 % and 50.3 % are reported for reference plants with post-combustion and pre-combustion capture, respectively, both using amine-based absorption. The reference plant with pre-combustion capture is interesting, as it is very similar to our process, except that the is a combination of GHR and ATR.

When comparing the results of our model with these works, it is important to be aware of some major differences. The process of Nord and Bolland (2011) is very relevant as it considers heat integration on a process very similar to ours, except the hot potassium carbonate absorption, which should not change the results dramatically. The precombustion reference process of Manzolini et al. (2011b) is also very relevant for comparison, and it is interesting to note the large difference in efficiency, with 5 percentage points more for the latter process. The high efficiency might come from the heat integration of the ATR product stream with the GHR in Manzolini et al. (2011b), and this high-temperature heat exchange might lead to metal dusting.

The efficiency of our optimized IRCC is much higher than that of Nord and Bolland (2011), but slightly lower than the reference plant of Manzolini et al. (2011b). If the high temperature heat integration of the latter is a problem, we might have found a very promising solution. However, in our model we assume maximum possible heat integration, without considering the actual heat exchanger network, and this is quite optimistic, even with the conservative choice of HRAT at 20 °C. Our model is also slightly limited because we look at a simplified process. Additionally, issues related to operability, controllability and flexibility are not considered in the design of our process, as this is outside the scope of this work.

5.5 Results for alternative scenarios

Table 5.4 shows the results for the four different scenarios described in section 5.5, with numbering as for the headlines. Flow-sheets, composite curves and grand composite curves for each scenario are available in appendix D.

5.5.1 Maximizing power

When maximizing the net power output instead of the efficiency, we do indeed get higher power output. This is accomplished by using even more methane, which reduce the efficiency. Interestingly, the ATR pressure is 18 bar in this solution, which is the lower bound, and not the maximum 30 bar as for the base case. The resulting steam to carbon ratio is at its upper bound of 2, which is also the opposite of the other cases. The reason for this is that, as more methane is supplied, the temperature in the combustor

Table 5.4: Results for alternative scenarios

Scenario	Base case	1	2	3	4	5	Unit
Gas turbine power (gross)	707.04	750.90	583.07	663.48	707.35	705.23	MW
Main air compressor work	-288.76	-288.76	-288.76	-288.76	-288.76	-288.76	MW
Second air compressor work	-0.00	-0.00	-13.73	-0.00	-0.00	-0.00	MW
Extra air compressor work	-71.76	-79.80	**-0.00**	-63.71	-71.95	-71.23	MW
HP steam turbine power	74.05	128.46	60.81	68.19	77.22	73.20	MW
IP steam turbine power	90.55	84.57	74.47	79.62	59.26	90.51	MW
LP steam turbine power	97.84	114.83	80.46	87.99	110.25	97.79	MW
CO_2 compression	-19.95	-24.70	-16.41	-18.07	-19.98	-18.59	MW
GT fuel compression work	-0.15	-19.89	-0.12	-1.06	-0.15	-0.15	MW
Auxiliary power consumption	-4.51	-5.93	-3.71	-4.04	-4.81	-4.47	MW
Net power output	584.35	**659.68**	476.07	523.63	568.42	583.53	MW
Energy input (CH_4 LHV)	1183.89	1408.32	973.43	1071.81	1185.37	1171.47	MW
Efficiency	49.36	46.84	48.91	48.85	47.95	**49.81**	%
Cold utilities (external cooling)	-8.39	-19.89	-6.80	-9.15	-8.40	-7.29	MW
Hot utilities (external heating)	-0.00	-0.00	-0.00	-0.00	-0.00	-0.00	MW
CO_2 capture ratio	90.00	**93.64**	90.00	90.02	90.00	**84.74**	%
ATR pressure	30.00	**18.00**	30.00	**28.69**	30.00	30.00	bar
Steam to carbon ratio	1.07	**2.00**	1.06	1.00	1.07	1.00	

will rise because the amount of air supplied to the combustor is limited to 640 kg/s. This must be compensated for, and by adding relatively more steam to the ATR, more air is needed there to reach the specified ATR temperature of 1000 °C, which give more nitrogen in the gas turbine fuel. The lower ATR pressure is chosen because increased steam supply to the ATR means that more power is lost if the stream is extracted from the HP steam cycle at a high pressure, thus it is more efficient to compress the syngas because this is done after the water removal, such that the steam does not need to be compressed. The higher steam to carbon ratio also leads to a higher CO_2 capture ratio.

The mechanisms behind these results are not necessarily wanted, but arise because only the size of the main air compressor is limited. To make the model more realistic when maximizing power, a cost should be assigned to the usage of methane, or limits should be given for the methane supply or flow through the gas turbine. Such changes are easy to implement.

There is also significantly more cooling in this solution, although this do not impact the objective much. As this solution was not proven optimal, these results are not necessarily the best possible, but the optimum is not likely to be very different as the gap between the upper and lower bounds was very low.

5.5.2 No choice of ATR air compressor

In the other solutions, the air to the ATR is compressed with the extra air compressor. When considering only the main air compressor and a second compressor to compress the air further, the net power output and the fuel consumption are naturally reduced, as not all the air from the main compressor can be used in the combustor. This choice of compressor reduces the efficiency by 0.45 percentage points.

5.5.3 Lower turbine inlet temperature

With a turbine inlet temperature of 1285 °C instead of 1350 °C, more air is needed per amount of fuel to keep the temperature down, and this is reflected in the solution with less fuel consumption leading to lower power output. The outlet temperature of the turbine is reduced from 630 to 589 °C, which means less heating for the steam cycles, such that also the steam turbines generate less power, even though the steam to the ATR is extracted at a slightly lower pressure. For this scenario, the steam to carbon ratio is at its lower bound of 1, and this is possible because the HTS have a lower inlet temperature to enhance the water gas shift. As expected, the efficiency for this process is lower, and the reduction is of 0.51 percentage points.

5.5.4 Higher steam pressure levels

Choosing 130 bar for HP and 8 bar for the LP steam cycle instead of 120 and 4, respectively, was expected to improve the efficiency slightly, as the heat from the syngas coolers and the gas turbine exhaust could be recovered at somewhat higher temperatures. However, the solution found for this scenario was different, with the efficiency reduced by 2.4 percentage points. The power output from the steam turbines is actually lower than in the base case, and changes in the low pressure level from 4 to 8 bar cause less power generated in the IP turbine and relatively more in the LP turbine.

The reason for the reduced efficiency should be investigated, but it should also be noted that the same temperatures were used for HP steam superheating and reheat as for the base case, and these temperatures have a larger impact on the efficiency than the pressure.

5.5.5 Lower limit for CO_2 capture

For the scenario without the constraint for minimum 90 % CO_2 captured, only 84.7 % is captured. Omitting the capture constraint lets both WGS reactors be operated at their highest temperatures to get more high temperature heat for integration, and the steam to carbon ratio is at its lower bound to get more steam through the IP turbine. The efficiency is, naturally, slightly higher than in the base case.

5.6 Computational performance

When trying to solve the complete model without relaxation, it does not converge after 24 hours. Moreover, the gap between lower bound and best solution found is 22.4 %, and have not improved during the last four hours. The best solution in these 24 hours was found after almost nine hours, with an objective value of 48.24 (% efficiency), while a solution of 45.62 % was found after less than seven minutes, and poorer solutions were found within seconds.

As these results were not satisfying, the relaxation of section 4.1.4 was made to get better bounds and solutions. With the iterative algorithm, the unrelaxed model was run for five minutes in step 1. In the intermediate solution found after five minutes, the pinch was at the inlet of the LP steam economizer, and thus LPE was added as the first member of the pinch candidate subset.

The relaxed problem considering only LPE as a pinch candidate, was solved to optimum in 7:09 minutes, but it was not feasible in the original problem. In this solution, the stream actually causing pinch was the CCU reboiler, and consequently this was added to the pinch candidate subset. With this relaxation, there were only two binary variables that could affect the solution.

The relaxed problem with $\mathcal{S}_{P*} = \{LPE, CCUreboiler\}$ took 4:24 hours to solve, and the LPE was responsible for pinch again, thus the solution must be optimal. This relaxation had four binary variables in effect.

The alternative scenarios were also solved with relaxation with only the LPE and the CCU reboiler considered as pinch candidates. The scenario for maximizing power required manual branching on the ATR pressure variable to find the optimal solution, as there were some problems as discussed in the next section, while the solution times for the other scenarios are as follows: Lower TIT: 1:48 hours, no extra air compressor: 24:25 minutes, higher steam pressures: 2:34 hours, less CO_2 capture: 47:45 minutes.

5.6.1 Reliability of BARON

BARON is a global solver, hence it should be able to find global optima, or at least not stop solving before it is known whether the solution is the global optimum, and in this work we have assumed that it is so. However, there might be errors in any computer code, and during the work with the IRCC optimization model, some situation questioning the reliability of BARON have occurred.

For an earlier instance of the model, BARON stated that it found the (global) optimum, while a better solution could be found by stopping BARON and letting CONOPT finish with local optimization around the current best solution found by BARON. This solution was better than the lower bound calculated by BARON in the first run. In another case, BARON actually found a solution itself which was better than the lower bound it had calculated.

These issues are distressing, but the difference between the lower bound and the better solution was never large, and the results of BARON are still assumed to hold.

Another peculiar issue is that in some cases, BARON ends just after presolving, and returns a solution, usually of low quality, which is reported to be optimal. This was the case for the scenario of maximizing power. This strange behavior was avoided by dividing the feasible interval for the ATR pressure, and solving the problem with for each sub-interval, but why the issue was avoided with this manual branching procedure is a mystery.

Chapter 6

Conclusion

The goal of this work was to make an optimization model for designing an IRCC with reduced energy consumption, including some use of metamodeling based on process simulation to approximate difficult relations.

The optimization model was successfully made and implemented. Its results are quite accurate, with a deviation in the objective value of about 1 % compared to process simulations in HYSYS. The optimization model is hard to solve for BARON, but with a relaxation procedure based on pinch analysis insights, optimal solutions was found and proven globally optimal within a few hours for different sets of input data. In other words, the balance between reality representation and solution speed is satisfactory, but further improvements are possible, as discussed in the next section.

The optimized IRCC process resulting from our model, with a CO_2 capture ratio of 90 percent, has an efficiency of 49.97 % when adjusted for approximating errors, and this is quite good compared to other IRCCs in the litterature. With extensive heat integration, the optimized process do not need any external heating or supplementary firing, and the utility cooling demand is less than one percent of the total cooling demand in the process. These results suggest that the optimization model of this work might become fruitful if more research is done to confirm the results under less idealistic assumptions.

6.1 Further work

There are many possible modifications that could improve the model, but it is important to accept that, with the technology of today, it is impossible to make an optimization model which both represents the process precisely and allows for optimal solutions to be found within short time.

An important piece of work that has to be done for exploiting the capability of this model, is to design a more detailed version of the optimized process. By designing a less simplified process, closer to what might be appropriate to build, based on the results from the IRCC optimization model, it is possible to see if the results are beneficial, or if the model do not represent the reality good enough. A large part of this work will consist of the heat exchanger network synthesis. Furthermore, it is important to account for operability, controllability and flexibility issues.

With respect to the heat integration model, it should be investigated whether the model of Grossmann et al. (1998) may enhance the computational performance, and whether the approximating functions used by Duran and Grossmann (1986) are acceptably accurate. The model is implemented in a way that makes such a substitution simple.

Another possible way of improving the model is applying metamodeling on a larger part of the model, with the gas turbine and the air compressors as apparent candidates, as the physical modeling of these include the most challenging equations for the solver. If the metamodeling is to be extended, higher order polynomials and possibly other regression functions should be considered.

As the current model is hard to solve without the relaxation, and will probably be harder if more features are added to the model, it could be worth looking at heuristics like multi-start local search, simulated annealing or other methods for solving the model. The issues with BARON discussed in section 5.6.1 are additional arguments for using heuristics.

If a successful solution method is found, more accurate modeling of the actual IRCC process should be investigated, for example by modeling the LP steam turbine and the CCU more accurately, considering natural gas with the associated pre-processing instead of pure methane and including multiple stages and intercooling for the air compressors. Some of these improvements might not even complicate the model.

Finally, an improvement that will inevitably complicate the model, but is in line with the holistic ideal behind this work, is including more parameters as variables, for example the steam levels and the combustion temperature, thus making the model more general. While this could lead to better designs, it is important to work with the solving if the model is to be extended.

Bibliography

Aspen HYSYS Tutorials and Applications. Aspen Technology, Cambridge, MA, USA, 2006.

O. T. Berglihn. Hysyslib, May 9, 2011. URL `http://www.pvv.org/~olafb/software/hysyslib/`.

O. Bolland. Power Generation: CO_2 Capture and Storage. Draft manuscript for a coming book, September 2010.

A. Brooke, D. Kendrick, A. Meeraus, R. Raman, and R. E. Rosenthal. *GAMS, A User's Guide.* GAMS Development Corporation, Washington, DC, USA, 2003.

P. Chiesa, G. Lozza, and L. Mazzocchi. Using hydrogen as gas turbine fuel. *Journal of Engineering for Gas Turbines and Power*, 127:73–80, 2005.

J. David and H. Herzog. The cost of carbon capture. In *Fifth international conference on greenhouse gas control technologies, Cairns, Australia*, pages 13–16, 2000.

M.A. Duran and I.E. Grossmann. Simultaneous optimization and heat integration of chemical processes. *AIChE journal*, 32(1):123–138, 1986.

F. Emun, M. Gadalla, T. Majozi, and D. Boer. Integrated gasification combined cycle (IGCC) process simulation and optimization. *Computers & Chemical Engineering*, 34(3):331–338, 2010. ISSN 0098-1354.

M.C. Ferris, R. Jain, and S. Dirkse. *GDXMRW: Interfacing GAMS and MATLAB*, 2010.

J. Gibbins and H. Chalmers. Carbon capture and storage. *Energy Policy*, 36(12):4317–4322, 2008. ISSN 0301-4215.

HJ Grabke, R. Krajak, and JC Nava Paz. On the mechanism of catastrophic carburization: metal dusting. *Corrosion science*, 35(5-8):1141–1145, 1993.

I.E. Grossmann, H. Yeomans, and Z. Kravanja. A rigorous disjunctive optimization model for simultaneous flowsheet optimization and heat integration. *Computers & chemical engineering*, 22:S157–S164, 1998.

T. Gundersen. A process integration primer. Technical report, SINTEF Energy Research and International Energy Agency, Trondheim, Norway, 2000.

E. L. Johnsen. Metamodeling approach for optimization of an integrated reforming combined cycle with CO2 capture, focusing on process integration. Project thesis in the course TIØ4500, NTNU, 2010.

B. Linnhoff and E. Hindmarsh. The pinch design method for heat exchanger networks. *Chemical Engineering Science*, 38(5):745 – 763, 1983. ISSN 0009-2509. doi: DOI:10.1016/0009-2509(83)80185-7. URL `http://www.sciencedirect.com/science/article/pii/0009250983801857`.

G. Manzolini, E. Macchi, M. Binotti, and M. Gazzani. Integration of sewgs for carbon capture in natural gas combined cycle. part a: Thermodynamic performances. *International Journal of Greenhouse Gas Control*, 5:200–213, 2011a.

G. Manzolini, E. Macchi, M. Binotti, and M. Gazzani. Integration of SEWGS for carbon capture in natural gas combined cycle. part b: Reference case comparison. *International Journal of Greenhouse Gas Control*, 5:214–225, 2011b.

E. Martelli. *Numerical optimization of heat recovery steam cycles for highly integrated energy systems & application to low carbon emission plants.* PhD dissertation, Politecnico di Milano, Department of energy, 2010.

B. A. McCarl, A. Meeraus, P. van der Eijk, M. Busseck, S. Dirkse, P. Steacy, and F. Nelissen. *McCarl GAMS User Guide.* GAMS Development Corporation, Washington, DC, USA, 2011.

M.D. McKay, R.J. Beckman, and WJ Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

B. Metz, O. Davidson, H. de Coninck, M. Loos, and L. Meyer. IPCC special report on carbon dioxide capture and storage. Technical report, Intergovernmental Panel on Climate Change, Geneva, Switzerland, 2005.

L.O. Nord and O. Bolland. Hrsg design for integrated reforming combined cycle with co capture. *Journal of Engineering for Gas Turbines and Power*, 133:011702, 2011.

L.O. Nord, R. Anantharaman, and O. Bolland. Design and off-design analyses of a pre-combustion CO2 capture process in a natural gas combined cycle power plant. *International Journal of Greenhouse Gas Control*, 3 (4):385–392, 2009. ISSN 1750-5836.

B. Nygreen, M. Christiansen, K. Haugen, T. Bjørkvoll, and Ø. Kristiansen. Modeling Norwegian petroleum production and transportation. *Annals of Operations Research*, 82:251–268, 1998. ISSN 0254-5330.

K. Palmer and M. Realff. Metamodeling Approach to Optimization of Steady-State Flowsheet Simulations: Model Generation. *Chemical Engineering Research and Design*, 80(7):760–772, 2002a. ISSN 0263-8762.

K. Palmer and M. Realff. Optimization and validation of steady-state flowsheet simulation metamodels. *Chemical Engineering Research and Design*, 80(7):773–782, 2002b. ISSN 0263-8762.

C. Rhodes, G.J. Hutchings, and A.M. Ward. Water-gas shift reaction: finding the mechanistic boundary. *Catalysis Today*, 23(1):43–58, 1995. ISSN 0920-5861.

C. Rhodes, B. Peter Williams, F. King, and G.J. Hutchings. Promotion of Fe3O4/Cr2O3 high temperature water gas shift catalyst. *Catalysis Communications*, 3(8):381–384, 2002. ISSN 1566-7367.

J. Sadhukhan and X.X. Zhu. Integration strategy of gasification technology: A gateway to future refining. *Ind. Eng. Chem. Res*, 41(6):1528–1544, 2002. ISSN 0888-5885.

N. Sahinidis. *BARON — Branch And Reduce Optimization Navigator*. University of Illinois at Urbana-Champaign, Urbana, IL, USA, 2000.

N. Sahinidis and M. Tawarmalani. *BARON*. GAMS Development Corporation, Washington, DC, USA, 2010.

List of Figures

List of Tables

Appendix A

The complete IRCC optimization model

From section 3.2.2:

$$\text{minimize } z \tag{A.1}$$

subject to

$$Q_{LHV,CH_4}\ f_{in(ATR),CH_4}\ z = \sum_{e \in E_P} w_e - \sum_{e \in E_W} w_e \tag{A.2}$$

From section 3.3:

$$x_d - \left(\alpha_{d,0} + \sum_{i \in \mathcal{I}_{1,d}} \alpha_{d,i}\ x_i + \sum_{i \in \mathcal{I}_{2,d}} \sum_{j \in \mathcal{J}_{d,i}} \beta_{d,i,j}\ x_i\ x_j \right) f_{in(ATR),CH_4} = 0 \quad \forall d \in \mathcal{D}_F \tag{A.3}$$

$$x_d - \sum_{i \in \mathcal{I}_{1,d}} \alpha_{d,i}\ x_i - \sum_{i \in \mathcal{I}_{2,d}} \sum_{j \in \mathcal{J}_{d,i}} \beta_{d,i,j}\ x_i\ x_j = \alpha_{d,0} \quad \forall d \in \mathcal{D}_S \tag{A.4}$$

$$w_e - W_e\ f_{in(e),H_2O} = 0 \forall e \in \mathcal{E}_{T0} \tag{A.5}$$

$$w_e - (B_{W,e} + A_{W,e}\ p_{in(ATR)}) f_{in(e),H_2O} = 0 \qquad \forall e \in \mathcal{E}_{T1} \tag{A.6}$$

$$t_e - (B_{T,e} + A_{T,e}\ p_{in(ATR)}) = 0 \qquad \forall e \in \mathcal{E}_{T1} \tag{A.7}$$

From section 3.4:

$$\sum_{s\in\mathcal{S}_{IN,e}}\sum_{k\in\mathcal{K}_{F,s}} Q_{LHV,k}\ f_{s,k} = \sum_{s\in\mathcal{S}_{IN,e}}\sum_{k\in\mathcal{K}_s} Q_{s,k} f_{s,k} \quad \forall e\in\mathcal{E}_R \tag{A.8}$$

$$f_{out(e),k} = \sum_{s\in\mathcal{S}_{IN,e}}\sum_{l\in\mathcal{K}_s} G_{e,k,l}\ f_{s,l} \quad \forall k\in\mathcal{K}_{out(e)}, e\in\mathcal{E}_R \tag{A.9}$$

$$w_e - \sum_{k\in K_{in(e)}} \eta_e\ f_{in(e),k}\sum_{n=1}^{5} H_{k,n}({T_{in(e)}}^n - {t_{I,e}}^n) = 0 \qquad \forall e\in\mathcal{E}_{T2} \tag{A.10}$$

$$t_{I,e} - T_{in(e)}\left(\frac{p_{out(e)}}{P_{in(e)}}\right)^{a_e} = 0 \qquad \forall e\in\mathcal{E}_{T2} \tag{A.11}$$

$$t_{out(e)} - |\eta_e|\ t_{I,e} = (1-|\eta_e|)\ T_{in(e)} \qquad \forall e\in\mathcal{E}_{T2} \tag{A.12}$$

$$a_e\ w_e - \eta_e\ (T_{in(e)} - t_{I,e})\sum_{k\in\mathcal{K}_e} R_k\ f_{k,in(e)} = 0 \qquad \forall e\in\mathcal{E}_{T2} \tag{A.13}$$

$$f_{in(e),k} = \sum_{s\in\mathcal{S}_{OUT,e}} f_{s,k} \qquad \forall e\in\mathcal{E}_S, k\in\mathcal{K}_{in(e)} \tag{A.14}$$

$$f_{out(e),k} = \sum_{s\in\mathcal{S}_{IN,e}} f_{s,k} \qquad \forall e\in\mathcal{E}_M, k\in\mathcal{K}_{in(e)} \tag{A.15}$$

$$t_{out(e)}\sum_{k\in K_{out(e)}} f_{out(e),k} = \sum_{s\in\mathcal{S}_{IN,e}} t_s \sum_{k\in\mathcal{K}_s} f_{s,k} \qquad \forall e\in\mathcal{E}_M \tag{A.16}$$

$$f_{s,k} = V_{e,s,k}\ f_{in(e),k} \qquad \forall e\in\mathcal{E}_{SE}, s\in\mathcal{S}_{OUT,e}, k\in\mathcal{K}_{in(e)} \tag{A.17}$$

$$q_{CCU} = Q_{CC,CO_2}\ f_{CC,CO_2} \tag{A.18}$$

$$w_{CCU} = W_{CCU}\ f_{CC,CO_2} \tag{A.19}$$

$$f_{CC,CO_2} \geq U\ f_{in(ATR),CH_g} \tag{A.20}$$

From section 3.5.1:

$$\begin{aligned} q_H \geq & \sum_{c\in C} C_{P,c}(t_{OUT,c} - t_{M,c,p}) f_{C,c} + \sum_{c\in\mathcal{C}_I} f_{QP,c,p}\ - \\ & \sum_{h\in H} C_{P,h}(t_{IN,h} - t_{M,h,p}) f_{C,h} - \sum_{h\in\mathcal{H}_I} f_{QP,h,p} \end{aligned} \qquad \forall p\in\mathcal{S}_P \tag{A.21}$$

$$q_C - q_H = \sum_{s \in \mathcal{S}_N} (t_{IN,s} - t_{OUT,s}) f_{C,s} + \sum_{h \in \mathcal{H}_I} f_{Q,h} - \sum_{c \in \mathcal{C}_I} f_{Q,c} \tag{A.22}$$

$$t_{M,h,p} \geq t_{OUT,h} \qquad \forall h \in \mathcal{H},\ p \in \mathcal{S}_P \tag{A.23}$$
$$t_{M,h,p} \geq t_{IN,p} + T_{ADJ,p} - T_{ADJ,h} - M_{1,h,p} y_{h,p} \qquad \forall h \in \mathcal{H},\ p \in \mathcal{S}_P \tag{A.24}$$
$$t_{M,h,p} \geq t_{IN,h} - M_{2,h,p}(1 - y_{h,p}) \qquad \forall h \in \mathcal{H},\ p \in \mathcal{S}_P \tag{A.25}$$
$$t_{M,c,p} \leq t_{OUT,c} \qquad \forall c \in \mathcal{C},\ p \in \mathcal{S}_P \tag{A.26}$$
$$t_{M,c,p} \leq t_{IN,p} + T_{ADJ,p} - T_{ADJ,c} - M_{1,c,p}(1 - y_{c,p}) \forall c \in \mathcal{C},\ p \in \mathcal{S}_P \tag{A.27}$$
$$t_{M,c,p} \leq t_{IN,c} - M_{2,c,p} y_{c,p} \qquad \forall c \in \mathcal{C},\ p \in \mathcal{S}_P \tag{A.28}$$

$$t_{IN,h} \geq t_{IN,p} + T_{ADJ,p} - T_{ADJ,h} - M_{1,h,p} y_{h,p} \qquad \forall h \in \mathcal{H}_I,\ p \in \mathcal{S}_P \tag{A.29}$$
$$t_{IN,c} \leq t_{IN,p} + T_{ADJ,p} - T_{ADJ,c} + M_{1,c,p}(1 - y_{c,p}) \forall c \in \mathcal{C}_I,\ p \in \mathcal{S}_P \tag{A.30}$$
$$f_{QP,h,p} \leq M_{2,h,p}(1 - y_{h,p}) \qquad \forall h \in \mathcal{H}_I,\ p \in \mathcal{S}_P \tag{A.31}$$
$$f_{QP,h,p} \leq \ f_{Q,h} \qquad \forall h \in \mathcal{H}_I,\ p \in \mathcal{S}_P \tag{A.32}$$
$$f_{QP,c,p} \geq f_{Q,c} - M_{2,c,p} y_{c,p} \qquad \forall c \in \mathcal{C}_I,\ p \in \mathcal{S}_P \tag{A.33}$$

$$t_{IN,s} = t_p \forall (s,p) \in \mathcal{S}_{S,TI} \tag{A.34}$$
$$t_{OUT,s} = t_p \forall (s,p) \in \mathcal{S}_{S,TO} \tag{A.35}$$
$$f_{C,s} = \sum_{k \in \mathcal{K}_p} C_{P,p,k}\ f_{p,k} \forall (s,p) \in \mathcal{S}_{S,F1} \tag{A.36}$$
$$f_{C,s} = f_{C,p} \forall (s,p) \in \mathcal{S}_{S,F2} \tag{A.37}$$
$$f_{Q,s} = q_e \forall (s,e) \in \mathcal{S}_{E,Q}, \tag{A.38}$$

$$Q_{p,H_2O}\ f_{C,s} = C_{P,s,H_2O} f_{Q,p} \forall (s,p) \in \mathcal{S}_{S,Q} \tag{A.39}$$

$$y_{s,p} \in \{0,\ 1\} \qquad \forall s \in \mathcal{S},\ p \in \mathcal{S}_P \tag{A.40}$$
$$q_H, q_C, t_{IN,s}, t_{OUT,s}, t_{M,s,p}, f_{C,s}, f_{Q,s}, f_{QP,s,p} \geq 0 \tag{A.41}$$

Upper and lower bounds:

$$\underline{x}_i \le x_i \le \overline{x}_i \qquad \forall i \in \mathcal{I} \cup \mathcal{D} \tag{A.42}$$

$$\underline{w}_e \le e_e \le \overline{w}_e \qquad \forall e \in \mathcal{W}_P \cup \mathcal{E}_W \tag{A.43}$$

$$\underline{f}_{s,k} \le f_{s,k} \le \overline{f}_{s,k} \qquad \forall s \in \mathcal{S}_1, k \in \mathcal{K}_s \tag{A.44}$$

$$\underline{p}_s \le p_s \le \overline{p}_s \qquad \forall s \in \mathcal{S}_1 \tag{A.45}$$

$$\underline{t}_s \le t_s \le \overline{t}_s \qquad \forall s \in \mathcal{S}_1 \tag{A.46}$$

$$\underline{t}_{I,e} \le t_{I,e} \le \overline{t}_{I,e} \qquad \forall e \in \mathcal{E}_{T2} \tag{A.47}$$

$$\underline{a}_e \le a_e \le \overline{a}_e \qquad \forall a \in \mathcal{E}_{T2} \tag{A.48}$$

$$\underline{t}_s \le t_s \le \overline{t}_s \qquad \forall s \in \mathcal{S}_1 \tag{A.49}$$

$$\underline{q}_e \le q_e \le \overline{q}_e \qquad \forall e \in \{CCU\} \tag{A.50}$$

$$\underline{t}_{IN,s} \le t_{IN,s} \le \overline{t}_{IN,s} \qquad \forall s \in \mathcal{S} \tag{A.51}$$

$$\underline{t}_{OUT,s} \le t_{OUT,s} \le \overline{t}_{OUT,s} \qquad \forall s \in \mathcal{S}_N \tag{A.52}$$

$$\underline{t}_{M,s,p} \le t_{M,s,p} \le \overline{t}_{M,s,p} \qquad \forall s \in \mathcal{S}_N, p \in \mathcal{S}_P \tag{A.53}$$

$$\underline{f}_{C,s} \le f_{C,s} \le \overline{f}_{C,s} \qquad \forall s \in \mathcal{S}_N \tag{A.54}$$

$$\underline{f}_{Q,s} \le f_{Q,s} \le \overline{f}_{Q,s} \qquad \forall s \in \mathcal{S}_I \tag{A.55}$$

$$\underline{f}_{QP,s,p} \le f_{QP,s,p} \le \overline{f}_{QP,s,p} \qquad \forall s \in \mathcal{S}_I, p \in \mathcal{S}_P \tag{A.56}$$

Appendix B

Regression coefficients

The regression coefficients resulting from the metamodeling approach are presented in tables B.2 and B.3. It is important to be aware of that these coefficients are for the variables scaled to the interval $\left\langle -\frac{1}{2}, \frac{1}{2} \right\rangle$ between the maximum and minimum values of tables 4.1 and B.1.

Table B.1: Minumum and maximum values for simulation output variables

d	Variable	unit	Minimum	Maximum
1	Mass flow of CO_2 after LTS	kg/s	22.6	27.3
2	Mass flow of CO after LTS	kg/s	0.0562	2.89
3	Mass flow of H_2 after LTS	kg/s	2.93	3.29
4	HTS product temperature	°C	342	537
5	LTS product temperature	°C	172	316
6	GT fuel preheating	MW/°C	0.0105	0.0281
7	HTS product heat capacity flow	MW/°C	0.0344	0.0489
8	LTS product heat capacity flow	MW/°C	0.0393	0.135
9	ATR product heat capacity flow	MW/°C	0.036	0.0517
10	Syngas compressor outlet temperature	°C	30	134
11	Mass flow of H_2O after WR	kg/s	0.144	0.328
12	Mass flow of CO after WR	kg/s	0.0562	2.89
13	Mass flow of CO_2 after WR	kg/s	22.6	27.3
14	Mass flow of H_2 after WR	kg/s	2.93	3.29
15	Mass flow of CH_4 after WR	kg/s	0.0064	0.166
16	Mass flow of N_2 after WR	kg/s	42.2	54.4
17	GT fuel compression work	MW	0	2.83

Table B.2: Regression coefficients, part 1

d	1	2	3	4	5	6	7	8	9
$\alpha_{d,0}$	0.367	-0.377	0.095	0.030	-	0.104	-	-0.141	-
$\alpha_{d,1}$	-0.054	0.050	0.337	-	-	-	-0.090	-0.055	-0.089
$\alpha_{d,2}$	-0.255	0.269	-0.151	-	0.642	-	-	-0.148	-
$\alpha_{d,3}$	-	-	-0.067	-	-	0.909	-	-	-
$\alpha_{d,4}$	-0.094	0.099	-0.056	0.913	0.219	-	-	-0.046	0.051
$\alpha_{d,5}$	0.496	-0.482	-0.251	-0.082	-0.230	-	0.814	0.736	0.815
$\alpha_{d,6}$	-0.043	0.041	0.280	-	-	-	-0.075	-0.046	-0.074
$\alpha_{d,7}$	-	-	0.088	-	-	-	-	-	-
$\beta_{d,1,1}$	-	-	0.067	-	-	-	-	-	-
$\beta_{d,2,2}$	-0.090	0.096	-0.054	-	-	-	-	0.081	-
$\beta_{d,3,3}$	-	-	-	-	-	-0.484	-	-	-
$\beta_{d,4,4}$	-	-	-	-	-	-	-	-	-
$\beta_{d,5,5}$	-0.812	0.814	-0.554	-	-	-	-	0.124	-
$\beta_{d,6,6}$	-	-	-	-	-	-	-	-	-
$\beta_{d,7,7}$	-	-	-	-	-	-	-	-	-
$\beta_{d,1,2}$	-	-	-	-	-	-	-	-	-
$\beta_{d,1,3}$	-	-	-	-	-	-	-	-	-
$\beta_{d,1,4}$	-	-	-	-	-	-	-	-	-
$\beta_{d,1,5}$	0.157	-0.151	0.107	-	-	-	-	-	-
$\beta_{d,1,6}$	-	-	-	-	-	-	-	-	-
$\beta_{d,1,7}$	-	-	-	-	-	-	-	-	-
$\beta_{d,2,3}$	-	-	-	-	-	-	-	-	-
$\beta_{d,2,4}$	-	0.070	-0.042	-	-	-	-	-	-
$\beta_{d,2,5}$	0.404	-0.426	0.241	-	0.108	-	-	-0.341	-
$\beta_{d,2,6}$	-	-	-	-	-	-	-	-	-
$\beta_{d,2,7}$	-	-	-	-	-	-	-	-	-
$\beta_{d,3,4}$	-	-	-	-	-	-	-	-	-
$\beta_{d,3,5}$	-	-	0.049	-	-	-	-	-	-
$\beta_{d,3,6}$	-	-	-	-	-	-	-	-	-
$\beta_{d,3,7}$	-	-	-	-	-	-	-	-	-
$\beta_{d,4,5}$	0.201	-0.215	0.120	0.102	-0.091	-	-	-0.085	-
$\beta_{d,4,6}$	-	-	-	-	-	-	-	-	-
$\beta_{d,4,7}$	-	-	-	-	-	-	-	-	-
$\beta_{d,5,6}$	0.117	-0.121	0.121	-	-	-	-	-	-
$\beta_{d,5,7}$	-	-	0.109	-	-	-	-	-	-
$\beta_{d,6,7}$	-	-	-	-	-	-	-	-	-

Table B.3: Regression coefficients, part 2

d	10	11	12	13	14	15	16	17
$\alpha_{d,0}$	-0.138	-0.191	-0.377	0.367	0.095	-0.350	-0.015	-0.145
$\alpha_{d,1}$	-	-0.022	0.050	-0.054	0.337	0.069	-0.295	-
$\alpha_{d,2}$	-	-	0.269	-0.257	-0.151	-	-	-
$\alpha_{d,3}$	-0.968	-0.866	-	-	-0.067	0.307	-	-0.954
$\alpha_{d,4}$	-	-	0.099	-0.094	-0.056	-	-	-
$\alpha_{d,5}$	-	0.051	-0.482	0.488	-0.251	-0.379	0.473	-
$\alpha_{d,6}$	-	-0.020	0.041	-0.043	0.280	0.055	-0.244	-
$\alpha_{d,7}$	-	-	-	-	0.088	-	-0.076	-
$\beta_{d,1,1}$	-	-	-	-	0.067	0.033	-0.069	-
$\beta_{d,2,2}$	-	-	0.096	-0.090	-0.054	-	-	-
$\beta_{d,3,3}$	0.512	0.720	-	-	-	0.100	-	0.514
$\beta_{d,4,4}$	-	-	-	-	-	-	-	-
$\beta_{d,5,5}$	-	-	0.814	-0.817	-0.554	0.427	-	-
$\beta_{d,6,6}$	-	-	-	-	-	-	-	-
$\beta_{d,7,7}$	-	-	-	-	-	-	-	-
$\beta_{d,1,2}$	-	-	-	-	-	-	-	-
$\beta_{d,1,3}$	-	-	-	-	-	0.092	-	-
$\beta_{d,1,4}$	-	-	-	-	-	-	-	-
$\beta_{d,1,5}$	-	-	-0.151	0.158	0.107	-0.135	-	-
$\beta_{d,1,6}$	-	-	-	-	-	-	-	-
$\beta_{d,1,7}$	-	-	-	-	-	-	-	-
$\beta_{d,2,3}$	-	-	-	-	-	-	-	-
$\beta_{d,2,4}$	-	-	0.070	-	-0.042	-	-	-
$\beta_{d,2,5}$	-	-	-0.426	0.406	0.241	-	-	-
$\beta_{d,2,6}$	-	-	-	-	-	-	-	-
$\beta_{d,2,7}$	-	-	-	-	-	-	-	-
$\beta_{d,3,4}$	-	-	-	-	-	-	-	-
$\beta_{d,3,5}$	-	-	-	-	0.049	-0.503	-	-
$\beta_{d,3,6}$	-	-	-	-	-	0.078	-	-
$\beta_{d,3,7}$	-	-	-	-	-	-	-	-
$\beta_{d,4,5}$	-	-	-0.215	0.202	0.120	-	-	-
$\beta_{d,4,6}$	-	-	-	-	-	-	-	-
$\beta_{d,4,7}$	-	-	-	-	-	-	-	-
$\beta_{d,5,6}$	-	-	-0.121	0.117	0.121	-0.097	-	-
$\beta_{d,5,7}$	-	-	-	-	0.109	-	-0.074	-
$\beta_{d,6,7}$	-	-	-	-	-	-	-	-

Appendix C

The exponents a_e

In equation (3.14), the exponents a_e, corresponding to the expression $\frac{\kappa-1}{\kappa}$ in equations (2.7) and (2.9), are used. In the following the expression for a_i used in equation (3.16) is derived.

The heat capacity ratio κ is defined as

$$\kappa = \frac{c_p}{c_v}, \tag{C.1}$$

where c_p is the specific heat capacity for constant pressure, and c_v is the specific heat capacity at constant volume, and for an ideal gas, $c_v = c_p - R$, where R is the gas constant. As the assumpion of ideal gas is necessary for modeling turbines and compressors with equations (2.7)–(2.10), we can use this. Then we have

$$\kappa = \frac{c_p}{c_v} = \frac{c_p}{c_p - R} \frac{\kappa - 1}{\kappa} = \frac{\frac{c_p}{c_p - R} - 1}{\frac{c_p}{c_p - R}} = \frac{\frac{c_p}{c_p - R} - \frac{c_p - R}{c_p - R}}{\frac{c_p}{c_p - R}} = \frac{\frac{R}{c_p - R}}{\frac{c_p}{c_p - R}} = \frac{R}{c_p}. \tag{C.2}$$

The average gas constant for a mixed stream is found as the mass average,

$$R = \frac{\sum_{k \in K} f_k \; R_k}{\sum_{k \in K} f_k}, \tag{C.3}$$

where the notation is as earlier, but with the stream index skipped. The average heat capacity is found as enthalpy change divided by temperature change, and again mass average:

$$c_p = \frac{\Delta h}{\Delta T} = \frac{\sum_{k\in K} f_k \frac{\Delta h_k}{\Delta T}}{\sum_{k\in K} f_k} = \frac{\sum_{k\in K} f_k\ \left(h_k(T_{IN}) - h_k(T_{OUT})\right)}{(T_{IN} - T_{OUT})\sum_{k\in K} f_k} = \frac{\sum_{k\in K} f_k \sum_{n=1}^{5} H_{k,n}\left({T_{IN}}^n - {T_{OUT}}^n\right)}{(T_{IN} - T_{OUT})\sum_{k\in K} f_k} \tag{C.4}$$

Recalling from equation (3.13) that $w = \sum_{k\in K} \eta\ f_k \sum_{n=1}^{5} H_{k,n}({T_{IN}}^n - {t_{OUT}}^n)$, we can substitute the numerator with w/η, and get

$$c_p = \frac{w}{\eta\ (T_{IN} - T_{OUT})\sum_{k\in K} f_k}. \tag{C.5}$$

Assembling, we get that

$$a = \frac{\kappa - 1}{\kappa} = \frac{R}{c_p} = \frac{\frac{\sum_{k\in K} f_k\ R_k}{\sum_{k\in K} f_k}}{\frac{w}{\eta\ (T_{IN}-T_{OUT})\sum_{k\in K} f_k}} = \frac{\eta\ (T_{IN} - T_{OUT})\sum_{k\in K} f_k\ R_k}{w}, \tag{C.6}$$

and rearranging to avoid any division, we get

$$a\ w = \eta\ (T_{IN} - T_{OUT}) \sum_{k\in K} f_k\ R_k, \tag{C.7}$$

which is the same as (3.16) if we add subscripts for stream and equipment.

Appendix D

Figures for alternative scenarios

This appendix include flowsheets, composite curves and grand composite curves for the alternative scenarios presented in sections 4.4 and 5.5.

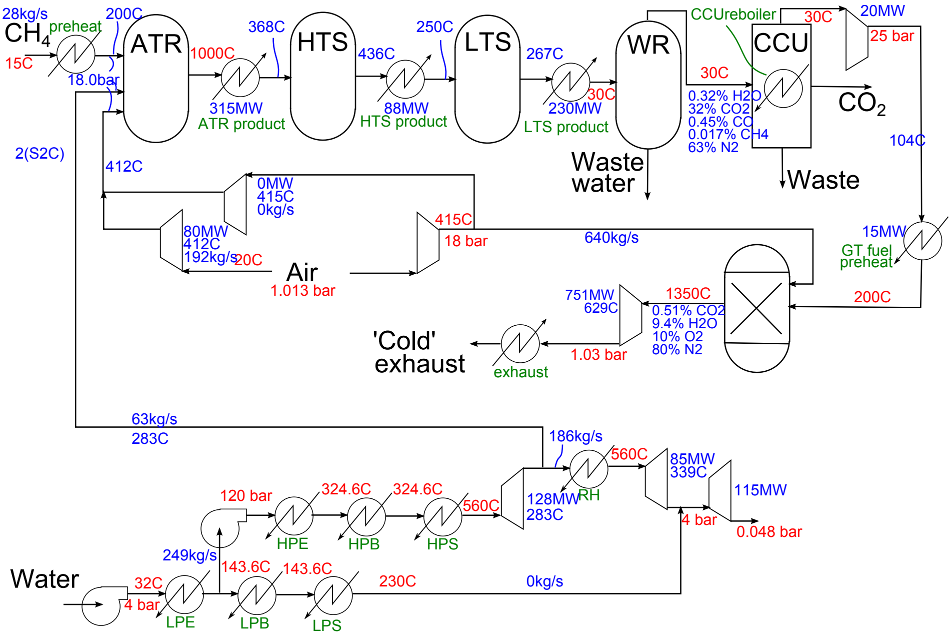


Figure D.1: Flowsheet for process with maximized power output

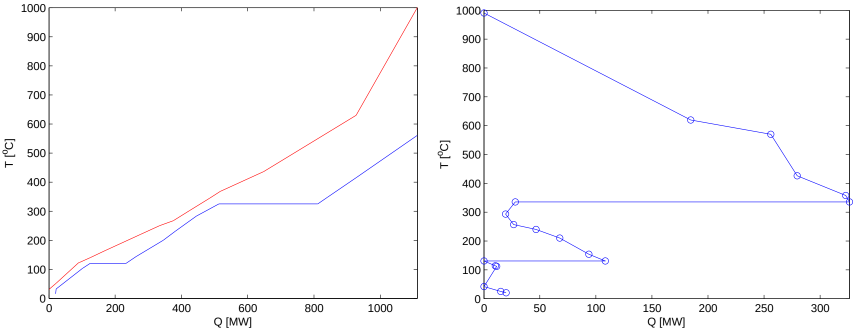


Figure D.2: Composite curves and grand composite curve for process with maximized power output

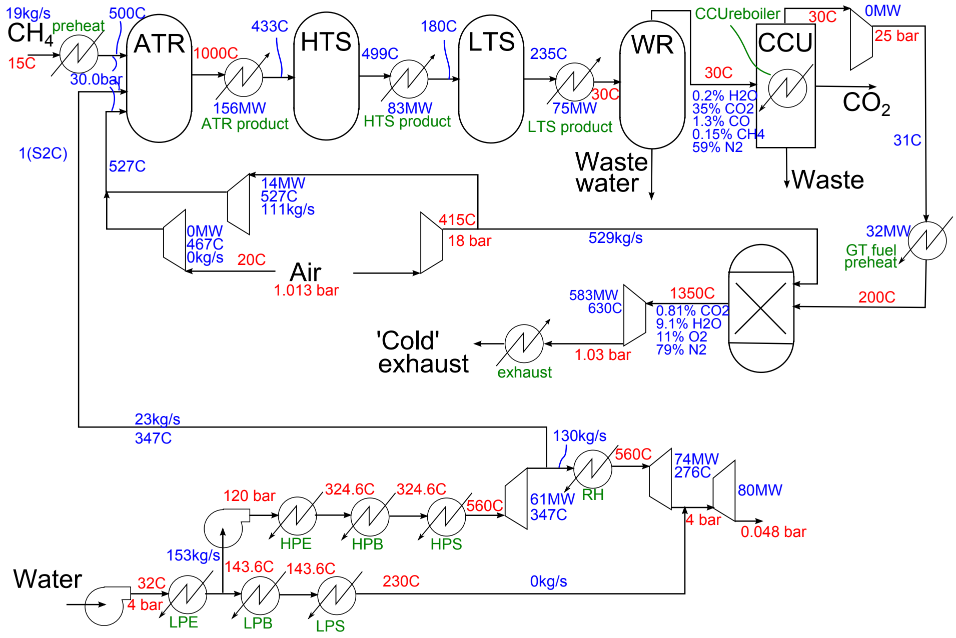


Figure D.3: Flowsheet for process with no extra ATR air compressor

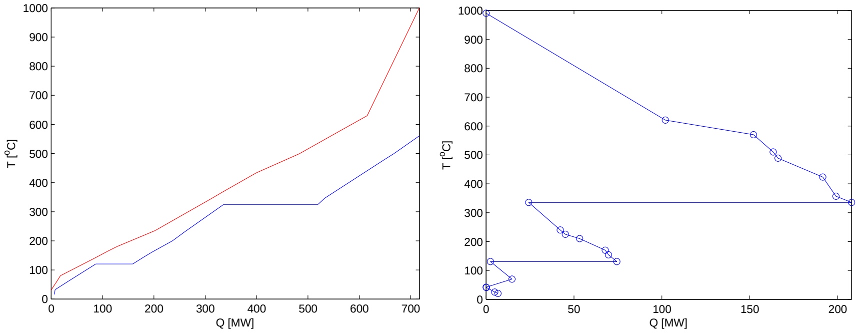


Figure D.4: Composite curves and grand composite curve for process with no extra ATR air compressor

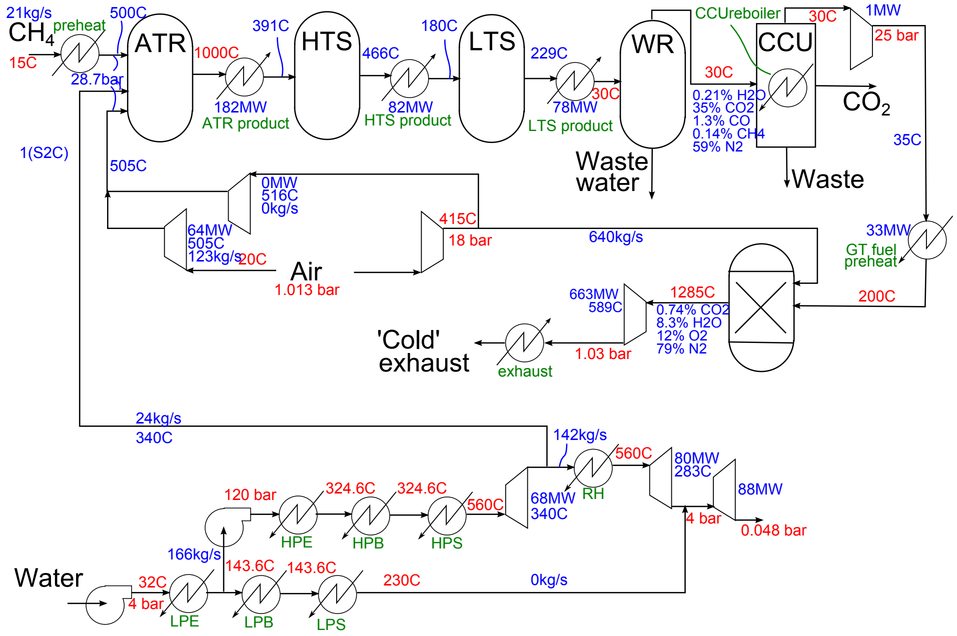


Figure D.5: Flowsheet for process with lower TIT

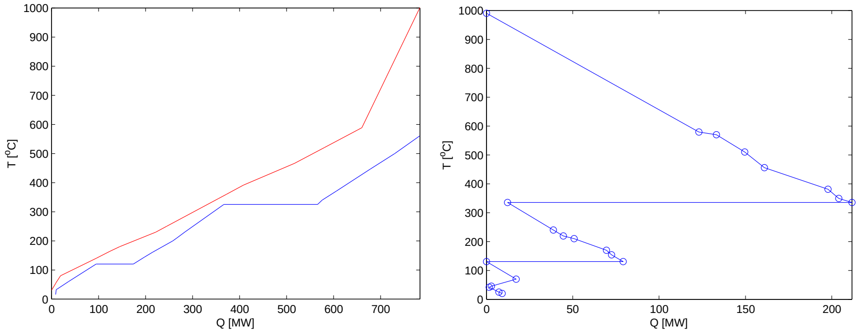


Figure D.6: Composite curves and grand composite curve for process with lower TIT

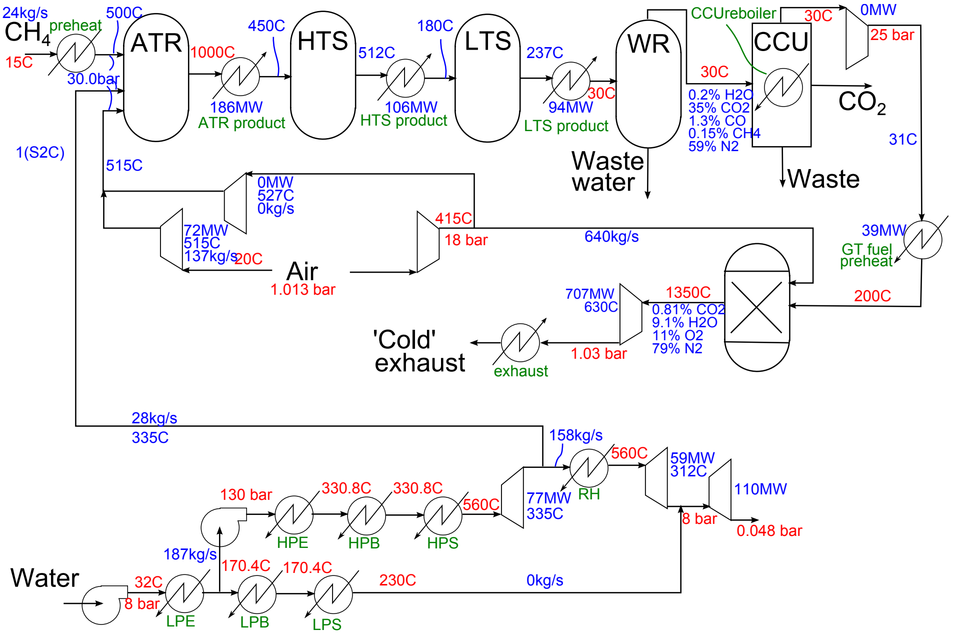


Figure D.7: Flowsheet for process with higher steam pressures

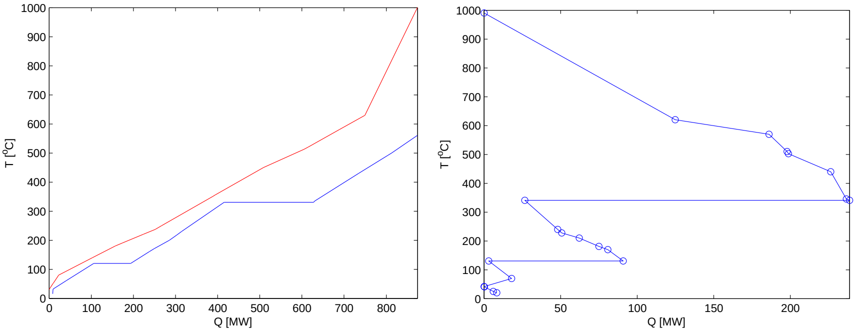


Figure D.8: Composite curves and grand composite curve for process with higher steam pressures

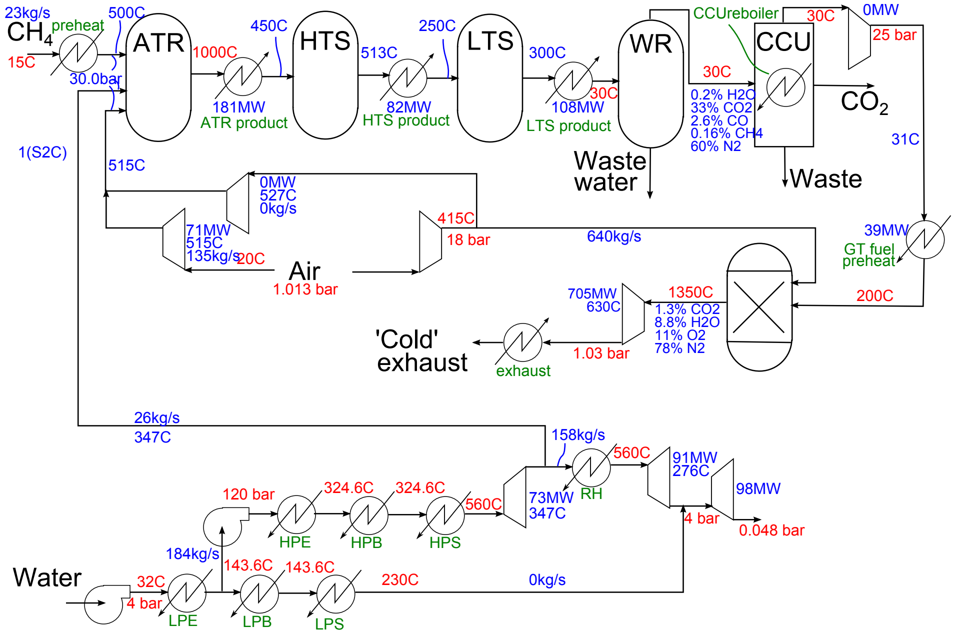


Figure D.9: Flowsheet for process with less CO$_2$ captured

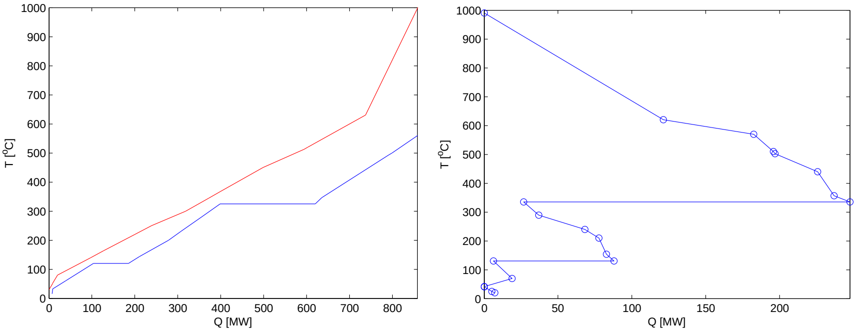


Figure D.10: Composite curves and grand composite curve for process with less CO$_2$ captured

Appendix E

Data

The model is based on the following specifications:

ATR	Temperature	1000 °C
	Pressure loss	1 bar
ATR product cooling	Pressure loss	1 bar
	Inlet temperature	1000 °C
HTS	Pressure loss	0.5 bar
HTS product cooling	Pressure loss	1 bar
LTS	Pressure loss	0.5 bar
LTS product cooling	Pressure loss	1 bar
	Outlet temperature	30 °C
WR	Temperature	30 °C
CCU	CO_2 capture rate	96.03 %
	Fuel outlet temperature	30 °C
Fuel compressor	Efficiency (isentropic)	86 %
	Outlet pressure	25 bar
GT fuel preheating	Outlet temperature	200 °C
Combustor	Temperature	1350 °C
	Outlet pressure	17.5 bar
Gas turbine	Efficiency (isentropic)	91 %
	Inlet temperature	1350 °C
	Inlet pressure	17.5 °C
	Outlet pressure	1.03 bar
Main air compressor	Efficiency (isentropic)	92 %
	Outlet pressure	18 bar
	Outlet temperature	415 °C

	Mass flow rate	640 kg/s
	Effect	288.76 MW
Second air compressor	Efficiency (isentropic)	88 %
	Inlet pressure	18 bar
	Inlet temperature	415 °C
Extra air compressor	Efficiency (isentropic)	88 %
	Inlet pressure	1.013 bar
	Inlet temperature	15 °C
LP steam pump	Outlet pressure	4 bar
	Outlet temperature	32 °C
	Specific work	0.403 kJ/kg
HP steam pump	Outlet pressure	120 bar
	Outlet temperature	143.6 °C
	Specific work	16.65 kJ/kg
LPE	Mass specific heating	470 kJ/kg
	Inlet temperature	32 °C
	Outlet temperature	143.6 °C
LPB	Mass specific heating	2134 kJ/kg
	Temperature	143.6 °C
LPS	Mass specific heating	186 kJ/kg
	Outlet temperature	230 °C
HPE	Mass specific heating	880 kJ/kg
	Inlet temperature	143.6 °C
	Outlet temperature	324.6 °C
HPB	Mass specific heating	1198 kJ/kg
	Temperature	324.6 °C
LPS	Mass specific heating	816 kJ/kg
	Outlet temperature	560 °C
RH	Outlet temperature	560 °C
HP steam turbine	Efficiency (isentropic)	94 %
IP steam turbine	Efficiency (isentropic)	92 %
LP steam turbine	Efficiency (isentropic)	88 %
	Outlet pressure	0.048 bar
	MORE TEMPERATURES	and (Δ)P
Methane feed	Temperature	15 °C
Cooling water	Inlet temperature	10 °C
	Outlet temperature	25 °C
	Pressure loss	1 bar
Cooling water pump	Work per MJ of cooling	2.1 kJ/MJ
CO_2 compression	Mass specific work	342 kJ/kg
	Outlet pressure	110 bar

CCU reboiler	Heating per kg of CO_2 captured	1500 kJ/kg
	Temperature	120 °C

Appendix F

Source code

In this appendix, the code used for the implemented model is presented. The first section shows the GAMS code for the optimization, which is the most important part. The MATLAB code used for handling data, calling HYSYS for process simulations, regression, calculations and solution output is included below. It is lengthy, but the code would be incomplete if some of these parts were left out. A large amount of code, including that used for the accuracy comparison with HYSYS, and many scripts for testing and analyzing results, is not presented here, as this is not central parts of the work and this appendix is already long enough.

F.1 The optimization model written in GAMS

F.1.1 The main file — ircc.gms

```
********************************************************************************
* ircc.gms - Integrated Reforming Combined Cycle
********************************************************************************
* OTHER FILES NEEDED:
* GAMS SCRIPTS:
*       combustion.gms, air.gms, turbo.gms, hrsgconnected.gms (using hr.gms)
* GDX DATA FILES:
*       masterset.gdx, regcoeffs.gdx, modelparameters.gdx,
*       (and indirectly: combustion.gdx, air.gdx, turbo.gdx, hrsgconnected.gdx)
********************************************************************************
```

```
*Reading sets and parameter values from gdx files
*Reading the full set...
$Gdxin masterset.gdx
Set      v_all                    indices for ALL variables;
$load v_all
Set      v_bounded(v_all)         the variables with bounds;
$load v_bounded
Parameters
         lower(v_all)             lower bounds
         upper(v_all)             upper bounds
         objcoeffs(v_all)         objective function coefficients
         objtype                  the type of objective (power or efficiency)
         maxtime                  maximum solution time;
$load lower, upper, objcoeffs, objtype, maxtime
$Gdxin
*Reading regression coefficients and related sets and parameters
$Gdxin regcoeffs.gdx
Set      v(v_all)                 indices for all variables in regression;
$load v
Sets
         iv(v)                    indices for the independent variables
         dv(v)                    indices for the dependent variables
         dv_all(v_all)
         b                        indices for all regression coefficients;
$load iv, dv, b
Parameters
         betas(b,dv)              regression coefficients
         high(v)                  highest value of variables in the data set
         low(v)                   lowest value of variables in the data set;
Sets
         lb(iv,b)                 indices for linear regression coefficients
         qb(iv,b)                 indices for quadratic regression coefficients
         bb(iv,iv,b)              indices for bilinear regression coefficients;
$load betas, lb, qb, bb, high, low
$Gdxin
*Reading objective coefficients ("costs"), upper and lower bounds, linear
*constraint coefficients and bounds and starting point, with related sets
$Gdxin modelparameters.gdx
Sets
         feedflow(v_all)         the flow rate of the methane feed
         lc                      indices for the linear constraints
         ec                      indices for linear equality constraints
         pr                      indices for product constraints;
$load lc, feedflow, ec, pr
alias(v_all, v_all2, v_all3);
Parameters
         linconstr(lc, v_all)    coefficient matrix for linear constraints
         linconstrbound(lc)      vector of bounds for the linear constraints
         eqconstr(ec, v_all)     coefficient matrix for linear equality constrs.
         eqconstrconst(ec)       vector of constants for the lin. eq. constrs.
         prodconstr(pr,v_all,v_all2, v_all3) coefficients for product constrs.
         startingpoint(v_all)    starting values
         feedflowdep(dv)         variables directly depending on feedflow;
$load linconstr, linconstrbound, eqconstr, eqconstrconst, prodconstr
$load startingpoint, feedflowdep
$Gdxin
*End of reading

*Declaring variables and equations
Variables
         x(v_all)        all variables
         z               objective;

Equations
         interpol(dv)    relations between independent and dependent variables
         linear(lc)      linear constraints
         eqlin(ec)       equality constraints
         product(pr, v_all, v_all2, v_all3)     products
         powerobjective  objective for maximizing power
         effobjective    objective for maximizing efficiency;
dv_all(v_all)=yes;       dv_all(iv)=no;
alias(iv, iv2);
* Setting the dependent variables with the regression equations
interpol(dv) ..  x(dv) =e= (.5*(high(dv)+low(dv))+(high(dv)-low(dv))*
         (betas("beta_0",dv) + sum(lb(iv,b), x(iv)*betas(b, dv)) +
         sum(qb(iv,b), x(iv)*x(iv)*betas(b, dv)) +
         sum(bb(iv,iv2,b), x(iv)*x(iv2)*betas(b,dv))))
         *(1 - feedflowdep(dv) + feedflowdep(dv) * sum(feedflow,x(feedflow)));

* Setting the linear constraints
linear(lc) ..    sum(iv$(linconstr(lc,iv)<>0),
         (x(iv)*(high(iv)-low(iv))+.5*(high(iv)+low(iv)))*linconstr(lc,iv)) +
         sum(dv_all$(linconstr(lc,dv_all)<>0), x(dv_all)*linconstr(lc, dv_all))
         =l= linconstrbound(lc);
eqlin(ec) ..    sum(iv$(eqconstr(ec,iv)<>0),
         (x(iv)*(high(iv)-low(iv))+.5*(high(iv)+low(iv)))* eqconstr(ec,iv)) +
         sum(dv_all$(eqconstr(ec,dv_all)<>0), x(dv_all)*eqconstr(ec, dv_all))=e=
         eqconstrconst(ec);
Parameter coeff(v_all), const(v_all);
coeff(v_all)=1;                         const(v_all)=0;
coeff(iv)=high(iv)-low(iv);             const(iv)=.5*(high(iv)+low(iv));
* Setting product equality constraints
product(pr,v_all,v_all2, v_all3)$(prodconstr(pr,v_all,v_all2, v_all3)<>0) ..
         coeff(v_all)*x(v_all)+const(v_all) =e=
         prodconstr(pr,v_all,v_all2, v_all3)*
         (coeff(v_all2)*x(v_all2)+const(v_all2))*
         (coeff(v_all3)*x(v_all3)+const(v_all3));

* Lower and upper bounds
x.lo(v_bounded)=lower(v_bounded);
x.up(v_bounded)=upper(v_bounded);
*Normalizing the bounds for the input variables
x.lo(iv) = (lower(iv) - .5*(high(iv)+low(iv))) / (high(iv)-low(iv));
x.up(iv) = (upper(iv) - .5*(high(iv)+low(iv))) / (high(iv)-low(iv));

* Starting values
x.l(feedflow)=lower(feedflow);
x.l(iv) = (startingpoint(iv)-.5*(high(iv)+low(iv)))/(high(iv)-low(iv));
x.l(dv) = (1 - feedflowdep(dv) + feedflowdep(dv) * x.l("feedflow"))
         *.5*(high(dv)+low(dv))+(high(dv)-low(dv))*(betas("beta_0",dv) +
         sum(lb(iv,b), x.l(iv)*betas(b, dv)) +
         sum(qb(iv,b), x.l(iv)*x.l(iv)*betas(b, dv)) +
         sum(bb(iv,iv2,b), x.l(iv)*x.l(iv2)*betas(b,dv)));

* Including scripts for "other modules"
$include combustion.gms
$include air.gms
$include turbo.gms
$include hrsgconnected.gms

* Setting the objective
*** Net power output
powerobjective$(objtype eq 1) ..z =e= sum(v_all, objcoeffs(v_all)*x(v_all))
```

```
         + Q_H*Q_Hcost + Q_C*Q_Ccost + sum(s, flow(s)*flowcost(s));
*** Net efficiency
effobjective$(objtype eq 2) .. z*x("feedflow") =e=
         sum(v_all, objcoeffs(v_all)*x(v_all)) +
         Q_H*Q_Hcost + Q_C*Q_Ccost + sum(s, flow(s)*flowcost(s));

* Defining the model (with all equations)
model ircc /all/;

* Treat fixed variables as constant
ircc.holdfixed=1;
* Max solution time
ircc.reslim=maxtime;
* Some options
ircc.optcr=1e-9;
option sys12 = 1;
option limrow = 50;
option limcol = 50;
option minlp=baron;

* Setting bounds for the ATR pressure (used for manual branching)
*x.lo("feedpressure")=-.307;
*x.up=.5;

*Solving the optimization model
Solve ircc using minlp minimizing z;

*Preparing the solution to be written to file
Parameter x_opt(v_all);
x_opt(v_all)=x.l(v_all);
x_opt(iv)=.5*(high(iv)+low(iv))+(high(iv)-low(iv))*x.l(iv);

*Preparing information about the solution
Parameters       z_opt
                 exectime
                 iterations
                 modelstatus;
z_opt=z.l;
exectime=-5.0;
exectime=ircc.Resusd;
iterations=-5.0;
```

```
iterations=ircc.Iterusd;
modelstatus=ircc.ModelStat;


*Writing output to file
Execute_unload "irccoptimum.gdx" x_opt, z_opt, exectime, iterations, modelstatus
*** Heat integration solution
Parameters Tin(streams), Tout(streams), mCp(streams);
Tin(s)=T_in.l(s);
Tout(s)=T_out.l(s);
mCp(s)=flow.l(s);
Execute_unload "hrsgopt.gdx" Tin, Tout, mCp, HRAT, Q_H, Q_C, z.l;
```

F.1.2 Combustion modeling — combustion.gms

```
********************************************************************************
* combustion.gms
********************************************************************************

*Reading sets and parameter values from gdx file
$Gdxin combustion.gdx
Set                v_comb(v_all)           Set of variables related to combustion;
$load v_comb
Sets
                   exhcomp(v_comb)         The componets in the exhaust
                   comb_airflow(v_comb)    Air flowrate to the combustor;
$load exhcomp, comb_airflow
Parameter          combreaction(v,exhcomp) Reaction stoichiometry
                   air(exhcomp)            Air composition
                   q_comb_air              Heating of air to combustion temp.(TIT)
                   LHV(v)                  Lower heating values
                   CpDT(exhcomp)           Heating of combustion products to TIT
                   tempscale               Scaling factor for some temperatures;
$load combreaction, air, q_comb_air, LHV, CpDT
$load tempscale
$Gdxin
*End of reading

Equations
         combustion(exhcomp, comb_airflow) combustion reactions
         heatbalance(comb_airflow)         heat balance to determine air excess;
heatbalance(comb_airflow)    ..
         sum(exhcomp, CpDT(exhcomp)*(x(exhcomp)-air(exhcomp)*x(comb_airflow)) +
         air(exhcomp)*x(comb_airflow)*q_comb_air) =e= sum(v, LHV(v) * x(v));
combustion(exhcomp, comb_airflow) .. x(exhcomp) =e=
         sum(v$(combreaction(v,exhcomp)<>0), combreaction(v,exhcomp)*x(v)) +
         air(exhcomp)*x(comb_airflow);
```

F.1.3 Air distribution modeling — air.gms

```
********************************************************************************
* air.gms
********************************************************************************

*Reading sets and parameter values from gdx file
$Gdxin air.gdx
Set        v_air(v_all)          Set of variables related to air supply;
$load v_air
Sets       p_feed(iv)            Feed pressure
           T_airfeed(iv)         Air feed temperature
           flow_ATRair(v_air)    Air flowrates to the ATR
           p_ATRair(v_air)       Pressure of the air flows to the ATR
           T_ATRair(v_air)       Temperature of the air flows to the ATR;
$load p_feed, T_airfeed, flow_ATRair, T_ATRair, p_ATRair
Sets       flow_ATRair_2nd(flow_ATRair)           air flows into the ATR
           ATRair(p_ATRair, T_ATRair, flow_ATRair) (connecting variables);
Parameters ATR_air(v)                     stoichiometry for reforming reactions
           p_and_T(p_ATRair, T_ATRair)    (used for connecting cariables)
           airflow                        flow rate through main compressor;
$load ATRair, flow_ATRair_2nd, ATR_air, p_and_T,
$load airflow

$Gdxin
*End of reading

Equations
         airbalance                      determining the amount of ATR air needed
         ATRpressure(p_ATRair, p_feed)  ATR pressure=air compression pressure
         airtemperaturesum(T_airfeed)   mixing temperature
         AF                              balance for air flow from main compr.;
```

```
airbalance .. sum(flow_ATRair, x(flow_ATRair)) =e=
         sum(v$(ATR_air(v) <> 0), x(v)*ATR_air(v));
ATRpressure(p_ATRair, p_feed).. x(p_ATRair) =e=
         .5*(high(p_feed)+low(p_feed))+(high(p_feed)-low(p_feed))*x(p_feed);
airtemperaturesum(T_airfeed) .. sum(ATRair(p_ATRair, T_ATRair, flow_ATRair),
         tempscale*x(T_ATRair) * x(flow_ATRair)) =e= (273.15+.5*(high(T_airfeed)
         +low(T_airfeed))+(high(T_airfeed)-low(T_airfeed))*x(T_airfeed)) *
         sum(flow_ATRair, x(flow_ATRair));
AF.. sum(comb_airflow,x(comb_airflow))+sum(flow_ATRair_2nd,x(flow_ATRair_2nd))
         =e= airflow;
```

F.1.4 Turbo-machinery modeling — turbo.gms

```
********************************************************************************
* turbo.gms
********************************************************************************

Sets     trb                               set of turbines
         trb1(trb)                         turbomachinery modeled physically
         trb2(trb)                         turbomach. modeled from simulations
         trbflows(v_all)                   turbomach. flow variables
         p_out(v_all)                      outlet pressure variables
         T_trb_out(v_all)                  outlet temperature variables
         Ti_trb_out(v_all)                 ideal outlet temperature variables
         W(v_all)                          power or work variables
         kappafrac(v_all)                  (kappa-1) divided by kappa exponents
         exp                               set for temperature exponents;
Parameter
         eff(trb)                          turbine efficiencies
         turboflows(trb,trbflows)          which flows for which turbomachine
         Cpcoeffs(trbflows,exp)            specific heat equation coefficients
         T_trb_in(trb)                     inlet temperatures
         p_in(trb)                         inlet pressures
         kappafracsign(trb)                (used for changing p_in with p_out)
         R(trbflows)                       gas constants
         exponents(exp)                    exponents for specific heat eqn.
         w0(trb2)                          work for p_ATR=0 (constant term)
         dwdp(trb2)                        coefficient for pressure in work eqn.
         T0(trb2)                          temperature for p_ATR=0 (constant term)
         dTdp(trb2)                        coefficient for pressure in temp. eqn.;
$GDXIN turbo.gdx
```

```
$load trb,trb1,trb2,trbflows, p_out, T_trb_out,Ti_trb_out, W, kappafrac, exp
$load eff, turboflows, Cpcoeffs, T_trb_in, p_in, kappafracsign, R, exponents
$load w0, dwdp, T0, dTdp
$GDXIN

Equations
         turbine1(trb1, Ti_trb_out, W)                         power for trb1
         turbine2(trb2,W,p_feed)                               power for trb2
         idealturbinetemp(trb, Ti_trb_out, p_out, kappafrac) ideal outlet temp.s
         turbinetemp(trb1, T_trb_out, Ti_trb_out)              trb1 outlet temp.s
         turbinetemp2(trb2, T_trb_out, p_feed)                 trb2 outlet temp.s
         comprexp(trb1,Ti_trb_out,kappafrac,W)     exponents in idealturbinetemp
;
*Variable deltaH(trb);

turbine1(trb1, Ti_trb_out, W)$(ord(Ti_trb_out) eq ord(trb1) and
         ord(W) eq ord(trb1)) .. x(W) =e= eff(trb1)*
         sum(trbflows$(turboflows(trb1,trbflows) eq 1), x(trbflows)*
         sum(exp, Cpcoeffs(trbflows,exp) *
         (T_trb_in(trb1)**exponents(exp)-x(Ti_trb_out)**exponents(exp)) /
         exponents(exp)));

turbine2(trb2,W,p_feed)$(ord(W) eq ord(trb2)+card(trb1)) .. x(W) =e=
         sum(trbflows$(turboflows(trb2,trbflows) eq 1), x(trbflows)*
         (w0(trb2) + dwdp(trb2)*(const(p_feed) + coeff(p_feed)*x(p_feed))));

idealturbinetemp(trb1,Ti_trb_out,p_out,kappafrac)$(ord(Ti_trb_out) eq ord(trb1)
         and ord(p_out) eq ord(trb1) and ord(kappafrac) eq ord(trb1)) ..
         x(Ti_trb_out) =e=
         T_trb_in(trb1) * (x(p_out)/p_in(trb1))**x(kappafrac);

turbinetemp(trb1,T_trb_out,Ti_trb_out)$(ord(T_trb_out) eq ord(trb1)
         and ord(Ti_trb_out) eq ord(trb1)) .. x(T_trb_out) =e=
         abs(eff(trb1))*x(Ti_trb_out)+(1-abs(eff(trb1)))*T_trb_in(trb1);
turbinetemp2(trb2, T_trb_out, p_feed)$(ord(T_trb_out) eq ord(trb2)+card(trb1))..
         x(T_trb_out) =e=
         T0(trb2) + dTdp(Trb2)*(const(p_feed)+coeff(p_feed)*x(p_feed));

comprexp(trb1,Ti_trb_out,kappafrac,W)$(ord(Ti_trb_out) eq ord(trb1) and
         ord(kappafrac) eq ord(trb1) and ord(W) eq ord (trb1)) ..
         eff(trb1) * sum(trbflows$(turboflows(trb1, trbflows) eq 1),
         x(trbflows)*R(trbflows)) * tempscale*(T_trb_in(trb1)-x(Ti_trb_out)) =e=
         x(kappafrac) * x(W);
```

F.1.5 Connecting to the heat integration model — hrsgconnected.gms

```
Set     streams                  set of all heat integration streams;
Sets    noniso(streams)          set of non-isothermal streams
        iso(streams)             set of isothermal streams
        cold(noniso)             set of cold non-isothermal streams
        hot(noniso)              set of hot non-isothermal streams
        isocold(iso)             set of cold isothermal streams
        isohot(iso)              set of hot isothermal streams
        allcold(streams)         set of all cold streams
        allhot(streams)          set of all hot streams
        pc(streams)              set of pinch candidate streams;

Parameters
        T_in_up(streams)         upper bound for inlet temperatures
        T_in_lo(streams)         lower bound for inlet temperatures
        T_out_up(streams)        upper bound for outlet temperatures
        T_out_lo(streams)        lower bound for outlet temperatures
        flow_lo(streams)         upper bound for flows (heat capacity or heat)
        flow_up(streams)         lower bound for flows (heat capacity or heat)
        HRAT                     heat recovery approach temperature
        Q_Hcost                  cost for hot utilities
        Q_Ccost                  cost for cold utilities
        flowcost(streams)        costs for streams
        T_inconn(streams,v_all) coefficients for connecting T_in variables
        T_outconn(streams,v_all) coefficients for connecting T_out variables
        flowconn(streams,v_all) coefficients for connecting flow variables
        flowconn2(streams,streams) coefficients for internally connecting flows
        T_inconnconst(streams)  (constant terms for T_in connections)
        T_outconnconst(streams) (constant terms for T_out connections);

* Reading data from GDX file
$GDXIN 'hrsgconnected.gdx'
$load streams, noniso, iso, cold, hot, isocold, isohot, allcold, allhot, pc
$load T_in_up, T_in_lo, T_out_up, T_out_lo
$load flow_up, flow_lo, HRAT, Q_Hcost, Q_Ccost, flowcost
$load T_inconn, T_outconn, flowconn, flowconn2, T_inconnconst, T_outconnconst
```

```
$GDXIN

* Defining subsets without the dummy stream
Sets     s(streams)
         h(hot)
         c(cold)
         ni(noniso)
         i(iso)
         ih(isohot)
         ic(isocold)
         p(pc)
         ah(allhot)
         ac(allcold);
s(streams)=yes;  s("dummy")=no;
h(hot)=yes;      h("dummy")=no;
c(cold)=yes;     c("dummy")=no;
ni(noniso)=yes;  ni("dummy")=no;
i(iso)=yes;      i("dummy")=no;
ih(isohot)=yes;  ih("dummy")=no;
ic(isocold)=yes; ic("dummy")=no;
p(pc)=yes;       p("dummy")=no;
ah(allhot)=yes;  ah("dummy")=no;
ac(allcold)=yes; ac("dummy")=no;

* Defining variables
Positive variables
         T_in(streams)   inlet (supply) temperatures
         T_out(streams)  outlet (target) temperatures
         flow            heat capacity or heat flows
         Q_H             hot utility usage
         Q_C             cold utility usage;

Equations
         T_inconnection(streams)    connecting T_in with other temperatures
         T_outconnection(streams)   connecting T_out with other temperatures
         flowconnection(streams)    connecting heat int. flows with other flows
         flowconnection2(streams)   connecting heat integration flows;

T_inconnection(s)$(sum(v_all, T_inconn(s,v_all)**2)>0 or T_inconnconst(s)<>0)..
         T_in(s) =e= sum(v_all$(T_inconn(s,v_all) <> 0),
         T_inconn(s,v_all)*x(v_all)) + T_inconnconst(s);
```

```
T_outconnection(s)$(sum(v_all, T_outconn(s,v_all)**2)>0 or T_outconnconst(s)<>0)
         .. T_out(s) =e= sum(v_all$(T_outconn(s,v_all) <> 0),
         T_outconn(s, v_all)*x(v_all)) + T_outconnconst(s);
flowconnection(s)$(sum(v_all, flowconn(s,v_all)**2)>0) ..
         flow(s) =e= sum(v_all$(flowconn(s,v_all) <> 0),
         flowconn(s, v_all)*x(v_all));
flowconnection2(s)$(sum(streams, flowconn2(s,streams)**2)>0) ..
         flow(s) =e= sum(streams$(flowconn2(s,streams) <> 0),
         flowconn2(s,streams)*flow(streams));

T_in.up(s)=T_in_up(s);
T_out.lo(s)=T_out_lo(s);
flow.up(s)=flow_up(s);

T_in.lo(s)=T_in_lo(s);
T_out.up(s)=T_out_up(s);
flow.lo(s)=flow_lo(s);

* Including the heat integration model
$include hr.gms
```

F.1.6 The heat integration model — hr.gms

```
Parameters
        M1(streams,pc)          big M constant
        M2(streams,pc)          big M constant
        Tadj(streams)           constant for adjusting for HRAT;

Variables
        T_m(streams,pc)         (see t_M in Model chapter)
        Qp(iso,pc)              (see f_QP in Model chapter);

Binary variables
        y_hr(streams,pc);

Equations
        Q_H_eqn(pc)     Heating above pinch for every pinch candidate
        Q_balance       Total heat balance
        T_m1h(hot,pc)   these next equations are used to set T_M such that
        T_m2h(hot,pc)           Q_H_eqn accounts for only the heating and
        T_m3h(hot,pc)           cooling above pinch...
```

```
         T_m1c(cold,pc)
         T_m2c(cold,pc)
         T_m3c(cold,pc)
         Tisoh(isohot,pc)  likewise for isothermal streams...
         Tisoc(isocold,pc)
         Qpeqnh(isohot,pc)
         Qpeqn2h(isohot,pc)
         Qpeqnc(isocold,pc);

*Adjust cold streams with HRAT
Tadj(allcold)=HRAT;
Tadj(allhot)=0;

*Setting the big M constants
M1(streams,p)=1000;
M2(streams,p)=1000;
M1(ah,p) = T_in_up(p) + Tadj(p) - T_in_lo(ah) - Tadj(ah);
M2(h,p) = T_in_up(h) + Tadj(h) - T_in_lo(p) - Tadj(p);
M1(ac,p) = -T_in_lo(p) - Tadj(p) + T_in_up(ac) + Tadj(ac);
M2(c,p) = -T_in_lo(c) - Tadj(c) + T_in_up(p) + Tadj(p);
M2(i,p) = flow_up(i);
M1(s,p)$(M1(s,p)<0)=0;

Q_H_eqn(p) .. Q_H =g= sum(c, flow(c)*(T_out(c)-T_m(c,p)))
         - sum(h, flow(h)*(T_in(h)-T_m(h,p)))
         + sum(ic, Qp(ic,p)) - sum(ih, Qp(ih,p))
         ;
Q_balance .. Q_C - Q_H =e=
         sum(ni,(T_in(ni)-T_out(ni))*flow(ni))
         + sum(ih, flow(ih)) - sum(ic, flow(ic))
         ;

T_m1h(h,p) .. T_m(h,p) =g= T_out(h);
T_m2h(h,p) .. T_m(h,p) =g= T_in(p) + Tadj(p) - Tadj(h) - y_hr(h,p)*M1(h,p);
T_m3h(h,p) .. T_m(h,p) =g= T_in(h) - (1 - y_hr(h,p)) * M2(h,p);

T_m1c(c,p) .. T_m(c,p) =l= T_out(c);
T_m2c(c,p) .. T_m(c,p) =l= T_in(p) + Tadj(p) - Tadj(c) + (1-y_hr(c,p))*M1(c,p);
T_m3c(c,p) .. T_m(c,p) =l= T_in(c) + y_hr(c,p) * M2(c,p);

*sametempchange(s) .. T_in(s) - T_out(s) =e= T_in_up(s) - T_out_up(s);
```

```
Tisoh(ih,p) .. T_in(ih) =g= T_in(p) + Tadj(p) - Tadj(ih) - y_hr(ih,p)*M1(ih,p);
Tisoc(ic,p) .. T_in(ic) =l= T_in(p) + Tadj(p)-Tadj(ic)+(1-y_hr(ic,p))*M1(ic,p);
Qpeqnh(ih,p) .. Qp(ih,p) =l= (1-y_hr(ih,p)) * M2(ih,p);
Qpeqn2h(ih,p) .. Qp(ih,p) =l= flow(ih);
Qpeqnc(ic,p) .. Qp(ic,p) =g= flow(ic) - y_hr(ic,p)*M2(ic,p);


Parameter ord2(streams)    (help parameter for the next equations);
ord2(streams)=ord(streams);

* Applying logic to fix binary variables
y_hr.fx(ah,p)$(ord2(ah) eq ord2(p))=1;
y_hr.fx(ac,p)$(ord2(ac) eq ord2(p))=0;
y_hr.fx(s,p)$(T_in_up(s)<T_in_lo(p)) = 1;
y_hr.fx(s,p)$(T_in_lo(s)>T_in_up(p)) = 0;

* Setting bounds for the 'constructed variables'
Qp.lo(i,p)=0;
Qp.up(i,p)=flow_up(i);
T_m.lo(h,p)=T_out_lo(h);
T_m.up(h,p)=max(T_in_up(h),T_in_up(p));
T_m.lo(c,p)=min(T_in_lo(c),T_in_lo(p));
T_m.up(c,p)=T_out_up(c);
```

F.2 Data handling and regression code in MATLAB

F.2.1 Main file — ircc.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% OPTIMIZATION OF IRCC
%
% Functions/scripts used:
%   "HOME MADE":
%   – regcoeffs.m (using polyreg2.m)
%   – setmodelparameters.m
%   – solveircc.m
%   – combustion.m
%   – air.m
%   — turbo.m
%   — findpos.m
%   GDXMRW package:
%   — writegdx.m (using wgdx.m)
%   — readgdx.m (using rgdx.m)
%   — sp2full.m (and full2sp.m?)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear all

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Control parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tighterbounds=0;    % 0: no tightening, 1: tighten bounds, 2: tighten more
tempscale=800;      % Scaling of temperatures in turbines and compressors
flowscale=3600e3;   % Scaling of all mass and energy flows
rungams=false;      % Run GAMS from MATLAB or not
maxtime=30;         % Max. solution time in seconds
runsimulation=false;% Do the HYSYS simulations or use saved simulation file
runregression=false;% Do regression or use saved regression coefficients
if runsimulation
    runregression=true;
end

objtype=2;          % 1: Maximize power, 2: Maximize efficiency
aircompressorfix=0; % 0: Try both, 1: Only second, 2: Only extra
lowerTIT=1285;         % 0: TIT = 1350 C, else TIT = lowerTIT [C]
otherLP=0;          % 0: LP = 4 bar, 8: LP = 8 bar
otherHP=0;          % 0: HP = 120 bar, 130: HP = 130 bar
relax=1;            % 0: No relaxation, 1: Include fewer pinch candidates

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setting some basic parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
irccdata

if objtype==2
    objscale=1/CH4_in/LHV_CH4*100; % Scaling of the objective function
else
    objscale=1;
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calling simulation to do run LHS sampling and HYSYS process simulations
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if runsimulation
    simulation
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calling regcoeffs to fit data to 2nd order polynomials and provide GAMS
% with the regression coefficients
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
regcoeffs

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Starting point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
startingpoint=[
    feedtemp        500
    LTStemp         250
    feedpressure    30
    HTStemp         300
    S2C             1
    airtemp         500
    steamtemp       345
    feedflow        8.7482];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calling setmodelparameters to provide GAMS with some linear constraints,
% linear equality constraints, product constraints and a starting point
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
setmodelparameters

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setting parameters related to combustion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
combustion

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setting parameters related to air supply
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
air

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setting parameters related to turbines and compressors
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
turbo

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setting the objective function coefficients (the objective funtction is
% linear in the form cost*variable)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
objective=[...
%    feedflow            8e7/flowscale; ...
    CCU_CO2             w_CO2compr*split.CO2(2); ...
    fuelcompr           1; ...
    W_GT                -1; ...
    W_2ndcompr          1; ...
    W_extcompr          1; ...
    W_HPturb            -(1-STcondensercooling); ...
    W_IPturb            -(1-STcondensercooling); ...
    comb_airflow        w_air; ...
    flow_ATRair_2nd     w_air];
objective(:,2)=objscale*objective(:,2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setting lower and upper bounds for the variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bounds

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Including sets and parameters related to heat integration
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hrsg

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Writing objective, bounds and the set of all parameters to GDX file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
writegdx('masterset.gdx', ...
    'set',       'v_all',       v_all, ...
    'set',       'v_bounded',   v_all, ...
    'parameter', 'lower',       [v_all lowerbound(v_all')'], ...
    'parameter', 'upper',       [v_all upperbound(v_all')'], ...
    'parameter', 'objcoeffs',   objective, ...
    'parameter', 'objtype',     objtype, ...
    'parameter', 'maxtime',     maxtime, ...
    uels);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Solving the model with GAMS/BARON
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
gamsdir='C:\Programfiler\GAMS23.6\';     %The directory where GAMS is installed
filepath='';
gamsfile=['"' filepath 'ircc.gms"']; %The gams file for the optimization
gamsoption=' MINLP BARON';             %Using BARON as MINLP solver
if rungams
    dos([gamsdir 'GAMSKEEP ' gamsfile gamsoption]);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reading the solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
getsolution

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Calculating other variables (postprocessing) and displaying the solution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
irccoutput
```

F.2.2 Input of basic data — irccdata.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% irccdata.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Including the gdxmrw package
addpath 'C:\Programfiler\GAMS23.6';
% Including the hysyslib toolbox
addpath 'C:\Users\Erik\Documents\MATLAB\hysyslib';

% The components present in the process
components={'CO' 'CO2' 'H2O' 'H2' 'CH4' 'O2' 'N2'};
% Molar mass of the components
M=struct('CO', 28.01, 'CO2', 44.01, 'H2O', 18.0153, 'H2', 2.0159,...
         'CH4', 16.042, 'O2', 31.9988, 'N2', 28.0134);
% Air composition
airO2=.21;              airN2=.79;
airO2mass=airO2*M.O2/(airO2*M.O2+airN2*M.N2);
airN2mass=airN2*M.N2/(airO2*M.O2+airN2*M.N2);
% Lower heating values
LHV_CO=10.9*1000;   % [kJ/kg]
LHV_H2=120.1*1000;  % [kJ/kg]
LHV_CH4=50.1*1000;  % [kJ/kg]
% CCU Splitting factors [H2 rich Syngas, CO2 to compression, Waste stream]
%   (Copy/paste from HYSYS)
split.CH4=[0.996000000000000 8.50000000000000e-004  3.14999999999999e-003];
split.H2O=[7.99000000000000e-002 0.916690000000000  3.41000000000002e-003];
split.CO= [0.997200000000000 4.40000000000000e-004  2.36000000000003e-003];
split.H2= [0.997100000000000 4.60000000000000e-004  2.44000000000000e-003];
split.CO2=[3.15000000000000e-002 0.960330000000000  8.17000000000001e-003];
split.N2= [0.998100000000000 2.20000000000000e-004  1.68000000000001e-003];
split.O2= [1.00000000000000  0.000000000000000      0.000000000000000];
% Flowrate of methane into the reformer
CH4_in=10000/flowscale;

% Some temperatures
T_refprod = 1000;       % Reformer product temperature [C]
T_WRfeed = 30;          % Water removal feed temperature [C]
CCUoutlettemp = 30;     % Temperature at the CCU outlet [C]
C2K = 273.15;           % Difference between Celsius and Kelvin scales
```

```
% Minimum capture ratio
mincapture=.9;

iv  = 7;                % Number of independent variables
dv  =17;                % Number of dependent variables
v   =iv+dv;
uels=cell(v,1);

% 'Independent variables'
feedtemp    =  1;       uels{feedtemp}    = 'feedtemp';
LTStemp     =  2;       uels{LTStemp}     = 'LTStemp';
feedpressure=  3;       uels{feedpressure}= 'feedpressure';
HTStemp     =  4;       uels{HTStemp}     = 'HTStemp';
S2C         =  5;       uels{S2C}         = 'S2C';
airtemp     =  6;       uels{airtemp}     = 'airtemp';
steamtemp   =  7;       uels{steamtemp}   = 'steamtemp';

% 'Dependent variables'
CS_CO2      =  1+iv;    uels{CS_CO2}      = 'CS_CO2';
CS_CO       =  2+iv;    uels{CS_CO}       = 'CS_CO';
CS_H2       =  3+iv;    uels{CS_H2}       = 'CS_H2';
HTStemp2    =  4+iv;    uels{HTStemp2}    = 'HTStemp2';
LTStemp2    =  5+iv;    uels{LTStemp2}    = 'LTStemp2';
mCp_GTfuel  =  6+iv;    uels{mCp_GTfuel}  = 'mCp_GTfuel';
mCp_HTSprod =  7+iv;    uels{mCp_HTSprod} = 'mCp_HTSprod';
mCp_LTSprod =  8+iv;    uels{mCp_LTSprod} = 'mCp_LTSprod';
mCp_refprod =  9+iv;    uels{mCp_refprod} = 'mCp_refprod';
GTfueltemp  = 10+iv;    uels{GTfueltemp}  = 'GTfueltemp';
CCU_H2O     = 11+iv;    uels{CCU_H2O}     = 'CCU_H2O';
CCU_CO      = 12+iv;    uels{CCU_CO}      = 'CCU_CO';
CCU_CO2     = 13+iv;    uels{CCU_CO2}     = 'CCU_CO2';
CCU_H2      = 14+iv;    uels{CCU_H2}      = 'CCU_H2';
CCU_CH4     = 15+iv;    uels{CCU_CH4}     = 'CCU_CH4';
CCU_N2      = 16+iv;    uels{CCU_N2}      = 'CCU_N2';
fuelcompr   = 17+iv;    uels{fuelcompr}   = 'fuelcompr';
regvarflows=[CS_CO2 CS_CO CS_H2 mCp_GTfuel mCp_HTSprod ...
    mCp_LTSprod mCp_refprod CCU_H2O CCU_CO CCU_CO2 CCU_H2 CCU_CH4 ...
    CCU_N2 fuelcompr];
feedflowdep=[CS_CO2 CS_CO CS_H2 mCp_GTfuel mCp_HTSprod mCp_LTSprod ...
    mCp_refprod CCU_H2O CCU_CO CCU_CO2 CCU_H2 CCU_CH4 CCU_N2 fuelcompr];
CCU_set=[CCU_H2O CCU_CO CCU_CO2 CCU_H2 CCU_CH4 CCU_N2];
v_set=(1:v)';
iv_set=(1:iv)';
dv_set=(1:dv)'+iv;

% Other variables (more are defined in combustion.m and air.m)
feedflow    = 1+iv+dv;  uels{feedflow}  = 'feedflow';
```

```
steamfeed   = 2+iv+dv;  uels{steamfeed} = 'steamfeed';
V=steamfeed;

% Combustion
exh_H2O     = 1+V;      uels{exh_H2O}   = 'exh_H2O';
exh_CO2     = 2+V;      uels{exh_CO2}   = 'exh_CO2';
exh_O2      = 3+V;      uels{exh_O2}    = 'exh_O2';
exh_N2      = 4+V;      uels{exh_N2}    = 'exh_N2';
exhaust_set=(exh_H2O:exh_N2)';
V=exh_N2;

% Turbo
T_GT        = 1+V;      uels{T_GT}       = 'T_GT';
T_ATRair_2nd= 2+V;      uels{T_ATRair_2nd}='T_ATRair_2nd';
T_ATRair_ext= 3+V;      uels{T_ATRair_ext}='T_ATRair_ext';
T_HPturb    = 4+V;      uels{T_HPturb}  = 'T_HPturb';
T_IPturb    = 5+V;      uels{T_IPturb}  = 'T_IPturb';
T_ATRair_set= [T_ATRair_2nd; T_ATRair_ext];
outlettemp_set=(T_GT:T_IPturb)';
V=T_IPturb;

W_GT       = 1+V;       uels{W_GT}      = 'W_GT';
W_2ndcompr  = 2+V;      uels{W_2ndcompr}= 'W_2ndcompr';
W_extcompr  = 3+V;      uels{W_extcompr}= 'W_extcompr';
W_HPturb    = 4+V;      uels{W_HPturb}  = 'W_HPturb';
W_IPturb    = 5+V;      uels{W_IPturb}  = 'W_IPturb';
W_set=(W_GT:W_IPturb)';
V=W_IPturb;

p_GT        = 1+V;      uels{p_GT}      = 'p_GT';
p_ATRair_2nd= 2+V;      uels{p_ATRair_2nd}='p_ATRair_2nd';
p_ATRair_ext= 3+V;      uels{p_ATRair_ext}='p_ATRair_ext';
p_HPturb    = 4+V;      uels{p_HPturb}  = 'p_HPturb';
p_IPturb    = 5+V;      uels{p_IPturb}  = 'p_IPturb';
p_ATRair_set= [p_ATRair_2nd; p_ATRair_ext];
p_out_set = (p_GT:p_IPturb)';
V=p_IPturb;

kappafrac_GT=1+V;    uels{kappafrac_GT}='kappafrac_GT';
kappafrac_2ndcompr=2+V;uels{kappafrac_2ndcompr}='kappafrac_2ndcompr';
kappafrac_extcompr=3+V; uels{kappafrac_extcompr}='kappafrac_extcompr';
kappafrac_HPturb=4+V;   uels{kappafrac_HPturb}='kappafrac_HPturb';
kappafrac_IPturb=5+V;   uels{kappafrac_IPturb}='kappafrac_IPturb';
kappafrac_set=(kappafrac_GT:kappafrac_IPturb)';
V=kappafrac_IPturb;

flow_ATRair_2nd=1+V;    uels{flow_ATRair_2nd}='flow_ATRair_2nd';
flow_ATRair_ext=2+V;    uels{flow_ATRair_ext}='flow_ATRair_ext';
```

```
flow_ATRair_set=[flow_ATRair_2nd; flow_ATRair_ext];
HPsteamflow =3+V;       uels{HPsteamflow}    = 'HPsteamflow';
IPsteamflow =4+V;       uels{IPsteamflow}    = 'IPsteamflow';
trbflows_set=[exhaust_set; flow_ATRair_set; HPsteamflow; IPsteamflow];
V=IPsteamflow;

comb_airflow= 1+V;       uels{comb_airflow}='comb_airflow';
V=comb_airflow;

Ti_GT        = 1+V;      uels{Ti_GT}      = 'Ti_GT';
Ti_ATRair_2nd=2+V;       uels{Ti_ATRair_2nd}='Ti_ATRair_2nd';
Ti_ATRair_ext= 3+V;      uels{Ti_ATRair_ext}='Ti_ATRair_ext';
Ti_HPturb    = 4+V;      uels{Ti_HPturb}  = 'Ti_HPturb';
Ti_IPturb    = 5+V;      uels{Ti_IPturb}  = 'Ti_IPturb';
Ti_ATRair_set= [Ti_ATRair_2nd; Ti_ATRair_ext];
idealoutlettemp_set=(Ti_GT:Ti_IPturb)';
V=Ti_IPturb;

v_all=(1:V)';

temperatures=[feedtemp LTStemp HTStemp airtemp steamtemp LTStemp2 ...
    HTStemp2 GTfueltemp T_GT T_ATRair_2nd T_ATRair_ext T_HPturb T_IPturb];
massflows=[CS_CO2 CS_CO CS_H2 CCU_H2 CCU_CO CCU_CO2 CCU_H2O CCU_N2 ...
    CCU_CH4 feedflow steamfeed exh_H2O exh_CO2 exh_O2 exh_N2 ...
    flow_ATRair_2nd flow_ATRair_ext HPsteamflow IPsteamflow comb_airflow];
energyflows=[mCp_GTfuel mCp_HTSprod mCp_LTSprod mCp_refprod W_GT ...
    W_extcompr W_2ndcompr W_HPturb W_IPturb fuelcompr];
pressures=[feedpressure p_GT p_ATRair_2nd p_ATRair_ext p_HPturb p_IPturb];
```

F.2.3 Handling the HYSYS process simulations — simulation.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% simulation.m — running similations in HYSYS based on LHS sampling
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% HOME MADE:    sampling.m      making Latin hypercube sampling scheme
%               hywait.m        waiting for HYSYS to be ready (necessary?)
%                               — using hyissolving.m from hysyslib
% HYSYSLIB:     hyconnect.m     connecting to HYSYS case
%               hyspread.m      connecting to spreadsheet in HYSYS case
%               hycell.m        connecting to cell in HYSYS spreadsheet
%               hyvalue.m       getting value of a HYSYS spreadcheet cell
%               hyunits.m       getting unit for a HYSYS spreadcheet cell
%               hyrelease.m     releasing the HYSYS case
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% If run separately, remember to run irccdata first

% Connecting to HYSYS case and spreadsheet
hysys=hyconnect([pwd '\ircc7.hsc']);
sheet=hyspread(hysys,'I/O');

hywait(hysys)

% Connecting to cells of the HYSYS spreadsheet
hc=cell(1,v);
for i=1:v
    hc{i}=hycell(sheet,['A' num2str(i)]);
end

% Defining feasible ranges for input variables
low=zeros(1,iv);
high=zeros(1,iv);
step=ones(1,iv);
low(feedtemp)=200;
high(feedtemp)=500;
low(LTStemp)=180;
high(LTStemp)=250;
low(feedpressure)=18;
high(feedpressure)=30;
low(HTStemp)=300;
high(HTStemp)=450;
low(S2C)=1;
high(S2C)=2;
low(airtemp)=410;
high(airtemp)=530;
low(steamtemp)=280;
high(steamtemp)=345;

% Changing units
unitmult=ones(1,iv);
if strcmp(hyunits(hc{feedpressure}{1}),'kPa')
    unitmult(feedpressure)=100;
end
unitmult(S2C)=10000*M.H2O/M.CH4/3600;
low=low.*unitmult;
high=high.*unitmult;

% Widening the ranges to avoid bad fit around the bounds
a=.2;
high=(1+a)*high-a*low;
low=(1+a)*low-a*high;
```

```
% Setting the number of simulations to be done
sims=5000;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calling the function sampling.m to make a latin hypercube sampling scheme
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sample=(sampling(sims,iv)-rand(sims,iv))./sims*diag(high-low) ...
    +ones(sims,1)*low;

% Running HYSYS for the different values of the input variables, and
% saving the results for all regression variables in the matrix A
A=zeros(sims,v);
ii=1;
sample(:,feedpressure)=sample(:,feedpressure)- ...
    (sample(:,feedpressure)>3000).*(sample(:,feedpressure)-3000);
for si=1:sims
    for j=1:iv
        hyset(hc{j},sample(si,j));
    end
    hywait(hysys)
    for jj=1:v
        A(ii,jj)=hyvalue(hc{jj});
    end
    ii=ii+1;
end
hyrelease(hysys);

% Modifying the A matrix
% Changing from kPa to bar
A(:,feedpressure)=A(:,feedpressure)*.01;
% Scaling mass and energy flows
A(:,regvarflows)=A(:,regvarflows)/flowscale;
% Going from steam flow rate to steam-to-carbon ratio
A(:,S2C)=A(:,S2C).*M.CH4./(M.H2O*CH4_in*flowscale);
% Going from cooling/heating duty to mCp-flow [HRSG]
A(:,mCp_refprod)=A(:,mCp_refprod)./(T_refprod - A(:,HTStemp))*3600;
A(:,mCp_HTSprod)=A(:,mCp_HTSprod)./(A(:,HTStemp2) - A(:,LTStemp))*3600;
A(:,mCp_LTSprod)=A(:,mCp_LTSprod)./(A(:,LTStemp2) + C2K - T_WRfeed)*3600;
A(:,mCp_GTfuel)=A(:,mCp_GTfuel)./(200-30)*3600;
% Changing from kW to kJ/h
A(:,fuelcompr)=A(:,fuelcompr)*3600;

% Saving the matrix A in A_matrix7.mat
save('A_matrix7', 'A')
```

F.2.4 Making Latin hypercube sampling scheme — sampling.m

```
function sample=sampling(n, m)
% SAMPLING(n,m) returns a [n x m] matrix with rows representing different
% combinations of numbers (a Latin Hypercube)

    % Initializing
    M=cell(m,1);
    for j=1:m
        % List of unused intervals for each variable
        M{j}=1:n;
    end
    sample=zeros(n,m);

    for i=1:n
        for j=1:m
            % Picking an unused interval for each sample
            sample(i,j)=M{j}(ceil(rand*(n-i+1)));
            % Removing the picked interval from the list of unused ones
            M{j}=setdiff(M{j},sample(i,j));
        end
    end
return
```

F.2.5 Dealing with regression coefficients and related sets — regcoeffs.m

```
%REGCOEFFS fits second order polynomial curves to variables represented
%by the columns in A (containing simulation results), and writes the
%regression coefficients and other information to a .gdx file

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Functions used:
%   "HOME MADE":
%   - polyreg2.m
%   GDXMRW package:
%   - writegdx.m (using wgdx.m)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input variables which theoretically cannot impact the output variable
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
noreg=zeros(v-iv, iv);
% HTS product temperature does not change as the LTS feed temperature
% is changed (because the latter is downstream of the former)
noreg(HTStemp2-iv,LTStemp)=1;
```

```
% Reformer product cooling does not change as the LTS feed temperature
% is changed (because the latter is downstream of the former)
noreg(mCp_refprod-iv,LTStemp)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reading simulation data from file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
load A_matrix7

vars=length(uels);

% Specifying the file to write to
filetowriteto='regcoeffs.gdx';

% Normalizing all values to the interval [-.5 , .5]
high=zeros(1,v);
low=high;
for i=1:v
    high(i)=max(A(:,i));
    low(i)=min(A(:,i));
    A(:,i)=(A(:,i)-.5*(high(i)+low(i)))./(high(i)-low(i));
end

% Fitting the data to a second order polynomial function of the dependent
% variables
nbeta=1+iv*(1.5+.5*iv);
if runregression
    y_hat=zeros(v-iv,nbeta);
    RSS=zeros(v-iv,1);
    for i=iv+1:v
        [y_hat(i-iv,:) RSS(i-iv,1)]=polyreg2(A(:,1:iv),A(:,i), noreg(i-iv,:));
    end
    save('regressioncoefficients','y_hat','RSS')
else
    load regressioncoefficients
end

% Writing the calculated regression coefficients to a .gmx-file
uels{vars+1}='beta_0';
i=1;
y_hat_gdx=zeros(nbeta*(v-iv),3);
for j=1:nbeta
    for k=1:dv
        y_hat_gdx(i,1)=j+vars;
        y_hat_gdx(i,2)=k+iv;
        y_hat_gdx(i,3)=y_hat(k,j);
        i=i+1;
    end
```

```
end
linbeta=zeros(iv,2);
quadbeta=linbeta;
for i=1:iv
    linbeta(i,1)=i;
    linbeta(i,2)=i+1+vars;
    quadbeta(i,1)=i;
    quadbeta(i,2)=i+1+iv+vars;
    uels{vars+1+i}=['beta_' num2str(i)];
    uels{vars+1+iv+i}=['beta_' num2str(i) ',' num2str(i)];
end
bilinbeta=zeros(.5*iv*(iv-1),3);
k=1;
for i=1:iv
    for j=i+1:iv
        bilinbeta(k,1)=i;
        bilinbeta(k,2)=j;
        bilinbeta(k,3)=k+1+2*iv+vars;
        uels{k+1+2*iv+vars}=['beta_' num2str(i) ',' num2str(j)];
        k=k+1;
    end
end
high_gdx=zeros(v,2);
low_gdx=high_gdx;
for i=1:v
    high_gdx(i,1)=i;
    high_gdx(i,2)=high(i);
    low_gdx(i,1)=i;
    low_gdx(i,2)=low(i);
end

b_set=(1:nbeta)'+vars;
writegdx(filetowriteto, ...
    'parameter', 'betas',   y_hat_gdx,...
    'set',       'v',       v_set, ...
    'set',       'iv',      iv_set, ...
    'set',       'dv',      dv_set, ...
    'set',       'b',       b_set, ...
    'set',       'lb',      linbeta, ...
    'set',       'qb',      quadbeta, ...
    'set',       'bb',      bilinbeta, ...
    'parameter', 'high',    high_gdx, ...
    'parameter', 'low',     low_gdx, ...
    uels);
```

F.2.6 Regression — polyreg2.m

```
function [betas RSS] = polyreg2(x,y, varargin)
%POLYREG2 2nd order polynomial regression
%   POLYREG2(x,y) performs a second order polynomial regression, trying to
%   fit the data as, y_hat=f(beta_hat,x)
%       x(s,v)      = value of input variable v in simulation s
%       y(s)        = value of the output variable from simulation s
%   POLYREG2(x,y,noreg) performs the regression "without" the variables
%       represented by noreg)
%   POLYREG2(x,y,noreg,limit) does the same as POLYREG2(x,y,noreg), but
%       eliminating betas with lower absolute value than specified by limit

    %Setting up the C matrix (with 2nd degree polynomials)
    [sims iv]=size(x);
    n=1+iv*(1.5+.5*iv); % number of betas (regression coefficients)
    C=zeros(sims,n);
    for s=1:sims
        C(s,1)=1;                               % Constant term
        for v=1:iv
            C(s,1+v)=x(s,v);                    % First−order terms
            C(s,iv+1+v)=x(s,v)^2;               % Second−order terms
        end
        jj=2+2*iv;
        for v1=1:iv−1;
            for v2=v1+1:iv
                C(s,jj)=x(s,v1)*x(s,v2);    % Bilinear terms
                jj=jj+1;
            end
        end
    end

    % The limit to be used if not specified by optional argument
    limit=5e−2;

    % Dealing with optional arguments
    if size(varargin,2)>=1
        % Removing regression coefficients as set by 'noreg'
        if sum(varargin{1})>0
            for v=1:iv
                check{v}=[v+1 v+iv+1];
            end
            jj=2+2*iv;
            for v1=1:iv−1;
                for v2=v1+1:iv
                    check{v1}=union(check{v1},jj);
                    check{v2}=union(check{v2},jj);
                    jj=jj+1;
                end
            end
            for i=find(varargin{1},iv)
                C(:,check{i})=0;
                for j=check{i}
                    sims=sims+1;
                    C(sims,j)=1;
                    y(sims)=0;
                end
            end
        end
        % Setting the limit to the corresponding optional argument
        if size(varargin,2)>=2
            limit=varargin{2};
        end
    end

    for loop=1:5
        % Solving with respect to least squares
        betas = C\y;
        %       betas is now the solution (with respect to least squares)
        %       of the equation set C*betas=y.
        RSS = (y-C*betas)'*(y-C*betas); %The residual sum of squares

        % Removing "insignificant" coefficients
        % 1.    Adjusting the coefficients to account for that the
        %       variables are in the range [-.5, .5]
        betas_adj=[betas(1)*4;betas(2:iv+1)*2;betas(iv+2:2*iv+1);...
            betas(2*iv+2:n)];
        % 2.    Determining which coefficients too keep
        keep=abs(betas_adj)>max(betas_adj)*limit;
        if sum(betas.*(1-keep))==0
            return;
        end
        for i=1:n
            if keep(i)==0 & betas(i)~=0
                % Removing the impact of beta(i) from the regression
                % equations
                C(:,i)=0;

                % Making a new constraint to make sure beta(i) is
                % set to zero (when minimizing least squares)
                sims=sims+1;
                C(sims,:)=zeros(1,n);
                C(sims,i)=1;
                y(sims)=0;
            end
        end
    end
```

```
    % In case there are more "insignificant" regression coefficients left
    % after the loops: show it.
   for i=1:n
       if abs(betas(i))<1e−12
           betas(i)=0;
       end
   end
   if sum(betas~=betas.*keep);
    disp('WARNING: There are still insignificant regression coefficients.')
    disp(betas.*(1−keep))
   end
```

F.2.7 Setting some model parameters and basic equations — setmodelparameters.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% setmodelparameters.m − setting some general equations and parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% GDXMRW:       writegdx.m      writing data to gdx file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Setting (some) linear inequality constraints
% (linconstr * x <= linconstrbound)
% Minimum 90 % CO2 captured:
    CO2captureconstr=1+length(uels);  uels{CO2captureconstr}='CO2capture';
linconstr_set=(CO2captureconstr:CO2captureconstr)';
linconstr=[ CO2captureconstr     CCU_CO2     −split.CO2(2)/M.CO2;
            CO2captureconstr     feedflow    CH4_in/M.CH4*mincapture];
linconstrbound=[CO2captureconstr            0];

% Setting (some) linear equality constraints (eqconstr * x = eqconstrconst)
% HP steam flow = LP steam flow + ATR steam flow:
    HPsteambalanceconstr=1+length(uels);
    uels{HPsteambalanceconstr}='HPsteambalance';
% Intermediate pressure = ATR pressure:
    IPconstr=1+HPsteambalanceconstr;     uels{IPconstr}='IP=feedpressure';
    IPconstr2=1+IPconstr;                uels{IPconstr2}='IP=p_HPturb_out';
% HP turbine outlet temperature = temperature of steam to ATR:
    steamtempconstr=1+IPconstr2; uels{steamtempconstr}='steam temperature';
eqconstr_set=(HPsteambalanceconstr:steamtempconstr)';
eqconstr=[HPsteambalanceconstr   HPsteamflow     1;
          HPsteambalanceconstr   IPsteamflow     −1;
          HPsteambalanceconstr   steamfeed       −1;
          IPconstr               feedpressure    1;
          IPconstr               p_HPturb        −1;
          IPconstr2               p_IPturb        1;
          IPconstr2               p_HPturb        -1;
          steamtempconstr         steamtemp       1;
          steamtempconstr         T_HPturb        -tempscale];
eqconstrconst= [HPsteambalanceconstr    0;
                IPconstr                0;
                IPconstr2               0;
                steamtempconstr         -C2K];

% Setting (some) product constraints (x1=k*x2*x3)
% Steam feed flow = methane feed flow * steam-to-carbon ratio * M.H2O/M.CH4
    steamfeedconstr=1+length(uels);     uels{steamfeedconstr}='Steam feed';
prodconstr_set=(steamfeedconstr:steamfeedconstr)';
prodconstr=[steamfeedconstr steamfeed S2C feedflow CH4_in*M.H2O/M.CH4];

% Dependent variables that are a function of the feed flow rate
feedflowdep=[feedflowdep' ones(length(feedflowdep), 1)];

% Writing data to file
writegdx('modelparameters.gdx', ...
    'set',       'lc',                linconstr_set, ...
    'set',       'ec',                eqconstr_set, ...
    'set',       'pr',                prodconstr_set, ...
    'set',       'feedflow',          feedflow, ...
    'parameter', 'feedflowdep',       feedflowdep, ...
    'parameter', 'linconstr',         linconstr, ...
    'parameter', 'linconstrbound',    linconstrbound, ...
    'parameter', 'eqconstr',          eqconstr, ...
    'parameter', 'eqconstrconst',     eqconstrconst, ...
    'parameter', 'prodconstr',        prodconstr, ...
    'parameter', 'startingpoint',     startingpoint, ...
    uels);
```

F.2.8 Calculating combustion parameters — combustion.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% combustion.m - setting parameters and sets related to combustion
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% GDXMRW:       writegdx.m      writing data to gdx file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Heat capacities (based on enthalpy equations)
% From HYSYS H = a+bT+cT^2+dT^3+eT^4+fT^5 [kJ/kg], T [K]
% Valid for T between -270 and 5000 C. [b c d e f]
HCp_coeffs.H2O=[1.9145 -3.9574e-4 8.76206e-7 -4.95055e-10 1.03846e-13];
HCp_coeffs.CO=[1.0739 -1.7265e-4 3.02226e-7 -1.37526e-10 2.00356e-14];
HCp_coeffs.CO2=[0.618139 4.84485e-4 -1.49353e-7 2.2905e-11 -1.37045e-15];
HCp_coeffs.H2=[13.8376 2.999806e-4 3.458931e-7 -9.712927e-11 7.731201e-15];
HCp_coeffs.CH4=[2.36459 -2.13247e-3 5.6618e-6 -3.72476e-9 8.60896e-13];
HCp_coeffs.N2=[.982747 9.71424e-5 -4.15795e-10 -3.65548e-12 4.05013e-16];
HCp_coeffs.O2=[.952 -2.81129e-4 6.55206e-7 -4.52296e-10 1.08756e-13];
for j = components
    i=char(j);
    eval(['HCp_coeffs.' i '=HCp_coeffs.' i '.*[1 2 3 4 5];']);
end

TIT=1350+C2K;
if lowerTIT>0
    TIT=lowerTIT+C2K;
end
T_air=415+C2K;
T_fuel=200+C2K;

% Calculating the amount of heat needed to heat one kg of air from the
% temperature out of the compressor to TIT
a=HCp_coeffs.O2;
q_comb_air=airO2mass*(a(1)*(TIT-T_air)+.5*a(2)*(TIT^2-T_air^2)+...
    a(3)/3*(TIT^3-T_air^3)+.25*a(4)*(TIT^4-T_air^4)+...
    .2*a(5)*(TIT^5-T_air^5));                              % [kJ/kg]
a=HCp_coeffs.N2;
q_comb_air=q_comb_air+airN2mass*(a(1)*(TIT-T_air)+...
    .5*a(2)*(TIT^2-T_air^2)+a(3)/3*(TIT^3-T_air^3)+...
    .25*a(4)*(TIT^4-T_air^4)+.2*a(5)*(TIT^5-T_air^5)); % [kJ/kg]

% Calculating the amount of heat needed to heat one kg from the preheat
% temperature to TIT for each combustion component
for j=components
    i=char(j);
    eval(['a=HCp_coeffs.' i ';']);
    q=(a(1)*(TIT-T_fuel)+.5*a(2)*(TIT^2-T_fuel^2)+...
        a(3)/3*(TIT^3-T_fuel^3)+.25*a(4)*(TIT^4-T_fuel^4)+...
        .2*a(5)*(TIT^5-T_fuel^5));
    eval(['CpdeltaT.' i '=q;']); % [kJ/kg/K]
end
CpDT=[CpdeltaT.H2O; CpdeltaT.CO2; CpdeltaT.O2; CpdeltaT.N2];   %[kJ/kg/K]

% q=[CpdeltaT.H2O                                  % H2O
%     CpdeltaT.CO2-.5*CpdeltaT.O2                  % CO
%     CpdeltaT.CO2                                 % CO2
%     CpdeltaT.H2O-.5*CpdeltaT.O2                  % H2
%     CpdeltaT.CO2+2*(CpdeltaT.H2O-CpdeltaT.O2)    % CH4
%     CpdeltaT.N2];                                % N2
% Stoichiometry of the combustion reactions
combreaction=zeros(v,4);
combreaction(CCU_H2O,1)=1                                 *split.H2O(1);
combreaction(CCU_CO,:)=[0 1*M.CO2 -.5*M.O2 0]       /M.CO *split.CO (1);
combreaction(CCU_CO2,2)=1                                 *split.CO2(1);
combreaction(CCU_H2,:)=[1*M.H2O 0 -.5*M.O2 0]       /M.H2 *split.H2 (1);
combreaction(CCU_CH4,:)=[2*M.H2O 1*M.CO2 -2*M.O2 0] /M.CH4*split.CH4(1);
combreaction(CCU_N2,4)=1                                  *split.N2 (1);

% The composition of air, with respect to exhaust components
aircomp=[exh_O2 airO2mass;
         exh_N2 airN2mass];

% Lower heating value in the gas turbine fuel
LHV=zeros(v,1);
LHV(CCU_CO)=LHV_CO*split.CO(1);     %[kJ/kg]
LHV(CCU_H2)=LHV_H2*split.H2(1);     %[kJ/kg]
LHV(CCU_CH4)=LHV_CH4*split.CH4(1);  %[kJ/kg]

% Set of exponents for Cp (or H) equations
expset=(1:5)'+length(uels);
uels(expset)={'exp1', 'exp2', 'exp3', 'exp4', 'exp5'};

% Set of combustion related variables
v_comb=union(exhaust_set, [outlettemp_set; W_set; comb_airflow]);

% Writing data to file
combreaction_gdx=zeros(v*length(exhaust_set),3);
k=1;
for i=1:v
    for j=1:4
        combreaction_gdx(k,1)=i;
        combreaction_gdx(k,2)=exhaust_set(j);
        combreaction_gdx(k,3)=combreaction(i,j);
        k=k+1;
    end
end
writegdx('combustion.gdx', ...
    'set',       'v_comb',       v_comb, ...
    'set',       'exhcomp',      exhaust_set , ...
    'set',       'T_out',        outlettemp_set, ...
    'set',       'W',            W_set, ...
    'set',       'comb_airflow', comb_airflow, ...
    'parameter', 'combreaction', combreaction_gdx, ...
    'parameter', 'q_comb_air',   q_comb_air, ...
    'parameter', 'air',          aircomp, ...
    'parameter', 'LHV',          [(1:v)' LHV], ...
    'parameter', 'CpDT',         [exhaust_set CpDT], ...
    'parameter', 'tempscale',    tempscale, ...
    uels);
```

F.2.9 Calculating parameters for air distribution — air.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% air.m - setting parameters and sets related to air distribution
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% GDXMRW:       writegdx.m      writing data to gdx file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% The amount of air needed in the ATR per mass flow of CO2, CO and H2 in
% the cooled syngas
ATR_air=zeros(v,1);
ATR_air(CS_CO2)=2/M.CO2;
ATR_air(CS_CO)=1.5/M.CO;
ATR_air(CS_H2)=-.5/M.H2;
ATR_air=ATR_air*M.O2/airO2mass;

% The amount of air compressed in the main (large-scale) air compressor
airflow=640*3600/flowscale;

% Air related variables
v_air=union(flow_ATRair_set, [p_ATRair_set; T_ATRair_set]);

% Writing data to file
writegdx('air.gdx', ...
    'set',       'v_air',       v_air, ...
    'set',       'p_feed',      feedpressure, ...
    'set',       'T_airfeed',   airtemp, ...
    'set',       'flow_ATRair', flow_ATRair_set, ...
    'set',       'flow_ATRair_2nd', flow_ATRair_2nd, ...
    'set',       'p_ATRair',    p_ATRair_set, ...
    'set',       'T_ATRair',    T_ATRair_set, ...
    'set',       'ATRair',      [p_ATRair_set T_ATRair_set flow_ATRair_set],...
    'parameter', 'p_and_T',     [p_ATRair_set, T_ATRair_set, ones(2,1)],...
    'parameter', 'ATR_air',     [v_set ATR_air], ...
    'parameter', 'airflow',     airflow, ...
    uels);
```

F.2.10 Calculating parameters for modeling of turbo-machinery — turbo.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% turbo.m — setting parameters and sets related to turbines and compressors
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% GDXMRW:       writegdx.m      writing data to gdx file
% and indirectly the home made script HPturbCp.m for simulations and
% regression for HP and IP steam turbines
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Fixed temperatures
T_air= 415+C2K;  % [K]
TIT=  1350+C2K;  % [K]
T_fuel=200+C2K;  % [K]
T_air_2nd=T_air; % [K]
T_air_ext=15+C2K;% [K]
T_HP = 560 + C2K;% [K]
T_RH = 560 + C2K;% [K]
p_air_2nd=18;  %[bar]
p_air_ext=1.01; %[bar]
HP=120;         %[bar]
LP=4;           %[bar]

if lowerTIT>0
    TIT=lowerTIT+C2K;
end

Cpcoeffs=[  HCp_coeffs.H2O
            HCp_coeffs.CO2
            HCp_coeffs.O2
            HCp_coeffs.N2
            airO2mass*HCp_coeffs.O2+airN2mass*HCp_coeffs.N2
            airO2mass*HCp_coeffs.O2+airN2mass*HCp_coeffs.N2
            HCp_coeffs.H2O
            HCp_coeffs.H2O];

%TEMPSCALE
Cpcoeffs=Cpcoeffs*diag(tempscale.^(1:5));
TIT=TIT/tempscale;
T_air_2nd=T_air_2nd/tempscale;
T_air_ext=T_air_ext/tempscale;
T_HP=T_HP/tempscale;
T_RH=T_RH/tempscale;

% Some constants
Combustionpressure=18;              %[bar]
Outletpressure=1.03;                %[bar]
Pressureloss=.5;                    %[bar]
Turbineefficiency=.91;
Compressorefficiency=.88;
```

```
HPturbineefficiency=.94;
IPturbineefficiency=.92;
Mainaircompressorefficiency=.92;

% The work of CO2 compression, per kg of CO2 captured
w_CO2compr=95*3.6; % [kJ/kg]

% Calculating the work of compressing 1 kg of air to 18 bar
a=HCp_coeffs.O2;
T1=20+C2K; T2=415+C2K;
w_air=airO2mass*(a(1)*(T2-T1)+.5*a(2)*(T2^2-T1^2)+...
    a(3)/3*(T2^3-T1^3)+.25*a(4)*(T2^4-T1^4)+...
    .2*a(5)*(T2^5-T1^5));                               % [kJ/kg]
a=HCp_coeffs.N2;
w_air=w_air+airN2mass*(a(1)*(T2-T1)+.5*a(2)*(T2^2-T1^2)+...
    a(3)/3*(T2^3-T1^3)+.25*a(4)*(T2^4-T1^4)+...
    .2*a(5)*(T2^5-T1^5));                               % [kJ/kg]
w_air=w_air/Mainaircompressorefficiency;

Cpcoeffs_gdx=zeros(length(expset)*(length(trbflows_set)),3);
k=1;
for i=1:length(trbflows_set);
    for j=1:length(expset)
        Cpcoeffs_gdx(k,1)=trbflows_set(i);
        Cpcoeffs_gdx(k,2)=expset(j);
        Cpcoeffs_gdx(k,3)=Cpcoeffs(i,j);
        k=k+1;
    end
end

GT_set=length(uels)+1;
secondcompressor_set=GT_set+1;
extracompressor_set=secondcompressor_set+1;
HPsteamturb_set=extracompressor_set+1;
IPsteamturb_set=HPsteamturb_set+1;
uels{GT_set}='Gas turbine';
uels{secondcompressor_set}='Second compressor';
uels{extracompressor_set}='Extra compressor';
uels{HPsteamturb_set}='HP steam turbine';
uels{IPsteamturb_set}='IP steam turbine';

% The set of turbomachinery to be modelled with thermodynamics equations
trb1_set=[GT_set secondcompressor_set extracompressor_set]';
% The set of turbomachinery to be modelled with linear regression from
% simulations
trb2_set=[HPsteamturb_set IPsteamturb_set]';
% The set of turbines and compressors modelled in one of these ways
trb_set=[trb1_set; trb2_set];
```

```
% Connecting flowrates to turbines/compressors
turboflows=[GT_set*ones(4,1) exhaust_set ones(4,1);
    [secondcompressor_set;extracompressor_set] flow_ATRair_set ones(2,1);
    HPsteamturb_set     HPsteamflow     1;
    IPsteamturb_set     IPsteamflow     1];

% Efficiencies for the turbines/compressors
efficiency=[GT_set              Turbineefficiency;
            secondcompressor_set —1/Compressorefficiency;
            extracompressor_set —1/Compressorefficiency;
            HPsteamturb_set     HPturbineefficiency;
            IPsteamturb_set     IPturbineefficiency];
% Inlet pressures for the turbines/compressors
p_in=[  GT_set                  Combustionpressure—Pressureloss;
        secondcompressor_set    p_air_2nd;
        extracompressor_set     p_air_ext;
        HPsteamturb_set         HP;
        IPsteamturb_set         LP]; %This is really the outlet pressure
% Inlet temperatures for the turbines/compressors
inlettemps=[GT_set              TIT;
        secondcompressor_set    T_air_2nd;
        extracompressor_set     T_air_ext;
        HPsteamturb_set         T_HP;
        IPsteamturb_set         T_RH];
% Parameter to modify turbo—equations
kappafracsign=[GT_set           1;
        secondcompressor_set    1;
        extracompressor_set     1;
        HPsteamturb_set         1;
        IPsteamturb_set         —1]; %This interchanges p_in and x(p_out)

% Universal gas constant [kJ/kmol/K]
R=8.314472;
% 'Molar mass of air' [kg/kmol]
M.air=airO2*M.O2+airN2*M.N2;
% Gas constants with adjusted for molar mass [kJ/kg/K]
R_m=[trbflows_set R./[M.H2O M.CO2 M.O2 M.N2 M.air M.air M.H2O M.H2O]'];


% Setting regression coefficients for the turbines/compressors modelled
% with linear regression and simulation. The numbers are calculated in
% HPturbCp.m
% The equations are on the form w(p) = w0 + dwdp*p and T(p) = T0 + dTdp*p.
w0=[    HPsteamturb_set        694.90347928384426;
        IPsteamturb_set         279.90610444742094];
dwdp=[  HPsteamturb_set         —9.937395471077858;
        IPsteamturb_set         9.7361026004373343];
```

```
T0=[    HPsteamturb_set         (186.53771204510576+C2K)/tempscale;
        IPsteamturb_set         (432.07460037381867+C2K)/tempscale];
dTdp=[  HPsteamturb_set         5.3325971476587695/tempscale;
        IPsteamturb_set         −5.1884382644666687/tempscale];
% Changing these data if the a different high pressure level is chosen
if otherHP==130
    w0=[    HPsteamturb_set         704.5183;
            IPsteamturb_set         57.0137];
    dwdp=[  HPsteamturb_set         −9.6975;
            IPsteamturb_set         10.5734];
    T0=[    HPsteamturb_set         (178.3142+C2K)/tempscale;
            IPsteamturb_set         (478.5753+C2K)/tempscale];
    dTdp=[  HPsteamturb_set         5.2336/tempscale;
            IPsteamturb_set         −5.5605/tempscale];
end

% Coefficient to find condenser cooling work
STcondensercooling=.005; % 0.5 percent of power from steam turbines

% Writing data to file
writegdx('turbo.gdx', ...
    'set',       'v_all',        v_all, ...
    'set',       'trb',          trb_set, ...
    'set',       'trb1',         trb1_set, ...
    'set',       'trb2',         trb2_set, ...
    'set',       'T_trb_out',    outlettemp_set, ...
    'set',       'Ti_trb_out',   idealoutlettemp_set, ...
    'set',       'W',            W_set, ...
    'set',       'trbflows',     trbflows_set, ...
    'set',       'p_out',        p_out_set, ...
    'set',       'exp',          expset, ...
    'set',       'kappafrac',    kappafrac_set, ...
    'parameter', 'turboflows',   turboflows, ...
    'parameter', 'T_trb_in',     inlettemps, ...
    'parameter', 'p_in',         p_in, ...
    'parameter', 'exponents',    [expset (1:5)'], ...
    'parameter', 'Cpcoeffs',     Cpcoeffs_gdx, ...
    'parameter', 'R',            R_m, ...
    'parameter', 'eff',          efficiency, ...
    'parameter', 'kappafracsign',kappafracsign, ...
    'parameter', 'w0',           w0, ...
    'parameter', 'dwdp',         dwdp, ...
    'parameter', 'T0',           T0, ...
    'parameter', 'dTdp',         dTdp, ...
    uels);
```

F.2.11 Setting and calculating upper and lower bounds — bounds.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% bounds.m - setting upper and lower bounds for each variable
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used: none
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% 'Independent variables'
lowerbound(feedtemp)        = 200;  upperbound(feedtemp)        = 500;
lowerbound(LTStemp)         = 180;  upperbound(LTStemp)         = 250;
lowerbound(feedpressure)    =  18;  upperbound(feedpressure)    =  30;
lowerbound(HTStemp)         = 300;  upperbound(HTStemp)         = 450;
lowerbound(S2C)             =   1;  upperbound(S2C)             =   2;
lowerbound(airtemp)         = 410;  upperbound(airtemp)         = 530;
lowerbound(steamtemp)       = ...
    (T0(1,2)+lowerbound(feedpressure)*dTdp(1,2))*tempscale-C2K;
upperbound(steamtemp)       = ...
    (T0(1,2)+upperbound(feedpressure)*dTdp(1,2))*tempscale-C2K;

% 'Dependent variables'
lowerbound(CS_CO2)          =   0;  upperbound(CS_CO2) =CH4_in/M.CH4*M.CO2;
lowerbound(CS_CO)           =   0;  upperbound(CS_CO)  =CH4_in/M.CH4*M.CO;
lowerbound(CS_H2)           =   0;
upperbound(CS_H2)=CH4_in/M.CH4*M.H2*(2+upperbound(S2C));

lowerbound(HTStemp2) = lowerbound(LTStemp);   upperbound(HTStemp2) = 1000;
lowerbound(LTStemp2) = T_WRfeed;              upperbound(LTStemp2) = 500;

lowerbound(mCp_refprod)     = low(mCp_refprod);
upperbound(mCp_refprod)     = high(mCp_refprod);
lowerbound(mCp_HTSprod)     = low(mCp_HTSprod);
upperbound(mCp_HTSprod)     = high(mCp_HTSprod);
lowerbound(mCp_LTSprod)     = low(mCp_LTSprod);
upperbound(mCp_LTSprod)     = high(mCp_LTSprod);
lowerbound(mCp_GTfuel)      = low(mCp_GTfuel);
upperbound(mCp_GTfuel)      = high(mCp_GTfuel);

lowerbound(GTfueltemp)      = CCUoutlettemp;
upperbound(GTfueltemp) = T_fuel-C2K;

lowerbound(CCU_H2O)         =   0;
upperbound(CCU_H2O)=CH4_in/M.CH4*M.H2O*(2+upperbound(S2C));
lowerbound(CCU_CO)          =   0;  upperbound(CCU_CO) = CH4_in/M.CH4*M.CO;
lowerbound(CCU_CO2)         =   0;  upperbound(CCU_CO2)=CH4_in/M.CH4*M.CO2;
lowerbound(CCU_H2)          =   0;
upperbound(CCU_H2)=CH4_in/M.CH4*M.H2*(2+upperbound(S2C));
lowerbound(CCU_CH4)         =   0;  upperbound(CCU_CH4)=CH4_in*split.CH4(1);
lowerbound(CCU_N2)          =   0;
upperbound(CCU_N2)=CH4_in/M.CH4*M.O2/airO2mass*airN2mass*2;

lowerbound(fuelcompr)       =   0;
upperbound(fuelcompr) = CH4_in*LHV_CH4*.1;

% ATR inlet flow variables
lowerbound(feedflow) = airflow/CH4_in*.02;
upperbound(feedflow) = airflow/CH4_in*.05;

lowerbound(steamfeed)=lowerbound(S2C)*lowerbound(feedflow)*CH4_in*M.H2O/M.CH4;
upperbound(steamfeed)=upperbound(S2C)*upperbound(feedflow)*CH4_in*M.H2O/M.CH4;

% 'Combustion related variables'
lowerbound(exhaust_set)         = 0;
upperbound(exh_H2O)=airflow*airO2mass*.5*M.H2O*M.O2;
    %upperbound(feedflow)*CH4_in*(upperbound(S2C)+2)*M.H2O/M.CO2;
upperbound(exh_CO2)=upperbound(feedflow)*CH4_in*M.CO2/M.CH4;
upperbound(exh_O2)=airflow*airO2mass;
upperbound(exh_N2)=airflow*airN2mass + ...
        upperbound(feedflow)*CH4_in*(2)*M.O2/M.CO2/airO2mass;

% 'Air related variables'
lowerbound(comb_airflow)        = 0;
upperbound(comb_airflow)        = airflow;

% Turbo
lowerbound(W_GT)                = .2*lowerbound(feedflow)*CH4_in*LHV_CH4;
upperbound(W_GT)                = upperbound(feedflow)*CH4_in*LHV_CH4;
lowerbound(W_2ndcompr)          = 0;
upperbound(W_2ndcompr)          = upperbound(W_GT);
lowerbound(W_extcompr)          = 0;
upperbound(W_extcompr)          = upperbound(W_GT);
lowerbound(W_HPturb)            = 0;
upperbound(W_HPturb)            = upperbound(W_GT);
lowerbound(W_IPturb)            = 0;
upperbound(W_IPturb)            = upperbound(W_HPturb);

lowerbound(T_GT)                = 500+C2K;
upperbound(T_GT)                = 700+C2K;
    lowerbound(Ti_GT)               = 500+C2K;
    upperbound(Ti_GT)               = 600+C2K;
lowerbound(T_ATRair_2nd)        = T_air_2nd*tempscale;
lowerbound(T_ATRair_ext)        = T_air_ext*tempscale;
upperbound(T_ATRair_set)        = 600+C2K;
```

```
    lowerbound(Ti_ATRair_2nd)       = T_air_2nd*tempscale;
    lowerbound(Ti_ATRair_ext)       = T_air_ext*tempscale;
    upperbound(Ti_ATRair_set)       = 600+C2K;
lowerbound(T_HPturb)            = lowerbound(steamtemp)+C2K;
upperbound(T_HPturb)            = upperbound(steamtemp)+C2K;
    lowerbound(Ti_HPturb)           = 100+C2K;
    upperbound(Ti_HPturb)           = T_HP*tempscale;
lowerbound(T_IPturb)            = 100+C2K;
upperbound(T_IPturb)            = T_RH*tempscale;
    lowerbound(Ti_IPturb)           = 100+C2K;
    upperbound(Ti_IPturb)           = T_RH*tempscale;

lowerbound(p_GT)                = Outletpressure;
upperbound(p_GT)                = Outletpressure;
lowerbound(p_ATRair_set)        = lowerbound(feedpressure);
upperbound(p_ATRair_set)        = upperbound(feedpressure);
lowerbound(p_HPturb)            = lowerbound(feedpressure);
upperbound(p_HPturb)            = upperbound(feedpressure);
lowerbound(p_IPturb)            = lowerbound(feedpressure);
upperbound(p_IPturb)            = upperbound(feedpressure);

lowerbound(flow_ATRair_2nd)     = 0;
upperbound(flow_ATRair_2nd)     = ...
    upperbound(feedflow)*CH4_in/M.CH4*M.O2/airO2mass*2;
lowerbound(flow_ATRair_ext)     = 0;
upperbound(flow_ATRair_ext)     = ...
    upperbound(feedflow)*CH4_in/M.CH4*M.O2/airO2mass*2;
lowerbound(HPsteamflow)         = 0;
upperbound(HPsteamflow)         = ...
    upperbound(W_HPturb)/(w0(1,2)+upperbound(feedpressure)*dwdp(1,2));
lowerbound(IPsteamflow)         = 0;
upperbound(IPsteamflow)         = upperbound(HPsteamflow);

if aircompressorfix==1;
    upperbound(W_extcompr)=0;
    upperbound(flow_ATRair_ext)=0;
elseif aircompressorfix==2
    upperbound(W_2ndcompr)=0;
    upperbound(flow_ATRair_2nd)=0;
end

lowerbound(kappafrac_GT) =R/M.CO2/(HCp_coeffs.CO2*(TIT*tempscale).^(0:4)');
upperbound(kappafrac_GT) = R/M.N2/(HCp_coeffs.N2*lowerbound(T_GT).^(0:4)');
lowerbound(kappafrac_2ndcompr) = (airO2mass*R/M.O2+airN2mass*R/M.N2)/ ...
    (airO2mass*HCp_coeffs.O2*upperbound(Ti_ATRair_2nd).^(0:4)'+...
    airN2mass*HCp_coeffs.N2*upperbound(Ti_ATRair_2nd).^(0:4)');
upperbound(kappafrac_2ndcompr) = (airO2mass*R/M.O2+airN2mass*R/M.N2)/ ...
    (airO2mass*HCp_coeffs.O2*(T_air_2nd*tempscale).^(0:4)'+...
    airN2mass*HCp_coeffs.N2*(T_air_2nd*tempscale).^(0:4)');
lowerbound(kappafrac_extcompr) = lowerbound(kappafrac_2ndcompr);
upperbound(kappafrac_extcompr) = (airO2mass*R/M.O2+airN2mass*R/M.N2)/ ...
    (airO2mass*HCp_coeffs.O2*(T_air_ext*tempscale).^(0:4)' + ...
    airN2mass*HCp_coeffs.N2*(T_air_ext*tempscale).^(0:4)');
lowerbound(kappafrac_HPturb) =.1;
    %R/M.H2O/(HCp_coeffs.H2O*(T_HP*tempscale).^(0:4)');
upperbound(kappafrac_HPturb) =.5;
    %R/M.H2O/(HCp_coeffs.H2O*lowerbound(T_HPturb).^(0:4)');
lowerbound(kappafrac_IPturb) =.1;
    %R/M.H2O/(HCp_coeffs.H2O*(T_RH*tempscale).^(0:4)');
upperbound(kappafrac_IPturb) =.5;
    %R/M.H2O/(HCp_coeffs.H2O*lowerbound(T_IPturb).^(0:4)');

for i=feedflowdep(:,1)
    lowerbound(i)=lowerbound(i)*lowerbound(feedflow);
    upperbound(i)=upperbound(i)*upperbound(feedflow);
end

%TEMPSCALE
upperbound(outlettemp_set)=upperbound(outlettemp_set)./tempscale;
lowerbound(outlettemp_set)=lowerbound(outlettemp_set)./tempscale;
upperbound(idealoutlettemp_set)=upperbound(idealoutlettemp_set)./tempscale;
lowerbound(idealoutlettemp_set)=lowerbound(idealoutlettemp_set)./tempscale;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The following part is included mainly because there have been problems
% with too wide bounds, and is meant to deal with solver stability problems
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Tightening bounds around the best solution found so far
if tighterbounds>0
    if tighterbounds==2
        load boundsfile
    end
    a=0;
    b=.1;
    for i=v_all'
        lowerbound(i)=max((1—a)*x_opt(i)+a*lowerbound(i), x_opt(i)/(1+b));
        upperbound(i)=min((1—a)*x_opt(i)+a*upperbound(i), x_opt(i)*(1+b));
    end
end
% Saving the bounds in case they need to be tightened
save('boundsfile', 'lowerbound', 'upperbound')
```

F.2.12 Setting and calculating parameters for heat integration — hr.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% hr.m — setting and calculating parameters for heat integration
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% GDXMRW:       writegdx.m      writing data to gdx file
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Defining streams (index and name)
h1=length(uels)+1;      uels{h1}='h1';
h2=h1+1;                uels{h2}='h2';
h3=h2+1;                uels{h3}='h3';
h4=h3+1;                uels{h4}='exhaust';
c1=h4+1;                uels{c1}='c1';
LPE=c1+1;               uels{LPE}='LPE';
LPB=LPE+1;              uels{LPB}='LPB';
LPS=LPB+1;              uels{LPS}='LPS';
HPE=LPS+1;              uels{HPE}='HPE';
HPB=HPE+1;              uels{HPB}='HPB';
HPS=HPB+1;              uels{HPS}='HPS';
RH=HPS+1;               uels{RH}='RH';
preheat=RH+1;           uels{preheat}='preheat';
CCUreboiler=preheat+1;  uels{CCUreboiler}='CCUreboiler';
dummy=CCUreboiler+1;    uels{dummy}='dummy';

% Steam sets
hot=[h1 h2 h3 h4]';                     % Hot non—isothermal streams
cold=[c1 LPE LPS HPE HPS RH preheat]';  % Cold non—isothermal streams
isohot=[]';                             % Hot isothermal streams
isocold=[LPB HPB CCUreboiler]';         % Cold isothermal streams
iso=union(isohot', isocold')';          % All isothermal streams
noniso=union(hot', cold')';             % All non—isothermal streams
allhot=union(hot', isohot')';           % All hot streams
allcold=union(cold', isocold')';        % All cold streams
streams=union(iso', noniso')';          % All streams
st=length(streams);
%Set of pinch candidates
if relax==0
    pinchcandidates=[h1 h2 h3 h4 c1 LPE LPB HPE HPB RH preheat CCUreboiler]';
else
    pinchcandidates=[LPE CCUreboiler]';
end %LPE usually causes pinch (or CCUreboiler)

% Heat recovery approach temperature
HRAT=20;

% Limits for the exhaust outlet temperature
T_HRSG_out_lo=80;
T_HRSG_out_up=320;
```

```
% Given temperatures for the HRSG
T_water=32;
T_LPB=143.6;
T_HPB=324.6;
T_LPS=230;
T_HPS=560;
T_reheat=560;
T_CCUreboiler=120;

T_CH4supply=15;

% Calculating representative Cp values for the exhaust
T_exh1=630+C2K;
T_exh2=T_HRSG_out_lo+C2K;
Cp_H2O=HCp_coeffs.H2O./(1:5)*((T_exh1).^(1:5)'-(T_exh2).^(1:5)')/ ...
    (T_exh1-T_exh2);
Cp_CO2=HCp_coeffs.CO2./(1:5)*((T_exh1).^(1:5)'-(T_exh2).^(1:5)')/ ...
    (T_exh1-T_exh2);
Cp_O2=HCp_coeffs.O2./(1:5)*((T_exh1).^(1:5)'-(T_exh2).^(1:5)')/ ...
    (T_exh1-T_exh2);
Cp_N2=HCp_coeffs.N2./(1:5)*((T_exh1).^(1:5)'-(T_exh2).^(1:5)')/ ...
    (T_exh1-T_exh2);


Cp_RH=2.2318361545918672; %Calculated from HYSYS simulations
Cp_preheat=3.0748993810388652; %From HYSYS with feedtemp = 500 C

q_CCU=1500;
% Steam heating [kJ/h] per kg/h, calculated in HYSYS
q_LPE=470.418064578476;
q_LPB=2133.53929895280;
q_LPS=185.560334573640;
q_HPE=879.958181216821;
q_HPB=1197.71574220887;
q_HPS=816.054958821273;

w_LP=618.016839336527;
w_LPpump=.403;
w_HPpump=16.65;

if otherLP~=0
    if otherLP==8;
        T_LPB=170.4;
        T_LPS=230;
        q_LPE=586.4;
        q_LPB=2047;
        q_LPS=139.1;
        w_LP=696.2;
        w_LPpump=.9361;
    else
        disp(['Numbers for this pressure level LP = ' ...
                num2str(otherLP) ' not found.'])
    end
end
if otherHP~=0
    if otherHP==130;
        T_HPB=330.8;
        T_HPS=560;
        T_reheat=560;
        q_HPE=793.2;
        q_HPB=1135;
        q_HPS=828.3;
        Cp_RH=2.2381677011670842;
        if otherLP==8
            w_HPpump=18.14;
        else
            disp(['Numbers for HP = ' num2str(otherHP) ...
                    ' not available for LP = ' num2str(otherLP)])
        end
    else
        disp(['Numbers for this pressure level HP= ' ...
                num2str(otherHP) ' not found.'])
    end
end


% Bounds on temperatures and flows
T_in_lo=[   h1      T_refprod;
            h2      lowerbound(HTStemp2);
            h3      lowerbound(LTStemp2);
            h4      lowerbound(T_GT)*tempscale-C2K;
            c1      lowerbound(GTfueltemp);
            LPE     T_water;
            LPB     T_LPB;
            LPS     T_LPB;
            HPE     T_LPB;
            HPB     T_HPB
            HPS     T_HPB;
            RH      lowerbound(steamtemp);
            preheat T_CH4supply;
            CCUreboiler T_CCUreboiler];
T_in_up=[   h1      T_refprod;
            h2      upperbound(HTStemp2);
            h3      upperbound(LTStemp2);
            h4      upperbound(T_GT)*tempscale-C2K;
            c1      upperbound(GTfueltemp);
            LPE     T_water;
            LPB     T_LPB;
            LPS     T_LPB;
            HPE     T_LPB;
            HPB     T_HPB;
            HPS     T_HPB;
            RH      upperbound(steamtemp);
            preheat T_CH4supply;
            CCUreboiler T_CCUreboiler];
T_out_lo=[  h1      lowerbound(HTStemp);
            h2      lowerbound(LTStemp);
            h3      T_WRfeed;
            h4      T_HRSG_out_lo;
            c1      T_fuel-C2K;
            LPE     T_LPB;
            LPS     T_LPS;
            HPE     T_HPB;
            HPS     T_HPS;
            RH      T_reheat;
            preheat lowerbound(feedtemp)];
T_out_up=[  h1      upperbound(HTStemp);
            h2      upperbound(LTStemp);
            h3      T_WRfeed;
            h4      T_HRSG_out_up;
            c1      T_fuel-C2K;
            LPE     T_LPB;
            LPS     T_LPS;
            HPE     T_HPB;
            HPS     T_HPS;
            RH      T_reheat;
            preheat upperbound(feedtemp)];
flow_lo=[   h1      lowerbound(mCp_refprod);
            h2      lowerbound(mCp_HTSprod);
            h3      lowerbound(mCp_LTSprod);
            h4      CH4_in*lowerbound(feedflow);%Not so good...
            c1      lowerbound(mCp_GTfuel);
            preheat lowerbound(feedflow)*CH4_in*Cp_preheat;
            CCUreboiler lowerbound(CCU_CO2)*split.CO2(2)*q_CCU];
flow_up=[   h1      upperbound(mCp_refprod);
            h2      upperbound(mCp_HTSprod);
            h3      upperbound(mCp_LTSprod);
            h4      airflow+CH4_in*(1+upperbound(S2C)*M.H2O/M.CH4 + ...
                                    M.O2/M.CH4/airO2mass)*upperbound(feedflow);
            c1      upperbound(mCp_GTfuel);
            LPE     upperbound(W_GT);
            LPB     upperbound(W_GT);
            LPS     upperbound(W_GT);
            HPE     upperbound(W_GT);
            HPB     upperbound(W_GT);
            HPS     upperbound(W_GT);
            RH      upperbound(W_GT);
            preheat upperbound(feedflow)*CH4_in*Cp_preheat;
            CCUreboiler upperbound(CCU_CO2)*split.CO2(2)*q_CCU];

%Linking T_in variables with other temperature variables (or a constant)
T_inconn=[  h2  HTStemp2    1;
            h3  LTStemp2    1;
            h4  T_GT        tempscale;
            c1  GTfueltemp  1;
            RH  T_HPturb    tempscale];
T_inconnconst=[ h4          −C2K;
                LPE         T_water;
                LPB         T_LPB;
                LPS         T_LPB;
                HPE         T_LPB;
                HPB         T_HPB;
                HPS         T_HPB;
                RH          −C2K];
%Linking T_out variables with other temperature variables (or a constant)
T_outconn=[ h1  HTStemp     high(HTStemp)−low(HTStemp);
            h2  LTStemp     high(LTStemp)−low(LTStemp);
            preheat feedtemp high(feedtemp)−low(feedtemp)];
T_outconnconst=[h1          .5*(high(HTStemp)+low(HTStemp));
                h2          .5*(high(LTStemp)+low(LTStemp));
                LPE         T_LPB;
                LPS         T_LPS;
                HPE         T_HPB;
                HPS         T_HPS;
                RH          T_reheat;
                preheat     .5*(high(feedtemp)+low(feedtemp))];
%Linking flow variables from the heat recovery model with other flow
%variables
flowconn=[  h1      mCp_refprod 1;
            h2      mCp_HTSprod 1;
            h3      mCp_LTSprod 1;
            h4      exh_H2O     Cp_H2O;
            h4      exh_CO2     Cp_CO2;
            h4      exh_O2      Cp_O2;
            h4      exh_N2      Cp_N2;
            c1      mCp_GTfuel  1;
            HPE     HPsteamflow q_HPE/(T_HPB−T_LPB);
            RH      IPsteamflow Cp_RH;
            preheat feedflow    CH4_in*Cp_preheat;
            CCUreboiler CCU_CO2 split.CO2(2)*q_CCU];
%Linking flow variables from the heat recovery model with eachother
```

```
flowconn2=[ LPE LPB     q_LPE/(T_LPB−T_water)/q_LPB;
            LPE HPE     q_LPE/(T_LPB−T_water)/q_HPE*(T_HPB−T_LPB);
            LPB LPS     q_LPB/q_LPS*(T_LPS−T_LPB);
            HPE HPB     q_HPE/(T_HPB−T_LPB)/q_HPB;
            HPB HPS     q_HPB/q_HPS*(T_HPS−T_HPB)];

% Costs for utilities and streams
Q_Ccost=2.09793063613580e−3*objscale;
Q_Hcost=1*objscale;
flowcost=[  LPB −(w_LP*(1−STcondensercooling)−w_LPpump)/q_LPB;
            HPB (w_LPpump+w_HPpump)/q_HPB;
            RH  −w_LP*(1−STcondensercooling)/Cp_RH];
flowcost(:,2)=objscale*flowcost(:,2);

% Writing data to GDX file (hrsgconnected.gdx)
writegdx('hrsgconnected.gdx', ...
    'set',       'streams',      [streams; dummy], ...
    'set',       'hot',          [hot; dummy], ...
    'set',       'cold',         [cold; dummy], ...
    'set',       'noniso',       [noniso; dummy], ...
    'set',       'iso',          [iso; dummy], ...
    'set',       'isohot',       [isohot; dummy], ...
    'set',       'isocold',      [isocold; dummy], ...
    'set',       'allhot',       [allhot; dummy], ...
    'set',       'allcold',      [allcold; dummy], ...
    'set',       'pc',           [pinchcandidates; dummy], ...
    'parameter', 'T_in_lo',      T_in_lo, ...
    'parameter', 'T_in_up',      T_in_up, ...
    'parameter', 'T_out_lo',     T_out_lo, ...
    'parameter', 'T_out_up',     T_out_up, ...
    'parameter', 'flow_lo',      flow_lo, ...
    'parameter', 'flow_up',      flow_up, ...
    'parameter', 'T_inconn',     T_inconn, ...
    'parameter', 'T_outconn',    T_outconn, ...
    'parameter', 'flowconn',     flowconn, ...
    'parameter', 'flowconn2',    flowconn2, ...
    'parameter', 'T_outconnconst', T_outconnconst, ...
    'parameter', 'T_inconnconst', T_inconnconst, ...
    'parameter', 'HRAT',         HRAT, ...
    'parameter', 'Q_Hcost',      Q_Hcost, ...
    'parameter', 'Q_Ccost',      Q_Ccost, ...
    'parameter', 'flowcost',     flowcost, ...
    uels);
```

F.2.13 Reading solution from file — getsolution.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% getsolution.m - reading the solution of the model from irccoptimum.gdx
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% GDXMRW:   readgdx.m       reading data from.gdx files
%           sp2full.m       changing matrix from 'sparse' to 'full' format
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% Specifying the file to read from
filetoreadfrom='irccoptimum.gdx';

% Reading the solution from a gdx file produced by GAMS
x_opt=readgdx(filetoreadfrom,'x_opt');
x_opt(:,1)=v_all(x_opt(:,1));
x_opt=sp2full(x_opt,'parameter')';

try
    info(1)=sp2full(readgdx(filetoreadfrom,'z_opt'),'parameter')';
catch
    disp('No information about objective')
    info(1)=NaN;
end
try
info(2)=sp2full(readgdx(filetoreadfrom,'exectime'),'parameter')';
catch
    disp('No information about execution time')
    info(2)=-5;
end
try
    info(3)=sp2full(readgdx(filetoreadfrom,'iterations'),'parameter')';
catch
    disp('No information about iterations')
    info(3)=-5;
end
try
    info(4)=sp2full(readgdx(filetoreadfrom,'modelstatus'),'parameter')';
catch
    disp('No information about model status')
    info(4)=-5;
end

x_opt(length(x_opt)+1:length(uels))=0;
%Saving the solution if the model found one
if info(4)==1 || info(4)==2
    tid=datestr(now);
    save('A_matrix', 'x_opt', 'tid', '-append')
end
```

F.2.14 Post-processing and output of the solution — irccoutput.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% irccoutput.m — processing and displaying results (and reading some heat
%                 integration results)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% GDXMRW:   readgdx.m       reading data from.gdx files
%           sp2full.m       changing matrix from 'sparse' to 'full' format
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Names of structs
%   in  = Reformer feed
%   CSg = Cooled Syngas
%   SCC = Syngas to CCU
%   Con = Condensate
%   GTf = Gas Turbine fuel
%   CC  = Carbon Capture stream (CO2 to compression)
%   Ws  = Waste stream
% Unit: kmol/h

CH4_IN=CH4_in*x_opt(feedflow);

%Fixed values
in.CH4=CH4_IN/M.CH4;
CSg.O2=0/M.O2;
SCC.O2=0/M.O2;

% Values from x_opt
in.H2O=x_opt(S2C)*in.CH4;
CSg.CO=x_opt(CS_CO)/M.CO;
CSg.CO2=x_opt(CS_CO2)/M.CO2;
CSg.H2=x_opt(CS_H2)/M.H2;
SCC.CO=x_opt(CCU_CO)/M.CO;
SCC.CO2=x_opt(CCU_CO2)/M.CO2;
SCC.H2=x_opt(CCU_H2)/M.H2;
SCC.CH4=x_opt(CCU_CH4)/M.CH4;
SCC.H2O=x_opt(CCU_H2O)/M.H2O;
SCC.N2=x_opt(CCU_N2)/M.N2;

% Atom balance over reformer and shift reactors
CSg.CH4=in.CH4—CSg.CO—CSg.CO2;                  % Carbon balance
CSg.H2O=in.H2O—CSg.H2+2*(in.CH4—CSg.CH4);       % Hydrogen balance
in.O2=CSg.O2+CSg.CO2+.5*(CSg.CO+CSg.H2O—in.H2O);% Oxygen balance
in.N2=.79/.21*in.O2;                            % Air composition
```

```
CSg.N2=in.N2;                                   % Nitrogen balance


% Calculating the streams after the CCU split, and the condensate
for j=components
    i=char(j);
    % Condensate from water removal/separator
    eval(['Con.' i '=CSg.' i '-SCC.' i ';']);
    % Streams out of CCU
    eval(['GTf.' i '=SCC.' i '*split.' i '(1);']);
    eval(['CC.' i '=SCC.' i '*split.' i '(2);']);
    eval(['Ws.' i '=SCC.' i '*split.' i '(3);']);
end

% Displaying the solution
frmt='%35s %10.4g %12.4g %8s \n';
obj=sp2full(objective,'parameter');
if length(obj)<V
    obj(V)=0;
end
disp('Variable                              Obj.coeff.  Optimal value')
for i=v_all'
    if upperbound(i)==lowerbound(i)
        te='Fixed';
    elseif x_opt(i)==upperbound(i)
        te='UPPER';
    elseif x_opt(i)==lowerbound(i)
        te='lower';
    else
        te=num2str((x_opt(i)-lowerbound(i))/ ...
                (upperbound(i)-lowerbound(i)),'%6.4f');
    end
    fprintf(frmt, uels{i}, obj(i), x_opt(i),te);
end

% Displaying molar flow of components in GT fuel and carbon capture stream
GTf
CC
disp('[kmol/h]')

% Calculating and displaying excess air ratio
disp(['Excess air ratio:     ' ...
    num2str(x_opt(exh_O2)/(x_opt(comb_airflow)*airO2mass-x_opt(exh_O2)))...
    '    (' num2str(x_opt(exh_O2)/sum(x_opt(exhaust_set))*100) ...
    ' % O2 by mass in exhaust)']);

% Displaying results for turbines and compressors
disp('Turbines and compressors')
```

```
frmt='%16s %16s %16s %16s %16s \n';
fprintf(frmt,uels{trb_set})
frmt='%16.6g %16.6g %16.6g %16.6g %16.6g \n';
fprintf(frmt,x_opt(outlettemp_set)*tempscale-C2K)
fprintf(frmt,x_opt(p_out_set))
fprintf(frmt,x_opt(W_set))
fprintf(frmt,x_opt(kappafrac_set))
fprintf(frmt,sum(x_opt(exhaust_set)), x_opt(flow_ATRair_set), ...
        x_opt(HPsteamflow), x_opt(IPsteamflow))

% Reading some heat integration results
mCp_opt=sp2full(readgdx('hrsgopt.gdx','mCp'),'parameter')';
try
    Q_H_opt=readgdx('hrsgopt.gdx','Q_H')';
    if length(Q_H_opt)<1;Q_H_opt=0;end
catch
    Q_H_opt=0;
end
try
    Q_C_opt=readgdx('hrsgopt.gdx','Q_C')';
    if length(Q_C_opt)<1;Q_C_opt=0;end
catch
    Q_H_opt=0;
end

% Calculating power output and consumtions
W_air=w_air*airflow;
W_LP=w_LP*(mCp_opt(LPB)/q_LPB+x_opt(IPsteamflow));
W_CO2compr=w_CO2compr*split.CO2(2)*x_opt(CCU_CO2);
condenserpumpwork=STcondensercooling*(x_opt(W_HPturb)+x_opt(W_IPturb)+W_LP);
pumpwork=(w_HPpump+w_LPpump)*x_opt(HPsteamflow)+w_LPpump*mCp_opt(LPB)/q_LPB;
poweroutput = x_opt(W_GT)-W_air-x_opt(W_2ndcompr)-x_opt(W_extcompr)+ ...
  x_opt(W_HPturb) + x_opt(W_IPturb) + W_LP - x_opt(fuelcompr) - ...
  condenserpumpwork - pumpwork - W_CO2compr - ...
  Q_C_opt*Q_Ccost/objscale - Q_H_opt*Q_Hcost/objscale;
W_aux=condenserpumpwork + pumpwork +...
    Q_Ccost*Q_C_opt/objscale + Q_Hcost*Q_H_opt/objscale;
disp('POWER OUTPUT SUMMARY')
frmt='%36s %9.2f %4s \n';
unit='MW';
captureratio= x_opt(CCU_CO2)*split.CO2(2)/M.CO2*M.CH4./CH4_IN;
powertabledata=[x_opt(W_GT) -W_air -x_opt(W_2ndcompr) -x_opt(W_extcompr)...
    x_opt(W_HPturb) x_opt(W_IPturb) W_LP -W_CO2compr -x_opt(fuelcompr) ...
    -W_aux poweroutput LHV_CH4*CH4_IN poweroutput/(LHV_CH4*CH4_IN)*100 ...
    -Q_C_opt -Q_H_opt captureratio x_opt(feedpressure) x_opt(S2C)]';
fprintf(frmt, 'Gas turbine power (gross):', x_opt(W_GT), unit)
fprintf(frmt, 'Main air compressor work:', -W_air, unit)
fprintf(frmt, 'Second air compressor work:', -x_opt(W_2ndcompr), unit)
fprintf(frmt, 'Extra air compressor work:', -x_opt(W_extcompr), unit)
fprintf(frmt, 'HP steam turbine power:', x_opt(W_HPturb), unit)
fprintf(frmt, 'IP steam turbine power:', x_opt(W_IPturb), unit)
fprintf(frmt, 'LP steam turbine power:', W_LP, unit)
fprintf(frmt, 'CO2 compression:', -W_CO2compr, unit)
fprintf(frmt, 'GT fuel compression work:', -x_opt(fuelcompr), unit)
fprintf(frmt, 'Condenser cooling (work)', -condenserpumpwork, unit)
fprintf(frmt, 'Pumps for LP and HP water:', -pumpwork, unit)
fprintf(frmt, 'Cooling water pumps:', -Q_Ccost*Q_C_opt/objscale, unit)
fprintf(frmt, 'External heating:', -Q_Hcost*Q_H_opt/objscale, unit)
disp('------------------------------------------------------------------')
fprintf(frmt, 'Net power output:', poweroutput, unit)
disp('==================================================================')
fprintf(frmt, 'Energy input (CH4 LHV):', LHV_CH4*CH4_IN, unit)
disp('==================================================================')
fprintf(frmt, 'Efficiency:', poweroutput/(LHV_CH4*CH4_IN)*100, '%')
disp('==================================================================')
fprintf(frmt, 'Cold utilities (external cooling):', -Q_C_opt, unit)
fprintf(frmt, 'Hot utilities (external heating):', -Q_H_opt, unit)
disp('==================================================================')
fprintf(frmt, 'CO2 capture ratio:', captureratio*100, '%')
fprintf(frmt, 'ATR pressure:', x_opt(feedpressure), 'bar')
fprintf(frmt, 'Steam to carbon ratio:', x_opt(S2C), '')

disp(info(1));
modelstat={'Optimal', 'Locally Optimal', 'Unbounded', 'Infeasible', ...
        'Locally Infeasible', 'Intermediate Infeasible', ...
        'Intermediate Nonoptimal', 'Integer Solution', ...
        'Intermediate Non-Integer', 'Integer Infeasible', ...
        'Licencing Problems - No Solution', 'Error Unknown', ...
        'Error No Solution', 'No Solution Returned', 'Solved Unique', ...
        'Solved', 'Solved Singular', 'Unbounded - No Solution', ...
        'Infeasible - No Solution'};
disp(['Model status ' num2str(info(4)) ' - ' modelstat{info(4)}]);
```

F.2.15 Post-processing and output of heat integration results — cascade.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% cascade.m - analyzing heat integration results
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Functions and scripts used:
% GDXMRW:    readgdx.m       reading data from.gdx files
%            sp2full.m       changing matrix from 'sparse' to 'full' format
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Reading the solution
try
    filetoreadfrom='hrsgopt.gdx';
    T_in=sp2full(readgdx(filetoreadfrom,'Tin'),'parameter')';
    T_out=sp2full(readgdx(filetoreadfrom,'Tout'),'parameter')';
    mCp=sp2full(readgdx(filetoreadfrom,'mCp'),'parameter')';
    HRAT=sp2full(readgdx(filetoreadfrom,'HRAT'),'parameter');
    cost=sp2full(readgdx(filetoreadfrom,'z'),'parameter');
    if length(mCp)<streams(st)
        mCp(streams(st))=0;
    end
catch
    disp(['COULD NOT READ FROM ' filetoreadfrom])
end

% Adjusting to get right dimension
if length(T_in_up)<streams(st)
    T_in_up=sp2full(T_in_up,'parameter')';
    T_in_lo=sp2full(T_in_lo,'parameter')';
    T_out_lo=sp2full(T_out_lo,'parameter')';
    flowcost=sp2full(flowcost,'parameter')';
    file='ircc.gdx';
    if length(T_in)<streams(st)
        T_in(streams(st))=0;
    end
    if length(T_in_lo)<streams(st)
        T_in_lo(streams(st))=0;
    end
    if length(T_in_up)<streams(st)
        T_in_up(streams(st))=0;
    end
    if length(T_out)<streams(st)
        T_out(streams(st))=0;
    end
    if length(T_out_lo)<streams(st)
        T_out_lo(streams(st))=0;
    end
    if length(flowcost)<streams(st)
        flowcost(streams(st))=0;
    end
end
T_out(iso)=T_in(iso);
T_out_lo(iso)=T_in_lo(iso);
% Adjusting for HRAT
T_in(allcold)=T_in(allcold)+HRAT;
T_out(cold)=T_out(cold)+HRAT;

% Calculating heating and cooling for each stream for each temperature
% interval
T=unique([T_in(streams) T_out(noniso)]); % All unique temperatures
t=length(T)-1;
Qcold=zeros(1,t);
Qhot=Qcold;
Qisocold=Qcold;
Qisohot=Qisocold;
for i=1:t;
    Qcold(i)=sum((T_in(cold)<=T(i)).*(T_out(cold)>=T(i+1))*mCp(cold)')*...
        (T(i+1)-T(i));
    Qhot(i)=sum((T_in(hot)>=T(i+1)).*(T_out(hot)<=T(i))*mCp(hot)'*...
        (T(i+1)-T(i)));
    Qisocold(i)=sum((T_in(isocold)==T(i))*mCp(isocold)');
    Qisohot(i)=sum((T_in(isohot)==T(i))*mCp(isohot)');
end
% Calculating residuals
demand=Qhot-Qcold+Qisohot-Qisocold;
Qisocold(t+1)=sum((T_in(isocold)==T(t+1))*mCp(isocold)');
Qisohot(t+1)=sum((T_in(isohot)==T(t+1))*mCp(isohot)');
demand(t+1)=Qisohot(t+1)-Qisocold(t+1);
for i=t:-1:1
    demand(i)=demand(i)+demand(i+1);
end
% Finding actual utility usage
[Q_H, p]=max([-demand 0]);
Q_C=(T_in(noniso)-T_out(noniso))*mCp(noniso)'+Q_H + sum(mCp(isohot)) - ...
    sum(mCp(isocold));
% Finding the pinch temperature
T_pinch=[T(p) T(p)-HRAT];
Q=demand+Q_H;
demand=Qhot-Qcold;

% Calculating and drawing composite curves and grand composite curve
coldcomposite=zeros(1,2*t+2);
coldcomposite(1)=Q_C;
hotcomposite=zeros(1,2*t+2);
Temps=zeros(1,2*t+2);
for i=1:t
    Q(2*i)=Q(2*i-1)+Qisocold(i)-Qisohot(i);
    Temps(2*i-1)=T(i);
    Q(2*i+1)=Q(2*i)+Qcold(i)-Qhot(i);
    Temps(2*i)=T(i);
    coldcomposite(2*i)=coldcomposite(2*i-1)+Qisocold(i);
    coldcomposite(2*i+1)=coldcomposite(2*i)+Qcold(i);
    hotcomposite(2*i)=hotcomposite(2*i-1)+Qisohot(i);
    hotcomposite(2*i+1)=hotcomposite(2*i)+Qhot(i);
end
Temps(2*t+1)=T(t+1);
```

```
Q(2*t+2)=Q(2*t+1)+Qisocold(t+1)-Qisohot(t+1);
coldcomposite(2*t+2)=coldcomposite(2*t+1)+Qisocold(t+1);
hotcomposite(2*t+2)=coldcomposite(2*t+1)+Qisohot(t+1);
Temps(2*t+2)=T(t+1);
grandcomposite=coldcomposite-hotcomposite;
figure(1)
plot(grandcomposite,Temps-.5*HRAT,'o-')
xlabel('Q [MW]')
ylabel('T [^{o}C]')
xlim([min(grandcomposite) max(grandcomposite)])
figure(2)
frmtc='b-';
frmth='r-';
evalstring='plot(';
for i=2:2*t+2;
    if coldcomposite(i)~=coldcomposite(i-1)
        evalstring=[evalstring '[' ...
            num2str(coldcomposite(i-1)) ',' num2str(coldcomposite(i)) ...
            '],[' num2str(Temps(i-1)-HRAT) ',' num2str(Temps(i)-HRAT) ...
            '],frmtc,'];
    end
    if hotcomposite(i)~=hotcomposite(i-1)
        evalstring=[evalstring '[' ...
            num2str(hotcomposite(i-1)) ',' num2str(hotcomposite(i)) ...
            '],[' num2str(Temps(i-1)) ',' num2str(Temps(i)) '],frmth,'];
    end
end
evalstring(length(evalstring))=')';
eval(evalstring)
xlabel('Q [MW]')
ylabel('T [^{o}C]')
xlim([min([coldcomposite hotcomposite]) max([coldcomposite hotcomposite])])

% Calculating data for a detailed heat cascade (could have been based on
% previous code)
ho=length(hot);
co=length(cold);
hc=max([ho co]);
Qcold=zeros(hc,t);
Qhot=Qcold;
for i=1:t;
    for j=1:co
        Qcold(j,i)=(T_in(cold(j))<=T(i)).*(T_out(cold(j))>=T(i+1))*...
            mCp(cold(j))*(T(i+1)-T(i));
    end
    for j=1:ho
        Qhot(j,i)=(T_in(hot(j))>=T(i+1)).*(T_out(hot(j))<=T(i))*...
            mCp(hot(j))*(T(i+1)-T(i));
    end
end
% Re-adjusting for HRAT
T_in(allcold)=T_in(allcold)-HRAT;
T_out(cold)=T_out(cold)-HRAT;
% "Drawing" a detailed heat cascade
frmt='%9s'; frmtl='%-9s';
bl=9;
disp(['Q_H = ' num2str(Q_H)])
disp([num2str(Qisohot(t+1)) blanks(2*bl+7) num2str(-Qisocold(t+1))])
if t+1==p
    disp('         Pinch')
end
for i=t:-1:1
    disp(['   ' sprintf(frmt,num2str(T(i+1))) '_________' ...
        sprintf(frmtl,num2str(T(i+1)-HRAT))])
    disp([blanks(bl+2) '|' sprintf(frmt,num2str(demand(i))) '|'])
    for j=1:hc
        disp([sprintf(frmt,num2str(Qhot(j,i))) ' -|' blanks(bl) '|- ' ...
            sprintf(frmtl,num2str(Qcold(j,i)))])
    end
    disp([blanks(bl+2) '|_________|'])
    disp([num2str(Qisohot(i)) blanks(2*bl+7) num2str(-Qisocold(i))])
    if i==p
        disp([blanks(bl) 'Pinch'])
    end
end
disp(['   ' sprintf(frmt,num2str(T(1))) blanks(bl) ...
    sprintf(frmtl,num2str(T(1)-HRAT))])
disp(['Q_C = ' num2str(Q_C)])

% Displaying stream data
Qflow=zeros(1,st);
Qflow(noniso)=(T_in(noniso)-T_out(noniso)).*mCp(noniso);
Qflow(isohot) =mCp(isohot);
Qflow(isocold)=-mCp(isocold);
frmt='%11s %10.4g %10.4g %10.4g %12.4g %10.4g %10.4g \n';
disp('     Stream       T_in       T_out        mCp     Qflow')
for i=streams'
    fprintf(frmt, uels{i}, T_in(i), T_out(i), mCp(i), Qflow(i), ...
        T_in_up(i)-T_in(i), T_out(i)-T_out_lo(i));
end
disp(['Cost: ' num2str(mCp*flowcost'+Q_H*Q_Hcost+Q_C*Q_Ccost) ...
    ' (calculated from "optimal" stream data)'])
if strncmp('hrsg', file, 4)
    disp(['Cost: ' num2str(cost) ' (optimal objective value from ' ...
        file ')'])
end
```