NTNU – Trondheim
Norwegian University of
Science and Technology

Optimal Operation of Parallel Heat Exchanger Networks

Stian Aaltvedt

Chemical Engineering and Biotechnology
Submission date: June 2013
Supervisor: Sigurd Skogestad, IKP
Co-supervisor: Johannes Jäschke, IKP

Norwegian University of Science and Technology
Department of Chemical Engineering

Abstract

Optimal operation of parallel heat exchanger networks is desirable for many processes aiming to achieve increased supply and potentially higher profit. The aim is to control the final outlet temperature within a certain range, which in many cases includes a trade off between maximum outlet temperature and minimum operating costs.

The goal with this study has been to investigate the performance of the self-optimizing Jäschke temperature control variable, proposed by post doctor Johannes Jäschke. The Jäschke temperature approach seeks to achieve near optimal operation of parallel heat exchanger networks, exclusively by manipulation of the bypass selection - only based on simple temperature measurements. The method has been demonstrated for several different cases and investigated both at steady state and dynamically.

For balanced heat exchanger networks, with evenly distributed hot stream heat capacities throughout the network, the Jäschke temperature showed good performance for all cases studied. The simulations revealed satisfactory disturbance rejection and very close to optimal operation. For cases suffering a more uneven heat capacity distribution, the method did not give near optimal operation. Also, exposed to major, non-realistic disturbances the Jäschke temperature control configuration gave poor performance due to singularities in the control variable when certain temperatures achieved equal values. In the presence of such incidents, a modified control variable was implemented by re-writing the expression controlling the Jäschke temperatures to a denomiator-free form. This gave slightly better performance and was concluded to operate the system satisfactory.

Sammendrag

Optimal drift av parallelle varmevekslernettverk er ønskelig for mange prosesser med mål om økt etterspørsel og potensielt større profitt. Målet er å kontrollere utgangstemperaturen innenfor et bestemt intervall, som i mange sammenhenger er en balanse mellom høyest mulig utgangstemperatur og lavest mulig driftskostnader.

Målet med denne studien har vært å undersøke ytelsen til den selv-optimaliserende Jäschke temperatur reguleringsvariabelen, forslått av postdoktor Johannes Jäschke. Jäschke temperatur-metoden forsøker å oppnå en drift så nært optimum som mulig, kun ved justering av strømsplitten – utelukkende basert på enkle temperaturmålinger. Metoden har blitt demonstrert for flere ulike tilfeller av varmevekslernettverk og blitt undersøkt både i stabil tilstand og dynamisk.

For balanserte varmevekslernettverk med jevn fordeling av de ulike varmestrømmenes varmekapasitet, viste Jäschke temperatur-konfigurasjonen god ytelse for alle undersøkte tilfeller av varmevekslernettverk. Simuleringene gav god forstyrrelsesavvisning og svært nær optimal drift. For tilfeller hvor varmekapasitetene var ujevnt fordelt i varmevekslernettverket, gav ikke metoden nær optimal drift. Utsatt for større og mer urealistiske forstyrrelser viste Jäschke temperatur-metoden dårlig ytelse grunnet singulariteter i reguleringsvariabelen i tilfeller hvor enkelte temperaturer fikk samme verdi. I slike tilfeller ble reguleringsvariablene modifisert ved å unnlate bruken av brøk i ligningen. Dette gav bedre ytelse og ble konkludert til å gi god drift av systemet.

1 Preface

This master thesis was completed during the spring semester of 2013, and was the very final compulsory part of the 5 year integrated master program in Chemical Engineering and Biotechnology at Norwegian University of Science and Technology (NTNU).

The task of this thesis has applied to me as very interesting, and I feel honored of having the opportunity to work together with Johannes Jäschke on his patent application. It has been a great factor of motivation, knowing that my work has, to some extent, contributed to his research on one of todays most important global concerns of energy saving. I would like to thank Johannes for being so helpful and inspirational. I have learned a lot from working with Johannes, you have given me a solid lesson on heat exchange and self-optimizing control. Additionally, I have become way more experienced with MATLAB and LaTeX because of you. Thank you!

A huge thanks also goes out to Sigurd Skogestad, my main supervisor. You have an incredible high level of knowledge and skills. I admire your ability to always have such a good overview of the whole porcess-systems engineering group and each group members individual work. Thank you for being a very good and unique team leader.

Last but not least I would like to thank all the friends that I've made during my years at NTNU. You all certainly made the time in Trondheim very memorable!

I declare that this is an independent work according to the exam regulations of the Norwegian University of Science and Technology (NTNU).

Date and signature

7/6 - 2013

. . . .

Contents

List of Figures

List of Tables

List of Symbols

Symbol	Explanation	Unit
ΔT_{AM}	Arithmetic Mean Temperature Difference	[°C]
ΔT_{LM}	Logarithmic Mean Temperature Difference	[°C]
ΔT_{min}	Minimum temperature difference at heat exchanger ends	[°C]
ΔT_{UN}	Underwood's approximated temperature difference	[°C]
ϵ	Effectiveness of a heat exchanger	[-]
θ	Temperature difference at heat exchanger ends	[°C]
θ	Transport delay	[sec]
ρ	Density	[$^{kg}/_{m^3}$]
τ_f	Filter time constant	[sec]
τ_I	PI controller time constant	[sec]
A	Heat exchanger area	[m^2]
c	Control variable	[°C]
c_{mod}	Modified control variable	[°C^4]
$C_{min/max}$	smallest/biggest heat capacity rate	[$^{kW}/_{°C}$]
C_p	Heat capacity	[$^{kW}/_{kg°C}$]
C_r	Heat capacity ratio	[-]
$\bar{c}$	Steady state value for controller	[°C]
d	Disturbance	[various]
e	Error signal to controller	[°C]
g	Equality constraint vector	[various]
h	Inequality constraint vector	°C]
h	Heat transfer coefficient	[$^{kW}/_{°Cm^2}$]
J	Cost function	[$-\$$]
$JT_{i,j}$	Jäschke temperature for heat exchanger i on branch j	[°C]

K_c	PI controller gain	$[^{\circ}C/_{kg/s}]$
K_f	Filter gain	$[^{\circ}C/_{kg/s}]$
L	Loss	[°C]
m	Mass flow	$[^{kg}/_{s}]$
M	Number of heat exchangers on the lower branch	[-]
N	Number of heat exchangers on the upper branch	[-]
NTU	Number of Transit Units	[-]
$P_{i,j}$	Price constant heat exchanger i on branch j	$[^{\$}/_{kW}]$
Q	Heat	$[kW]$
R	Model order in dynamic calculations	[-]
t	Time	[sec]
T	Temperature	[°C]
u	Degrees of Freedom (DOF)	[-]
u	Stream split to upper branch	[-]
u_t	Manipulated variables	[various]
U	Overall heat transfer coefficient	$[^{kW}/_{^{\circ}C m^2}]$
V	Volume	$[m^3]$
w	Heat capacity rate	$[^{kW}/_{^{\circ}C}]$
x	State variables	[various]

2 Introduction

In a modern industrial and technological world where energy and power consumption serves as one of the most essential global concerns, there are enhanced requirements for all production processes to be sustainable to future generations of our planet. In the chemical industry, especially including todays great petroleum activity, an overall goal of using the available energy sources in the most efficient way can be satisfied by optimal heat recovery from different parts of a given process (Zhang, Yang, Pan & Gao 2011).

The need for research and development in this industry is one very important aspect of the issues associated with energy efficient processes. The trade off between a business goal seeking increased supply in an attempt to generate large profit margins - and still obey the sustainable methods to meet the energy demands - is rather complex (Zhang et al. 2011). Good heat recovery from a given process can be achieved through effective use of heat exchangers. Often, heat exchangers are combined in a heat exchanger network to distribute the available hot streams in the most effective way (Sinnott & Towler 2009). A simplified general heat exchanger network with N heat exchangers in series on the upper branch and M in series on the lower branch is presented in Figure 2.1.

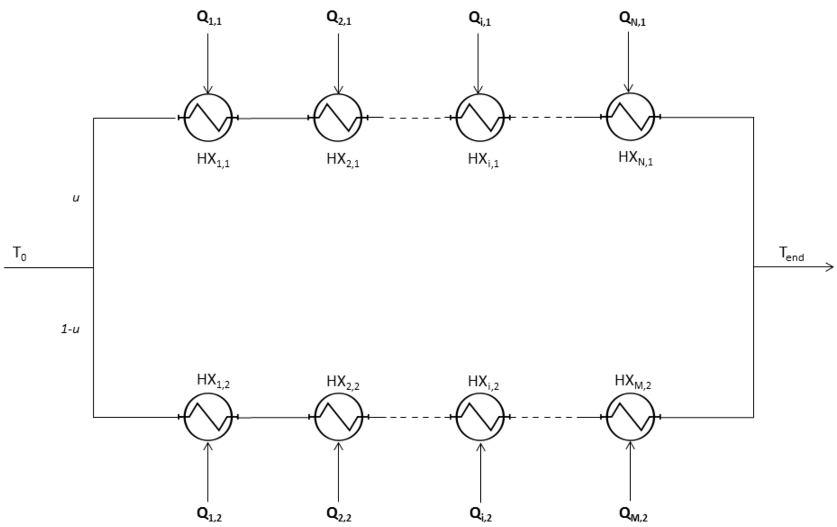


Figure 2.1: A simplified general heat exchanger network with N heat exchanger in series on the upper branch (branch 1) and M heat exchangers in series on the lower branch (branch 2)

A heat exchanger network should be designed allowing for the best possible heat integration. At the same time, operating with reasonable heat exchanger duties is necessary in order to minimize the operation costs (Jensen & Skogestad 2008). Marselle, Morari & Rudd (Marselle, Morari & Rudd 1982) were some of the first to discuss optimal operation problems of heat exchanger networks, where simultaneous regulation and optimization were considered as a possible control configuration. Since that, among other publications, Mathisen, Morari & Skogestad (Mathisen, Morari & Skogestad 1994*b*) have proposed a method to operate heat exchanger networks that also minimizes utility consumption. Recently, Jäschke (Jaeschke 2012) derived the self-optimizing Jäschke temperature variable for operation of heat exchanger networks. According to Skogestad (Skogestad 2004), the use of self-optimizing control does not require simultaneous regulation and optimization when disturbances are present. Additionally, the method proposed by Jäschke includes utility costs, hence operation is also subject to each heat exchangers associated cost. The self-optimizing Jäschke temperature variable seeks to operate certain heat exchanger networks with the split u (see Figure 2.1) as the only manipulated variable. The method is claimed to achieve near-optimal operation with constant setpoints for the control variable (Jaeschke 2012). Usually operation of heat exchanger networks involves several different manipulated variables (e.g. bypass selection and hot stream flows), relying on both temperature and flow measurements (González & Marchetti 2005). With the Jäschke temperature, only temperature measurements are needed. Compared to flow measurements, temperature measurements are cheaper, faster and more exact which makes the control structure proposed by Jäschke easy to implement and use.

This study investigates optimal operation of heat exchanger networks. The aim is to continue the work done on the Jäschke temperature (Jaeschke 2012) in the specialization project (Aaltvedt 2012). The specialization project investigated optimal design and optimal steady state operation of parallel heat exchanger networks limited by three heat exchangers in series. Recently, Jäschke proposed a general equation applying for N heat exchangers in series (Jaeschke 2012), which, among other cases, will be investigated in this study.

During the progress of this study the Jäschke temperature control configuration is considered a patent application. The overall goal with this study is therefore to search for and investigate cases where the Jäschke temperature gives non-optimal operation and/or poor control. First, a steady state analysis is done. Operation using the Jäschke temperature control variable is compared to optimal operation for several different heat exchanger networks. The downstream temperature loss associated with Jäschke temperature operation is investigated for each case. The Jäschke temperature will also be tested in the presence of measurement errors. Secondly, a dynamic analysis is done. The goal with this analysis is to relieve any poor control resulting from the Jäschke temperature in the presence of different disturbances, where temperature fluctuations will serve as the main source for disturbance. In addition, for a heat exchanger network of two heat exchanger in series parallel to one heat exchanger, a comprehensive analysis is done for an extreme case where a decreasing hot stream temperature in one heat exchanger gives a cooling effect.

3 Heat Exchanger Modelling

With heat exchange the overall goal is to transfer heat from a hot source to a cold source (Skogestad 2003*a*). The heat transfer process can be carried out by three different mechanisms (Geankoplis 2003):

- Conduction heat transfer
- Convection heat transfer
- Radiation heat transfer

For most industrial processes where heat is transfered from one fluid to another through a solid wall, conduction is the main mechanism for heat transfer (Geankoplis 2003). This heat transfer is conducted in a heat exchanger, where the cold fluid is to be heated by the hot fluid. The most effective way of heat transfer is done through a *counter current* heat exchanger (Geankoplis 2003) shown in Figure 3.1. Here, Q $[kW]$ represents the transfered heat and T_h and T_c [°C] are the temperatures of the hot and cold stream, respectively.

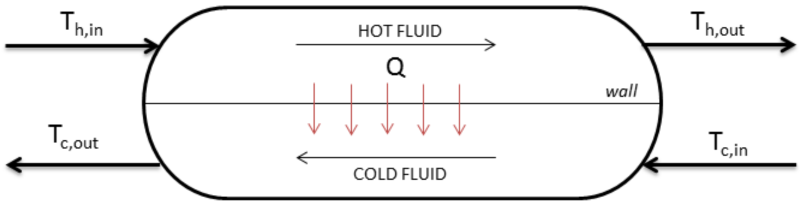


Figure 3.1: The counter current heat exchanger

3.1 Steady state model

In an ideal counter current heat exchanger the outlet hot stream temperature equals the entering cold stream temperature (Bartlett 1996). That is, $T_{h,out} = T_{c,in}$ in Figure 3.1, and the heat exchangers effect is said to be maximized. For an ideal counter current heat exchanger constant inlet temperatures ($T_{h,in}$ and $T_{c,in}$ in Figure 3.1) can be assumed at steady state. The heat Q transfered form hot to cold side can be expressed by the heat exchanger equation (Skogestad 2003*a*)

$$Q = UA\Delta T_{LM} \tag{3.1}$$

Where U is the over all heat transfer coefficient [$^{kW}/_{^\circ C m^2}$] and A is the total area of the heat exchanger [m^2]. For many ideal cases the the overall heat transfer coefficient U can be written as (Incorpera & DeWitt 2007)

$$U = \frac{h_c h_h}{h_c + h_h} \tag{3.2}$$

Here, h_c and h_h represents the heat transfer coefficients for cold and hot fluid, respectively. The ΔT_{LM} term is the Logarithmic Mean Temperature Difference (LMTD). For a counter current heat exchanger it is given as (Skogestad 2003*a*)

$$\Delta T_{LM} = \frac{(T_{h,in} - T_{c,out}) - (T_{h,out} - T_{c,in})}{\ln(\frac{T_{h,in} - T_{c,out}}{T_{h,out} - T_{c,in}})} = \frac{\theta_1 - \theta_2}{\ln(\frac{\theta_1}{\theta_2})} \tag{3.3}$$

The energy balance for the ideal counter current heat exchanger in Figure 3.1 is (Skogestad 2003*a*)

$$Q = m_c Cp_c (T_{c,out} - T_{c,in}) \tag{3.4}$$

$$Q = m_h Cp_h (T_{h,in} - T_{h,out}) \tag{3.5}$$

Cp_c, Cp_h and m_c, m_h represents the heat capacities [$^{kW}/_{kg^\circ C}$] and the mass flows [$^{kg}/_{s}$] for the cold and hot fluid, respectively. Since this is a steady state model, the heat capacities can be assumed to be constant. The product mCp is called the *heat capacity flow rate* (Sinnott & Towler 2009), given in [$^{kW}/_{^\circ C}$].

$$m_c Cp_c = w_c \tag{3.6}$$

$$m_h Cp_h = w_h \tag{3.7}$$

From the principle of energy- and mass conservation the correlation between Equation 3.1, 3.4 and 3.5 is

$$Q = UA\Delta T_{LM} = w_c(T_{c,out} - T_{c,in}) = w_h(T_{h,in} - T_{h,out}) \tag{3.8}$$

3.1.1 Approximations and Transformations

Associated with steady state is the already mentioned assumptions of constant heat capacities and constant inlet hot and cold stream temperatures. For the steady state investigation the mass flows of the cold stream and every hot stream will also be treated as constant. In addition, single phase flow for hot streams, that is no phase transfer during heat transfer, will also be assumed in the steady state analysis.

Approximation of the Logarithmic Mean Temperature Difference (LMTD)

Application of the LMTD equation might lead to numerical challenges. If the LMTD were to be applied on a transient in which the temperature difference had different signs on the two sides of the heat exchanger, the argument to the logarithmic function would be negative, which is not allowable (Kay & Nedderman 1985). Skogestad (Skogestad 2003*a*) states that the Logarithmic Mean Temperature Difference (LMTD) in Equation 3.3 can be approximated to an Arithmetic Mean Temperature Difference (AMTD). If $^1/_{1.4} < {}^{\theta_1}/_{\theta_2} < 1.4$, i.e. the temperature difference between the cold and hot side is fairly constant, the error of using AMTD instead of LMTD is less than 1%. The arithmetic mean temperature difference, AMTD is given as (Skogestad 2003*a*)

$$\Delta T_{AM} = \frac{\theta_1 + \theta_2}{2} \tag{3.9}$$

Another and more robust approximation to the LMTD is made by Underwood (Underwood 1933) and is given as

$$\Delta T_{UN} = \left(\frac{\theta_1^{\frac{1}{3}} + \theta_2^{\frac{1}{3}}}{2} \right)^3 \tag{3.10}$$

To avoid the numerical issues associated with the LMTD and due to the robustness of the approximation, the Underwood approximation (Underwood 1933) will be used in parts of the steady state simulations where the LMTD needs to be approximated.

Transformation of the Model Equations to the NTU Method

The Number of Transfer Units (NTU) Method is used to calculate the steady state rate of heat transfer in heat exchangers where there is insufficient information to calculate the Logarithmic Mean Temperature Difference (LMTD) (Incorpera & DeWitt 2007). If both the heat exchanger area and the hot and cold mass flows together with the respective inlet temperatures are known, the NTU method can be applied for simulations of heat exchangers. The NTU method calculates the effectiveness of a heat exchanger based on the flow with the limiting heat capacity. The energy equations are the same as the ones given in Section 3, only expressed in a different way. The number of transfer units is defined as (Incorpera & DeWitt 2007)

$$NTU = \frac{UA}{C_{min}} \tag{3.11}$$

Where C_{min} is the smallest heat capacity rate, that is $C_{min} = min\{w_c, w_h\}$. For counter current flow, the effectiveness ε is given by (Incorpera & DeWitt 2007)

$$\varepsilon = \frac{1 - \exp(-NTU(1 - C_r))}{1 - C_r \exp(-NTU(1 - C_r))} \tag{3.12}$$

Here, C_r is defined as the ratio $\frac{C_{min}}{C_{max}}$ and $C_{max} = max\{w_c, w_h\}$. If C_r in Equation 3.12 becomes singular the equation can not be used. In that case, for counter current flow, ε becomes (Incorpera & DeWitt 2007)

$$\varepsilon = \frac{NTU}{1 + NTU} \tag{3.13}$$

From this, the hot and cold outlet temperatures from a heat exchanger can be found

$$T_{h,out} = (1 - C_r\varepsilon)T_{h,in} + C_r\varepsilon T_{c,in} \tag{3.14}$$

$$T_{c,out} = \varepsilon T_{h,in} + (1 - \varepsilon)T_{c,in} \tag{3.15}$$

According to these equations, the NTU-method yields a linear relationship between the inlet temperatures and the resulting outlet temperatures. However, the outlet temperature is nonlinearly dependent on the flow rate.

3.2 Dynamic Model

Dynamic models are needed to assess controllability of heat exchangers and heat exchanger networks (Mathisen, Morari & Skogestad 1994*a*). In order to verify whether the control configuration proposed by Jäschke (Jaeschke 2012) gives satisfactory control, dynamic simulations and control behavior of heat exchanger networks should also be taken into account.

The dynamic analysis includes simulations present to disturbances. For these parts the assumptions of constant cold and hot stream temperatures will no be longer valid. The cold stream mass flow will also serve as a disturbance and will thereby neither be treated as constant. However, single phase flow will still be assumed.

3.2.1 The Mixed Tanks in Series Model

Wolff, Mathisen and Skogestad (Wolff, Mathisen & Skogestad 1991) states that a heat exchanger can be approximated as a lumped model and thus be expressed as *mixed tanks in series*. Modeling the temperature development for a given stream in a heat exchanger as mixed tanks in series is desirable because of the simple expression that result. A modified version of this lumped model is presented in Figure 3.2 (Wolff et al. 1991)

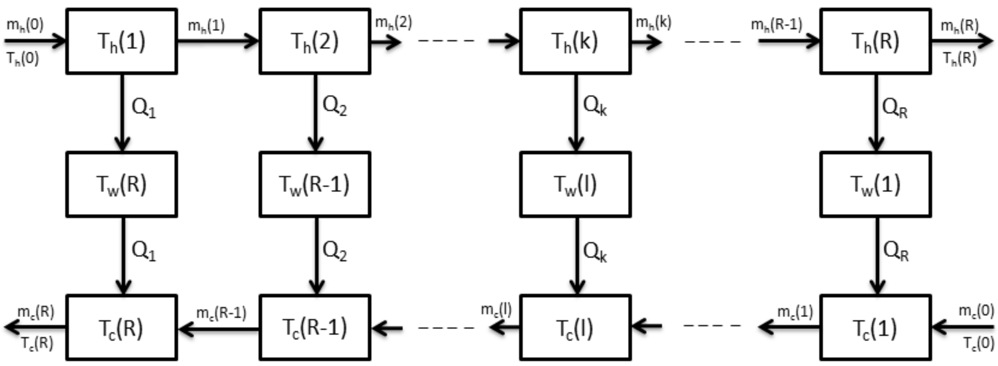


Figure 3.2: The mixed tanks heat exchanger model, modified

Here, $m_h(0)$ and $T_h(0)$, $m_c(0)$ and $T_c(0)$ is the inlet mass flow and temperature on hot and cold side, respectively. $T_h(k)$ and $T_c(l)$ is the hot stream and cold stream outlet temperatures in tank k and l, respectively. T_w is the wall temperature and Q is the transfered heat in each tank. The lumped model consists of R equal mixing tanks, in which the total heat exchanger area A and volume V is assumed to be equally distributed throughout the R tanks. Negligible heat loss and pressure drop, constant heat capacity and fluid density are also assumed. Relevant heat exchanger data are given in Table B.1 in Appendix B From Mathisen et al. (Mathisen et al. 1994*a*), the differential equations resulting from the energy balance are

$$\frac{dT_h(k)}{dt} = \left(T_h(k-1) - T_h(k) - \frac{h_h A}{w_h R}\Delta T_h(k)\right)\frac{m_h R}{\rho_h V_h} \tag{3.16}$$

$$\frac{dT_w(l)}{dt} = ((h_h \Delta T_{w,h}(l) - h_c \Delta T_{w,c}(l))\frac{A}{\rho_w c_{p,w} V_w} \tag{3.17}$$

$$\frac{dT_c(l)}{dt} = \left(T_c(l-1) - T_c(l) - \frac{h_c A}{w_c R}\Delta T_c(l)\right)\frac{m_c R}{\rho_c V_c} \tag{3.18}$$

Where the subscript c, h and w denotes cold fluid, hot fluid and wall, respectively. Further, h is the heat transfer coefficient for each fluid, given in $[{}^{kW}/_{^\circ Cm^2}]$, ρ is density given in $[{}^{kg}/_{m^3}]$, R is the number of cells, V is volume given in $[m^3]$ and t is time in [sec]. A complete derivation can be found in Mathisen et al. (Mathisen et al. 1994*a*). According to the authors, a model order of $R > 6$ is typical to ensure satisfactory prediction. In this study a model order of 10 is used.

4 Optimization of Heat Exchanger Networks

For many processes, the overall goal is to maximize the income of the plant (Jensen & Skogestad 2008). In a perfect world, optimal heat-transfer performance would be achieved without compromise. Systems would require minimal heat exchanger area, with minimal cost associated with heat exchange equipment. In the real world, however, economic losses can begin as early as the preliminary design phase. The design must accommodate uncertainties and assumptions, adding to the projects capital investment and operating costs (Gramble 2006). Out of several factors, profitability associated with heat exchangers relies on the effectiveness of the heat transfer. However, there are two contradictory factors for cost-effective heat transfer. Obtaining the highest possible outlet temperature is desirable regarding the final product quality and the potential profit. At the same time, operating with reasonable heat exchanger duties is an equally important factor for keeping the operation costs low (Jensen & Skogestad 2008). Optimization of heat exchanger networks are based on an objective function J that includes capital and operation costs (Jensen & Skogestad 2008).

Subject to optimization is also equality and inequality constraints. These need to be satisfied in order for the optimization to be valid within the systems defined limits. In this case, each heat exchangers performance is limited by the design and its available hot stream. From Skogestad (Skogestad 2004) the goal of an optimization problem is to minimize an objective function J subject to its given constraints g and h

$$\text{minimize } J(x, u_t, d) \tag{4.1}$$

$$\text{subject to equality constraints: } g(x, u_t, d) = 0 \tag{4.2}$$

$$\text{subject to inequality constrains: } h(x, u_t, d) \geq 0 \tag{4.3}$$

where J is the objective function, x the state variables, u_t is the manipulated variables and d the disturbances. The manipulated variables also denotes the systems degrees of freedom (DOFs). The equality constraints g include the model equations, whereas the *inequality* constraints for the cases studied in this report includes the ΔT_{min} for each heat exchanger. The inequality constraints are only present for numerical purposes as it prevents the heat exchangers from unwanted temperature cross.

From a control perspective the task is to decide what to control with the available degrees of freedom, u. If the states x are eliminated by use of the model equations g the remaining unconstrained problem is

$$min_u \ J(u, d) = J(u_{opt}, d) = J_{opt}(d) \tag{4.4}$$

Here, u_{opt} is to be found and $J_{opt}(d)$ is the optimal value of the objective function J. Jensen and Skogestad (Jensen & Skogestad 2008) state that the total annualized costs associated with operation of heat exchanger networks are divided into operation costs and capital costs.

$$min_u(J_{operation} + J_{capital}) \tag{4.5}$$

Where u is the degrees of freedom which includes all the equipment data and operating variables. As this study investigates *operation* of heat exchanger networks, only the operation costs ($J_{opertaion}$) in Equation 4.5 will be considered. A general heat exchanger network with N heat exchanger in series on the upper branch and M heat exchangers in series on the lower branch is presented in Figure 4.1.

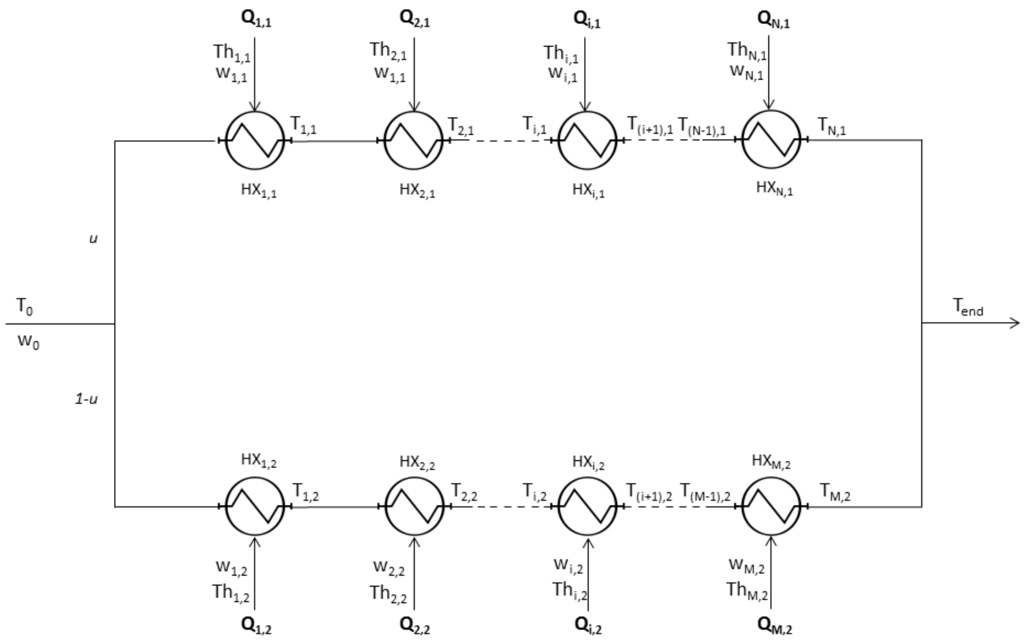


Figure 4.1: A general heat exchanger network with N heat exchanger in series on the upper branch (branch 1) and M heat exchangers in series on the lower branch (branch 2)

4.1 Optimal Operation Problems

As different sources of heat may have different prices, Jäschke (Jaeschke 2012) has proposed a cost function for operation of a general heat exchanger network. For a heat exchanger network in Figure 4.1, consisting of N heat exchangers in series on the upper branch ($j = 1$) and M heat exchangers in series on the lower branch ($j = 2$), the cost function proposed by Jäschke is

$$\begin{aligned} J = (P_{i,1}(T_{i,1} - T_{i-1,1}) + \cdots + P_{N,1}(T_{N,1} - T_{N-1,1}))uw_0 \\ +(P_{i,2}(T_{i,2} - T_{i-1,2}) + \cdots + P_{M,2}(T_{M,2} - T_{M-1,2}))(1-u)w_0 \end{aligned} \tag{4.6}$$

Where all $P_{i,1}$ and $P_{i,2}$ are negative price constants given in [$^{\$}/_{kW}$] associated with the price of transferring the heat $Q_{i,1}$ and $Q_{i,2}$ through heat exchanger $HX_{i,1}$ and $HX_{i,2}$, respectively. $T_{i-1,1}$ and $T_{i,1}$ are the temperature of the cold stream entering and leaving heat exchanger i on branch 1, respectively. Branch 1 is associated with the split u, and branch 2 with the remaining $(1-u)$, hence the product $(T_{i,1} - T_{i-1,1})uw_0$ resembles the transferred heat $Q_{i,1}$ in heat exchanger i on branch 1 given in Figure 4.1. The same applies for all heat exchangers on branch 2. This product serves as an extended version of the energy balance in Equation 3.4. Doing an unit analysis, the cost function to be minimized is the negative of the total costs given in [$]. This means that the lower the negative $P_{i,j}$ value for a certain heat exchanger, the cheaper it is to operate. If all price constants are equal, this cost function corresponds to maximizing the total transfered heat (Jaeschke 2012).

The Underwood approximation (Underwood 1933) given in Equation 3.10, Section 3.1.1 is used in simulations investigating optimal operation. Moreover, as this study takes on to operation of heat exchanger *networks* the notation in the original model equations from Section 3.1 is adjusted. For the general heat exchanger network in Figure 4.1, the heat exchanger equation for one given heat exchanger is thereby

$$Q_{i,j} = UA_{i,j}\Delta T_{UN_{i,j}} \tag{4.7}$$

Here, $UA_{i,j}$ is the respective UA design value for heat exchanger i on branch j. The total mass balance of the system is

$$w_0 = uw_0 + (1-u)w_0 \tag{4.8}$$

From this the overall energy balance with N heat exchanger on branch 1 and M heat exchangers on branch 2 becomes

$$w_0T_{end} = uw_0T_{N,1} + (1-u)w_0T_{M,2} \tag{4.9}$$

Applying the same notation for the energy balances given in Equation 3.4 and 3.5, the equality constraints for a general heat exchanger network with N heat exchangers on branch 1 and M heat exchangers on branch 2 is

$$
g = \begin{pmatrix}
Q_{1,1} - (uw_0(T_{1,1} - T_0)) \\
Q_{1,1} + (w_{1,1}(Th^{out}_{1,1} - Th_{1,1})) \\
Q_{1,1} - (UA_{1,1}\Delta T_{(1,1)UN}) \\
\vdots \\
Q_{N,1} - (uw_0(T_{N,1} - T_{(N-1),1})) \\
Q_{N,1} + (w_{N,1}(Th^{out}_{N,1} - Th_{N,1})) \\
Q_{N,1} - (UA_{N,1}\Delta T_{(N,1)UN}) \\
\\
Q_{1,2} - ((1-u)w_0(T_{1,2} - T_0)) \\
Q_{1,2} + (w_{1,2}(Th^{out}_{1,2} - Th_{1,2})) \\
Q_{1,2} - (UA_{1,2}\Delta T_{(1,2)UN}) \\
\vdots \\
Q_{M,2} - ((1-u)w_0(T_{M,2} - T_{(M-1),2})) \\
Q_{M,2} + (w_{M,2}(Th^{out}_{M,2} - Th_{M,2})) \\
Q_{M,2} - (UA_{M,2}\Delta T_{(M,2)UN}) \\
\\
uw_0 + (1-u)w_0 - w_0 \\
uw_0 T_{N,1} + (1-u)w_0 T_{M,2} - w_0 T_{end}
\end{pmatrix} = 0 \tag{4.10}
$$

where $Th^{out}_{i,j}$ is the hot stream outlet temperature associated with heat exchanger i on branch j.

Inequality constraints includes the ΔT_{min} constraint and is only included to ensure that the temperature difference on hot and cold side always is > 0, and thereby prevent from complex solutions. The value of ΔT_{min} is chosen to be 0.5. The temperature difference ΔT is illustrated in Figure 4.2.

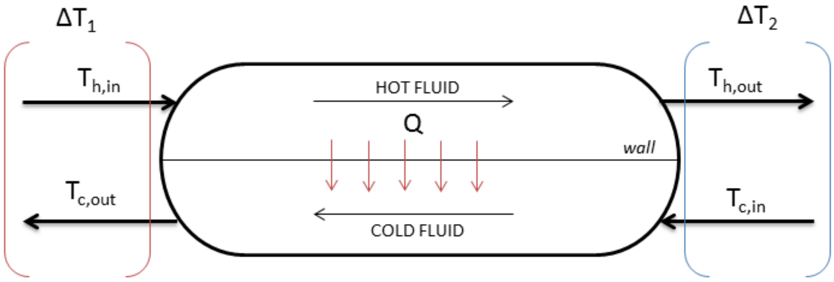


Figure 4.2: ΔT in a heat exchanger

The general inequality constraint vector can then be written

$$h = \begin{pmatrix} Th_{1,1} - T_{1,1} - \Delta T_{min} \\ Th_{1,1}^{out} - T_0 - \Delta T_{min} \\ \vdots \\ Th_{N,1} - T_{N,1} - \Delta T_{min} \\ Th_{N,1}^{out} - T_{(N-1),1} - \Delta T_{min} \\ \\ Th_{1,2} - T_{1,2} - \Delta T_{min} \\ Th_{1,2}^{out} - T_0 - \Delta T_{min} \\ \vdots \\ Th_{M,2} - T_{M,2} - \Delta T_{min} \\ Th_{M,2}^{out} - T_{(M-1),2} - \Delta T_{min} \end{pmatrix} \geq 0 \tag{4.11}$$

5 Self-Optimizing Control

Self-optimizing control is when near-optimal operation is achieved with constant setpoints for the controlled variables (Skogestad 2004). The advantage with self-optimizing control is that it does not need re-optimization when disturbances are present.

5.1 General Idea

The aim for self-optimizing control is to find a subset of the measured variables named c to keep constant at the optimal values c_{opt} (Skogestad 2004). The ideal case would give a disturbance-insensitive c_{opt} to obtain optimal operation. However, in practice, there is a loss associated with keeping the controlled variable constant. Therefore, the goal is an operation as *close to* optimum as possible. The loss can be expressed as

$$L(u,d) = J(u,d) - J_{opt}(d) \tag{5.1}$$

Skogestad (Skogestad 2000) presents the following guidelines for selecting controlled variables:

- c_{opt} should be insensitive to disturbances
- c should be easy to measure and control accurately
- c should be sensitive to change in the manipulated variables (degrees of freedom)
- For cases with more than one unconstrained degree of freedom, the selected controlled variables should be independent

Proposed by Halvorsen & Skogestad (Halvorsen & Skogestad 1997), an ideal self-optimizing variable is the gradient of the objective function J:

$$c_{ideal} = \frac{\partial J}{\partial u} \tag{5.2}$$

To ensure optimal operation for all disturbances, this gradient should be zero, but measurements of the gradient is usually not available. Therefore, computing this gradient requires values of unmeasured disturbances. To find the best suitable variables for approximations of the gradient, several methods can be used, including:

- Exact local method (Halvorsen, Skogestad, Morud & Alstad 2003)
- Direct evaluation of loss for all disturbances ("brute force") (Skogestad 2000)
- Maximum (scaled) gain method (Halvorsen et al. 2003)
- The null space method (Alstad & Skogestad 2007)

5.2 Jäschke Temperatures

For operation and control of different heat exchanger networks, Jäschke has proposed a self-optimizing control structure, currently considered as a patent application (Jaeschke 2012). The idea with the control structure proposed by Jäschke is to achieve near optimal operation by only manipulating the split u in the network, exclusively based on simple temperature measurements. The control variable is the *Jäschke temperature*, in which each heat exchangers respective Jäschke temperature on one branch is summed up to a total Jäschke temperature for the whole series. For a general heat exchanger network given in Figure 4.1, Equations 5.3 - 5.6 gives the Jäschke temperature ($JT_{i,1}$) for each heat exchanger on the upper branch ($j = 1$).

$$JT_{1,1} = P_{1,1}\frac{(T_{1,1} - T_0)^2}{Th_{1,1} - T_0} \tag{5.3}$$

$$JT_{2,1} = P_{2,1}\frac{((T_{2,1} - T_{1,1})(T_{2,1} + T_{1,1} - 2T_0 - JT_{1,1}))}{Th_{2,1} - T_{1,1}} \tag{5.4}$$

$$\vdots$$

$$JT_{i,1} = P_{i,1}\frac{((T_{i,1} - T_{(i-1),1})(T_{i,1} + T_{(i-1),1} - 2T_0 - JT_{i-1,1}))}{Th_{i,1} - T_{(i-1),1}} \tag{5.5}$$

$$\vdots$$

$$JT_{N,1} = P_{N,1}\frac{((T_{N,1} - T_{(N-1),1})(T_{N,1} + T_{(N-1),1} - 2T_0 - JT_{(N-1),1}))}{Th_{N,1} - T_{(N-1),1}} \tag{5.6}$$

Here, subscript $i, 1$ means heat exchanger i on the upper branch (branch 1). Further, P is the price constant introduced in Equation 4.6 in Section 4.1, T is still the temperature of the cold stream and Th is the temperature of hot stream.

The weighted sum of all Jäschke temperatures on the upper branch is defined as (Jaeschke 2012)

$$c_1 = JT_{1,1} + JT_{2,1} + \ldots + JT_{N,1} = \sum_{i=1}^{N} P_{i,1} JT_{i,1} \tag{5.7}$$

The same equations applies for the lower branch ($j = 2$), and the resulting weighted Jäschke temperature for the M heat exchangers in series on this branch is

$$c_2 = JT_{1,2} + JT_{2,2} + \ldots + JT_{M,2} = \sum_{i=1}^{M} P_{i,2} JT_{i,2} \tag{5.8}$$

According to Jäschke (Jaeschke 2012), near optimal operation is achieved when the Jäschke temperature for the upper branch equals the Jäschke temperature for the lower branch

$$JT = c_1 - c_2 = 0 \tag{5.9}$$

Hence, the control variable c is

$$c = JT \tag{5.10}$$

The only degree of freedom is the split u (See Figure 4.1), which will be adjusted to satisfy Equation 5.9.

6 Steady State Analysis Results

The specialization project (Aaltvedt 2012) confirmed that the Jäschke temperature gave close to optimal operation at steady state for various heat exchanger networks limited by 3 heat exchanger in series on one branch. In *this* study, two networks were analyzed first, one with four heat exchanger in series and another one with six heat exchangers in series. These two cases were simulated using MATLAB and `fmincon`. The procedure is further explained in the next section. Of these two cases, only the case with four heat exchangers in series is presented in the report. See Appendix A.2 for the case with six heat exchangers in series. Additional simulation results are also given for the case with four heat exchangers in series in Appendix A.1.

For a simpler network of two heat exchanger in parallel, several more comprehensive steady state analyzes were done using the NTU Method described in Section 3.1.1. The detailed method are described in Section 6.2, and are followed by the the following investigations:

- Investigation of Jäschke temperature operation for a base case with evenly distributed heat capacities (Case II)
- Investigation of Jäschke temperature operation for two extreme cases with uneven distribution of heat capacities (Case II-a and II-b)
- Investigation of Jäschke temperature operation subject to measurement errors

6.1 Case I: Four Heat Exchangers in Series and One in Parallel

The network of four heat exchanger in series parallel to one heat exchanger are shown in Figure 6.1. The respective parameters are given in Table 6.1 and the respective price constants $P_{i,j}$ are given in Table 6.2. With the given design parameters, outlet temperatures and split (given in red in Figure 6.1) were to be determined.

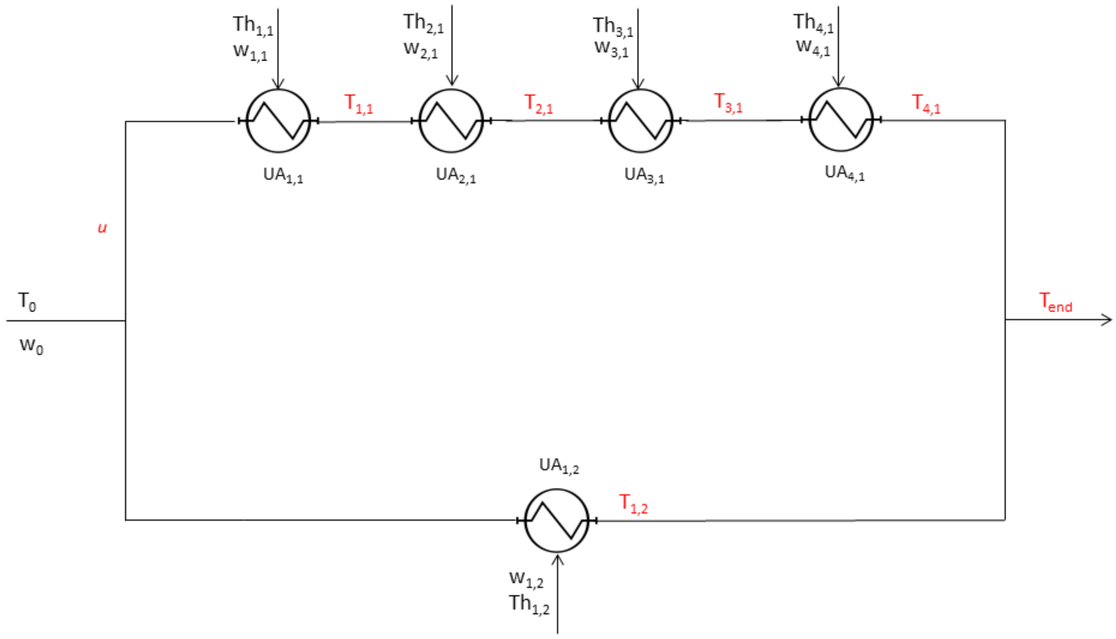


Figure 6.1: Case I: Four heat exchangers in series parallel to one heat exchanger

Table 6.1: Case I parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	190	[°C]
$Th_{2,1}$	203	[°C]
$Th_{3,1}$	220	[°C]
$Th_{4,1}$	235	[°C]
$Th_{1,2}$	210	[°C]
w_0	100	[$kW/°C$]
$w_{1,1}$	50	[$kW/°C$]
$w_{2,1}$	30	[$kW/°C$]
$w_{3,1}$	15	[$kW/°C$]
$w_{4,1}$	25	[$kW/°C$]
$w_{1,2}$	70	[$kW/°C$]
$UA_{1,1}$	5	[$kWm^2/°C$]
$UA_{2,1}$	7	[$kWm^2/°C$]
$UA_{3,1}$	10	[$kWm^2/°C$]
$UA_{4,1}$	12	[$kWm^2/°C$]
$UA_{1,2}$	9	[$kWm^2/°C$]

Table 6.2: Case I price constants

Parameter	Value	Unit
$P_{1,1}$	-1	[$/kW]
$P_{2,1}$	-1.2	[$/kW]
$P_{3,1}$	-1.3	[$/kW]
$P_{4,1}$	-1.5	[$/kW]
$P_{1,2}$	-1.4	[$/kW]

Subject to the equality and inequality constraints given in Section 4.1 (Vector 4.10 and 4.11, respectively), optimal operation and operation using the Jäschke temperature was determined by the use of the build-in MATLAB function `fmincon`. The cost function proposed by Jäschke (Jaeschke 2012) in Equation 4.6 was used as objective function, and the Underwood Approximation (Underwood 1933) was used as an approximation to the LMTD. The results from optimal operation was compared to the Jäschke temperature operation and are given in Table 6.3

Table 6.3: Optimal operation and Jäschke temperature operation for Case I

	Optimal operation	Jäschke temperature operation
T_{end} [°C]	207.87	207.84
u [%]	64.15	70.66

As the results from Table 6.3 indicates, the Jäschke temperature operates the system close to optimum, as the outlet temperature from Jäschke temperature operation only differs 0.03 °C from optimal outlet temperature. The split, however, is different. This can imply that the optimum is very flat, i.e. the highest outlet temperatures covers a great range of possible splits.

The same observation can be seen for a system of six heat exchanger in series and one in parallel. Complete simulations results for both cases are given in Appendix A

6.2 Case II: Two Heat Exchangers in Parallel

From Section 6.1 and Appendix A the Jäschke temperature showed satisfactory control for a heat exchanger network with four and six heat exchangers in series. Therefore, to reveal any limitations associated with the Jäschke temperature operation, a smaller system with two heat exchangers in parallel was used in the proceeding steady state analysis. A small system like this is easier to work with, and can at the same time be a good representative for the behavior of more complex systems. The Case II network is presented in Figure 6.2.

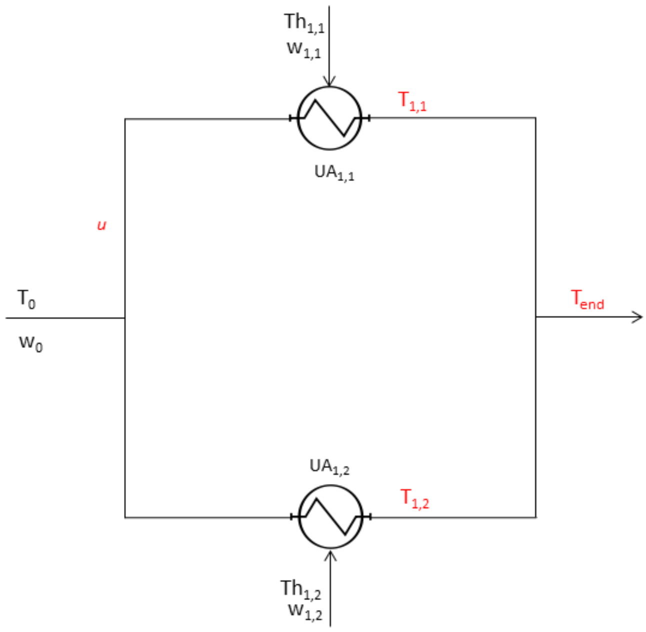


Figure 6.2: Case II: Two heat exchangers in parallel

In the following steady state simulations, the NTU-method from Section 3.1.1 was used for all heat exchanger calculations. Both heat exchangers respective outlet temperatures together with the control variable JT controlling the Jäschke temperatures were calculated for all splits $u \in [0,1]$. From this, optimal operation was determined from the split u that gave the highest outlet temperature T_{end}, and optimal Jäschke temperature operation was calculated from the point where $JT = c_1 - c_2 = 0$ (Equation 5.9). The two results were compared and the loss (in terms of outlet temperature) associated with the Jäschke temperature operation was calculated.

For this network, a base case was studied first, with parameters included in Table 6.4. The price constants for this case was all decided to be 1. The simulation results are shown in Figure 6.3. Here, the control variable JT and outlet temperature T_{end} are plotted as a function of split u (with respect to branch 1). The red and black dotted lines shows optimal operation and optimal Jäschke temperature operation, respectively. As expected from the results from the specialization project (Aaltvedt 2012), the Jäschke temperature operation showed close to optimal operation.

Table 6.4: Case II parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	203	[°C]
$Th_{1,2}$	248	[°C]
w_0	100	[$kW/°C$]
$w_{1,1}$	50	[$kW/°C$]
$w_{1,2}$	50	[$kW/°C$]
$UA_{1,1}$	10	[$kWm^2/°C$]
$UA_{1,2}$	30	[$kWm^2/°C$]

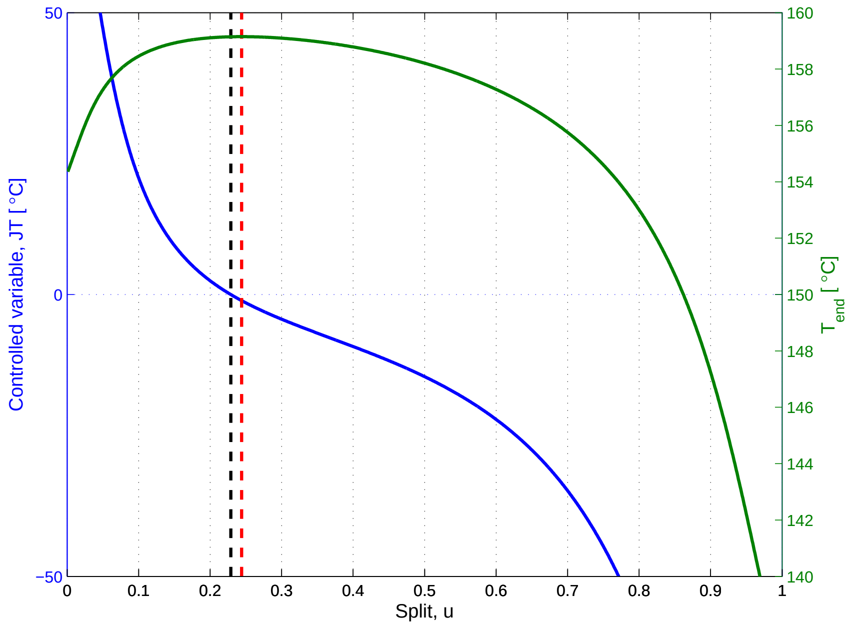


Figure 6.3: Control variable JT and T_{end} as a function of split u for Case II. The red and black dotted lines show optimal split considering outlet temperature and control variable, respectively

The plot shows a very flat optimum, i.e. several different splits allow close to optimal outlet temperature. Outlet temperature from optimal operation and Jäschke temperature operation was 159.15 and 159.14 °C, respectively, giving a small 0.01 °C temperature loss.

To investigate whether the Jäschke temperature fails to operate the system close to its optimum, more complex cases with a more uneven distribution of heat capacities were studied. This was done using the same method, and is presented in the next sections.

6.2.1 Jäschke Temperature Operation at Extreme Cases

The first extreme case, Case II-a, included a combination of one large heat exchanger with a correspondingly large heat capacity rate of the hot stream, and a small heat exchanger with a correspondingly small heat capacity rate of the hot stream. The second extreme case, Case II-b, included the same two very different hot stream heat capacities but two equally big heat exchanger areas. Both these cases corresponds to poor design, and is not realistic. However, it was included in order to study how the Jäschke temperature approach behaves in extreme cases.

The detailed parameters for Case II-a and Case II-b are given in Table 6.5 and 6.6, respectively.

Table 6.5: Case II-a parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	203	[°C]
$Th_{1,2}$	248	[°C]
w_0	100	$[{}^{kW}/_{\circ C}]$
$w_{1,1}$	400	$[{}^{kW}/_{\circ C}]$
$w_{1,2}$	100	$[{}^{kW}/_{\circ C}]$
$UA_{1,1}$	1000	$[{}^{kWm^2}/_{\circ C}]$
$UA_{1,2}$	100	$[{}^{kWm^2}/_{\circ C}]$

Table 6.6: Case II-b parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	203	[°C]
$Th_{1,2}$	248	[°C]
w_0	100	$[{}^{kW}/_{\circ C}]$
$w_{1,1}$	400	$[{}^{kW}/_{\circ C}]$
$w_{1,2}$	100	$[{}^{kW}/_{\circ C}]$
$UA_{1,1}$	1000	$[{}^{kWm^2}/_{\circ C}]$
$UA_{1,2}$	1000	$[{}^{kWm^2}/_{\circ C}]$

These parameter selections gave a more distinct optimum, which makes these cases good examples of which the Jäschke temperature did *not* operate the system close to its optimum. For Case II-a, this can be seen in Figure 6.4, where the control variable JT and outlet temperature T_{end} are plotted as function of the split u.

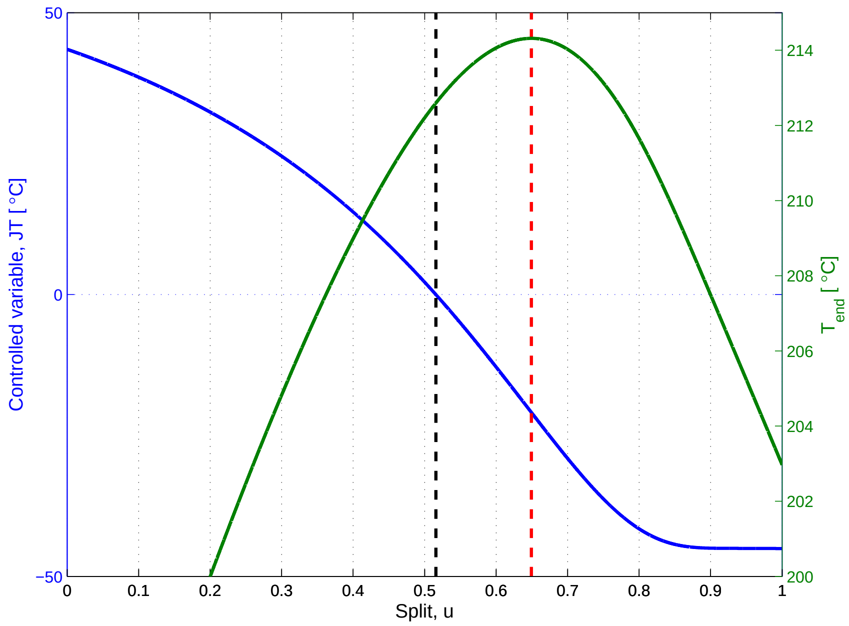


Figure 6.4: Control variable JT and T_{end} as a function of split u for Case II-a. The red and black dotted lines show optimal split considering outlet temperature and control variable, respectively

As Figure 6.4 for Case II-a indicates, the point where $JT = c_1$ - $c_2 = 0$ (optimal control variable) differs significantly from the point of optimal operation. The outlet temperature associated with optimal operation and Jäschke temperature operation was 214.32 and 212.60 °C, respectively, giving a loss of 1.72 °C. The optimum is steep, which gives few possible splits for the highest outlet temperature.

For the second extreme case, Case II-b, the area $A_{1,2}$ of heat exchanger $HX_{1,2}$ on the lower branch took the same value as heat exchanger $HX_{1,1}$. This will, together with the originally low heat capacity rate $w_{1,2}$, allow for a much better heat transfer on the lower branch. Figure 6.5 presents the control variable JT and outlet temperature T_{end} plotted as function of the split u for Case II-b. As Figure 6.5 indicates, the Jäschke temperature diverged and ended up at a steady state value where $c_1 \neq c_2$ and thereby $JT \neq 0$.

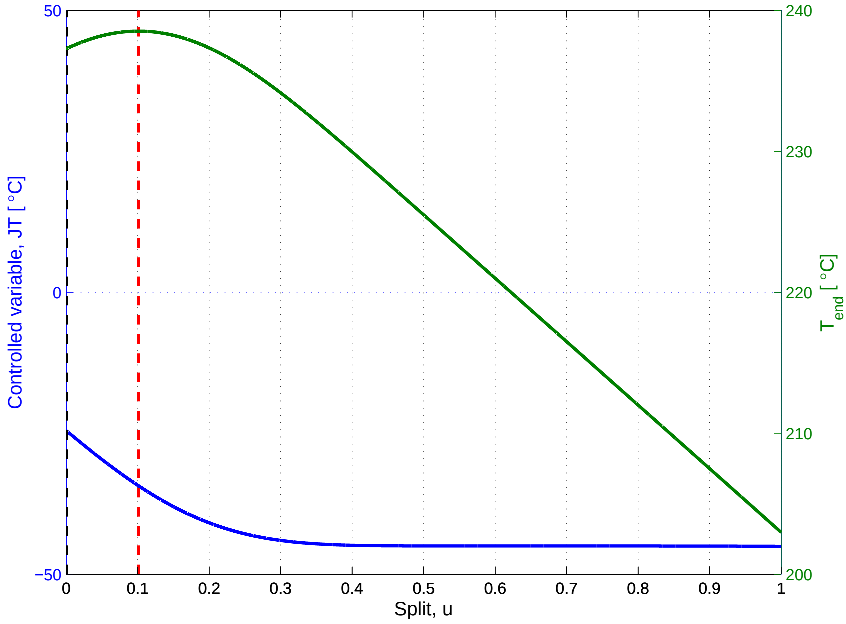
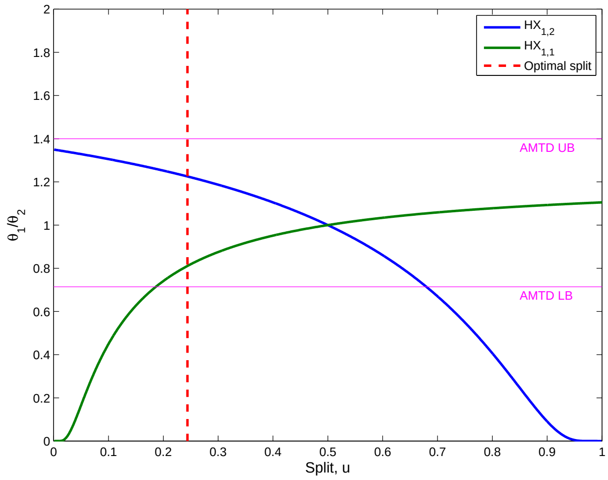


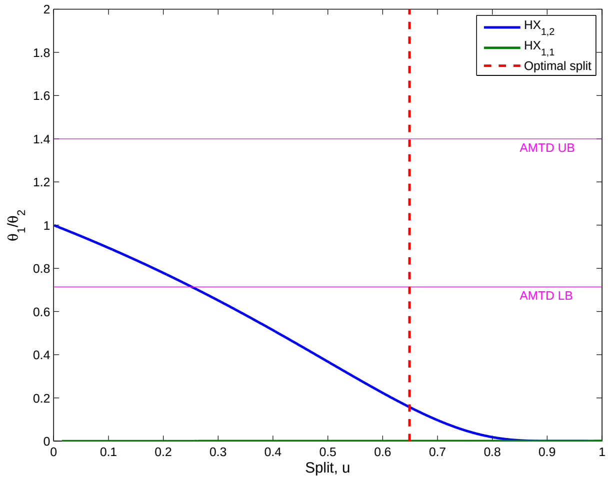
Figure 6.5: Control variable JT and T_{end} as a function of split u for Case II-b. The red and black dotted lines show optimal split considering outlet temperature and control variable, respectively

The split resulted from Jäschke temperature operation was $u = 0.01$, giving a very small cold stream distribution through the upper branch. The optimal split was $u = 0.10$. However, the outlet temperature T_{end} associated with the Jäschke temperature operation was still relatively close to the optimal outlet temperature, 237.61 vs 238.53 °C giving a temperature loss of 0.92 °C.

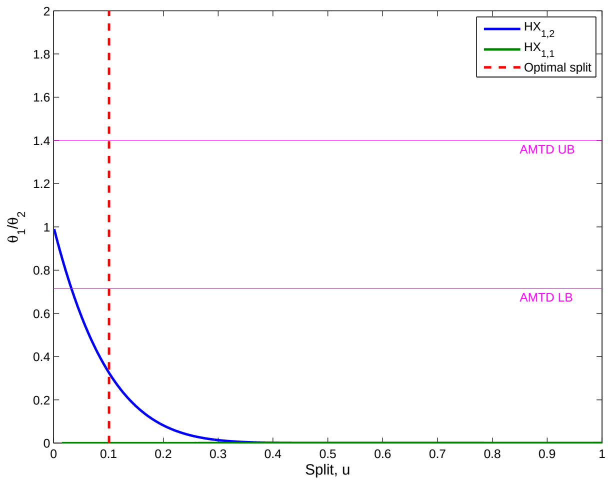
The observed error caused by operating the system with the Jäschke temperature can be traced back to the AMTD approximation (Equation 3.10, Section 3.1.1). The derivation of the Jäschke temperature is based on systems of which the AMTD approximation is valid (Jaeschke 2012). The plots in Figure 6.6 show each heat exchangers θ_1/θ_2 relationship (recall Section 3.1.1) with the split u for the base case and both extreme cases Case II-a and Case II-b, respectively. Compared to the base case it is indicated that the AMTD serves as a very bad approximation for both extreme cases, as θ_1/θ_2 is way out of the bounds of $1/1.4 < \theta_1/\theta_2 < 1.4$ proposed by Skogestad (Skogestad 2003*a*). The AMTD bounds are defined by the magenta lines in Figure 6.6, where UB is the upper bound ($\theta_1/\theta_2 = 1.4$) and LB is the lower bound ($\theta_1/\theta_2 = 1/1.4$). The plots are based on a plotting command from Edvardsen (Edvardsen 2011).



(a) Base case



(b) Extreme case Case II-a



(c) Extreme case Case II-b

Figure 6.6: Validity of the AMTD approximation, $\frac{\theta_1}{\theta_2}$ as a function of split u

According to Skogestad (Skogestad 2003a), within the horizontal magenta lines in Figure 6.6, the AMTD will serve as a satisfactory approximation to the LMTD. For Case II-a, around the optimal split of $u = 0.65$, none of the heat exchangers showed a θ_1/θ_2 ratio within this interval. The same pattern applied for Case II-b around the split $u = 0.10$. This will result in inaccurate temperature calculations in each heat exchanger, serving the controller with wrong data and eventually result in a far from optimum operation.

Equal simulations were done for two additional cases, Case II-c and Case II-d, respectively. The respective inlet parameters together with the simulation results are given in Section A.3.1 and A.3.2 in Appendix A, respectively.

6.2.2 Jäschke Temperature Operaton Subject to Measurement Errors

The accuracy of control instrumentation is very important with accuracy requirements related to control system objectives (Seborg, Edgar, Mellichamp & Doyle 2011). Therefore, in order to further investigate whether the Jäschke temperature control configuration operates a parallel heat exchanger network satisfactory, steady state simulations with implemented measurement errors were done.

Based on the case parameters for the base case, Case II-a and Case II-b in Table 6.4, 6.5 and 6.6, optimal operation was determined. Then, in the presence of measurement errors, the corresponding Jäschke temperature operation was calculated. The measurement errors were limited to span from +/– 2 °C from each respective measured temperature, and were determined by the build-in MATLAB function `rand`.

Both optimal operation and Jäschke temperature operation were calculated based on the NTU-method described in Section 3.1.1. The final results are based on 1000 simulations with random measurement error. The same measurement errors were used for every case. The loss associated with keeping the control variable constant was given in Equation 5.1. For this case the loss was seen in terms of outlet temperature, T_{end}:

$$L = T_{end}^{opt} - T_{end}^{JT} \tag{6.1}$$

Where T_{end}^{opt} is the outlet temperature from optimal operation (*without* the

Jäschke temperature), and T_{end}^{JT} is the actual outlet temperature from operation using Jäschke temperature in the presence of measurement errors. The maximum and average loss that occurred were detected and are given in Table 6.7

Table 6.7: Temperature loss associated with measurement errors

Case	Worst case loss [°C]	Average loss [°C]
Base case	0.039	0.007
Case II-a	3.141	1.602
Case II-b	0.921	0.921

For the base case, both the worst case and the average loss is small enough to give satisfactory near-optimal operation. However, the simulations of the extreme cases showed that the Jäschke temperature gave a significant error in the presence of measurement noise. For the worst case loss in Case II-a, a temperature loss almost twice as big as the temperature loss found for the exact measurement simulation in Section 6.2.1 was observed. On the other hand, the average loss, which in general is more applicable, showed a slightly *lower* temperature loss than the temperature loss observed with exact measurement. 1.60 °C versus 1.72 °C, respectively.

For Case II-b both the average and the worst case losses are equal to the temperature loss associated with the exact measurements found in Section 6.2.1. This can be related to the divergence of the Jäschke temperature, resulted in a control variable $JT \neq 0$. As seen from Figure 6.5, the point favoring optimal control variable is $u \to 0$. This means that for this case, within the limits of u, the Jäschke temperature has its absolute minimum and optimal point at the boundary of u - giving the controller no choice but to stay on this boundary.

In summary, it was found that controlling the Jäschke temperatures to equal values gives good performance in the presence of noise when the heat exchanger network is balanced (approximately similar heat capacities on the hot and cold side). However, for a unbalanced network, with large differences in the total heat capacity on each branch, noise can significantly deteriorate the performance. Equal simulations were also done for the two additional cases, Case II-c and Case II-d, respectively. These results are given in Appendix A

7 Dynamic Analysis Results

Using the equations presented in Section 3.2 on dynamic heat exchanger modeling, several heat exchanger networks were modeled using the Simulink software.

- Dynamic case I: Two heat exchangers in parallel
- Dynamic case II (base case): Two heat exchangers in series parallel to one heat exchanger
- Dynamic case III: Three heat exchangers in series parallel to two heat exchangers
- Dynamic case IV: Four heat exchangers in series parallel to one heat exchanger
- Dynamic case V: Six heat exchangers in series parallel to one heat exchanger

For all networks, the parameters for each respective heat exchanger in Dynamic case I - III were the same as used in the steady state analysis in the specialization project (Aaltvedt 2012). For Dynamic case IV and V, the parameters were the same as the ones used in the steady state analysis from *this* study (Section 6). All parameters associated with Dynamic case I - III are reprinted in the report. However, the heat transfer coefficient $h_{i,j}$ and heat exchanger area $A_{i,j}$ associated with each heat exchanger were estimated by simulations to match the resulting optimal operation variables found in both steady state analyzes. The estimations of $h_{i,j}$ and $A_{i,j}$ gave new design variables (UA values) for each heat exchanger, different from the originally optimal designed UA values. In steady state simulations where the Underwood approximation (Underwood 1933) was used (Dynamic case I - III) the new UA values turned out higher. In steady state simulations approximated by the AMTD (Skogestad 2003a) (Dynamic case IV and V), the new design values were observed lower. The estimations of $h_{i,j}$ and $A_{i,j}$ together with other relevant heat exchanger data are given in respective tables for each case in Appendix B.

A model order of $R = 10$ was used for all simulations in order to assure good accuracy. A transport delay of $\theta = 2$ sec was implemented for each measurement (i.e. temperatures) in each network. For Dynamic case I - III, each heat exchangers respective price constant $P_{i,j}$ was chosen to be 1, which means that the price had no influence on the Jäschke temperature operation. For the two last cases, Dynamic case IV and V, different price constants were used. For all dynamic simulations, ode15s (Stiff/DNF) was used as numerical solver.

PI controllers were used for all heat exchanger networks. The controller for each network was tuned using the Skogestad IMC (SIMC) rules (Skogestad 2003*b*) on a step response of 10 % increase in the cold fluid mass flow m_1 to the upper branch (i.e. making a step change in the split u).

A base case, denoted Dynamic case II, of two heat exchangers in series parallel to one heat exchanger are presented in the report.

The Dynamic case II heat exchanger network is given in Figure 7.1 and the full Simulink block diagram, `dynamic_21_1.mdl` is given in Figure 7.2. The inlet parameters with the new UA values are given in Table 7.1. The estimated variables $h_{i,j}$ and $A_{i,j}$ are given in Table B.7 in Appendix B. The step and control variable response from the tuning are presented in Figure 7.3. PI tuning parameters are given in Table 7.2. Complete and additional simulation results for all dynamic cases I - V are given in Appendix B.

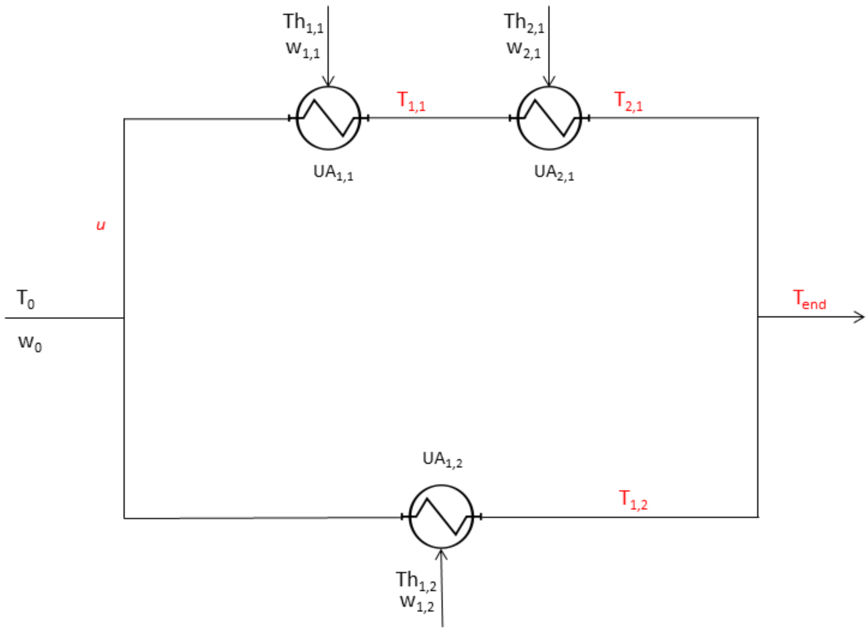


Figure 7.1: The dynamic case II (base case) heat exchanger network

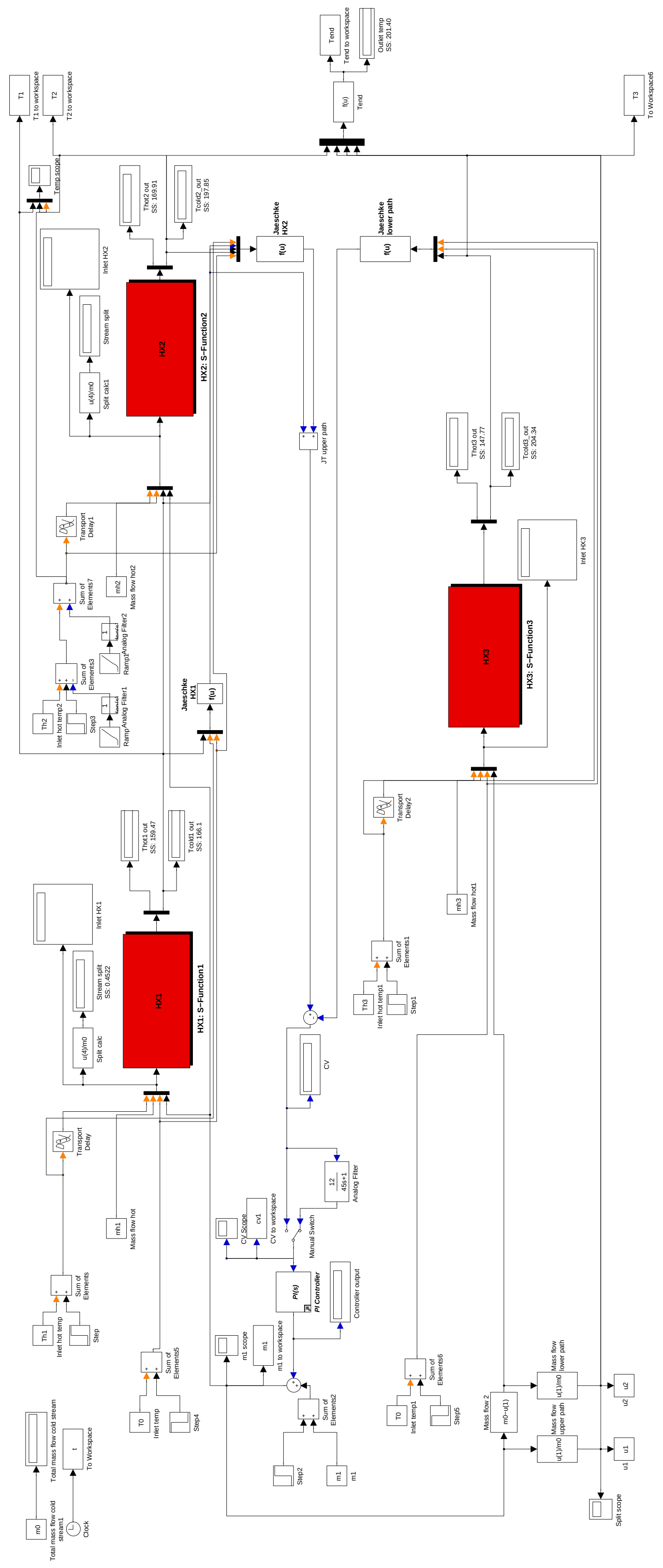


Figure 7.2: Simulink block diagram for Dynamic case II, `dynamic_21_1.mdl`

Table 7.1: Dynamic case II parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	203	[°C]
$Th_{2,1}$	255	[°C]
$Th_{1,2}$	248	[°C]
w_0	160	$[kW/°C]$
$w_{1,1}$	60	$[kW/°C]$
$w_{2,1}$	27	$[kW/°C]$
$w_{1,2}$	65	$[kW/°C]$
$UA_{1,1}$	17.78	$[kWm^2/°C]$
$UA_{2,1}$	31.18	$[kWm^2/°C]$
$UA_{1,2}$	57.79	$[kWm^2/°C]$

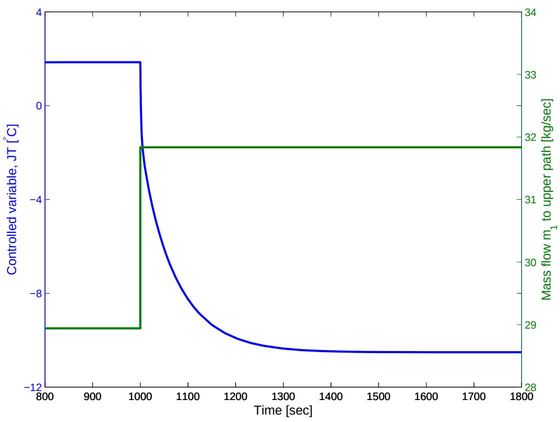


Figure 7.3: Open loop step response of control variable JT on a 10 % increase in inlet mass flow m_1 for Dynamic case II

Table 7.2: PI tuning parameters for Dynamic case II

Tuning parameter	Value	Unit
K_c	1.59	$[°C/kg/s]$
τ_I	10	[sec]

7.1 Closed Loop Steady State Parameters

Using the tuning parameters given in Table 7.2, closed loop operation variables (outlet temperatures and split) were compared to the open loop operation variables matching the steady state variables (Aaltvedt 2012).

Table 7.3: Open loop and closed loop operation variables for Dynamic case II

Operating variable	Open loop value	Closed loop value
$T_{1,1}$ [°C]	166.0	165.6
$T_{2,1}$ [°C]	197.9	197.2
$T_{1,2}$ [°C]	204.3	204.9
$Th_{1,1}^{out}$ [°C]	159.4	159.3
$Th_{2,1}^{out}$ [°C]	169.8	169.3
$Th_{1,2}^{out}$ [°C]	147.8	148.0
T_{end} [°C]	201.4	201.4
u	0.4522	0.4589

After closing the controller loop it was observed a small change in the internal system variables, i.e. outlet temperatures of each heat exchanger. Also, the split differed from the open loop simulation, but the outlet temperature T_{end} takes on the same value, 201.4 °C. These inner variations might be traced back to a flat optimum allowing several splits for maximum outlet temperature, in addition to the two different models used. The open loop values are based on a steady state simulation using the Underwood approximation (Underwood 1933), while the dynamic closed loop values are based on the mixed tank in series model (Wolff et al. 1991). Similar results for Dynamic case I and III - V are given in Appendix B.

7.2 Jäschke Temperature Operation at Small Disturbances

For the Dynamic case II system, two disturbances were applied in a close sequence over a 2000 second interval. At $t = 1000$ sec, a temperature step of +10 °C was applied in the inlet cold stream temperature T_0. Then, at $t = 1600$ sec, a negative temperature step of 25 °C in the hot stream temperature of heat exchanger $HX_{1,2}$

on the lower branch, $Th_{1,2}$ (See Figure 7.1) was applied to the system. As the controller response showed significant over- and undershoot, an analog filter was implemented filtering the signals entering the PI controller. The filter parameters are given in Table 7.4.

Table 7.4: Analog filter parameters for Dynamic case II

Filter parameter	Value	Unit
K_f	12	[$^{\circ}C/kg/s$]
τ_I	45	[sec]

The response of the control variable (JT) is shown in Figure 7.4. Included in the plot are both behaviors with and without the analog filter, as red and blue lines, respectively. The same applies for the resulting effect on the split u, shown in Figure 7.5. Similar plots are shown for Dynamic case I and III - V in Appendix B.

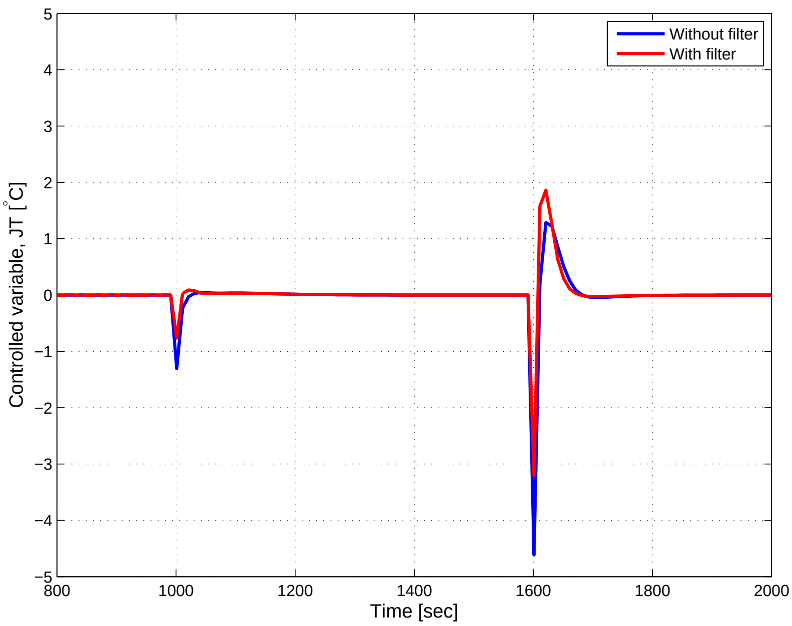


Figure 7.4: Control variable response when T_0 is increased 10 °C and $Th_{1,2}$ decreased 25 °C at $t = 1000$ and 1600 sec, respectively

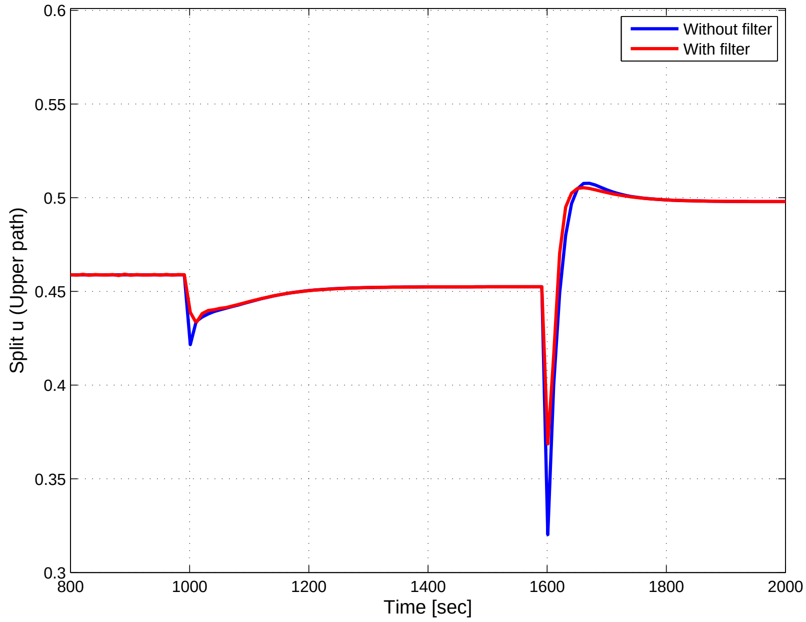


Figure 7.5: Split response when T_0 is increased 10 °C and $Th_{1,2}$ decreased 25 °C at $t = 1000$ and 1600 sec, respectively

Both plots show satisfactory disturbance rejection and system control. The split response for the temperature step in T_0 at $t = 1000$ sec was observed to be slower than the same response for the temperature drop in $Th_{1,2}$ at $t = 1600$ sec. From Figure 7.5 inverse response was observed with the second applied disturbance. This feature arise from competing dynamic effects that operate on two different time scales (Seborg et al. 2011). In this case, an immediate change in $Th_{1,2}$ at $t = 1600$ sec results in a sudden change in the Jäschke temperature for the lower branch (Equation 5.8). The impacts of decreasing $Th_{1,2}$ is not seen in the associated cold stream outlet temperature $T_{1,2}$ until some time due to the counter current stream configuration in the heat exchanger. These two different temperatures on different time scales creates the inverse response.

Both the control variable response (Figure 7.4) and the split response (Figure 7.5) experienced a significant reduction in their respective over- and undershoot with the analog filter implemented (Table 7.4). As the red lines in Figure 7.4 and 7.5 indicates, the magnitude of the peaks are almost decreased to half its original value. The settling time for the control variable was about 400 sec for the applied disturbance in inlet temperature T_0 at $t = 1000$ sec. For the disturbance applied in $Th_{1,2}$ the settling time was only about 200 sec, even though the magnitude of this disturbance was significantly higher. However, both can be considered as fast responses since temperature changes are slow processes. The outlet temperature profiles ($T_{1,1}$, $T_{2,1}$, $T_{1,2}$ and T_{end}) with the analog filter implemented were plotted as a function of time t. The temperature profiles are presented in Figure 7.6.

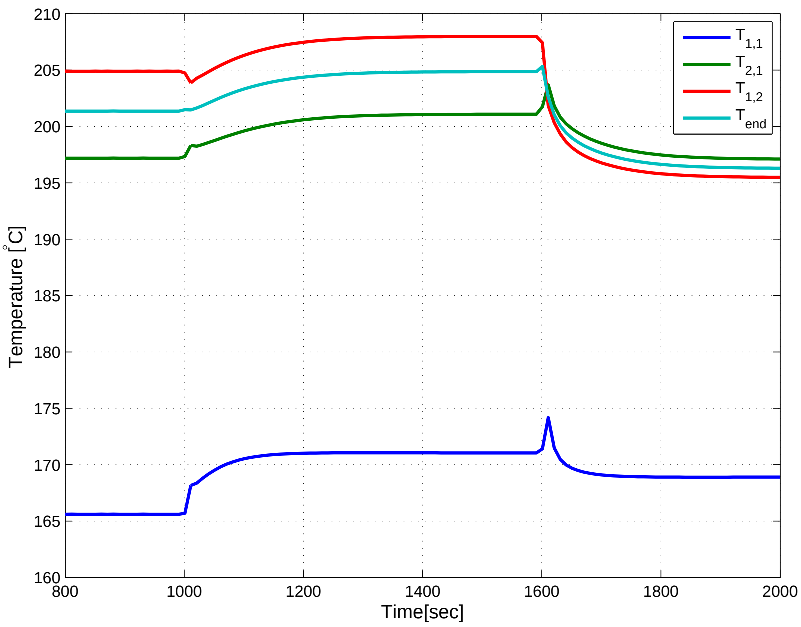


Figure 7.6: Outlet temperature response when T_0 is increased 10 °C and $Th_{1,2}$ decreased 25 °C at $t = 1000$ and 1600 sec, respectively

Worth noticing from Figure 7.6 is the temperature drop resulted from the disturbance in $Th_{1,2}$ at $t = 1600$ sec. This was observed for all potted temperature profiles. For the cold stream entering heat exchanger $HX_{1,2}$, suffering the negative temperature step change in $Th_{1,2}$, the cold stream temperature is just a direct effect of decreased heat transfer. For the cold stream passing through the *upper* branch, on the other hand, the temperature decrement is a result of the split response associated with the disturbance in $Th_{1,2}$. As Figure 7.5 indicated, the stream split through the upper branch was increased as a result of this disturbance, eventually giving more fluid to heat which resulted in lower outlet temperatures on this branch.

Also here, inverse response was observed with the 25 °C negative step change in $Th_{1,2}$ at time $t = 1600$ sec. Note that the cold stream temperature $T_{1,2}$ (red line) does not suffer from inverse response associated with the step change made in the hot stream temperature $Th_{1,2}$ at time $t = 1600$ sec.

7.3 Jäschke Temperature Operation at Major Disturbances

The results from the last section demonstrated satisfactory control by the Jäschke temperature control configuration (Jaeschke 2012) for a system present to small disturbances. To reveal any vulnerabilities associated with the Jäschke temperature the following investigation includes a system subject to more comprehensive disturbances. For the same topology, a case was studied were the hot stream temperature $Th_{2,1}$ of heat exchanger $HX_{2,1}$ experienced a slowly decrement over a 4000 sec time interval resulting in an eventually *cooling* effect in the given heat exchanger. In the presence of such an incident, the optimal operation would be to set the bypass of the current branch suffering this cooling effect to zero. In order for this to be fast and manageable enough to work with, some of the case parameters were changed. The temperatures $Th_{1,1}$ and $Th_{2,1}$ were increased and decreased, respectively, making the temperature difference between $T_{1,1}$ and $T_{2,1}$ smaller. The hot stream temperature $Th_{1,2}$ in heat exchanger $HX_{1,2}$ was also decreased. This new case was called Dynamic case II-a, with the new case parameters given in Table 7.5.

In this analysis it was decided to modify the expression for the control variable JT to prevent the simulation from singular solutions. Errors associated with singularity was observed when $T_{1,1}$ took on the same value as $Th_{2,1}$ due to the decaying temperature of $Th_{2,1}$. These two streams, the cold stream and hot stream entering heat exchanger $HX_{2,1}$ approached each other when $Th_{2,1}$ kept decreasing and u went toward zero. As a result of that, a very sudden increase in $T_{1,1}$ was observed, aimed to match the inlet hot stream temperature of heat exchanger $HX_{1,1}$. During this sudden increase, the temperatures $T_{1,1}$ and $Th_{2,1}$ crossed each other, resulted in a denominator-zero in the Jäschke temperature for heat exchanger $HX_{2,1}$ in Equation 5.4, which again resulted in a singular solution.

Therefore, it was decided to modify control variable JT adjusting the Jäschke temperatures. This was done by re-writing it to a denominator-free form. Another way of keeping the control variable JT in Equation 5.9 at its set point ($JT = 0$), is by letting the numerator of each respective heat exchangers Jäschke temperature equal zero. Therefore, for this case in particular, a modification was done, putting the control variabel JT for this system on a common denominator. Then, by use of the resulting numerator as the new control variable with a setpoint $\bar{c} = 0$, it should give the same results as the original Jäschke temperature. This modified control variable c_{mod} is given in Equation 7.1.

$$\begin{aligned} c_{mod} = (T_{1,1} - T_0)^2(Th_{2,1} - T_{1,1})(T_{1,2} - T_0) \\ +((T_{2,1} - T_{1,1})(T_{2,1} + T_{1,1} - 2T_0 - JT_{1,1}))(Th_{1,2} - T_0)(Th_{1,1} - T_0) \\ -(T_{1,2} - T_0)^2(Th_{2,1} - T_{1,1})(Th_{1,1} - T_0) \end{aligned} \tag{7.1}$$

With this new control variable the system was re-tuned using the Skogestad IMC (SIMC) rules (Skogestad 2003*b*). The controllers were tuned based on a step response of a 10 % increase in the cold fluid mass flow. The step and control variable response are given in Figure 7.7, and the resulting tuning parameters are given in Table 7.6.

Table 7.5: Dynamic case II-a parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	240	[°C]
$Th_{2,1}$	255	[°C]
$Th_{1,2}$	220	[°C]
w_0	160	$[kW/°C]$
$w_{1,1}$	60	$[kW/°C]$
$w_{2,1}$	27	$[kW/°C]$
$w_{1,2}$	65	$[kW/°C]$
$UA_{1,1}$	17.78	$[kWm^2/°C]$
$UA_{2,1}$	31.18	$[kWm^2/°C]$
$UA_{1,2}$	57.79	$[kWm^2/°C]$

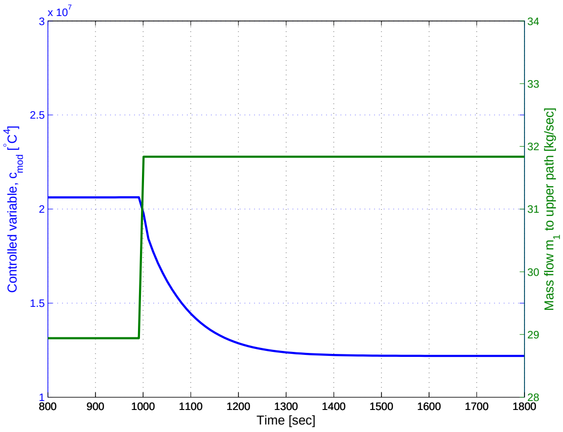


Figure 7.7: Open loop step response of modified control variable c_{mod} on a 10 % increase in inlet mass flow m_1 for Dynamic case II-a

Table 7.6: Tuning parameters for Dynamic case II-a

Tuning parameter	Value	Unit
K_f	$6.45{\cdot}10^{-6}$	[$^{\circ}$C/kg/s]
τ_I	93	[sec]

However, since the tuning was done with the original $Th_{2,1}$ at 255 °C, it was decided to increase the controller gain in order to improve the controller performance at lower values of $Th_{2,1}$. By trial and error, different tuning parameters were tested as the system showed various behavior at different controller gains. Therefore, two other sets of tuning parameters were used for this case. Results from both sets are given in the report. The new tuning parameters are given in Table 7.7 and 7.8 as set 1 and set 2, respectively.

Table 7.7: PI tuning parameters for Dynamic case II-a, set 1

Tuning parameter	Value	Unit
K_c	$6.25 \cdot 10^{-3}$	[$^{\circ}$C/kg/s]
τ_I	93	[sec]

Table 7.8: PI tuning parameters for Dynamic case II-a, set 2

Tuning parameter	Value	Unit
K_c	$6.25 \cdot 10^{-5}$	[$^{\circ}$C/kg/s]
τ_I	93	[sec]

The disturbance were simulated using the build-in `ramp` block in Simulink. Starting at $t = 2000$ sec, the hot stream temperature of heat exchanger $HX_{2,1}$, $Th_{2,1}$, was decreased with a slope of 0.05 ending up at a steady state 180 °C at time $t = 6000$ sec. This gave $Th_{2,1}$ a total temperature drop of 75 °C. The ramp signals were filtered making the slope even more smooth. The filter parameters for the ramp signals are given in Table 7.9. The full Simulink block diagram is given in Figure D.3 Appendix D

Table 7.9: Analog filter parameters for ramp signals in Dynamic case II-a

Filter parameter	Value	Unit
K_f	1	[$^{\circ}C/_{kg/s}$]
τ_I	100	[sec]

For both sets of tuning parameters, the modified control variable showed satisfactory system control in the presence of a cooling heat exchanger. The modified control variable lead the split u to zero bypass on the upper branch at the point where $Th_{2,1} < T_{1,1}$ and heat exchanger $HX_{2,1}$ gave a cooling effect. The temperature profiles for set 1 are plotted as a function of time t and are given in Figure 7.8. Only the temperature profiles for tuning set 1 was included in the report due to similar temperature response with both tuning sets. Certain temperature profiles are omitted from the plot ($Th_{1,1}$, $Th_{1,2}$ and $T_{1,2}$). This is simply because they are either constant or are not directly affected by the changes in heat exchanger $HX_{2,1}$.

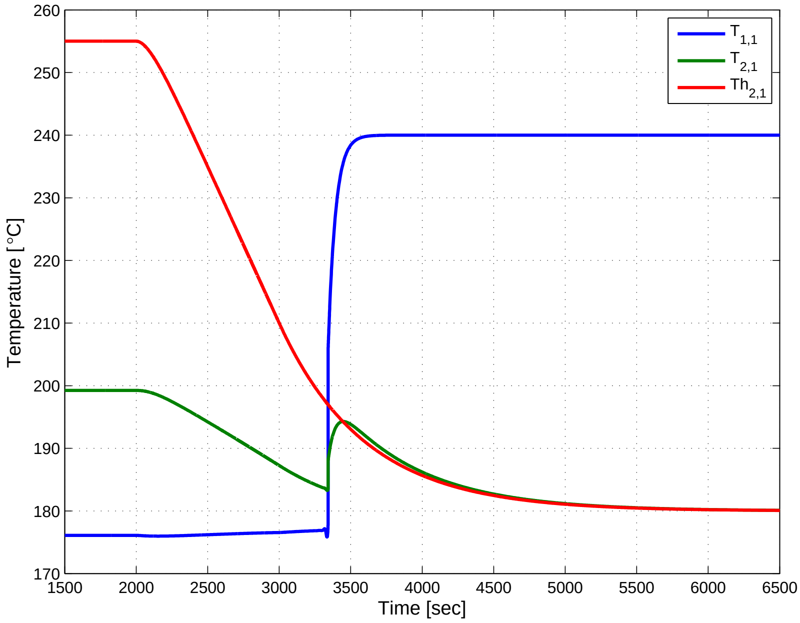
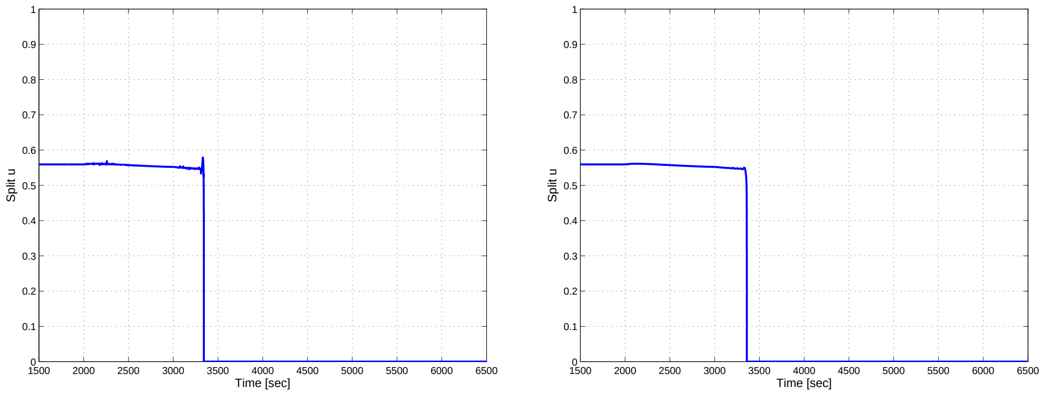


Figure 7.8: A selection of outlet temperature responses for tuning set 1 when $Th_{2,1}$ is decreased from 255 - 180 °C from time $t = 2000$ to 6000 sec

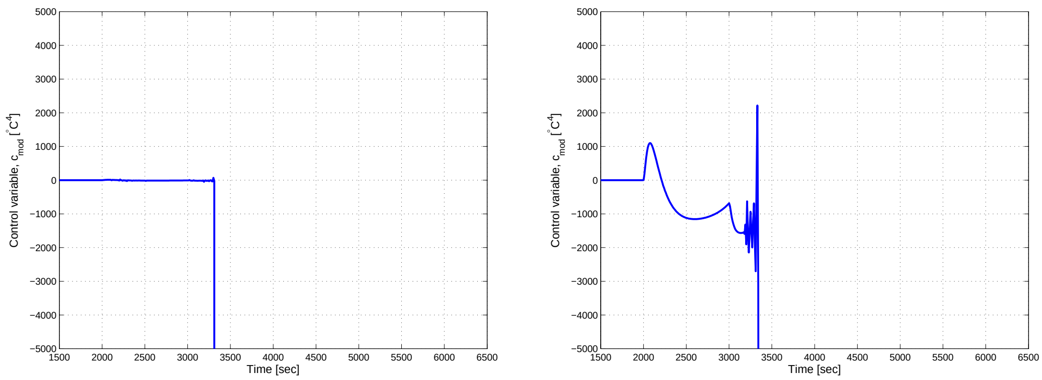
The response of the directly affected temperatures on the upper branch was as expected. As the hot stream temperature $Th_{2,1}$ in heat exchanger $HX_{2,1}$ decreased, so did the cold stream outlet temperature $T_{2,1}$ from the same heat exchanger. In other words, the heat transfer decreased as the hot stream temperature decreased, which is in good correlation with the expected behavior. The cold stream outlet temperature $T_{1,1}$ of heat exchanger $HX_{1,1}$ showed a small increment as $Th_{2,1}$ decreased. This temperature rise can be related to a simultaneously small decrement in the stream split to the upper branch. A temperature decrement in $Th_{2,1}$ makes the upper branch less favorable regarding maximum outlet temperature.

After about $t = 3350$ sec, both $T_{1,1}$ and $T_{2,1}$ experienced a very sudden increase and took on the same value as their respective hot stream inlet temperatures. $T_{1,1}$ quickly stabilized at $Th_{1,1}$ of 240 °C, and $T_{2,1}$ followed the still ongoing temperature drop of $Th_{2,1}$. This sudden temperature change was a result of a split $u \rightarrow 0$ to the upper branch. The split behavior for both sets of tuning parameters are presented in Figure 7.9, showing the split u as a function of time t. The control variable behavior for both tuning sets are presented in Figure 7.9.



(a) Split u with tuning set 1 (b) Split u with tuning set 2

Figure 7.9: Split u as a function of time t when $Th_{2,1}$ is decreased from 255 - 180 °C from time t = 2000 and 6000 sec



(a) c_{mod} with tuning set 1 (b) c_{mod} with tuning set 2

Figure 7.10: Modified control variable c_{mod} as a function of time t when $Th_{2,1}$ is decreased from 255 - 180 °C from time t = 2000 and 6000 sec

The split response for each set slightly deviate from each other. For both tuning parameter sets, the split u runs immediately to zero around t = 3350 sec. However, the split response from set 1 showed small oscillations from t = 2000 to about 3350 sec, while the resulting split response from set 2 has a more smooth decrease over the same time interval. This slightly different behavior can be related to the modified control variable c_{mod}, presented in Figure 7.10. In both cases the control variable ends up at a value of -10^7. The full range of the control variable on the ordinate axis is not included in the report due to readability. It is, however, included in Figure B.8 in Appendix B.3.

As Figure 7.10 indicates, the control variable shows a far more violent behavior for set 2, resulting in a more smooth split behavior in Figure 7.9b. As the controller gain for set 1 is 100 times bigger than the controller gain for set 2, the controller output from using set 1 will give a much bigger system input. Since the manipulated variable is the split u, this will result in greater variation in the split. The small oscillations observed in Figure 7.9a confirms this.

8 Discussion and Further Work

The discussion is organized in three parts - two parts discussing the steady state and dynamic analysis results and one part presenting further work.

8.1 Steady State Analysis Discussion

Systems with a very distinctive optimum might suffer from poor operation with the Jäschke temperature control configuration. For unbalanced heat exchanger networks with an uneven distribution of hot stream heat capacities, the self-optimizing Jäschke temperature variable showed inadequate operation as it differed at the maximum 1.72 °C from optimal operation. In the presence of the worst case measurement errors the deviation was nearly doubled. However, looking at the *average error* caused by the measurement errors for systems with a more balanced heat capacity distribution, this type of noise was not associated with the factors that influenced the operation the most. As the Jäschke temperature did not show significant aggravated behavior, this makes the Jäschke temperature a robust control configuration for balanced heat exchanger networks in terms of measurement sensitivity.

The weakness associated with unevenly distributed heat capacities throughout the network can be associated with systems where the AMTD failed to approximate the LMTD with reasonable error (Skogestad 2003*a*). System like this included the extreme cases studied in Section 6.2.1. Here, the Jäschke temperature showed relatively far from optimal operation. However, in reality heat exchanger networks should be arranged differently to achieve best possible heat integration. A system like Case II-b, with two different hot stream heat capacity rates and very big heat exchanger areas would not be optimal. It is not profitable to provide a 1000 m^2 heat exchanger with a hot stream having a heat capacity rate of 1000 $\frac{kWm^2}{°\mathrm{C}}$. This is supported by the result presented in Figure 6.5, where it was shown that the heat exchanger with these parameters only supplied 10% of the total transfered heat. This makes this configuration unlikely for a real big scale system. Additionally, according to the results from the optimization done in the specialization project (Aaltvedt 2012), it was indicated that a design allowing for an approximately 50/50 distribution to each branch was favorable for optimal operation. Heat exchanger networks with a design allowing for the AMTD approximation to be used in each heat exchanger, are both better candidates for real big scale processes and at the same time a configuration where the Jäschke temperature gives close to optimal operation.

8.2 Dynamic Analysis Discussion

Inverse response, over- and undershoot was a consistent observed phenomenon in dynamic simulations for every heat exchanger network investigated in this study. As explained in Section 7.2, two factors were causing this; the fact that counter current heat exchangers always suffers from competing dynamic effects on different time scales (Seborg et al. 2011) and the Jäschke temperature control configuration. Of these two, it is the Jäschke temperature that might be dominating, especially in the presence of disturbances of greater magnitude. The Jäschke temperatures for each heat exchanger in a given series (Equation 5.3 - 5.6 in Section 5.2), all include squared sized measurements which can apply to responses of significant magnitude. For systems like heat exchanger networks, such behavior can result in excessively big mass flows, over and above that for which certain heat exchangers originally was designed, causing structural failure and can potentially trig disasters (Sinnott & Towler 2009).

The dynamic case II-b revealed a case where the Jäschke temperature control variable failed to operate the system properly. As explained in Section 7.3, the Jäschke temperature took a negative infinite value as the temperatures in the denominator, in this case $Th_{2,1}$ and $T_{1,1}$ in Equation 5.4, approached each other. At the temperature cross where $Th_{2,1} = T_{1,1}$ a singular solution occurred causing the simulation to crash. Due to the implemented saturation limits in the controller, the resulting system input gave either a maximum or a minimum stream split to the upper branch, i.e. it showed a very unstable behavior. In the presence of such an incident, the Jäschke temperature did not show satisfactory control. For a real, large scale plant, an incident like this, with the resulting violently oscillating system input could also give a unfortunate and detrimental effect. Modifying the control variable (Equation 7.1) improved the performance of the controller. But like the original control variable did at the point where the singular solution stopped the simulation, neither the modified control variable converged to the set point ($c_1 = c_2$) at steady state. The observed response was far from smooth, as the bypass on the upper branch immediately shut down as $Th_{2,1}$ decreased further below 200 °C (Figure 7.9). From the modified control variable in Equation 7.1, each of the three terms include different temperature differences. At the point where temperature crosses are observed (Figure 7.8), violent behavior occurs as terms cancel out in the presence of a zero multiplication in one given term. As a result, big oscillations were seen in the control variable. At the point where $T_{1,1} > Th_{2,1}$ resulting in $T_{1,1} > T_{2,1}$, two of the three terms change signs form positive to negative. This makes c_{mod} all negative and the controller will immediately close the cold stream distribution to the upper branch and thereby $u \rightarrow 0$.

However, in all the cases presented in this study, the Jäschke temperature operation showed relatively close to optimal operation and good system control. Also considering the observation of a diverged steady state Jäschke temperature of $c_1 \neq c_2$ and that the control was not smooth, it still managed to operate the system satisfactory. In the presence of smaller and more realistic disturbances, the Jäschke temperature showed tight control and good disturbance rejection for all dynamic cases studied in this report.

8.3 Further Work

For all steady state and dynamic cases investigated in this study, single phase flow was assumed. In the presence of such an assumption, the Jäschke temperature showed satisfactory control and close to optimal operation for systems of which the AMTD served as a valid approximation (Skogestad 2003*a*). However, multiphase flows show an increased frequency in many of todays big industries, including the chemical, petroleum and power generation industry (Gidaspow 1994). The challenges associated with this phenomenon increase the requirements for control configurations that handle multiphase flows. For the Jäschke temperature approach, more research is needed in the presence phase transfer, as heat transfer rates are highly dependent on the phase of the fluid.

In this study, neither the matter that being heated nor the matter that is heating are given any further attention than just a constant heat capacity. The related assumption of constant mass flows of both hot and cold fluids makes the heat capacity rate, w, constant throughout all investigations. This strongly relates to the issue of phase transfer and multiphase flow. It is known that the heat capacity rate at constant pressure will vary with temperature (Sinnott & Towler 2009). Together with the heat capacity's dependency on fluid phase, occurrences like these will have a significant influence on the heat transfer when temperature disturbances resulting in phase transfer are present. For the Jäschke temperature to be versatile enough to be implemented in processes present to such temperature fluctuations, more comprehensive analyzes will be needed, emphasizing the heat capacity's complexity.

This study investigated configurations based on two parallel branches of heat exchangers, where each heat exchanger was supplied with one distinct, and most often constant hot stream. Usually, when designing heat exchanger networks, it is desirable to utilize each energy source to the maximum, achieving best possible energy recovery. That is, the available hot streams should be distributed throughout the network, finding feasible matches between streams and thereby serve several heat exchangers (Rathore & Powers 1975). With cross-overs like this, new challenges arise as noise and disturbances affect multiple heat exchangers, causing more challenging control problems. The configurations studied in this report only included two parallel branches. Aiming for the best possible heat integration it might also be desirable to include more possible branches, ending up with a more complex bypass regulation. Edvardsen (Edvardsen 2011) demonstrated that the Jäschke temperature control variable gave satisfactory control for a three branched case study, using two controllers - one controlling two branches, and the other one controlling the third branch. For more specific determination of the Jäschke temperature control variable and any versatility on different and more complex configurations, further investigations taking on to these issues are needed.

Another important issue that was not taken into great consideration in this study was the operation with different price constants, $P_{i,j}$. Associated with a general heat exchanger network is the price constant of each particular heat exchanger. With the exception of the networks included four and six heat exchanger in series, parallel to one heat exchanger, respectively, all price constants were chosen to be equal to unity throughout all investigations done in this study. This eventually gave a cost function aiming to maximize the total transfered heat, Q, not taking into account that different sources of heat may have different prices

(Jaeschke 2012). As stated in the introduction, optimal operation of heat exchanger networks is a very important aspect in the issue of obtaining maximum heat recovery from the available energy sources (Zhang et al. 2011). In the case of big scale industries, it is often necessary to supply additional energy *beyond* what's already accessible from other parts of the plant (Rathore & Powers 1975). Doing this can be expensive, as additional heat may need to be generated at the plant or outsourced from a third part service (Sinnott & Towler 2009). Therefore, optimal operation of heat exchanger networks needs to include these issues, and further investigation on these topics considering the Jäschke temperature operation will be needed. Luckily, the Jäschke temperature includes price constants in the weighted sum in Equation 5.7 and 5.8, allowing for different priced energy sources. The method can then easily be further tested for these types of configurations.

9 Conclusions

In this study the Jäschke temperature control configuration was evaluated for several different cases of parallel heat exchanger networks. The goal was to further investigate the properties of the Jäschke temperature and determine any limitations. Among the cases studied, both steady state and dynamic behavior were investigated. Far from optimal operation was revealed for systems with an uneven distribution of hot stream heat capacities. For such a system with two heat exchangers in parallel, the steady state temperature loss was 1.72 °C, feeding the control variable with exact measurement data. For the same system subject to measurement noise spanning +/– 2 °C from each respective temperature, the worst case temperature loss was 3.14 °C. Considering the *average* measurement error, the Jäschke temperature showed good robustness for this kind of noise for systems with evenly distributed heat capacities.

Poor control was observed in the presence of a decreasing hot stream temperature in one out of several heat exchangers. This feature was demonstrated for a system of two heat exchangers in series parallel to one heat exchanger. This resulted in a cooling effect, and the Jäschke temperature failed to simulate the system due to singular solutions. To prevent from singularity, the control variable was re-written to a denominator-free form, resulting in satisfactory control.

However, for systems with an even heat capacity distribution, the Jäschke temperature showed very close to optimal operation. Present to smaller and more realistic disturbances together with well tuned controllers, tight control and good disturbance rejection was achieved. This was demonstrated for all cases up to six heat exchanger in series on one branch.

Advantages with the Jäschke temperature control configuration is a control variable only dependent on simple temperature measurements, with the split u serving as the only manipulated variable. Disadvantages with this method is the inverse response and occasionally violent control behavior resulting from the Jäschke temperature equation with squared sized measurements. Also, potentially denominator-zeros as a result of temperature cross may lead to singularity, with resulting poor and sometimes wrong control. Assumptions including single phase flow and constant heat capacities were used in all simulations.

References

Aaltvedt, S. (2012), Specialization project fall 2012: Optimal operation of parallel systems, Norwegian University of Science and Technology, Trondheim.

Alstad, V. & Skogestad, S. (2007), 'Null space methods for selecting optimal measurement combinations as controlled variables', *Industrial & Chemical Engineering Research* **43**(3), 846–853.

Bartlett, D. A. (1996), 'The fundamentals of heat exchangers', *The Industrial Physicist* pp. 18–21.

Edvardsen, D. G. (2011), Master thesis: Optimal operation of heat exchanger networks, Norwegian University of Science and Technology, Trondheim.

Geankoplis, C. J. (2003), *Transport Processes and Separation Process Principles*, 4 edn, Pearson Education, Inc.

Gidaspow, D. (1994), *Multiphase Flow and Fluidization: Continuum and Kinetic Theory Description*, Academic Press, Inc.

González, A. H. & Marchetti, J. L. (2005), Minimum-cost operation in heat-exchanger networks, The 15th European Symposium on Computer Aided Process Engineering.

Gramble, C. E. (2006), 'Cost management in heat-transfer-fluid systems', *Chemical Engineering Progress (CEP)* .

Halvorsen, I. J. & Skogestad, S. (1997), Indirect on-line optimization through set-point control, Prepared for presentation at the AIChe 1997 Annual Meeting, Los Angeles, CA.

Halvorsen, I. J., Skogestad, S., Morud, J. C. & Alstad, V. (2003), 'Optimal selection of controlled variables', *Industrial & Chemical Engineering Research* **42**, 3273–3284.

Incorpera, F. P. & DeWitt, D. P. (2007), *Introduction to Heat Transfer*, 2nd edition edn, John Wiley and sons, New York.

Jaeschke, J. (2012), United Kingdom Patent Application No. 1207770.7: Parallel Heat Exchanger Control.

Jensen, J. B. & Skogestad, S. (2008), 'Problems with specifying δt_{min} in the design of processes with heat exchangers', *Industrial and Engineering Chemistry Research (ACS Publications)* **47**(9), 3071–3075.

Kay, J. & Nedderman, R. (1985), *Fluid mechanics and transfer processes*, Cambridge University Press.

Marselle, D., Morari, M. & Rudd, D. F. (1982), 'Design of resilient processing plants: Design and control of energy management systems', *Chemical Engineering Science* **37**(2).

Mathisen, K. W., Morari, M. & Skogestad, S. (1994*a*), 'Dynamic models for heat exchangers and heat exchanger networks', *Computers and Chemical Engineering* **18**(1).

Mathisen, K. W., Morari, M. & Skogestad, S. (1994*b*), Optimal operation of heat exchanger networks, Presented at Process Systems Engineering (PSE '94), Kyongju, Korea.

Rathore, R. N. S. & Powers, G. J. (1975), 'A forward branching scheme for the synthesis of energy recovery', *Industrial Engineering Chemical Process Design Development* **14**(175).

Seborg, D. E., Edgar, T. F., Mellichamp, D. A. & Doyle, F. J. (2011), *Process Dynamics and Control*, 3 edn, John Wiley & Sons, Inc.

Sinnott, R. & Towler, G. (2009), *Chemical Engineering Design*, 5 edn, Coulson and Richardson's Chemical Engineering Series.

Skogestad, S. (2000), 'Plantwide control: the search for the self-optimizing control structure', *Journal of Process Control* **10**(5), 487–507.

Skogestad, S. (2003*a*), *Prosessteknikk*, 2 edn, Tapir Akademiske Forlag.

Skogestad, S. (2003*b*), 'Simple analytic rules for model reduction and pid controller design', *Journal of Process Control* **13**.

Skogestad, S. (2004), 'Near-optimal operation by self-optimizing control: from process control and marathon running and business systems', *Computers and Chemical Engineering* (29), 127–137.

Underwood, A. J. V. (1933), 'Graphical computation of logarithmic mean temperature difference', *Industrial Chemist and Chemical Manufacturer* (9), 167–170.

Wolff, E. A., Mathisen, K. W. & Skogestad, S. (1991), Dynamics and controllability of heat exchanger networks, Prepared for the COPE-91, Barcelona, Spain.

Zhang, D., Yang, Y., Pan, M. & Gao, Z. (2011), 'Toward a heat recovery chimney', *Sustainability* **3**(11), 2115–2128.

A Steady State Analysis

A.1 Four Heat Exchanger in Series and One in Parallel

Table A.1: Complete optimal and operating results for the 4:1 heat exchanger network

	Optimal operation	Jäschke temperature operation
T_{end} [°C]	207.87	207.84
u_1 [%]	64.15	70.66
$T_{1,1}$ [°C]	162.86	160.87
$T_{1,2}$ [°C]	178.44	176.35
$T_{1,3}$ [°C]	189.49	187.18
$T_{1,4}$ [°C]	207.33	204.80
$T_{2,1}$ [°C]	208.84	215.16
$Th_{1,1}^{out}$ [°C]	147.84	146.37
$Th_{1,2}^{out}$ [°C]	169.67	166.54
$Th_{1,3}^{out}$ [°C]	172.76	169.00
$Th_{1,4}^{out}$ [°C]	189.23	185.18
$Th_{2,1}^{out}$ [°C]	169.62	174.31

A.2 Six Heat Exchangers in Series and One in Parallel

The network of 6 heat exchanger in series parallel to one heat exchanger are shown in Figure A.1. The respective parameters are given in Table A.2 and the price constants are given in Table A.3.

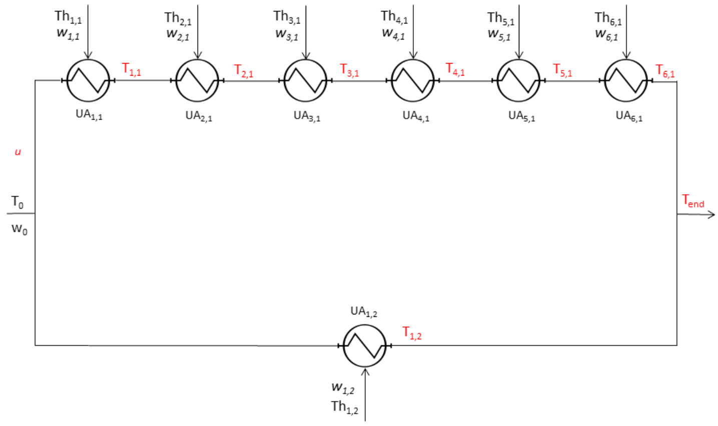


Figure A.1: The 6:1 heat exchanger network

Table A.2: Case parameters, 6 heat exchangers in series with one heat exchanger in parallel

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	190	[°C]
$Th_{2,1}$	203	[°C]
$Th_{3,1}$	220	[°C]
$Th_{4,1}$	235	[°C]
$Th_{5,1}$	240	[°C]
$Th_{6,1}$	245	[°C]
$Th_{1,2}$	225	[°C]
w_0	100	$[kW/°C]$
$w_{1,1}$	50	$[kW/°C]$
$w_{2,1}$	30	$[kW/°C]$
$w_{3,1}$	15	$[kW/°C]$
$w_{4,1}$	25	$[kW/°C]$
$w_{5,1}$	40	$[kW/°C]$
$w_{6,1}$	35	$[kW/°C]$
$w_{1,2}$	30	$[kW/°C]$
$UA_{1,1}$	5	$[kWm^2/°C]$
$UA_{2,1}$	7	$[kWm^2/°C]$
$UA_{3,1}$	10	$[kWm^2/°C]$
$UA_{4,1}$	12	$[kWm^2/°C]$
$UA_{5,1}$	9	$[kWm^2/°C]$
$UA_{6,1}$	8	$[kWm^2/°C]$
$UA_{1,2}$	11	$[kWm^2/°C]$

Table A.3: Price constants, six heat exchanger in series parallel to one heat exchanger

Parameter	Value	Unit
$P_{1,1}$	-1	$[\frac{\$}{kW}]$
$P_{2,1}$	-1.2	$[\frac{\$}{kW}]$
$P_{3,1}$	-1.3	$[\frac{\$}{kW}]$
$P_{4,1}$	-1.5	$[\frac{\$}{kW}]$
$P_{5,1}$	-1.4	$[\frac{\$}{kW}]$
$P_{6,1}$	-1.7	$[\frac{\$}{kW}]$
$P_{1,2}$	-1.4	$[\frac{\$}{kW}]$

Subject to the equality and inequality constraints given in Section 4.1, optimal operation was determined by the use of the build-in matlab function `fmincon`. Operation using the Jäschke temperature was also determined and compared to optimal operation. The results are given in the following Table A.4

Table A.4: Complete optimal and operating results for the case of six heat exchanger in series parallel to one heat exchanger

	Optimal operation	Jäschke temperature operation
T_{end} [°C]	226.27	226.27
u_1 [%]	85.53	89.06
$T_{1,1}$ [°C]	157.13	156.37
$T_{1,2}$ [°C]	172.11	171.20
$T_{1,3}$ [°C]	182.41	181.38
$T_{1,4}$ [°C]	199.48	198.30
$T_{1,5}$ [°C]	215.16	214.12
$T_{1,6}$ [°C]	224.43	233.56
$T_{2,1}$ [°C]	237.12	247.73
$Th_{1,1}^{out}$ [°C]	143.59	143.02
$Th_{1,2}^{out}$ [°C]	160.30	158.99
$Th_{1,3}^{out}$ [°C]	161.23	159.54
$Th_{1,4}^{out}$ [°C]	176.62	174.73
$Th_{1,5}^{out}$ [°C]	206.46	204.77
$Th_{1,6}^{out}$ [°C]	222.36	220.99
$Th_{2,1}^{out}$ [°C]	173.34	182.08

A.3 Two Heat Exchangers in Parallel

The following sections contains complete simulations results for different cases studied.

A.3.1 Case II-c

The following parameters applies to Case II-c, given in Table A.5. The results are given in Table A.6 and pictured in Figure A.2 and Figure A.3. Temperature loss due to measurement errors are given in Table A.9

Table A.5: Case II-c parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	203	[°C]
$Th_{1,2}$	248	[°C]
w_0	50	[kW/°C]
$w_{1,1}$	100	[kW/°C]
$w_{1,2}$	100	[kW/°C]
$UA_{1,1}$	10	[kWm^2/°C]
$UA_{1,2}$	30	[kWm^2/°C]

Table A.6: A selection of optimal and operating results for Case II-c

	Optimal operation	Jäschke temperature operation
T_{end} [°C]	184.96	184.95
u_1 [%]	21.30	20.00

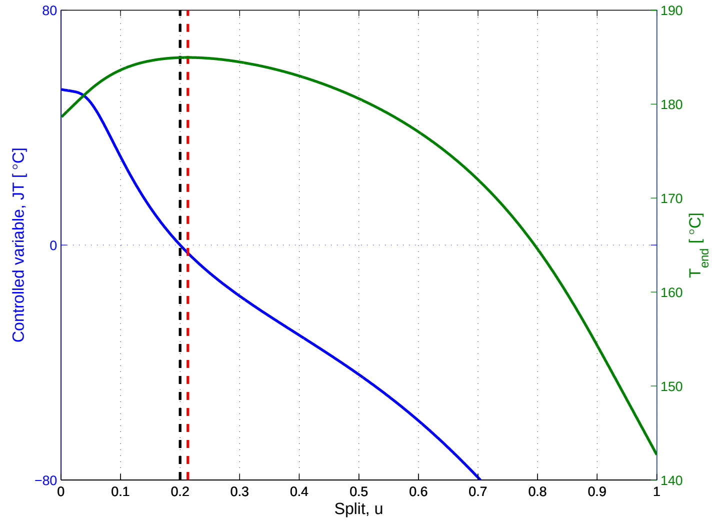


Figure A.2: T_{end} and control variable JT as a function of split u for case II-c. The red and black dotted lines shows optimal split considering outlet temperature and control variable, respectively

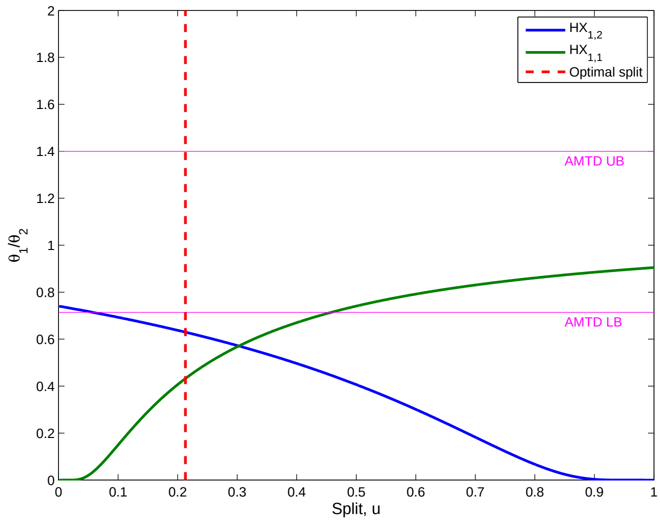


Figure A.3: AMTD approximation. $\frac{\theta_1}{\theta_2}$ as a function of split u for Case II-c

A.3.2 Case II-d

The following parameters applies to Case II-d, given in Table A.7. The results are given in Table A.8 and pictured in Figure A.4 and Figure A.5. Temperature loss due to measurement errors are given in Table A.9

Table A.7: Case II-d parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	203	[°C]
$Th_{1,2}$	248	[°C]
w_0	50	[kW/°C]
$w_{1,1}$	100	[kW/°C]
$w_{1,2}$	100	[kW/°C]
$UA_{1,1}$	100	[kWm^2/°C]
$UA_{1,2}$	300	[kWm^2/°C]

Table A.8: A selection of optimal and operating results for Case II-d

	Optimal operation	Jäschke temperature operation
T_{end} [°C]	206.11	204.90
u_1 [%]	40.70	30.90

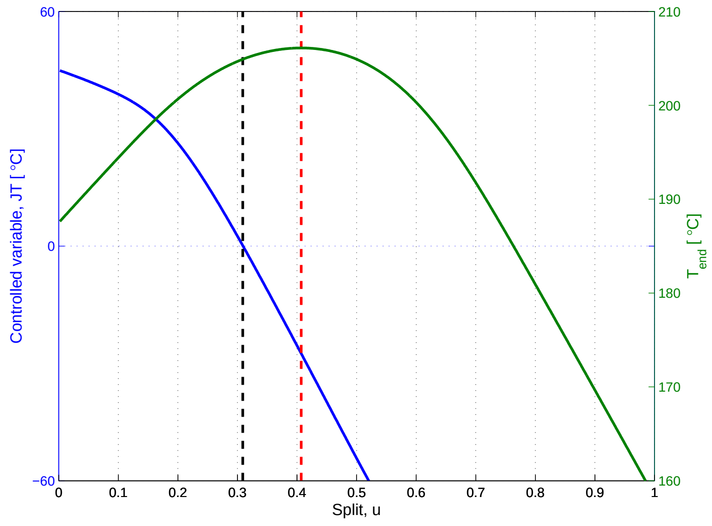


Figure A.4: T_{end} and control variable JT as a function of split u for case II-d. The red and black dotted lines shows optimal split considering outlet temperature and control variable, respectively

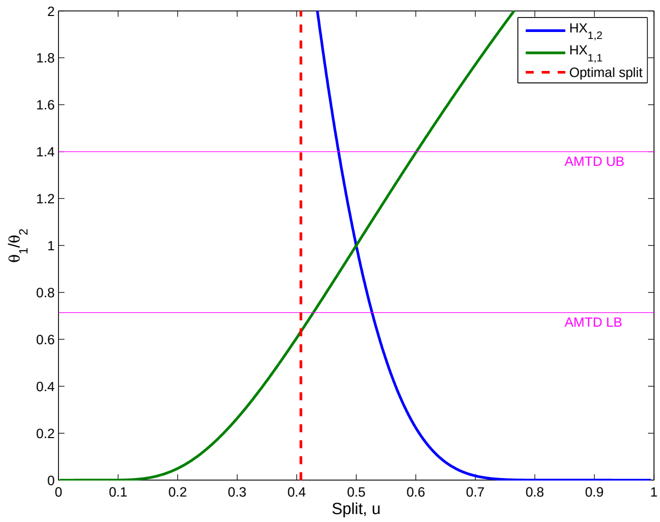


Figure A.5: AMTD approximation. $\frac{\theta_1}{\theta_2}$ as a function of split u for Case II-d

A.3.3 Jäschke Temperature and Measurement Errors

Table A.9: Temperature loss associated with measurement errors

Case	Worst case loss [°C]	Average loss [°C]
Case II-c	0.082	0.016
Case II-d	1.807	1.144

B Dynamic Analysis

Heat exchanger data valid for all heat exchangers in every case, are given in Table B.1

Table B.1: Heat exchanger and heat transfer data

Description	Symbol	Value	Unit
Total wall mass	m_{wall}	3000	$[kg]$
Wall density	ρ_{wall}	7850	$[\frac{kg}{m^3}]$
Wall volume	V_{wall}	0.3821	$[m^3]$
Heat capacity wall	Cp_{wall}	0.49	$[\frac{kW}{kg°\text{C}}]$
Density cold fluid	ρ_c	1000	$[\frac{kg}{m^3}]$

Selected plots are given for all cases modeled dynamically.

B.1 Dynamic case I

Estimated heat transfer variables are given in Table B.2 Inlet parameters for the dynamic Case II are given in Table B.3. Open loop and closed loop outlet variables are given in Table B.5 The PI controller was tuned using the Skogestad IMC (SIMC) rules (Skogestad 2003*b*) on a step response of 10 % increase in the cold fluid mass flow. The step response is shown in Figure B.1. The resulting tuning parameters are given in Table B.4, and filter parameters in Table B.6
The Simulink block diagram is given in Figure D.1 in Section D.

A negative step change in inlet cold stream temperature T_0 of 4 °C was introduced at time $t = 1000$ sec, and a positive step change in hot stream temperature $Th_{1,1}$ of 4 °C at time $t = 1600$ sec. Control variable response and split response are shown both with and without the analog filter in Figure B.2 and B.3. Outlet temperature responses with the analog filter implemented are shown in Figure B.4.

Table B.2: Heat transfer data Dynamic case I

Description	Symbol	Value	Unit
Heat transfer coefficient cold stream	h_c	0.17	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (1,1)	$h_{1,1}$	0.223	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (1,2)	$h_{1,2}$	0.187	$[\frac{kW}{°Cm^2}]$
Area heat exchanger (1,1)	$A_{1,1}$	250	$[m^2]$
Area heat exchanger (1,2)	$A_{1,2}$	700	$[m^2]$

Table B.3: Dynamic Case I parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	203	[°C]
$Th_{1,2}$	248	[°C]
w_0	95	$[kW/°C]$
$w_{1,1}$	60	$[kW/°C]$
$w_{1,2}$	65	$[kW/°C]$
$UA_{1,1}$	24.10	$[kWm^2/°C]$
$UA_{1,2}$	62.33	$[kWm^2/°C]$

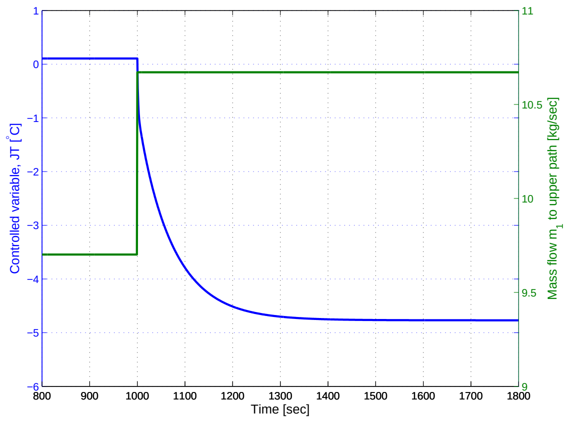


Figure B.1: Open loop step response of control variable JT on a 10 % increase in inlet mass flow m_1 for Dynamic Case I

Table B.4: PI tuning parameters for Case II

Tuning parameter	Value	Unit
K_c	5.97	$[\frac{°C}{kg/s}]$
τ_I	10	[sec]

Table B.5: Open loop and closed loop operating variables for Dynamic Case I

Operating variable	Open loop value	Closed loop value
$T_{1,1}$ [°C]	199.2	199.2
$T_{1,2}$ [°C]	217.9	218.0
$Th_{1,1}^{out}$ [°C]	175.0	174.9
$Th_{1,2}^{out}$ [°C]	152.3	152.3
u	0.2553	0.2559
T_{end} [°C]	213.2	213.2

Table B.6: Analog filter parameters for Dynamic Case I

Filter parameter	Value	Unit
K_f	13	$[\frac{°C}{kg/s}]$
τ_I	60	[sec]

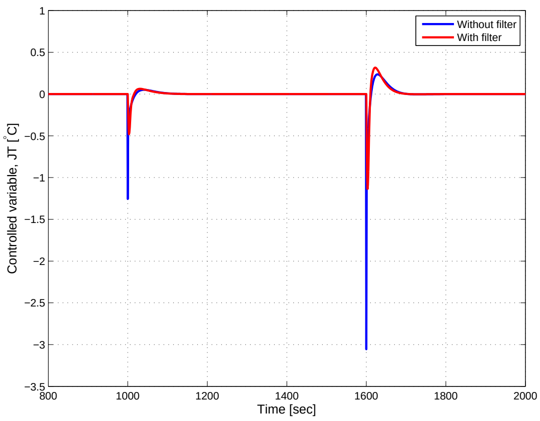


Figure B.2: Response of control variable JT when T_0 is decreased and $Th_{2,1}$ increased 4 °C at t = 1000 and 1600 sec, respectively

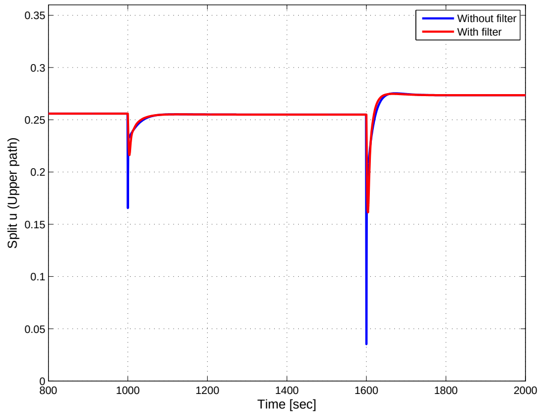


Figure B.3: Response of split u when T_0 is decreased and $Th_{2,1}$ increased 4 °C at $t = 1000$ and 1600 sec, respectively

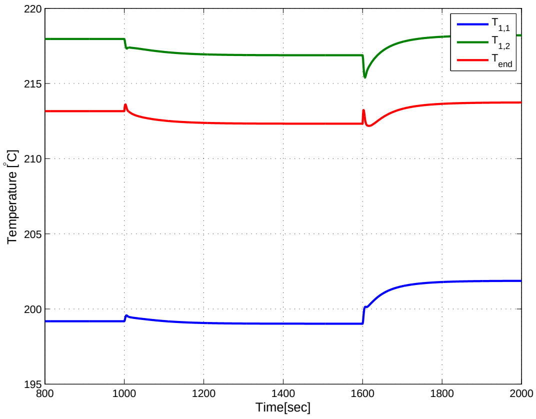


Figure B.4: Response of outlet temperatures when T_0 is decreased and $Th_{2,1}$ increased 4 °C at $t = 1000$ and 1600 sec, respectively

B.2 Dynamic case II

Inlet parameters, outlet variables, tuning parameter, filter parameters and Simulink block diagram were given i Section 7.
Estimated heat transfer variables are given in Table B.7

Table B.7: Heat transfer data Dynamic case II

Description	Symbol	Value	Unit
Heat transfer coefficient cold stream	h_c	0.10	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (1,1)	$h_{1,1}$	0.109	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (2,1)	$h_{2,1}$	0.103	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (1,2)	$h_{1,2}$	0.107	$[\frac{kW}{°Cm^2}]$
Area heat exchanger (1,1)	$A_{1,1}$	341	$[m^2]$
Area heat exchanger (2,1)	$A_{2,1}$	616	$[m^2]$
Area heat exchanger (1,2)	$A_{1,2}$	1118	$[m^2]$

A negative step change in inlet cold stream temperature T_0 of 4 °C was introduced at time $t = 1000$ sec, and a positive step change in hot stream temperature $Th_{1,1}$ of 4 °C at time $t = 2000$ sec. Control variable response and split response are shown both with and without the analog filter in Figure B.5 and B.6. Outlet temperature responses with the analog filter implemented are shown in Figure B.7.

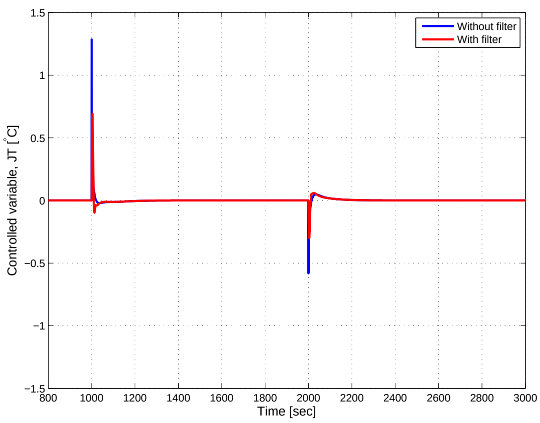


Figure B.5: Response of control variable JT when T_0 is decreased and $Th_{1,1}$ increased 4 °C at $t = 1000$ and 2000 sec, respectively

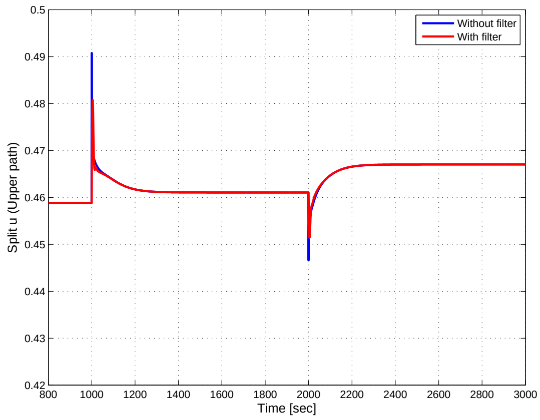


Figure B.6: Response of split u when T_0 is decreased and $Th_{1,1}$ increased 4 °C at $t = 1000$ and 2000 sec, respectively

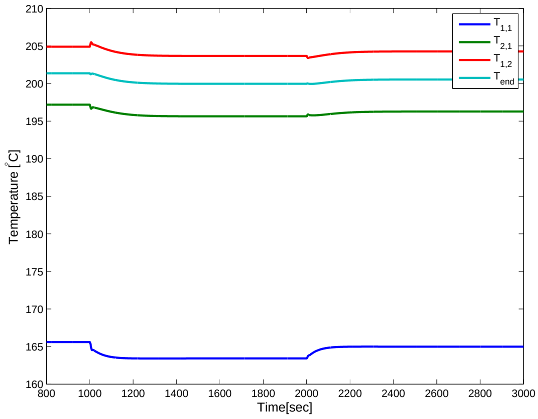
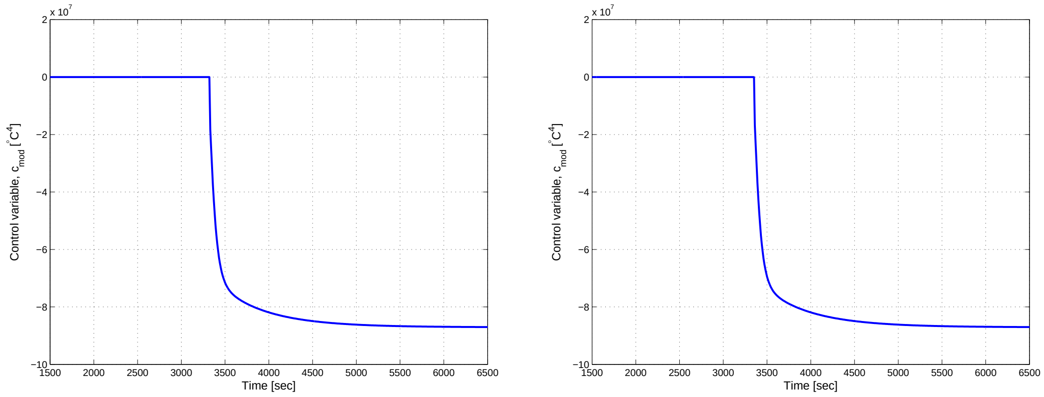


Figure B.7: Response of outlet temperatures when T_0 is decreased and $Th_{1,1}$ increased 4 °C at $t = 1000$ and 2000 sec, respectively

B.3 Dynamic Case II-a

The following figure shows the complete plot of control variable response in the case of a decaying hot stream temperature $Th_{2,1}$ (Extended plot of Figure 7.10). The full Simulink block diagram are given in Figure D.3 in Section D.



(a) c_{mod} with tuning set 1

(b) c_{mod} with tuning set 2

Figure B.8: Full plot of modified control variable c_{mod} as a function of time t when $Th_{2,1}$ is decreased from 255 - 180 °C from time t = 2000 - 6000 sec

B.4 Dynamic Case III

The network of 6 heat exchanger in series parallel to one heat exchanger are shown in Figure B.9. Estimated heat transfer variables are given in Table B.8. The respective parameters are given in Table B.9.

Table B.8: Heat transfer data Dynamic case III

Description	Symbol	Value	Unit
Heat transfer coefficient cold stream	h_c	0.10	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (1,1)	$h_{1,1}$	0.111	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (2,1)	$h_{2,1}$	0.109	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (3,1)	$h_{3,1}$	0.107	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (1,2)	$h_{1,2}$	0.107	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (2,2)	$h_{2,2}$	0.100	$[\frac{kW}{°Cm^2}]$
Area heat exchanger (1,1)	$A_{1,1}$	112.5	$[m^2]$
Area heat exchanger (2,1)	$A_{2,1}$	102	$[m^2]$
Area heat exchanger (3,1)	$A_{3,1}$	85	$[m^2]$
Area heat exchanger (1,2)	$A_{1,2}$	800	$[m^2]$
Area heat exchanger (2,2)	$A_{2,2}$	765	$[m^2]$

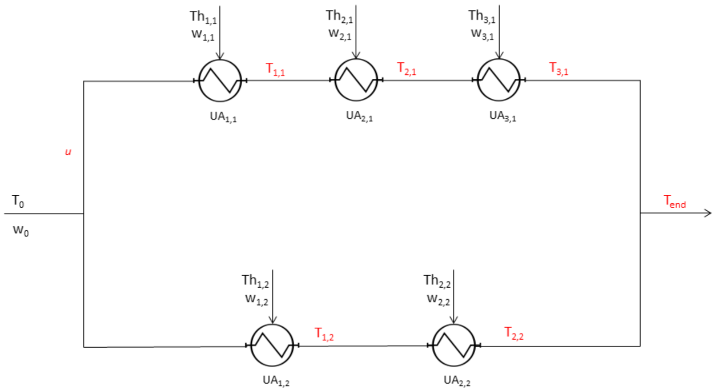


Figure B.9: Dynamic case III: Three heat exchangers in series parallel with two heat exchangers

Open loop and closed loop outlet variables are given in Table B.11 The PI controller was tuned using the Skogestad IMC (SIMC) rules (Skogestad 2003*b*) on a step response of 10 % increase in the cold fluid mass flow. The step response is shown in Figure B.10. The resulting tuning parameters are given in Table B.10, and filter parameters in Table B.12
The Simulink block diagram is given in Figure D.4 in Section D.

A negative step change in inlet cold stream temperature T_0 of 4 °C was introduced at time $t = 1000$ sec, and a positive step change in hot stream temperature $Th_{1,2}$ of 4 °C at time $t = 2000$ sec. Control variable response and split response are shown both with and without the analog filter in Figure B.11 and B.12. Outlet temperature responses with the analog filter implemented are shown in Figure B.13.

Table B.9: Dynamic case III parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	190	[°C]
$Th_{2,1}$	203	[°C]
$Th_{3,1}$	220	[°C]
$Th_{1,2}$	220	[°C]
$Th_{2,2}$	248	[°C]
w_0	150	[kW/°C]
$w_{1,1}$	50	[kW/°C]
$w_{2,1}$	30	[kW/°C]
$w_{3,1}$	15	[kW/°C]
$w_{1,2}$	70	[kW/°C]
$w_{1,2}$	20	[kW/°C]
$UA_{1,1}$	5.92	[kWm^2/°C]
$UA_{2,1}$	5.31	[kWm^2/°C]
$UA_{3,1}$	4.39	[kWm^2/°C]
$UA_{1,2}$	41.32	[kWm^2/°C]
$UA_{2,2}$	38.25	[kWm^2/°C]

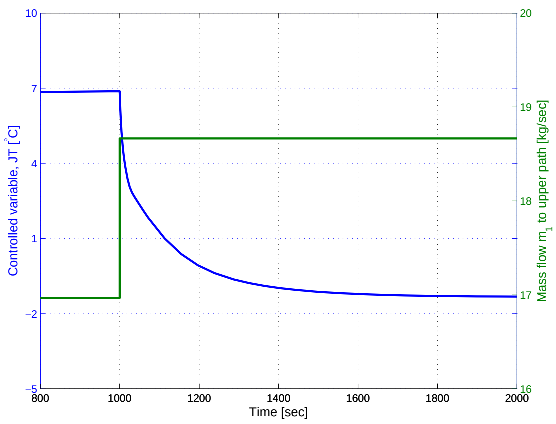


Figure B.10: Open loop step response of control variable JT on a 10 % increase in inlet mass flow m_1 for Dynamic case III

Table B.10: PI tuning parameters for Dynamic case III

Tuning parameter	Value	Unit
K_c	1.44	$[\frac{°C}{kg/s}]$
τ_I	40	[sec]

Table B.11: Open loop and closed loop operating variables for Dynamic case III

Operating variable	Open loop value	Closed loop value
$T_{1,1}$ [°C]	154.2	154.7
$T_{2,1}$ [°C]	170.7	168.6
$T_{3,1}$ [°C]	182.5	180.1
$T_{1,2}$ [°C]	176.6	177.8
$T_{2,2}$ [°C]	189.8	191.2
$Th^{out}_{1,1}$ [°C]	169.5	169.2
$Th^{out}_{2,1}$ [°C]	179.7	178.7
$Th^{out}_{3,1}$ [°C]	186.7	185.1
$Th^{out}_{1,2}$ [°C]	148.3	148.7
$Th^{out}_{2,2}$ [°C]	176.9	178.1
u	0.2828	0.3063
T_{end} [°C]	187.7	187.8

Table B.12: Analog filter parameters for Dynamic case III

Filter parameter	Value	Unit
K_f	1.5	$[\frac{°C}{kg/s}]$
τ_I	85	[sec]

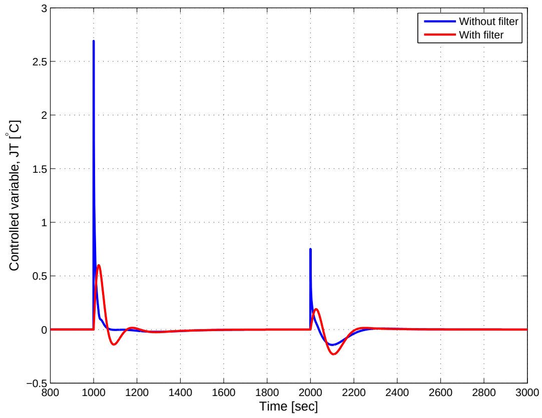


Figure B.11: Response of control variable JT when T_0 is decreased and $Th_{1,2}$ increased 4 °C at $t = 1000$ and 2000 sec, respectively

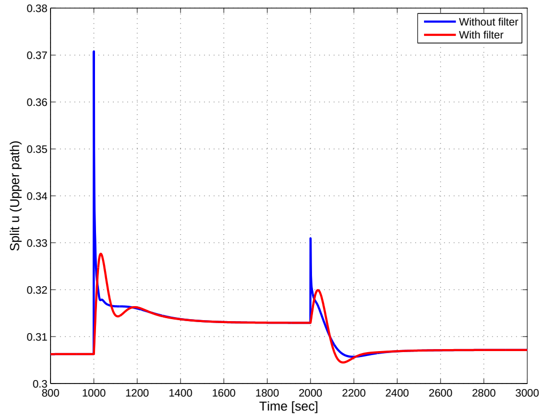


Figure B.12: Response of split u when T_0 is decreased and $Th_{1,2}$ increased 4 °C at $t = 1000$ and 2000 sec, respectively

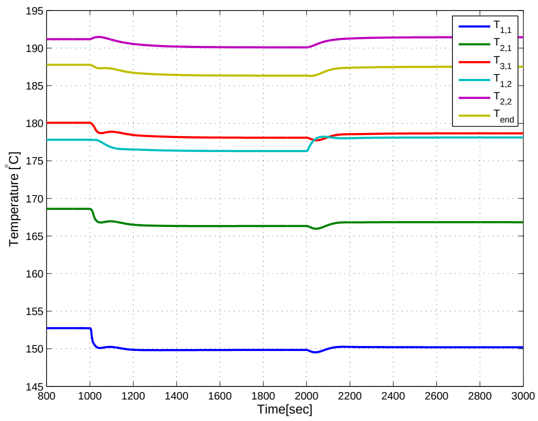


Figure B.13: Response of outlet temperatures when T_0 is decreased and $Th_{1,2}$ increased 4 °C at $t = 1000$ and 2000 sec, respectively

B.5 Dynamic Case IV

Different from the case studied in Section 6.1, h and A were estimated such that the dynamic open loop outlet variables matched the steady state outlet variables found by using the AMTD approximation, rather than the Underwood approximation. Therefore, the estimated UA values for the dynamic analysis are *smaller* than the UA values used in the steady state analysis. For the same reason, also each outlet temperature are lower than what was seen in Section 6.1.
Estimated heat transfer variables are given in Table B.13. The respective parameters are given in Table B.14.

Table B.13: Heat transfer data Dynamic case IV

Description	Symbol	Value	Unit
Heat transfer coefficient cold stream	h_c	0.10	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (1,1)	$h_{1,1}$	0.120	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (2,1)	$h_{2,1}$	0.142	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (3,1)	$h_{3,1}$	0.139	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (4,1)	$h_{4,1}$	0.070	$[\frac{kW}{°Cm^2}]$
Heat transfer coefficient hot stream (1,2)	$h_{1,2}$	0.143	$[\frac{kW}{°Cm^2}]$
Area heat exchanger (1,1)	$A_{1,1}$	19	$[m^2]$
Area heat exchanger (2,1)	$A_{2,1}$	29.5	$[m^2]$
Area heat exchanger (3,1)	$A_{3,1}$	43.7	$[m^2]$
Area heat exchanger (1,2)	$A_{4,1}$	103	$[m^2]$
Area heat exchanger (2,2)	$A_{1,2}$	38.3	$[m^2]$

The open loop and closed loop outlet variables are given in Table B.16.

The PI controller was tuned using the Skogestad IMC (SIMC) rules (Skogestad 2003*b*) on a step response of 10 % increase in the cold fluid mass flow. The step response is shown in Figure B.14. The resulting tuning parameters are given in Table B.15. Analog filter was not implemented for this case.
The Simulink block diagram is given in Figure D.5 in Section D.

A positive step change in hot stream temperature $Th_{1,1}$ of 4 °C was introduced at time t = 1000 sec, a negative step change in hot stream temperature $Th_{3,1}$ of 4 °C at time t = 2000 sec and a positive step change in hot stream temperature

$Th_{1,2}$ of 4 °C at time $t = 3000$ sec. Control variable response and split response are shown in Figure B.15 and B.16. Outlet temperature responses are shown in Figure B.17.

Table B.14: Dynamic case IV parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	190	[°C]
$Th_{2,1}$	203	[°C]
$Th_{3,1}$	220	[°C]
$Th_{4,1}$	235	[°C]
$Th_{1,2}$	210	[°C]
w_0	130	[kW/°C]
$w_{1,1}$	50	[kW/°C]
$w_{2,1}$	30	[kW/°C]
$w_{3,1}$	15	[kW/°C]
$w_{4,1}$	25	[kW/°C]
$w_{1,2}$	70	[kW/°C]
$UA_{1,1}$	1.23	[kWm^2/°C]
$UA_{2,1}$	1.73	[kWm^2/°C]
$UA_{3,1}$	2.54	[kWm^2/°C]
$UA_{4,1}$	4.24	[kWm^2/°C]
$UA_{1,2}$	2.25	[kWm^2/°C]

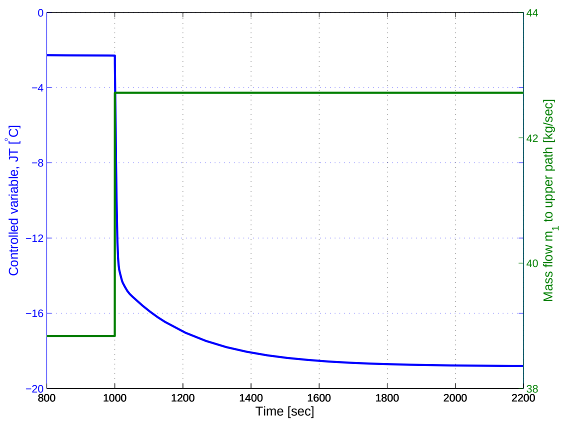


Figure B.14: Open loop step response of control variable JT on a 10 % increase in inlet mass flow m_1 for Dynamic case IV

Table B.15: PI tuning parameters for Dynamic case IV

Tuning parameter	Value	Unit
K_c	2.05	$[\frac{°C}{kg/s}]$
τ_I	10	[sec]

Table B.16: Open loop and closed loop operating variables for Dynamic case IV

Operating variable	Open loop value	Closed loop value
$T_{1,1}$ [°C]	133.6	133.6
$T_{2,1}$ [°C]	139.0	139.0
$T_{3,1}$ [°C]	146.4	146.4
$T_{4,1}$ [°C]	156.8	156.8
$T_{1,2}$ [°C]	155.5	155.5
$Th_{1,1}^{out}$ [°C]	184.4	184.4
$Th_{2,1}^{out}$ [°C]	189.0	189.0
$Th_{3,1}^{out}$ [°C]	181.3	181.3
$Th_{4,1}^{out}$ [°C]	202.6	202.6
$Th_{1,2}^{out}$ [°C]	201.8	201.8
u	0.7767	0.7763
T_{end} [°C]	156.5	156.5

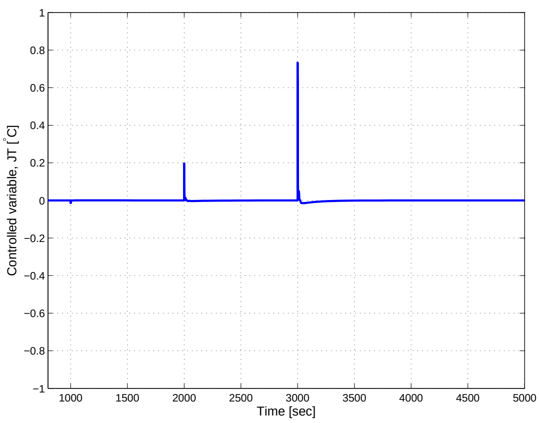


Figure B.15: Response of control variable JT when $Th_{1,1}$ is increased, $Th_{3,1}$ decreased and $Th_{1,2}$ increased 4 °C at t = 1000, 2000 and 3000 sec, respectively

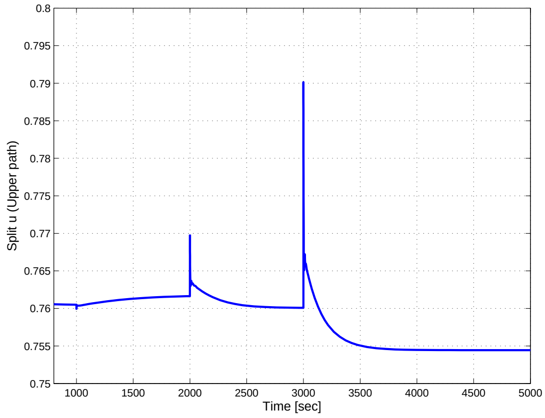


Figure B.16: Response of split u when $Th_{1,1}$ is increased, $Th_{3,1}$ decreased and $Th_{1,2}$ increased 4 °C at t = 1000, 2000 and 3000 sec, respectively

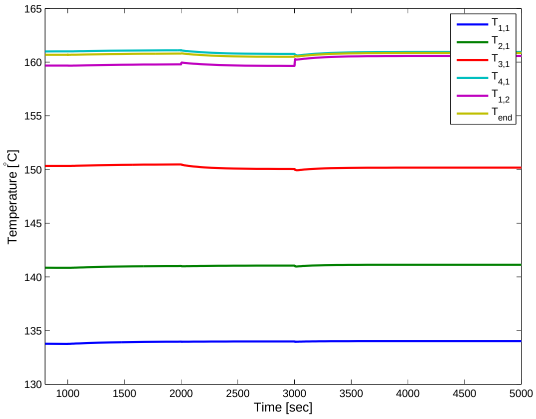


Figure B.17: Response of outlet temperatures when $Th_{1,1}$ is increased, $Th_{3,1}$ decreased and $Th_{1,2}$ increased 4 °C at t = 1000, 2000 and 3000 sec, respectively

B.6 Dynamic Case V

Inlet parameters for Case VI are given in Table A.2.

As for the simulation in Section **??**, h and A were estimated such that the dynamic open loop outlet variables matched the steady state outlet variables found by using the AMTD approximation, rather than the Underwood approximation. Therefore, the estimated UA values for the dynamic analysis are *smaller* than the UA values used in the steady state analysis.

Estimated heat transfer variables are given in Table B.13. The respective parameters are given in Table B.18.

Table B.17: Heat transfer data Dynamic case V

Description	Symbol	Value	Unit
Heat transfer coefficient cold stream	h_c	0.10	$[\frac{kW}{^\circ C m^2}]$
Heat transfer coefficient hot stream (1,1)	$h_{1,1}$	0.110	$[\frac{kW}{^\circ C m^2}]$
Heat transfer coefficient hot stream (2,1)	$h_{2,1}$	0.108	$[\frac{kW}{^\circ C m^2}]$
Heat transfer coefficient hot stream (3,1)	$h_{3,1}$	0.108	$[\frac{kW}{^\circ C m^2}]$
Heat transfer coefficient hot stream (4,1)	$h_{4,1}$	0.107	$[\frac{kW}{^\circ C m^2}]$
Heat transfer coefficient hot stream (5,1)	$h_{5,1}$	0.110	$[\frac{kW}{^\circ C m^2}]$
Heat transfer coefficient hot stream (6,1)	$h_{6,1}$	0.110	$[\frac{kW}{^\circ C m^2}]$
Heat transfer coefficient hot stream (1,2)	$h_{1,2}$	0.110	$[\frac{kW}{^\circ C m^2}]$
Area heat exchanger (1,1)	$A_{1,1}$	20.50	$[m^2]$
Area heat exchanger (2,1)	$A_{2,1}$	23.30	$[m^2]$
Area heat exchanger (3,1)	$A_{3,1}$	42.60	$[m^2]$
Area heat exchanger (4,1)	$A_{4,1}$	49.95	$[m^2]$
Area heat exchanger (5,1)	$A_{5,1}$	36.50	$[m^2]$
Area heat exchanger (6,1)	$A_{6,1}$	32.50	$[m^2]$
Area heat exchanger (1,2)	$A_{1,2}$	43.50	$[m^2]$

The open loop and closed loop outlet variables are given in Table B.20.

The PI controller was tuned using the Skogestad IMC (SIMC) rules (Skogestad 2003*b*) on a step response of 10 % increase in the cold fluid mass flow. The step response is shown in Figure B.18. The resulting tuning parameters are given in Table B.19. Analog filter was not implemented for this case.
The Simulink block diagram is given in Figure D.6 in Section D.

A positive step change in hot stream temperature $Th_{1,1}$ of 4 °C was introduced at time t = 1000 sec, a negative step change in hot stream temperature $Th_{6,1}$ of 4 °C at time t = 2000 sec and a positive step change in hot stream temperature $Th_{1,2}$ of 4 °C at time t = 3000 sec. Control variable response and split response are shown in Figure B.19 and B.20. Outlet temperature responses are shown in Figure B.21.

Table B.18: Dynamic case V parameters

Parameter	Value	Unit
T_0	130	[°C]
$Th_{1,1}$	190	[°C]
$Th_{2,1}$	203	[°C]
$Th_{3,1}$	220	[°C]
$Th_{4,1}$	235	[°C]
$Th_{5,1}$	240	[°C]
$Th_{6,1}$	245	[°C]
$Th_{1,2}$	225	[°C]
w_0	100	$[kW/°C]$
$w_{1,1}$	50	$[kW/°C]$
$w_{2,1}$	30	$[kW/°C]$
$w_{3,1}$	15	$[kW/°C]$
$w_{4,1}$	25	$[kW/°C]$
$w_{5,1}$	40	$[kW/°C]$
$w_{6,1}$	35	$[kW/°C]$
$w_{1,2}$	30	$[kW/°C]$
$UA_{1,1}$	1.07	$[kWm^2/°C]$
$UA_{2,1}$	1.47	$[kWm^2/°C]$
$UA_{3,1}$	2.21	$[kWm^2/°C]$
$UA_{4,1}$	2.58	$[kWm^2/°C]$
$UA_{5,1}$	1.91	$[kWm^2/°C]$
$UA_{6,1}$	1.70	$[kWm^2/°C]$
$UA_{1,2}$	2.39	$[kWm^2/°C]$

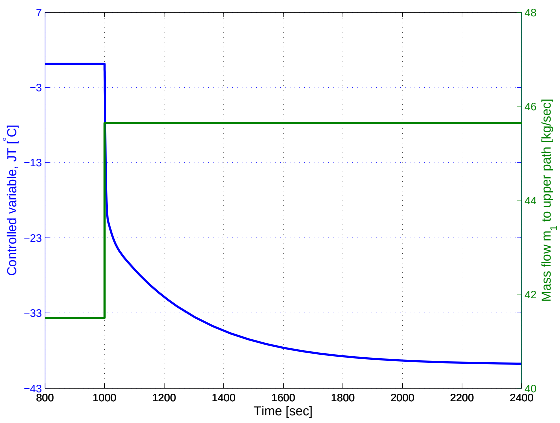


Figure B.18: Open loop step response of control variable JT on a 10 % increase in inlet mass flow m_1 for Case VI

Table B.19: PI tuning parameters for Dynamic case V

Tuning parameter	Value	Unit
K_c	1.18	$[\frac{°C}{kg/s}]$
τ_I	40	[sec]

Table B.20: Open loop and closed loop operating variables for Dynamic case V

Operating variable	Open loop value	Closed loop value
$T_{1,1}$ [°C]	133.4	133.4
$T_{2,1}$ [°C]	138.4	138.4
$T_{3,1}$ [°C]	145.5	145.5
$T_{4,1}$ [°C]	155.3	155.3
$T_{5,1}$ [°C]	163.2	163.1
$T_{6,1}$ [°C]	170.0	170.0
$T_{1,2}$ [°C]	170.7	170.8
$Th_{1,1}^{out}$ [°C]	184.4	184.4
$Th_{2,1}^{out}$ [°C]	189.0	189.0
$Th_{3,1}^{out}$ [°C]	181.0	181.0
$Th_{4,1}^{out}$ [°C]	202.2	202.1
$Th_{5,1}^{out}$ [°C]	223.7	223.7
$Th_{6,1}^{out}$ [°C]	228.9	228.9
$Th_{1,2}^{out}$ [°C]	201.8	201.8
u	0.8299	0.8304
T_{end} [°C]	170.1	170.1

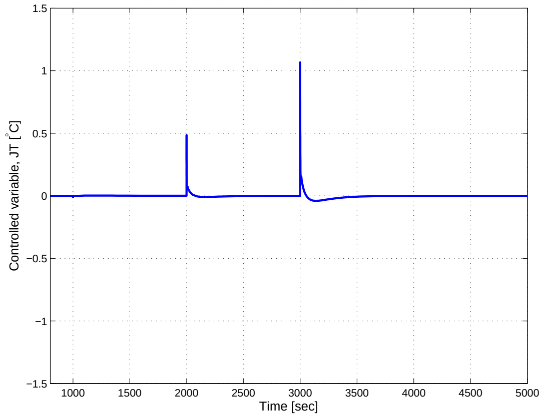


Figure B.19: Response of control variable JT when $Th_{1,1}$ is increased, $Th_{6,1}$ decreased and $Th_{1,2}$ increased 4 °C at t = 1000, 2000 and 3000 sec, respectively

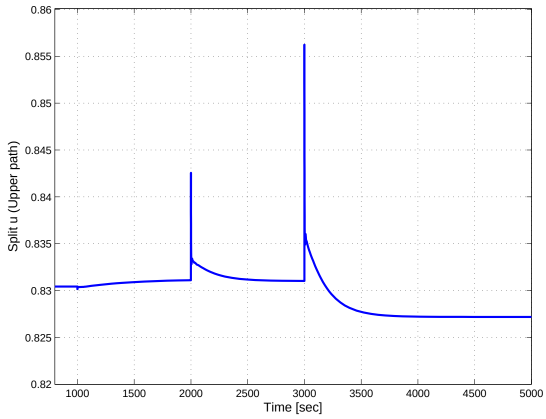


Figure B.20: Response of split u when $Th_{1,1}$ is increased, $Th_{6,1}$ decreased and $Th_{1,2}$ increased 4 °C at t = 1000, 2000 and 3000 sec, respectively

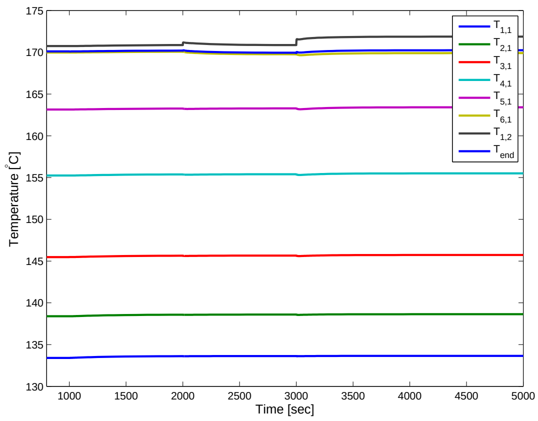


Figure B.21: Response of outlet temperatures when $Th_{1,1}$ is increased, $Th_{6,1}$ decreased and $Th_{1,2}$ increased 4 °C at t = 1000, 2000 and 3000 sec, respectively

C Matlab Scripts

C.1 Steady State Analysis Scripts

Case I: Four Heat Exchangers in Series and One in Parallel

RunHEN_41.m

```
%% Model to simulate a steady state 4:1 HEN
% Topology to be investigated:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            1     2     3     4           %
%        ----0-----0-----0-----0----       %
%   ----|                           |----  %
%        -------------0-------------       %
%                     5                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


close all;
clear all;
clc;

%% Parameters

% Heat Capacity rates
par.w0  = 100;   %[kW/degC] w= miCpi
par.wh1 = 50;    %[kW/degC]
par.wh2 = 30;    %[kW/degC]
par.wh3 = 15;    %[kW/degC]
par.wh4 = 25;    %[kW/degC]
par.wh5 = 70;    %[kW/degC]

% Hot streams inlet temperature
par.Th1 = 190;  %[degC]
par.Th2 = 203;  %[degC]
par.Th3 = 220;  %[degC]
par.Th4 = 235;  %[degC]
par.Th5 = 210;  %[degC]

```

```
% Cold stream inlet temperature
par.T0 = 130;   %[degC]

% UA values for each heat exchanger
par.UA1 = 5;    %[kWm2/degC]
par.UA2 = 7;    %[kWm2/degC]
par.UA3 = 10;   %[kWm2/degC]
par.UA4 = 12;   %[kWm2/degC]
par.UA5 = 9;    %[kWm2/degC]

% Operating prices for each heat exchanger
par.P1 = 1;     %[$/kW]
par.P2 = 1.2;   %[$/kW]
par.P3 = 1.3;   %[$/kW]
par.P4 = 1.5;   %[$/kW]
par.P5 = 1.4;   %[$/kW]

%Inequality constraint
par.DeltaTmin = 0.5; %[degC]

% Scaling vector
par.sc.x = [200*ones(11,1);100;100;1000*ones(5,1)];
par.sc.j = 200;

% Defining parameters
Th1 = par.Th1; Th2 = par.Th2; Th3 = par.Th3; Th4 = par.Th4; ...
    Th5 = par.Th5;
T0 = par.T0;

%% OPTIMAL OPERATION

% Guessing outlet variables
% x0 = [Tend T1 T2 T3 T4 T5 Th1out Th2out Th3out Th4out Th5out ...
    w1 w2 ...
%       [Q1 Q2 Q3 Q4 Q5]

x0    = [138 131 133 138 138 140 188 198 200 215 190 60 40 ...
          59 137 297 333 200]';
% x0    = [207 160 176 187 204 215 146 166 169 185 174 71 29 ...
%            1.9224e+03 778.4439 581.1345 921.1994 3.3767e+03]';

```

```
% Scaling variables
% x0 = x0./par.sc.x;

% Minimizing cost function based on equality constraints
% using fmincon
A = []; b = []; Aeq = []; Beq = [];
LB = 0*ones(23,1); UB = inf*ones(23,1);

options = ...
    optimset('Algorithm','interior-point','display','iter',...
     'MaxFunEvals',9000,'TolCon',1e-12,'TolX',1e-12);

options = optimset('Algorithm','active-set','display','iter',...
     'MaxFunEvals',9000,'TolCon',1e-11,'TolX',1e-11);

options = optimset('display','iter',...
     'MaxFunEvals',9000,'TolCon',1e-10,'TolX',1e-10);

[x,J,exitflag] = fmincon(@(x)Object_41(x,par),x0,A,b,Aeq,Beq,...
     LB,UB,@(x)HEN_Constraints_41(x,par),options);
exitflag

% Unscaling variables
% x = x.*par.sc.x;


% RESULTS
% Outlet temperatures
Tend = x(1);
T1 = x(2); T2 = x(3); T3 = x(4); T4 = x(5); T5 = x(6);
Th1out = x(7); Th2out = x(8); Th3out = x(9); Th4out = x(10);
Th5out = x(11);
% Split
w1 = x(12); w2 = x(13);
% Heat transfer
Q1 = x(14); Q2 = x(15); Q3 = x(16); Q4 = x(17); Q5 = x(18);
% Split ratio
w1_rat = w1/par.w0;
w2_rat = w2/par.w0;
```

```
% Delta Ts
DeltaT_hot1 = Th1 - T1;
DeltaT_hot2 = Th2 - T2;
DeltaT_hot3 = Th3 - T3;
DeltaT_hot4 = Th4 - T4;
DeltaT_hot5 = Th5 - T5;
DeltaT_cold1 = Th1out - T0;
DeltaT_cold2 = Th2out - T1;
DeltaT_cold3 = Th3out - T2;
DeltaT_cold4 = Th4out - T3;
DeltaT_cold5 = Th5out - T0;

% Displaying the results
display(['  Tend [degC] = '])
disp(Tend)
display(['   T1           T2          T3         T4         T5   ...
    [degC]'])
disp([T1 T2 T3 T4 T5])
display(['    Th1out     Th2out     Th3out     Th4out     Th5out    ...
    [degC]'])
disp([Th1out Th2out Th3out Th4out Th5out])
display(['      w1         w2'])
disp([w1 w2])
display(['  w1 ratio   w2 ratio [%]'])
disp([w1_rat w2_rat])
display([' DeltaT hot side '])
display(['    HX1         HX2        HX3        HX4        HX5    '])
disp([DeltaT_hot1 DeltaT_hot2 DeltaT_hot3 DeltaT_hot4 ...
    DeltaT_hot5])
display([' DeltaT cold side '])
display(['    HX1         HX2        HX3        HX4        HX5    '])
disp([DeltaT_cold1 DeltaT_cold2 DeltaT_cold3 DeltaT_cold4 ...
    DeltaT_cold5])


%% OPERATION USING THE JAESCHKE TEMPERATURE

% Guessing outlet variables
% x0 = [Tend T1 T2 T3 T4 T5 Th1out Th2out Th3out Th4out Th5out ...
    w1 w2...
%       [Q1 Q2 Q3 Q4 Q5]
```

```
x0    = [138 131 133 138 138 140 188 198 200 215 190 60 40 ...
          59 137 297 333 200]';

% Scaling variables
% x0 = x0./par.sc.x;

% Defining parameters
Th1 = par.Th1; Th2 = par.Th2; Th3 = par.Th3; Th4 = par.Th4; ...
    Th5 = par.Th5;
T0 = par.T0;


% Minimizing cost function based on equality constraints and ...
    Jaeschke temp
% using fmincon
A = []; b = []; Aeq = []; Beq = [];
LB = 0*ones(23,1); UB = inf*ones(23,1);

options = ...
    optimset('Algorithm','interior-point','display','iter',...
     'MaxFunEvals',9000,'TolCon',1e-12,'TolX',1e-12);

options = optimset('Algorithm','active-set','display','iter',...
     'MaxFunEvals',9000,'TolCon',1e-11,'TolX',1e-11);

options = optimset('display','iter',...
     'MaxFunEvals',9000,'TolCon',1e-10,'TolX',1e-10);

[xDJT,J,exitflag] = ...
    fmincon(@(x)Object_41(x,par),x0,A,b,Aeq,Beq,...
     LB,UB,@(x)HEN_Constraints_41_DJT(x,par),options);
exitflag

%Unscaling variables
% xDJT = xDJT.*par.sc.x;


% RESULTS
% Outlet temperatures
Tend_DJT = xDJT(1);
```

```
T1_DJT = xDJT(2); T2_DJT = xDJT(3); T3_DJT = xDJT(4); T4_DJT = ...
    xDJT(5);
T5_DJT = xDJT(6);
Th1out_DJT = xDJT(7); Th2out_DJT = xDJT(8); Th3out_DJT = xDJT(9);
Th4out_DJT = xDJT(10); Th5out_DJT = xDJT(11);
% Split
w1_DJT = xDJT(12); w2_DJT = xDJT(13);
% Heat transfer
Q1_DJT = xDJT(14); Q2_DJT = xDJT(15); Q3_DJT = xDJT(16); ...
    Q4_DJT = xDJT(17);
% Split ratio
w1_rat_DJT = w1_DJT/par.w0;
w2_rat_DJT = w2_DJT/par.w0;
% Delta Ts
DeltaT_hot1_DJT = Th1 - T1_DJT;
DeltaT_hot2_DJT = Th2 - T2_DJT;
DeltaT_hot3_DJT = Th3 - T3_DJT;
DeltaT_hot4_DJT = Th4 - T4_DJT;
DeltaT_hot5_DJT = Th5 - T5_DJT;
DeltaT_cold1_DJT = Th1out_DJT - T0;
DeltaT_cold2_DJT = Th2out_DJT - T1_DJT;
DeltaT_cold3_DJT = Th3out_DJT - T2_DJT;
DeltaT_cold4_DJT = Th4out_DJT - T3_DJT;
DeltaT_cold5_DJT = Th5out_DJT - T0;

% Displaying the results
display(['  Tend DJT [degC] = '])
disp(Tend_DJT)
display(['   T1 DJT   T2 DJT     T3 DJT     T4 DJT     T5 DJT  ...
    [degC]'])
disp([T1_DJT T2_DJT T3_DJT T4_DJT T5_DJT])
display(['Th1out DJT  Th2out DJT Th3out DJT Th4out DJT Th5out ...
    DJT  [degC]'])
disp([Th1out_DJT Th2out_DJT Th3out_DJT Th4out_DJT Th5out_DJT])
display(['    w1 DJT     w2 DJT'])
disp([w1_DJT w2_DJT])
display(['  w1 ratio   w2 ratio [%]'])
disp([w1_rat_DJT w2_rat_DJT])
display([' DeltaT hot side '])
display(['    HX1        HX2       HX3       HX4       HX5   '])
```

```
disp([DeltaT_hot1_DJT DeltaT_hot2_DJT DeltaT_hot3_DJT ...
    DeltaT_hot4_DJT DeltaT_hot5_DJT])
display([' DeltaT cold side '])
display(['    HX1        HX2        HX3        HX4        HX5    '])
disp([DeltaT_cold1_DJT DeltaT_cold2_DJT DeltaT_cold3_DJT ...
    DeltaT_cold4_DJT DeltaT_cold5_DJT])
```

HEN_Constraints_41.m

```
% HEN_Constraints function 4:1 HEN for simulation of optimal ...
    operation
% Nonlinear constraints for optimizing a HEN
% Includes mass, energy and steady state balances

%%
function [Cineq, Res] = HEN_Constraints_41(x,par)

% Defining state variables
Tend = x(1); T1 = x(2); T2 = x(3); T3 = x(4); T4 = x(5); T5 = ...
    x(6);
Th1out = x(7); Th2out = x(8); Th3out = x(9); Th4out = x(10);
Th5out = x(11);
w1 = x(12); w2 = x(13);
Q1 = x(14); Q2 = x(15); Q3 = x(16); Q4 = x(17); Q5 = x(18);

% Defining parameters
w0 = par.w0;
wh1 = par.wh1; wh2 = par.wh2; wh3 = par.wh3; wh4 = par.wh4; ...
    wh5 = par.wh5;
Th1 = par.Th1; Th2 = par.Th2; Th3 = par.Th3; Th4 = par.Th4; ...
    Th5 = par.Th5;
T0 = par.T0;
UA1 = par.UA1; UA2 = par.UA2; UA3 = par.UA3; UA4 = par.UA4; ...
    UA5 = par.UA5;
DeltaTmin = par.DeltaTmin;



%% INEQUALITY CONSTRAINTS

```

```
% HX1
Cineq1 = -(Th1-T1-DeltaTmin); % HOT SIDE HX1
Cineq2 = -(Th1out-T0-DeltaTmin); % COLD SIDE HX1

% HX2
Cineq3 = -(Th2-T2-DeltaTmin); % HOT SIDE HX2
Cineq4 = -(Th2out-T1-DeltaTmin); % COLD SIDE HX2

% HX3
Cineq5 = -(Th3-T3-DeltaTmin); % HOT SIDE HX3
Cineq6 = -(Th3out-T2-DeltaTmin); % COLD SIDE HX3

% HX4
Cineq7 = -(Th4-T4-DeltaTmin); % HOT SIDE HX4
Cineq8 = -(Th4out-T3-DeltaTmin); % COLD SIDE HX4

% HX 5
Cineq9 = -(Th5-T5-DeltaTmin); % HOT SIDE HX5
Cineq10 = -(Th5out-T0-DeltaTmin); % COLD SIDE HX5

Cineq = ...
    [Cineq1;Cineq2;Cineq3;Cineq4;Cineq5;Cineq6;Cineq7;Cineq8;...
         Cineq9;Cineq10];
Cineq = [];



%% MODEL EQUATIONS

% AMTD
% DeltaT1 = 0.5*((Th1out-T0)+(Th1-T1));
% DeltaT2 = 0.5*((Th2out-T1)+(Th2-T2));
% DeltaT3 = 0.5*((Th3out-T2)+(Th3-T3));
% DeltaT4 = 0.5*((Th4out-T3)+(Th4-T4));
% DeltaT5 = 0.5*((Th5out-T0)+(Th5-T5));

%UNDERWOOD APPROXIMATION
 DeltaT1 = ((((Th1out-T0)^1/3)+((Th1-T1)^1/3))/2)^3;
 DeltaT2 = ((((Th2out-T1)^1/3)+((Th2-T2)^1/3))/2)^3;
 DeltaT3 = ((((Th3out-T2)^1/3)+((Th3-T3)^1/3))/2)^3;
 DeltaT4 = ((((Th4out-T3)^1/3)+((Th4-T4)^1/3))/2)^3;
```

```
 DeltaT5 = ((((Th5out-T0)^1/3)+((Th5-T5)^1/3))/2)^3;



 %% EQUALITY CONSTRAINTS

Res = [ % Upper path, 1st HX
         Q1-(w1*(T1-T0));                % Cold Stream, w1
         Q1+(par.wh1*(Th1out-Th1));      % Hot Stream, wh1
         Q1-(UA1*DeltaT1);               % HX Design Equation


        % Upper path, 2nd HX
         Q2-(w1*(T2-T1));                % Cold Stream, w1
         Q2+(par.wh2*(Th2out-Th2));      % Hot Stream, wh2
         Q2-(UA2*DeltaT2);               % HX Design Equation


        % Upper path, 3rd HX
         Q3-(w1*(T3-T2));                % Cold Stream, w1
         Q3+(par.wh3*(Th3out-Th3));      % Hot Stream, wh3
         Q3-(UA3*DeltaT3);               % HX Design equation

        % Lower path, 4th HX
         Q4-(w1*(T4-T3));                % Cold stream, w2
         Q4+(par.wh4*(Th4out-Th4));      % Hot stream, wh4
         Q4-(UA4*DeltaT4);               % HX design equation

         % Lower path, 5th HX
         Q5-(w2*(T5-T0));                % Cold stream, w2
         Q5+(par.wh5*(Th5out-Th5));      % Hot stream, wh4
         Q5-(UA5*DeltaT5);               % HX design equation


        % Mass balance
         w1+w2-w0;

        % Energy balance
         (w0*Tend)-(w1*T4)-(w2*T5)];

end
```

HEN_Constraints_41_DJT.m

```matlab
% HEN_Constraints function 4:1 HEN for simulations with the ...
    Jaeschke temp

% Nonlinear constraints for optimizing a HEN
% Includes mass, energy and steady state balances and the ...
    Jaeschke temp


function [Cineq, Res] = HEN_Constraints_41_DJT(x,par)

% Defining state variables
Tend = x(1); T1 = x(2); T2 = x(3); T3 = x(4); T4 = x(5); T5 = ...
    x(6);
Th1out = x(7); Th2out = x(8); Th3out = x(9); Th4out = x(10);
Th5out = x(11);
w1 = x(12); w2 = x(13);
Q1 = x(14); Q2 = x(15); Q3 = x(16); Q4 = x(17); Q5 = x(18);

% Defining parameteres
w0 = par.w0;
wh1 = par.wh1; wh2 = par.wh2; wh3 = par.wh3; wh4 = par.wh4; ...
    wh5 = par.wh5;
Th1 = par.Th1; Th2 = par.Th2; Th3 = par.Th3; Th4 = par.Th4; ...
    Th5 = par.Th5;
T0 = par.T0;
UA1 = par.UA1; UA2 = par.UA2; UA3 = par.UA3; UA4 = par.UA4; ...
    UA5 = par.UA5;
DeltaTmin = par.DeltaTmin;
P1 = par.P1; P2 = par.P2; P3 = par.P3; P4 = par.P4; P5 = par.P5;



%% INEQUALITY CONSTRAINTS

% HX1
Cineq1 = -(Th1-T1-DeltaTmin); % HOT SIDE HX1
Cineq2 = -(Th1out-T0-DeltaTmin); % COLD SIDE HX1

% HX2
```

```
Cineq3 = -(Th2-T2-DeltaTmin); % HOT SIDE HX2
Cineq4 = -(Th2out-T1-DeltaTmin); % COLD SIDE HX2

% HX3
Cineq5 = -(Th3-T3-DeltaTmin); % HOT SIDE HX3
Cineq6 = -(Th3out-T2-DeltaTmin); % COLD SIDE HX3

% HX4
Cineq7 = -(Th4-T4-DeltaTmin); % HOT SIDE HX4
Cineq8 = -(Th4out-T3-DeltaTmin); % COLD SIDE HX4

% HX 5
Cineq9 = -(Th5-T5-DeltaTmin); % HOT SIDE HX5
Cineq10 = -(Th5out-T0-DeltaTmin); % COLD SIDE HX5

Cineq = ...
    [Cineq1;Cineq2;Cineq3;Cineq4;Cineq5;Cineq6;Cineq7;Cineq8;...
         Cineq9;Cineq10];
Cineq = [];



%% MODEL EQUATIONS

% % AMTD
% DeltaT1 = 0.5*((Th1out-T0)+(Th1-T1));
% DeltaT2 = 0.5*((Th2out-T1)+(Th2-T2));
% DeltaT3 = 0.5*((Th3out-T2)+(Th3-T3));
% DeltaT4 = 0.5*((Th4out-T3)+(Th4-T4));
% DeltaT5 = 0.5*((Th5out-T0)+(Th5-T5));

%UNDERWOOD APPROXIMATION
 DeltaT1 = ((((Th1out-T0)^1/3)+((Th1-T1)^1/3))/2)^3;
 DeltaT2 = ((((Th2out-T1)^1/3)+((Th2-T2)^1/3))/2)^3;
 DeltaT3 = ((((Th3out-T2)^1/3)+((Th3-T3)^1/3))/2)^3;
 DeltaT4 = ((((Th4out-T3)^1/3)+((Th4-T4)^1/3))/2)^3;
 DeltaT5 = ((((Th5out-T0)^1/3)+((Th5-T5)^1/3))/2)^3;



%% JAESCHKE TEMPERATURES
```

```
% Upper path
JT11 = P1*(T1-T0)^2/(Th1-T0);
JT12 = P2*((T2-T1)*(T2+T1-2*T0-JT11))/(Th2-T1);
JT13 = P3*((T3-T2)*(T3+T2-2*T0-JT12))/(Th3-T2);
JT14 = P4*((T4-T3)*(T4+T3-2*T0-JT13))/(Th4-T3);
% Lower path
JT21 = P5*(T5-T0)^2/(Th5-T0);



%% EQUALITY CONSTRAINTS
Res = [ % Upper path, 1st HX
         Q1-(w1*(T1-T0));                % Cold Stream, w1
         Q1+(par.wh1*(Th1out-Th1));      % Hot Stream, wh1
         Q1-(UA1*DeltaT1);               % HX Design Equation


        % Upper path, 2nd HX
         Q2-(w1*(T2-T1));                % Cold Stream, w1
         Q2+(par.wh2*(Th2out-Th2));      % Hot Stream, wh2
         Q2-(UA2*DeltaT2);               % HX Design Equation


        % Upper path, 3rd HX
         Q3-(w1*(T3-T2));                % Cold Stream, w1
         Q3+(par.wh3*(Th3out-Th3));      % Hot Stream, wh3
         Q3-(UA3*DeltaT3);               % HX Design equation

        % Lower path, 4th HX
         Q4-(w1*(T4-T3));                % Cold stream, w2
         Q4+(par.wh4*(Th4out-Th4));      % Hot stream, wh4
         Q4-(UA4*DeltaT4);               % HX design equation

         % Lower path, 5th HX
         Q5-(w2*(T5-T0));                % Cold stream, w2
         Q5+(par.wh5*(Th5out-Th5));      % Hot stream, wh4
         Q5-(UA5*DeltaT5);               % HX design equation

        % Mass balance
         w1+w2-w0;

```

```
        % Energy balance
         (w0*Tend)-(w1*T4)-(w2*T5);

        % Jaeschke temperature
          (JT11+JT12+JT13+JT14)-JT21];

end
```

Object_41.m

```
% Object function to be minimized
% for the 4:1 HEN

function[J] = Object_41(x,par)

% Unscale variables
% x = x.*par.sc.x;

% Defining parameters
P1 = par.P1;
P2 = par.P2;
P3 = par.P3;
P4 = par.P4;
P5 = par.P5;

% Defining outlet variables
T0 = par.T0;

w1 = x(12);
w2 = x(13);

T1 = x(2);
T2 = x(3);
T3 = x(4);
T4 = x(5);
T5 = x(6);

Tend = x(1);


```

```
% Cost function
J = -(P1*(T1-T0)*w1 + P2*(T2-T1)*w1 + P3*(T3-T2)*w1 + ...
    P4*(T4-T3)*w1 + P5*(T5-T0)*w2);
% J = J/1000;
end
```

Six Heat Exchangers in Series and One in Parallel

RunHEN_61.m

```
%% Model to simulate a steady state 6:1 HEN
% Topology to be investigated:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            1     2     3     4     5     6           %
%        ----0-----0-----0-----0-----0-----0----       %
%   ----|                                      |----   %
%        ---------------------0-----------------       %
%                             7                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

close all;
clear all;
clc;

%% Parameters

% Heat Capacity rates
par.w0 = 100;    %[kW/degC] w= miCpi
par.wh1 = 50;    %[kW/degC]
par.wh2 = 30;    %[kW/degC]
par.wh3 = 15;    %[kW/degC]
par.wh4 = 25;    %[kW/degC]
par.wh5 = 40;    %[kW/degC]
par.wh6 = 35;    %[kW/degC]
par.wh7 = 30;    %[kW/degC]

% Hot stream inlet temperature
par.Th1 = 190;   %[degC]
par.Th2 = 203;   %[degC]
par.Th3 = 220;   %[degC]
par.Th4 = 235;   %[degC]
par.Th5 = 240;   %[degC]
par.Th6 = 245;   %[degC]
par.Th7 = 225;   %[degC]

% Cold stream inlet temperture
```

```
par.T0 = 130;    %[degC]

% UA values for each heat exchanger
par.UA1 = 5;     %[kWm2/degC]
par.UA2 = 7;     %[kWm2/degC]
par.UA3 = 10;    %[kWm2/degC]
par.UA4 = 12;    %[kWm2/degC]
par.UA5 = 9;     %[kWm2/degC]
par.UA6 = 8;     %[kWm2/degC]
par.UA7 = 11;    %[kWm2/degC]

% Operating prices for each heat exchanger
par.P1 = 1;      %[$/kW]
par.P2 = 1.2;    %[$/kW]
par.P3 = 1.3;    %[$/kW]
par.P4 = 1.5;    %[$/kW]
par.P5 = 1.4;    %[$/kW]
par.P6 = 1.7;    %[$/kW]
par.P7 = 1.5;    %[$/kW]

% Scaling vector
par.sc.x = [200*ones(15,1);100;100;500*ones(7,1)];

% Defining parameters
Th1 = par.Th1; Th2 = par.Th2; Th3 = par.Th3; Th4 = par.Th4; ...
    Th5 = par.Th5;
Th6 = par.Th6; Th7 = par.Th7;
T0 = par.T0;


%% OPTIMAL OPERATION

% Guessing outlet variables
% x0 = [Tend T1 T2 T3 T4 T5 T6 T7 Th1 Th2 Th3 Th4 Th5 Th6 Th7 ...
    w1 w2 ...
%       Q1 Q2 Q3 Q4 Q5 Q6 Q7]
x0 = [148 131 133 138 138 140 145 150 188 198 200 215 190 230 ...
    200 50 50 ...
      59 137 297 333 200 250 300]';

% Minimizing cost function based on equality constraints
```

```
% using fmincon
A = []; b = []; Aeq = []; Beq = [];
LB = 0*ones(24,1); UB = inf*ones(24,1);

options = optimset('display','iter',...
    'MaxFunEvals',9000,'TolCon',1e-10,'TolX',1e-10);

[x,J,exitflag] = fmincon(@(x)Object_61(x,par),x0,A,b,Aeq,Beq,...
    LB,UB,@(x)HEN_Constraints_61(x,par),options);
exitflag

% RESULTS
% Outlet temperatures
Tend = x(1);
T1 = x(2); T2 = x(3); T3 = x(4); T4 = x(5); T5 = x(6); T6 = ...
   x(7); T7 = x(8);
Th1out = x(9); Th2out = x(10); Th3out = x(11); Th4out = x(12);
Th5out = x(13); Th6out = x(14); Th7out = x(15);
% Split
w1 = x(16); w2 = x(17);
% Heat transfer
Q1 = x(18); Q2 = x(19); Q3 = x(20); Q4 = x(21); Q5 = x(22);
Q6 = x(23); Q7 = x(24);
% Split ratio
w1_rat = w1/par.w0;
w2_rat = w2/par.w0;
% Delta Ts
DeltaT_hot1 = Th1 - T1;
DeltaT_hot2 = Th2 - T2;
DeltaT_hot3 = Th3 - T3;
DeltaT_hot4 = Th4 - T4;
DeltaT_hot5 = Th5 - T5;
DeltaT_hot6 = Th6 - T6;
DeltaT_hot7 = Th7 - T7;
DeltaT_cold1 = Th1out - T0;
DeltaT_cold2 = Th2out - T1;
DeltaT_cold3 = Th3out - T2;
DeltaT_cold4 = Th4out - T3;
DeltaT_cold5 = Th5out - T4;
DeltaT_cold6 = Th6out - T5;
DeltaT_cold7 = Th7out - T0;
```

```

% Displaying the results
display(['  Tend [degC] = '])
disp(Tend)
display(['   T1          T2          T3          T4          T5 ...
            T6          T7    [degC]'])
disp([T1 T2 T3 T4 T5 T6 T7])
display(['    Th1out     Th2out     Th3out     Th4out     Th5out ...
        Th6out     Th7out    [degC]'])
disp([Th1out Th2out Th3out Th4out Th5out Th6out Th7out])
display(['      w1         w2'])
disp([w1 w2])
display(['  w1 ratio   w2 ratio [%]'])
disp([w1_rat w2_rat])
display([' DeltaT hot side '])
display(['    HX1          HX2         HX3         HX4         HX5 ...
           HX6         HX7    '])
disp([DeltaT_hot1 DeltaT_hot2 DeltaT_hot3 DeltaT_hot4 ...
    DeltaT_hot5 DeltaT_hot6 DeltaT_hot7])
display([' DeltaT cold side '])
display(['    HX1          HX2         HX3         HX4         HX5 ...
           HX6         HX7    '])
disp([DeltaT_cold1 DeltaT_cold2 DeltaT_cold3 DeltaT_cold4 ...
    DeltaT_cold5 DeltaT_cold6 DeltaT_cold7])


%% OPERATION USING THE JAESCHKE TEMPERATURE

% Guessing outlet variables
% x0 = [Tend T1 T2 T3 T4 T5 T6 T7 Th1 Th2 Th3 Th4 Th5 Th6 Th7 ...
    w1 w2 ...
%        Q1 Q2 Q3 Q4 Q5 Q6 Q7]
x0 = [148 131 133 138 138 140 145 150 188 198 200 215 190 230 ...
    200 50 50 ...
       59 137 297 333 200 250 300]';

% Defining parameters
Th1 = par.Th1; Th2 = par.Th2; Th3 = par.Th3; Th4 = par.Th4; ...
    Th5 = par.Th5;
Th6 = par.Th6; Th7 = par.Th7;
T0 = par.T0;
```

```


% Minimizing cost function based on equality constraints and ...
    Jaeschke temp
% using fmincon
A = []; b = []; Aeq = []; Beq = [];
LB = 0*ones(24,1); UB = inf*ones(24,1);

options = optimset('display','iter',...
     'MaxFunEvals',9000,'TolCon',1e-10,'TolX',1e-10);

[xDJT,J,exitflag] = ...
    fmincon(@(x)Object_61(x,par),x0,A,b,Aeq,Beq,...
     LB,UB,@(x)HEN_Constraints_61_DJT(x,par),options);
exitflag


% RESULTS
% Outlet temperatures
Tend_DJT = xDJT(1);
T1_DJT = xDJT(2); T2_DJT = xDJT(3); T3_DJT = xDJT(4); T4_DJT = ...
    xDJT(5); T5_DJT = xDJT(6); T6_DJT = xDJT(7); T7_DJT = xDJT(8);
Th1out_DJT = xDJT(9); Th2out_DJT = xDJT(10); Th3out_DJT = ...
    xDJT(11); Th4out_DJT = xDJT(12);
Th5out_DJT = xDJT(13); Th6out_DJT = xDJT(14); Th7out_DJT = ...
    xDJT(15);
% Split
w1_DJT = xDJT(16); w2_DJT = xDJT(17);
% Heat transfer
Q1_DJT = xDJT(18); Q2_DJT = xDJT(19); Q3_DJT = xDJT(20); ...
    Q4_DJT = xDJT(21); Q5_DJT = xDJT(22);
Q6_DJT = xDJT(23); Q7_DJT = xDJT(24);
% Split ratio
w1_rat_DJT = w1_DJT/par.w0;
w2_rat_DJT = w2_DJT/par.w0;
% Delta Ts
DeltaT_hot1_DJT = Th1 - T1_DJT;
DeltaT_hot2_DJT = Th2 - T2_DJT;
DeltaT_hot3_DJT = Th3 - T3_DJT;
DeltaT_hot4_DJT = Th4 - T4_DJT;
DeltaT_hot5_DJT = Th5 - T5_DJT;
```

```
DeltaT_hot6_DJT = Th6 - T6_DJT;
DeltaT_hot7_DJT = Th7 - T7_DJT;
DeltaT_cold1_DJT = Th1out_DJT - T0;
DeltaT_cold2_DJT = Th2out_DJT - T1_DJT;
DeltaT_cold3_DJT = Th3out_DJT - T2_DJT;
DeltaT_cold4_DJT = Th4out_DJT - T3_DJT;
DeltaT_cold5_DJT = Th5out_DJT - T4_DJT;
DeltaT_cold6_DJT = Th6out_DJT - T5_DJT;
DeltaT_cold7_DJT = Th7out_DJT - T0;

% Displaying the results
display(['  Tend [degC] = '])
disp(Tend)
display(['   T1          T2          T3          T4          T5 ...
          T6          T7     [degC]'])
disp([T1_DJT T2_DJT T3_DJT T4_DJT T5_DJT T6_DJT T7_DJT])
display(['    Th1out     Th2out     Th3out     Th4out     Th5out ...
      Th6out     Th7out     [degC]'])
disp([Th1out_DJT Th2out_DJT Th3out_DJT Th4out_DJT Th5out_DJT ...
    Th6out_DJT Th7out_DJT])
display(['      w1         w2'])
disp([w1_DJT w2_DJT])
display(['  w1 ratio    w2 ratio [%]'])
disp([w1_rat_DJT w2_rat_DJT])
display([' DeltaT hot side '])
display(['    HX1         HX2         HX3         HX4         HX5 ...
         HX6         HX7     '])
disp([DeltaT_hot1_DJT DeltaT_hot2_DJT DeltaT_hot3_DJT ...
    DeltaT_hot4_DJT DeltaT_hot5_DJT DeltaT_hot6_DJT ...
    DeltaT_hot7_DJT])
display([' DeltaT cold side '])
display(['    HX1         HX2         HX3         HX4         HX5 ...
         HX6         HX7     '])
disp([DeltaT_cold1_DJT DeltaT_cold2_DJT DeltaT_cold3_DJT ...
    DeltaT_cold4_DJT DeltaT_cold5_DJT DeltaT_cold6_DJT ...
    DeltaT_cold7_DJT])
```

HEN_Constraints_61.m

```
% HEN_Constraints function 6:1 HEN for simulations of optimal ... operation
% Nonlinear constraints for optimizing a HEN
% Includes mass, energy and steady state balances


function [Cineq, Res] = HEN_Constraints_61(x,par)

% Defining state variables
Tend = x(1);
T1 = x(2); T2 = x(3); T3 = x(4); T4 = x(5); T5 = x(6); T6 = ... x(7); T7 = x(8);
Th1out = x(9); Th2out = x(10); Th3out = x(11); Th4out = x(12);
Th5out = x(13); Th6out = x(14); Th7out = x(15);
w1 = x(16); w2 = x(17);
Q1 = x(18); Q2 = x(19); Q3 = x(20); Q4 = x(21); Q5 = x(22);
Q6 = x(23); Q7 = x(24);

% Defining parameters
Th1 = par.Th1; Th2 = par.Th2; Th3 = par.Th3; Th4 = par.Th4; ... Th5 = par.Th5;
Th6 = par.Th6; Th7 = par.Th7;
T0 = par.T0;
UA1 = par.UA1; UA2 = par.UA2; UA3 = par.UA3; UA4 = par.UA4; ... UA5 = par.UA5;
UA6 = par.UA6; UA7 = par.UA7;



%% INEQUALITY CONSTRAINTS
Cineq = [];



%% MODEL EQUATIONS
% AMTD
DeltaT1 = 0.5*((Th1out-T0)+(Th1-T1));
DeltaT2 = 0.5*((Th2out-T1)+(Th2-T2));
DeltaT3 = 0.5*((Th3out-T2)+(Th3-T3));
```

```
DeltaT4 = 0.5*((Th4out-T3)+(Th4-T4));
DeltaT5 = 0.5*((Th5out-T4)+(Th5-T5));
DeltaT6 = 0.5*((Th6out-T5)+(Th6-T6));
DeltaT7 = 0.5*((Th7out-T0)+(Th7-T7));

%UNDERWOOD APPROXIMATION
 DeltaT1 = ((((Th1out-T0)^1/3)+((Th1-T1)^1/3))/2)^3;
 DeltaT2 = ((((Th2out-T1)^1/3)+((Th2-T2)^1/3))/2)^3;
 DeltaT3 = ((((Th3out-T2)^1/3)+((Th3-T3)^1/3))/2)^3;
 DeltaT4 = ((((Th4out-T3)^1/3)+((Th4-T4)^1/3))/2)^3;
 DeltaT5 = ((((Th5out-T4)^1/3)+((Th5-T5)^1/3))/2)^3;
 DeltaT6 = ((((Th6out-T5)^1/3)+((Th6-T6)^1/3))/2)^3;
 DeltaT7 = ((((Th7out-T0)^1/3)+((Th7-T7)^1/3))/2)^3;


 %% EQUALITY CONSTRAINTS
Res = [
        % Upper path, 1st HX
         Q1-(w1*(T1-T0));                 % Cold Stream, w1
         Q1+(par.wh1*(Th1out-Th1));       % Hot Stream, wh1
         Q1-(UA1*DeltaT1);                % HX Design Equation


        % Upper path, 2nd HX
         Q2-(w1*(T2-T1));                 % Cold Stream, w1
         Q2+(par.wh2*(Th2out-Th2));       % Hot Stream, wh2
         Q2-(UA2*DeltaT2);                % HX Design Equation


        % Upper path, 3rd HX
         Q3-(w1*(T3-T2));                 % Cold Stream, w1
         Q3+(par.wh3*(Th3out-Th3));       % Hot Stream, wh3
         Q3-(UA3*DeltaT3);                % HX Design equation

        % Lower path, 4th HX
         Q4-(w1*(T4-T3));                 % Cold stream, w2
         Q4+(par.wh4*(Th4out-Th4));       % Hot stream, wh4
         Q4-(UA4*DeltaT4);                % HX design equation

        % Lower path, 5th HX
         Q5-(w1*(T5-T4));                 % Cold stream, w2
```

```
            Q5+(par.wh5*(Th5out−Th5));       % Hot stream, wh4
            Q5−(UA5*DeltaT5);                % HX design equation

           % Upper path, 6th HX
           Q6−(w1*(T6−T5));                  % Cold stream, w1
           Q6+(par.wh6*(Th6out−Th6));        % Hot stream, wh1
           Q6−(UA6*DeltaT6);                 % HX Design Equation

           % Lower path, 7th HX
           Q7−(w2*(T7−T0));                  % Cold stream, w1
           Q7+(par.wh7*(Th7out−Th7));        % Hot stream, wh1
           Q7−(UA7*DeltaT7);                 % HX Design Equation

           % Mass balance
           par.w0−(w1+w2);

           % Energy balance;
           par.w0*Tend−(w1*T6+w2*T7)];

end
```

HEN_Constraints_61_DJT.m

```
% HEN_Constraints function 6:1 HEN for simulations with the ...
    Jaeschke temp

% Nonlinear constraints for optimizing a HEN
% Includes mass, energy and steady state balances and the ...
    Jaeschke temp

%%
function [Cineq, Res] = HEN_Constraints_61_DJT(x,par)

% Defining state variables
Tend = x(1);
T1 = x(2); T2 = x(3); T3 = x(4); T4 = x(5); T5 = x(6); T6 = ...
    x(7); T7 = x(8);
Th1out = x(9); Th2out = x(10); Th3out = x(11); Th4out = x(12);
Th5out = x(13); Th6out = x(14); Th7out = x(15);
w1 = x(16); w2 = x(17);
```

```
Q1 = x(18); Q2 = x(19); Q3 = x(20); Q4 = x(21); Q5 = x(22);
Q6 = x(23); Q7 = x(24);

% Defining parameters
Th1 = par.Th1; Th2 = par.Th2; Th3 = par.Th3; Th4 = par.Th4; ...
    Th5 = par.Th5;
Th6 = par.Th6; Th7 = par.Th7;
T0 = par.T0;
UA1 = par.UA1; UA2 = par.UA2; UA3 = par.UA3; UA4 = par.UA4; ...
    UA5 = par.UA5;
UA6 = par.UA6; UA7 = par.UA7;
P1 = par.P1; P2 = par.P2; P3 = par.P3; P4 = par.P4; P5 = par.P5;
P6 = par.P6; P7 = par.P7;



%% INEQUALITY CONSTRAINTS
Cineq = [];



%% MODEL EQUATIONS

% AMTD
DeltaT1 = 0.5*((Th1out-T0)+(Th1-T1));
DeltaT2 = 0.5*((Th2out-T1)+(Th2-T2));
DeltaT3 = 0.5*((Th3out-T2)+(Th3-T3));
DeltaT4 = 0.5*((Th4out-T3)+(Th4-T4));
DeltaT5 = 0.5*((Th5out-T4)+(Th5-T5));
DeltaT6 = 0.5*((Th6out-T5)+(Th6-T6));
DeltaT7 = 0.5*((Th7out-T0)+(Th7-T7));

%UNDERWOOD APPROXIMATION
 DeltaT1 = ((((Th1out-T0)^1/3)+((Th1-T1)^1/3))/2)^3;
 DeltaT2 = ((((Th2out-T1)^1/3)+((Th2-T2)^1/3))/2)^3;
 DeltaT3 = ((((Th3out-T2)^1/3)+((Th3-T3)^1/3))/2)^3;
 DeltaT4 = ((((Th4out-T3)^1/3)+((Th4-T4)^1/3))/2)^3;
 DeltaT5 = ((((Th5out-T4)^1/3)+((Th5-T5)^1/3))/2)^3;
 DeltaT6 = ((((Th6out-T5)^1/3)+((Th6-T6)^1/3))/2)^3;
 DeltaT7 = ((((Th7out-T0)^1/3)+((Th7-T7)^1/3))/2)^3;

```

```
%% JAESCHKE TEMPERATURES
% Upper path
JT11 = P1*(T1−T0)^2/(Th1−T0);
JT12 = P2*((T2−T1)*(T2+T1−2*T0−JT11))/(Th2−T1);
JT13 = P3*((T3−T2)*(T3+T2−2*T0−JT12))/(Th3−T2);
JT14 = P4*((T4−T3)*(T4+T3−2*T0−JT13))/(Th4−T3);
JT15 = P5*((T5−T4)*(T5+T4−2*T0−JT14))/(Th5−T4);
JT16 = P6*((T6−T5)*(T6+T5−2*T0−JT15))/(Th6−T5);
% Lower path
JT21 = P7*(T7−T0)^2/(Th7−T0);



%% EQUALITY CONSTRAINTS
Res = [
        % Upper path, 1st HX
         Q1−(w1*(T1−T0));                 % Cold Stream, w1
         Q1+(par.wh1*(Th1out−Th1));       % Hot Stream, wh1
         Q1−(UA1*DeltaT1);                % HX Design Equation


        % Upper path, 2nd HX
         Q2−(w1*(T2−T1));                 % Cold Stream, w1
         Q2+(par.wh2*(Th2out−Th2));       % Hot Stream, wh2
         Q2−(UA2*DeltaT2);                % HX Design Equation


        % Upper path, 3rd HX
         Q3−(w1*(T3−T2));                 % Cold Stream, w1
         Q3+(par.wh3*(Th3out−Th3));       % Hot Stream, wh3
         Q3−(UA3*DeltaT3);                % HX Design equation

        % Lower path, 4th HX
         Q4−(w1*(T4−T3));                 % Cold stream, w2
         Q4+(par.wh4*(Th4out−Th4));       % Hot stream, wh4
         Q4−(UA4*DeltaT4);                % HX design equation

        % Lower path, 5th HX
         Q5−(w1*(T5−T4));                 % Cold stream, w2
```

```
        Q5+(par.wh5*(Th5out-Th5));       % Hot stream, wh4
        Q5-(UA5*DeltaT5);                % HX design equation

       % Upper path, 6th HX
       Q6-(w1*(T6-T5));                  % Cold stream, w1
       Q6+(par.wh6*(Th6out-Th6));        % Hot stream, wh1
       Q6-(UA6*DeltaT6);                 % HX Design Equation

       % Lower path, 7th HX
       Q7-(w2*(T7-T0));                  % Cold stream, w1
       Q7+(par.wh7*(Th7out-Th7));        % Hot stream, wh1
       Q7-(UA7*DeltaT7);                 % HX Design Equation


       % Mass balance
       par.w0-(w1+w2);

       % Energy balance;
       par.w0*Tend-(w1*T6+w2*T7)

       % Jaeschke temperature
       (JT11+JT12+JT13+JT14+JT15+JT16)-JT21];

end
```

Object_61.m

```
% Object function to be minimized
% for the 6:1 HEN

function[J] = Object_61(x,par)
% Unscale variables
% x = x.*par.sc.x;

% Defining parameters
P1 = par.P1;
P2 = par.P2;
P3 = par.P3;
P4 = par.P4;
P5 = par.P5;
```

```
P6 = par.P6;
P7 = par.P7;

% Defining outlet variables
T0 = par.T0;

w1 = x(16);
w2 = x(17);

T1 = x(2);
T2 = x(3);
T3 = x(4);
T4 = x(5);
T5 = x(6);
T6 = x(7);
T7 = x(8);

% Cost function
J = -(P1*(T1-T0)*w1 + P2*(T2-T1)*w1 + P3*(T3-T2)*w1 + ...
    P4*(T4-T3)*w1 + P5*(T5-T4)*w1 + P6*(T6-T5)*w1 + P7*(T7-T0)*w2);
J = J/1000;
end
```

Case II: Two Heat Exchangers in Parallel

OptCalc.m

```
% Optimal operation of a 1:1 HEN and
% operation using the Jaeschke temperature.
% Simulations are based on the NTU-method


% Topology to be investigated

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                1                %
%        -------0-------          %
%   ----|               |----     %
%        -------0-------          %
%                2                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc;
clear all;
close all;

% Defining parameters

% Cases evaluated
% Vector parameters: [T0 w0 wh1 wh2 Th1in Th2in UA1 UA2]

caseI       = [130 100  50 50  203 248  10 30];
caseII      = [130 100 50 50  203 248  31.1 93.9];
caseIII     = [130 50 100 100  203 248  10 30];
caseIV      = [130 100 50 50 203 248 100 300];
caseV       = [130 100 400 100 203 248 1000 100];
caseVI      = [130 100 400 100 203 248 1000 1000];

% Select case
casesel = caseI;

% Operation parameters
T0     = casesel(1); % Feed stream temperature [degC]
```

```
w0       = casesel(2); % [kW/K]

% Utility parameters
wh1      = casesel(3); % Hot stream 1 Heat Capacity rate [kW/K]
wh2      = casesel(4); % Hot stream 2 Heat Capacity rate [kW/K]
Th1in    = casesel(5); % Hot stream 1 Temperature [degC]
Th2in    = casesel(6); % Hot stream 2 Temperature [degC]

% Design parameters
UA1      = casesel(7);       % [kW/K]
UA2      = casesel(8);       % [kW/K]

% Number of iterations
N=1000;

n = zeros(N,1);
T1=n; T2=n; Th1=n; Th2=n; Tmix=n; e1=n; eh1=n; e2=n; eh2=n;
C1=n; C2=n; NTU1=n; NTU2=n; U=n;

% Calculating HX based on the NTU-method for all splits ...
    ranging [0,1]:
for i=1:N

    u = i/N;
    U(i)=u;

%     Calculating outlet temperatures and info about HEs
%      (only u is changing)
    [T HE] = TempCalc(T0,w0,UA1,UA2,Th1in,wh1,Th2in,wh2,u);

    T1(i)=T(1); T2(i)=T(2); Th1(i)=T(3); Th2(i)=T(4);
    Tmix(i)=T(5); e1(i)=HE(1); eh1(i)=HE(2); e2(i)=HE(3);
    eh2(i)=HE(4); C1(i)=HE(5); C2(i)=HE(6); NTU1(i)=HE(7);
    NTU2(i)=HE(8);

end


% RESULTS

% Finding optimal split
```

```
[Tmixm,nr]=max(Tmix);

split=U(nr);
T1m=T1(nr);
Th1m=Th1(nr);
T2m=T2(nr);
Th2m=Th2(nr);
Tmixm
split

% Finding the self-optimizing split

% Jaeschke Temperature for HX1 and HX2
JT = (T1-T0).^2./(Th1in-T0) - (T2-T0).^2./(Th2in-T0);

[JTmin,nr2]=min(abs(JT));

JT_opt=JT(nr);
JTsplit=U(nr2);
T1JT=T1(nr2);
Th1JT=Th1(nr2);
T2JT=T2(nr2);
Th2JT=Th2(nr2);
JTmin;
JTTmax=Tmix(nr2);
JTTmax
JTsplit

% Jaeschke temperature in the presence of measurement errors, max

JTTmax_vec = [];
TempLoss = [];

nT0 = 0;
nTh1 = 0;
nTh2 = 0;
nT1 = 0;
nT2 = 0;

M = 1000;

```

```
% Simulating HX with measurement errors, with given ...
    Measurement errors
% (data file Measurement_Errors.m)
for j=1:M;

    load Measurement_Errors

    nT0 = noise(1,j);
    nTh1 = noise(2,j);
    nTh2 = noise(3,j);
    nT1 = noise(4,j);
    nT2 = noise(5,j);

    % Implementing the noise in the control variable
    JT_noise = ((T1+nT1)-(T0+nT0)).^2./((Th1in+nTh1)-(T0+nT0)) ...
        - ((T2+nT2)-(T0+nT0)).^2./((Th2in+nTh2)-(T0+nT0));

    [JTmin_noise,nr3] = min(abs(JT_noise));
    JT_noise_split = U(nr3);
    JTnoiseTmax = Tmix(nr3);

    JTTmax_vec(j) = JTnoiseTmax;
    TempLoss(j) = Tmixm-JTnoiseTmax;

    noise(:,j) = [nT0, nTh1, nTh2, nT1, nT2]';

end

% Worst case loss and avergae loss
WCloss = max(TempLoss);
AVGloss = sum(TempLoss)/M;
WCloss
AVGloss

% % Calculating temperature difference on each side of each HX

dTcold1=Th1-T0;
dThot1=Th1in-T1;

dTcold2=Th2-T0;
dThot2=Th2in-T2;
```

```
% Calculating errors from AMTD approximation
[eU1 eU2 eAM1 eAM2] = ErrorCalc(dTcold1, dThot1, dTcold2, dThot2);

% Calculating the AMTD approximation valid range..
theta1 = dThot1./dTcold1;
theta2 = dThot2./dTcold2;


% PLOTTING THE RESULTS

% Temperature and control variable profile with split u
% return
h = figure;
% return
% figure(1)
y1start = 160; y1end = 210; y1step = 10;
y2start = -60; y2end = 60; y2step = 60;

split = [split split];
JTs = [JTsplit JTsplit];
y11 = [y1start y1end];
y22 = [y2start y2end];

[AX,H1,H2] = plotyy(U,JT,U,Tmix);
set(get(AX(2),'Ylabel'),'String',...
    'T_{end} [ \circC]','fontsize',12)
set(get(AX(1),'Ylabel'),'String',...
    'Controlled variable, JT [ \circC]','fontsize',12)
axis(AX(2),[0 1 y1start y1end]);
axis(AX(1),[0 1 y2start y2end]);
set(AX(2),'YLim',[y1start y1end])
set(AX(2),'YTick',y1start:y1step:y1end)
set(AX(1),'YLim',[y2start y2end])
set(AX(1),'YTick',y2start:y2step:y2end)
set(H1,'linewidth',2)
set(H2,'linewidth',2)
xlabel('Split, u','fontsize',12);
hold on;
H3 = plot(JTs,y22,'Color','k','LineStyle','--','LineWidth',2);
hold on
```

```
H4 = plot(split,y22,'Color','r','LineStyle','--','LineWidth',2);

set(H3,'parent',AX(1));
% hold on;
grid on;
print(h,'-depsc','CaseIId_optCalc.eps');


% Validity of AMTD approximation
UB = [1.4 1.4]; % Upper AMTD limit
LB = [(1/1.4) (1/1.4)]; % Lower AMTD limit
s = [0 1];

k = figure;
% figure(6);
plot(U,theta2,U,theta1,'LineWidth',2);
xlabel('Split, u','fontsize',12);
ylabel('\theta_{1}/\theta_{2}','fontsize',12);
% legend('HX_{1,2}','HX_{1,1}','fontsize',12);
axis([0 1 0 2]);
% Using hline.m to include upper and lower bounds:
hline([1/1.4 1.4],{'m','m'},{'AMTD LB','AMTD UB'})
hold on;
plot(splitline,solid,'Color','r','LineStyle','--','LineWidth',2);
legend('HX_{1,2}','HX_{1,1}','Optimal split','fontsize',11);
print(k,'-depsc','AMTD_CaseIIb.eps');
```

TempCalc.m

```
% TempCalc function to calculate HX with the NTU-method

function [T HE] = TempCalc(T0,w0,UA1,UA2,Th1in,wh1,...
                           Th2in,wh2,u)

% Cold stream heat capacity rates
w1 = u*w0;
w2 = (1-u)*w0;

% Number of transit units (NTU)
NTU1 = UA1/w1;
```

```
NTU2 = UA2/w2;

% Heat capacity ratios
C1 = w1/wh1;
C2 = w2/wh2;

% Preventing from singular solutions
if(C1>0.999 && C1<1.001)
    C1=0.999;
end

if(C2>0.999 && C2<1.001)
    C2=0.999;
end

% Calculating the effectiveness of HXs
e1 = (1-exp(-NTU1*(C1-1)))/(C1-exp(-NTU1*(C1-1)));
e2 = (1-exp(-NTU2*(C2-1)))/(C2-exp(-NTU2*(C2-1)));
eh1 = e1*C1;
eh2 = e2*C2;

% Calculating outlet temperatures
T1  = e1*Th1in + (1-e1)*T0;
T2  = e2*Th2in + (1-e2)*T0;
Th1 = (1-eh1)*Th1in + eh1*T0;
Th2 = (1-eh2)*Th2in + eh2*T0;
Tmix = u*T1+(1-u)*T2;

T = [T1 T2 Th1 Th2 Tmix];
HE = [e1 eh1 e2 eh2 C1 C2 NTU1 NTU2];
```

ErrorCalc.m

```
% ErrorCalc function to calculate errors associated with using the
% AMTD and Underwood approximation

function [eU1 eU2 eAM1 eAM2] = ErrorCalc(dTcold1, dThot1, ...
    dTcold2, dThot2)


```

```
%Logarithmic mean temperature difference
LM1 = (dThot1-dTcold1)./log(dThot1./dTcold1);
LM2 = (dThot2-dTcold2)./log(dThot2./dTcold2);

%Arithmetic mean temperature difference
AM1 = (dTcold1+dThot1)./2;
AM2 = (dTcold2+dThot2)./2;

% Underwood temperature difference
U1 = ((((dTcold1).^(1/3))+((dThot1).^(1/3)))./2).^3;
U2 = ((((dTcold2).^(1/3))+((dThot2).^(1/3)))./2).^3;

%AMTD error
eAM1 = (AM1-LM1)./LM1*100;
eAM2 = (AM2-LM2)./LM2*100;

%Underwood error
eU1 = (U1-LM1)./LM1*100;
eU2 = (U2-LM2)./LM2*100;

end
```

hline.m

```
function hhh=hline(y,in1,in2)
% function h=hline(y, linetype, label)
%
% Draws a horizontal line on the current axes at the location ...
    specified by 'y'.  Optional arguments are
% 'linetype' (default is 'r:') and 'label', which applies a ...
    text label to the graph near the line.  The
% label appears in the same color as the line.
%
% The line is held on the current axes, and after plotting the ...
    line, the function returns the axes to
% its prior hold state.
%
% The HandleVisibility property of the line object is set to ...
    "off", so not only does it not appear on
```

```
% legends, but it is not findable by using findobj.  ...
    Specifying an output argument causes the function to
% return a handle to the line, so it can be manipulated or ...
    deleted.  Also, the HandleVisibility can be
% overridden by setting the root's ShowHiddenHandles property ...
    to on.
%
% h = hline(42,'g','The Answer')
%
% returns a handle to a green horizontal line on the current ...
    axes at y=42, and creates a text object on
% the current axes, close to the line, which reads "The Answer".
%
% hline also supports vector inputs to draw multiple lines at ...
    once.  For example,
%
% hline([4 8 12],{'g','r','b'},{'l1','lab2','LABELC'})
%
% draws three lines with the appropriate labels and colors.
%
% By Brandon Kuczenski for Kensington Labs.
% brandon_kuczenski@kensingtonlabs.com
% 8 November 2001

if length(y)>1  % vector input
    for I=1:length(y)
        switch nargin
        case 1
            linetype='r:';
            label='';
        case 2
            if ~iscell(in1)
                in1={in1};
            end
            if I>length(in1)
                linetype=in1{end};
            else
                linetype=in1{I};
            end
            label='';
        case 3
```

```
                if ~iscell(in1)
                    in1={in1};
                end
                if ~iscell(in2)
                    in2={in2};
                end
                if I>length(in1)
                    linetype=in1{end};
                else
                    linetype=in1{I};
                end
                if I>length(in2)
                    label=in2{end};
                else
                    label=in2{I};
                end
            end
            h(I)=hline(y(I),linetype,label);
        end
else
    switch nargin
    case 1
        linetype='r:';
        label='';
    case 2
        linetype=in1;
        label='';
    case 3
        linetype=in1;
        label=in2;
    end




    g=ishold(gca);
    hold on

    x=get(gca,'xlim');
    h=plot(x,[y y],linetype);
    if ~isempty(label)
```

```
        yy=get(gca,'ylim');
        yrange=yy(2)-yy(1);
        yunit=(y-yy(1))/yrange;
        if yunit<0.2
            text(x(1)+0.85*(x(2)-x(1)),y+0.02*yrange,label,...
                'color',get(h,'color'))
        else
            text(x(1)+0.85*(x(2)-x(1)),y-0.02*yrange,label,...
                'color',get(h,'color'))
        end
    end

    if g==0
    hold off
    end
    set(h,'tag','hline','handlevisibility','off') % this last ...
        part is so that it doesn't show up on legends
end % else

if nargout
    hhh=h;
end
```

C.2 Dynamic Analysis Scripts

Dynamic Case I: Two Heat Exchangers in Parallel

Run.m

```
% RUN FILE FOR DYNAMIC SIMULATION OF THE 1:1 HEN

% Topology to be investigated:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                1                %
%        ———————0———————          %
%   ————|               |————     %
%        ———————0———————          %
%                2                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;

% Calling parameters from Data.m file
[T0,Th1,Th2,...
            m0,m1,m2,mh1,mh2...
            rho_0, hc, Cp0,...
            Vwall, rho_wall, Cp_wall,...
            P1, P2] = Data;



sim('dynamic_11_1')



% % TUNING OF CONTROLLER
% % 10% STEP CHANGE INLET MASS FLOW COLD STREAM
% % TUNING PLOT
% t0 = 800;
% tend = 1800;
%
```

```
% cv1_0 = -6;
% cv1_end = 1;
% cv1_step = 1;
%
% m1_0 = 9;
% m1_end = 11;
% m1_step = 0.5;
%
% k = figure;
% [AX,H1,H2] = plotyy(t,cv1,t,m1);
% set(get(AX(1),'Ylabel'),'String','Controlled variable, JT ...
    [^{\circ}C]','fontsize',12)
% set(get(AX(2),'Ylabel'),'String','Mass flow m_1 to upper ...
    path [kg/sec]','fontsize',12)
% axis(AX(1),[t0 tend cv1_0 cv1_end]);
% axis(AX(2),[t0 tend m1_0 m1_end]);
% set(AX(1),'YLim',[cv1_0 cv1_end])
% set(AX(1),'YTick',cv1_0:cv1_step:cv1_end)
% set(AX(2),'YLim',[m1_0 m1_end])
% set(AX(2),'YTick',m1_0:m1_step:m1_end)
% xlabel('Time [sec]','fontsize',12)
% set(H1,'linewidth',2)
% set(H2,'linewidth',2)
% grid on
% print(k,'-depsc','tune_11.eps');


% IMPLEMENTING FILTERS - SIMULATING BEHAVIOR WITH AND WITHOUT ...
    FILTE
% % Without Filter
% cv1_noAF = cv1;
% u1_noAF = u1;
% T1_noAF = T1;
% T2_noAF = T2;
% Tend_noAF = Tend;

% save no_filter

% % With Filter
% cv1_AF = cv1;
% u1_AF = u1;
```

```
% T1_AF = T1;
% T2_AF = T2;
% Tend_AF = Tend;

% save filter


% PLOTING THE RESULTS

t0 = 800;
tend = 2000;

cv1_0 = -3.5;
cv1_end = 1;
cv1_step = 0.1;


% CONTROLLED VARIABLE PROFILES
k = figure;
plot(t,cv1_noAF,'b',t,cv1_AF,'r','LineWidth',2)
legend('Without filter','With filter')
xlabel('Time [sec]','fontsize',12);
ylabel('Controlled variable, JT [^{\circ}C]','fontsize',12)
axis([t0 tend cv1_0 cv1_end])
grid on
% print(k,'-depsc','CV_11.eps');

% SPLIT
i = figure;
plot(t,u1_noAF,'b',t,u1_AF,'r','LineWidth',2)
legend('Without filter','With filter')
xlabel('Time [sec]','fontsize',12)
ylabel('Split u (Upper path)','fontsize',12)
axis([t0 tend 0 0.36])
grid on
% print(i,'-depsc','Split_11.eps');

% TEMPERATURE PROFILES
j = figure;
plot(t,T1_AF,t,T2_AF,t,Tend_AF,'LineWidth',2)
xlabel('Time[sec]','fontsize',12)
```

```
ylabel('Temperature [^{\circ}C]','fontsize',12)
axis([t0 tend 195 220])
legend('T_{1,1}','T_{1,2}','T_{end}')
grid on
% print(j,'-depsc','T_11.eps');
```

Data.m

```
% DATA FILE
% STREAM AND HEAT EXCHANGER DATA FOR THE 1:1 HEN

function [T0,Th1,Th2,...
            m0,m1,m2,mh1,mh2...
            rho_0, hc, Cp0,...
            Vwall, rho_wall, Cp_wall,...
            P1, P2] = Data


% COLD STREAM
T0 = 130; % Inlet cold stream temperature [degC]
rho_0 = 1000; % Density cold stream [kg/m3]
hc = 0.17; % Heat transfer coeffsient cold fluid (water) ... [kW/m2degC]
m0 = 38; % Mass flow cold stream [kg/sek]
Cp0 = 2.5; % Heat capacity cold stream [kJ/kgdegC]
m1 = m0*0.2553; % Bypass to upper branch, start value for ... simulation
m2 = m0-m1; % Bypass to lower branch, start value for simulation

% HEAT EXCHANGER 1
Th1 = 203; % Inlet hot stream temperature [degC]
mh1 = 30; % Mass flow hot stream [kg/sek]
P1 = 1; % Price constant

% HEAT EXCHANGER 2
Th2 = 248; % Inlet hot stream temperature [degC]
mh2 = 21.67; % Mass flow hot stream [kg/sek]
P2 = 1; % Price constant

% HEAT EXCHANGER DATA
```

```
m_wall = 3000; % Wall weight HXers [kg]
rho_wall = 7850; % Wall density CS [kg/m3] %7850
Vwall = m_wall/rho_wall; % Volume walls [m3]
Cp_wall = 0.49; % Heat capacity walls CS [kW/kgdegC]

end
```

Dynamic.m

```
% DYNAMIC FUNCTION AND STATE EQUATIONS FOR THE 1:1 HEN

function xprime = Dynamic(t,X,U,N,HXindex)

% Defining the outlet varibles
Th_out = X(1:N);
Twall = X(N+1:2*N);
Tc_out = X(2*N+1:3*N);

% Defining inlet parameters from Simulink
Th_in(1) = U(1);
mh_in = U(2);
Tc_in(1) = U(3);
m0_in = U(4);

% Calling parameters from Data.m file
[T0,Th1,Th2,...
            m0,m1,m2,mh1,mh2,...
            rho_0, hc, Cp0,...
            Vwall, rho_wall, Cp_wall] = Data;


if HXindex == 1
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.31*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
```

```
    Ai = 250;



elseif HXindex == 2
    Cph = 3;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.1*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 700;


end


% STATE EQUATIONS

% Hot stream
dThotdt(1) = ...
    (Th_in(1)-Th_out(1)-((U*Ai)/(wh*N))*(Th_out(1)-Twall(N))...
    *(mh_in*N)/(rho_h*Vhot));

% Wall
dTwalldt(1) = (hh*(Th_out(N)-Twall(1))-hc*(Twall(1)-Tc_out(1)))...
    *(Ai/(rho_wall*Cp_wall*Vwall));

% Cold stream
dTcolddt(1) ...
    =(Tc_in(1)-Tc_out(1)-((U*Ai)/(w0*N))*(Tc_out(1)-Twall(1)))...
    *((m0_in*N)/(rho_0*Vcold));


for i = 2:N
    j = N-i+1;
    dThotdt(i) = (Th_out(i-1)-Th_out(i)-((U*Ai)/(wh*N))...
        *(Th_out(i)-Twall(j))*(mh_in*N)/(rho_h*Vhot));
end
```

```
for j = 2:N
    i = N-j+1;
    dTwalldt(j) = ...
        (hh*(Th_out(i)-Twall(j))-hc*(Twall(j)-Tc_out(j)))...
        *(Ai/(rho_wall*Cp_wall*Vwall));
    dTcolddt(j)=(Tc_out(j-1)-Tc_out(j)-((U*Ai)/(w0*N))...
        *(Tc_out(j)-Twall(j))*((m0_in*N)/(rho_0*Vcold)));
end


xprime = [dThotdt, dTwalldt, dTcolddt];
end
```

HX1.m

```
% HEAT EXCHANGER 1

function [sys,x0] = HX1(t,x,u,flag)

HXindex = 1; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

```

```
end
```

HX2.m

```
% HEAT EXCHANGER 2

function [sys,x0] = HX2(t,x,u,flag)

HXindex = 2; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

ssvar.m

```
% STEADY STATE VARIABLES FOR EACH HEAT EXCHANGER
% IN THE 1:1 HEN

function [x0] = ssvar(HXindex,N)

    if HXindex == 1

```

```
            x0 = [202.4350
                  201.6831
                  200.6825
                  199.3507
                  197.5784
                  195.2197
                  192.0806
                  187.9029
                  182.3430
                  174.9436
                  156.5233
                  168.5020
                  177.5028
                  184.2660
                  189.3478
                  193.1663
                  196.0355
                  198.1914
                  199.8113
                  201.0286
                  132.3926
                  150.3702
                  163.8786
                  174.0288
                  181.6556
                  187.3864
                  191.6925
                  194.9281
                  197.3593
                  199.1861];




    elseif HXindex == 2


            x0 = [238.5844
                  229.1347
                  219.6505
                  210.1320
```

```
                        200.5788
                        190.9910
                        181.3683
                        171.7107
                        162.0179
                        152.2900
                        142.1443
                        151.9090
                        161.6383
                        171.3324
                        180.9914
                        190.6155
                        200.2047
                        209.7592
                        219.2791
                        228.7645
                        130.9841
                        140.7891
                        150.5587
                        160.2929
                        169.9919
                        179.6558
                        189.2846
                        198.8787
                        208.4380
                        217.9627];



        end
```

Dynamic Case II: Two Heat Exchangers in Series Parallel to One Heat Exchanger

Run.m

```
% RUN FILE FOR DYNAMIC SIMULATION OF THE 2:1 HEN

% Topology to be investigated:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             1     2             %
%         ----0-----0----         %
%    ----|               |----    %
%         -------0-------         %
%                3                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;

% Calling parameters from Data.m
[T0,Th1,Th2,Th3...
            m0,m1,m2,mh1,mh2,mh3...
            rho_0, hc, Cp0,...
            Vwall, rho_wall, Cp_wall,...
            filterk, filtert,...
            P1, P2, P3] = Data;


% SIMULINK FILE FOR SIMULATION WITH THE MODIFIED CV
% sim('dynamic_21_1_1')

% SIMULINK FILE FOR SIMULATION WITH THE ORIGINAL CV
sim('dynamic_21_1')



% % TUNING OF CONTROLLER
% % 10% STEP CHANGE INLET MASS FLOW COLD STREAM
% % TUNING PLOT
```

```
% t0 = 800;
% tend = 1800;
%
% cv1_0 = 1e7;
% cv1_end = 3e7;
% cv1_step = 0.5e7;
%
% m1_0 = 28;
% m1_end = 34;
% m1_step = 1;
%
% k = figure;
% [AX,H1,H2] = plotyy(t,cv1,t,m1);
% set(get(AX(1),'Ylabel'),'String','Controlled variable, JT ...
    [^{\circ}C]','fontsize',12)
% set(get(AX(2),'Ylabel'),'String','Mass flow m_1 to upper ...
    path [kg/sec]','fontsize',12)
% axis(AX(1),[t0 tend cv1_0 cv1_end]);
% axis(AX(2),[t0 tend m1_0 m1_end]);
% set(AX(1),'YLim',[cv1_0 cv1_end])
% set(AX(1),'YTick',cv1_0:cv1_step:cv1_end)
% set(AX(2),'YLim',[m1_0 m1_end])
% set(AX(2),'YTick',m1_0:m1_step:m1_end)
% xlabel('Time [sec]','fontsize',12)
% set(H1,'linewidth',2)
% set(H2,'linewidth',2)
% grid on
% print(k,'-depsc','tune_21_numJT.eps');


% % IMPLEMENTING FILTERS - SIMULATING BEHAVIOR WITH AND ...
    WITHOUT FILTER
% % Without Filter
% cv1_noAF = cv1;
% u1_noAF = u1;
% T1_noAF = T1;
% T2_noAF = T2;
% T3_noAF = T3;
% Tend_noAF = Tend;
%
% save no_filter
```

```
% % With Filter
% cv1_AF = cv1;
% u1_AF = u1;
% T1_AF = T1;
% T2_AF = T2;
% T3_AF = T3;
% Tend_AF = Tend;
%
% save filter



% PLOTING THE RESULTS

t0 = 800;
tend = 2000;

cv1_0 = -5;
cv1_end = 5;
cv1_step = 5;


% % RESULTS FOR THE CASE WITH COOLING HX (MOD. CV)

% % TEMPERATURE PROFILES W/ COOLING TH2
% h = figure;
% figure(1)
% plot(t,T1,t,T2,t,Th2_d,'LineWidth',2)
% xlabel('Time [sec]','fontsize',12);
% ylabel('Temperature [ \circC]','fontsize',12);
% legend('T_{1,1}','T_{2,1}','Th_{2,1}')
% axis([t0 tend 170 260])
% grid on
% % print(h,'-depsc','T_coolHX2_numJT_Tune1.eps');
%
% % SPLIT PROFILE W/ COOLING TH2
% j = figure;
% figure(2)
% plot(t,u1,'LineWidth',2)
% xlabel('Time [sec]','fontsize',12);
```

```
% ylabel('Split u','fontsize',12);
% % legend('T1','T2','Th2')
% axis([t0 tend 0 1])
% grid on
% % print(j,'-depsc','Split_coolHX2_numJT_Tune1.eps');


% % RESULTS FOR THE ORIGINAL CASE (ORG. CV)

% % CONTROLLED VARIABLE PROFILE WITHOUT FILTER
% k = figure;
% % figure(3)
% plot(t,cv1,'LineWidth',2)
% % h=BreakXAxis(t,cv1,-1e7,-5000,1000);
% % legend('Without AF','With AF')
% % title('CV (J1-J2)')
% xlabel('Time [sec]','fontsize',12);
% ylabel('Mod. control variable, JT [^{\circ}C^4]','fontsize',12)
% axis([t0 tend cv1_0 cv1_end])
% grid on
% print(k,'-depsc','CV_coolHX2_fullplot_Tune2.eps');


% % SPLIT WITHOUT FILTER
% % figure(3)
% i = figure;
% plot(t,u1,'LineWidth',2)
% % legend('Without AF','With AF')
% % title('CV (J1-J2)')
% xlabel('Time [sec]','fontsize',12)
% ylabel('Split u (Upper path)','fontsize',12)
% axis([t0 tend 0.1 0.8])
% grid on
% print(i,'-depsc','Split_21.eps');



% CONTROLLED VARIABLE PROFILE WITH FILTER
l = figure;
plot(t,cv1_noAF,'b',t,cv1_AF,'r','LineWidth',2)
legend('Without filter','With filter')
```

```
xlabel('Time [sec]','fontsize',12);
ylabel('Controlled variable, JT [^{\circ}C]','fontsize',12)
axis([t0 tend cv1_0 cv1_end])
grid on
% print(l,'-depsc','CV_filter_21.eps');

% SPLIT WITH FILTER
i = figure;
plot(t,u1_noAF,'b',t,u1_AF,'r','LineWidth',2)
legend('Without filter','With filter')
xlabel('Time [sec]','fontsize',12)
ylabel('Split u (Upper path)','fontsize',12)
axis([t0 tend 0.3 0.601])
grid on
% print(i,'-depsc','Split_filter_21.eps');

% TEMPERATURE PROFILES WITH FILTER
j = figure;
plot(t,T1_AF,t,T2_AF,t,T3,t,Tend_AF,'LineWidth',2)
xlabel('Time[sec]','fontsize',12)
ylabel('Temperature [^{\circ}C]','fontsize',12)
axis([t0 tend 160 210])
legend('T_{1,1}','T_{2,1}','T_{1,2}','T_{end}')
grid on
% print(j,'-depsc','T_21.eps');
```

Data.m

```
% DATA FILE
% STREAM AND HEAT EXCHANGER DATA FOR THE 2:1 HEN

function [T0,Th1,Th2,Th3...
            m0,m1,m2,mh1,mh2,mh3...
            rho_0, hc, Cp0,...
            Vwall, rho_wall, Cp_wall,...
            filterk, filtert,...
            P1, P2, P3] = Data;


% COLD STREAM DATA
```

```
T0 = 130; % Inlet cold stream temperature [degC]
rho_0 = 1000; % Density cold stream [kg/m3]
hc = 0.10; % Heat transfer coeffsient cold fluid (water) ...
    [kW/m2degC]
m0 = 64; % Mass flow cold stream [kg/sek]
Cp0 = 2.5; % Heat capacity cold stream [kJ/kgdegC]
m1 = m0*0.4522; % Bypass to upper branch, start value for ...
    simulation
m2 = m0-m1; % Bypass to lower branch, start value for simulation

% HEAT EXCHANGER 1
Th1 = 203; % Inlet hot stream temperature [degC]
mh1 = 30; % Mass flow hot stream [kg/sec]
P1 = 1; % Price constant

% HEAT EXCHANGER 2
Th2 = 255; % Inlet hot stream temperature [degC]
mh2 = 13.5; % Mass flow hot stream [kg/sec]
P2 = 1; % Price constant

% HEAT EXCHANGER 3
Th3 = 248; % Inlet hot stream temperature [degC]
mh3 = 21.67; % Mass flow hot stream [kg/sec]
P3 = 1; % Price constant

% HEAT EXCHANGER DATA
m_wall = 3000; % Wall weight HXers [kg]
rho_wall = 7850; % Wall density CS [kg/m3] %7850
Vwall = m_wall/rho_wall; % Wall volume [m3]
Cp_wall = 0.49; % Heat capacity wall CS [kW/kgdegC]


end
```

Dynamic.m

```
% DYNAMIC FUNCTION AND STATE EQUATIONS FOR THE 2:1 HEN

function xprime = Dynamic(t,X,U,N,HXindex)

```

```
% Defining outlet variables
Th_out = X(1:N);
Twall = X(N+1:2*N);
Tc_out = X(2*N+1:3*N);

% Defining inlet parameters from Simulink
Th_in(1) = U(1);
mh_in = U(2);
Tc_in(1) = U(3);
m0_in = U(4);

% Calling additional parameters from Data.m
[T0,Th1,Th2,Th3...
            m0,m1,m2,mh1,mh2,mh3...
            rho_0, hc, Cp0,...
            Vwall, rho_wall, Cp_wall,...
            filterk, filtert,...
            P1, P2, P3] = Data;


if HXindex == 1
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.089*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 341;


elseif HXindex == 2
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.025*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
```

```
    Ai = 616;
else HXindex == 3
    Cph = 3;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.070*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 1118;
end
% STATE EQUATIONS
% Hot stream
dThotdt(1) = ...
   (Th_in(1)-Th_out(1)-((U*Ai)/(wh*N))*(Th_out(1)-Twall(N))...
    *(mh_in*N)/(rho_h*Vhot));
% Wall
dTwalldt(1) = (hh*(Th_out(N)-Twall(1))-hc*(Twall(1)-Tc_out(1)))...
    *(Ai/(rho_wall*Cp_wall*Vwall));
% Cold stream
dTcolddt(1) ...
   =(Tc_in(1)-Tc_out(1)-((U*Ai)/(w0*N))*(Tc_out(1)-Twall(1)))...
    *((m0_in*N)/(rho_0*Vcold));
for i = 2:N
    j = N-i+1;
    dThotdt(i) = (Th_out(i-1)-Th_out(i)-((U*Ai)/(wh*N))*...
        (Th_out(i)-Twall(j))*(mh_in*N)/(rho_h*Vhot));
end
```

```
for j = 2:N
    i = N-j+1;
    dTwalldt(j) = ...
        (hh*(Th_out(i)-Twall(j))-hc*(Twall(j)-Tc_out(j)))...
        *(Ai/(rho_wall*Cp_wall*Vwall));

    dTcolddt(j)=(Tc_out(j-1)-Tc_out(j)-((U*Ai)/(w0*N))*...
        (Tc_out(j)-Twall(j))*((m0_in*N)/(rho_0*Vcold)));
end

% Outlet variables
xprime = [dThotdt, dTwalldt, dTcolddt];
end
```

HX1.m

```
% HEAT EXCHANGER 1

function [sys,x0] = HX1(t,x,u,flag)

HXindex = 1; % HX number
N = 10; % Model order


if abs(flag) == 1
    display('flag = 1')
    sys = Dynamic(t,x,u,N,HXindex);
    disp(sys)

elseif abs(flag) == 3
    display('flag = 3')
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)
    disp(sys)

elseif flag == 0
    display('flag = 0')
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];
    disp(sys)
```

```
else
    sys = [];

end

end
```

HX2.m

```
% HEAT EXCHANGER 2

function [sys,x0] = HX2(t,x,u,flag)

HXindex = 2; % HX number
N = 10; % Model order


if abs(flag) == 1
     sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
     sys(1,1) = x(N); % Outlet hot temperature
     sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
     x0 = ssvar(HXindex,N);
     sys = [3*N,0,2,4,0,0];

else
    sys = [];

end

end
```

HX3.m

```
% HEAT EXCHANGER 3
```

```
function [sys,x0] = HX3(t,x,u,flag)

HXindex = 3; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

ssvar.m

```
% STEADY STATE VARIABLES FOR EACH HEAT EXCHANGER
% IN THE 2:1 HEN

function [x0] = ssvar(HXindex,N)

    if HXindex == 1

                x0 = [198.3549
                      193.7732
                      189.2542
                      184.7968
                      180.4004
                      176.0641
```

```
                  171.7870
                  167.5684
                  163.4074
                  159.3032
                  145.4507
                  149.3629
                  153.3294
                  157.3508
                  161.4278
                  165.5614
                  169.7523
                  174.0012
                  178.3089
                  182.6763
                  130.3653
                  134.0685
                  137.8231
                  141.6297
                  145.4890
                  149.4017
                  153.3687
                  157.3906
                  161.4683
                  165.6024];

    elseif HXindex == 2


            x0 = [234.0031
                  217.7572
                  205.1873
                  195.4616
                  187.9366
                  182.1142
                  177.6093
                  174.1237
                  171.4268
                  169.3401
                  167.5332
                  169.0914
                  171.1054
```

```
                    173.7083
                    177.0724
                    181.4204
                    187.0398
                    194.3026
                    203.6894
                    215.8213
                    165.6811
                    166.6977
                    168.0116
                    169.7098
                    171.9046
                    174.7412
                    178.4074
                    183.1458
                    189.2699
                    197.1848];




    else HXindex == 3



              x0 = [235.0515
                    222.8678
                    211.4038
                    200.6169
                    190.4672
                    180.9169
                    171.9308
                    163.4754
                    155.5194
                    148.0334
                    139.6122
                    146.5695
                    153.9637
                    161.8220
                    170.1736
                    179.0494
```

```
                            188.4824
                            198.5076
                            209.1621
                            220.4854
                            130.6014
                            136.9932
                            143.7862
                            151.0056
                            158.6782
                            166.8324
                            175.4985
                            184.7086
                            194.4969
                            204.8996];



    end
```

Dynamic Case III: Three Heat Exchangers in Series Parallel to Two Heat Exchangers

Run.m

```
% RUN FILE FOR DYNAMIC SIMULATION OF THE 3:2 HEN

% Topology to be investigated:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            1      2      3            %
%        -----0------0------0-----      %
%   ----|                       |----   %
%        ---------0-------0-------      %
%                 4       5             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;

% Calling parameters from Data.m file
[T0,Th1,Th2,Th3,Th4,Th5,...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,...
            rho_0,hc,Cp0,...
            Vwall,rho_wall,Cp_wall,...
            P1,P2,P3,P4,P5] = Data;



sim('dynamic_32')


% % TUNING OF CONTROLLER
% % 10% STEP CHANGE INLET MASS FLOW COLD STREAM
% % TUNING PLOT
% t0 = 800;
% tend = 2000;
%
% cv1_0 = -5;
% cv1_end = 10;
```

```
% cv1_step = 3;
%
% m1_0 = 16;
% m1_end = 20;
% m1_step = 1;
%
% [AX,H1,H2] = plotyy(t,cv1,t,m1);
% set(get(AX(1),'Ylabel'),'String','Controlled variable, JT ...
    [^{\circ}C]','fontsize',12)
% set(get(AX(2),'Ylabel'),'String','Mass flow m_1 to upper ...
    path [kg/sec]','fontsize',12)
% axis(AX(1),[t0 tend cv1_0 cv1_end]);
% axis(AX(2),[t0 tend m1_0 m1_end]);
% set(AX(1),'YLim',[cv1_0 cv1_end])
% set(AX(1),'YTick',cv1_0:cv1_step:cv1_end)
% set(AX(2),'YLim',[m1_0 m1_end])
% set(AX(2),'YTick',m1_0:m1_step:m1_end)
% xlabel('Time [sec]','fontsize',12)
% set(H1,'linewidth',2)
% set(H2,'linewidth',2)
% grid on
% print(k,'-depsc','tune_32.eps');


% IMPLEMENTING FILTERS - SIMULATING BEHAVIOR WITH AND WITHOUT ...
    FILTE
% % Without Filter
% cv1_noAF = cv1;
% u1_noAF = u1;
% T1_noAF = T1;
% T2_noAF = T2;
% Tend_noAF = Tend;
%
% save no_filter

% % With Filter
% cv1_AF = cv1;
% u1_AF = u1;
% T1_AF = T1;
% T2_AF = T2;
% T3_AF = T3;
```

```
% T4_AF = T4;
% T5_AF = T5;
% Tend_AF = Tend;
%
% save filter


% PLOTING THE RESULTS

t0 = 800;
tend = 3000;

cv1_0 = -0.5;
cv1_end = 3;
cv1_step = 0.1;


% CONTROLLED VARIABLE PROFILE
k = figure;
plot(t,cv1_noAF,'b',t,cv1_AF,'r','LineWidth',2)
legend('Without filter','With filter')
xlabel('Time [sec]','fontsize',12);
ylabel('Controlled variable, JT [^{\circ}C]','fontsize',12)
axis([t0 tend cv1_0 cv1_end])
grid on
% print(k,'-depsc','CV_32.eps');

% SPLIT
i = figure;
plot(t,u1_noAF,'b',t,u1_AF,'r','LineWidth',2)
legend('Without filter','With filter')
xlabel('Time [sec]','fontsize',12)
ylabel('Split u (Upper path)','fontsize',12)
axis([t0 tend 0.3 0.38])
grid on
% print(i,'-depsc','Split_32.eps');

% TEMPERATURE PROFILES
j = figure;
plot(t,T1_AF,t,T2_AF,t,T3_AF,t,T4_AF,t,T5_AF,t,...
    Tend_AF,'LineWidth',2)
```

```
xlabel('Time[sec]','fontsize',12)
ylabel('Temperature [^{\circ}C]','fontsize',12)
axis([t0 tend 145 195])
legend('T_{1,1}','T_{2,1}','T_{3,1}','T_{1,2}','T_{2,2}','T_{end}')
grid on
% print(j,'-depsc','T_32.eps');
```

Data.m

```
% DATA FILE
% STREAM AND HEAT EXCHANGER DATA FOR THE 3:2 HEN

function [T0,Th1,Th2,Th3,Th4,Th5,...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,...
            rho_0,hc,Cp0,...
            Vwall,rho_wall,Cp_wall,...
            P1,P2,P3,P4,P5] = Data


% COLD STREAM
T0 = 130; % Inlet cold stream temperature [degC]
rho_0 = 1000; % Density cold stream [kg/m3]
hc = 0.10; % Heat transfer coeffsient cold fluid (water) ...
    [kW/m2degC]
m0 = 60; % Mass flow cold stream [kg/sek]
Cp0 = 2.5; % Heat capacity cold stream [kJ/kgdegC]
m1 = m0*0.2828; % Bypass to upper branch, start value for ...
    simulation
m2 = m0-m1; % Bypass to lower branch, start value for simulation

% HEAT EXCHANGER 1
Th1 = 190; % Inlet hot stream temperature [degC]
mh1 = 25; % Mass flow hot stream [kg/sec]
P1 = 1; % Price constant

% HEAT EXCHANGER 2
Th2 = 203; % Inlet hot stream temperature [degC]
mh2 = 15; % Mass flow hot stream [kg/sec]
P2 = 1; % Price constant

```

```
% HEAT EXCHANGER 3
Th3 = 220; % Inlet hot stream temperature [degC]
mh3 = 7.5; % Mass flow hot stream [kg/sec]
P3 = 1; % Price constant

% HEAT EXCHANGER 4
Th4 = 220; % Inlet hot stream temperature[degC]
mh4 = 17.5; % Mass flow hot stream [kg/sec]
P4 = 1; % Price constant

% HEAT EXCHANGER 5
Th5 = 248; % Inlet hot stresm temperature [degC]
mh5 = 10; % Mass flow hot stream [kg/sec]
P5 = 1; % Price constant

% HEAT EXCHANGER DATA
m_wall = 3000; % Wall weight HXers [kg]
rho_wall = 7850; % Wall density CS [kg/m3] %7850
Vwall = m_wall/rho_wall; % Volume walls [m3]
Cp_wall = 0.49; % Heat capacity walls CS [kW/kgdegC]



end
```

Dynamic.m

```
% DYNAMIC FUNCTION AND STEADY STATE EQUATIONS FOR THE 3:2 HEN

function xprime = Dynamic(t,X,U,N,HXindex)

% Defining the outlet variables
Th_out = X(1:N);
Twall = X(N+1:2*N);
Tc_out = X(2*N+1:3*N);

% Defining inlet parameters from Simulink
Th_in(1) = U(1);
mh_in = U(2);
Tc_in(1) = U(3);
```

```
m0_in = U(4);

% Calling parameters from Data.m file
[T0,Th1,Th2,Th3,Th4,Th5,...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,...
            rho_0,hc,Cp0,...
            Vwall,rho_wall,Cp_wall,...
            P1,P2,P3,P4,P5] = Data;

if HXindex == 1
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.109*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 112.5;


elseif HXindex == 2
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.088*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 102;


elseif HXindex == 3
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.07*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
```

```
    w0 = m0_in*Cp0;
    Ai = 85;


elseif HXindex == 4
    Cph = 4;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.068*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 800;


else HXindex == 5
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 765;


end


% STATE EQUATIONS

% Hot stream
dThotdt(1) = (Th_in(1)-Th_out(1)-((U*Ai)/(wh*N))*...
    (Th_out(1)-Twall(N))*(mh_in*N)/(rho_h*Vhot));

% Wall
dTwalldt(1) = ...
    (hh*(Th_out(N)-Twall(1))-hc*(Twall(1)-Tc_out(1)))*...
    (Ai/(rho_wall*Cp_wall*Vwall));
```

```

% Cold stream
dTcolddt(1) ...
    =(Tc_in(1)-Tc_out(1)-((U*Ai)/(w0*N))*(Tc_out(1)-Twall(1)))*...
     ((m0_in*N)/(rho_0*Vcold));


for i = 2:N
    j = N-i+1;
    dThotdt(i) = (Th_out(i-1)-Th_out(i)-((U*Ai)/(wh*N))*...
        (Th_out(i)-Twall(j))*(mh_in*N)/(rho_h*Vhot));
end

for j = 2:N
    i = N-j+1;
    dTwalldt(j) = ...
        (hh*(Th_out(i)-Twall(j))-hc*(Twall(j)-Tc_out(j)))*...
        (Ai/(rho_wall*Cp_wall*Vwall));
    dTcolddt(j)=(Tc_out(j-1)-Tc_out(j)-((U*Ai)/(w0*N))*...
        (Tc_out(j)-Twall(j))*((m0_in*N)/(rho_0*Vcold)));
end


xprime = [dThotdt, dTwalldt, dTcolddt];
```

HX1.m

```
% HEAT EXCHANGER 1

function [sys,x0] = HX1(t,x,u,flag)

HXindex = 1; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
```

```
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX2.m

```
% HEAT EXCHANGER 2

function [sys,x0] = HX2(t,x,u,flag)

HXindex = 2; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX3.m

```
% HEAT EXCHANGER 3

function [sys,x0] = HX3(t,x,u,flag)

HXindex = 3; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX4.m

```
% HEAT EXCHANGER 4

function [sys,x0] = HX4(t,x,u,flag)

HXindex = 4; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);
```

```

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX4.m

```
% HEAT EXCHANGER 4

function [sys,x0] = HX4(t,x,u,flag)

HXindex = 4; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

```

```
end

end
```

ssvar.m

```
% STEADY STATE VARIABLES FOR EACH HEAT EXCHANGER
% IN THE 3:2 HEN

function [x0] = ssvar(HXindex,N)

    if HXindex == 1

                x0 = [188.0976
                      186.1641
                      184.1991
                      182.2021
                      180.1724
                      178.1097
                      176.0132
                      173.8826
                      171.7172
                      169.5165
                      150.9158
                      153.4152
                      155.8744
                      158.2941
                      160.6750
                      163.0177
                      165.3228
                      167.5908
                      169.8225
                      172.0183
                      130.2877
                      133.1182
                      135.9033
                      138.6437
                      141.3400
                      143.9931
                      146.6036
```

```
                        149.1722
                        151.6996
                        154.1863];



        elseif HXindex == 2

                  x0 = [200.4704
                        197.9864
                        195.5472
                        193.1520
                        190.8000
                        188.4904
                        186.2224
                        183.9953
                        181.8084
                        179.6609
                        167.5396
                        169.4645
                        171.4247
                        173.4209
                        175.4538
                        177.5241
                        179.6323
                        181.7792
                        183.9656
                        186.1921
                        154.3516
                        156.0343
                        157.7479
                        159.4930
                        161.2701
                        163.0799
                        164.9229
                        166.7997
                        168.7110
                        170.6574];
```

```
    elseif HXindex == 3

                 x0 = [215.3492
                          211.0567
                          207.0951
                          203.4387
                          200.0641
                          196.9495
                          194.0750
                          191.4219
                          188.9733
                          186.7134
                          178.9982
                          180.6139
                          182.3645
                          184.2613
                          186.3165
                          188.5433
                          190.9559
                          193.5701
                          196.4025
                          199.4714
                          170.7429
                          171.6693
                          172.6731
                          173.7608
                          174.9392
                          176.2160
                          177.5994
                          179.0984
                          180.7224
                          182.4821];




     elseif HXindex == 4

                 x0 = [210.3670
                          201.3865
```

```
                        193.0143
                        185.2092
                        177.9328
                        171.1493
                        164.8253
                        158.9296
                        153.4334
                        148.3094
                        139.6279
                        144.1211
                        148.9407
                        154.1106
                        159.6561
                        165.6045
                        171.9851
                        178.8293
                        186.1709
                        194.0458
                        130.3561
                        134.1756
                        138.2726
                        142.6673
                        147.3813
                        152.4379
                        157.8618
                        163.6799
                        169.9206
                        176.6149];




    elseif HXindex == 5

                  x0 = [219.5400
                        202.4055
                        192.0895
                        185.8787
                        182.1394
                        179.8882
                        178.5328
```

```
                                177.7168
                                177.2255
                                176.9297
                                176.7750
                                176.9686
                                177.2901
                                177.8241
                                178.7110
                                180.1842
                                182.6312
                                186.6955
                                193.4462
                                204.6590
                                176.6204
                                176.7117
                                176.8634
                                177.1154
                                177.5339
                                178.2291
                                179.3837
                                181.3015
                                184.4870
                                189.7779];



        end
```

Dynamic Case IV: Four Heat Exchangers in Series Parallel to One Heat Exchanger

Run.m

```
% RUN FILE FOR DYNAMIC SIMULAITON OF THE 4:1 HEN

% Topology to be investigated:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%             1      2      3      4            %
%         ----0------0------0------0----        %
%    ----|                            |----   %
%         -------------0--------------        %
%                      5                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;

% Calling parameters from Data.m file
[T0,Th1,Th2,Th3,Th4,Th5,...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,...
            rho_0,hc,Cp0,...
            Vwall,rho_wall,Cp_wall,...
            P1,P2,P3,P4,P5] = Data;



sim('dynamic_41')


% % TUNING OF CONTROLLER
% % 10% STEP CHANGE INLET MASS FLOW COLD STREAM
% % TUNING PLOT
% t0 = 800;
% tend = 2200;
%
% cv1_0 = -20;
% cv1_end = 0;
```

```
% cv1_step = 4;
%
% m1_0 = 38;
% m1_end = 44;
% m1_step = 2;
%
% % figure(1)
% k = figure;
% [AX,H1,H2] = plotyy(t,cv1,t,m1);
% set(get(AX(1),'Ylabel'),'String','Controlled variable, JT ...
    [^{\circ}C]','fontsize',12)
% set(get(AX(2),'Ylabel'),'String','Mass flow m_1 to upper ...
    path [kg/sec]','fontsize',12)
% axis(AX(1),[t0 tend cv1_0 cv1_end]);
% axis(AX(2),[t0 tend m1_0 m1_end]);
% set(AX(1),'YLim',[cv1_0 cv1_end])
% set(AX(1),'YTick',cv1_0:cv1_step:cv1_end)
% set(AX(2),'YLim',[m1_0 m1_end])
% set(AX(2),'YTick',m1_0:m1_step:m1_end)
% xlabel('Time [sec]','fontsize',12)
% set(H1,'linewidth',2)
% set(H2,'linewidth',2)
% grid on
% print(k,'-depsc','tune_41.eps');

% PLOTTING THE RESULTS

t0 = 800;
tend = 5000;

cv1_0 = -1;
cv1_end = 1;
cv1_step = 0.5;

u_0 = 0.75;
u_end = 0.80;

% CONTROL VARIABLE PROFILES
h = figure;
plot(t,cv1,'LineWidth',2)
xlabel('Time [sec]','fontsize',12)
```

```
ylabel('Controlled variable, JT [^{\circ}C]','fontsize',12)
axis([t0 tend cv1_0 cv1_end])
grid on
% print(h,'-depsc','CV_41.eps');

% SPLIT
j = figure;
plot(t,u1,'LineWidth',2)
xlabel('Time [sec]','fontsize',12)
ylabel('Split u (Upper path)','fontsize',12)
axis([t0 tend u_0 u_end])
grid on
% print(j,'-depsc','Split_41.eps');

% TEMPERATURE PROFILES
k = figure;
plot(t,T1,t,T2,t,T3,t,T4,t,T5,t,Tend,'LineWidth',2)
legend('T_{1,1}','T_{2,1}','T_{3,1}','T_{4,1}','T_{1,2}','T_{end}')
xlabel('Time [sec]','fontsize',12)
ylabel('Temperature [^{\circ}C]','fontsize',12)
axis([t0 tend 130 165])
% print(k,'-depsc','T_41.eps');
```

Data.m

```
% DATA FILE
% STREAM AND HEAT EXCHANGER DATA FOR THE 4:1 HEN

function [T0,Th1,Th2,Th3,Th4,Th5,...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,...
            rho_0,hc,Cp0,...
            Vwall,rho_wall,Cp_wall,...
            P1,P2,P3,P4,P5] = Data


% COLD STREAM
T0 = 130; % Inlet cold stream temperature [degC]
rho_0 = 1000; % Density cold stream [kg/m3]
hc = 0.10; % Heat transfer coeffsient cold fluid (water) ...
    [kW/m2degC]
```

```
m0 = 50; % Mass flow cold stream [kg/sek]
Cp0 = 2; % Heat capacity cold stream [kJ/kgdegC]
m1 = m0*0.7767; % Bypass to upper branch, start value for ...
    simulation
m2 = m0-m1; % Bypass to lower branch, start value for simulation

% HEAT EXCHANGER 1
Th1 = 190; % Inlet hot stream temperature [degC]
mh1 = 25; %  Mass flow hot stream [kg/sec]
P1 = 1; % Price constant

% HEAT EXCHANGER 2
Th2 = 203; % Inlet hot stream temperature [degC]
mh2 = 15; % Mass flow hot stream [kg/sec]
P2 = 1.2; % Price constant

% HEAT EXCHANGER 3
Th3 = 220; % Inlet hot stream temperature [degC]
mh3 = 7.5; % Mass flow hot stream [kg/sec]
P3 = 1.3; % Price constant

% HEAT EXCHANGER 4
Th4 = 235; % Inlet hot stream temperature [degC]
mh4 = 12.5; % Mass flow hot stream [kg/sec]
P4 = 1.5; % Price constant

% HEAT EXCHANGER 5
Th5 = 210; % Inlet hot stream temperature [degC]
mh5 = 35; % Mass flow hot stream [kg/sec]
P5 = 1.4; % Price constant

% HEAT EXCHANGER DATA
m_wall = 3000; % Wall weight HXers [kg]
rho_wall = 7850; % Wall density CS [kg/m3] %7850
Vwall = m_wall/rho_wall; % Volume walls [m3]
Cp_wall = 0.49; % Heat capacity walls CS [kW/kgdegC]


end
```

Dynamic.m

```
% DYNAMIC FUNCTION AND STATE EQUATIONS FOR THE 4:1 HEN

function xprime = Dynamic(t,X,U,N,HXindex)

% Defining the outlet varibles
Th_out = X(1:N);
Twall = X(N+1:2*N);
Tc_out = X(2*N+1:3*N);

% Defining inlet parameters from Simulink
Th_in(1) = U(1);
mh_in = U(2);
Tc_in(1) = U(3);
m0_in = U(4);

% Calling parameters from Data.m file
[T0,Th1,Th2,Th3,Th4,Th5,...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,...
            rho_0,hc,Cp0,...
            Vwall,rho_wall,Cp_wall,...
            P1,P2,P3,P4,P5] = Data;

if HXindex == 1
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.2*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 19;


elseif HXindex == 2
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.42*hc;
```

```
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 29.5;


elseif HXindex == 3
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.389*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 43.7;


elseif HXindex == 4
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 0.70*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 103;


else HXindex == 5
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.43*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 38.3;
```

```
end


% STATE EQUATIONS

% Hot stream
dThotdt(1) = ...
    (Th_in(1)-Th_out(1)-((U*Ai)/(wh*N))*(Th_out(1)-Twall(N))*...
    (mh_in*N)/(rho_h*Vhot));

% Wall
dTwalldt(1) = ...
    (hh*(Th_out(N)-Twall(1))-hc*(Twall(1)-Tc_out(1)))*...
    (Ai/(rho_wall*Cp_wall*Vwall));

% Cold stream
dTcolddt(1) ...
    =(Tc_in(1)-Tc_out(1)-((U*Ai)/(w0*N))*(Tc_out(1)-Twall(1)))*...
    ((m0_in*N)/(rho_0*Vcold));


for i = 2:N
    j = N-i+1;
    dThotdt(i) = (Th_out(i-1)-Th_out(i)-((U*Ai)/(wh*N))*...
        (Th_out(i)-Twall(j))*(mh_in*N)/(rho_h*Vhot));
end

for j = 2:N
    i = N-j+1;
    dTwalldt(j) = ...
        (hh*(Th_out(i)-Twall(j))-hc*(Twall(j)-Tc_out(j)))*...
        (Ai/(rho_wall*Cp_wall*Vwall));
    dTcolddt(j)=(Tc_out(j-1)-Tc_out(j)-((U*Ai)/(w0*N))*...
        (Tc_out(j)-Twall(j))*((m0_in*N)/(rho_0*Vcold)));
end

xprime = [dThotdt, dTwalldt, dTcolddt];
```

HX1.m

```
% HEAT EXCHANGER 1

function [sys,x0] = HX1(t,x,u,flag)

HXindex = 1; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX2.m

```
% HEAT EXCHANGER 2

function [sys,x0] = HX2(t,x,u,flag)

HXindex = 2; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);
```

```

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX3.m

```
% HEAT EXCHANGER 3

function [sys,x0] = HX3(t,x,u,flag)

HXindex = 3; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

```

```
end

end
```

HX4.m

```
% HEAT EXCHANGER 4

function [sys,x0] = HX4(t,x,u,flag)

HXindex = 4; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX5.m

```
% HEAT EXCHANGER 5

function [sys,x0] = HX5(t,x,u,flag)

HXindex = 5; % HX number
```

```
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

ssvar.m

```
% STEADY STATE VARIABLES FOR EACH HEAT EXCHANGER
% IN THE 4:1 HEN


function [x0] = ssvar(HXindex,N)

    if HXindex == 1

                x0 = [189.4314
                      188.8645
                      188.2992
                      187.7357
                      187.1738
                      186.6137
                      186.0552
                      185.4984
                      184.9432
```

```
                        184.3897
                        161.9307
                        162.4168
                        162.9043
                        163.3933
                        163.8838
                        164.3758
                        164.8692
                        165.3641
                        165.8605
                        166.3584
                        130.0389
                        130.4293
                        130.8208
                        131.2135
                        131.6074
                        132.0025
                        132.3988
                        132.7962
                        133.1949
                        133.5947];

      elseif HXindex == 2

                  x0 = [201.5099
                        200.0399
                        198.5896
                        197.1589
                        195.7474
                        194.3550
                        192.9812
                        191.6260
                        190.2890
                        188.9701
                        166.1107
                        167.1177
                        168.1385
                        169.1732
                        170.2220
                        171.2852
                        172.3628
```

```
                        173.4552
                        174.5624
                        175.6848
                        133.6504
                        134.2144
                        134.7862
                        135.3658
                        135.9533
                        136.5488
                        137.1524
                        137.7643
                        138.3845
                        139.0132];




    elseif HXindex == 3

                  x0 = [215.1294
                        210.5147
                        206.1425
                        202.0001
                        198.0753
                        194.3568
                        190.8336
                        187.4956
                        184.3330
                        181.3365
                        163.6465
                        165.6618
                        167.7889
                        170.0340
                        172.4036
                        174.9046
                        177.5443
                        180.3305
                        183.2711
                        186.3748
                        139.0750
                        139.7276
```

```
                    140.4164
                    141.1434
                    141.9107
                    142.7205
                    143.5753
                    144.4775
                    145.4297
                    146.4348];


    elseif HXindex == 4

                x0 = [231.2829
                    227.6813
                    224.1915
                    220.8101
                    217.5336
                    214.3589
                    211.2828
                    208.3022
                    205.4141
                    202.6158
                    169.6252
                    171.3663
                    173.1633
                    175.0178
                    176.9318
                    178.9071
                    180.9457
                    183.0496
                    185.2210
                    187.4619
                    146.5318
                    147.5329
                    148.5660
                    149.6323
                    150.7328
                    151.8685
                    153.0406
                    154.2503
                    155.4988
```

```
                    156.7872];


    elseif HXindex == 5

                x0 = [209.2873
                    208.5518
                    207.7927
                    207.0093
                    206.2009
                    205.3665
                    204.5055
                    203.6169
                    202.6998
                    201.7534
                    172.3595
                    174.2179
                    176.0185
                    177.7633
                    179.4540
                    181.0922
                    182.6796
                    184.2177
                    185.7082
                    187.1524
                    130.3264
                    133.4886
                    136.5528
                    139.5220
                    142.3990
                    145.1868
                    147.8880
                    150.5055
                    153.0418
                    155.4994];


    end
```

Dynamic Case V: Six Heat Exchangers in Series Parallel to One Heat Exchanger

Run.m

```
% RUN FILE FOR DYNAMIC SIMULATION OF THE 6:1 HEN

% Topology to be investigated:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            1     2     3     4     5     6           %
%        ----0-----0-----0-----0-----0-----0----       %
%   ----|                                     |----    %
%        --------------------0-------------------      %
%                            7                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
close all;
clc;

[T0,Th1,Th2,Th3,Th4,Th5,Th6,Th7...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,mh6,mh7...
            rho_0,hc,Cp0...
            Vwall,rho_wall,Cp_wall...
            P1,P2,P3,P4,P5,P6,P7] = Data



sim('dynamic_61')


% % TUNING OF CONTROLLER
% % 10% STEP CHANGE INLET MASS FLOW COLD STREAM
% % TUNING PLOT
% t0 = 800;
% tend = 2400;
%
% cv1_0 = -43;
% cv1_end = 7;
% cv1_step = 10;
```

```
%
% m1_0 = 40;
% m1_end = 48;
% m1_step = 2;
%
% k = figure;
% [AX,H1,H2] = plotyy(t,cv1,t,m1);
% set(get(AX(1),'Ylabel'),'String','Controlled variable, JT ...
    [^{\circ}C]','fontsize',12)
% set(get(AX(2),'Ylabel'),'String','Mass flow m_1 to upper ...
    path [kg/sec]','fontsize',12)
% axis(AX(1),[t0 tend cv1_0 cv1_end]);
% axis(AX(2),[t0 tend m1_0 m1_end]);
% set(AX(1),'YLim',[cv1_0 cv1_end])
% set(AX(1),'YTick',cv1_0:cv1_step:cv1_end)
% set(AX(2),'YLim',[m1_0 m1_end])
% set(AX(2),'YTick',m1_0:m1_step:m1_end)
% xlabel('Time [sec]','fontsize',12)
% set(H1,'linewidth',2)
% set(H2,'linewidth',2)
% grid on
% print(k,'-depsc','tune_61.eps');


% PLOTTING THE RESULTS

t0 = 800;
tend = 5000;

cv1_0 = -1.5;
cv1_end = 1.5;
cv1_step = 0.5;

u_0 = 0.82;
u_end = 0.8601;

% CONTROLLED VARIABLE PROFILES
h = figure;
plot(t,cv1,'LineWidth',2)
xlabel('Time [sec]','fontsize',12)
ylabel('Controlled variable, JT [^{\circ}C]','fontsize',12)
```

```
axis([t0 tend cv1_0 cv1_end])
grid on
% print(h,'-depsc','CV_61.eps');

% SPLIT
j = figure;
plot(t,u1,'LineWidth',2)
xlabel('Time [sec]','fontsize',12)
ylabel('Split u (Upper path)','fontsize',12)
axis([t0 tend u_0 u_end])
grid on
% print(j,'-depsc','Split_61.eps');

% TEMPERATURE PROFILES
k = figure;
plot(t,T1,t,T2,t,T3,t,T4,t,T5,t,T6,t,T7,t,Tend,'LineWidth',2)
legend('T_{1,1}','T_{2,1}','T_{3,1}','T_{4,1}','T_{5,1}',...
    'T_{6,1}','T_{1,2}','T_{end}')
xlabel('Time [sec]','fontsize',12)
ylabel('Temperature [^{\circ}C]','fontsize',12)
axis([t0 tend 130 175])
% print(k,'-depsc','T_61.eps');
```

Data.m

```
% DATA FILE
% STREAM AND HEAT EXCHANGER DATA FOR THE 6:1 HEN


function [T0,Th1,Th2,Th3,Th4,Th5,Th6,Th7,...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,mh6,mh7,...
            rho_0,hc,Cp0,...
            Vwall,rho_wall,Cp_wall,...
            P1,P2,P3,P4,P5,P6,P7] = Data


% COLD STREAM
T0 = 130; % Inlet cold stream temperature [degC]
rho_0 = 1000; % Density cold stream [kg/m3]
```

```
hc = 0.10; % Heat transfer coeffsient cold fluid (water) ...
    [kW/m2degC]
m0 = 50; % Mass flow cold stream [kg/sek]
Cp0 = 2; % Heat capacity cold stream [kJ/kgdegC]
m1 = m0*0.8299; % Bypass to upper branch, start value for ...
    simulation
m2 = m0-m1; % Bypass to lower branch, start value for simulation

% HEAT EXCHANGER 1
Th1 = 190; % Inlet hot stream temperature [degC]
mh1 = 25; % Mass flow hot stream [kg/sec]
P1 = 1; % Price constant

% HEAT EXCHANGER 2
Th2 = 203; % Inlet hot stream temperature [degC]
mh2 = 15; % Mass flow hot stream [kg/sec]
P2 = 1.2; % Price constant

% HEAT EXCHANGER 3
Th3 = 220; % Inlet hot stream temperature [degC]
mh3 = 7.5; % Mass flow hot stream [kg/sec]
P3 = 1.3; % Price constant

% HEAT EXCHANGER 4
Th4 = 235; % Inlet hot stream temperature [degC]
mh4 = 12.5; % Mass flow hot stream [kg/sec]
P4 = 1.5; % Price constant

% HEAT EXCHANGER 5
Th5 = 240; % Inlet hot stream temperature [degC]
mh5 = 20; % Mass flow hot stream [kg/sec]
P5 = 1.4; % Price constant

% HEAT EXCHANGER 6
Th6 = 245; % Inlet hot stream temperature [degC]
mh6 = 17.5; % Mass flow hot stream [kg/sec]
P6 = 1.7; % Price constant

% HEAT EXCHANGER 7
Th7 = 225; % Inlet hot stream temperature [degC]
mh7 = 15; % Mass flow hot stream [kg/sec]
```

```
P7 = 1.5; % Price constant

% HEAT EXCHANGER DATA
m_wall = 3000; % Wall weight HXers [kg]
rho_wall = 7850; % Wall density CS [kg/m3] %7850
Vwall = m_wall/rho_wall; % Volume walls [m3]
Cp_wall = 0.49; % Heat capacity walls CS [kW/kgdegC]

end
```

Dynamic.m

```
% DYNAMIC FUNCTION AND STATE EQUATIONS FOR THE 6:1 HEN

function xprime = Dynamic(t,X,U,N,HXindex)

% Defining the outlet varibles
Th_out = X(1:N);
Twall = X(N+1:2*N);
Tc_out = X(2*N+1:3*N);

% Defining inlet parameters from Simulink
Th_in(1) = U(1);
mh_in = U(2);
Tc_in(1) = U(3);
m0_in = U(4);

% Calling parameters from Data.m file
[T0,Th1,Th2,Th3,Th4,Th5,Th6,Th7,...
            m0,m1,m2,mh1,mh2,mh3,mh4,mh5,mh6,mh7,...
            rho_0,hc,Cp0,...
            Vwall,rho_wall,Cp_wall,...
            P1,P2,P3,P4,P5,P6,P7] = Data;

if HXindex == 1
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.10*hc;
    U = (hh*hc)/(hh+hc);
```

```
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 20.5;


elseif HXindex == 2
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.08*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 28.3;


elseif HXindex == 3
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.08*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 42.6;


elseif HXindex == 4
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.07*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 49.95;
```

```

elseif HXindex == 5
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.10*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 36.5;


elseif HXindex == 6
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.10*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 32.5;


else HXindex == 7
    Cph = 2;
    wh = Cph*mh_in;
    rho_h = rho_0;
    hh = 1.109*hc;
    U = (hh*hc)/(hh+hc);
    Vhot = mh_in/rho_h;
    Vcold = m0_in/rho_0;
    w0 = m0_in*Cp0;
    Ai = 45.5;


end


% STATE EQUATIONS
```

```
% Hot stream
dThotdt(1) = (Th_in(1)-Th_out(1)-((U*Ai)/(wh*N))*...
    (Th_out(1)-Twall(N))*(mh_in*N)/(rho_h*Vhot));

% Wall
dTwalldt(1) = ...
    (hh*(Th_out(N)-Twall(1))-hc*(Twall(1)-Tc_out(1)))*...
    (Ai/(rho_wall*Cp_wall*Vwall));

% Cold stream
dTcolddt(1) ...
    =(Tc_in(1)-Tc_out(1)-((U*Ai)/(w0*N))*(Tc_out(1)-Twall(1)))*...
    ((m0_in*N)/(rho_0*Vcold));


for i = 2:N
    j = N-i+1;
    dThotdt(i) = (Th_out(i-1)-Th_out(i)-((U*Ai)/(wh*N))*...
        (Th_out(i)-Twall(j))*(mh_in*N)/(rho_h*Vhot));
end

for j = 2:N
    i = N-j+1;
    dTwalldt(j) = ...
        (hh*(Th_out(i)-Twall(j))-hc*(Twall(j)-Tc_out(j)))*...
        (Ai/(rho_wall*Cp_wall*Vwall));
    dTcolddt(j)=(Tc_out(j-1)-Tc_out(j)-((U*Ai)/(w0*N))*...
        (Tc_out(j)-Twall(j))*((m0_in*N)/(rho_0*Vcold)));
end

xprime = [dThotdt, dTwalldt, dTcolddt];
```

HX1.m

```
% HEAT EXCHANGER 1

function [sys,x0] = HX1(t,x,u,flag)

HXindex = 1; % HX number
```

```
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX2.m

```
% HEAT EXCHANGER 2

function [sys,x0] = HX2(t,x,u,flag)

HXindex = 2; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
```

```
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX3.m

```
% HEAT EXCHANGER 3

function [sys,x0] = HX3(t,x,u,flag)

HXindex = 3; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX4.m

```
% HEAT EXCHANGER 4

function [sys,x0] = HX4(t,x,u,flag)

HXindex = 4; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX5.m

```
% HEAT EXCHANGER 5

function [sys,x0] = HX5(t,x,u,flag)

HXindex = 5; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);
```

```

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

end

end
```

HX6.m

```
% HEAT EXCHANGER 6

function [sys,x0] = HX6(t,x,u,flag)

HXindex = 6; % HX number
N = 10; % Model order


if abs(flag) == 1
    sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
    sys(1,1) = x(N); % Outlet hot temperature
    sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
    x0 = ssvar(HXindex,N);
    sys = [3*N,0,2,4,0,0];

else
   sys = [];

```

```
end

end
```

HX7.m

```
% HEAT EXCHANGER 7

function [sys,x0] = HX7(t,x,u,flag)

HXindex = 7; % HX number
N = 10; % Model order


if abs(flag) == 1
     sys = Dynamic(t,x,u,N,HXindex);

elseif abs(flag) == 3
     sys(1,1) = x(N); % Outlet hot temperature
     sys(2,1) = x(3*N); % Outlet cold temperature (Tend)

elseif flag == 0
     x0 = ssvar(HXindex,N);
     sys = [3*N,0,2,4,0,0];

else
    sys = [];

end

end
```

ssvar.m

```
% STEADY STATE VARIABLES FOR EACH HEAT EXCHANGER
% IN THE 6:1 HEN

function [x0] = ssvar(HXindex,N)

```

```
    if HXindex == 1

                x0 = [189.4271
                      188.8561
                      188.2871
                      187.7201
                      187.1549
                      186.5917
                      186.0305
                      185.4711
                      184.9137
                      184.3582
                      158.4909
                      158.9577
                      159.4261
                      159.8961
                      160.3677
                      160.8409
                      161.3158
                      161.7923
                      162.2704
                      162.7502
                      130.0368
                      130.4060
                      130.7765
                      131.1482
                      131.5212
                      131.8955
                      132.2711
                      132.6479
                      133.0261
                      133.4055];

    elseif HXindex == 2

                x0 = [201.5136
                      200.0480
                      198.6031
                      197.1783
                      195.7736
                      194.3886
```

```
                    193.0230
                    191.6766
                    190.3491
                    189.0402
                    162.3173
                    163.2459
                    164.1878
                    165.1431
                    166.1119
                    167.0945
                    168.0912
                    169.1020
                    170.1271
                    171.1669
                    133.4566
                    133.9745
                    134.4999
                    135.0327
                    135.5730
                    136.1211
                    136.6770
                    137.2407
                    137.8125
                    138.3925];




    elseif HXindex == 3

                x0 = [215.0666
                    210.3961
                    205.9744
                    201.7884
                    197.8254
                    194.0736
                    190.5217
                    187.1591
                    183.9757
                    180.9619
                    160.5241
```

```
                    162.3875
                    164.3558
                    166.4349
                    168.6310
                    170.9508
                    173.4011
                    175.9893
                    178.7232
                    181.6110
                    138.4513
                    139.0723
                    139.7283
                    140.4212
                    141.1531
                    141.9262
                    142.7428
                    143.6054
                    144.5165
                    145.4789];


    elseif HXindex == 4

              x0 = [231.2098
                    227.5417
                    223.9918
                    220.5563
                    217.2314
                    214.0136
                    210.8995
                    207.8857
                    204.9690
                    202.1462
                    174.8146
                    176.7276
                    178.7043
                    180.7467
                    182.8572
                    185.0379
                    187.2912
                    189.6195
```

```
                192.0253
                194.5111
                145.5698
                146.5094
                147.4802
                148.4833
                149.5198
                150.5909
                151.6975
                152.8410
                154.0226
                155.2435];


    elseif HXindex == 5

          x0 = [238.2897
                236.5973
                234.9227
                233.2655
                231.6257
                230.0031
                228.3974
                226.8085
                225.2363
                223.6805
                191.1307
                192.3423
                193.5668
                194.8042
                196.0547
                197.3184
                198.5954
                199.8860
                201.1902
                202.5082
                155.3259
                156.1590
                157.0009
                157.8516
                158.7114
```

```
                    159.5802
                    160.4583
                    161.3456
                    162.2423
                    163.1485];

    elseif HXindex == 6

                x0 = [243.3016
                    241.6237
                    239.9663
                    238.3289
                    236.7115
                    235.1137
                    233.5353
                    231.9760
                    230.4357
                    228.9141
                    197.6308
                    198.7678
                    199.9189
                    201.0841
                    202.2636
                    203.4577
                    204.6664
                    205.8899
                    207.1285
                    208.3824
                    163.2190
                    163.9331
                    164.6560
                    165.3878
                    166.1286
                    166.8784
                    167.6375
                    168.4060
                    169.1838
                    169.9713];

    elseif HXindex == 7

```

```
                x0 = [223.0233
                      220.9773
                      218.8595
                      216.6675
                      214.3987
                      212.0504
                      209.6198
                      207.1039
                      204.4999
                      201.8046
                      168.0086
                      171.8484
                      175.5582
                      179.1424
                      182.6052
                      185.9507
                      189.1829
                      192.3057
                      195.3227
                      198.2375
                      130.5288
                      135.6380
                      140.5741
                      145.3430
                      149.9505
                      154.4019
                      158.7025
                      162.8575
                      166.8718
                      170.7502];


    end
```

D Simulink Block Diagrams

Simulink block diagrams for all dynamic cases are given in the following Section. The longest networks of four and six heat exchangers in series tended to give a very small figure. The dynamic case I with two heat exchangers in parallel (Figure D.1) is big enough to be read without difficulties and represents the repeating pattern for bigger networks.

Dynamic Case I Block Diagram: `dynamic_11_1.mdl`

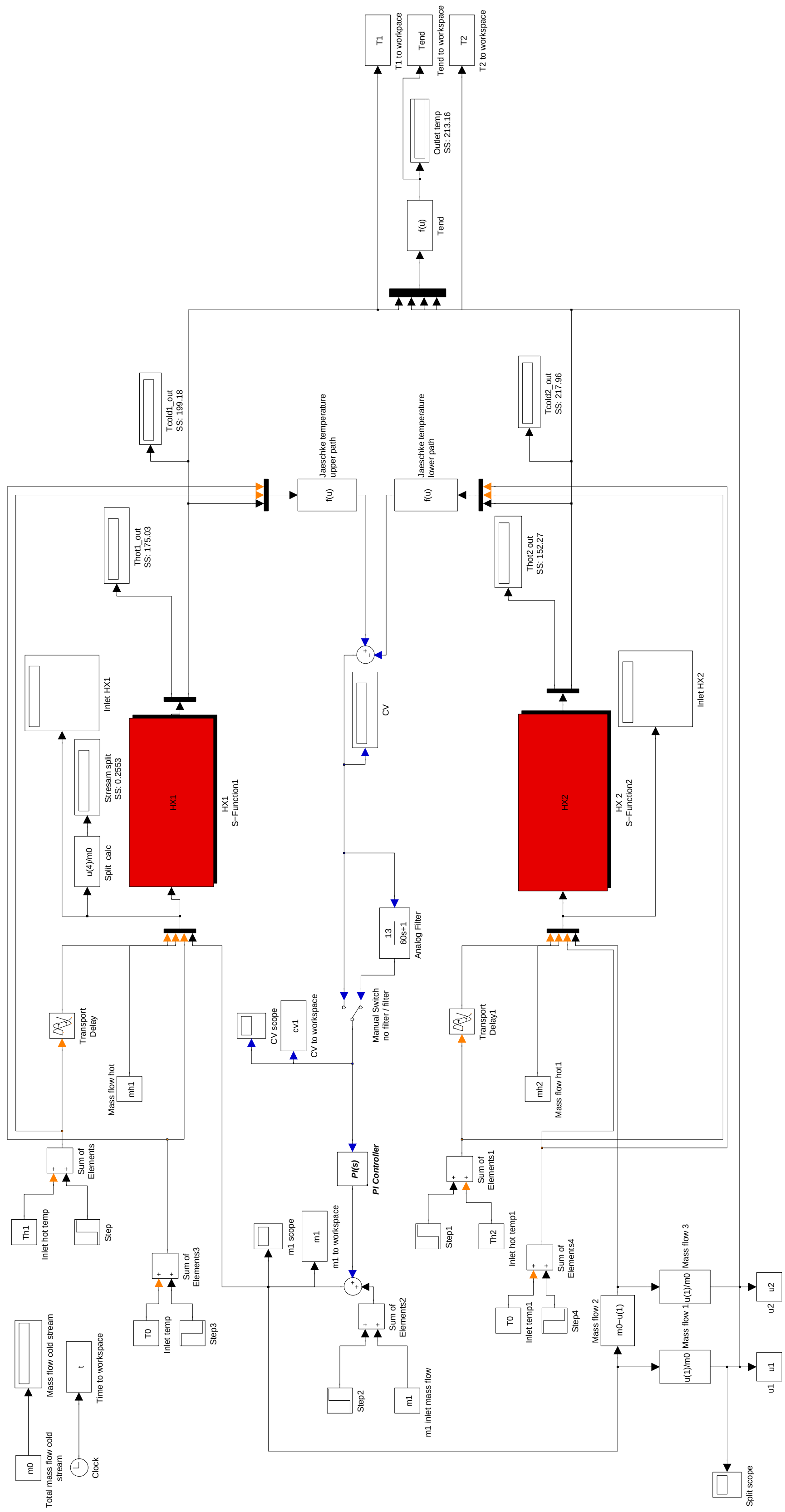


Figure D.1: Simulink block diagram Dynamic case I

Dynamic Case II Block Diagram: `dynamic_21_1.mdl`

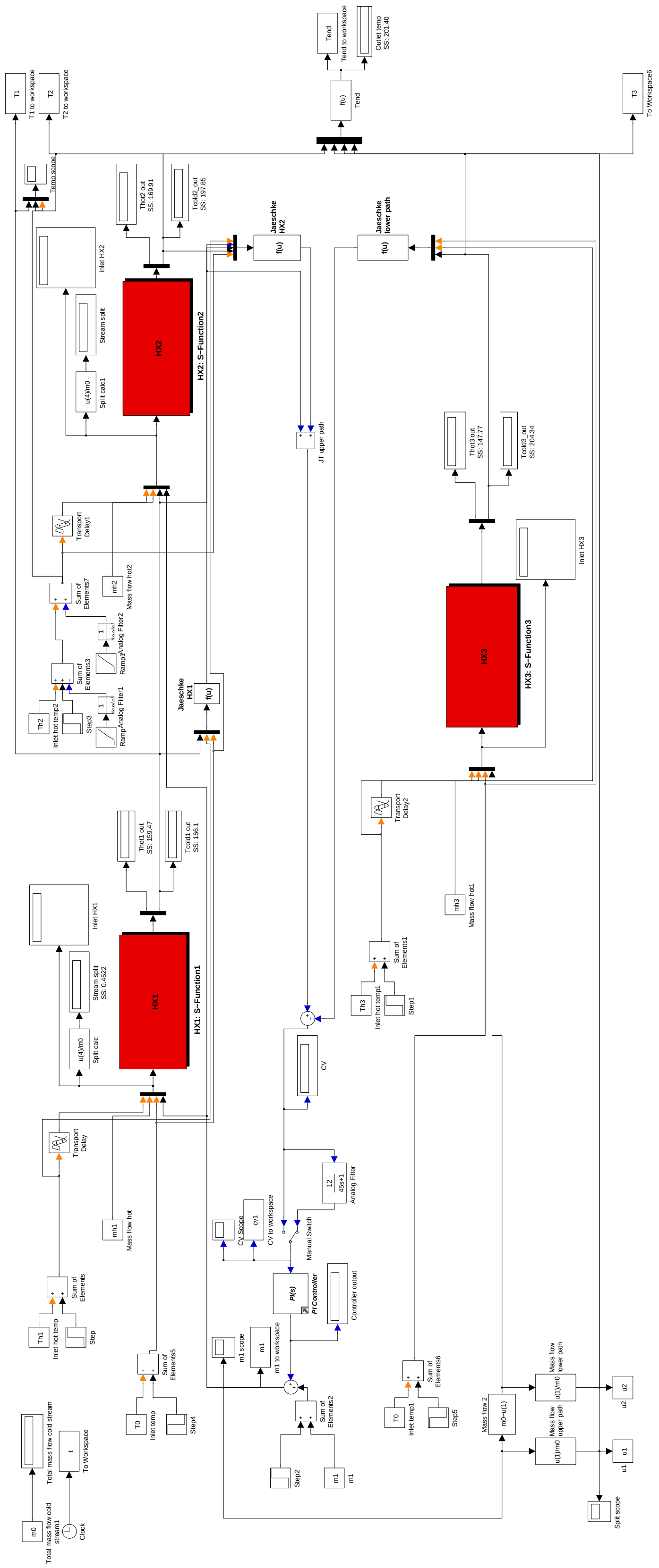


Figure D.2: Simulink block diagram Dynamic case II

Dynamic Case II-a Block Diagram: `dynamic_21_1_1.mdl`

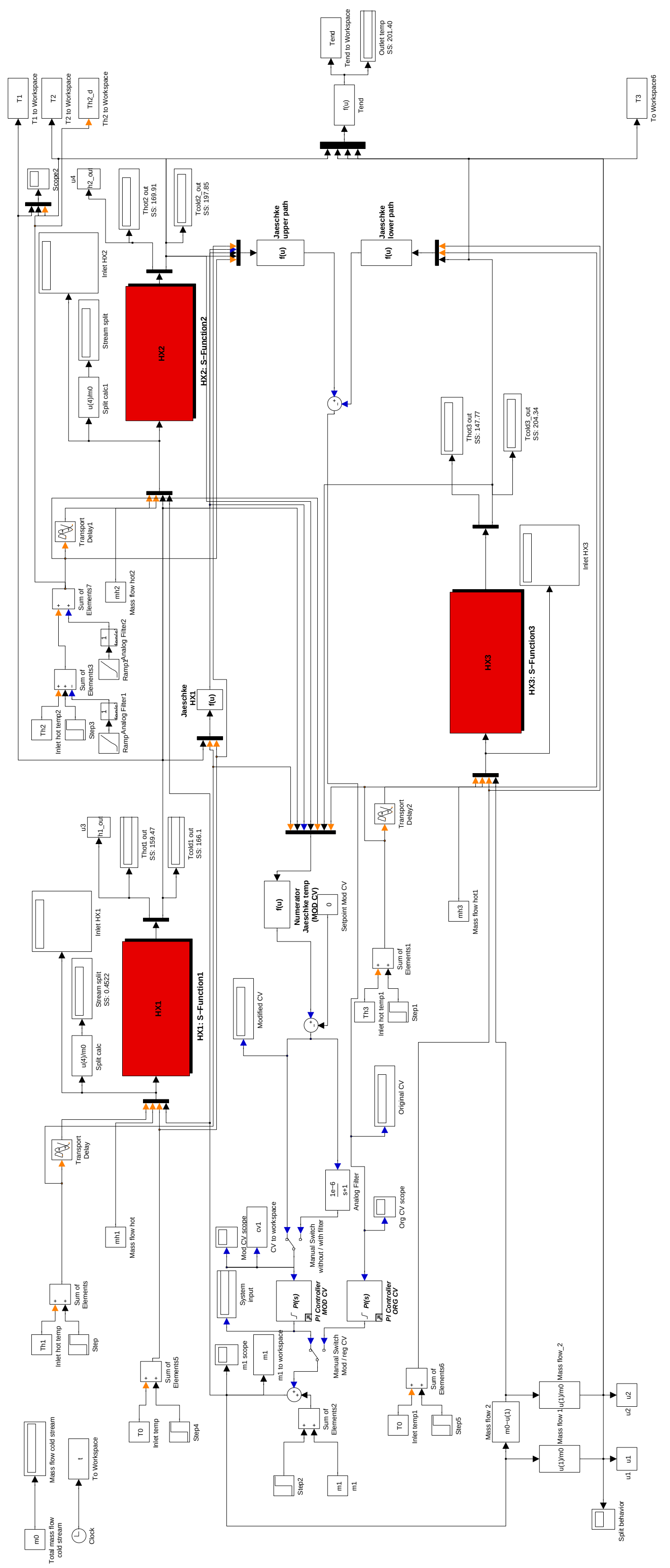


Figure D.3: Simulink block diagram Dynamic case II-a

Dynamic Case III Block Diagram: `dynamic_32.mdl`

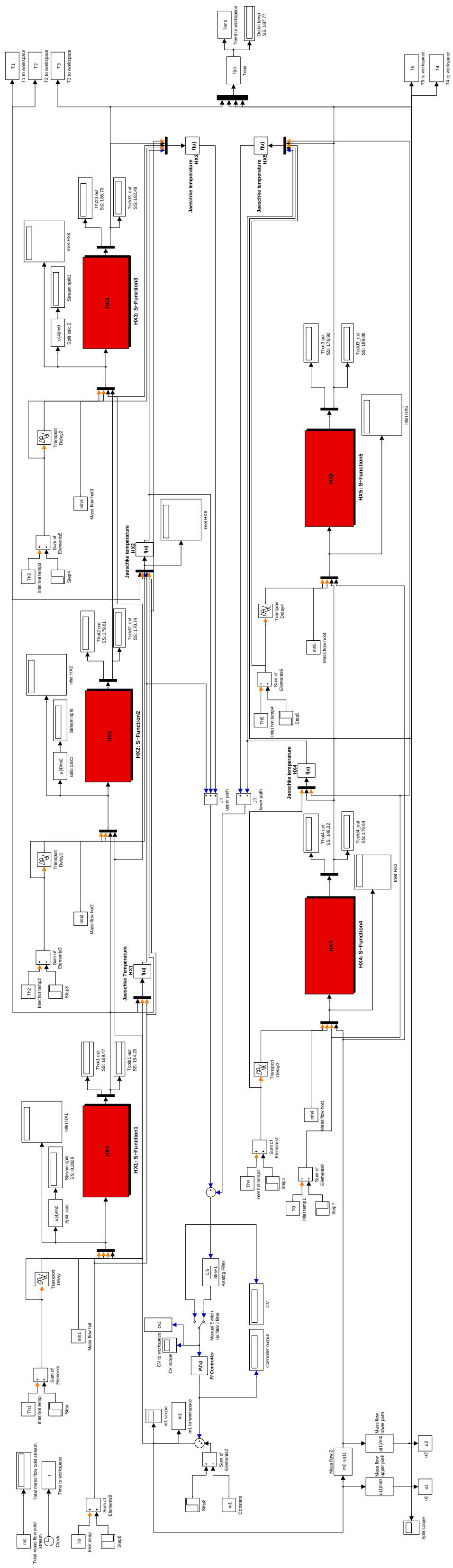


Figure D.4: Simulink block diagram Dynamic case III

Dynamic Case IV Block Diagram: `dynamic_41.mdl`

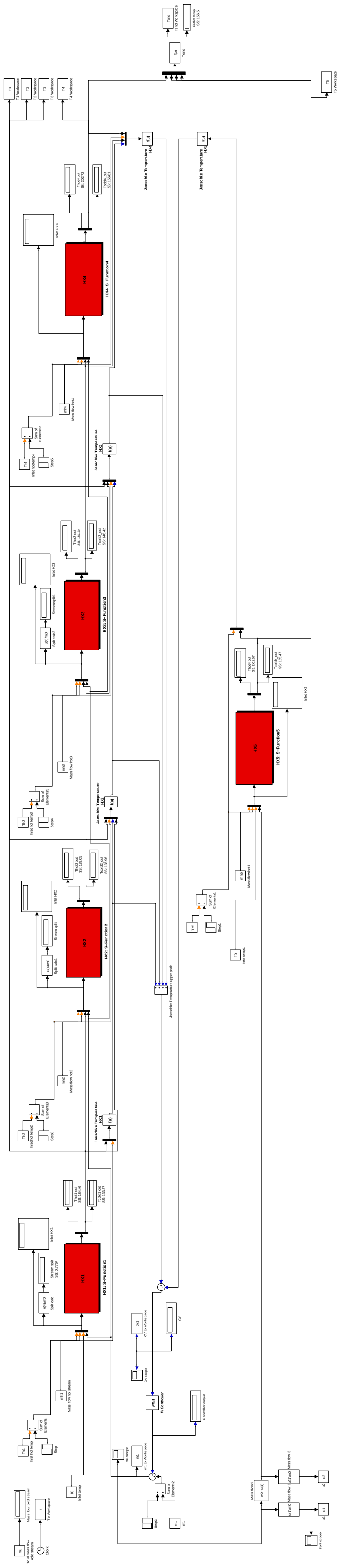


Figure D.5: Simulink block diagram Dynamic case IV

Dynamic Case V Block Diagram: `dynamic_61.mdl`

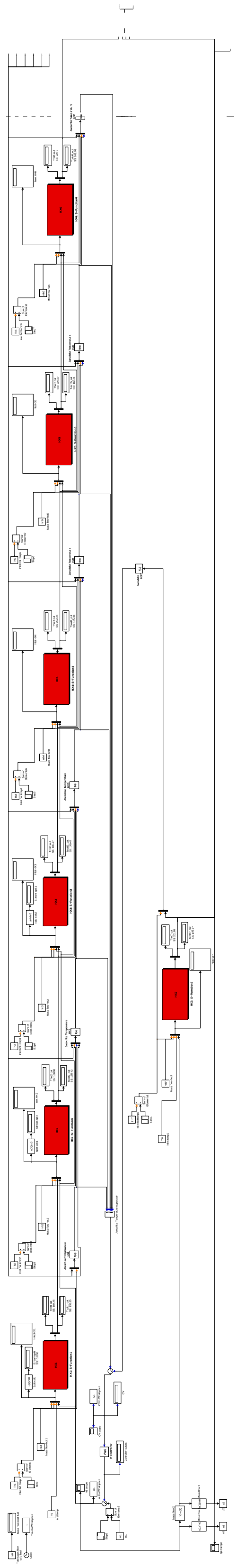


Figure D.6: Simulink block diagram Dynamic case V