:

DESIGN OF A SOFTWARE FOR THE CONSTRUCTION OF HEAT DISTRIBUTION NETWORKS WITH CONCENTRATED SOLAR THERMAL INTEGRATION BASED ON PINCH AND EXERGY ANALYSES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MURAT EKİN İNCE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MECHANICAL ENGINEERING

AUGUST 2019

Approval of the thesis:

DESIGN OF A SOFTWARE FOR THE CONSTRUCTION OF HEAT DISTRIBUTION NETWORKS WITH CONCENTRATED SOLAR THERMAL INTEGRATION BASED ON PINCH AND EXERGY ANALYSES

submitted by **MURAT EKİN İNCE** in partial fulfillment of the requirements for the degree of **Master of Science in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. M.A. Sahir Arıkan _____
Head of Department, **Mechanical Engineering**

Prof. Dr. Tuba Okutucu Özyurt _____
Supervisor, **Mechanical Engineering Department, METU**

Prof. Dr. Derek K. Baker _____
Co-supervisor, **Mechanical Engineering Department, METU**

Examining Committee Members:

Prof. Dr. İlker Tarı
Mechanical Engineering Department, METU _____

Prof. Dr. Tuba Okutucu Özyurt
Mechanical Engineering Department, METU _____

Prof. Dr. Halil Kalıpçılar
Chemical Engineering Department, METU _____

Assist. Prof. Dr. Özgür Bayer
Mechanical Engineering Department, METU _____

Assist. Prof. Dr. Onur Baş
Mechanical Engineering Department, TED University _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: Murat Ekin İnce

Signature :

ABSTRACT

DESIGN OF A SOFTWARE FOR THE CONSTRUCTION OF HEAT DISTRIBUTION NETWORKS WITH CONCENTRATED SOLAR THERMAL INTEGRATION BASED ON PINCH AND EXERGY ANALYSES

İnce, Murat Ekin
M.S., Department of Mechanical Engineering
Supervisor : Prof. Dr. Tuba Okutucu Özyurt
Co-Supervisor : Prof. Dr. Derek K. Baker

August 2019, 202 pages

Searching for a sustainable solution for industry, one of the common strategies is using a Heat Distribution Network (HDN). Pinch Analysis is a common methodology to calculate the pinch point and minimum hot and cold external utilities for a HDN, which then serve as inputs to Pinch Design to build a robust HDN. Pinch Analysis and Design can be supplemented with exergy analysis to identify locations in a HDN with the largest potentials to improve thermodynamic performance. Additionally, Concentrating Solar Thermal (CST) can be used as a sustainable hot utility for a HDN. While automated Pinch Analysis and Design tools exist, they are closed and do not include or allow for exergy analysis or CST integration. In this thesis a new open-source tool called Pinch Concentrated Solar Thermal Exergy Tool (PCSTET) is presented. The tool uses subfunctions for pinch analysis, creation of the HDN including automatic splitting and division of streams, optimization, and plots of HDN diagrams. PCSTET is benchmarked using common examples from the literature and good agreement is found. Additionally, a case study from the literature is analyzed with PCSTET. The optimization results in a 47.7% decrease in exergy destruction as

well as 87.7% reduction in cold utility and 38.2% increase in hot utility requirements. The dependency of exergy destruction reduction and the number of heat exchangers (HEX) is also investigated. Results show that there is no association between the number of HEXs and the amount of exergy destruction. The outputs of PCSTET are intended to be synthesized with other inputs by an expert HDN designer to yield high-performing HDNs.

Keywords: Heat Distribution Network, Pinch Analysis, Exergy, Industrial Parks

ÖZ

ISIL GÜNEŞ ENERJİSİ ENTEGRASYONLU ISI DAĞITIM HATLARININ EKZERJİ-PİNCH ANALAZİNE DAYANAN YAZILIMININ TASARIMI

İnce, Murat Ekin
Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Tuba Okutucu Özyurt
Ortak Tez Yöneticisi : Prof. Dr. Derek K. Baker

Ağustos 2019 , 202 sayfa

Endüstri için sürdürülebilir çözümler bulmak adına en popüler stratejilerden biri ısı dağıtım hatlarının kullanılmasıdır. Pinch analizi bir ısı dağıtım hattı için pinch noktasını, minimum sıcak ve soğuk harici kaynak gereksinimini hesaplamakta kullanılan, bir yöntemdir. Bu hesaplamalar ısı dağıtım hattının tasarımı gerekli girdilerdir. Olası termodinamik iyileştirmelerin belirlenmesi adına pinch analizi ve dizaynı, ekserji ile birlikte kullanılabilmektedir. Ayrıca odaklanmış güneş enerjisi, ısı dağıtım hattı için sürdürülebilir sıcak harici kaynak olarak kullanılabilmektedir. Pinch analizi ve dizaynı için yazılımlar bulunmaktadır. Fakat bu yazılımlar geliştirmeye kapalı olup ekserji analizi veya odaklanmış günej enerjisi entegrasyonuna izin vermemektedir. Bu tezde yeni abir açık kaynak yazılımı olan Pinch Concentrated Solar Thermal Exergy Tool (PCSTET) sunulmuştur. Program pinch analizi, ısı dağıtım hattının oluşturulması, akış ayrılması ve bölünmesi, optimizasyon ve ağ gösterimi çizimi isimli alt fonksiyonlar kullanmaktadır. Program tasarımı tamamlanırken literatürden bulunan örnekler ile yapılan doğrulamalar uyumlu sonuçlar göstremiştir. Ek olarak literatürden bulunan bir vaka çalışması PCSTET ile incelenmiştir. Optimizasyon ekjerji yıkımını 47.7% azalmış, harici soğutma ihtiyacını 87.7% azaltırken harici ısıtma

ihtiyacını 38.2% arttırmıştır. Ayrıca ekserji yıkımı ve ısı değiştiricisi sayısı arasındaki ilişki kontrol edilmiştir. Sonuçlar ısı değiştiricisi sayısı ve ekserji yıkım miktarı arasında bir bağlantı olmadığını göstermiştir. PCSTET aracılığı ile bulunan sonuçların, uzman bir ısı dağıtım hattı tasarımcısı tarafından, yüksek performanslı bir ısı dağıtım hattı tasarımı için başka girdiler ile sentezlenmesi beklenmektedir.

Anahtar Kelimeler: Isı Dağıtım Hatları, Pinç Analizi, Ekserji, Organize Sanayi Bölgeleri

to my dearest family and lovely Ayçin

ACKNOWLEDGMENTS

One of the most influential people that get me here is Prof. Dr. Derek Baker. He was always there for me whenever I have a problem. He informed me about academia, opportunities, trends, and research strategies. He always encouraged me and appreciated my studies. I have consulted him too many times on too many different topics. Having a co-supervisor that you can talk to about your carrier and personal life as well as your research is a blessing. I consider myself lucky to take many courses from him and having him as my co-supervisor.

I want to thank my supervisor Prof. Dr. Tuba Okutucu. Her kind acceptation of supervision, mid-term made my research process more comfortable. Her guidance through the process helped me.

The help of my old classmate Mert İşler was precious. Having a friend who is selfless as much as he is competent, comforted me through my studies.

I also want to thank Dr. Onur Baş and Dr. Omer Music for their time and guidance through coding and defense processes.

I've postponed countless meetings, missed so many events for our little group of friends. Although they gave me a hard time for planning and working too much, my big brothers Mustafa, Ünsal, and Doğuş were always there for me when I need a laugh or support.

I am really thankful for my family. They made my life easier; they believed in me and sincerely empathized with me for times of struggle. Their love and support brought me to these days.

For lastly, but most importantly, I want to thank my future wife, my girlfriend Ayçin. Her love and encouragement help me through dark times for the last 12 years. She always tried to find solutions to my problems. I always knew there is a person on the line that I can tell my problems and feelings without being judged. Goals and dreams

for our future together made me push myself one step further. Being best friends with your partner is the greatest gift in the world, and I am very grateful for that.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

PCSTET	Pinch Concentrated Solar Thermal Exergy Tool
HDN	Heat Distribution Network
INSHIP	Integrating National Research Agendas on Solar Heat for Industrial Processes
IEA	International Energy Agency
PV	Photo-Voltaic
DHC	District Heating and Cooling
HEN	Heat Exchanger Network
CSP	Concentrated Solar Power
FPC	Flat Plate Collector
ETC	Evacuated Tube Collector
CPC	Compound Parabolic Collector
LFC	Linear Fresnel Collector
PTC	Parabolic Trough Collector
CTC	Cylindrical Trough Collector
PDC	Parabolic Dish Collector
HFC	Heliostat Field Collector
T_{high}	High Temperature of the Stream
T_{low}	High Temperature of the Stream
ΔT_{min}	Minimum Temperature Difference Between Hot and Cold Streams
T_{pinch}	Pinch Temperature
CP	Constant Pressure Heat Capacity
H	Heat Load
H'	Modified Heat Load

CC	Composite Curve
GCC	Grand Composite Curve
ECC	Exergy Composite Curve
KE	Kinetic Exergy
PE	Potential Exergy
CE	Chemical Exergy
B	Exergy
b	Specific Exergy
b_f	Specific Flow Exergy
ϵ^{ch}	Specific Chemical Exergy
B_d	Exergy Destruction

CHAPTER 1

INTRODUCTION

In industrial parks, multiple functions including production, relaxation, and education are co-located in an industrial area with high value and employment. For the primary purposes of the economy, industrial parks should be designed in a way that, the design promotes growth in industrial services and production [6]. Energy management is quite vital for an industrial park as well as a single facility in the industrial park. The types of energy that are handled can be named as chemical, mechanical, electrical, and thermal energy.

Energy harvesting is often defined as a technique or a device to utilize unused energy. For instance, the solar radiation incidence to the uncultivated land in an industrial park can be utilized via collectors or photovoltaic panels (PV). In another example, the excess heat dumped out from any facility may be used to meet thermal duty for another facility with the use of Heat Distribution Networks (HDNs). Although harvesting from wind energy or kinetic energy from any moving part is also possible, this study will spotlight the first two examples. The first one is the use of concentrating solar thermal (CST) collectors in industrial parks to harvest energy, and the second one is the use of HDNs in industrial parks to recover thermal energy.

Before moving to the thesis overview, the concepts and technologies that the overall study is based on will be introduced briefly. Then how all of these are related to each other in the scope of this thesis will be clarified.

This research is motivated by and strongly aligned with the European Union (EU) Horizon 2020 project *Integrating National Research Agendas on Solar Heat for Industrial Processes (INSHIP)* as detailed in subsection 1.5.1.

1.1 Heat Distribution Networks

The utilization of thermal energy in different locations or processes requires a thermal network, and there are various kinds of names in the literature for this. District Heating and Cooling (DHC) is the most common and the oldest one. Heat Distribution Network (HDN) and Heat Exchanger Network (HEN) terms are relatively new but frequently used terminology. Initially, thermal networks were developed to transport thermal energy from a source to a single end-use. However, with the passing of time, it has become common for HDNs to consist of more complex systems of heat exchangers (HEXs) that potentially link multiple sources and end-uses [7].

It is possible to have quite different interacting sides for all of these actions. For example, it is possible to design a HDN in which a household and an industrial facility interact. Conversely, another HDN may connect different heat loads and duties in a single facility.

Despite the diversity of HDNs, in the literature, it is common to characterize HDNs as consisting of three main components. The first one is the central thermal energy such as a Central Heating Plant. The second one is the Distribution System. The third one is the end-use interconnection and Heat Consumers (Figure 1.1).

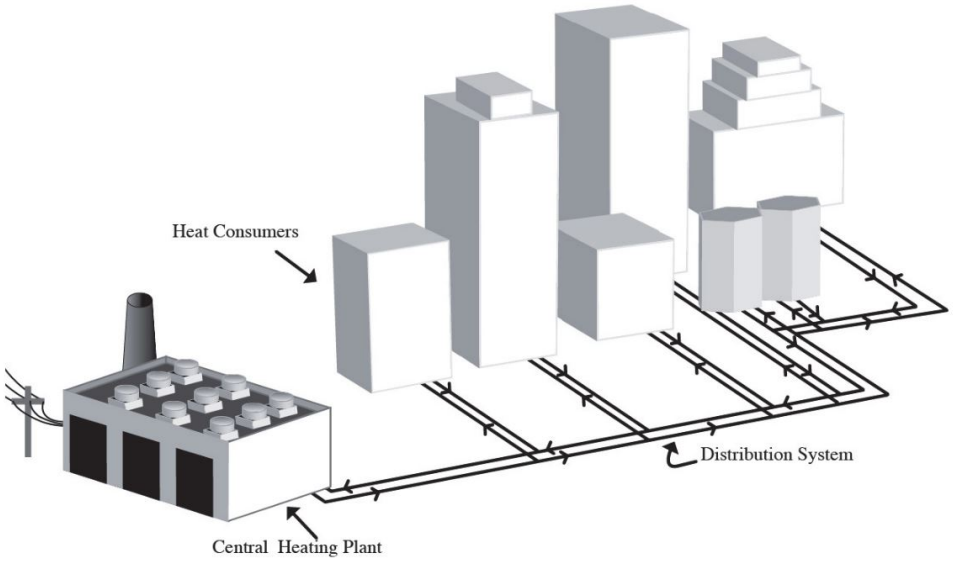


Figure 1.1: Representation of Heat Distribution System [3]

A HDN has two main benefits: environmental and economy. Central heat generation can yield environmental benefits relative to distributed heat generation, such as by enabling fuels to be used that are not feasible for use in industrial facilities, allowing for higher efficiency equipment, and allowing for tighter monitoring and control of this equipment. Considering the economic concerns, although central heat generation may require higher capital costs up-front, many factors can make central heating economically better option during operation. Staff costs can be reduced because HVAC systems can be automated. Dangerous equipment can be removed from buildings or facilities and maintained where the professionals are located, which means fewer accidents and as a result, reduced insurance costs. Large scale systems can reach high thermal and emission efficiencies when compared to small systems [3].

At this point, it is necessary to point out that, even though all these advantages are explained for central heat generation, they also apply to regenerative systems that use waste thermal energy. Thus, the integration of central heat generation using CST with distributed heat recovery through a HDN can have many positive benefits. Rather than using any fuel in the center and distribute the thermal energy produced, the current study will examine a different network. A hot stream from a specific facility is a source for a specific heating process. Conversely, a cold stream from another facility may be used to drive a cooling process.

1.2 Pinch

In order to transfer heat between distinct media, the necessary condition is a temperature difference between them. Pinch theory arises from this rule.

This work focuses on HEXs with two streams undergoing sensible heating or cooling, and the underlying methodology can be extended to streams undergoing latent heating or cooling. Initial and final temperatures of those streams also can be considered as objectives. As a constraint, a minimum temperature difference of ΔT_{min} to drive heat transfer between these two streams is imposed due to design considerations such as the type or size of the HEX. Newton's Law of Cooling dictates inverse proportionality between temperature difference and heat transfer area for a fixed rate of heat transfer.

The study tries to establish matches for pre-defined equipment and area. Thus, there is a pre-defined temperature difference. This concept between two streams can be projected to cases with multiple streams. For a cluster of streams, before the HDN establishment, whose initial and final temperature and heat capacity rates are pre-defined, the journey from initial to final temperature includes three classes of heat interactions;

Heat Recovery System in which the cooling of one stream is used to heat another stream.

Cold Utilities in which an external sink is used to cool a stream.

Hot Utilities in which an external source is used to heat a stream. The following sections will reveal that, for this study, some portion of the hot utilities are CST systems.

Benefiting from all available streams, with ΔT_{min} constraint a solution to meet all targets with minimum external utilities for heating and cooling is produced by pinch analysis. Due to having lots of different media, there are considerable amounts of different solutions. Offering one of these solutions, the possibility of improvement to it also revealed, by pinch analysis. If the requirement for hot and cold utilities match exactly with the values obtained prom pinch solution, one may say a solution with optimal performance is found.

In order to understand the overall process better, a clear distinction between pinch analysis and pinch design is made here.

The output of the pinch analysis is the pinch point and theoretical minimum hot and cold utilities, but not a HDN design. In the pinch design one specific arrangement for a HDN is created by comparing the performance of this specific arrangement to theoretical minimum hot and cold utilities that are found in the pinch analysis. The HEXs are identified by CP values, initial and final temperatures for streams going through them. Besides the streams that reach their final state, hot and cold utilities are identified with heat load values. The deviation of magnitudes of the hot and cold utilities' heat load values from that obtained from the pinch analysis is the

deviation from perfection. Study of both sections will demonstrate the details about the procedure.

1.2.1 Pinch Analysis

Although the final aim is the establishment of the HDN, values of pinch temperature (T_{pinch}), amount of hot and cold utility requirements should be found first, and this is called pinch analysis.

For pinch analysis, high (T_{high}) and low (T_{low}) temperatures are noted as well as the heat capacities (CP) of the streams at the beginning.

It is important to realize the distinction between T_{high} and T_{low} for every stream. For streams to be cooled, low temperature defines the target while for streams to be heated high temperature sets the final temperature value. From another perspective, one may think about the HEXs in the HDN and for a specific HEX;

$$T_{low,cold} = T_{inlet,cold} \tag{1.1a}$$

$$T_{high,cold} = T_{outlet,cold} \tag{1.1b}$$

$$T_{low,hot} = T_{outlet,hot} \tag{1.1c}$$

$$T_{high,hot} = T_{inlet,hot} \tag{1.1d}$$

Heat load is the absolute thermal energy change of the stream for a specific interval and calculated by Equation 1.2. This equation always gives positive values. The majority of the discussion uses positive values for both hot and cold streams. However, in reality, heat load values for cold streams are negative, and heat load values for hot streams are positive.

$$H = CP \cdot \Delta T = CP \cdot (T_{high} - T_{low}) \tag{1.2}$$

One of the most important visual representations in pinch discussion is Composite Curves (CC). Although herein these diagrams are perfectly linear due to constant CP values, in some studies, temperature and heat capacity dependency is taken into account, and those lines become curves. Since this is the common terminology in the literature, they are referred to as curves here.

In order to plot CC, first temperature and heat load curves ($T - H$) should be plotted. Starting from the smallest temperature value, the total heat load is calculated for every temperature interval for hot and cold streams separately. According to the availability of streams, total values are found by the addition of the values that are found with Equation 1.2.

Three of the possible cases for $T - H$ curves are studied here in order to provide a better understanding.

Firstly, T-H curves can barely touch each other (Figure 1.2).

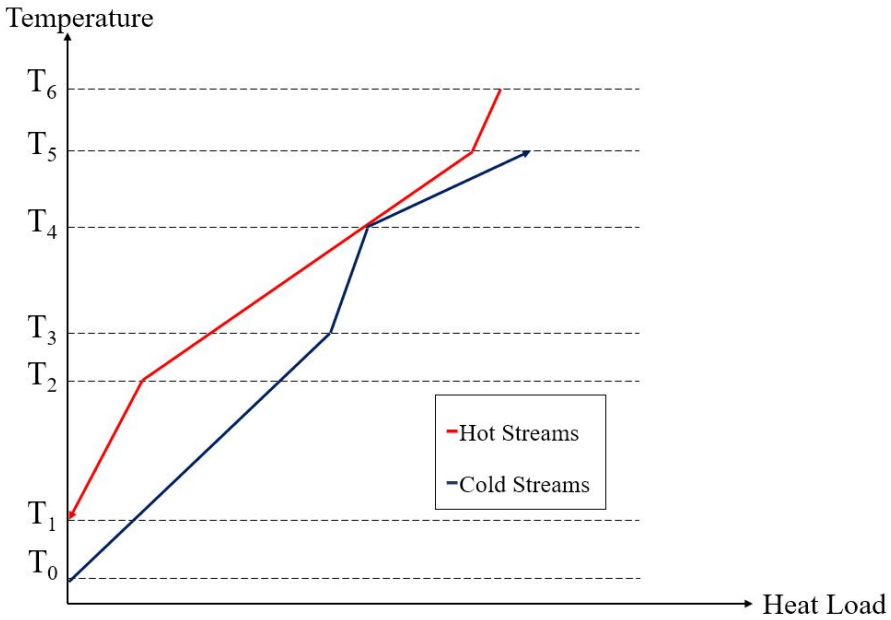


Figure 1.2: T-H Diagram

Secondly, the cold T-H curve may be above the hot T-H curve at some points (Figure 1.3).

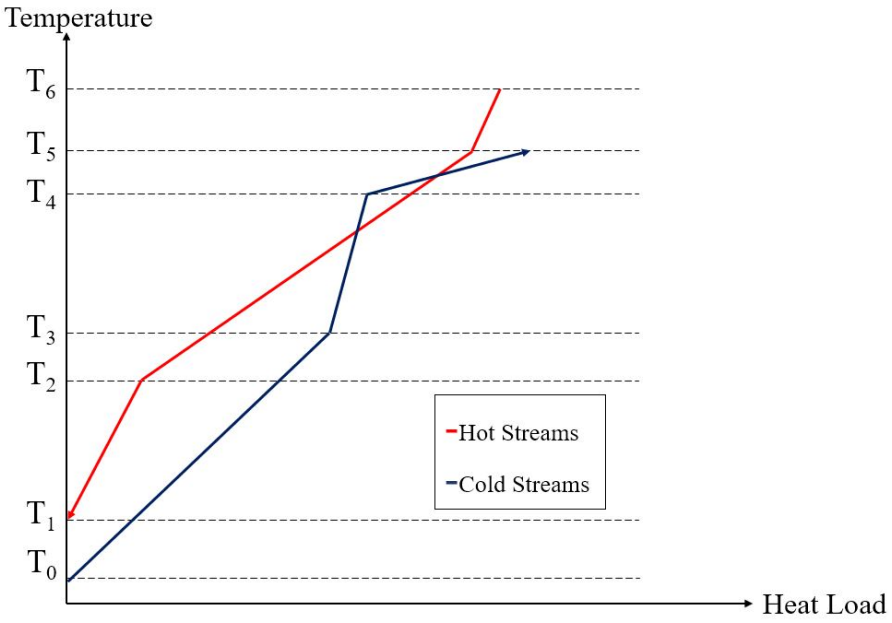


Figure 1.3: T-H Diagram

Lastly, the hot $T - H$ curve may be uniformly above the cold $T - H$ curve. Nevertheless, the distance may not be exactly equal to the ΔT_{min} constraint (Figure 1.4).

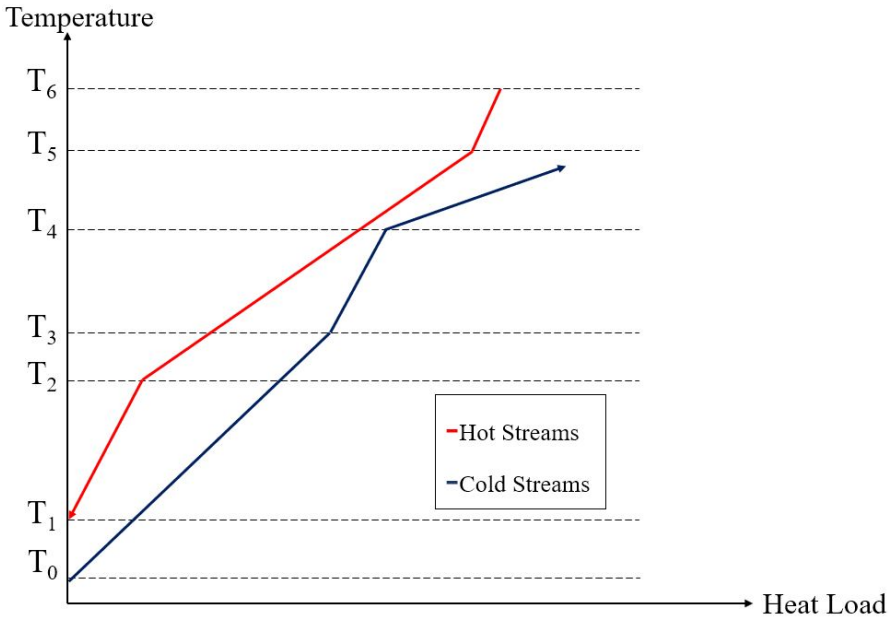


Figure 1.4: T-H Diagram

CCs (Figure 1.5) can be constructed from these T-H curves by moving cold the T-H curve horizontally along the heat load axis until the minimum ΔT between the hot and cold streams is equal to the ΔT_{min} constraint. The location of delTmin is referred to as the Pinch Point. In Figure 1.5, the heat load amount before the start of the cold composite curve shows the required cold utility amount (H_{cu}) and the heat load amount after the end of hot composite curve expresses the required hot utility amount (H_{hu}).

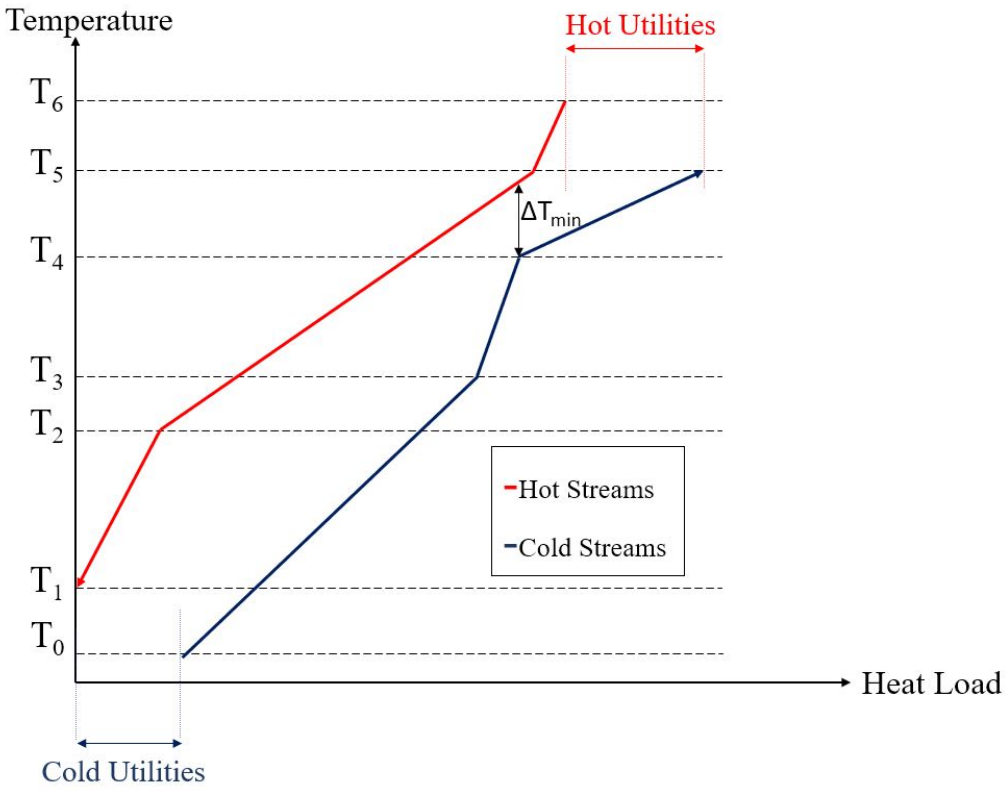


Figure 1.5: T-H Diagram

Two possible methods arise for CCs construction;

- Cold T-H curve is moved along heat load axis until minimum temperature difference between becomes the predefined ΔT_{min}. This final form of T-H curves is called CCs. The midpoint of hot and cold CCs where ΔT_{min} occurs, marks the T_{pinch}. As explained above, amount of shift is equal to H_{cu}, horizontal distance between end points of CCs gives the H_{hu}. Note that this method does not require any other information other than ΔT_{min} and $T - H$ curves and it is an

alternative to the problem table algorithm method discussed below. However, it is less robust.

- Alternatively and almost always, CCs are obtained with the help of the information obtained by problem table algorithm method. The amount of movement of the cold T-H curve along heat load axis is equal to H_{cu} found in problem table. H_{hu} and ΔT_{min} values in CCs are automatically equal to the values found in problem table.

The Pinch Point is defined as the location where the temperature difference $\Delta T = T_{hot} - T_{cold}$ is minimum. In Pinch Analysis, a HDN is assumed that results in this ΔT being equal to an inputted constraint ΔT_{min}. Therefore for a given set of inputs Pinch Analysis identifies the Pinch Point and H_{hu} and H_{cu}. However, in Pinch Analysis the detailed build of the HDN to achieve this Pinch Point, H_{hu} and H_{cu} is a black box and is specified through the subsequent Pinch Design step. A Pinch Analysis procedure is proposed as the problem table algorithm method by Linnhoff et al. [1]. Steps of entire procedure are explained as follows with a detailed example given in section section 2.1;

- Shifted temperatures are calculated whereby temperatures of the cold streams are increased $\Delta T_{min}/2$ whereas the temperatures of hot streams are decreased $\Delta T_{min}/2$. Since heat load value only depends on the amount of temperature increase of decrease (Equation 1.2) there is no harm in using shifted temperature values in heat load calculations. At the following steps, the point with total heat load of zero will be identified with shifted temperatures which requires same temperature values for hot and cold streams. By switching back to original (unshifted) temperature values, ΔT_{min} condition is satisfied automatically. This does not cause any problems since switching back does not change the temperature intervals and relative greatness of streams.

- All of the shifted temperature values are tabulated in ascending order.

- Temperature increment at every interval is calculated.

- Total heat load value (ΔH) for each interval should be calculated. In order to do so, Equation 1.2 is used. Instead of a single CP value, one may calculate

the difference between hot and cold CP values. The product of this total CP and related temperature interval gives the total heat value in the related interval.

- Starting from zero heat load at the highest temperature of the overall interval, cumulative of ΔH will give the load at the end of each interval (H).

- Some of those H values may be negative, which corresponds to the hot stream being at a lower temperature than the cold stream. Making the smallest value zero by subtracting the minimum value to all of the loads calculated for every interval marks the pinch point. These values are called modified loads and denoted by H'. The H' becomes zero only at T_{pinch}, and it is certain that all of the heat loads are positive. Besides, the plot of $T - H'$ is another important visual representation for pinch concept, and it is called Grand Composite Curve(GCC).

- The modified heat load value at the lowest temperature in the overall temperature interval is equal to H_{cu}. The modified heat load value at the highest temperature in the entire temperature interval expresses H_{hu}.

At the end of these steps T_{pinch}, H_{hu} and H_{cu} are calculated. Overall this methodology is called the problem table algorithm method. As noted above, a detailed example for this method can be found in section 2.1.

As a last note, the sole purpose of pinch analysis is to benefit most from available streams between endpoints by specifying the Pinch Point, H_{cu} and H_{hu}. The whole array of streams can be considered as a black box. Supplying thermal energy at the high-temperature end, all of the streams reach to the desired temperature with thermal energy disposal with the low-temperature end.

1.2.2 Pinch Design

Pinch design is the complete procedure of reaching the targets set in both the problem statement and the pinch analysis by the creation of HDN. Previously,the energy targets H_{hu} and H_{cu} are set through pinch analysis. The fundamentals of the procedure

are explained in this section. Initially, the basic guidelines to follow for pinch design are given. Then details are explained.

- The problem is broken into two as below and above pinch. However, the boundaries of those two sub-problems need extra attention. ΔT_{min} that is defined in the pinch analysis, in fact, determines the location of streams. Even though the breakpoint is T_{pinch}, in order to ensure correct amount of minimum temperature difference boundaries are defied as $T_{pinch} + \Delta T_{min}/2$ for hot and $T_{pinch} - \Delta T_{min}/2$ for cold streams.
- For both sections, the design should start from pinch and proceed towards the high, or low temperature ends. This is favorable for two reasons. The first one is the auto validation of ΔT_{min} criteria. If a hot and a cold stream passes through pinch point, it is certain that at the pinch point they satisfy ΔT_{min} condition. The second one is the fact that hot and cold utilities are located at two ends of the spectrum. So it makes more sense to move towards these ends from T_{pinch}.
- Provided that ΔT_{min} criteria is met, hot and cold streams are matched. Each pair represents a heat exchanger, and the overall layout defines the HDN. However, it should be taken into consideration that the minimum temperature difference is not the only factor to consider. Thus, building heat exchangers requires the impositions related to CP, and the number of streams N.

  Above pinch;

$$CP_{hot} \leq CP_{cold} \tag{1.3a}$$

$$N_{hot} \leq N_{cold} \tag{1.3b}$$

  Below pinch;

$$CP_{cold} \leq CP_{hot} \tag{1.4a}$$

$$N_{cold} \leq N_{hot} \tag{1.4b}$$

  conditions should be satisfied. In order to explain Equation 1.3a and Equation 1.4a 1st Law of Thermodynamics for an heat exchanger without any heat

loss while 2 streams enter and leave the control volume at steady state is recalled;

$$\dot{E}_{st} = \dot{E}_{in} - (\dot{E}_{out} + \dot{E}_{loss}) \tag{1.5}$$

$$\dot{E}_{st} = \dot{E}_{loss} = 0 \tag{1.6}$$

$$\dot{E}_{in} = \dot{E}_{out} = \dot{H}_{in} = \dot{H}_{out} \tag{1.7}$$

For the same amount of load change, conditions Equation 1.3a and Equation 1.4a are justified with four different cases (A-D) which are demonstrated in Figure 1.6.

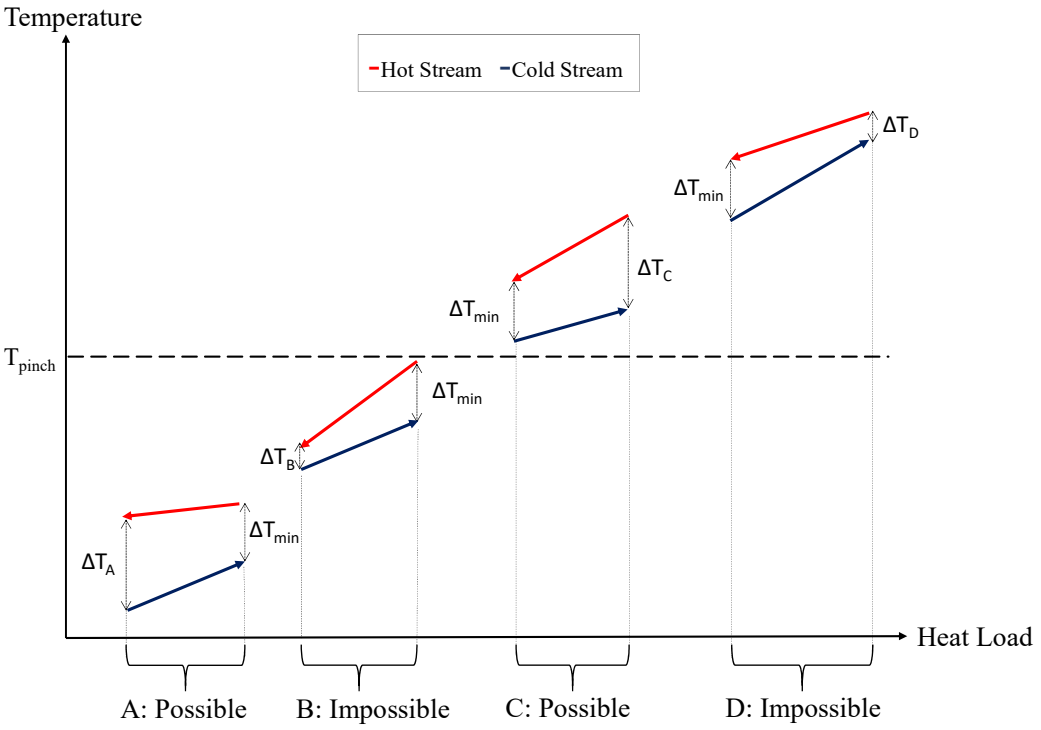


Figure 1.6: Hypothetical cases for the demonstration of CP requirements

It should be noted that streams are in counter-flow arrangement as shown here. Shown ΔT_{min} value is specified at the beginning of the problem statement. Referring to Equation 1.7 and Equation 1.2, bigger CP introduces a smaller

temperature change between inlet and exit. For the case A, CP value is greater for the hot stream than it is for the cold stream. So ΔT_A is greater than, ΔT_{min}. For the case B, CP of the hot is greater than the cold. This brings a temperature difference of ΔT_B, which is less than ΔT_{min}. Similar cases are valid for also the streams located above pinch. The case C includes a hot stream with a greater CP. In the end, ΔT_C turned out to be greater than ΔT_{min}. Nevertheless, for the case D, the cold stream's CP is greater, and thus ΔT_D is smaller than ΔT_{min}. To sum up, second and fourth examples are violations to Equation 1.3a and Equation 1.4a which results in violation to ΔT_{min} condition. The equal CP case is not demonstrated since it results in the same ΔT for the beginning and for the end. As a last note, it is possible to have a problem that satisfies the ΔT_{min} condition even if it violates the CP constraint. For instance, in case B, if the difference between hot and cold streams' temperature is higher than predefined ΔT_{min} at the right hand side, it is possible to satisfy ΔT_{min} condition at the left hand side even if CP constraint is violated. However, in order to make sure that there is a solution, it is best to manipulate streams in a way that they satisfy the CP and ΔT_{min} criteria together.

The reasoning behind Equation 1.3b and Equation 1.4b is quite straight forward. Logically in order to pair from a multiple-member group which includes only two different kinds of entities, if it is desired to consume the first kind entirely. It should be certain that the number of members from the second group should be equal or larger. Hence the validity of Equation 1.3b and Equation 1.4b, also can be comprehended.

- It is not natural to expect any of these conditions is fulfilled automatically. Generally, some of the streams do not meet the criteria mentioned. Arbitrary match of the streams may lead to poor usage of potential in terms of heat recovery. In order to obtain a solution to this problem, two techniques are employed. The first technique is called stream splitting. It is the apportion of streams in order to establish correct matches while meeting ΔT_{min} and CP conditions. For instance, having only one hot stream above pinch with a CP value bigger than all of the cold streams yields no stream match even if ΔT_{min} value is suitable. As a result, in order to satisfy Equation 1.3a, a portion of the hot stream is split. This

is repeated as necessary. Then, matches are placed accordingly. The second technique is not defined as a strategy in the literature, even though it is used. This is called stream division. Its main purpose is satisfying the 1st Law of Thermodynamics in a HEX with no heat loss. Thus heat loads are equated by breaking a stream's temperature continuum into pieces. Subsequently, by circulating a stream through different HEX in the right order, invalid ΔT values may become valid. In a way, this is also a division of stream. To elaborate, for a hypothetical case, Stream X can match with Stream Y and Stream Z. However, the initial pairing of X and Y may make Stream X unusable before the overall energy target is met. Thus, it may be more beneficial to follow a different path. Provided that pairing X', which is the output of X after X and Z match, with Y results in better heat recovery performance, the usage of stream divide for this case is justified. The division of streams is also required in almost all cases. Because it is quite rare to have hot and cold streams that automatically have the same heat loads.

- At a certain point, there will be no streams to match while obeying the CP and ΔT_{min} criteria. For this point, the employment of utilities allows them to achieve the target temperatures. If the total heat load values for the utilities calculated at this step are equal to H_{cu} and H_{hu} values, it can be said that optimum HDN is created with pinch design.

Source and sink terms are widely used in pinch studies in a different context. As it is known, above pinch, there is a hot utility requirement. Having a higher temperature media may fulfill this need. Since the flow of thermal energy is from high temperature to low temperature, the cluster of streams above pinch can be considered as a sink. A similar situation also applies to the cluster below the pinch. The requirement of cold utilities means that this group can be used as a heat source[4]. In the following sections, different use of source and sink terms will be revealed (section 3.3).

Guidelines are absolutely beneficial but are not complete without three golden rules;

1. Heat transfer should not occur from a hot stream above the pinch ($T_{hot} > T_{pinch} + \Delta T_{min}/2$) to a cold stream below the pinch ($T_{cold} < T_{pinch} - \Delta T_{min}/2$). In other words, it is not possible to match a hot stream above the pinch with a cold stream below the pinch for a HEX establishment.
2. Cold utilities should not be used above the pinch.
3. Hot utilities should not be used below the pinch.

The reasoning behind these golden rules is based on the four cases shown in Figure 1.7.

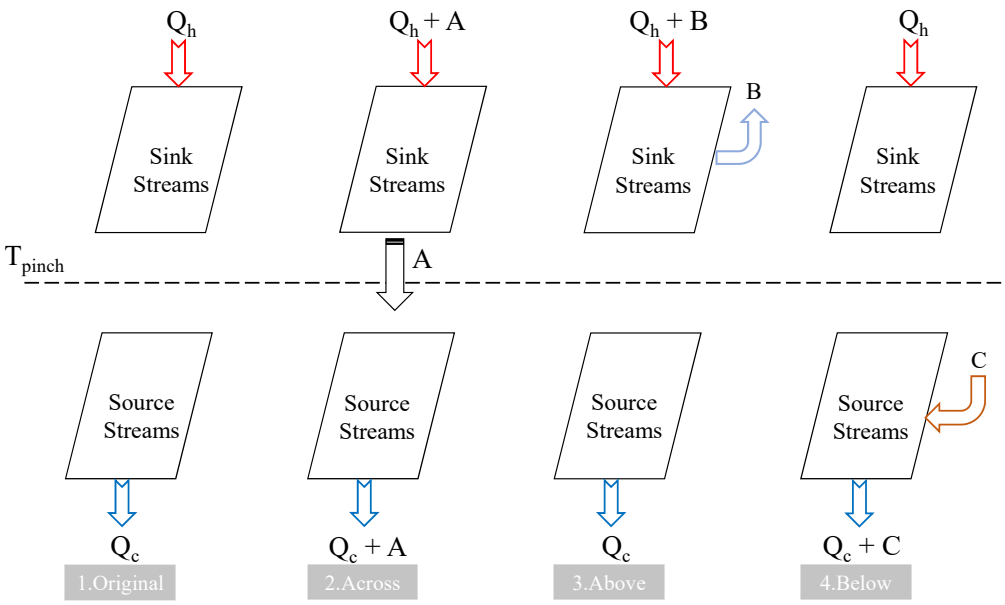


Figure 1.7: Four cases to illustrate the three Golden Rules for Pinch Design: 1.Original (just sink and source), 2.Across Case(heat transfer across the pinch), 3.Above (additional heat extraction above pinch) and 4.Below (additional heat addition below pinch), adapted from Linnhoff's study [4].

Compared to the original case, if heat transfer from the sink to the source occurs, the same amount of increase is observed for hot and cold utility requirements. This

basically results from the application of the 1st Law of Thermodynamics to sink and source blocks. Importantly, stream properties such as inlet, outlet temperatures, mass flow rates, and specific heats are fixed. In other words, both of the control volumes drawn around sink streams and source streams are satisfied in steady-state conditions. Violation of the last two golden rules and consequences are demonstrated in the last two representations of Figure 1.7. Above pinch, an extra amount of heat removal, which works as a cold utility in a sense, requires that much extra heat addition which increases hot utility. Below pinch additional heat input is required; in other words, the employment of a hot utility requires more cooling. So fundamentally any amount of addition requires an equivalent amount of removal.

1.3 Concentrated Solar Thermal (CST)

The use of solar energy to produce electricity and to heat domestic water is a well-known technology. Nevertheless, the mechanism used for converting solar energy to the type of energy that will be utilized by the end-user might not be that clear.

A system can be constructed in a way that insolation is concentrated by reflective surfaces named collectors. This concentrated energy is used to increase the temperature of a chosen working fluid. Obtained solar thermal energy can be converted to electricity employing a heat engine, used for industrial process heat directly, or used as a hot utility in a HDN. It is a valid way to categorize concentrated thermal according to the collector design as parabolic trough collector, linear Fresnel, central receiver and parabolic dish [5].

The main classes of CST technologies are summarized in Figure 1.8 based on key characteristics.

Various types of collectors are available in the market. One way to categorize them is mobility. Earth's rotation around its axis and the sun leads to daily and seasonal changes in both the amount and angle of incidence for solar radiation. Thus, in order to harvest more, tracking should be employed. The movement can be in single or multi-axis. Another way to classify a solar collector is by characterizing the working temperature interval for the whole operation. While some collectors perform better

in medium temperature range, others may only function in low temperatures. Size of temperature range for optimal performance also differ for collectors [8].

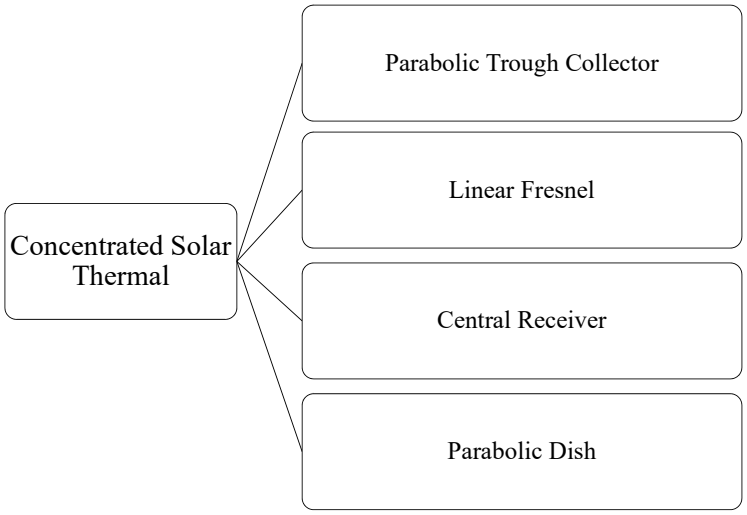


Figure 1.8: Solar Energy Classification (Adapted from Tyagi et al. [5])

As stated in the International Energy Agency (IEA) Solar Heating and Cooling (SHC) report by Horta [9]mentioned collectors are used in the industry's various branches. Textile, plastic, rubber, food, beverage, metal, and wood industries utilize the CST technology in different processes with different collectors and in different temperature ranges.

1.4 Exergy

Dealing with work derivation from any source it is not enough to focus on energetic potential only. The way the system is constructed, and intermediate steps, affects the maximum of what can be obtained from a source. Thus it is better to use energy and entropy simultaneously, namely exergy analysis. Equilibrium is a quite useful term to conceptualize this idea. Any specific equilibrium state stipulates no interaction in terms of the related affected property. For instance, force equilibrium leads to unaffected acceleration. On the other hand, radiative equilibrium requires stabilized radiated and absorbed energy. Since the ultimate system to cooperate with is the environment, it is a consistent point of view to define the capability of deriving maximum

work according to the environment.

The necessary and sufficient conditions for Thermodynamic Equilibrium are;

- Thermal Equilibrium: equality of temperature
- Mechanical Equilibrium: equality of pressure/force
- Chemical Equilibrium: equality of chemical potential

A system is at its Dead State if it is in Thermodynamic Equilibrium with the environment. Any system not at its dead state can undergo a reversible process to its dead state by only interacting with the environment. The useful work produced by this reversible process is referred to as the system's exergy, and any irreversible process with the same initial and final states will produce less useful work. Thus a system's exergy corresponds to the useful maximum work that this system can produce by only interacting with the environment. A system is at its Restricted Dead State if it is in thermal and mechanical but not chemical equilibrium with the environment. In non-reacting cases such as those treated herein where the system does not exchange any mass with the environment, the exergy can be defined with respect to the Restricted Dead State is appropriate, and this approach is adopted herein.

Exergy is an extensive property and the amount of exergy for a state (B) is defined as follows;

$$B = (U - U_0) + p_0(V - V_0) - T_0(S - S_0) + KE + PE \tag{1.8}$$

Here properties of the system at its initial state are not subscripted while properties at the Dead State are subscripted with "0". It should be noted that there is no commonly accepted symbol for exergy within the literature. Some texts use Greek letters for Exergy, and others use E_x. Exergy is denoted with B in this study with the inspiration from one of the most referenced Advanced Thermodynamics books by Bejan [10].

Equation 1.8 considers a transition to the restricted dead state irreversibly.

$$W_{useful} = (U - U_0) + p_0(V - V_0) - T_0(S - S_0) - T_0\sigma + KE + PE \tag{1.9}$$

As shown in Equation 1.9, W_{useful} decreases with increases in entropy generation. Therefore useful work is defined for the case of a reversible process as;

$$W_{useful} = (U - U_0) + p_0(V - V_0) - T_0(S - S_0) + KE + PE \tag{1.10}$$

Exergy property can also be defined intensively as specific exergy;

$$b = (u - u_0) + p_0(v - v_0) - T_0(s - s_0) + \frac{V^2}{2} + gz \tag{1.11}$$

In order to comprehend the importance of the exergy concept, it is crucial to recognize what every term corresponds to conceptually for an exergy balance for a control volume. In rate form, an exergy balance can be expressed as;

$$\frac{dB}{dt} = \underbrace{\sum_{j=1}^{\infty}\left(1 - \frac{T_0}{T_j}\right)\dot{Q}_j}_{\substack{\text{Exergy transfer}\\ \text{accompanying}\\ \text{heat transfer}}} - \underbrace{\dot{W}_{cv}}_{\substack{\text{Exergy transfer}\\ \text{accompanying}\\ \text{work}}} + \underbrace{\sum_{i=1}\dot{m}_i b_{fi}}_{\substack{\text{Inlet}\\ \text{flow exergy}}} - \underbrace{\sum_{e=1}\dot{m}_e b_{fe}}_{\substack{\text{exit}\\ \text{flow exergy}}} - \underbrace{\dot{B}_d}_{\substack{\text{Exergy}\\ \text{destruction}}} \tag{1.12}$$

where exergy of a flow is defined as;

$$b_f = h - h_0 - T_0(s - s_0) + \frac{V^2}{2} + gz \tag{1.13}$$

Exergy transfer accompanying heat transfer is product of heat transfer rate and Carnot factor (η_c), which is defined as;

$$\eta_c = \left(1 - \frac{T_0}{T_j}\right) \tag{1.14}$$

For all the Carnot factor calculations temperatures are absolute (in Kelvins).

1.4.1 Exergy in HEX

Before giving any information about exergetic properties in a heat exchanger (HEX), it should be noted that details will depend on the kind of HEXs. Additionally, categorization is done considering various factors. The nature of heat transfer divides HEXs as indirect and direct contact HEX. The layout of the flow as single or multi-pass is another way to classify HEXs. Stream movement directions such as parallel, cross, and parallel flow define the type of the HEX too. For the pinch analysis discussion, streams will be matched with counterflow arrangement. Although the number of fluids, compactness of surface and employed mechanism of heat transfer are alternatively used attributes in addition to the ones mentioned above, the essential way of categorization is built features. Tubular HEX, plate type HEX, extended surface HEX, and regenerators are the main ones with lots of sub-classes for every one of them individually [11]. Nevertheless, all of these factors and their effects on exergetic performance will be neglected. The majority of the discussion will be based on a simple model, as shown in Figure 1.9 and Figure 1.10.

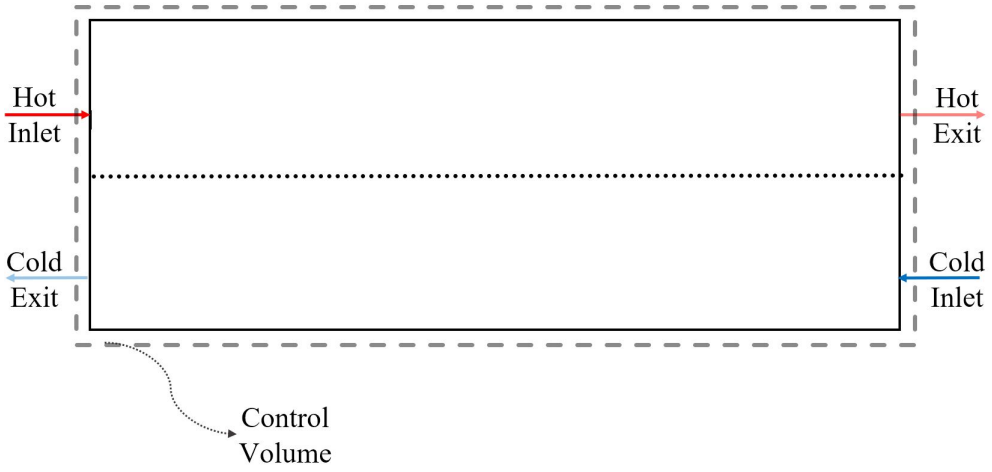


Figure 1.9: Simple representation of HEXs

For steady-state conditions, left-hand side of Equation 1.12 is zero. Exergy transfer accompanying work is nonexistent for a HEX. It is desired to exchange heat only between the streams, so any heat outwards from the control volume boundary with a temperature T_s is an exergy loss. Besides, some portion of the exergy is destroyed due to the structure of the equipment. Solving for B_D Equation 1.12 takes the following

form;

$$\dot{B}_d = \left(1 - \frac{T_0}{T_j}\right) Q_j + \dot{m}_i b_{fi,h} + \dot{m}_i b_{fi,c} - \dot{m}_i b_{fe,h} - \dot{m}_i b_{fe,c} \tag{1.15}$$

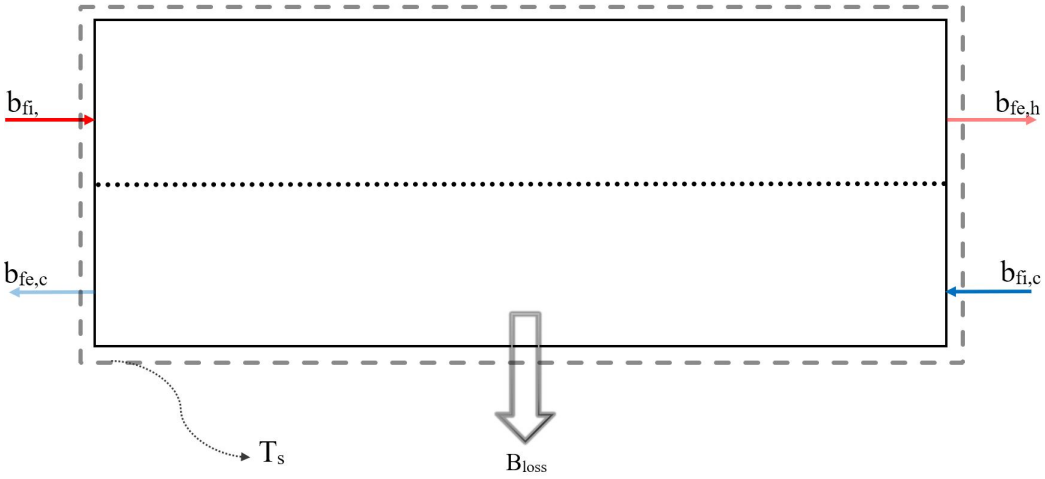


Figure 1.10: Exergy Flows for the HEX

The importance of this exergy destruction solution stands on inverse proportionality between exergy destruction and performance. Specifically, minimization of exergy destruction is a way to obtain better exergetic performance.

1.4.2 Exergy for CST

Study of solar thermal exergy calls for main concepts such as exergy of solar radiation and exergetic performance of solar collectors. While examining the journey of solar radiation, in order to find energy and entropy values for the eventual state, different models are used. According to Coulson, models of the atmosphere, solid angle, polarization, conversion device, and entropy should be applied to extraterrestrial solar radiation input [12]. Above mentioned conversion device can be a CST system. Consistent with the scope of this research as defined in (subsection 1.5.2), the exergy associated with the flow of heat transfer fluid in the collector is assumed to only be a function of temperature, specific heat, and mass flow rate.

All of the available collectors serve the primary purpose of transferring the maximum of what is received from the sun to the working fluid. A specific portion of thermal energy Q_c received from a collector with a temperature of T_c is used to heat a working fluid. A simple energy balance for collector states that for steady-state conditions, the temperature of the collector should not change with time. Because any increase would mean the incidence of radiation is used to rise the thermal energy of the collector, which is meaningless. With the purpose of a deeper understanding of this topic, non-dimensional collector temperature is introduced as;

$$\theta = \frac{T_c}{T_0} \tag{1.16}$$

Where T_c is the collector temperature and T_0 is the temperature of ambient. The collector reaches its maximum temperature, named stagnation temperature θ_{max}, when the rate of solar gain equals the rate of losses to the surroundings, and therefore the collector inlet and outlet temperatures are equal.

Referring to exergy associated with heat transfer in Equation 1.12, the exergy input from sun considering a solar radiation Q_{solar} and temperature T_s may be expressed as;

$$B_{in} = Q_{solar} \cdot \left(1 - \frac{T_0}{T_s}\right) \tag{1.17}$$

Q_{solar} is the total of diffuse and beam radiation on the collector. This is the simplest model for the exergy of solar radiation and is useful conceptually. A more sophisticated model is presented in section 2.4, Equation 2.7 and used in the design tool PCSTET developed herein.

Note the sun's apparent apparent blackbody temperature is $6000K$, and according to Petela [13], T_s value is 75% of blackbody temperature for sun.

$$T_s = 6000K \cdot 0.75 = 4500K \tag{1.18}$$

The exergy delivered to working fluid at T_c through heat transfer Q_c is defined as: is

defined as ;

$$B_{out} = Q_c \cdot \left(1 - \frac{T_0}{T_c}\right) \tag{1.19}$$

The difference between B_{in} and B_{out} reveals the exergy destruction in the collector. All of these definitions, concepts, and equations are introductory. The effort in this research is focused on building a tool that automatically creates HDN and linking it to exergy and CST.

1.5 Thesis Overview

In terms of organization of the study the following subsections provide insight into the flow of thought and details.

1.5.1 Objective

As it is the case for other engineering studies, the purpose of this study is revealing the method and possibilities to increase performance. Before the explanation of the specifics, the choice of topic by addressing the requirement for research should be given. One of the most important strategies employed during research topic selection is pivoting. When one meets a dead-end or unneeded subject matter one when investigating a research topic, should pivot and look for alternatives. However, pivoting also requires a starting point. This is INSHIP [14]. This EU project is a follow up from the International Energy Agency (IEA)'s Solar Heating and Cooling Programme, task 49[15]. Due to the partnership of Middle East Technical University (METU), lots of researchers were introduced to the INSHIP project. As the name suggested, the main objective is to come up with better solar heat solutions for industry. In order to function better, this main objective is divided between working packages, tasks, and sub-tasks. METU participates in several of these tasks including; SHIP for drying, solar metal production for metallurgical industry, solar lime production for cement industry, emerging process technologies and lastly industrial parks with heat distribution networks. Particularly efficiency strategies for resources and classification of industrial park supply structures sets the origin for this research.

Having an EU funded project with the requirement of research for a chosen topic is a powerful motivation and quite a good introduction.

Initial research on HDNs in industrial parks showed the implementation of pinch concept. Details showed that the usage of pinch concept and exergy with solar thermal applications is relatively contemporary.

There are lots of commercial tools to use in pinch analysis[16],[17],[18]. Environments such as Visual Basic and PHP are used. While some of them calculate pinch temperature and the required amount of external means, others show how to construct the HDN. The Objective of this work is to develop the first version of an open-source software to aide in the design of HDNs with CST integration based on pinch and exergy analyses. The target audience is expert HDN designers who are able to synthesize the outputs from this software with complementary inputs (e.g. economic) to design high-performing and practical HDNs that include CST.

While aiming for an improvement in the performance, relaxation is applied to some inputs. Then change in exergy destruction is evaluated. From all of the alternatives obtained by the variation of inputs, the set with the optimum performance is chosen. Exergy destruction value and utility requirements are calculated. Representative plots and HDN for original and optimum input sets are created.

1.5.2 Scope

Consistent with the objective to develop the first version of an open-source software, the scope of this work is intentionally kept narrow and focused on developing methods for relatively simple conditions that can be extended to more complex conditions in the future as follows:

- At most ten streams can be applied to this model.
- Phase change should not occur for any stream.
- At least one hot and one cold utility should be required in order for streams to reach their target temperature.

- Only hot utilities are taken into account when exergy destruction is calculated for CST. Note that some portion of these utilities are modeled as collectors with specified heat input calculated by inputs that are explained in section 3.1. Remaining hot and entire cold utilities are considered to be electrical heaters and coolers. Exergy discussion for them is left out of scope.
- The discussion does not consider an existing HDN. It just proposes a design for a new HDN.
- It is not possible to model every set of streams with the software designed here. For a specific input collection, if the solution for the HDN may require more external means than it is calculated in the pinch analysis, this input collection is omitted in the parametric study.

1.5.3 Organization

In this chapter, chapter 1, a brief introduction to pre-required concepts are given. Pinch analysis, pinch design, exergy, and CST are explained. Following with chapter 2 methodology will be explained with an example for pinch analysis and design, details of Exergy-Pinch Analysis, CST implementation, and parametric study. After, chapter 3 provides every detail of the design tool in terms of software development with explanations and flowcharts of the functions. Subsequently, the design tool is benchmarked with values from the literature. Evaluation of a system is presented with suggested design parameter variation for performance enhancement in chapter 4. Then finally, in chapter 5 comments and discussion about the overall study are presented.

CHAPTER 2

METHODOLOGY

The application of previously mentioned concepts requires a detailed and precisely defined methodology. In order to provide a better understanding, this chapter starts with an example of pinch analysis and pinch design for a HDN. After, details for combined pinch and exergy analysis are provided. The body continues with CST and finishes with the details of the parametric study conducted to find an optimum solution.

2.1 Demonstration of Pinch Analysis and Design

Four streams, as two hot and two cold, frequently used in literature ([1]), are chosen to summarize pinch analysis. The highest and lowest temperatures and heat capacity rate (CP) values for these streams are given in Table 2.1. ΔT_{min} is specified as $10°C$.

Table 2.1: Temperature intervals and heat capacities for given problem [1]

	Original		Shifted		
Stream Number *(Type)*	T_{low} *(°C)*	T_{high} *(°C)*	T_{low} *(°C)*	T_{high} *(°C)*	CP *(kW/K)*
1 *(Hot)*	60	170	55	165	3
2 *(Cold)*	20	135	25	140	2
3 *(Hot)*	30	150	25	145	1.5
4 *(Cold)*	80	140	85	145	4

First and third streams are the hot streams. They are, in fact, streams to be cooled. Similarly, second and fourth streams are cold streams, which means they are streams to be heated. Table 2.2 called problem table is provided below and constructed as follows.

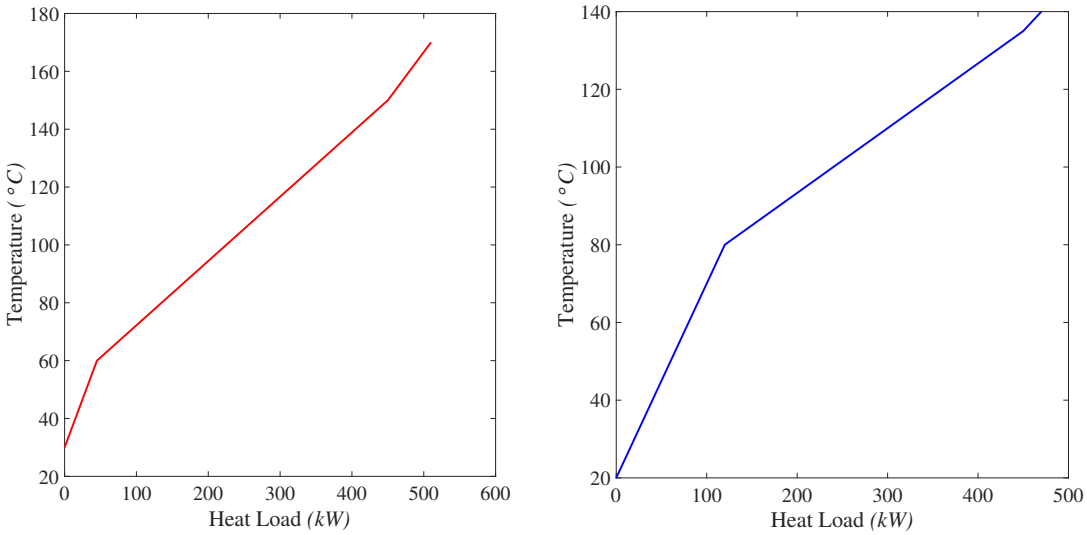
Table 2.2: Problem Table [1]

Temperatures *(Shifted)*	ΔT *(°C)*	Existing Streams	$\sum_{hot} CP - \sum_{cold} CP$ *(kW/K)*	ΔH *(kW)*	H *(kW)*	H' *(kW)*
165					0	20
	20	1	3	$+60$		
145					60	80
	5	1,3,4	0.5	$+2.5$		
140					62.5	82.5
	55	1,2,3,4	-1.5	-82.5		
85					**-20**	**0**
	30	1,2,3	2.5	$+75$		
55					55	75
	30	2,3	-0.5	-15		
25					40	60

Temperatures for hot streams are decreased $\Delta T_{min}/2 = 5°C$ while temperatures for cold streams are increased at the same amount. Shifted values that are shown in the Table 2.1 are listed in descending order in the first column. Temperature intervals (ΔT) are stated with the existent streams on them. Then the difference between hot and cold streams' CP values for every interval is calculated and presented in column four. Heat loads that are given in column five (ΔH) are calculated by the product of total CP values and ΔT values. The highest temperature value starts with zero heat load, and by adding ΔH values, the load for every temperature is calculated and presented in the sixth column (H). The smallest heat load is identified as $-20kW$.

The expectation is to have no load at the pinch point. Thus, it is only logical to add $20kW$ to every value in the sixth column and obtain seventh column (H'). The zero value will locate the T_{pinch} as $85°C$. At the low end of the temperature continuum, cold utility requirement, H_{cu}, is identified as $60kW$. At the high end of temperature interval, hot utility requirement, H_{hu}, is identified as $20kW$.

Previously explained composite curves constructed at the end of the well-defined procedure. Firstly, Temperature and heat load graphs are plotted separately for the hot and cold streams as shown in the Figure 2.1a and Figure 2.1b.

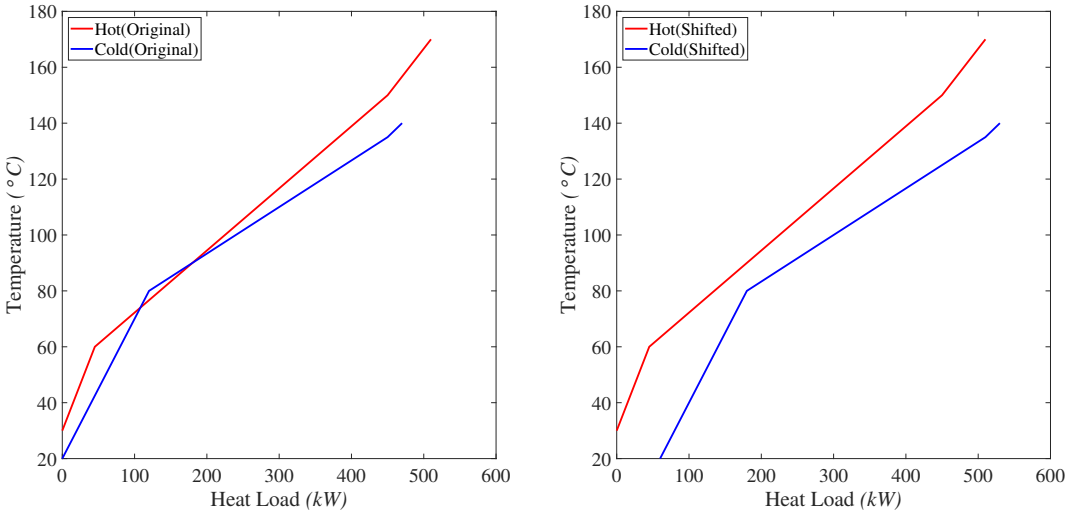


(a) T-H diagram for hot streams

(b) T-H diagram for cold streams

Figure 2.1: T-H diagrams separately

Then, they are represented in the same axes and they initially intersect as shown in Figure 2.2a. For the parts where the cold composite curve is located higher than the hot one, heat transfer from the hot to the cold stream requires a work input. However, shifting the T-H curve for cold streams as much as cold utility requirement, $60kW$, the final version of composite curves are obtained where the temperature difference between the hot and cold streams is ΔT_{min} at the Pinch Point, and greater than ΔT_{min} at all other points as shown in Figure 2.2b.



(a) Unshifted T-H Diagram

(b) Composite Curves

Figure 2.2: Shifted T-H Diagram Resulting in Composite Curves

The plot of first (Shifted Temperatures) and last (H') columns of Table 2.2 gives GCC for this problem as Figure 2.3.

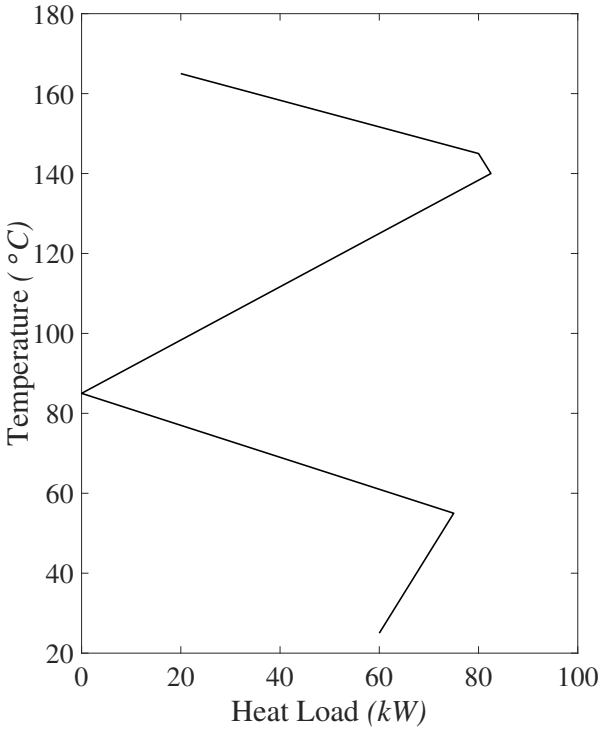


Figure 2.3: Grand Composite Curve of streams from Table 2.1

Notice that line touches the abscissa at $85°C$, which also equals to pinch temperature. Additionally, the area between the curve and the abscissa is equal to the area between hot and cold composite curves.

All of the information required to build HDN is obtained with pinch design. One should note that the general procedure will be explained here. Next chapter explains the actual procedure employed by the designed tool.

At the beginning pinch design divides the problem into as above and below the pinch(Table 2.3). For below pinch upper limits are updated as $T_{pinch} + \Delta T_{min}/2$ for hot streams which is $90°$. For the colds it is $80°$ which corresponds to $T_{pinch} - \Delta T_{min}/2$. The lower limits below pinch remain as the low temperatures of every stream. The lower limits above pinch are equal to upper limits for below pinch. These are $90°$ for hot ones and $80°$ for cold ones. Upper limits above the pinch are equal to the high temperature of every stream.

Table 2.3: Streams above and below the pinch temperature

	Below T_{pinch}		Above T_{pinch}			
Stream Number	T_{low} (°C)	T_{high} (°C)	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	c_p
1(hot)	60	90	90	170	1.000	3
2(hot)	30	90	90	150	1.000	1.5
3(cold)	20	80	80	135	1.000	2
4(cold)	-	-	80	140	1.000	4

By considering the ΔT_{min} and CP criteria, the stream matches are completed, which means HEXs are created. These HEXs are presented in Table 2.4 and Table 2.5. After all possible matches are established, remaining streams shown in Table 2.6 and Table 2.7 reach their target temperatures with the use of utilities.

Table 2.4: Heat Exchanger Information Above Pinch, manual solution

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	CP	ΔH (kW)
1	3	80	140	4	-240
	2	90	170	3	240
2	1	80	125	2	-90
	4	90	150	1.5	90

Table 2.5: Heat Exchanger Information Below Pinch, manual solution

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	CP	ΔH (kW)
1	1	35	80	2	-90
	2	60	90	3	90
2	1	20	35	2	-30
	4	70	90	1.5	30

Table 2.6: Streams to be cooled in cold utilities,manual solution

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	CP	ΔH (kW)
1	1	125	135	2	-20

Table 2.7: Streams to be heated in hot utilities,manual solution

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	CP	ΔH (kW)
1	4	30	70	1.5	60

The network is presented in Figure 2.4. If there is a line connecting two streams with circles on both on them, this means a HEX is built there. The HEX number is shown in the top circle and the absolute heat load is shown in the bottom circle.

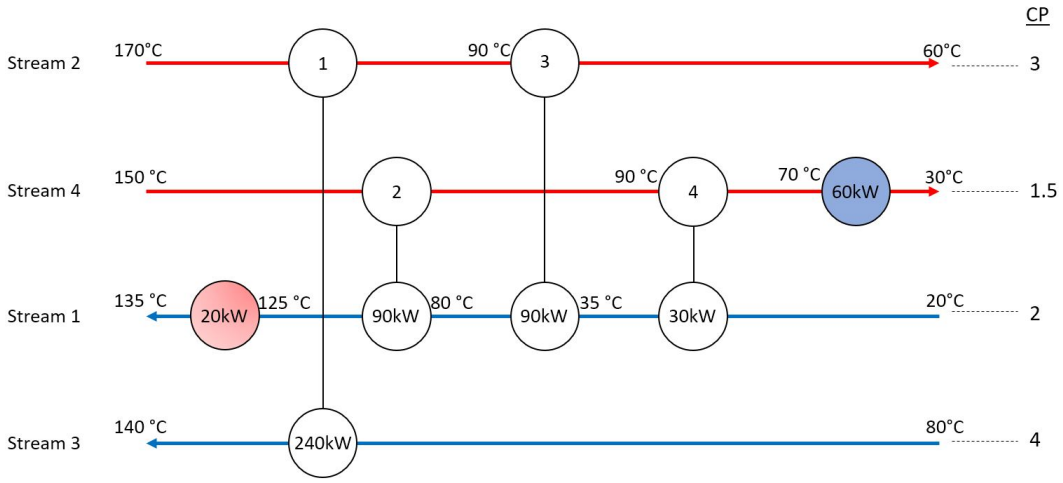


Figure 2.4: Network grid diagram of streams from Table 2.1

It is important to notice that, above pinch, the CP values are higher for cold streams than the hot streams with the matches that are established. Although below pinch HEX 3 also satisfies this condition, HEX 4 as it is explained in the subsection 1.2.2, satisfies ΔT_{min} with CP due to large temperature difference.

Obviously, this is one of the most simple problems. Actually, the necessity of software arises for more complex problems, which results in the requirement of split and division of streams. This example will be revisited in chapter 4and the methodology for more complicated problems will become clearer.

2.2 Combined Pinch - Exergy Method

All of the concepts explained as introductory material actually can be used as mechanisms. The current study will combine pinch and exergy concepts for industrial parks and come up with a heat distribution network design. The key point of heat recovery systems is utilizing most from what is available and is not used. Since the theory of exergy also is established on the concept of availability, combining those two makes sense naturally. Merging these two methods is common in literature. The reasoning is stated by Linnhoff et al. as follows. Pinch analysis provides limited accounting for heat loads and identifies the amount of requirement as well as the waste. On the other hand, the exergy analysis identifies wasteful procedures. Thus usage of them together compensates the negative aspects of each [1]. However, it should not be overlooked that combined pinch and exergy analysis does not mean all of the potential enhancement stated is possible. For instance, for arbitrarily chosen ΔT_{min} may not be feasible for available heat exchangers. In addition exergy loss,B_{loss}, consists of avoidable, AVO, and inevitable, INE, portions[19].

$$B_{loss} = AVO + INE \tag{2.1}$$

Similarly, unavoidable exergy destruction might be due to financial and technological limitations.

For complete comprehension of the procedure, the following sections are quite beneficial. However, before those, there is an important consideration to be taken into account. T_{pinch} is fixed by a defined ΔT_{min}. However, for a fixed ΔT_{min} and the rate of heat transfer, the rate of exergy destruction increases with the ratio of hot to cold temperatures. Therefore to minimize exergy destruction, ΔT_{min} should not be fixed but rather increase with temperature. Investments are made to have small ΔT_{min} for low-temperature heat exchanges while allowing for larger ΔT_{min} for high-temperature heat exchangers. However, as a first step in the present study the objective is to decrease exergy destruction with a fixed ΔT_{min}, and the problem of extending this work to allow for variable ΔT_{min} is left to future researchers.

2.3 Exergy Composite Curves

Foundation of the Exergy Composite Curve (ECC) and Grand Composite Exergy Curve (GECC) lies in Equation 1.12. Initially, both were just constructed for HDN's based on exergy accompanying heat transfer term in Equation 1.12. The abscissa shows heat load while ordinate shows the Carnot factor (η_c). For the calculation presented here for the η_c, T_0 is assumed to be $20°$.

In reality, every group of expressions in Equation 1.12 can be implemented to exergy composite curves. In such a case, the Carnot factor will not be sufficient. As a substitute term named energy level, Ω, is defined[19];

$$\Omega = \frac{ExergyTransfer}{EnergyTransfer} \tag{2.2}$$

Therefore, for work, it equals unity.

$$\Omega_w = 1 \tag{2.3}$$

For a flow at steady state;

$$\Omega_f = \frac{\Delta B}{\Delta H} \tag{2.4}$$

However for the current scope, only the exergy related to heat transfer will be taken into account for the construction of the exergy composite curves. So the energy level is equal to Carnot Factor.

$$\Omega_f = \eta_c \tag{2.5}$$

Two figures updated from Figure 2.2b and Figure 2.3 are constructed with same abscissa and ordinate of η_c. ECC and GECC from 4-stream example (Table 2.1) are provided in Figure 2.5 and Figure 2.6.

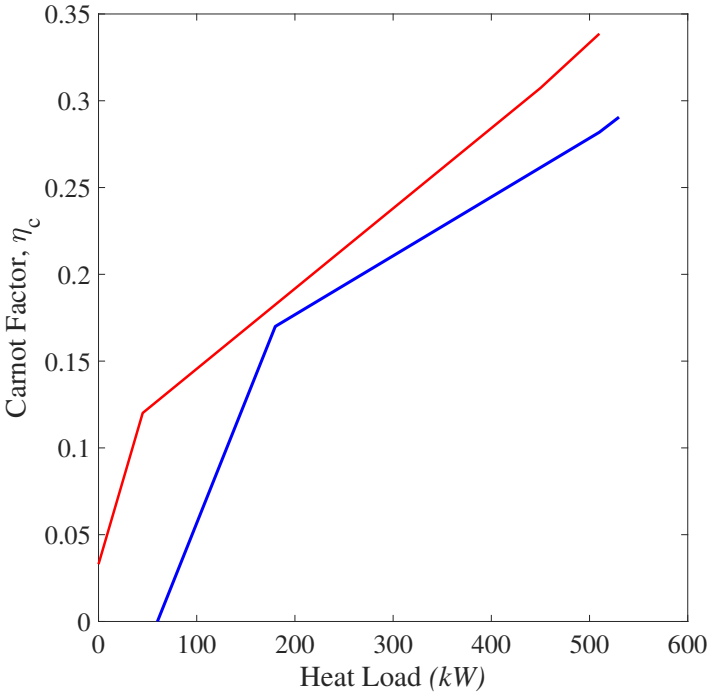


Figure 2.5: Exergy Composite Curve

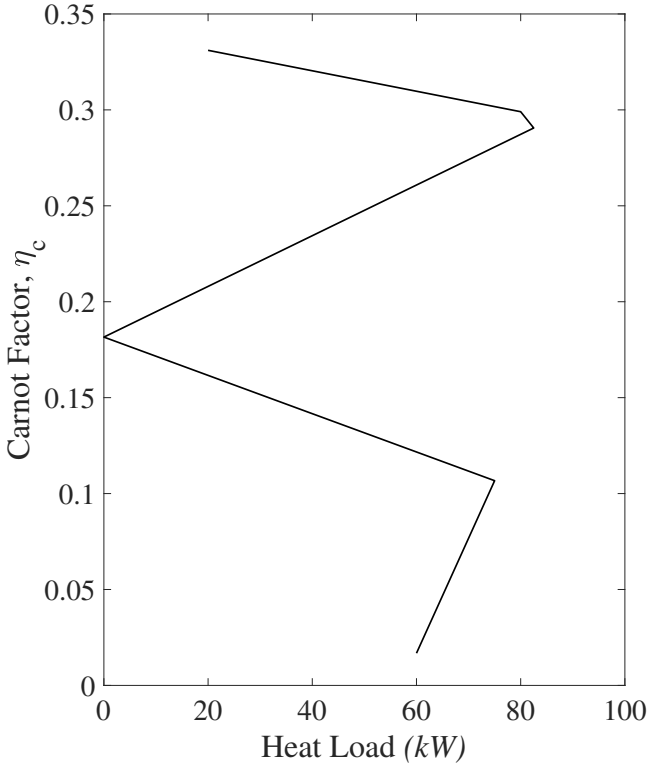


Figure 2.6: Grand Exergy Composite Curve

The area between cold and hot composite curves in ECC expresses the exergy destroyed through the build of HDN.

In literature, the majority of the ECC's are represented in a way that lowest and highest loads for hot and cold composite curves are the same. Hence inference of area between the hot and cold composite curves is relatively easy. However, often for the requirements for heating and cooling utilities, start and endpoints do not coincide horizontally. For the solution, ECC may be divided into three sections as in Figure 2.7. These are from zero heat load to the amount of cooling load (I), from cooling load to highest load for the hot composite curve (II) and finally from here to end of the cold composite curve (III). The last two sections will be examined in terms of exergy in the scope of this study. For the middle one exergy destruction will be expressed with the area between curves as stated previously. However, for the last one, exergy destruction will be calculated considering streams to be heated with CST and a considerable amount of exergy is destroyed and lost here. Details are provided in the following section.

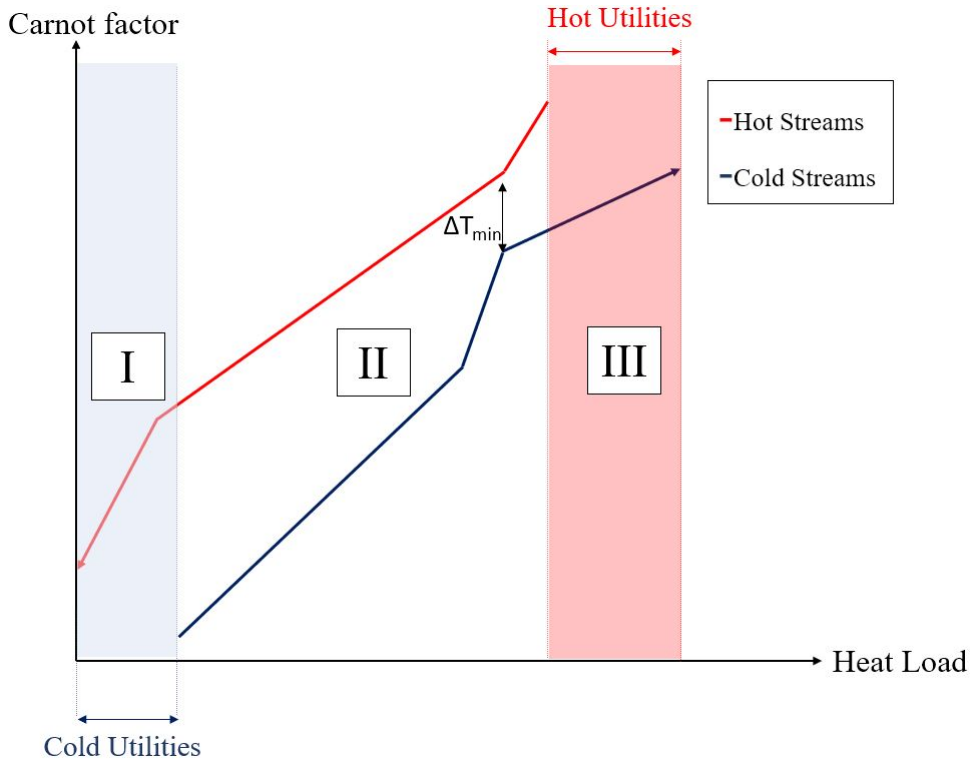


Figure 2.7: Three sections of Exergy Composite Curve

It is easily noticeable that GECC carries almost the same information with GCC. The usage of GECC is justified in the same way the usage of exergy is justified. Pinch analysis is a study of finding the best possible scenario to utilize most from what is available. In this sense, there is analogousness between pinch and exergy concepts. So using GECC is more consistent in a sense.

2.4 CST Implementation

Basic information about exergetic characteristics of solar thermal is presented in section 1.3. Furthermore, the values to be calculated in the design tool will be explained in the current section. The exergy balance of the system is stated in Equation 2.6.

$$B_s = B_u + B_{loss} + B_d \tag{2.6}$$

B_s expresses the magnitude of concentrated radiation exergy value, while B_u defines the amount of exergy that is utilized by the CST. So, it is desired to have B_s and B_u values as close as possible.

The simplest model for the exergy of solar energy is given in the literature demonstrated as Equation 1.17 and Equation 1.19. Relatively more sophisticated models are defined by various researchers. One of the most widely used models is suggested by Petela and can be calculated as [20];

$$B_s = Q_s \left[1 - \frac{4}{3} \cdot \left(\frac{T_0}{T_{sun}} \right) + \frac{1}{3} \cdot \left(\frac{T_0}{T_{sun}} \right)^4 \right] \tag{2.7}$$

Where T_{sun} is approximated as temperature of the sun, Q_s is available solar energy and approximated as by the product of assumed equivalent Direct Normal Insolation (DNI) and collector surface area A_{col};

$$Q_s = DNI \cdot A_{col} \tag{2.8}$$

In reality, this Q_s value is a theoretical maximum. In order to obtain a more accurate

estimate, one should account for losses due to optical efficiency and thermal losses. Factors such as reflectivity, absorptivity, transmissivity, incidence angle, geometry, mirror clearness, twisting errors, and receiver clearness affect optical efficiency. In addition, heat losses occur due to convection and radiation. Properly accounting for optical and thermal losses requires detailed information about the collector technology and meteorological conditions, which are outside the scope of the current work and therefore these losses are neglected.

According to Bellos et al. with the assumption of negligible pressure drop for specifical liquids, the amount of exergy that is utilized by CST, B_u, is expressed as[21];

$$B_u = Q_u - CP \cdot T_0 \cdot ln\left[\frac{T_0}{T_{sun}}\right] \tag{2.9}$$

Used thermal energy Q_u, is equal to thermal energy increase in the related stream, which is working fluid for the CST system and previously defined with Equation 1.2. Ideal but impossible case of having the same solar exergy supply and exergy usage requires zero exergy loss and destruction. Rearranging and substituting;

$$B_{loss} + B_d = B_s - B_u \tag{2.10}$$

$$B_{loss} + B_d = \left(DNI \cdot A_{col}\left[1 - \frac{4}{3} \cdot \left(\frac{T_0}{T_{sun}}\right) + \frac{1}{3} \cdot \left(\frac{T_0}{T_{sun}}\right)^4\right]\right) - Q_u - CP\left(\Delta T - \cdot T_0 \cdot ln\left[\frac{T_0}{T_{sun}}\right]\right) \tag{2.11}$$

So, the arrangement with the lowest value for Equation 2.11 will have the best exergetic performance. This model considers fixed DNI and A_{col} determines the potential for hot utilities. By the use of Equation 2.11, exergy destruction in hot utilities is calculated. Methodology related to exergy is disclosed in terms of equations. How those are applied to the tool that is designed will be detailed in chapter 3.

2.5 Parametric Study

Improvement in exergetic performance requires a change in initial and target temperatures or CP values [22]. Streams are classified as alterable or unalterable, where the temperatures and CP of alterable streams can be altered while that for unalterable streams cannot be altered. All alterations are applied to a set of initial inputs. A new stack of inputs is entered into the model, and all of the procedure is repeated. This application is not limited to the initial pinch analysis and exergy calculations. Because a variation in inputs also generates a different HDN. After, the input set with minimum exergy destruction and perfect obedience to restrictions is selected.

The tool is constructed in a way that, temperature inputs are categorized as high and low rather than initial and target. So, for newly created input sets, temperatures are increased and decreased. The model uses mass flow rate and specific heat values to calculate CP. So changing either of them will have the same effect. However herein only mass flow rate variations will be considered, as a change of specific heat requires a change of working fluid type and often this is not possible.

For the high and low temperature inputs, 10% of the difference between them added and subtracted to both, and two more alternatives in addition to the original are created. Also, as previously stated, it would be quite problematic to change specific heats since it will require a change of working fluid. So in order to alter CP, the flow rate is changed only. The three possibilities are the original value, 80%, and 120% of the original values. In the end, with the combinations of these three variations, nine alternatives for each stream are recorded. All of the alternatives created for 4 stream example mentioned before are presented in Appendix B. Revisiting this section after chapter 4 is studied may be helpful..

CHAPTER 3

DESIGN TOOL: PCSTET

The methodology explained in the previous chapter is not very hard to implement for the 4-stream example mentioned in chapter 1. Still, an increase in the number of streams can make the problem hard to deal with. In order to provide faster, easier, and more robust solution, the software PCSTET is developed in MATLAB environment. Although every language is different and has its advantages and disadvantages, MATLAB is chosen due to personal competences. Nevertheless, the approach may be easily projected on other programming languages. The majority of the programming is completed along the research process, and some minor parts are found from the file exchange source of Mathworks[23]and will be cited. All of the major parts are explained in this chapter so that with minor modifications, similar tools can be designed in another environment.

In fact, software solutions just for pinch analysis are available. But their source codes are not public. Both in order to contribute to the open literature and be able to build on pinch analysis with CST and exergy, the tool is designed from the beginning. A comparison of results and usage with existing tools will be presented.

The design tool includes visual representations for the user as well as plots for the presentation of the subject matter. Through this chapter explanation of the core parts are provided with flow charts.

Object-based programming is used for the coding for this specific problem. The all of the functions created for different purposes require multiple inputs and outputs. Some of the variables even appear as both input and output. So creating an object and related properties saves the programmer from tedious work.

Flow charts for tool and each individual function are presented in the following sections. Additional information and details are presented in the body. Before the explanation of functions, representative flowcharts are given as Figure 3.1 and Figure 3.2. For the entire set of flow charts;

- Function names are presented as the names in the source code (section A.1). These names are provided at the beginning of each section. Since these names are complicated, functions are referred with the names presented in titles of each section.

- The flowcharts which occupy more space than a single page are divided into parts. For only these divided flowcharts, at the end of each part a link to the next part is created.

- If any more connection is required for different parts, lines are formatted and color coded accordingly.

- Same kind of representations are used for similar parts. For instance;

  - Blue rounded rectangle for the initiation of a function
  - Red rounded rectangle for the termination of a function
  - Green parallelogram for the inputs
  - Red parallelogram for the outputs
  - Yellow rectangle for operations
  - White rectangle for manipulations and pauses
  - Rounded white rectangle for the loop index
  - Rotated pink rectangle for logicals
  - Grey circles for connection of different parts of flowchart

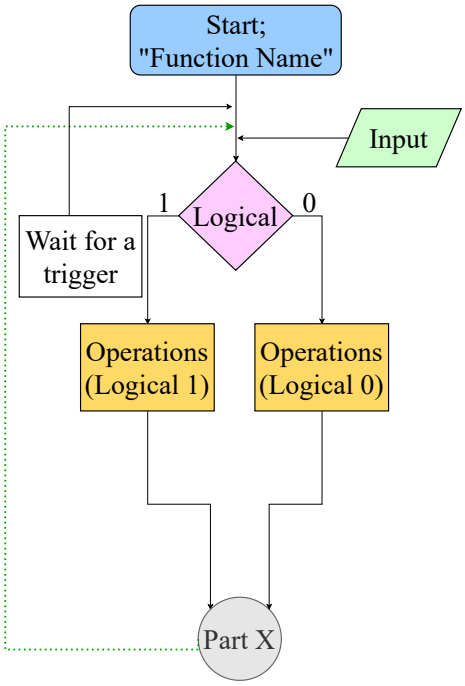


Figure 3.1: Representative Flowchart, Part 1

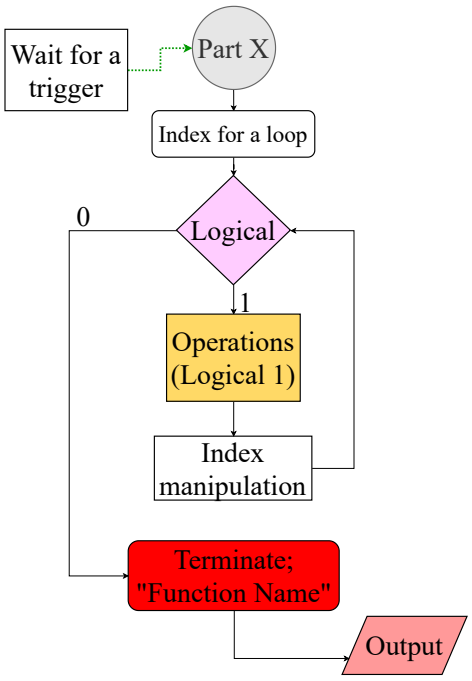


Figure 3.2: Representative Flowchart, Part 2

3.1 PCSTET Main Function

PCSTET is an abbreviation for Pinch-Concentrated Solar Thermal-Exergy Tool (PCSTET), and the following tool is referred with this abbreviation. In order to run the code, "PCSTET Main" Function (exergy_pinch_HDN) is called from the command window or run button on the editor is used.
Through the steps, many figures will be presented on the screen. Multiple GUI's are designed in a form that allows altering inputs and to re-run related functions. So, initially, all of the available figures are closed.
A GUI called "Pinch Analysis" is opened. This window includes axes for;

- CC
- GCC
- ECC
- GECC

Tables for the hot and cold streams' following properties;

- Inlet temperature
- Outlet temperature
- Mass flow rate
- Heat of change

"Heat of change" term is introduced to account for the effect of phase change. Part of the tool is Scott Rowe's Pinch tool which is available in Mathworks [24]. This tool is only capable of drawing composite curves, and phase change are valid up to this extent. However, PCSTET is not able to deal with phase change. Nevertheless, "Heat of change" is not removed or altered. In this context heat of change corresponds to specific heat c_p where $CP = \dot{m} \cdot c_p$. In addition, another table for collector surface area, ambient temperature, minimum temperature difference, and solar resources is

available to enter data. All of the information is gathered by pushing the "Read" button. This makes the "Pinch" button appear.

A click on the "Pinch" button, performs pinch analysis with provided inputs. T_{pinch} is calculated. Hot utility and cold utility amounts are revealed. CC and GCC appear on related axes. Then the "HEX" button appears next to the "Pinch" button. As soon as the user hits this button, the first procedure is breaking streams as above and below the pinch. Breakpoints are $T_{pinch} + \Delta T_{min}/2$ for hot and $T_{pinch} - \Delta T_{min}/2$ for cold streams.

Using a function to build HEX requires the employment of two other functions for splitting and dividing the streams. As a result, HEX information is stored as inlet and outlet temperatures, mass flow rate, specific heat, stream number, and heat load values for hot and cold streams entering the heat exchanger like they are shown in Table 4.2 or Table 4.3.

Utility information is also stored in a similar manner (Table 4.4, Table 4.5); the only difference is the fact that only one stream (hot or cold) information exists for a utility. Heat load value in utility array represents the power input required. Subsequently "Exergy" button is activated. It triggers the calculation of exergy destruction for HDN (recovery), utilities CST. ECC and GECC are drawn in related axes in "Pinch Analysis" GUI.

After that "Network" button is activated right next to the "Exergy" button. Two kinds of network diagrams are provided to the user. The first one appears with one click to "Network" button and draws streams entered by the user with a clear appearance of pinch temperature. This GUI includes three boxes. If any heat exchanger other than a utility exists, "Show HEXs Above/Below Pinch" buttons appear in associated sides. A press on any of these two buttons will show new network diagrams. These are above or below pinch according to the button that is used. Still, if no heat exchanger is available, a disabled text box is shown as "No HEX available above/below pinch.". The majority of the network diagrams in the literature have one weakness in common. The presentation method developed for PCSTET reduces this weakness. The structure of two-dimensional plots suggests that a straight line perpendicular to any axis would represent the same value of the quantity shown along this axis. For Net-

work diagram ordinate describes stream number while abscissa shows temperature values. Vertical connectors represent heat exchangers even though a line perpendicular to abscissa should define constant temperature. It is impossible to transfer heat if two streams have the same temperature. So rather than seeing inlet and outlet temperatures at the boundaries of heat exchangers and misrepresentation of connection, heat exchanger lines started from the average of high and low temperature for related streams and connected with a stepped line. Mass flow rates and heat load values will be presented in a spreadsheet.

A press on "Optimize" button initiates the loop that includes application of Pinch, and HEX functions yields to number of utilities and HEXs below and above pinch, inlet and outlet temperatures, mass flow rates, specific heats for each stream in every HEX and every stream in all of the utilities, exergy destruction for heat recovery and CST utility systems,CC, GCC, ECC and EGCC for optimized system.
The flowcharts of PCSTET main function are presented in Figure 3.3, Figure 3.4, Figure 3.5 and Figure 3.6.

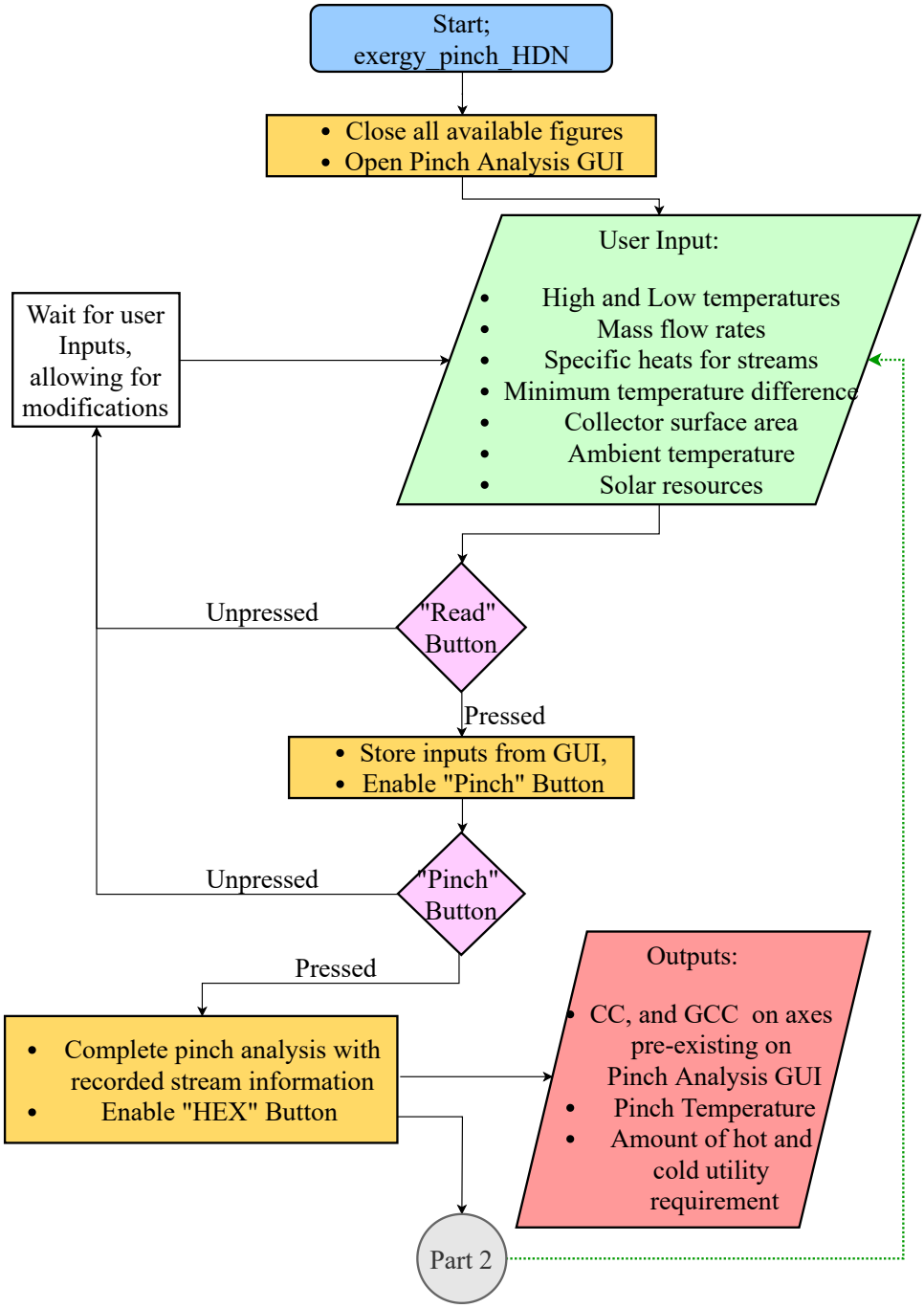


Figure 3.3: Flowchart of PCSTET Main Function, Part 1

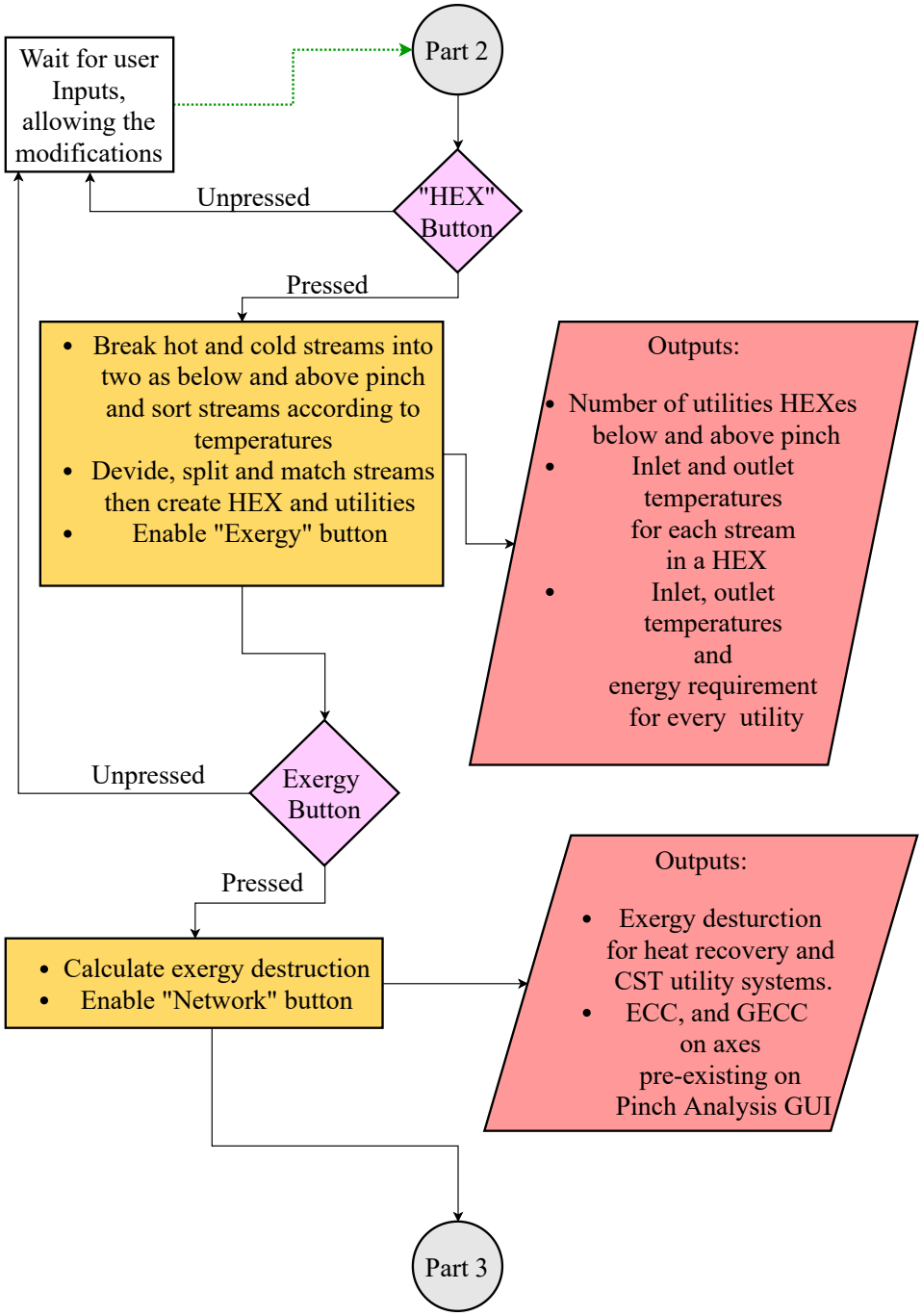


Figure 3.4: Flowchart of PCSTET Main Function, Part 2

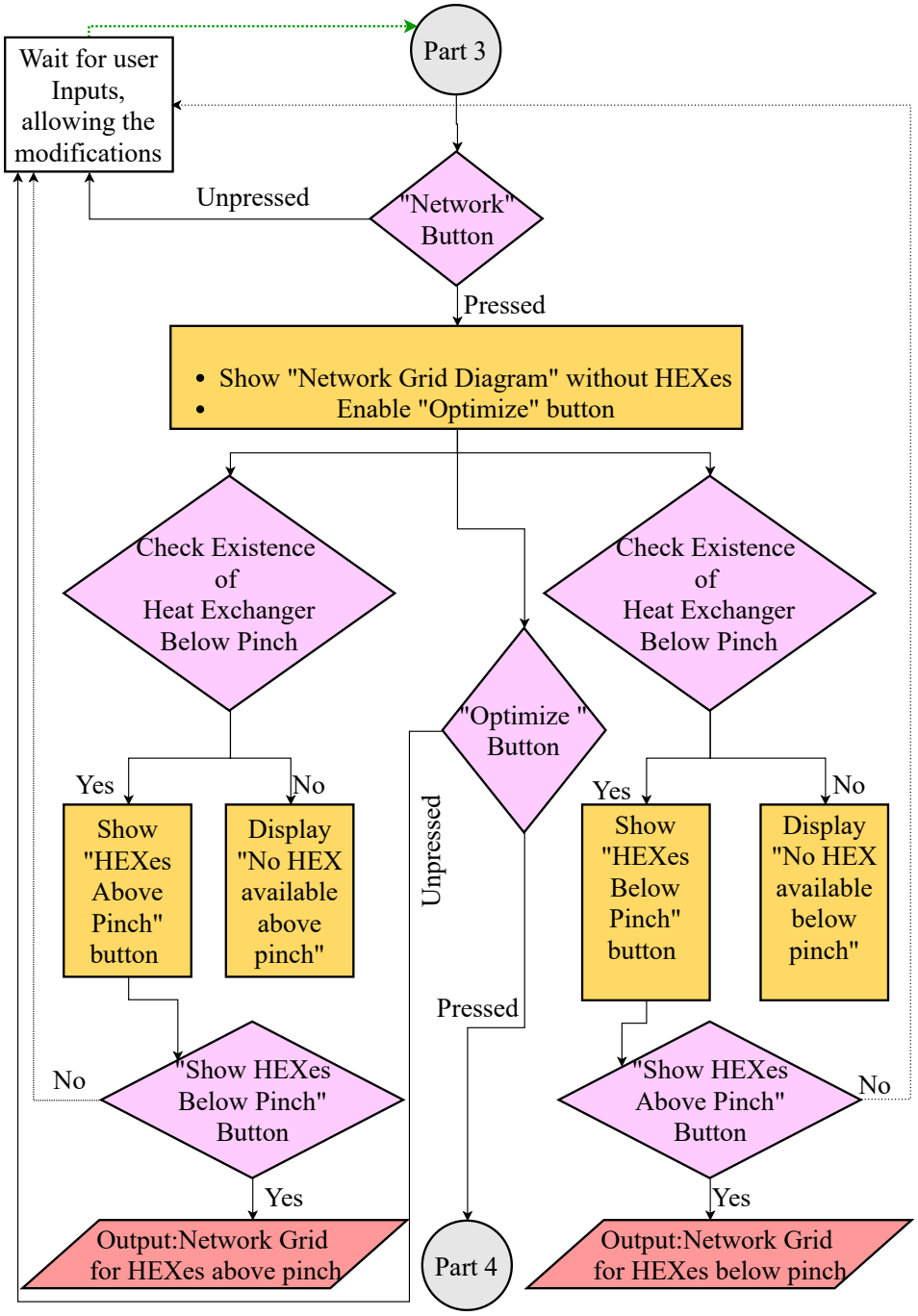


Figure 3.5: Flowchart of PCSTET Main Function, Part 3

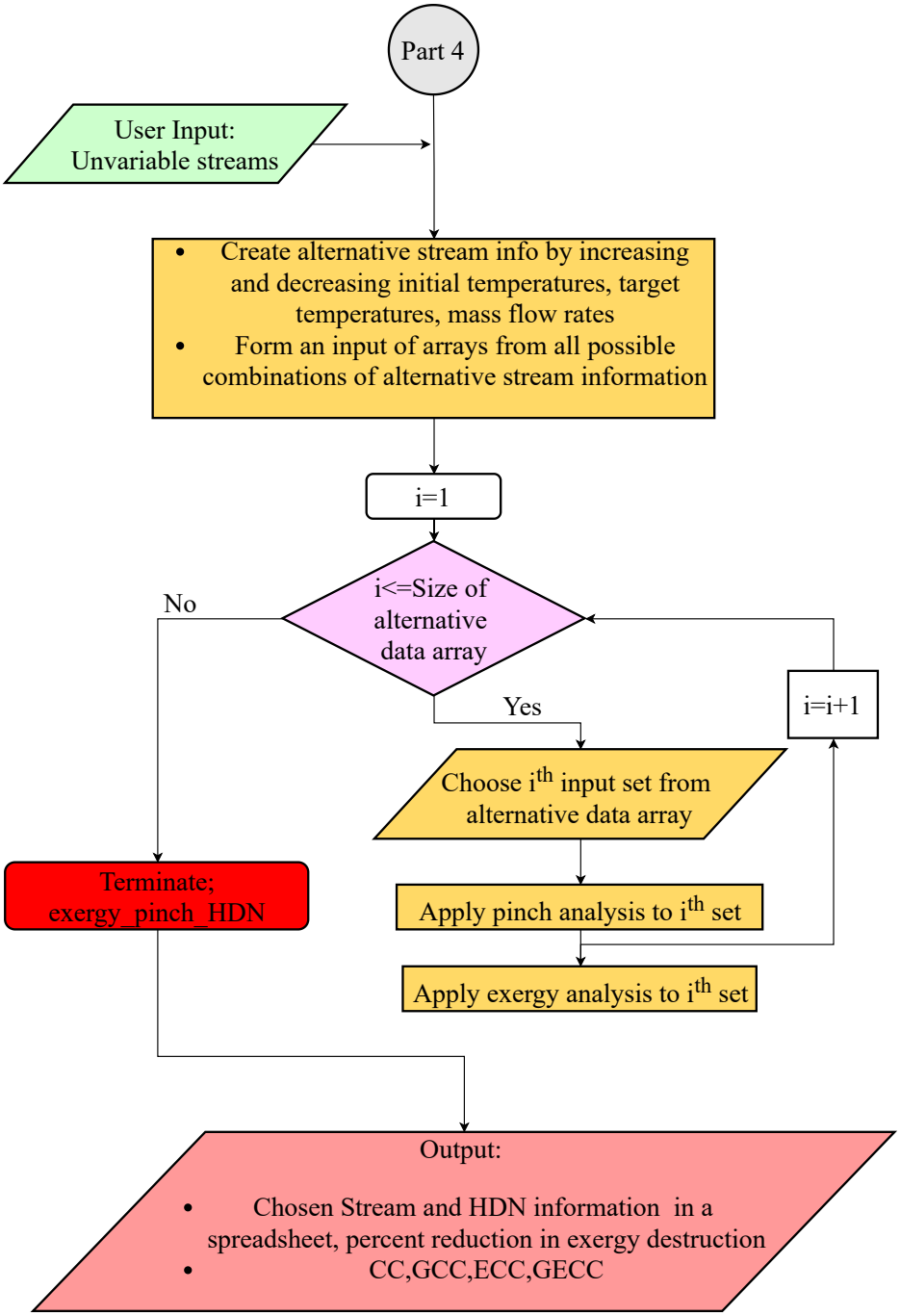


Figure 3.6: Flowchart of PCSTET Main Function, Part 4

3.2 Pinch Analysis Function

The flow charts describing the "Pinch Analysis" Function are presented in Figure 3.7, Figure 3.8 and Figure 3.9. The initial phases of PCSTET and the "Pinch Analysis" function (pinch_Calculate) are almost the same. After a press on the "Read" button, a related function (readUserInputs) starts and triggers a clean up (resetContainers_Pinch) in all of the properties for a possible re-run of the tool with any modification on inputs. Then existing information on the tables in Pinch Analysis GUI is collected. Then the "Pinch" button is activated.

Clicking on the "Pinch" button initially starts a function (populateStrms) to convert table information to usable stream information. Since tables for hot and cold streams are each capable of collecting information for up to five streams, with one row per stream, every stream is checked separately. If all the values are zero, that means there is not such a stream, and if they are not, the row is recorded as one stream. After all of the rows are checked, creation of stream information ends. In order to form temperature intervals for hot, cold, and grand scales, values are made unique and sorted, which finalizes the population of streams. Then heat load values are calculated for each interval with a new function (getEnergies). Subsequently, cumulative values for loads are calculated at every interval calling a new function (accumulate) once for each hot, cold and grand separately. According to minimum energy requirement the smallest load value locates where CCs touch, T_{pinch}. And last function (EnergyCompositeCurveDraw) that is called before the termination of Pinch Analysis draws CC and GCC on "Pinch Analysis" GUI. In conclusion hot and cold utility amounts, T_{pinch} and composite curves are obtained with Pinch Analysis function and the "HEX" button is activated..

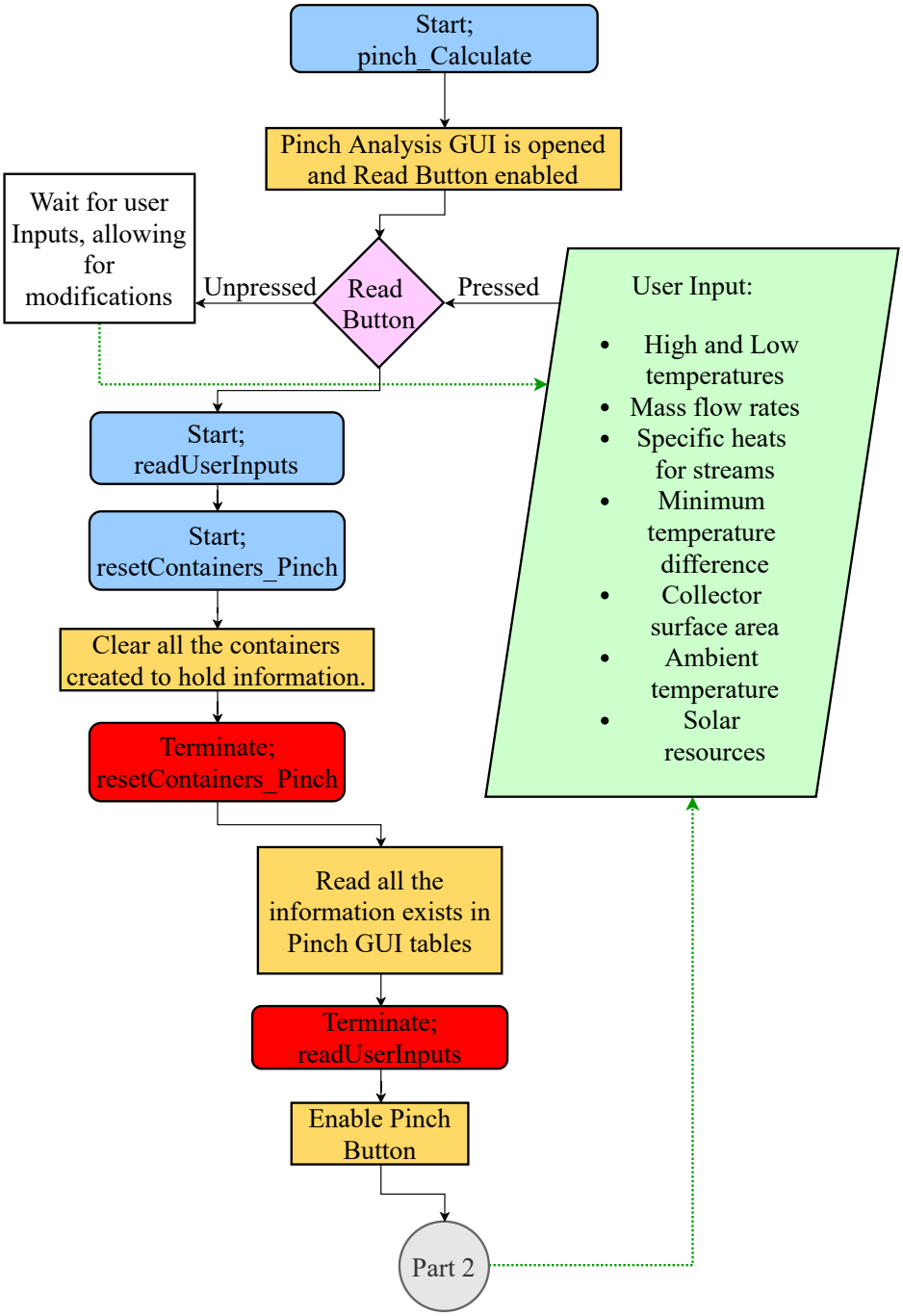


Figure 3.7: Flowchart of Pinch Analysis Function, Part 1

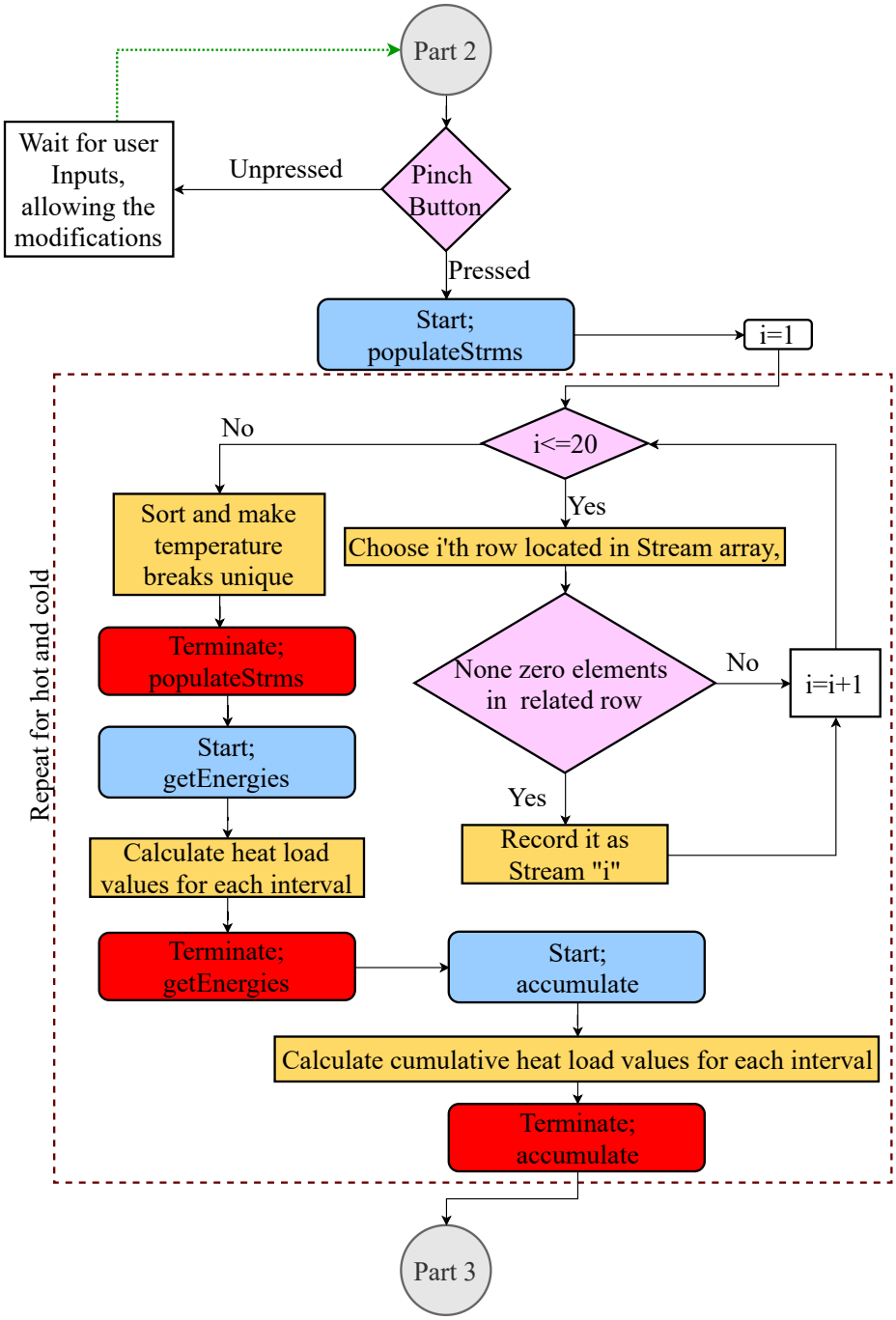


Figure 3.8: Flowchart of Pinch Analysis Function, Part 2

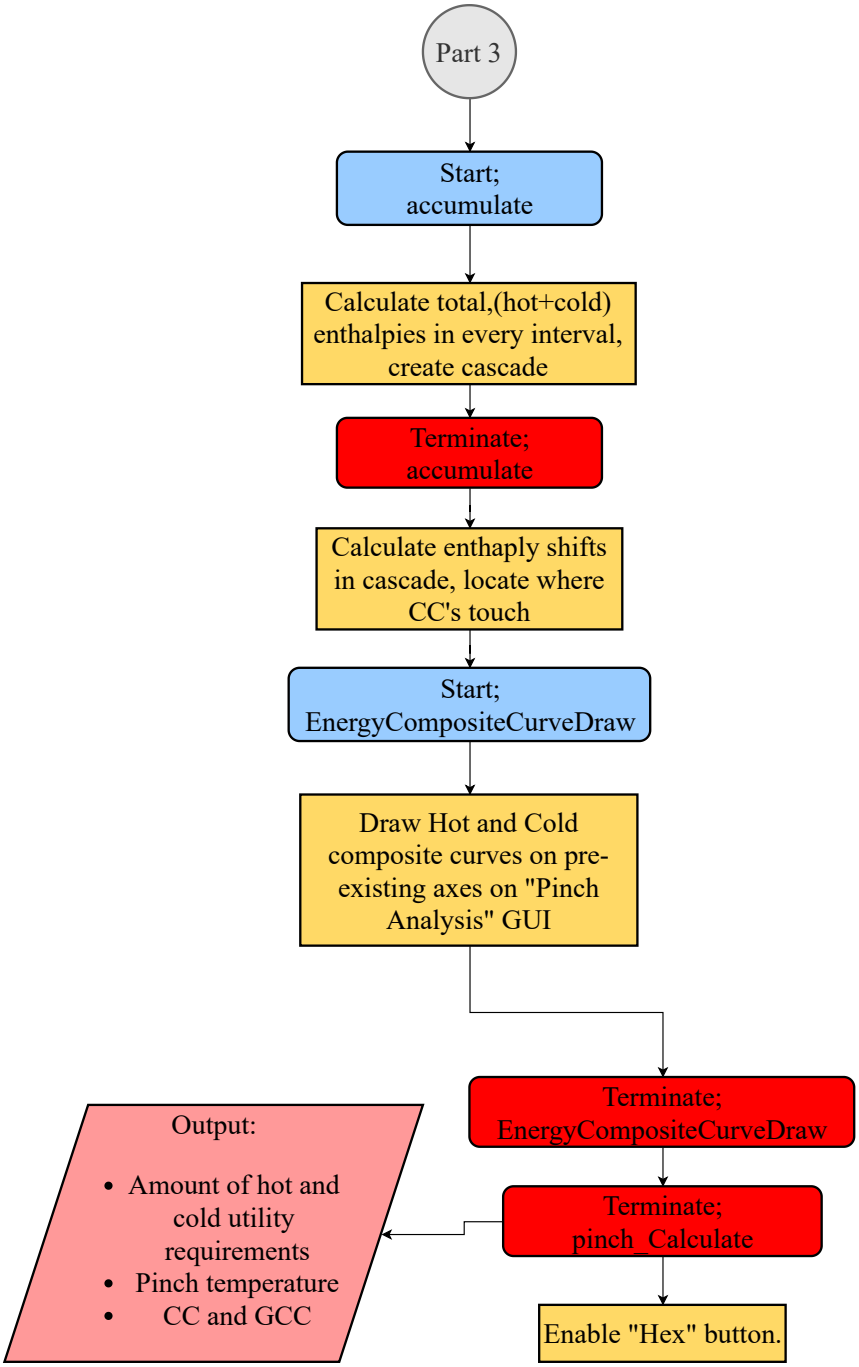


Figure 3.9: Flowchart of Pinch Analysis Function, Part 3

3.3 Create HDN Function

The main "Create HDN" Function (build_Hex_sourcesink) has only one main objective, and that is assigning streams into HEXs and utilities. However, in reality, it also constructs the mainframe for manipulating streams to HEX members. The flow chart for the HDN Tool/Function is presented in Figure 3.10.

Before explaining the flow of the code, the concepts of source and sink are revisited here. In section 1.1, these concepts are defined in a way that only hot streams are considered as sources. Though, since golden rules forbids any heat transfer from a hot stream above the pinch to a cold stream below the pinch, these concepts are redefined. Below pinch, all of the cold streams are matched with hot streams obeying all the constraints. After depletion of colds, cold utilities are employed. So, it is a valid perspective to think cold streams as the source and hot streams as the sink below the pinch. Similarly, hot streams become the source, and cold ones become sink above pinch. For either side sources are consumed entirely by sinks, then utilities are used for remaining sinks. According to this terminology Equation 1.4a and Equation 1.3a are combined as;

$$CP_{source} \leq CP_{sink} \tag{3.1}$$

Although Equation 1.3b and Equation 1.4b are also vital, no action is taken initially. Splitting streams without checking other necessary conditions would result in an inefficient match of streams. A press on the "HEX" button triggers the initialization of HDN. The outermost loop carries on as long as the source array is not empty.

The strategy here is choosing the first members of the source and sink arrays and looking for a match. Then streams that are assigned to a HEX, are removed from input arrays and this continues with the remaining streams until the source array becomes empty. At that point, there is no possibility to match remaining sink streams. Hence, they reach to their target temperature with the use of utilities.

Going back to the establishment procedure of HEX, three conditions should be satisfied. The first one is the abovementioned CP condition. The second one is the basis of all of the study, which is ΔT_{min}. The third one is the heat load balance between the source and the sink. It is a low probability to have two streams satisfying all of these

at the same time without any manipulation. Nevertheless, if any sink stream satisfies conditions for the first a HEX is created there.

A software solution to this problem is found as follows: if a relaxation amount between CP values is not stated, the program cannot decide if CP value is big enough. Instead, values are made equal by a function (splitstream_sourcesink). Then in order to satisfy energy balance, another function (divide_streams_sourcesink) is used, and the temperature continuum is divided into sections according to load amounts once the other two conditions are met. The majority of the division is completed with targeting. A temperature value for source or sink is aimed, and then the division strategy is employed. The procedure for no target and details of targeting will be explained in subsection 3.3.2.

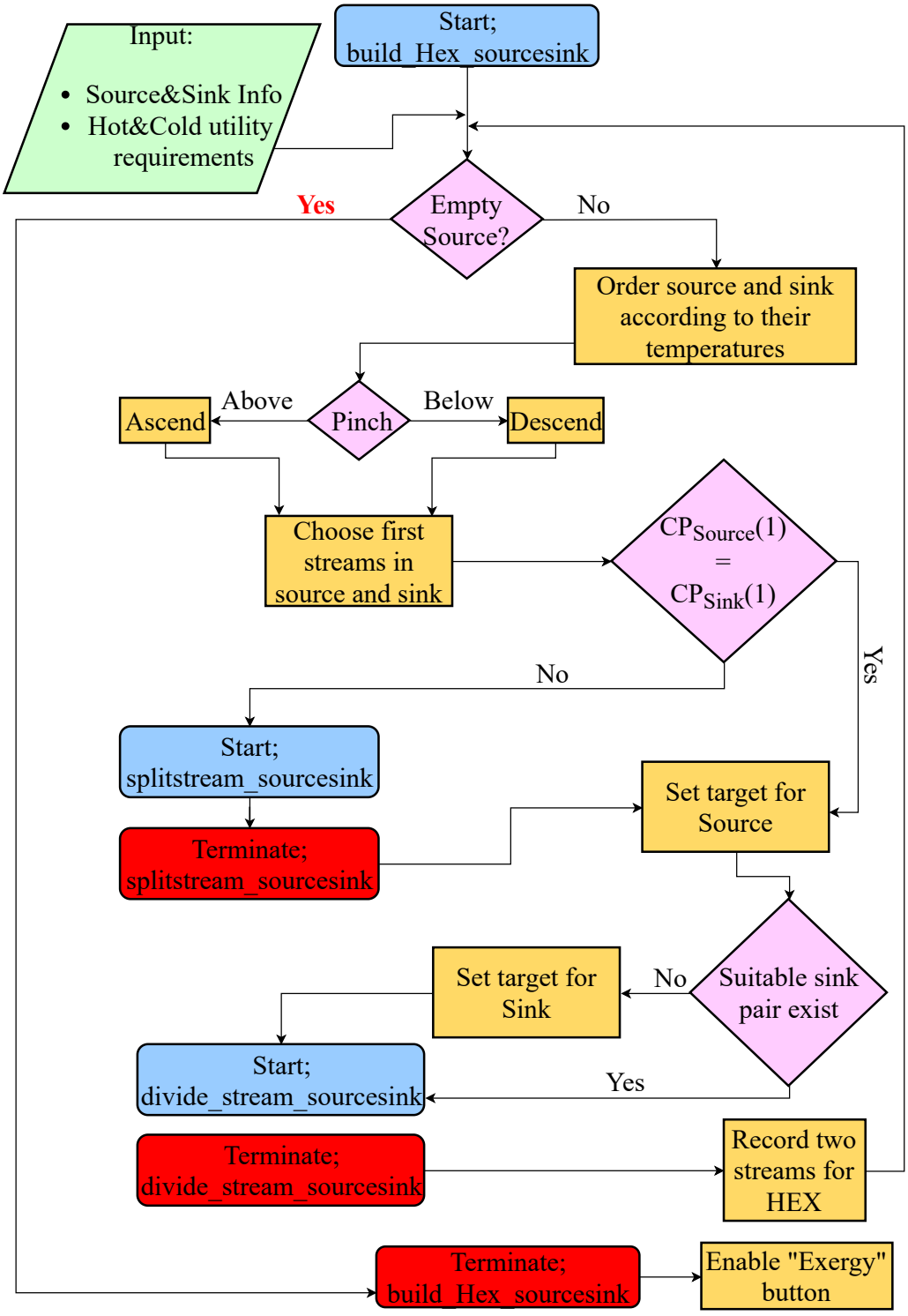


Figure 3.10: Flowchart of Create HDN Function

3.3.1 Split Stream Function

Stream splitting is generally used for sources because of the CP condition. Nevertheless, PCSTET requires equal values for them. For this purpose "Split Stream" Function(splitstream_sourcesink) is created which can be seen in Figure 3.11. The first two conditionals determine which CP value is higher than the other, and which stream to divide. Thus, if a split from any stream is necessary, the original stream is subtracted from the original array and obtained two streams added at the top of the related array. After a HEX connection is established, streams are ordered according to temperature. Eventually, the Create HDN function continues with the updated source and sink. The flow chart for the stream split function is presented in Figure 3.11.

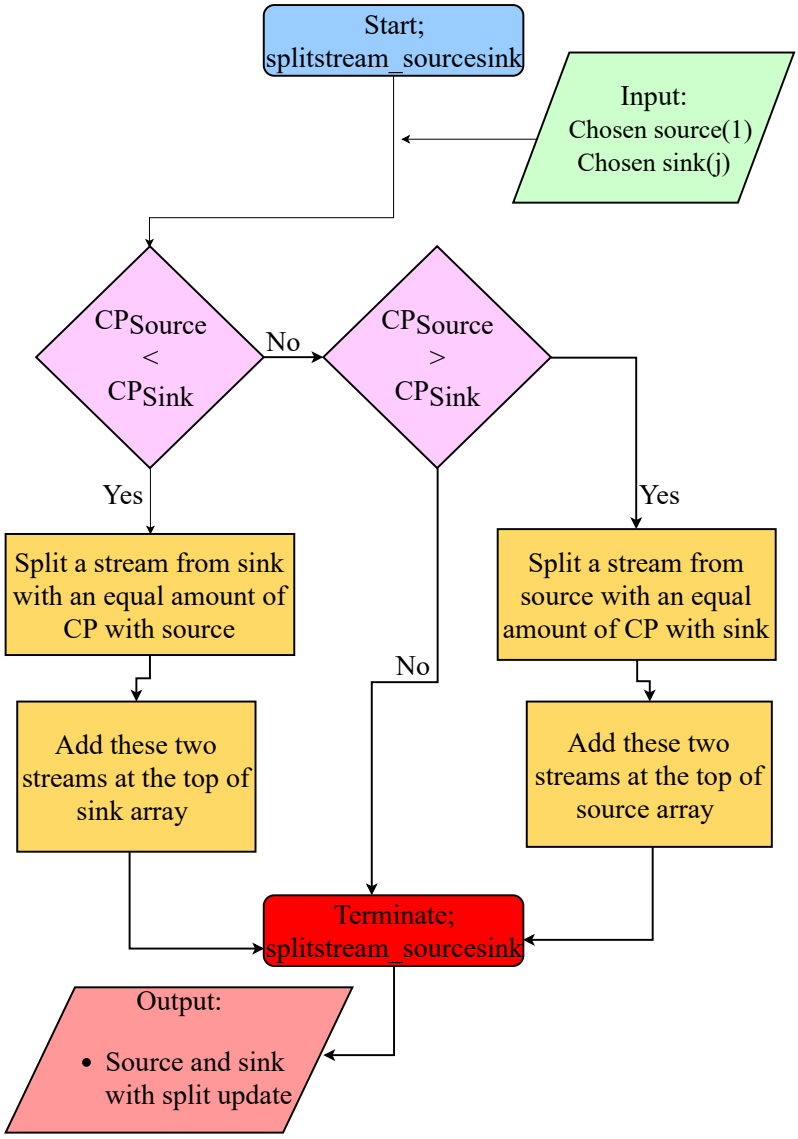


Figure 3.11: Flowchart of Split Stream Function

3.3.2 Divide Stream Function

Division of streams into two is a strategy used to meet energy targets inside the HEX. "Divide Stream" Function (divide_streams_sourcesink) is used for this purpose and it employs two entirely different methods for two different conditions(Figure 3.13, Figure 3.14). In order to be sure about the match decisions, a concept called targeting is suggested. The main idea is dividing the temperature continuum of the entire problem into sections according to the existence of the streams. First, all of the beginning and end temperatures for all of the streams are recorded. Then counter limits are also recorded. For instance, if the recorded value is for a hot stream, the counter limit will be the value ΔT_{min} lower than this value. For the cold streams, counter hot limits are located at a temperature that is ΔT_{min} higher than it. These values are recorded as target pairs. For the entire temperature interval, a pair is created for every start or end temperature of streams (Figure 3.12).

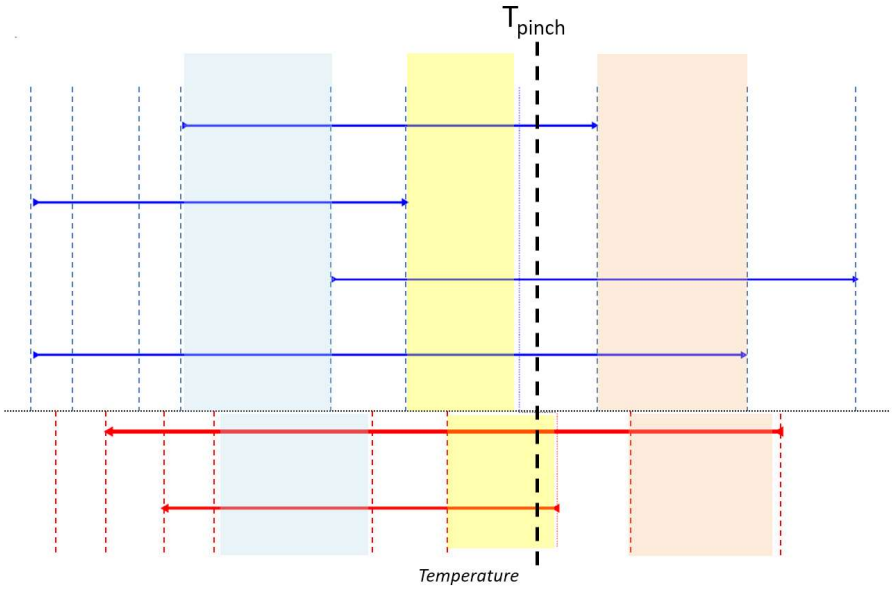


Figure 3.12: Target for streams

The main purpose is reaching the target temperature for the source first. The next sink is tried to be reached to the counter target. This continues until no source is available

for a sink to reach the target temperature, at which point a utility is used and is called the no target case.

If there is not a target that is set before the procedure, the stream with a bigger heat load value is divided. The first part starts from the side that is close to pinch. End temperature is calculated in a way that its heat load equals to heat load value of the smaller one. The second part is what is remaining. However, in some cases, it is possible to find a breakpoint that is outside of the interval for the related stream. In order to solve this problem, the division is conducted with a dummy variable. This dummy variable is only used if the breakpoint is in the interval and ΔT_{min} is satisfied.

Since this is the last function for the build of HEXs, manipulation of source and sink is completed here as well as the selection of streams for a single HEX. For instance, if the heat load value of the first element of the source is higher than the first element of the sink's, a dummy division array is created for the source. Breakpoint location and ΔT_{min} are tested. If dummy satisfies both, the element with equal heat load to sink and corresponding sink are recorded as HEX streams. For the source, the first element is replaced with what is left from the division, and the related element is removed from the sink. The same procedure is valid for the sink also.

As a last note, these chosen targets are not final decisions. Before dividing streams and trying to locate HEXs, targets are checked according to their end temperatures as well as the compatibility of left streams after division to the remaining streams. If the final temperature for the sink is outside of the interval when the source is targeted, a new target is chosen as the final temperature of the sink. For smaller values of pre-specified ΔT_{min}, a lot more criteria are needed to be implemented. These are the limits for the time being.

According to the order of streams on the source and sink array, if a target is missed, the next HEX connection will not be between streams used in the current connection. Thus, the solution does not diverge due to the target. Similarly, for the no target case, one of the streams is consumed entirely. This means the solution will not diverge to an infinite number of HEXs either.

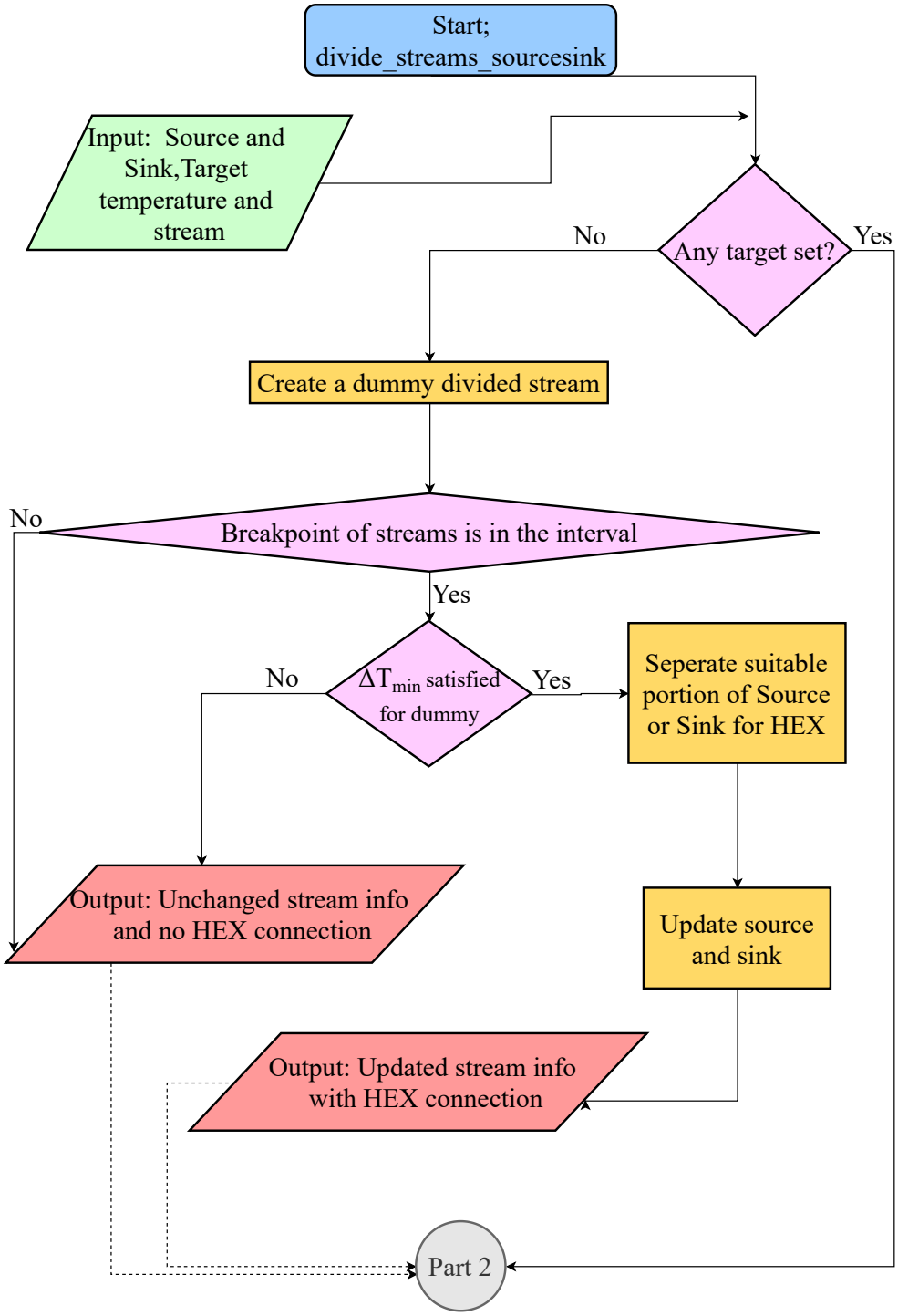


Figure 3.13: Flowchart of Divide Stream Function-Part 1

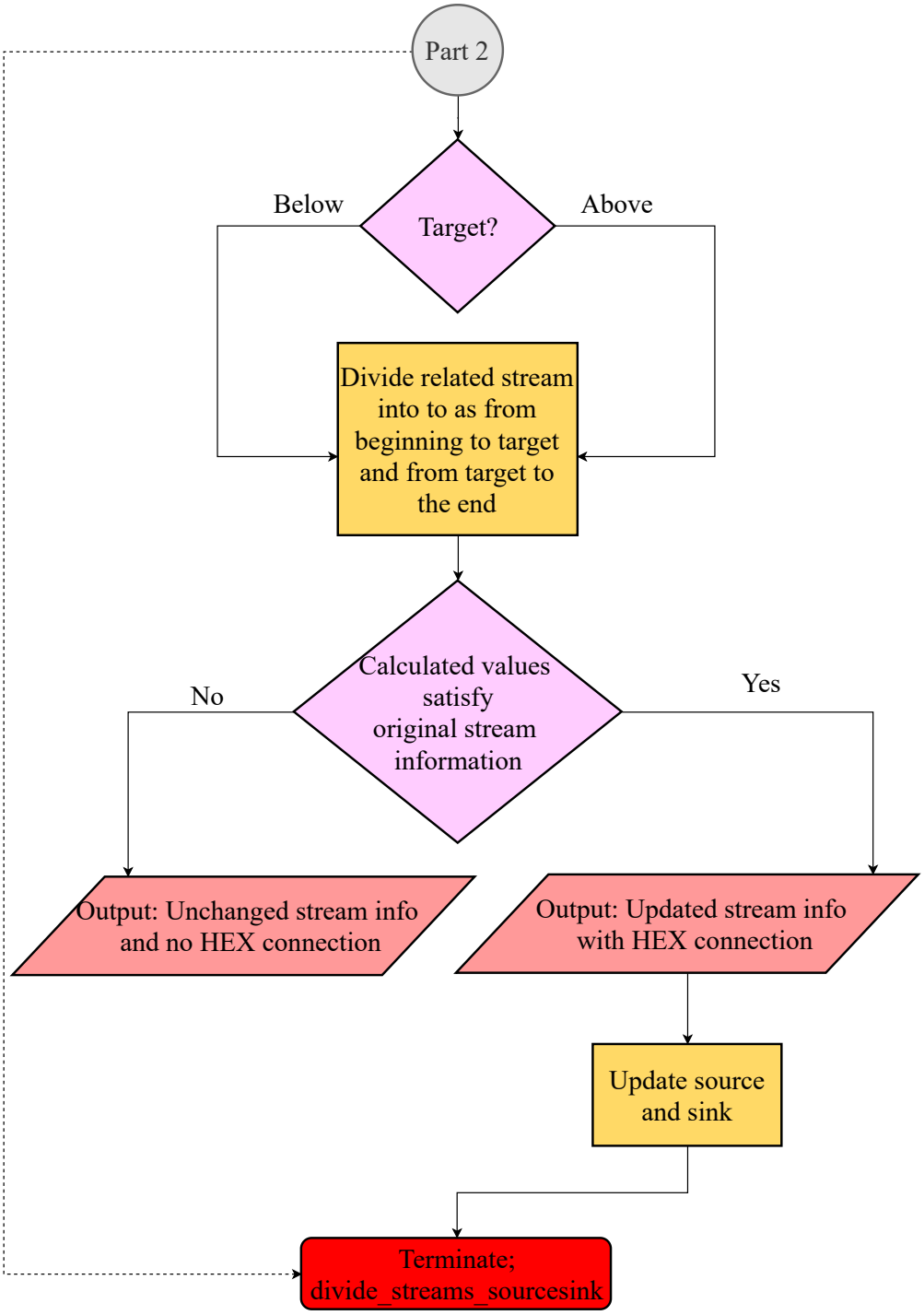


Figure 3.14: Flowchart of Divide Stream Function-Part 2

3.4 Exergy Calculation Function

As it can be seen from Figure 3.15 the first step of "Exergy Calculation" Function (exergy_Calculate) starts with a press on the "Exergy" button. This is the calculation of exergy destruction in HDN; in other words, exergy destroyed in the heat recovery system. The value also will be equal to the area under GECC. For the scope of this tool, hot utilities are provided by CST. The difference between the exergy of solar irradiation and exergy utilized by CST is another exergy destruction. This exergy destruction value includes destruction from the sun to the receiver, thermal losses, from receiver to fluid and optical losses. But as explained in section 2.4, only thermal losses should be considered for exergy destruction calculations. Calculation of exergetic thermal loss share requires receiver temperature. It is an unknown for this model. Bellos et all suggest 1.3 to 26.7% share for exergetic thermal loss share for different working fluids like oil and air [25]. In order to observe the effects of varying inputs on exergetic thermal loss share, the exergy destruction value is multiplied with a factor(0.03) that may be changed according to working fluid. As an output, an array with two kinds of exergy destruction and total exergy representation is obtained.

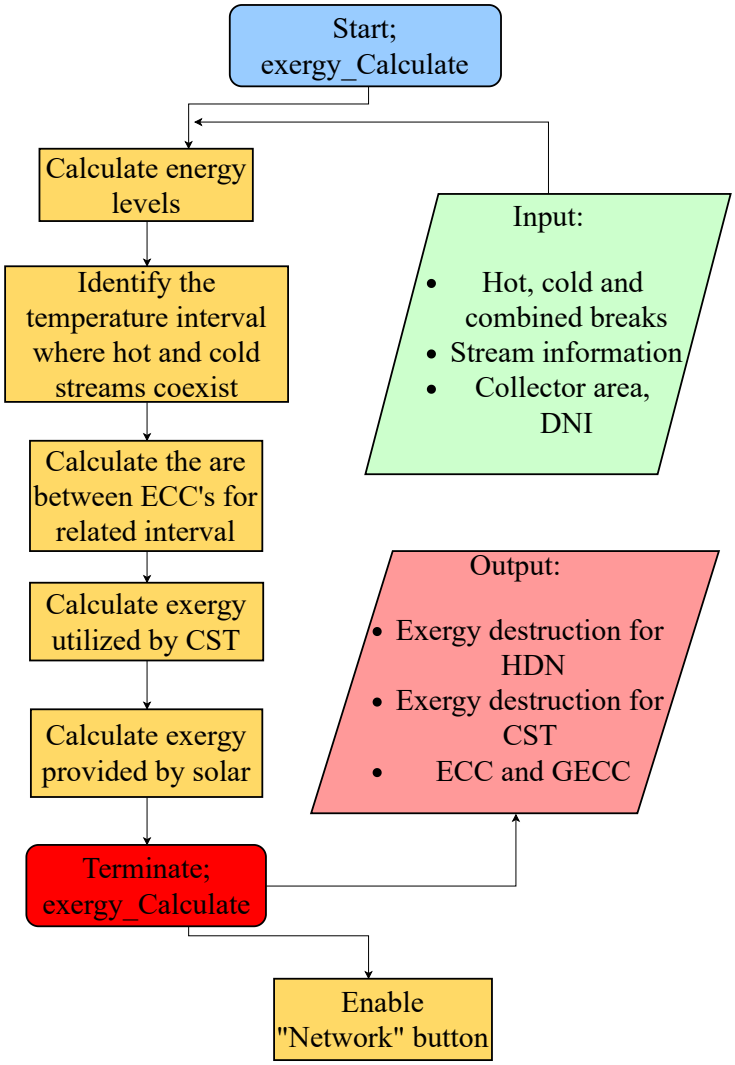


Figure 3.15: Flowchart of Exergy Calculation Function

3.5 Optimization Function

The optimization strategy here is to obtain an extensive data set with every possible variation of selected input parameters as defined in subsection 3.5.1.

In order to provide a better comparison between initial inputs and chosen inputs, the original stream and HDN information are stored in spreadsheets. Before manipulating the original inputs, the constraints are provided by the user. Checkboxes for every stream are shown in the Network GUI. There may be streams that can be altered and streams that cannot be altered. This is an implemented restriction to the optimization part of PCSTET. The user marks the ones that cannot be altered. Pressing on the "Optimize" button collects this information and initializes "Optimization" Function (OptimizationCallBack). For the streams that can be altered, all alternatives are stored in an array. This is actually a critical criterion to implement to PCSTET. Usually, cost, space, and time restrictions make some processes impossible to manipulate in any way. So providing the user an option to identify constant stream properties due to any reason is very beneficial.

Initially the original inputs are recorded as optimum data set with minimum exergy destruction. In a loop that is created for every possible combination, an alternative is chosen. For this one Pinch Analysis, Create HDN, Exergy Calculate functions are called in order. Then obtained exergy destruction is compared with the previously recorded minimum of exergy destruction. If the new value is smaller than the original, minimum exergy destruction and optimum inputs are updated as those values. If the found exergy destruction is greater than the original, no update occurs. This loop continues as many times as the number of possible alternatives, and at the end of the loop, stream and HDN information are written to new spreadsheets in the same file. The percentage of decrease in exergy destruction is displayed in the command window. This actually concludes one successful operation of PCSTET (Figure 3.16, Figure 3.17).

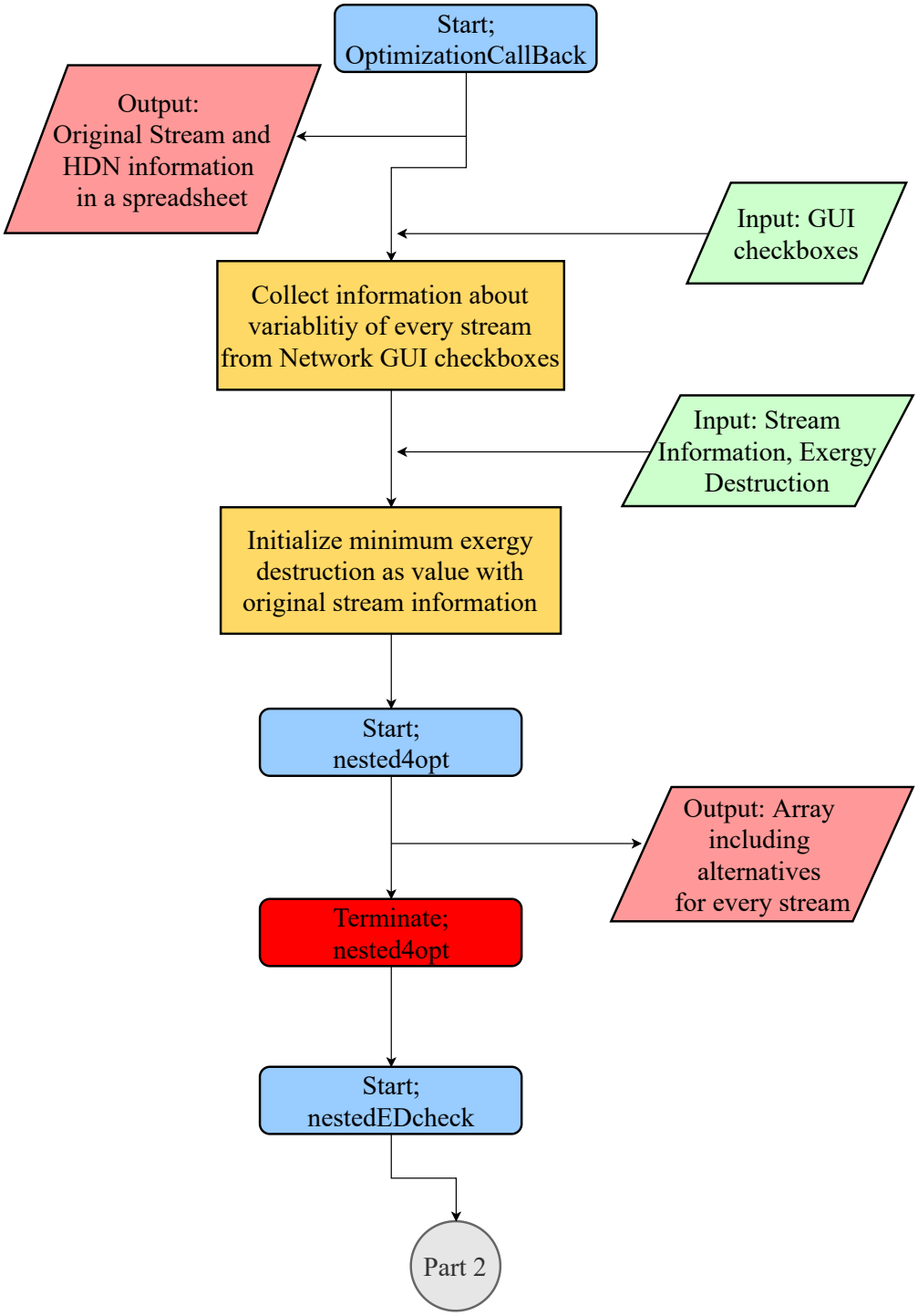


Figure 3.16: Flowchart of Optimization Function, Part 1

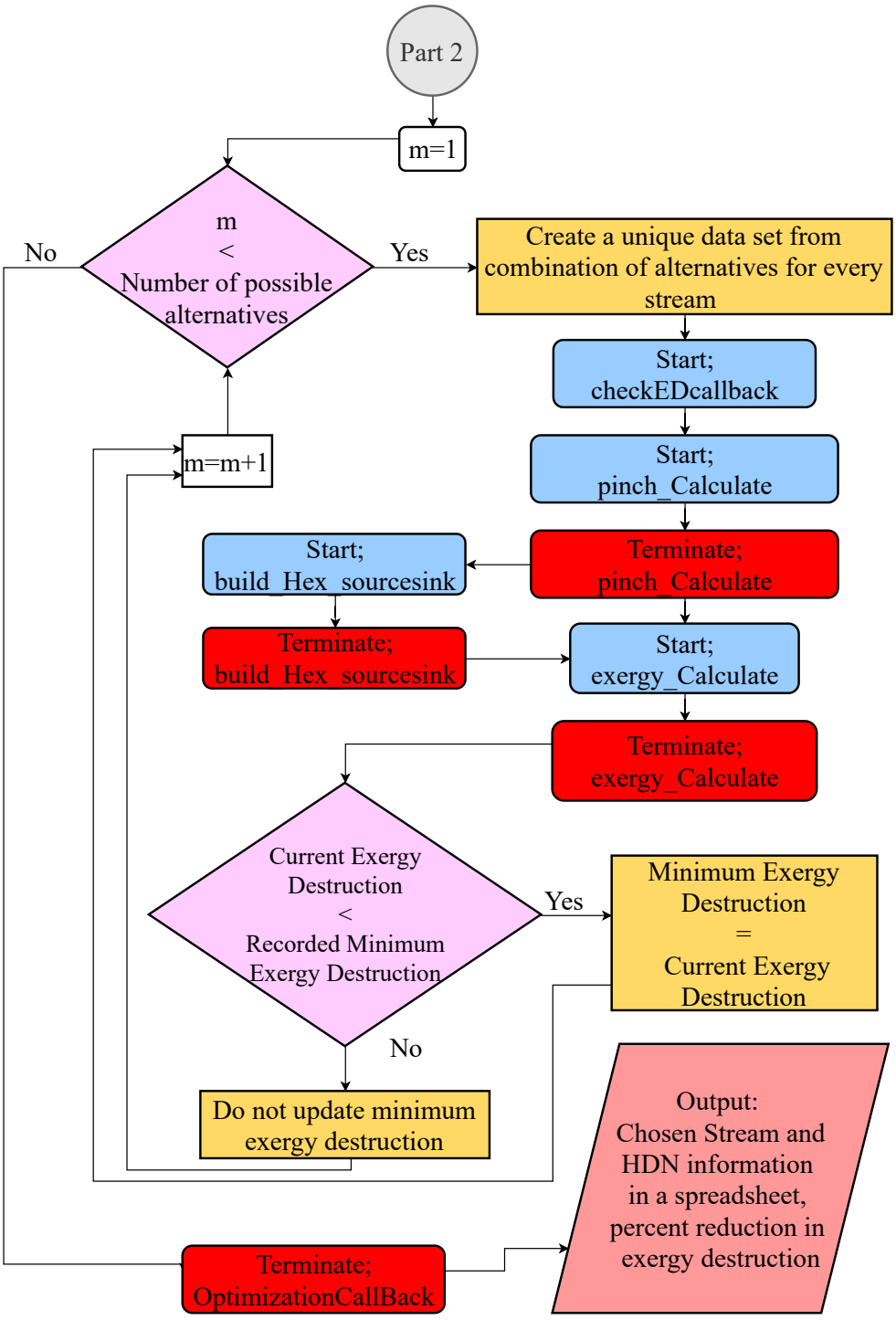


Figure 3.17: Flowchart of Optimization Function, Part 2

3.5.1 Alternative Function

In order to improve performance, the case study examples in the literature [22]usually change temperature and mass flow rates because remaining factors are hard or expensive to alter. The flowchart, Figure 3.18, demonstrates the creation of the alternatives by "Alternative" Function as explained in section 2.5.

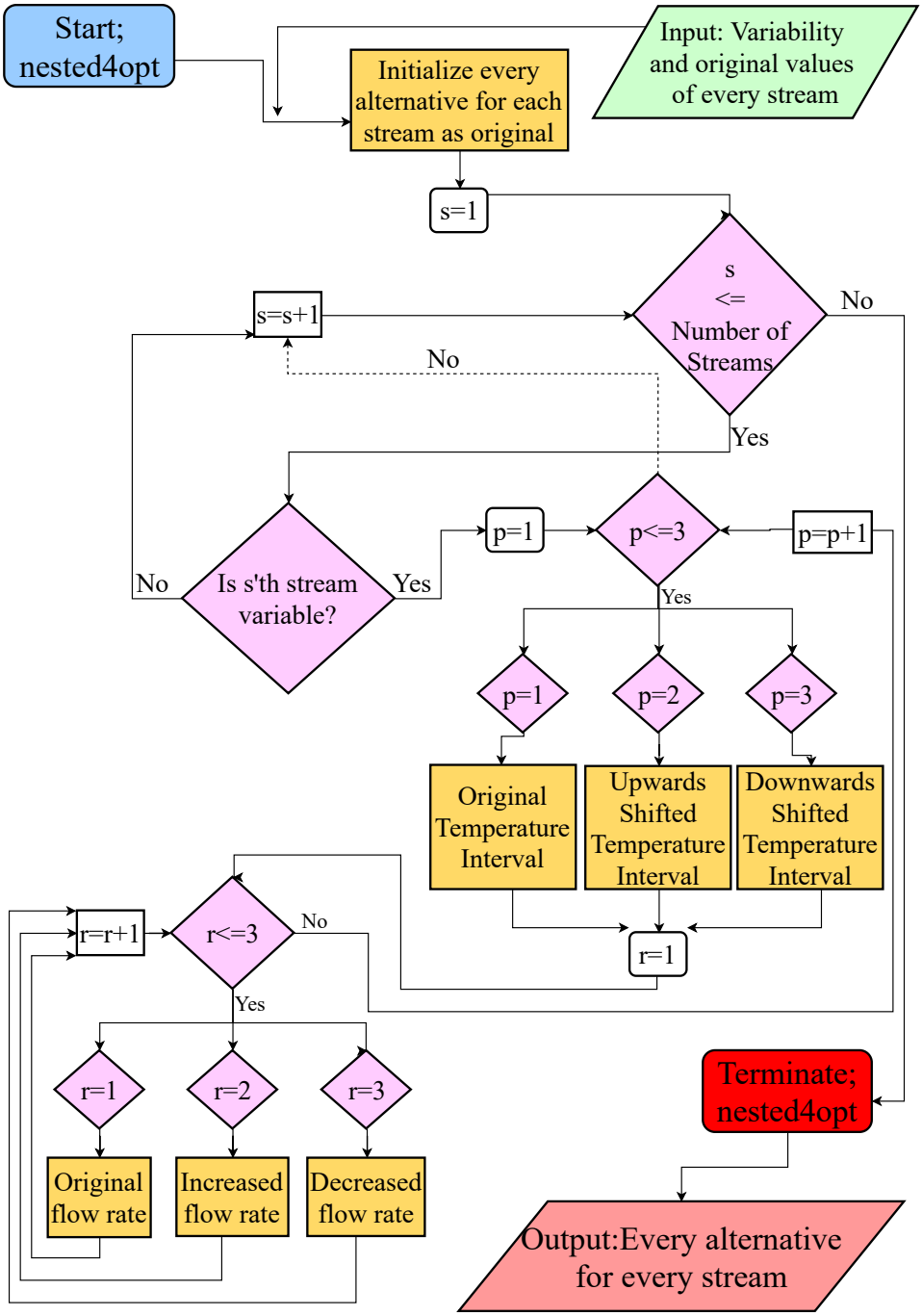


Figure 3.18: Flowchart of Alternative Function

CHAPTER 4

PCSTET USAGE, BENCHMARK AND CASE STUDY

Details of the construction and working principles of the PCSTET tool are explained in chapter 3. At first, it may seem like the current and the previous chapter are about the same topics. Nevertheless, chapter 3 explains underlying coding behind PCSTET and provides a guideline for a design of a tool that uses combined pinch and exergy analysis with CST implementation. However, chapter 4 explicitly focuses on the usage and validity of PCSTET. For the following sections, screenshots for GUIs will be provided as well as data tables. It should be noted that Figures 4.1 - 4.16 each require one full page, and to avoid having pages with only a few lines of text separating these sequential figures, all of the GUIs presented in chapter 4 are presented on sequential pages at the end of this section.

4.1 Usage of the PCSTET and Consistency Check

The amount of utility requirement is mentioned several times throughout the thesis. The importance of it is that the value of it will provide a basis for validating the correctness of the model, as explained in subsection 1.2.2. The nature of pinch analysis suggests that except the amount for hot and cold utilities all of the other heating and cooling requirements of streams are met by other streams. So, if in terms of heat load what is left after all possible stream matches are established, is equal to initially calculated utility requirement, it can be easily said tool works correctly in terms of HDN build up. However, this does not mean created HDN is the best. It only shows that the model is entirely consistent in terms of heat loads.This check is actually implemented into the model, which will be revealed through the usage explanation. Important de-

tails and visual aids are provided in this section. For the remaining benchmark and case study, only the necessary information is provided. However, all of the limitations that are explained in subsection 1.5.3, should be remembered before the details of the usage of PCSTET.

In order to show entire process, 4-stream example in section 1.4 is re-visited. By starting from scratch, all steps are explained as;

1. There are two possible methods to start PCSTET. The first one is calling "exergy_pinch_HDN" in MATLAB's command window. This requires setting the current folder as the same folder which "exergy_pinch_HDN" and "stream" class files located. The second one is opening the exergy_pinch_HDN file and pressing the "Run" button. This would open the "Pinch Analysis" GUI as presented (Figure 4.1).

2. GUI appears with some pre-existing inputs.

   - Stream inputs are updated with the data on Table 4.1.
   - Direct normal insolation is entered as $350W/m^2$.
   - Collector area is considered to be $6m^2$.
   - Dead state temperature is assumed to be $20°C$.
   - ΔT_{min} is $10°C$.

Table 4.1: 4 Stream Information

Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$
1(hot)	60	170	1.000	3
2(hot)	30	150	1.000	1.5
3(cold)	20	135	1.000	2
4(cold)	80	140	1.000	4

One adjustment is made for stream inputs. For Table 2.1, stream 1 is hot, stream 2 is cold, stream 3 hot and stream 4 is cold. This was the original presentation. But for PCSTET(Table 4.1), these numbers are started with the hot streams

first and after all hot streams are numbered cold streams are numbered. For instance, if original inputs contain four hot and three cold streams, the second stream corresponds to the second hot stream while the sixth stream corresponds to the second cold stream. For this specific example, stream 1 and stream 4 do not change their numbers. Stream 3 becomes second, and stream 2 becomes the third stream. Pressing on the "Read" button records inputs in tables on the upper right side and activates of Pinch button (Figure 4.2). The inputs may be altered as desired at any point. But in order to avoid errors and mistakes after any alteration, the user should go back to this step and start from pressing the Read button.

3. "Pinch" button completes pinch analysis and draws composite curves with utility requirement statements. For provided inputs, utility values turn out to be;

$$H_{hotutility} = 20kW$$

$$H_{coldutility} = 60kW$$

Pinch temperature is also found to be;

$$T_{pinch} = 85°C$$

The limit values are found by shifting T_{pinch} by amount of $\Delta T_{min}/2$ upwards for hot and downwards for cold streams;

$$T_{pinch,hot} = 90°C$$

$$T_{pinch,cold} = 80°C$$

On the upper left corner, composite curves are drawn in pre-existing axes. There are four plots on these. Hot and cold composite curves before and after shifting (section 1.4). On the upper middle location, the grand composite curve is presented in related axes. Then HEX button is activated (Figure 4.3). As stated before this is a very common example that is used to validate pinch models. PCSTET calculates the same values found with Linnhoff [1].

4. HEX button does not provide any visual aid at first. All of the calculations are run in the background. First streams are divided into two as above and

below pinch(Table 2.3). The heat exchangers are identified, and utilities are defined. For current inputs, constructed HEXs are provided as above (Table 4.2) and below (Table 4.3) pinch. In these tables, the first column with two rows expresses the HEX number while two elements in the second column define cold and hot streams in the HEX, respectively. The next two columns represent low and high temperatures. Then in order; mass flow rate, specific heat and heat loads for hot and cold streams are provided. A consistency check is performed at this stage. Above pinch, the summation of all heat loads for remaining streams after all possible matches are completed is compared with hot utilities. If they are equal, PCSTET prints, "Heat Exchangers are created successfully above pinch." on the command window. The same process is repeated below pinch with remainings and for the cold utilities. If they are equal too, PCSTET prints, "Heat Exchangers are created successfully below the pinch.". When they differ more than predetermined dimensionless error, which is 10^{-10}, a warning message is printed on the command window as "Work on build HEX function.". For the early stages of development and benchmarking, these messages printed. Nevertheless, they add extra load on the tool, and the last and most important part of the tool takes much time, which is up to days, to finish. Therefore, for instance if user want to use five hot and five cold streams and conduct an optimization where all of the streams can be altered in order to reduce the operation time, these messages can be disabled easily from the provided source code (section A.1). For this particular example, after a press on the HEX button; "Heat Exchangers are created successfully below the pinch." and "Heat Exchangers are created successfully above pinch." messages appeared in the command window. Next, the Exergy button is activated (Figure 4.4). Before moving forward with the remaining steps, it is necessary to touch on unnecessary heat exchangers here. The details will be explained in section 5.2 nevertheless, the main reasons for it may be listed as the requirement of equal CP values for source-sink pairs and targeting strategy.

5. Activated "Exergy" button completes the exergetic calculations for HDN and CST. The basis for exergy destruction minimization are these calculated exergy destruction values. Next, ECC and GECC are created on pre-existing axes on

Pinch Analysis GUI with the activation of the "Network" button(Figure 4.5).

6. A press on the "Network" button, shifts visual aid to a new GUI, namely "Network Grid" (Figure 4.6). Here horizontal axis shows temperature while the vertical axis represents stream numbers. Since keeping track of numbering and properties may be difficult, network diagrams that are presented in the body are visually modified. The lines are explicitly created for every stream individually. They carry some information with them. Color code is quite standard; red shows hot (to be cooled) streams and blue shows cold (to be heated) streams. The thickness of the lines is normalized according to CP values. It can not be said that measuring the thickness results in the identification of heat capacity. Nevertheless, relativity between the streams can be inferred from streams' line thickness. Pinch boundaries for hot and cold are shown with vertical dashed lines.

7. The visual presentation of the HDN is purposely divided as below and above pinch. Since one of the golden rules for pinch analysis forbids any heat transfer across the pinch(subsection 1.2.2), the HDN can be explained in a clear manner using different Network Grid diagrams for above and below the pinch. "Show HEXs Above Pinch" and "Show HEXs Below Pinch" buttons trigger these plots (Figure 4.7, Figure 4.8). The organization is exactly the same as the initial Network Grid diagram. Still, in addition, these plots include HEX lines which are connectors of the split, divided and matched streams. Endpoints of these HEX lines are actually average of T_{high} and T_{low}. As previously mentioned, heat exchanger lines are constructed in a stepped manner rather than traditional vertical lines. For further details and reasoning section 3.1 may be re-visited.

   As mentioned earlier, PCSTET also provides information on streams to be cooled or heated using utilities. These are provided in Table 4.4 and Table 4.5. These are also presented in the original form in the way that they are obtained from PCSTET. For instance, it may not be appropriate to use two different hot utilities here, and an expert HDN designer may decide to merge the first and second hot utility. In order to provide better information about the capabilities of PCSTET, adjustments are avoided. The same situation is valid for HEX information. Related comments are provided in section 5.2.

Table 4.2: Heat Exchanger Information Above Pinch, PCSTET

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	80	135	0.750	2.0	-82.5
	2	90	145	1.000	1.5	82.5
2	3	80	135	0.250	2.0	-27.5
	1	90	145	0.167	3.0	27.5
3	4	80	135	0.625	4.0	-137.5
	1	90	145	0.833	3.0	137.5
4	4	80	85	0.375	4.0	-7.5
	2	145	150	1.000	1.5	7.5
5	4	85	110	0.375	4.0	-37.5
	1	145	170	0.500	3.0	37.5
4	4	80	85	0.375	4.0	-7.5
	2	145	150	1.000	1.5	7.5
5	4	85	110	0.375	4.0	-37.5
	1	145	170	0.500	3.0	37.5
6	4	110	115	0.375	4.0	-7.5
	1	145	150	0.500	3	7.5
7	4	115	135	0.375	4.0	-30
	1	150	170	0.500	3.0	30

Table 4.3: Heat Exchanger Information Below Pinch, PCSTET

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	50	80	1.000	2.0	-60
	1	60	90	0.667	3.0	60
2	3	20	50	0.500	2.0	-30
	1	60	90	0.333	3.0	30
3	3	20	50	0.500	2.0	-30
	2	60	90	0.667	1.5	30

Table 4.4: Streams to be cooled by cold Utilities, PCSTET

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	2	30	60	0.667	1.5	-30
2	2	30	90	0.333	1.5	-30

Table 4.5: Streams to be heated by hot Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	135	140	0.375	4	7.5
2	4	135	140	0.375	4	7.5

8. Optimization process starts with a press on "Optimize" button. But before that user has to be sure that all unalterable streams are marked(Figure 4.9). For this data set stream 2 (2nd hot stream) is arbitrarily chosen as being unalterable. Initially the information provided in Table 4.2 and Table 4.3 are printed on a spreadsheet. Then the streams that can not be altered are recorded. Next all of the alternatives are created as explained in the subsection 3.5.1. Stream by stream all of the alternatives for all of the examples in the body are provided in Appendix B. Notice that for this specific example, alternatives for the second hot stream are all the same (section B.1). All possible combinations of these alternatives are subjected to Pinch Analysis, Create HDN and Exergy Calculate functions. From suggested alterations in the inputs, minimum exergy destruction is observed with data set provided in Table 4.6.

Table 4.6: 4 Stream Optimized Information

Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$
1(hot)	71	181	0.800	3
2(hot)	30	150	1.000	1.5
3(cold)	31.5	146.5	1.000	2
4(cold)	74	134	0.800	4

This set is recorded as optimum input set. Then Pinch Analysis, Create HDN and Exergy Calculate functions are re-applied to this set. As a result created object carries the information of optimum set. Then CC, GCC, ECC and GECC are plotted on related axes of new and final GUI(Figure 4.10).

Adjustments to stream information also affects HEX connections. Updated HEX array is provided as Table 4.7 and Table 4.8.

Table 4.7: Optimized Heat Exchanger Information Above Pinch,PCSTET

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	74	134	0.750	2	-90
	2	84	144	1.000	1.5	90
2	3	74	134	0.250	2	-30
	1	84	144	0.167	3	30
3	4	74	140	0.475	4	-114
	1	84	150	0.633	3	114
4	4	74	111	0.325	4	-7.8
	5	144	181	0.133	1.5	7.8
5	4	80	146.5	0.050	4	-1.2
	2	144	179.5	0.133	1.5	1.2
6	4	80	117	0.275	4	-40.7
	1	144	150	0.367	3	40.7
7	4	123	123	0.050	4	-7.4
	1	181	150	0.067	3	7.4
8	4	123	129	0.275	4	-6.6
	1	150	150	0.367	3	6.6
9	4	74	129	$2.200 \cdot 10^{-16}$	4	$-5.3 \cdot 10^{-15}$
	1	84	150	$3.000 \cdot 10^{-16}$	3	$5.3 \cdot 10^{-15}$
10	4	74	129.5	0.275	4	-7.15
	1	84	156.5	0.367	3	7.15
11	4	74	134	0.050	4	-2.2
	1	84	167.5	0.067	3	2.2
12	4	74	134	$2.200 \cdot 10^{-16}$	4	$-4.4 \cdot 10^{-15}$

Continued on next page

Table 4.7 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	1	144	161.5	$3.000 \cdot 10^{-16}$	3	$4.4 \cdot 10^{-15}$
13	4	111	134	0.225	4	-4.05
	1	144	161	0.300	3	4.05
14	4	111	134	0.050	4	-0.9
	1	144	165	0.067	3	0.9
15	3	117	146.5	0.350	2	-8.75
	1	144	173.5	0.233	3	8.75
16	3	123	146.5	$4.400 \cdot 10^{-16}$	2	$-1.1 \cdot 10^{-14}$
	1	144	174	$3.000 \cdot 10^{-16}$	3	$1.1 \cdot 10^{-14}$
17	3	74	146.5	0.100	2	-2.5
	1	84	178	0.067	3	2.5
18	3	74	146.5	0.100	2	-2.5
	1	84	180	0.067	3	2.5
19	3	74	141.5	0.350	2	-5.25
	1	84	181	0.233	3	5.25
20	3	74	141	$4.400 \cdot 10^{-16}$	2	$-6.2 \cdot 10^{-15}$
	1	144	181	$3.000 \cdot 10^{-16}$	3	$6.2 \cdot 10^{-15}$
21	3	111	137	0.100	2	-0.6
	1	144	181	0.067	3	0.6
22	3	111	140	$1.300 \cdot 10^{-15}$	2	$-8 \cdot 10^{-15}$

Continued on next page

Table 4.7 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	1	144	181	$8.900 \cdot 10^{-16}$	3	$-8 \cdot 10^{-15}$
23	3	117	138	0.100	2	-1.2
	1	144	181	0.067	3	1.2
24	3	123	139	$2.200 \cdot 10^{-15}$	2	$-4.4 \cdot 10^{-15}$
	1	144	181	$1.500 \cdot 10^{-15}$	3	$4.4 \cdot 10^{-15}$

Table 4.8: Heat Exchanger Information Below Pinch for Optimized Streams,PCSTET

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	61	74	1.000	2.0	-26
	1	71	84	0.667	3.0	26
2	3	48	61	0.200	2.0	-5.2
	1	71	84	0.133	3.0	5.2
3	3	31.5	61	0.75	2.0	-44.25
	2	54.5	84	1.00	1.5	44.25

As well as the number of utility requirements, the temperature intervals, and stream to be used on also changes. After optimization new streams that reach their target temperatures by utilities are obtained as Table 4.9 and Table 4.10.

Table 4.9: Updated streams to be cooled by cold Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	2	30	51	0.333	1.5	-10.5
2	2	30	54.5	0.267	1.5	-9.8
3	2	30	41.5	0.400	1.5	-6.9

Table 4.10: Optimized streams to be heated by hot Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	3	139	146.5	$2.22 \cdot 10^{-15}$	2	$3.33 \cdot 10^{-15}$
2	3	138	146.5	0.1	2	1.7
3	3	140	146.5	$1.33 \cdot 10^{-15}$	2	$1.73 \cdot 10^{-15}$
4	3	141	146.5	$4.44 \cdot 10^{-15}$	2	$4.88 \cdot 10^{-15}$
5	3	141.5	146.5	0.35	2	3.5

Optimized 4 stream example shows;

- Pinch temperature of $T_{pinch} = 79$
- External cooling of $H_{cu} = 27.2kW$, which means 54.67% decrease in cold utility requirement
- External heating of $H_{hu} = 5.2kW$, demonstrates 74% decrease in hot utility requirement
- 24.36% decrease in exergy destruction.

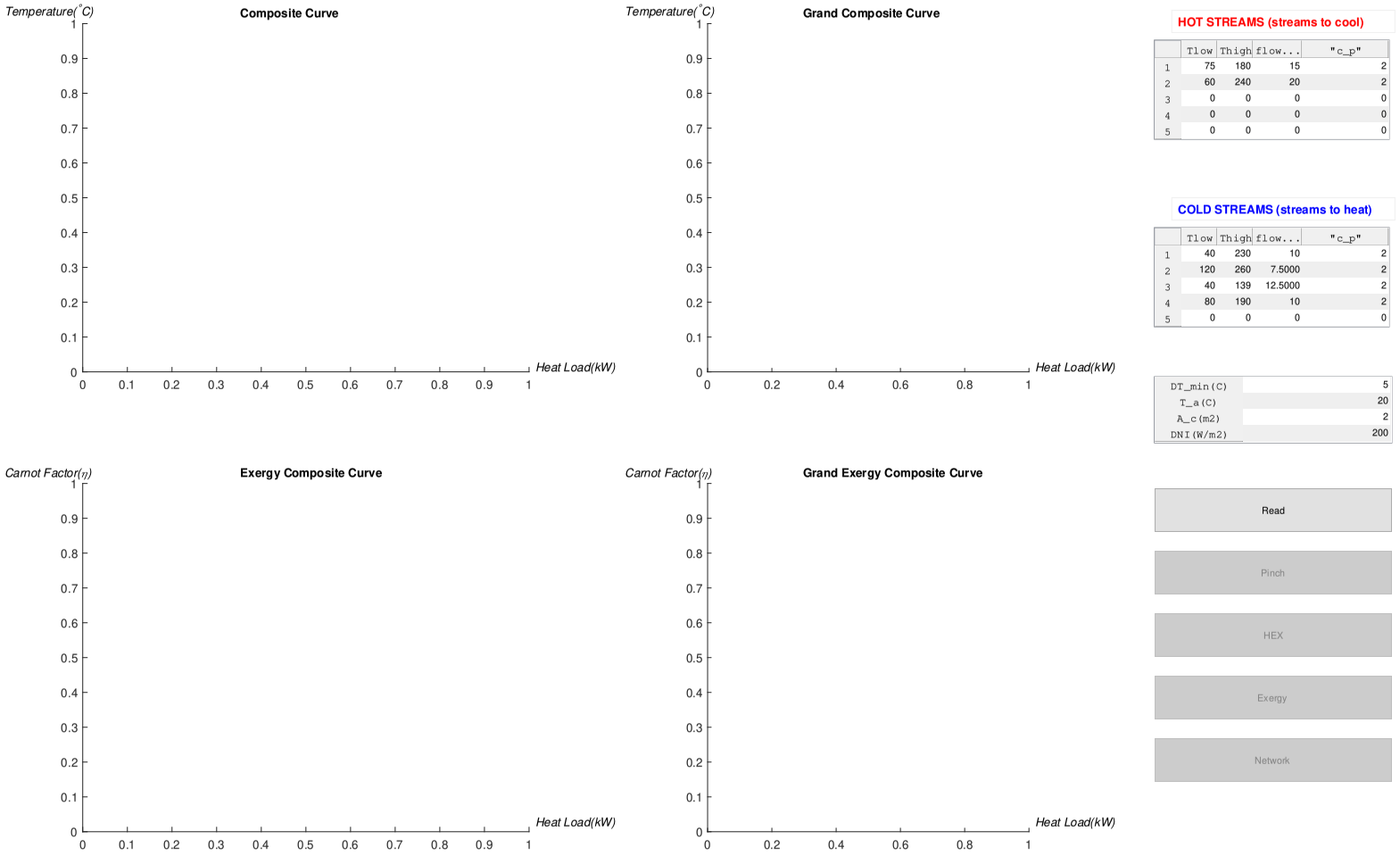


Figure 4.1: Pinch Analysis GUI-1

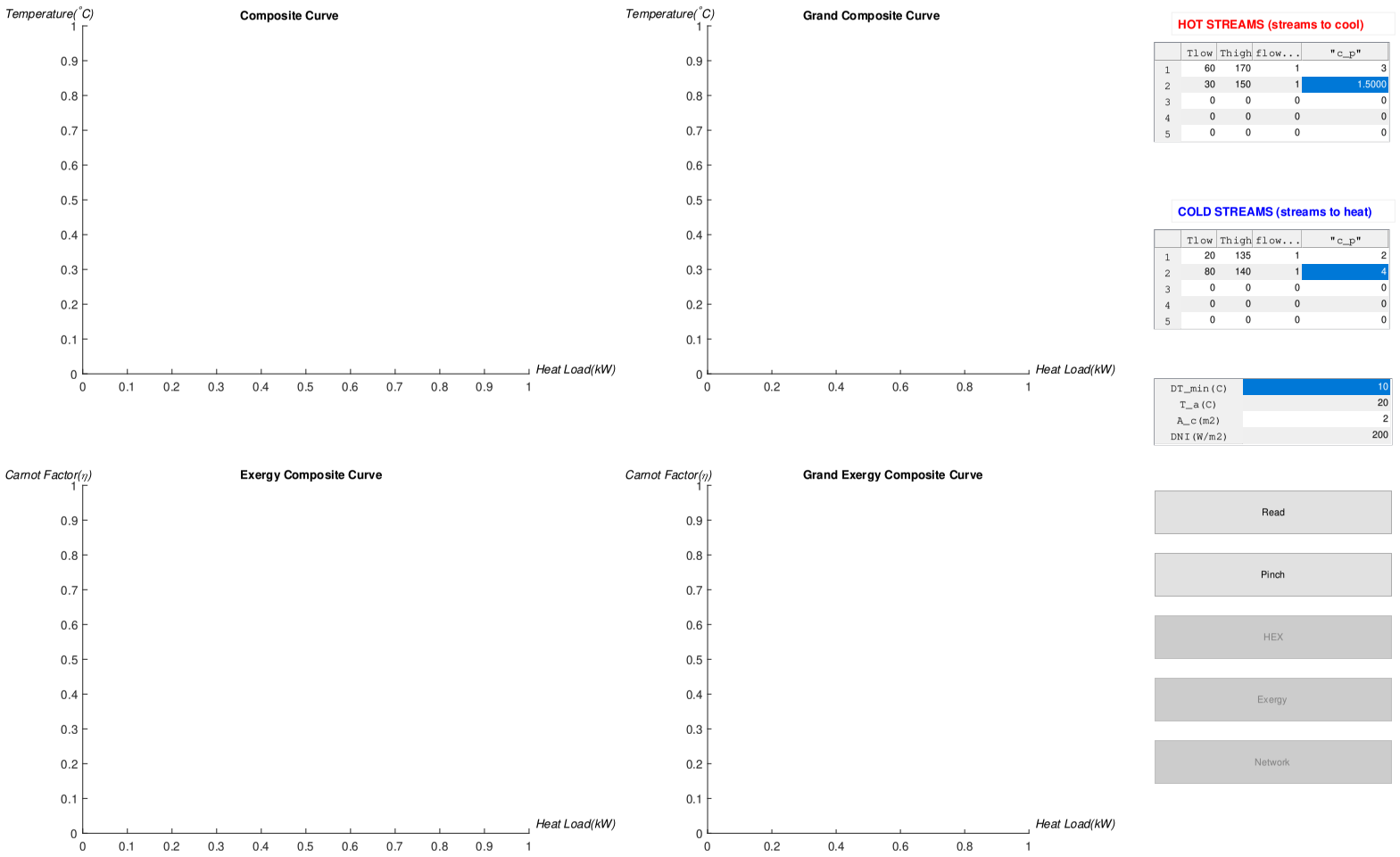


Figure 4.2: Pinch Analysis GUI-2

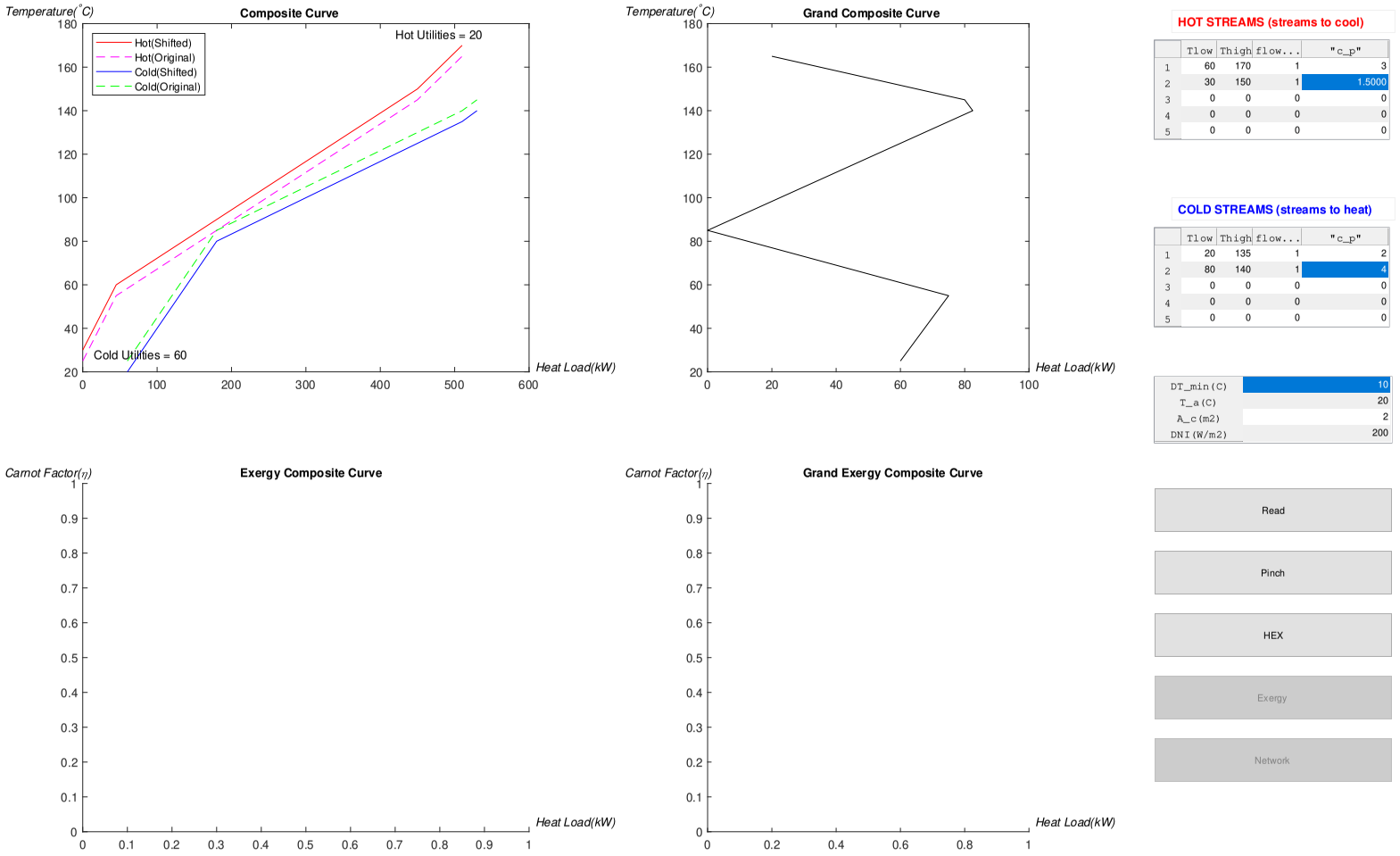


Figure 4.3: Pinch Analysis GUI-3

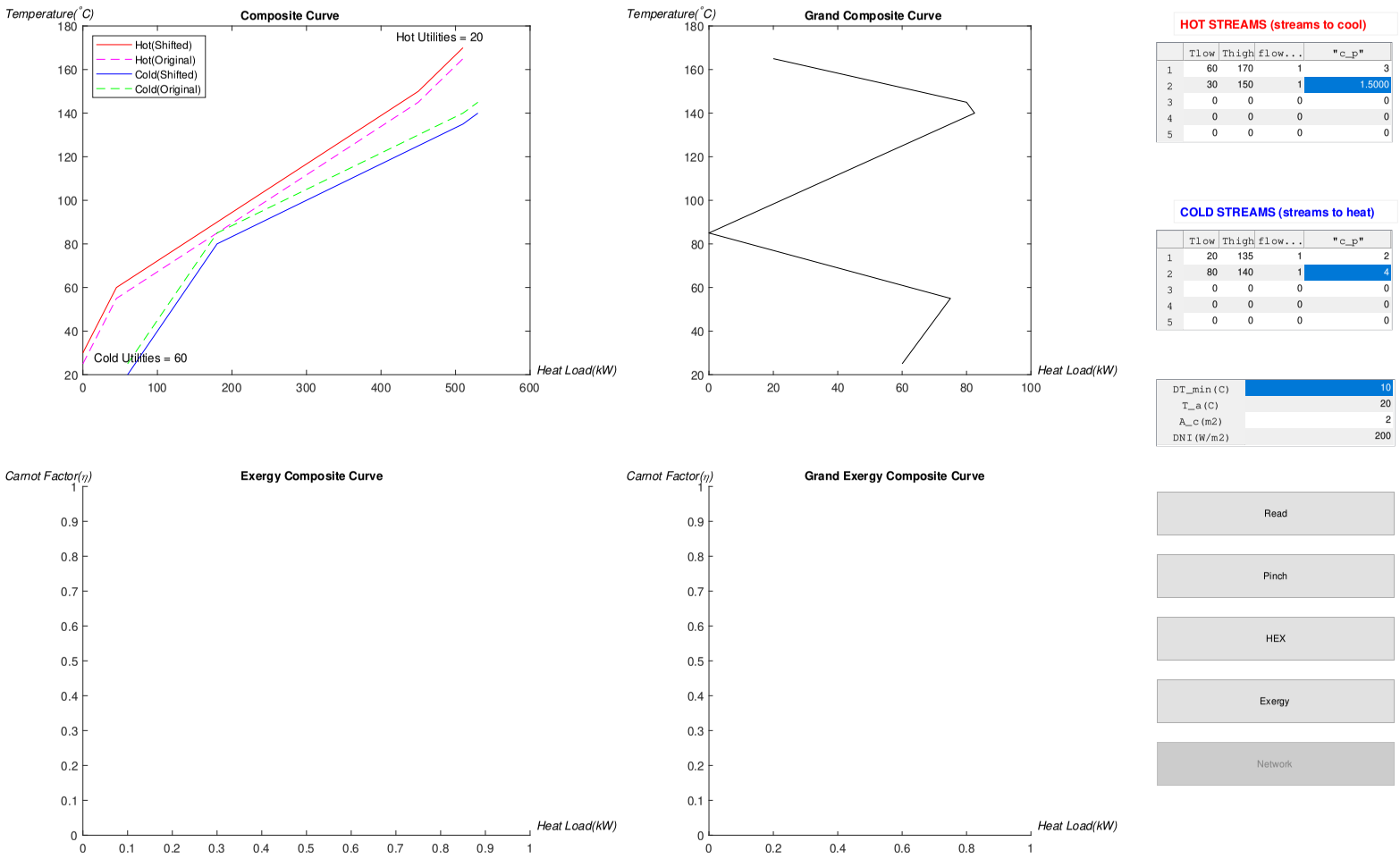


Figure 4.4: Pinch Analysis GUI-4

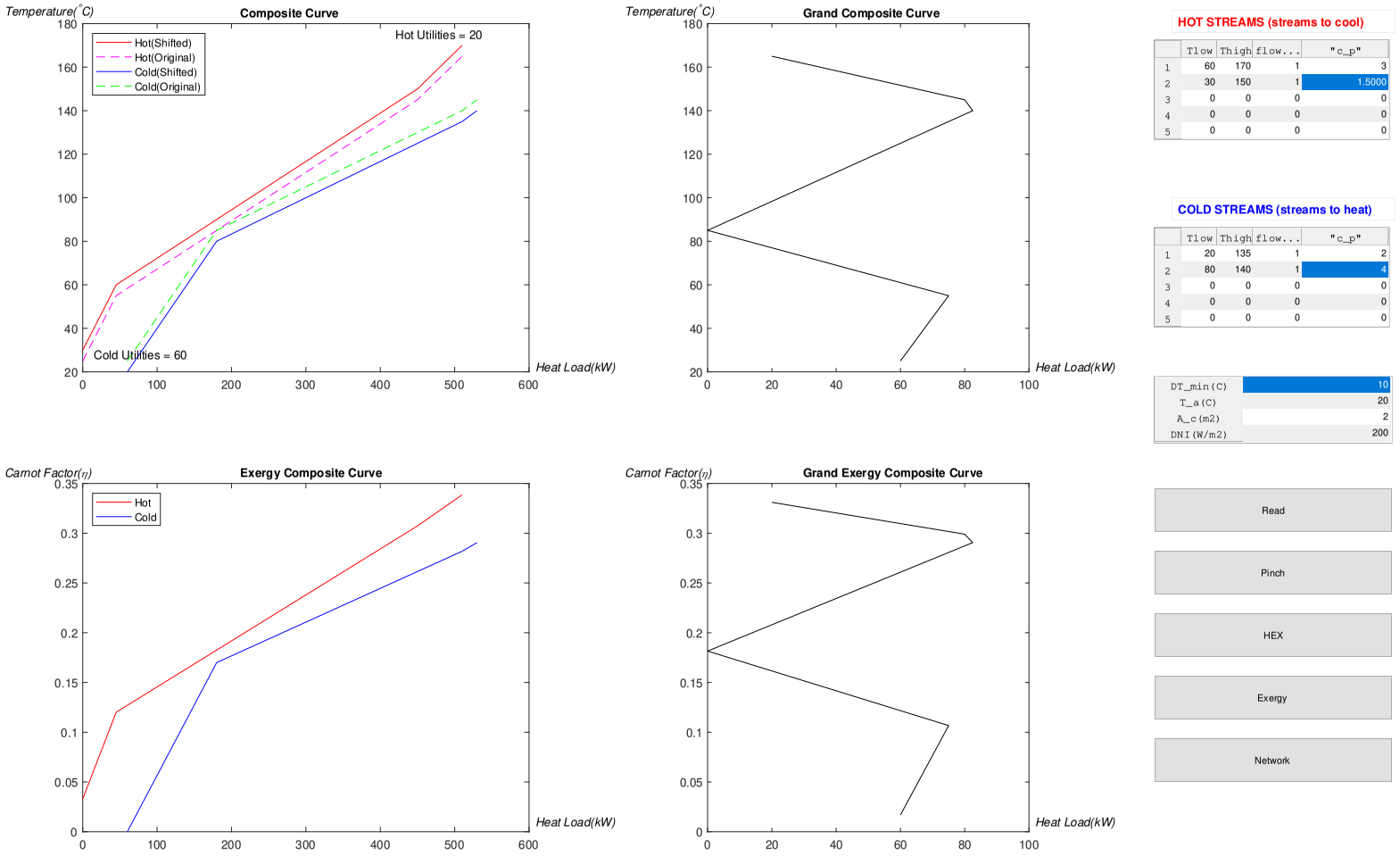


Figure 4.5: Pinch Analysis GUI-5

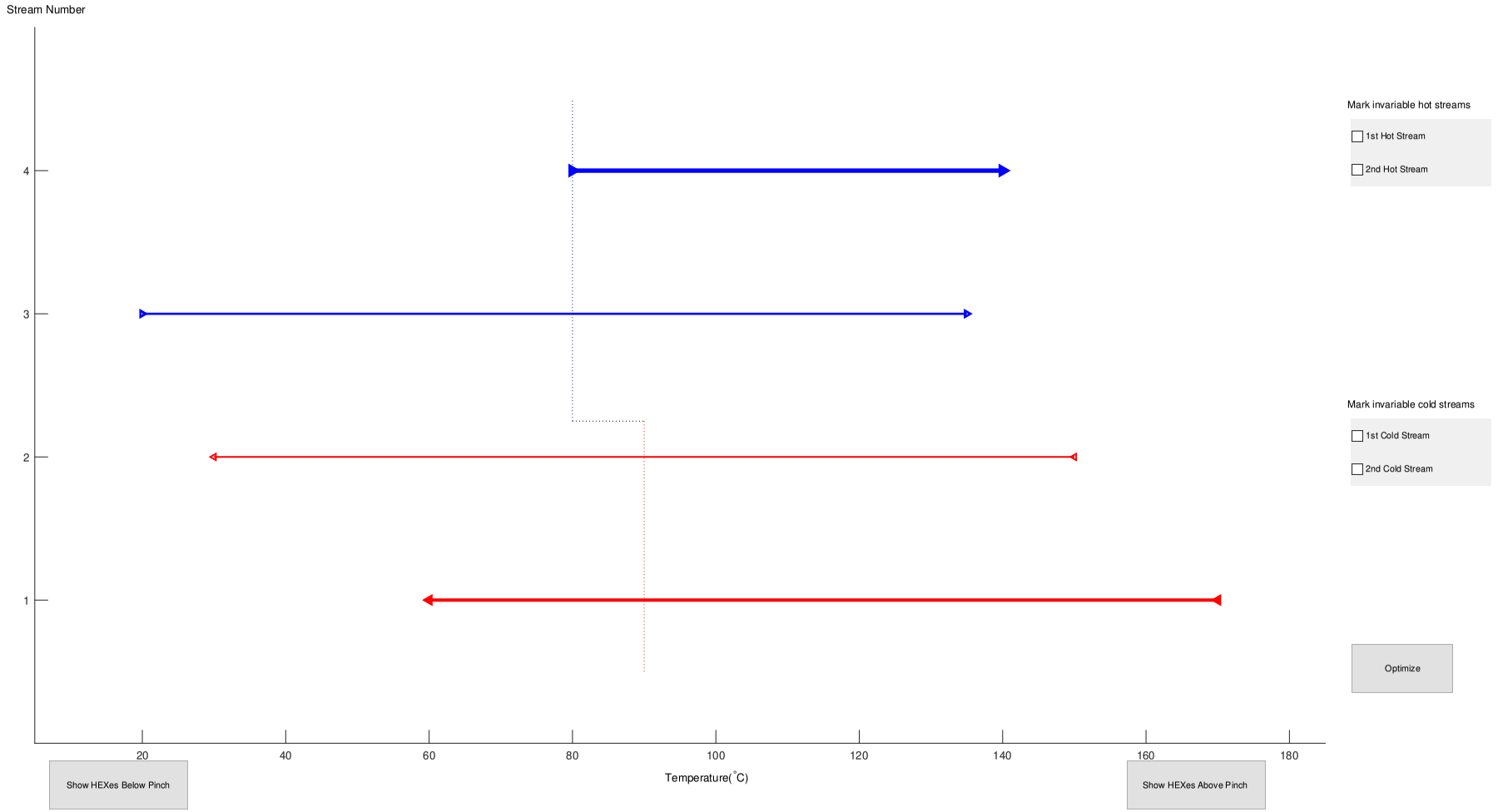


Figure 4.6: Network Grid GUI-1

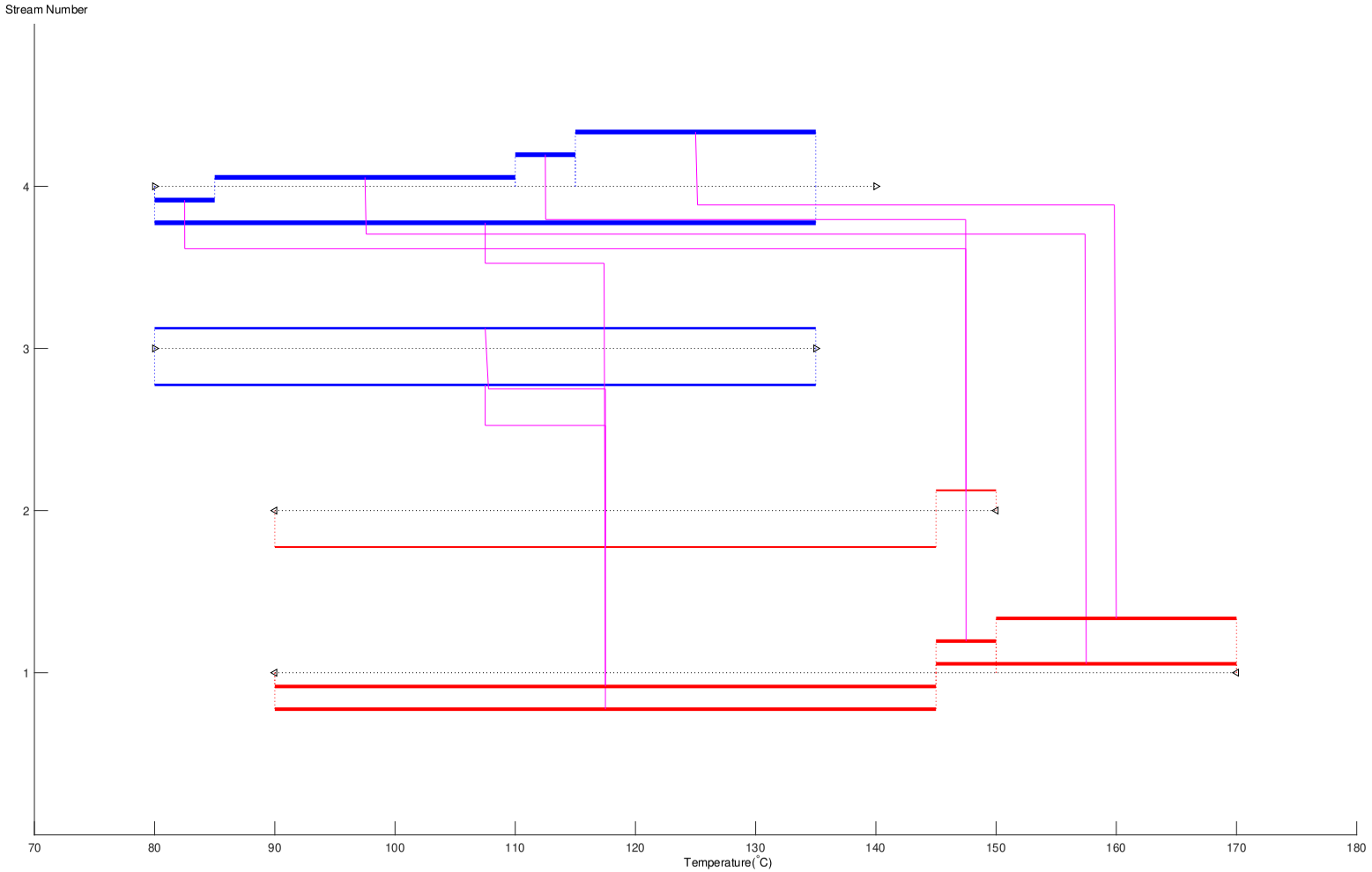


Figure 4.7: Above Pinch Network Grid GUI

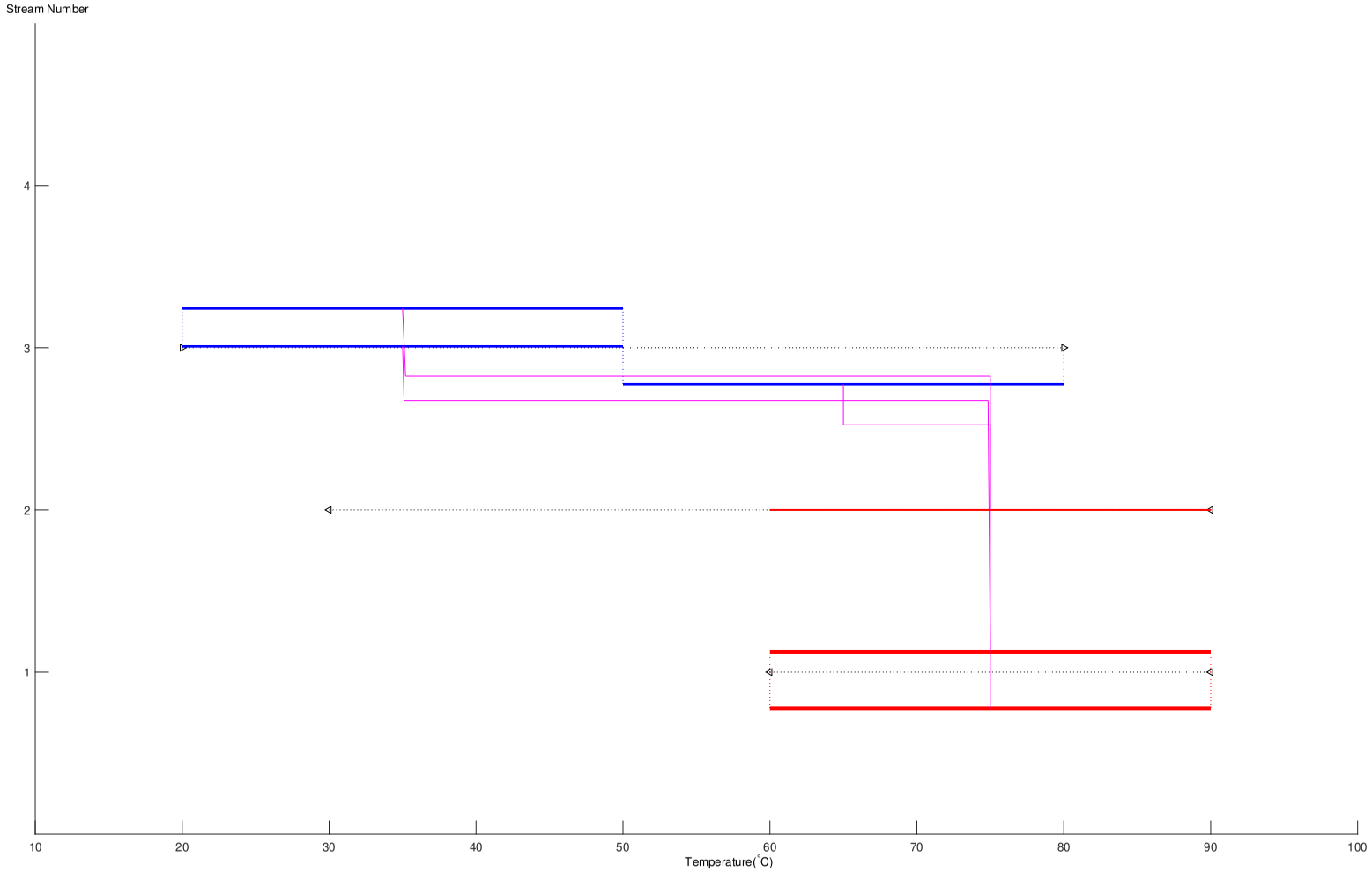


Figure 4.8: Below Pinch Network Grid GUI

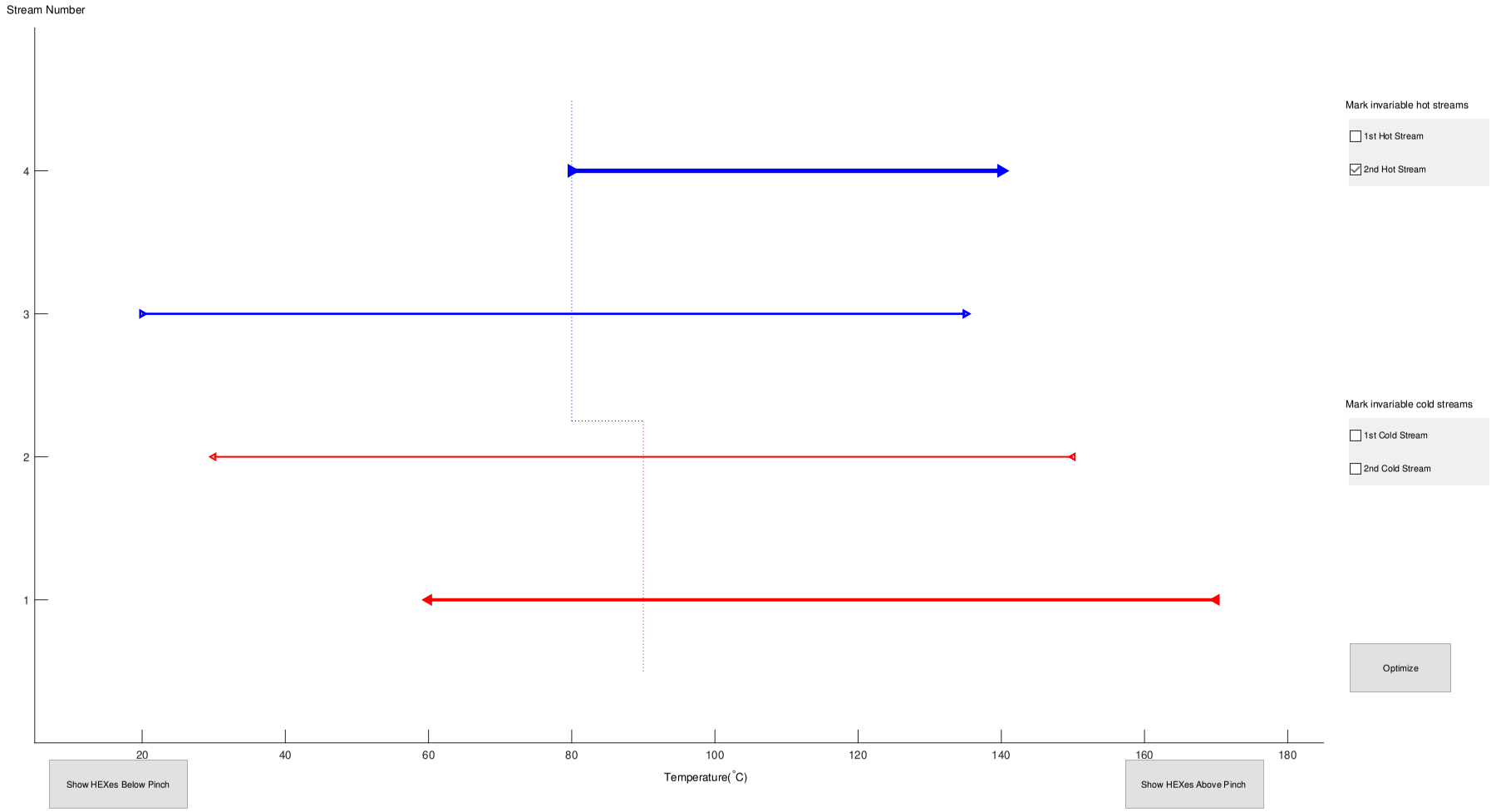


Figure 4.9: Network Grid GUI-1

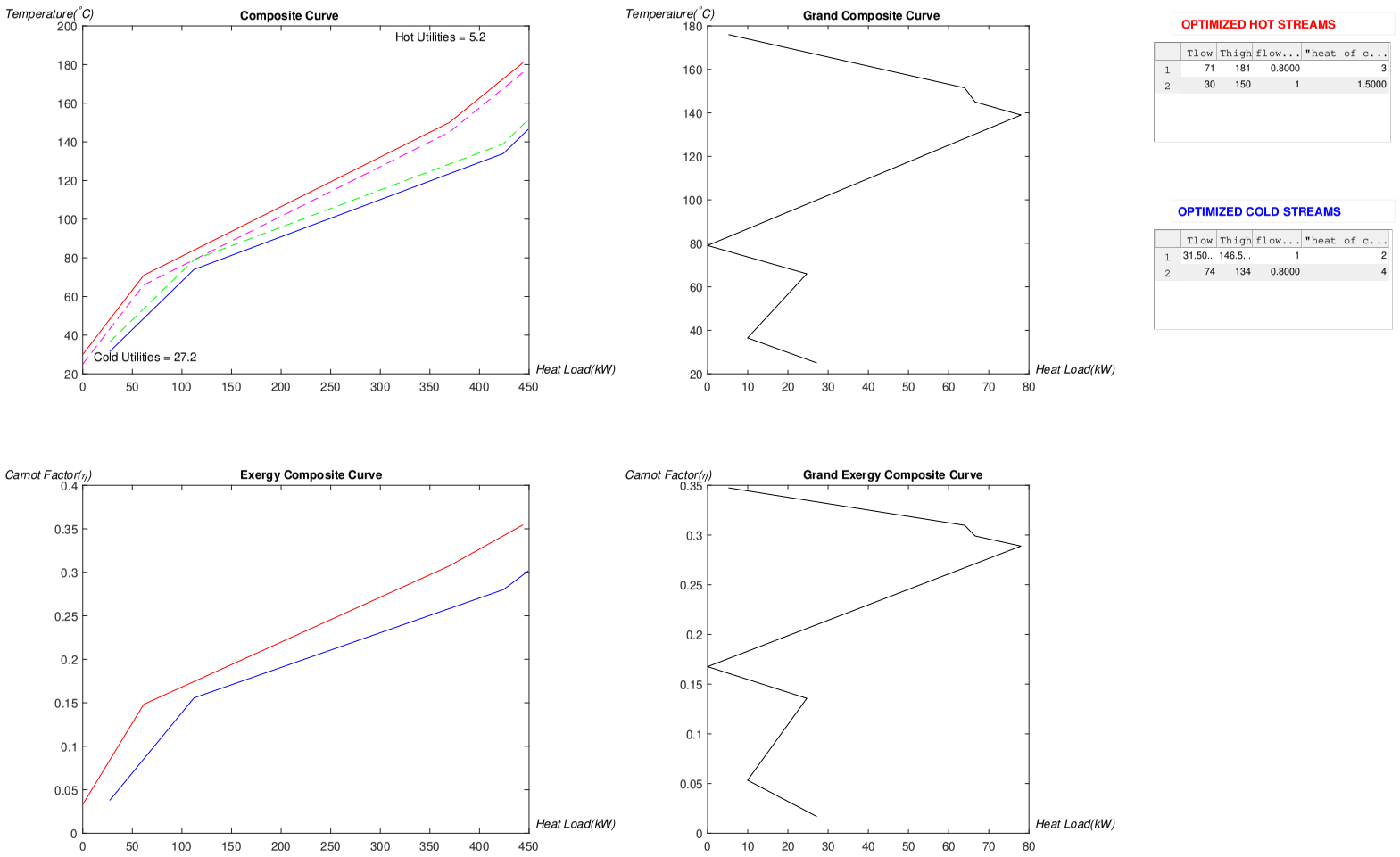


Figure 4.10: Pinch Analysis GUI-5

4.2 Benchmark with Other Software

In this section results from PCSTET are benchmarked against results from other available tools. Note that PCSTET, including the methodology employed, builds-on rather than replicates available tools, and therefore while the results for PCSTET and the available tools are expected to be consistent, they are not expected to match perfectly.

An online tool designed by Umbach [17] is used here in order to benchmark the pinch analysis part of the PCSTET. This online tool demands the user to decide how to divide streams. Then for the matches, provides alternatives to establish. Comparing PCSTET to Umbach's tool, the superiority of PCSTET is that the streams are divided automatically while the superiority of Umbach's tool is that no unnecessary HEXs are created. Also, for Umbach's tool the user may match streams as desired, then go back and change or cancel matches. However, this requires advanced understanding and experience about the pinch concept.

For the given set CP criteria (Equation 3.1), is satisfied above the pinch temperature for the sources and the sinks. However, below the pinch, 25% split of the third stream is needed to have CP values greater or equal to sink streams'. Then, by clicking on every source stream, possible alternatives are suggested by Umbach's tool. Finally HEXes are constructed as they are shown in Table 4.11 and Table 4.12.

Table 4.11: Heat Exchanger Information Below Pinch-Online Tool

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	3	20	80	0.250	2.0	-30
	1	80	90	1.000	3.0	30
2	3	20	80	0.750	2.0	-90
	2	30	90	0.500	3.0	90

Table 4.12: Heat Exchanger Information Above Pinch-Online Tool

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	80	140	1.000	4	-240
	1	90	170	1.000	3	240
2	3	80	125	1.000	2	-90
	2	90	150	1.000	1.5	90

Umbach's tool provides an output file with '.pdf' extension. This file includes stream information as source and target temperatures (T_{low} and T_{high} for PCSTET), heat loads and thermal capacity, portions of streams above and below pinch, CCs, GCC, HDN and HEX information as inlet, outlet temperatures and HEX load. If user has sophisticated understanding about pinch. Umbach's tool can be used for any set of inputs with the user's guidance.

4.3 Exergy Benchmarking

Besides energy and HDN, it is also essential to validate the exergetic aspect of the tool. For this purpose, Linnhoff's study will be used [2]. The study uses example problem data consisting of the two hot and two cold streams provided in Table 4.13. ΔT_{min} is considered to be $10°C$

Table 4.13: Stream information adopted from Linnhoff [2]

Stream Number	T_{low} (°C)	T_{high} (°C)	$CP(MW/K)$
1(hot)	80	245	0.10
2(hot)	105	180	0.30
3(cold)	70	120	0.05
4(cold)	50	215	0.20

The exergy destruction is calculated by the area between exergy composite curves in PCSTET. In one of the oldest and most cited studies by Linnhoff and Dhole [26], it is suggested that exergy destruction is proportional to exergy destruction with HEN including utilities when only streams are drawn in T-H diagram but not the utilities. Actually, exergy destruction and the area under GECC are equal when utilities are not taken into account.

Linnoff's study[2] suggested an exergy destruction value of 2.72MW and the value that is found with PCSTET is 2.29 MW, which corresponds to 15.7% discrepancy. Reasons for this difference are as follows. In PCSTET, the utilities and HDN parts are modeled separately and exergy calculations for utilites are completed considering a CST system. This does not mean PCSTET or Linnhoff's study is wrong. This comparison is provided to give an insight about the values.

Thus, in order to benchmark the exergy destruction calculation is narrowed down to HDN and data points for HDN (Figure 4.11) provided in Table 4.14. By the use of trapezoidal rule same value of 2.29MW is calculated manually for the exergy destruction, showing perfect agreement between PCSTET and Linhoff.

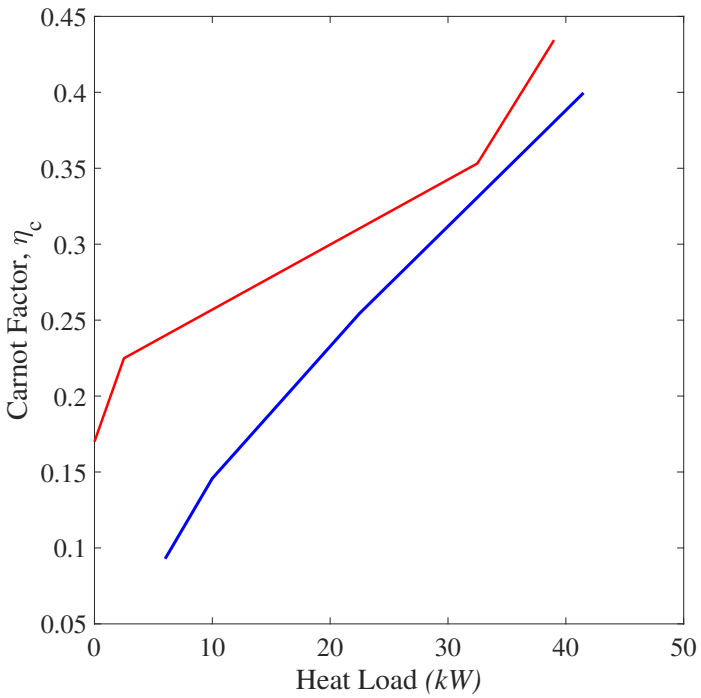


Figure 4.11: ECC

Table 4.14: Energy Level (Carnot Factor) and Heat Load Values for cold and hot composites

Hot		Cold	
$H_h(MW)$	η_h	$H_h(MW)$	η_c
0	0.1700	6	0.0929
2.5	0.2249	10	0.1458
30	0.3532	18.5	0.2545
6.5	0.4344	25	0.3996

4.4 Case Study

Bjork and Westrelund suggest a case presented in Table 4.15 [27]with a ΔT_{min} value of $5°C$.

Table 4.15: Case Study Stream Information

Stream Number	T_{low} ($°C$)	T_{high} ($°C$)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$
1(hot)	75	180	15	2
2(hot)	60	240	20	2
3(cold)	40	230	10	2
4(cold)	120	260	7.5	2
5(cold)	40	130	12.5	2
6(cold)	80	190	10	2

This case is run in PCSTET with the minimum temperature difference doubled, since PCSTET draws a better network for high ΔT_{min} values. In this section, only inputs and outputs are provided. CC, GCC, ECC and GECC diagrams are presented in Pinch GUI (Figure 4.12). HDNs for below and above pinch are presented in Figure 4.13

and Figure 4.14. Temperature continuum with streams are presented with restricted streams by checkboxes in Figure 4.15. Stream information, CC, GCC, ECC and GECC diagrams are provided with Figure 4.16. Discussion about results is presented in section 5.1.

Remaining inputs have minor alterations from 4 stream example (section 4.1).

- Direct normal insolation is updated as $200W/m^2$.
- Collector area is reduced to be $2m^2$
- Dead state temperature remains same value of, $20°C$
- ΔT_{min} is $10°C$.

Pinch analysis locates pinch temperature as $T_{pinch} = 175°C$. The amount of external cooling and heating required are calculated as $H_{cu} = 325kW$ and $H_{hu} = 550kW$ respectively.

HEXs located above and below pinch, obtained from PCSTET are presented in Table 4.16 and Table 4.17.

Hot and cold utilities for original data of case study are also presented as Table 4.18 and Table 4.19.

Table 4.16: Heat Exchanger Information Below Pinch for Case Study

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	139	170	10	2	-620
	1	149	180	10	2	620
2	6	139	170	5	2	-310
	1	149	180	5	2	310
3	6	139	170	5	2	-310
	2	149	180	5	2	310
4	4	139	170	7.5	2	-465
	2	149	180	7.5	2	465
5	5	120	139	7.5	2	-285
	2	161	180	7.5	2	285
6	3	127	139	7.5	2	-180
	2	149	161	7.5	2	180
7	6	120	139	10	2	-380
	1	130	149	10	2	380
8	5	120	139	5	2	-190
	1	130	149	5	2	190
9	3	120	139	2.5	2	-95
	2	130	149	2.5	2	95
10	4	120	139	7.5	2	-285
	2	130	149	7.5	2	285
11	3	120	127	7.5	2	-105
	2	142	149	7.5	2	105
12	5	101	120	2.5	2	-95
	2	130	149	2.5	2	95
13	5	108	120	7.5	2	-180
	2	130	142	7.5	2	180

Continued on next page

Table 4.16 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
14	3	80	120	10	2	-800
	1	90	130	10	2	800
15	6	80	120	5	2	-400
	1	90	130	5	2	400
16	6	80	120	5	2	-400
	2	90	130	5	2	400
17	5	80	120	2.5	2	-200
	2	90	130	2.5	2	200
18	5	80	108	7.5	2	-420
	2	102	130	7.5	2	420
19	5	80	101	2.5	2	-105
	2	109	130	2.5	2	105
20	5	65	80	2.5	2	-75
	2	115	130	2.5	2	75
21	5	65	80	2.5	2	-75
	2	100	115	2.5	2	75
22	3	65	80	2.5	2	-75
	2	94	109	2.5	2	75
23	3	68	80	7.5	2	-180
	2	90	102	7.5	2	180
24	5	40	80	2.5	2	-200
	2	60	100	2.5	2	200
25	5	76	80	2.5	2	-20
	2	90	94	2.5	2	20
26	5	65	80	2.5	2	-75
	1	75	90	2.5	2	75
27	5	65	76	2.5	2	-55

Continued on next page

Table 4.16 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	1	79	90	2.5	2	55
28	3	65	68	7.5	2	-45
	1	87	90	7.5	2	45
29	3	50	65	2.5	2	-75
	1	75	90	2.5	2	75
30	5	50	65	10	2	-300
	2	75	90	10	2	300
31	3	50	65	7.5	2	-225
	2	75	90	7.5	2	225
32	3	40	50	7.5	2	-150
	1	77	87	7.5	2	150
33	5	46	50	2.5	2	-20
	1	75	79	2.5	2	20
34	5	48	50	7.5	2	-30
	1	75	77	7.5	2	30
35	3	40	50	2.5	2	-50
	2	65	75	2.5	2	50
36	5	40	48	7.5	2	-120
	2	67	75	7.5	2	120
37	5	40	46	2.5	2	-30
	2	69	75	2.5	2	30

Table 4.17: Heat Exchanger Information Above Pinch for Case Study

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	170	190	7.5	2	-300
	2	180	200	7.5	2	300
2	3	170	230	10	2	-1200
	2	180	240	10	2	1200
3	6	170	190	2.5	2	-100
	2	180	200	2.5	2	100
4	6	170	190	7.5	2	-300
	2	200	220	7.5	2	300
5	4	190	230	2.5	2	-200
	2	200	240	2.5	2	200
6	4	190	210	5	2	-200
	2	220	240	5	2	200
7	4	210	230	2.5	2	-100
	2	220	240	2.5	2	100

Table 4.18: Case Study: Streams to be cooled by cold utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	2	60	69	2.5	2	-45
2	2	60	75	5	2	-150
3	2	60	67	7.5	2	-105
4	2	60	65	2.5	2	-25

Table 4.19: Case Study: Streams to be heated by hot Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	230	260	2.5	2	150
2	4	210	260	2.5	2	250
3	4	230	260	2.5	2	150

The optimization function of PCSTET with the restriction of first hot, first cold and fourth cold streams' variation, brings out the following modified input set(Table 4.20).

Table 4.20: Case Study: Optimized Stream Information

Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$
1(hot)	75	180	15	2
2(hot)	60	240	20	2
3(cold)	40	230	10	2
4(cold)	134	274	7.5	2
5(cold)	49.9	148.9	15	2
6(cold)	80	190	12	2

By using the modified inputs at the Table 4.20, Pinch Analysis, Create HDN and Exergy Calculate functions are called again. As a result, HEXs are created as presented in Table 4.21.

Table 4.21: Case Study: Heat Exchanger Information Below Pinch for modified inputs, PCSTET

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	148.9	170	10	2	-422
	1	158.9	180	10	2	422
2	6	148.9	170	5	2	-211
	1	158.9	180	5	2	211
3	6	148.9	170	5	2	-211
	2	158.9	180	5	2	211
4	4	148.9	170	7.5	2	-316.5
	2	158.9	180	7.5	2	316.5
5	5	134	148.9	7.5	2	-223.5
	2	165.1	180	7.5	2	223.5
6	3	142.7	148.9	7.5	2	-93
	2	158.9	165.1	7.5	2	93
7	5	134	148.9	7.5	2	-223.5
	1	144	158.9	7.5	2	223.5
8	6	134	148.9	7.5	2	-223.5
	1	144	158.9	7.5	2	223.5
9	3	134	148.9	2.5	2	-74.5
	2	144	158.9	2.5	2	74.5
10	6	134	148.9	2.5	2	-74.5
	2	144	158.9	2.5	2	74.5
11	4	134	148.9	7.5	2	-223.5
	2	144	158.9	7.5	2	223.5
12	3	134	142.7	7.5	2	-130.5
	2	150.2	158.9	7.5	2	130.5
13	5	127.8	134	7.5	2	-93

Continued on next page

Table 4.21 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	2	144	150.2	7.5	2	93
14	3	80	134	10	2	-1080
	1	90	144	10	2	1080
15	5	80	134	5	2	-540
	1	90	144	5	2	540
16	6	80	134	10	2	-1080
	2	90	144	10	2	1080
17	5	80	134	2.5	2	-270
	2	90	144	2.5	2	270
18	5	80	127.8	7.5	2	-717
	2	96.2	144	7.5	2	717
19	5	73.8	80	7.5	2	-93
	2	90	96.2	7.5	2	93
20	3	65	80	10	2	-300
	1	75	90	10	2	300
21	5	65	80	5	2	-150
	1	75	90	5	2	150
22	5	65	80	2.5	2	-75
	2	75	90	2.5	2	75
23	5	65	73.8	7.5	2	-132
	2	81.2	90	7.5	2	132
24	3	50	65	10	2	-300
	2	75	90	10	2	300
25	5	58.8	65	7.5	2	-93
	2	75	81.2	7.5	2	93
26	5	50	65	7.5	2	-225
	2	60	75	7.5	2	225

Continued on next page

Table 4.21 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
27	5	50	58.8	7.5	2	-132
	2	66.2	75	7.5	2	132
28	3	49.9	50	5	2	-1
	2	74.9	75	5	2	1
29	3	49.9	50	5	2	-1
	2	74.8	74.9	5	2	1
30	5	49.9	50	5	2	-1
	2	74.7	74.8	5	2	1
31	5	49.9	50	5	2	-1
	2	74.6	74.7	5	2	1
32	5	49.9	50	5	2	-1
	2	74.5	74.6	5	2	1
33	3	40	49.9	5	2	-99
	2	64.6	74.5	5	2	99
34	3	43.7	49.9	5	2	-62
	2	60	66.2	5	2	62
35	3	40	43.7	2.5	2	-18.5
	2	62.5	66.2	2.5	2	18.5
36	3	40	43.7	2.5	2	-18.5
	2	60.9	64.6	2.5	2	18.5

HDN information above the pinch turned out to be the same for original and modified inputs. Even though the second hot stream is not chosen as a restricted one, the optimum solution uses the unaltered version of it. So hot streams do not changed at all. As a result, HDN above pinch does not change. Besides above pinch cold streams extends further with increased flow rate, which easily can be compensated by changing utilities (Table 4.22). Change is reflected to hot utilities (Table 4.23).

Table 4.22: Case Study: Streams to be cooled by cold Utilities for modified inputs

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	2	60	60.9	2.5	2	-4.5
2	2	60	64.6	2.5	2	-23
3	2	60	62.5	2.5	2	-12.5

Table 4.23: Case Study: Hot Utilities for modified inputs

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	230	274	2.5	2	220
2	4	210	274	2.5	2	320
3	4	230	274	2.5	2	220

To sum up the results of case study, optimization brings out;

- Pinch temperature of $T_{pinch} = 175°C$
- External cooling of $H_{cu} = 40kW$, which means 87.69% decrease in cold utility requirement
- External heating of $H_{hu} = 760kW$, demonstrates 38.18% increase in hot utility requirement
- 47.7% decrease in exergy destruction.

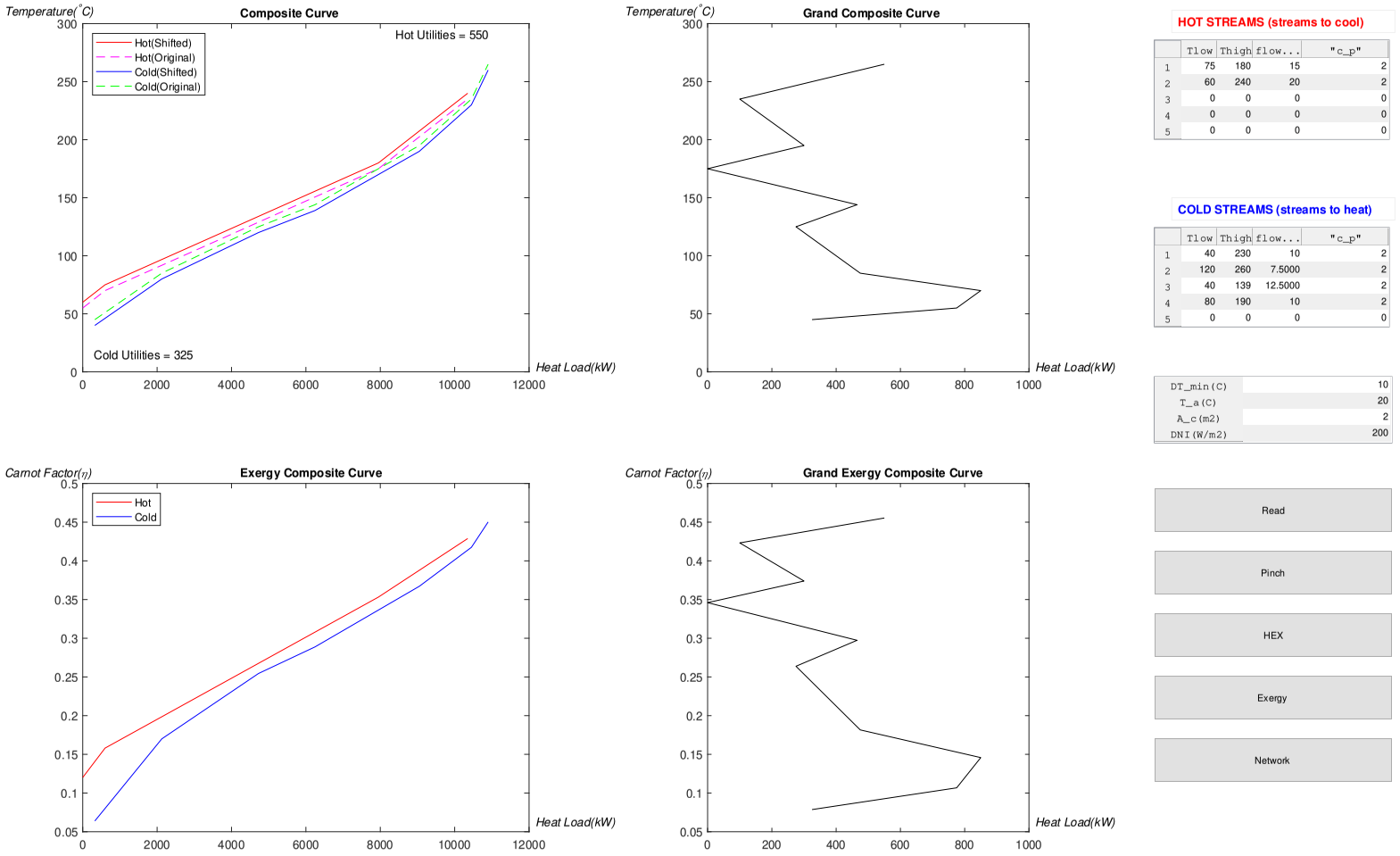


Figure 4.12: Case Study: Pinch Analysis GUI-5

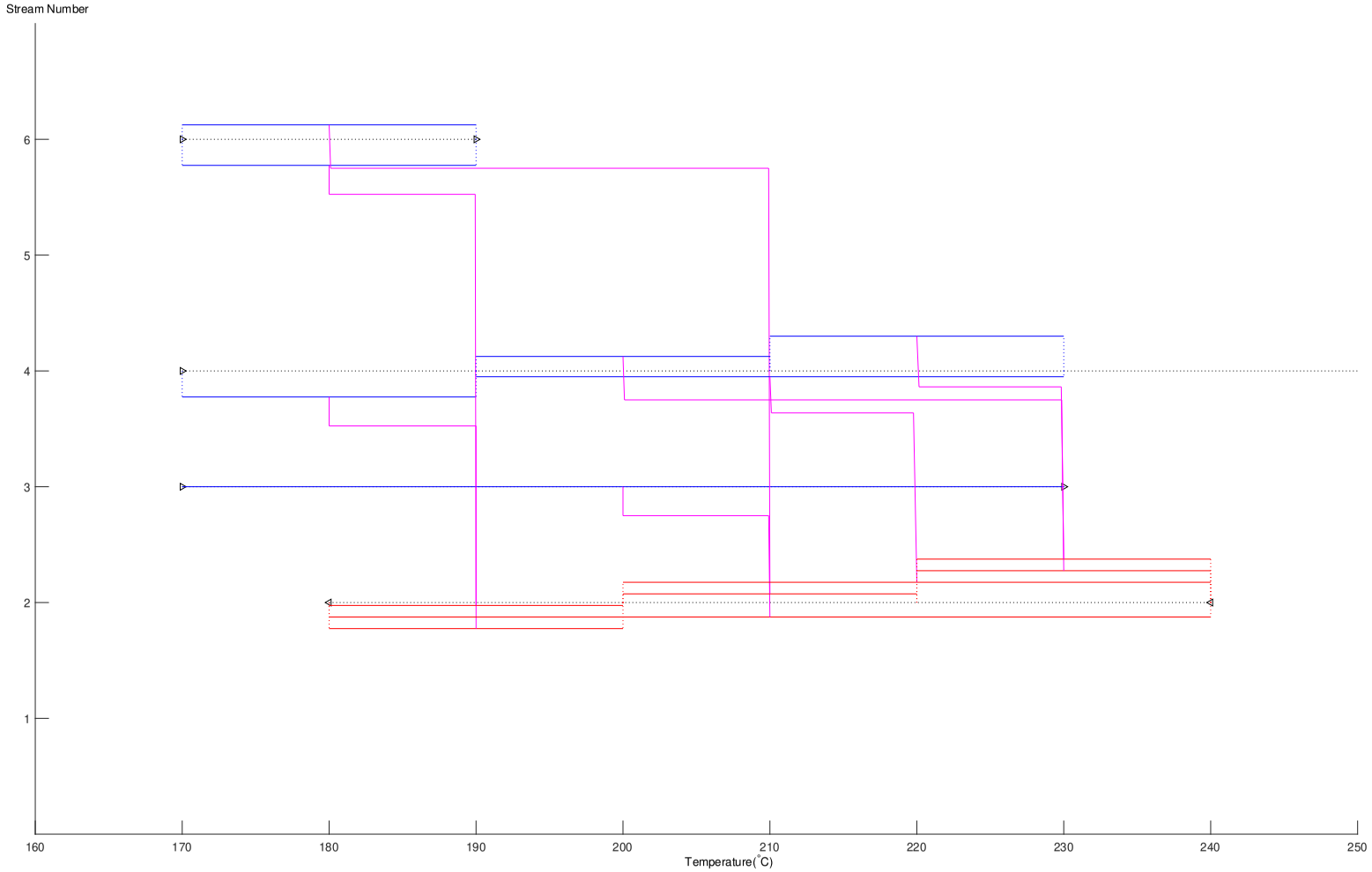


Figure 4.13: Case Study: Above Pinch Network Grid GUI

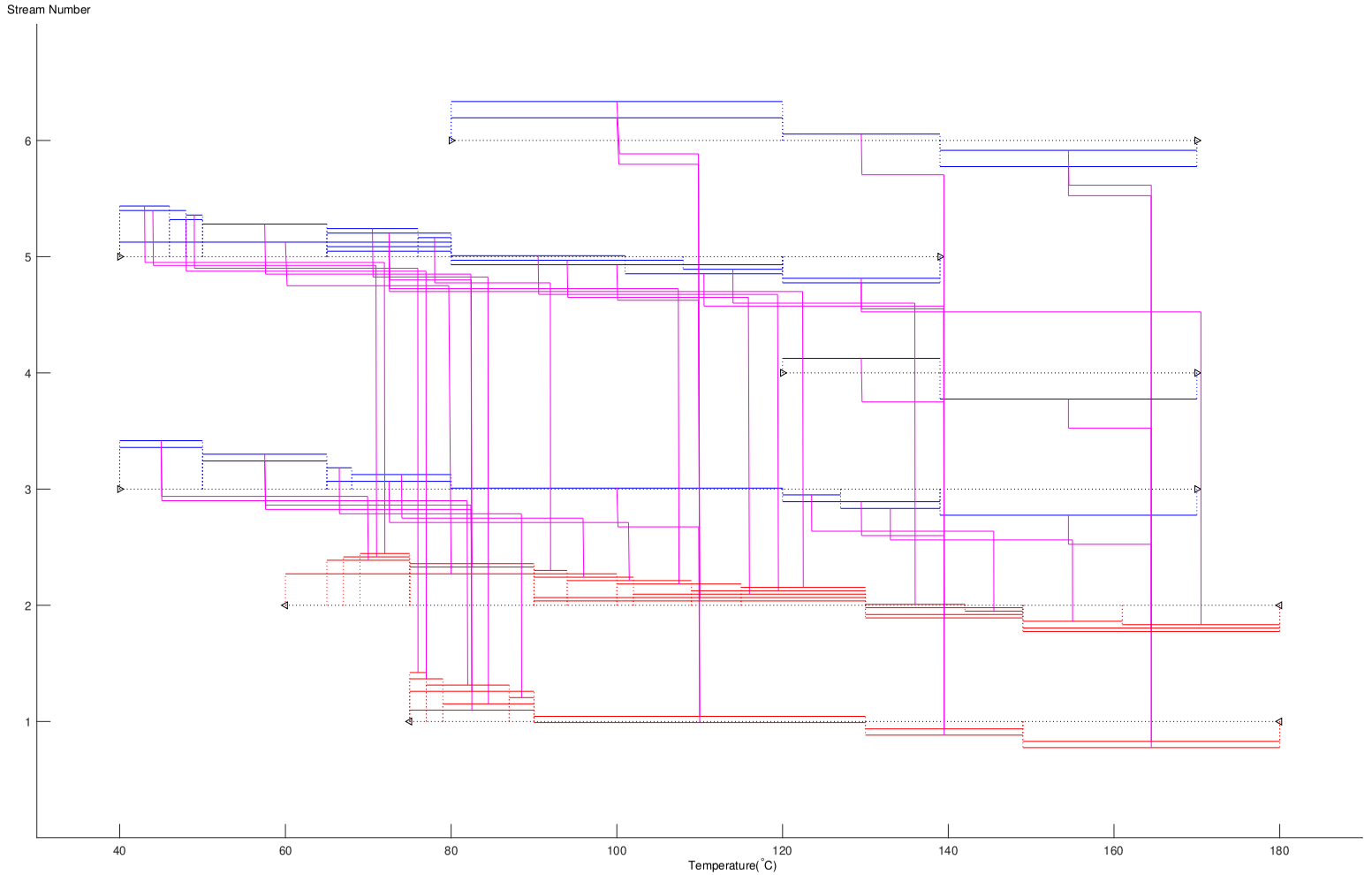


Figure 4.14: Case Study: Below Pinch Network Grid GUI

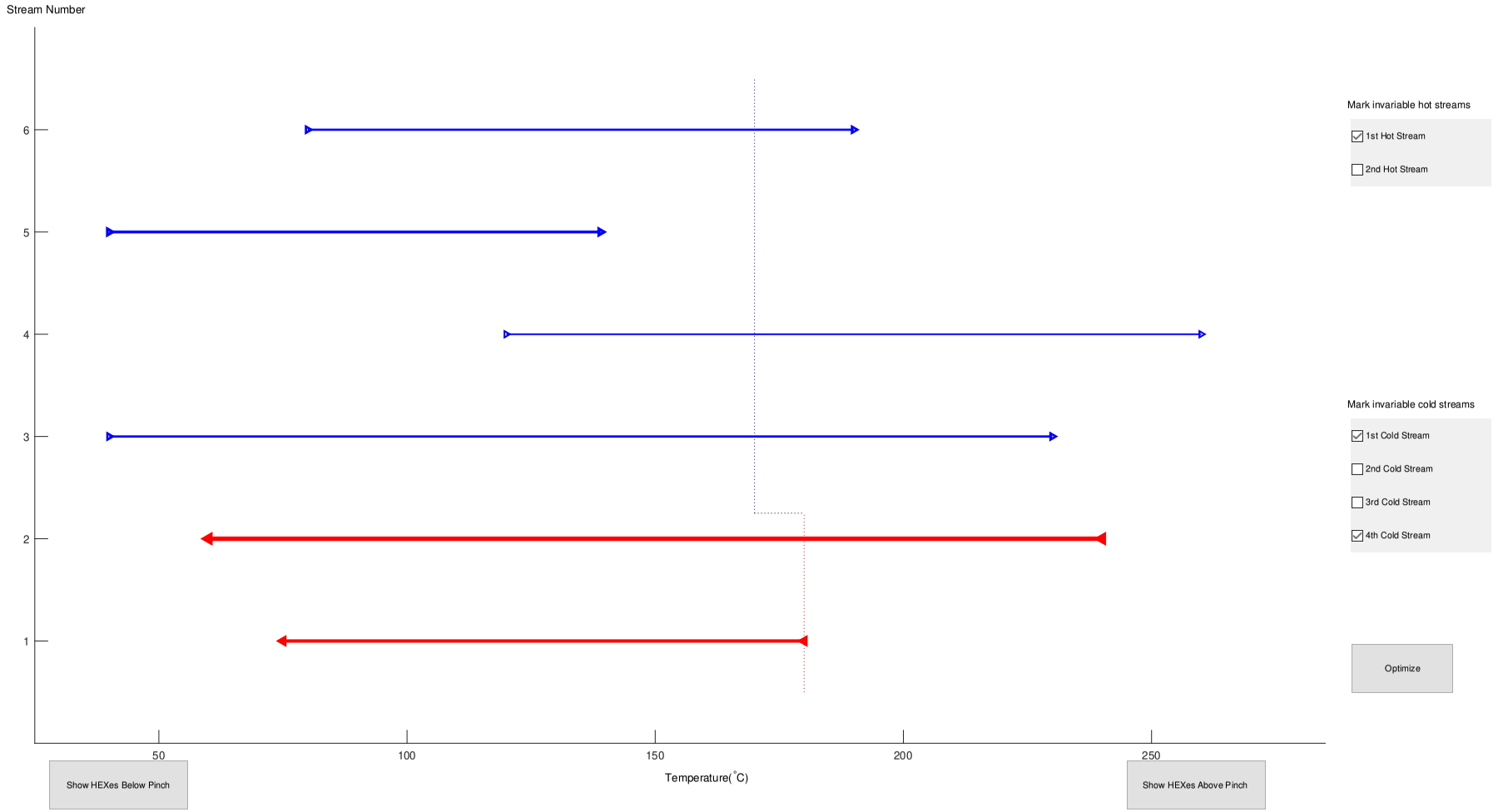
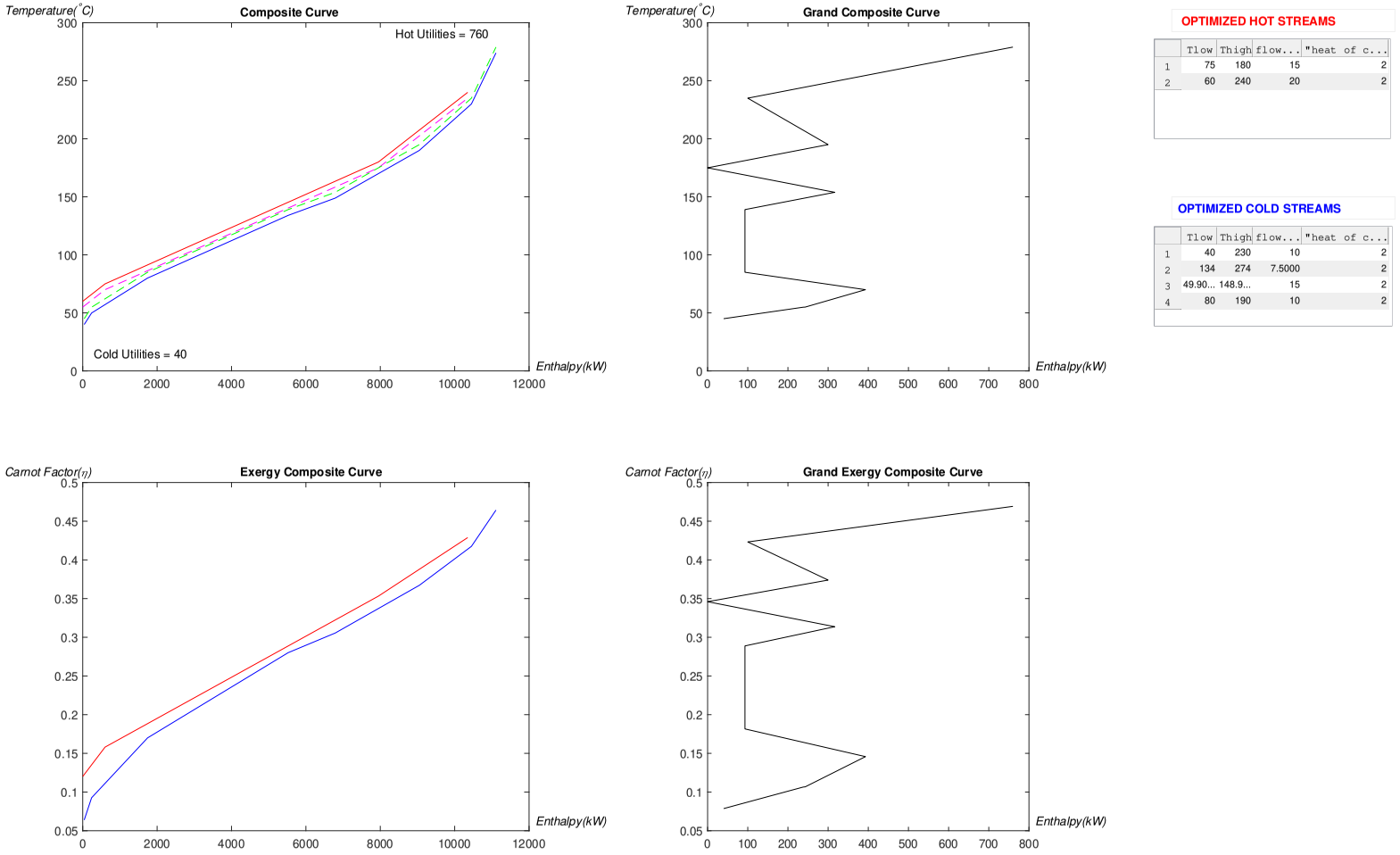


Figure 4.15: Case Study: Network Grid GUI-2



OPTIMIZED HOT STREAMS

	Tlow	Thigh	flow...	"heat of c...
1	75	180	15	2
2	60	240	20	2

OPTIMIZED COLD STREAMS

	Tlow	Thigh	flow...	"heat of c...
1	40	230	10	2
2	134	274	7.5000	2
3	49.90...	148.9...	15	2
4	80	190	10	2

Figure 4.16: Pinch Analysis GUI-5

4.5 Relationship Between Exergy Destruction Reduction and Number of HEXs

In order to learn about the relationship between exergy destruction reduction and the number of HEXs or utilities, an additional and temporary section was added PCSTET. That means this section is not included in the source code supplied in section A.1. As inputs, the set in section 4.4 is used (Table 4.15). Smallest five exergy destruction values are presented with number of HEX and number of utilities (Table 4.24). There is not a correlation between exergy destruction and number of HEXs as presented in the table.

Table 4.24: Number of HEX and Utilities for Different Amount of Exergy Destructions, Case Study

Exergy Destruction Reduction	Number of HEXs		Number of Utilities	
	Below Pinch	Above Pinch	Hot	Cold
0(original)	37	7	3	4
8.73%	18	27	3	5
13.72%	19	30	6	5
18.23%	10	33	5	2
31.83%	18	21	3	5
41.33%	10	24	5	2

4.6 Effect of Solar Thermal Input on PCSTET

In order to reveal the effect of solar thermal input on the performance of PCSTET, the DNI and collector area inputs of case study (Table 4.15) are varied. The amount of DNI is increased to $1000W/m^2$ and collector area is increased to $550m^2$. This resulting solar thermal input is the same value of hot utility requirement calculated in the pinch analysis. Through optimization with the same restrictions of invariable first hot, first cold and fourth cold streams 29.36% decrease in exergy destruction is observed.

CHAPTER 5

CONCLUSIONS

5.1 Discussion

This research is started with strong motivation and demand from one of the EU Horizon 2020 projects, namely *INSHIP*. One of the many work packages that METU is involved in is in the broad umbrella named "Hybrid energy systems and emerging process technologies". The present research is strongly aligned with Task 5.4 "Industrial Parks and Heat Distribution Networks" within this work package.Within Task 5.4. identifying critical aspects of hybrid systems for energy supply incorporated with resource strategies and characterization of these systems is chosen as the specific area to research.

The combination of pinch and exergy analyses is an emerging research field, as the strength of each method compensates for the weakness of the other method. Pinch analysis provides an understanding of how much can be improved in terms of energy (and exergy) while exergy pinpoints the locations to improve. The existing design tools that for pinch analyses are closed, and therefore they cannot be modified to integrate additional concepts, such as exergy or inclusion of CST. The problem statement for the current research is "Design of a HDN for a group of hot and cold streams with a CST system, whose energy and exergy output is predefined, according to pinch and exergy analyses", and the complete methodology for the solution of this problem is provided.

The tool benefits from open literature as well as it contributes to the open literature. About one-sixth of the coding is found online at Mathworks' file exchange, and the remaining is coded by the author for the open literature. The code will be shared with

online society as well as in section A.1 of this thesis. Building on what is found, all of the sub-steps are organized and presented in a way it can be projected to any other coding language or environment. The functions that are created are original but not unique. Presented flow charts explain these functions.

The amount of utilities required and exergy destruction are the two main operations, these are benchmarked with published and online values, and good agreement is found [17].

A case study from literature([27]) is modeled with PCSTET. Optimization results in 87.69% and 38.18% reduction in cold and hot utility demands without any change in pinch temperature. These changes are resulted from the changes in stream information by optimization. Exergy destruction is decreased by 47.7%. These numbers indicate noticeable performance improvement. PCSTET provides the stream information with the optimum data set. Designer should adjust original inputs in order to obtain optimum inputs. If it is not possible to reach optimum set that is suggested by PCSTET, designer should try to alter original inputs as close as possible to optimum set. Then designer should use the capabilities of PCSTET to see the performance of altered set. This way designer can have increased confidence about the performance of final data set.

From all of the alternatives for the case study(section 4.4), the five data sets with lowest exergy destruction are presented(section 4.5) and it is seen that there is no correlation between number of HEXs and the amount of exergy destruction reduction. The effect of changing CST input is also controlled with same set of inputs. Increasing CST energy input in a way that it corresponds to the exact value of hot utility requirement resulted in less exergy destruction reduction.

PCSTET provides insight into the complete optimized solution (section 1.5). As explained in the section 4.1, the outputs of PCSTET are highly valuable for a HDN designer. Starting from HDN design provided from PCSTET an expert designer is able to obtain a robust HDN design.

Overall, PCSTET is designed to be a very user-friendly and beneficial tool. GUI's and plots are designed neatly. Building blocks are created in the most organized way

possible so that, if any of the models are desired to be improved, modification of related function is enough.

The last but most important contribution of the detailed study is the explanation of the software pinch solution. It should be noted that the methodology presented here is also beneficial for non-software solutions. Often manual solutions randomly choose streams to match if they satisfy CP and ΔT_{min} conditions. HEXs are built one by one. If the solution comes to a point at which no matches are possible, a solution is sought by going a few steps backward and changing match decisions. Thus, the flow of work explained in chapter 3 may be beneficial for non-software solutions also.

5.2 Future Work

This tool and all of the methodology will provide a solid basis for a commercial or non-commercial tool for HDN build with combined pinch and energy considerations. The scope (subsection 1.5.2) indicates areas where the tool can be further improved, including the following:

- The first limitation is the inability to model phase change. In reality, other than the creation of HDN, all other functions are compatible with phase change. However, the "Create HDN" function can not deplete sources if phase change occurs.
- Second restraint is the requirement of at least one utility at both ends as below and above pinch. The negative effect of this limitation shows itself similar to the previous limitation, which is non-empty source arrays.
- The third possible improvement is related to solar modeling. The focus in the current work is on modeling the HDN and a relatively simple CST model is employed. PCSTET can be strengthened by expanding the CST model to account for variations in CST output with variations in location, time of day, day of year, and collector specifications.
- Fourth aspect that may be improved is related to the number of HEXs for recovery and the number of devices as external means. By analyzing the output

spreadsheets for heat exchangers and utilities, it can be easily realized there are several utilities covering the same temperature interval for the same streams. Also, for a couple of HEXs, it may be easily recognized that they include the same streams, and combining them does not violate CP or ΔT_{min} criteria. Current HEX and utility outputs of PCSTET can be improved. In order to improve on PCSTET, loops and Euler's General Network Theorem should be studied and implemented into the model. As a result, those extra HEXs, heaters, or coolers might be eliminated.

REFERENCES

[1] B. Linnhoff and J. R. Flower, "Synthesis of heat exchanger networks: I. Systematic generation of energy optimal networks," *AIChE Journal*, vol. 24, no. 4, pp. 633–642, 1978.

[2] B. Linnhoff, "Pinch Technology for the Synthesis of Optimal Heat and Power Systems," *Journal of Energy Resources Technology*, vol. 111, no. 3, p. 137, 1989.

[3] G. Phetteplace, "Introduction," in *District Heating*, ch. 1, pp. 2–5, 2013.

[4] B. Linnhoff and E. Hindmarsh, "The pinch design for the heat exchanger networks," *Chemical Engineering Science*, vol. 38, no. 5, pp. 745–763, 1982.

[5] H. Tyagi, A. K. Agarwal, P. R. Chakraborty, and S. Powar, *Advances in solar energy technology*, vol. 6. 2005.

[6] J. Vidová, "Industrial parks - history, present and its influence to the employment," *Review of Economic Perspectives*, vol. 10, no. 1, pp. 41–58, 2010.

[7] C. Winterscheid, J. O. Dalenbäck, and S. Holler, "Integration of solar thermal systems in existing district heating systems," *Energy*, vol. 137, pp. 579–585, 2017.

[8] S. A. Kalogirou, *Solar thermal collectors and applications*, vol. 30. 2004.

[9] Pedro Horta (FhG ISE), "Process Heat Collectors: State of the Art and available medium temperature collectors," Tech. Rep. Technical Report A.1.3, IEA SHC Task 49, 2016.

[10] A. Bejan, *Advanced Engineering Thermodynamics*. Durham,North Carolina: John Wiley & Sons, third ed., 2006.

[11] R. K. Shah, "Fundamentals of heat technology," in *Metallurgist*, vol. 4, ch. 1, pp. 1–74, 2004.

[12] K. L. Coulson, "Polarization of Light in the Atmosphere," in *Solar and Terrestrial Radiation,Methods and Measurements*, no. 1950, ch. 7, pp. 178–214, Academic Press, 1988.

[13] R. Petela, "Exergy of Heat Radiation," *Journal of Heat Transfer*, vol. 86, no. 2, p. 187, 1964.

[14] "INSHIP - Integrating National Research Agendas on Solar Heat for Industrial Processes."

[15] "Solar Heating and Cooling Programme, Task 49," 2016.

[16] Daniel Declercq, "PINCHCO," 2005.

[17] Jeffrey S. Umbach, "Online Pinch Analysis Tool," 2010.

[18] ProSim, "Simulis Pinch," 2014.

[19] X. Feng and X. Zhu, "Combining pinch and exergy analysis for process modifications," *Applied Thermal Engineering*, vol. 17, no. 3, pp. 249–261, 1997.

[20] R. Petela, "Exergy of undiluted thermal radiation," *Solar Energy*, vol. 74, no. 6, pp. 469–488, 2003.

[21] E. Bellos, C. Tzivanidis, K. A. Antonopoulos, and I. Daniil, "The use of gas working fluids in parabolic trough collectors – An energetic and exergetic analysis," *Applied Thermal Engineering*, vol. 109, pp. 1–14, 2016.

[22] C. Shuqing, *Integration of exergy analysis and pinch technology*. Requirements for the degree of doctor of philosphy, Marquette University, 1995.

[23] Mathworks, "File Exchange," 2018.

[24] S. Rowe, "Pinch," 2018.

[25] E. Bellos and C. Tzivanidis, "A detailed exergetic analysis of parabolic trough collectors," *Energy Conversion and Management*, vol. 149, pp. 275–292, 2017.

[26] B. Linnhoff and V. R. Dhole, "Shaftwork targets for low-temperature process design," *Chemical Engineering Science*, vol. 47, no. 8, pp. 2081–2091, 1992.

[27] K. M. Björk and T. Westerlund, "Global optimization of heat exchanger network synthesis problems with and without the isothermal mixing assumption," *Computers and Chemical Engineering*, vol. 26, no. 11, pp. 1581–1593, 2002.

APPENDIX A

MATLAB SOURCE CODE

A.1 PCSTET

About quarter of the code provided below is written by Scott Rowesection 3.1. The added sections are pointed out with comments.

```
classdef exergy_pinch_HDN < handle
    properties
    pinchFigure = {};
    pinchButton = {};
    coldTable = {};
    coldText = {};
    hotTable = {};
    hotText = {};
    pinchAxes = {};
    Gcc_Axes = {};
    hotStreams = {};
    coldStreams = {};
    hotBreaks = [];
    coldBreaks = [];
    hotEnergies = [];
    coldEnergies = [];
    allBreaks = [];
    allEnergies = [];
    allHeats = [];
    allColds = [];
    cascade = [];
    hotCoordinates = [];
    coldCoordinates = [];
    coldDuty = [];
    hotDuty = [];
    phaseText = [];
%------------------------------MURAT_Adds;-------------------------
    howaccurate = [];
    Tpinch = [];
    Thalve = [];
    Highest_T = [];
    Lowest_T = [];
    streams_cold_fromtable = [];
    streams_hot_fromtable = [];
    hotinfo = [];
    coldinfo = [];
    StreamInfo = [];
    HotStreams_BelowPinch = [];
    HotStreams_AbovePinch = [];
    ColdStreams_BelowPinch = [];
    ColdStreams_AbovePinch = [];
    Hexes_AbovePinch = [];
    Hexes_BelowPinch = [];
    hotutilities = [];
    coldutilities = [];
    NetworkBelowWithHexFigure = {};
    NetworkBelowWithHexAxes = {};
    NetworkAboveWithHexFigure = {};
    NetworkAboveWithHexAxes = {};
    E_level_cold = [];
    E_level_hot = [];
    E_level_grand = [];
```

```
    xlabelpinch = []
    ylabelpinch = []
    xlabelGcc = []
    ylabelGcc = []
    xlabelEcc = []
    ylabelEcc = []
    xlabelGEcc = []
    ylabelGEcc = []
    pinch_title = [];
    Gcc_title = [];
    Ecc_title = [];
    xlabelNetworkGrid = [];
    ylabelNetworkGrid = [];
    GEccAxes = [];
    Gecc_title = [];
    filename = [];
    readUserInputsButton = {};
    HexButton = {};
    exergyButton = {};
    GridButton = {};
    HexBuiltButton_BP = {};
    HexBuiltButton_AP = {};
    OptimizationButton = {};
    checkHS1 = {};
    checkHS2 = {};
    checkHS3 = {};
    checkHS4 = {};
    checkHS5 = {};
    checkCS1 = {};
    checkCS2 = {};
    checkCS3 = {};
    checkCS4 = {};
    checkCS5 = {};
    propertytable = {};
    NetworkGridFigure = {};
    NetworkGridAxes = {};
    HEXTable_BP = {};
    HEXTable_AP = {};
    NOHexBuilt_AP = [];
    NOHexBuilt_BP = [];
    ExergyFigure = {};
    Ecc_Axes = {};
    property_Info = [];
    hot_intCoordinates = [];
    cold_intCoordinates = [];
    E_level_inthot = [];
    E_level_intcold = [];
    exD_4recovery = [];
    exDL = [];
    ex_utilizedbysolarthermal = [];
    ex_solarirradiation = [];
    alternativesforstreams_Tlow = [];
```

```
105     alternativesforstreams_Thigh = [];
106     alternativesforstreams_flowrate = [];
107     alternativesforstreams_all = [];
108     hot_alternative = [];
109     cold_alternative = [];
110     optimizationcheck = [];
111     ex_DL_log = [];
112     ED_min = [];
113     opt_inputs = [];
114     IntrsctX = [];
115     IntrsctY = [];
116     ED_min_initial = [];
117     Percent_Redutction = [];
118     hotcheck_title = [];
119     coldcheck_title = [];
120     modify_check_hot = [];
121     modify_check_cold = [];
122     modify_check = [];
123     main_line_width = [];
124     opt_hots = [];
125     opt_colds = [];
126     FinalFigure = {};
127     hotFinalTable = [];
128     coldFinalTable = [];
129     hotFinalText = [];
130     coldFinalText = [];
131     pinchAxesFinal = [];
132     Gcc_AxesFinal = [];
133     pinch_titleFinal = [];
134     ylabelGEccFinal = [];
135     xlabelGEccFinal = [];
136     ylabelEccFinal = [];
137     xlabelEccFinal = [];
138     ylabelGccFinal = [];
139     xlabelGccFinal = [];
140     ylabelpinchFinal = [];
141     xlabelpinchFinal = [];
142     Gecc_titleFinal = [];
143     Gcc_titleFinal = [];
144     Ecc_AxesFinal = [];
145     Ecc_titleFinal = [];
146     GEccAxesFinal = [];
147     xlabel_AP_NetworkGrid = [];
148     ylabel_AP_NetworkGrid = [];
149     xlabel_BP_NetworkGrid = [];
150     ylabel_BP_NetworkGrid = [];
151
152 %-----------------------------MURAT_Done;----------------------------
153     end
154     methods
155 function obj = exergy_pinch_HDN()
156 clc
```

```
close all
%--------------------------------MURAT_Comments;----------------------
% build the GUI
%     In order;
%     Create Figure
%     Create Hot Streams Block with predefined temperatures
%     Create Cold Streams Block with predefined temperatures
%     Title of hot table
%     Title of cold table
%     Pinch button
%     DT_min box
%     + button
%     - button
%     DT_min title
%     Composite Curve axes
%     Grand Composite curve axes
%     Title
%--------------------------MURAT_Finishes;-----------------
obj.howaccurate=10^(-10);
obj.pinchFigure = figure('name','pinch analysis','numbertitle',...
    'off','ToolBar','none','MenuBar','None','Units','pixels',...
    'Position',[0 50 1600 980],'Resize','off');
obj.hotTable = uitable('Parent',obj.pinchFigure,...
    'Units','normalized',...
            'Data',[75          180         15          2
                    60          240         20          2
                    0           0           0           0
                    0           0           0           0
                    0           0           0           0],...
    'ColumnEditable',[true true true true true],...
    'Position',[1300/1600 810/980 264/1600 112/980],...
    'RowName',{'1','2','3','4','5'},...
    'ColumnName',{'Tlow','Thigh','flowrate','"c_p"'},...
    'ColumnWidth',{40,40,55,97});
obj.coldTable = uitable('Parent',obj.pinchFigure,...
    'Units','normalized',...
            'Data',[40          230         10          2
                    120         260         7.5         2
                    40          139         12.5        2
                    80          190         10          2
                    0           0           0           0],...
    'ColumnEditable',[true true true true true],...
    'Position',[1300/1600 600/980 264/1600 112/980],...
    'RowName',{'1','2','3','4','5'},...
    'ColumnName',{'Tlow','Thigh','flowrate','"c_p"'},...
    'ColumnWidth',{40,40,55,97});
obj.hotText = annotation('textbox',...
    'String','HOT STREAMS (streams to cool)',...
    'Color',[1 0 0],'FontWeight','bold',...
    'Position',[1320/1600 930/980 250/1600 25/980],...
    'Units','normalized','EdgeColor',0.9411*[1 1 1]);
obj.coldText = annotation('textbox',...
```

```
209     'String','COLD STREAMS (streams to heat)',...
210     'Color',[0 0 1],'FontWeight','bold',...
211     'Position',[1320/1600 720/980 250/1600 25/980],...
212     'Units','normalized','EdgeColor',0.9411*[1 1 1]);
213 obj.propertytable = uitable('Parent',obj.pinchFigure,...
214     'Units','normalized','Data',[5
215                                   20
216                                   2
217                                   200],...
218     'ColumnEditable',[true true],...
219     'Position',[1300/1600 470/980 267/1600 75/980],...
220     'RowName',{'DT_min(C)','T_a(C)','A_c(m2)','DNI(W/m2)'},...
221     'ColumnWidth',{165},'ColumnName',([]));
222 obj.readUserInputsButton = uicontrol('Parent',obj.pinchFigure,...
223     'Style','pushbutton','Units','normalized',...
224     'Position',[1300/1600 370/980 267/1600 50/980],...
225     'String','Read','Callback',...
226     @(src,evt)readUserInputs(obj,src,evt),...
227     'BusyAction','cancel', 'Interruptible','off');
228 obj.pinchAxes = axes('Units','normalized',...
229     'Position',[100/1600 550/980 500/1600 390/980],...
230     'Parent',obj.pinchFigure);
231 obj.pinch_title  = annotation('textbox','LineStyle','none',...
232    'String','Composite Curve','FontWeight','bold',...
233    'Position',...
234    [270/1600 925/980 200/1600 40/980],'Units','normalized');
235 obj.Gcc_Axes = axes('Units','normalized',...
236     'Position',[800/1600 550/980 360/1600 390/980],...
237     'Parent',obj.pinchFigure);
238 obj.Gcc_title  = annotation('textbox','LineStyle','none',...
239    'String','Grand Composite Curve','FontWeight','bold',...
240    'Position',[900/1600 925/980 200/1600 40/980],...
241    'Units','normalized');
242 obj.Ecc_Axes = axes('Units','normalized','Position',...
243     [100/1600 35/980 500/1600 390/980],'Parent',obj.pinchFigure);
244 obj.Ecc_title = annotation('textbox','LineStyle','none',...
245    'String','Exergy Composite Curve','FontWeight','bold',...
246    'Position',[270/1600 420/980 200/1600 30/980],...
247    'Units','normalized');
248 obj.GEccAxes = axes('Units','normalized',...
249     'Position',[800/1600 35/980 360/1600 390/980],...
250     'Parent',obj.pinchFigure);
251 obj.Gecc_title = annotation('textbox','LineStyle','none',...
252    'String','Grand Exergy Composite Curve','FontWeight','bold',...
253    'Position',[900/1600 420/980 300/1600 30/980],...
254    'Units','normalized');
255 obj.xlabelpinch = annotation('textbox','LineStyle','none',...
256    'FontAngle','italic','String','Temperature(^{\circ}C)',...
257    'Position',[5/1600 940/980 300/1600 30/980],...
258    'Units','normalized');
259 obj.ylabelpinch = annotation('textbox','LineStyle','none',...
260    'FontAngle','italic','String','Enthalpy(kW)',...
```

```
261     'Position',[600/1600 260/500 400/1600 30/500],...
262     'Units','normalized');
263 obj.xlabelGcc = annotation('textbox','LineStyle','none',...
264     'FontAngle','italic','String','Temperature(^{\circ}C)',...
265     'Position',[700/1600 940/980 300/1600 30/980],...
266     'Units','normalized');
267 obj.ylabelGcc = annotation('textbox','LineStyle','none',...
268     'FontAngle','italic','String','Enthalpy(kW)',...
269     'Position',[1160/1600 260/500 400/1600 30/500],...
270     'Units','normalized');
271 obj.xlabelEcc = annotation('textbox','LineStyle','none',...
272     'FontAngle','italic','String','Carnot Factor(\eta)',...
273     'Position',[5/1600 420/980 300/1600 30/980],...
274     'Units','normalized');
275 obj.ylabelEcc = annotation('textbox','LineStyle','none',...
276     'FontAngle','italic','String','Enthalpy(kW)',...
277     'Position',[600/1600 0.1/500 400/1600 30/500],...
278     'Units','normalized');
279 obj.xlabelGEcc = annotation('textbox','LineStyle','none',...
280     'FontAngle','italic','String','Carnot Factor(\eta)',...
281     'Position',[700/1600 420/980 300/1600 30/980],...
282     'Units','normalized');
283 obj.ylabelGEcc = annotation('textbox','LineStyle','none',...
284     'FontAngle','italic','String','Enthalpy(kW)',...
285     'Position',[1160/1600 0.1/500 400/1600 30/500],...
286     'Units','normalized');
287 obj.pinchButton = uicontrol('Parent',obj.pinchFigure,...
288     'Style','pushbutton','Units','normalized',...
289     'Position', [1300/1600 300/980 267/1600 50/980],...
290     'String','Pinch','Callback',...
291     {@(src,evt)pinchCallBack(obj,src,evt)},...
292     'BusyAction','cancel', 'Interruptible','off',...
293     'Enable','off');
294 obj.HexButton = uicontrol('Parent',obj.pinchFigure,...
295     'Style','pushbutton','Units','normalized',...
296     'Position', [1300/1600 230/980 267/1600 50/980] ,...
297     'String','HEX','Callback',...
298     {@(src,evt)HexCallBack(obj,src,evt)},...
299     'BusyAction','cancel', 'Interruptible','off',...
300     'Enable','off');
301 obj.exergyButton = uicontrol('Parent',obj.pinchFigure,...
302     'Style','pushbutton','Units','normalized',...
303     'Position', [1300/1600 160/980 267/1600 50/980] ,...
304     'String','Exergy','Callback',...
305     {@(src,evt)exergyCallBack(obj,src,evt)},...
306     'BusyAction','cancel', 'Interruptible','off',...
307     'Enable','off');
308 obj.GridButton = uicontrol('Parent',obj.pinchFigure,...
309         'Style','pushbutton','Units','normalized',...
310         'Position',[1300/1600 90/980 267/1600 50/980],...
311         'String','Network',...
312         'Callback',@(src,evt)NetworkGridDrawCallback(obj,src,evt),...
```

```
313         'BusyAction','cancel', 'Interruptible','off',...
314         'Enable','off');
315  print(obj.pinchFigure,'4st_1stepPinchGUI','-depsc2')
316 %--------------------------MURAT_Done;------------------------------
317 end
318 function pinchCallBack(obj,src,evt)
319 pinch_Calculate(obj,obj.streams_hot_fromtable,...
320                             obj.streams_cold_fromtable)
321 EnergyCompositeCurveDraw(obj)
322 obj.HexButton = uicontrol('Parent',obj.pinchFigure,...
323     'Style','pushbutton','Units','normalized',...
324     'Position', [1300/1600 230/980 267/1600 50/980] ,...
325     'String','HEX','Callback',...
326     {@(src,evt)HexCallBack(obj,src,evt)},...
327     'BusyAction','cancel', 'Interruptible','off',...
328     'Enable','on');
329 print(obj.pinchFigure,'4st_3stepPinchGUI','-depsc2')
330 end
331 function pinch_Calculate(obj,hots_Input,colds_Input)
332 resetContainers_Pinch(obj);
333 % Close the figure if exist
334 % composite curve for hot streams
335 [obj.hotStreams obj.hotBreaks obj.hotinfo] =...
336  populateStrms(obj,hots_Input,obj.hotStreams,-1);
337 % reinstate breaks for phase changes
338 obj.hotBreaks = addPhaseChgs(obj,obj.hotBreaks,obj.hotStreams,0);
339 % get the energy drop on each cooling interval along hot stream
340 obj.hotEnergies = ...
341 abs( getEnergies(obj,obj.hotBreaks,obj.hotStreams,0) );
342 obj.hotCoordinates = accumulate(obj,obj.hotEnergies);
343 % composite curve for cold streams
344 [obj.coldStreams obj.coldBreaks obj.coldinfo] =....
345  populateStrms(obj,colds_Input,obj.coldStreams,1);
346 % reinstate breaks for phase changes
347 obj.coldBreaks = ...
348 addPhaseChgs(obj,obj.coldBreaks,obj.coldStreams,0);
349 % get the energy gain on each heating interval along cold stream
350 obj.coldEnergies = ...
351 getEnergies(obj,obj.coldBreaks,obj.coldStreams,0);
352 obj.coldCoordinates = accumulate(obj,obj.coldEnergies);
353
354 %--------------------------------MURAT_Adds;--------------------------
355 %Last row +1 for hot stream -1 for cold stream
356 obj.StreamInfo=[obj.hotinfo 1*(ones(size(obj.hotinfo,1),1))...
357     ;obj.coldinfo -1*(ones(size(obj.coldinfo,1),1))];
358 obj.StreamInfo=[obj.StreamInfo(:,1:4)...
359     transpose(1:(size(obj.StreamInfo,1))) obj.StreamInfo(:,5)];
360 %--------------------------------MURAT_Done;--------------------------
361
362 obj.Thalve = obj.property_Info(1)/2;
363 % allBreaks is for grand composite curve (with obj.Thalve shifts)
364 obj.allBreaks = [ obj.coldBreaks+obj.Thalve
```

```
365     obj.hotBreaks-obj.Thalve ];
366 obj.allBreaks = sort(obj.allBreaks);
367 obj.allBreaks = unique(obj.allBreaks);  % cull redundant breaks
368 % reinstate breaks for stream phase changes
369 obj.allBreaks = ...
370 addPhaseChgs(obj,obj.allBreaks,obj.coldStreams,obj.Thalve);
371 obj.allBreaks = ...
372 addPhaseChgs(obj,obj.allBreaks,obj.hotStreams,-obj.Thalve);
373 obj.allBreaks = clean(obj,obj.allBreaks);   % cleanup allBreaks
374 obj.allHeats = ...
375 getEnergies(obj,obj.allBreaks,obj.hotStreams,-obj.Thalve);
376 obj.allColds = ...
377 getEnergies(obj,obj.allBreaks,obj.coldStreams, obj.Thalve);
378 % energy surpluses and deficits
379 obj.allEnergies = -(obj.allHeats + obj.allColds);
380 % cascade from the hottest temperature
381 obj.allEnergies = [obj.allEnergies(2:end)
382                    obj.allEnergies(1)];
383 obj.allEnergies = flipud(obj.allEnergies);
384 obj.cascade = accumulate(obj,obj.allEnergies);
385 obj.hotDuty = -min(obj.cascade);
386 obj.cascade = obj.cascade + obj.hotDuty;
387 obj.coldDuty = obj.cascade(end);
388 obj.coldEnergies = obj.coldEnergies + obj.coldDuty;
389 obj.cascade = flipud(obj.cascade);
390 min_pinch=obj.cascade(2);
391 for i_6=2:(size(obj.cascade)-1)
392     if obj.cascade(i_6)<=min_pinch
393         obj.Tpinch=obj.allBreaks(i_6);
394         min_pinch=obj.cascade(i_6);
395     end
396 end
397
398 end
399 function EnergyCompositeCurveDraw(obj)
400
401     if ishandle(obj.NetworkGridFigure)
402         close(obj.NetworkGridFigure)
403     end
404
405 cla(obj.pinchAxes);
406 axes(obj.pinchAxes);
407 plot(obj.hotCoordinates,obj.hotBreaks,'r');
408 hold on;
409 plot(obj.hotCoordinates,obj.hotBreaks-obj.Thalve,'--m');
410 plot(obj.coldCoordinates+obj.coldDuty,obj.coldBreaks,'b');
411 plot(obj.coldCoordinates+obj.coldDuty,obj.coldBreaks+obj.Thalve,...
412                                                         '--g');
413 text(0.025,0.05,['Cold Utilities = ',num2str(obj.coldDuty)],...
414                  'Units','normalized');
415 text(0.7,.97,['Hot Utilities = ',num2str(obj.hotDuty)],...
416              'Units','normalized');
```

```
417 legend('Hot(Shifted)','Hot(Original)',...
418        'Cold(Shifted)','Cold(Original)','Location','northwest')
419 cla(obj.Gcc_Axes);
420 axes(obj.Gcc_Axes);
421 plot(obj.cascade,obj.allBreaks,'k');
422
423 % f_1CC=figure('Name','Hot Stream T-H Curve before shift',...
424 %              'NumberTitle','off','Position',[0 0 960 980]);
425 %     plot(obj.hotCoordinates,obj.hotBreaks,'r','LineWidth',2.5);
426 %     xlabel('Enthalpy \it(kW)')
427 %     ylabel('Temperature \it(\circ C)')
428 %     y_label_object = get(gca,'ylabel');
429 %     py = get(y_label_object,'position');
430 %     % get the current position property
431 %     py(1) = 2*py(1) ;
432 %     %double the distance, negative values put the label below the axis
433 %     set(y_label_object,'position',py)
434 %     set(gca,'fontsize',26,'FontName','Times New Roman')
435 % % print(f_1CC,strcat('4st_hotstreamsT-H',,...
436 %datestr(now,'mm_dd_HH_MM'))
437 % %     '-depsc2')
438 % print(f_1CC,'4st_hotstream4st','-depsc2')
439 % f_2CC=figure('Name','Cold Stream T-H Curve before shift',...
440 %              'NumberTitle','off','Position',[0 10 1000 980]);
441 %     plot(obj.coldCoordinates,obj.coldBreaks,'b','LineWidth',2.5);
442 %     xlabel('Enthalpy \it(kW)')
443 %     ylabel('Temperature \it(\circ C)')
444 %     y_label_object = get(gca,'ylabel');
445 %     py = get(y_label_object,'position');
446 %     py(1) = 2*py(1);
447 %     set(y_label_object,'position',py)
448 %     set(gca,'fontsize',26,'FontName','Times New Roman')
449 %print(f_2CC,strcat('4st_coldstreamsT-H',...
450 %datestr(now,'mm_dd_HH_MM')),...
451 %   '-depsc2')
452 % print(f_2CC,'4st_coldstream4st','-depsc2')
453 % f_3CC=figure('Name','Energy Composite Curve Curve before shift',...
454 %              'NumberTitle','off','Position',[0 10 1000 980]);
455 %     plot(obj.hotCoordinates,obj.hotBreaks,'r','LineWidth',2.5);
456 %     hold on;
457 %     plot(obj.coldCoordinates,obj.coldBreaks,'b','LineWidth',2.5);
458 %     legend('Hot(Original)','Cold(Original)','Location','northwest')
459 %     xlabel('Enthalpy \it(kW)')
460 %     ylabel('Temperature \it(\circ C)')
461 %     y_label_object = get(gca,'ylabel');
462 %     py = get(y_label_object,'position');
463 %     py(1) = 2*py(1);
464 %     set(y_label_object,'position',py)
465 %     set(gca,'fontsize',26,'FontName','Times New Roman')
466 % % print(f_3CC,strcat('4st_hotandcoldstreamsT-H',...
467 % %     datestr(now,'mm_dd_HH_MM')),'-depsc2')
468 % print(f_3CC,'4st_hotandcoldstream4st','-depsc2')
```

```
% f_4CC=figure('Name','Energy Composite Curve Curve after shift',...
%              'NumberTitle','off','Position',[0 10 1000 980]);
%     plot(obj.hotCoordinates,obj.hotBreaks,'r','LineWidth',2.5);
%     hold on;
%     plot(obj.coldCoordinates+obj.coldDuty,obj.coldBreaks,...
%         'b','LineWidth',2.5);
%     legend('Hot(Shifted)','Cold(Shifted)','Location','northwest')
%     xlabel('Enthalpy \it(kW)')
%     ylabel('Temperature \it(\circ C)')
%     y_label_object = get(gca,'ylabel');
%     py = get(y_label_object,'position');
%     py(1) = 2*py(1);
%     set(y_label_object,'position',py)
%     set(gca,'fontsize',26,'FontName','Times New Roman')
% % print(f_4CC,strcat('4st_hotandcoldstreamsT-H',...
% %     datestr(now,'mm_dd_HH_MM')),'-depsc2')
% print(f_4CC,'4st_compositecurves4st','-depsc2')
%
% f_5CC=figure('Name','Grand Composite Curve Curve',...
%              'NumberTitle','off','Position',[0 10 800 980]);
%     plot(obj.cascade,obj.allBreaks,'k','LineWidth',1.5);
%     hold on;
%     xlabel('Enthalpy \it(kW)')
%     ylabel('Temperature \it(\circ C)')
%     y_label_object = get(gca,'ylabel');
%     py = get(y_label_object,'position');
%     py(1) = 2*py(1);
%     set(y_label_object,'position',py)
%     set(gca,'fontsize',26,'FontName','Times New Roman')
% % print(f_5CC,strcat('4st_grandcomposite',...
% %     datestr(now,'mm_dd_HH_MM')),'-depsc2')
% print(f_5CC,'4st_grandcomposite4st','-depsc2')
end
function readUserInputs(obj,src,evt)
   resetContainers_Pinch(obj);
   obj.ED_min_initial = [];
   obj.streams_cold_fromtable=obj.coldTable.Data;
   obj.streams_hot_fromtable=obj.hotTable.Data;
   obj.property_Info = obj.propertytable.Data;
   obj.optimizationcheck=0;
if ishandle(obj.NetworkGridFigure)
    close(obj.NetworkGridFigure)
end
if ishandle(obj.NetworkBelowWithHexFigure)
    close(obj.NetworkBelowWithHexFigure)
end
if ishandle(obj.NetworkAboveWithHexFigure)
    close(obj.NetworkAboveWithHexFigure)
end
obj.pinchButton = uicontrol('Parent',obj.pinchFigure,...
    'Style','pushbutton','Units','normalized',...
    'Position', [1300/1600 300/980 267/1600 50/980],...
```

```
    'String','Pinch','Callback',...
    {@(src,evt)pinchCallBack(obj,src,evt)},...
    'BusyAction','cancel', 'Interruptible','off',...
    'Enable','on');
obj.HexButton = uicontrol('Parent',obj.pinchFigure,...
    'Style','pushbutton','Units','normalized',...
    'Position', [1300/1600 230/980 267/1600 50/980] ,...
    'String','HEX','Callback',...
    {@(src,evt)HexCallBack(obj,src,evt)},...
    'BusyAction','cancel', 'Interruptible','off',...
    'Enable','off');
obj.exergyButton = uicontrol('Parent',obj.pinchFigure,...
    'Style','pushbutton','Units','normalized',...
    'Position', [1300/1600 160/980 267/1600 50/980] ,...
    'String','Exergy','Callback',...
    {@(src,evt)exergyCallBack(obj,src,evt)},...
    'BusyAction','cancel', 'Interruptible','off',...
    'Enable','off');
obj.GridButton = uicontrol('Parent',obj.pinchFigure,...
        'Style','pushbutton','Units','normalized',...
        'Position',[1300/1600 90/980 267/1600 50/980],...
        'String','Network',...
        'Callback',@(src,evt)NetworkGridDrawCallback(obj,src,evt),...
        'BusyAction','cancel', 'Interruptible','off',...
        'Enable','off');
print(obj.pinchFigure,'4st_2stepPinchGUI','-depsc2')
end
function [streamCells,breaks,streams_nonempty]=...
        populateStrms(obj,fromtable,streamCells,hot)
% populateStreams converts tables into stream objects
% obj         = this pinch object
%               2nd output is the composite curve less phase chges
% fromtable     = info from table
% streamCells = a cell array of stream objects to populate
% hot         = -1 if hot streams to cool or 1 if cold streams to heat
% breaks      = temperature discontinuities on the composite curve
%               breaks lacks repeated temperatures of phase changes
%--------------------------------MURAT_Comments;----------------------
%     ~= :Checks inequality Conditional check of sum below actually
%     checks whether the matrix is empty or not
%     Table constructed for 9 cold and 9 hot streams (i=1:9)
%     populateStrms fn goes to stream class and the stream fn there,
%     all inputs are assigned to obj there for each individual stream
%     as first stream:streamCells{1} second:streamCells{2}  etc.
%     This here the function definition; rather than defining hot=1
%     or -1 for cold&hot streams they are directly given as -1 or
%     1(After row250)Similarly fromtable is a generic name, which is
%     used at only definition table name is given below(After row250)
%     Streams is a 9x4 array holding Ti,To,cp,m for 9 stream.
%     first and second elements in related row(stream) gives
%     temperatures for streams(i,1) so they are used for break vector.
%     n starts from 1 for hot stream family and cold stream family
```

```
573 %--------------------------------MURAT_Finishes;----------------------
574         breaks = [];        % we do not know how long breaks will be!
575         streams = fromtable;
576         n = 1;
577         streams_nonempty = streams(any(streams,2),:);
578         % make streams from table and collect the breakpoints
579         i_1_upperlimit=(obj.optimizationcheck==0)*5+...
580                         (obj.optimizationcheck==1)*size(fromtable,1);
581         for i_1 = 1:i_1_upperlimit;
582             if sum(streams(i_1,:)) ~= 0
583                 streamCells{i_1} = stream(streams(i_1,:),hot);
584                 breaks(n,1) = streams(i_1,1);
585                 % cannot readily preallocate
586                 n = n+1;
587                 breaks(n,1) = streams(i_1,2);
588                 % cannot readily preallocate
589                 n = n+1;
590             end
591         end
592         % sort then cull redundant breakpoints
593         breaks = sort(breaks);
594         breaks = unique(breaks);
595     end
596 function breaks = addPhaseChgs(obj,breaks,streamCells,Thalve)
597 % addPhaseChgs puts phase changes into the composite curve, where:
598 % obj         = this pinch object
599 % breaks      = temperature discontinuities on the composite curve
600 % streamCells = a cell array of stream objects
601 % Thalve      = half the absolute pinch distance, if relevant
602 %--------------------------------MURAT_Comments;----------------------
603 %     obj as an input take all the variables as properties from
604 %     anything linked to the obj. Breaks are seperately defined for
605 %     hot/cold and they are not stored permanently.
606 %     Below "if case" works when a stream for phase change defined.
607 %     obj.Thalve given as zero when this function is called.
608 %     This function is written in order to differentiate between same
609 %     temperatures for different streams and same temperatures in same
610 %     streams due to phase change.
611 %     seems to me this is not the most effective way to do it no need
612 %     toadd obj.Thalve here.
613 %--------------------------------MURAT_Finishes;----------------------
614             for i_2 = 1:length(streamCells)
615                 if streamCells{i_2}.phase
616                 loc = find(breaks==(streamCells{i_2}.Tlo + Thalve));
617                 breaks = [ breaks(1:loc,1)
618                     streamCells{i_2}.Tlo+Thalve
619                     breaks(loc+1:end,1)
620                     ];
621                 end
622             end
623     end
624 function energies = getEnergies(obj,breaks,streamCells,Thalve)
```

```
% getEnergies calculates all power along a composite curve
% obj         = this pinch object
% breaks      = temperature discontinuities on the composite curve
% streamCells = a cell array of stream objects
% obj.Thalve      = half the absolute pinch distance, if relevant
%---------------------------------MURAT_Comments;--------------------
%  Number of intervals 1 less than break(length). For each interval
%  enthalpy is calculated considering all streams in that interval.
%  they are not stored permanently.
%  Energies array starts from zero for cold/hot composites. Looks
%  like this energy vvariable expresses the change not the actual
%   value. [breaks(i-1) breaks(i)] defines the related interval.
%---------------------------------MURAT_Finishes;--------------------

energies = zeros(length(breaks),1);           % preallocated
    for i_3 = 2:length(breaks)
        energy = 0;
        for j_1 = 1:length(streamCells)
            energy = energy + ...
                demand(streamCells{j_1},...
                [breaks(i_3-1) breaks(i_3)],...
                Thalve);
        end
        energies(i_3,1) = energy;
    end
end
function resetContainers_Pinch(obj)
    obj.hotStreams = {};
    obj.coldStreams = {};
    obj.hotBreaks = [];
    obj.coldBreaks = [];
    obj.hotEnergies = [];
    obj.coldEnergies = [];
    obj.allBreaks = [];
    obj.allEnergies = [];
    obj.allHeats = [];
    obj.allColds = [];
    obj.cascade = [];
    obj.hotCoordinates = [];
    obj.coldCoordinates = [];
    obj.coldDuty = [];
    obj.hotDuty = [];
%-------------------------------MURAT_Adds;-------------------------
    obj.Tpinch = [];
    obj.Thalve = [];
    obj.Highest_T = [];
    obj.Lowest_T = [];
    obj.hotinfo = [];
    obj.coldinfo = [];
    obj.StreamInfo = [];
    obj.HotStreams_BelowPinch = [];
    obj.HotStreams_AbovePinch = [];
```

```
677     obj.ColdStreams_BelowPinch = [];
678     obj.ColdStreams_AbovePinch = [];
679     obj.Hexes_AbovePinch = [];
680     obj.Hexes_BelowPinch = [];
681     obj.hotutilities = [];
682     obj.coldutilities = [];
683     obj.E_level_cold = [];
684     obj.E_level_hot = [];
685     obj.E_level_grand = [];
686     obj.hot_intCoordinates = [];
687     obj.cold_intCoordinates = [];
688     obj.E_level_inthot = [];
689     obj.E_level_intcold = [];
690     obj.exD_4recovery = [];
691     obj.exDL = [];
692     obj.ex_utilizedbysolarthermal = [];
693     obj.ex_solarirradiation = [];
694     obj.hot_alternative = [];
695     obj.cold_alternative = [];
696     obj.IntrsctX = [];
697     obj.IntrsctY = [];
698     obj.main_line_width = [];
699 %--------------------------------MURAT_Done;--------------------------
700     end
701 %-------------------------------MURAT_Comments;----------------------
702 %     Accumulate energies=hotEnergies/coldEnergies arrays expresses
703 %     increments in related intervals,accumulate adds them up.
704 %     pinch_exergy_hex_Calculate fn performs whole pinch analysis
705 %     boxCallback fn calls pinch_exergy_hex_Calculate when increase
706 %     or decrease occursfor DT_min
707 % ----------------------------MURAT_Finishes;-----------------------
708 function accumulated = accumulate(obj,array)
709     accumulated = zeros(length(array),1);
710     for i_4 = 2:length(array)
711         accumulated(i_4,1) = sum( array(1:i_4) );
712     end
713 end
714 function array = clean(obj,array)
715 % clean deletes extra elements phase chg can add to composite curve
716 % obj = this pinch object
717 % array = array to clean
718 % 2  redundant temperatures is indicative of phase change
719 % 2+ redundant temperatures is nonsensical
720 % call any 2+ redundant temperatures...
721     for i_5 = 1:length(array)
722         found = find( array == array(i_5) );
723         if length(found) > 2
724             array( found(1) ) = NaN;
725         end
726     end
727     array = array( ~isnan(array) );
728 end
```

```
function HexCallBack(obj,src,evt)
hex_Calculate(obj)
obj.exergyButton = uicontrol('Parent',obj.pinchFigure,...
    'Style','pushbutton','Units','normalized',...
    'Position', [1300/1600 160/980 267/1600 50/980] ,...
    'String','Exergy','Callback',...
    {@(src,evt)exergyCallBack(obj,src,evt)},...
    'BusyAction','cancel', 'Interruptible','off',...
    'Enable','on');
print(obj.pinchFigure,'4st_4stepPinchGUI','-depsc2')
end
function hex_Calculate(obj)

%Divide as below and above pinch& add m*cp column
for i_7=1:size(obj.hotinfo,1)
    if obj.StreamInfo(i_7,2)<=(obj.Tpinch+obj.Thalve)
        %all of it is below pinch
        obj.HotStreams_BelowPinch=[obj.HotStreams_BelowPinch;...
            obj.StreamInfo(i_7,:)];
    elseif obj.StreamInfo(i_7,1)>=(obj.Tpinch+obj.Thalve)
        %all of it is above pinch
        obj.HotStreams_AbovePinch=[obj.HotStreams_AbovePinch;...
            obj.StreamInfo(i_7,:)];
    else
        obj.HotStreams_BelowPinch=[obj.HotStreams_BelowPinch;...
            obj.StreamInfo(i_7,1),...
            (obj.Tpinch+obj.Thalve),...
            obj.StreamInfo(i_7,3:6)];
        obj.HotStreams_AbovePinch=[obj.HotStreams_AbovePinch;...
            (obj.Tpinch+obj.Thalve),...
            obj.StreamInfo(i_7,2:6)];

    end
end
for i_8=(size(obj.hotinfo,1)+1):size(obj.StreamInfo,1)
    if obj.StreamInfo(i_8,2)<=(obj.Tpinch-obj.Thalve)
        %all of it is below pinch
        obj.ColdStreams_BelowPinch=[obj.ColdStreams_BelowPinch;...
            obj.StreamInfo(i_8,:)];
    elseif obj.StreamInfo(i_8,1)>=(obj.Tpinch-obj.Thalve)
        %all of it is above pinch
        obj.ColdStreams_AbovePinch=[obj.ColdStreams_AbovePinch;...
            obj.StreamInfo(i_8,:)];
    else
        obj.ColdStreams_BelowPinch=[obj.ColdStreams_BelowPinch;...
            obj.StreamInfo(i_8,1),...
            (obj.Tpinch-obj.Thalve),...
            obj.StreamInfo(i_8,3:6)];
        obj.ColdStreams_AbovePinch=[obj.ColdStreams_AbovePinch;...
            (obj.Tpinch-obj.Thalve),...
            obj.StreamInfo(i_8,2:6)];

```

```
    end
end

%Order according to cp*m values & rearrange
if isempty(obj.HotStreams_BelowPinch)==0
    obj.HotStreams_BelowPinch=[obj.HotStreams_BelowPinch(:,1:4)...
    obj.HotStreams_BelowPinch(:,3).*obj.HotStreams_BelowPinch(:,4)...
    (obj.HotStreams_BelowPinch(:,2)-obj.HotStreams_BelowPinch(:,1))...
   .*obj.HotStreams_BelowPinch(:,3).*obj.HotStreams_BelowPinch(:,4)...
   .*obj.HotStreams_BelowPinch(:,6)...
    obj.HotStreams_BelowPinch(:,5)];
else
end

if isempty(obj.HotStreams_AbovePinch)==0
    obj.HotStreams_AbovePinch=[obj.HotStreams_AbovePinch(:,1:4)...
    obj.HotStreams_AbovePinch(:,3).*obj.HotStreams_AbovePinch(:,4)...
    (obj.HotStreams_AbovePinch(:,2)-obj.HotStreams_AbovePinch(:,1))...
   .*obj.HotStreams_AbovePinch(:,3).*obj.HotStreams_AbovePinch(:,4)...
    .*obj.HotStreams_AbovePinch(:,6)...
    obj.HotStreams_AbovePinch(:,5)];
else
end



if isempty(obj.ColdStreams_BelowPinch)==0
    obj.ColdStreams_BelowPinch=[obj.ColdStreams_BelowPinch(:,1:4)...
obj.ColdStreams_BelowPinch(:,3).*obj.ColdStreams_BelowPinch(:,4)...
(obj.ColdStreams_BelowPinch(:,2)-obj.ColdStreams_BelowPinch(:,1))...
.*obj.ColdStreams_BelowPinch(:,3).*obj.ColdStreams_BelowPinch(:,4)...
.*obj.ColdStreams_BelowPinch(:,6)...
obj.ColdStreams_BelowPinch(:,5)];
else
end


if isempty(obj.ColdStreams_AbovePinch)==0
    obj.ColdStreams_AbovePinch=[obj.ColdStreams_AbovePinch(:,1:4)...
obj.ColdStreams_AbovePinch(:,3).*obj.ColdStreams_AbovePinch(:,4)...
(obj.ColdStreams_AbovePinch(:,2)-obj.ColdStreams_AbovePinch(:,1))...
.*obj.ColdStreams_AbovePinch(:,3).*obj.ColdStreams_AbovePinch(:,4)...
.*obj.ColdStreams_AbovePinch(:,6)...
obj.ColdStreams_AbovePinch(:,5)];

else
end
```

```
833 if (isempty(obj.HotStreams_AbovePinch)==0)&&...
834         (isempty(obj.ColdStreams_AbovePinch)==0)
835     [obj.Hexes_AbovePinch,obj.hotutilities]=...
836         build_Hex_sourcesink(obj,obj.HotStreams_AbovePinch,...
837         obj.ColdStreams_AbovePinch);
838 elseif isempty(obj.HotStreams_AbovePinch)
839     obj.hotutilities=[obj.ColdStreams_AbovePinch(:,1:5),...
840     -obj.ColdStreams_AbovePinch(:,6),obj.ColdStreams_AbovePinch(:,7)];
841 end
842
843 if (isempty(obj.HotStreams_BelowPinch)==0)&&...
844         (isempty(obj.ColdStreams_BelowPinch)==0)
845     [obj.Hexes_BelowPinch,obj.coldutilities]=...
846         build_Hex_sourcesink(obj,obj.ColdStreams_BelowPinch,...
847         obj.HotStreams_BelowPinch);
848 elseif isempty(obj.ColdStreams_BelowPinch)
849     obj.coldutilities=[obj.HotStreams_BelowPinch(:,1:5),...
850       -obj.HotStreams_BelowPinch(:,6),obj.HotStreams_BelowPinch(:,7)];
851 end
852 end
853 function [hexstreamfrom_devided,hexstreamfrom_undevided,...
854     source_todivide,sink_todivide]=...
855     divide_streams_sourcesink(obj,source_todivide,...
856     sink_todivide,k_6,T_target,which1targeted)
857 if (source_todivide(1,6)==-sink_todivide(1,6))&&...
858         ((source_todivide(1,k_6)*sign(source_todivide(1,6))+...
859         (sink_todivide(1,k_6)*sign(sink_todivide(1,6))))>=...
860         (2*obj.Thalve))&&...
861         (source_todivide(1,5)<=sink_todivide(1,5))
862     hexstreamfrom_devided=sink_todivide(1,:);
863     hexstreamfrom_undevided=source_todivide(1,:);
864     sink_todivide(1,:)=[];
865     source_todivide(1,:)=[];
866 else
867     if nargin==4
868         if abs(source_todivide(1,6))-abs(sink_todivide(1,6))>...
869                                                     obj.howaccurate
870
871             %divide source
872             divided_stream=[source_todivide(1,1),...
873             source_todivide(1,k_6)-...
874             sink_todivide(1,6)/source_todivide(1,5),...
875             source_todivide(1,3:5),...
876             (k_6==2)*(-1*...
877             abs(abs(source_todivide(1,6))-abs(sink_todivide(1,6))))+...
878             (k_6==1)*(-sink_todivide(1,6)),...
879             source_todivide(1,7);...
880             source_todivide(1,k_6)-...
881             sink_todivide(1,6)/source_todivide(1,5),...
882             source_todivide(1,2:5),...
883             (k_6==2)*(-sink_todivide(1,6))+...
884             (k_6==1)*abs(abs(source_todivide(1,6))-...
```

```
            abs(sink_todivide(1,6))),source_todivide(1,7)];
            if ((divided_stream(k_6,k_6)*...
               sign(source_todivide(1,6))+(sink_todivide(1,k_6)*...
               sign(sink_todivide(1,6))))>=(2*obj.Thalve))&&...
              (divided_stream(1,2)>=source_todivide(1,1))&&...
              (divided_stream(1,2)<=source_todivide(1,2))
                hexstreamfrom_devided=divided_stream(k_6,:);
                hexstreamfrom_undevided=sink_todivide(1,:);
                source_todivide(1,:)=[];
                if isempty(source_todivide)
                    source_todivide=[(divided_stream(1,:))*...
                        (k_6==2)+(divided_stream(2,:))*(k_6==1)];
                else
                    source_todivide=[source_todivide;...
                        (divided_stream(1,:))*(k_6==2)+...
                        (divided_stream(2,:))*(k_6==1)];
                end
                sink_todivide(1,:)=[];
            else
                %do not change source and sink
               hexstreamfrom_devided=zeros(1,size(divided_stream,2));
               hexstreamfrom_undevided=zeros(1,size(sink_todivide,2));
            end
        elseif abs(sink_todivide(1,6))-abs(source_todivide(1,6))>...
                obj.howaccurate
            %dividehot
        divided_stream=[sink_todivide(1,1),...
        sink_todivide(1,k_6)+...
        source_todivide(1,6)/sink_todivide(1,5),...
        sink_todivide(1,3:5),...
        (k_6==2)*abs(abs(source_todivide(1,6))-...
        abs(sink_todivide(1,6)))+...
        (k_6==1)*(-source_todivide(1,6)),...
        sink_todivide(1,7);...
        sink_todivide(1,k_6)+...
        source_todivide(1,6)/sink_todivide(1,5),...
        sink_todivide(1,2:5),...
        (k_6==2)*(-source_todivide(1,6))+...
        (k_6==1)*(-1*...
        abs(abs(source_todivide(1,6))-abs(sink_todivide(1,6)))),...
        sink_todivide(1,7)];
            if ((divided_stream(k_6,k_6)*sign(sink_todivide(1,6))+...
               (source_todivide(1,k_6)*...
               sign(source_todivide(1,6))))>=(2*obj.Thalve))&&...
               (divided_stream(1,2)>=sink_todivide(1,1))&&...
               (divided_stream(1,2)<=sink_todivide(1,2))
                hexstreamfrom_devided=divided_stream(k_6,:);
                hexstreamfrom_undevided=source_todivide(1,:);
                sink_todivide(1,:)=[];
                if isempty(sink_todivide)
                    sink_todivide=[(divided_stream(1,:))*...
                        (k_6==2)+(divided_stream(2,:))*(k_6==1)];
```

```
937                 else
938                 sink_todivide=[sink_todivide;(divided_stream(1,:))*...
939                             (k_6==2)+(divided_stream(2,:))*(k_6==1)];
940                 end
941                 source_todivide(1,:)=[];
942             else
943                 %do not change source and sink
944             hexstreamfrom_devided=zeros(1,size(divided_stream,2));
945             hexstreamfrom_undevided=zeros(1,size(source_todivide,2));
946             end
947         else
948 %it is possible by accident,same temp difference for both streams by
949 %chance to begin with energies are different by they become same after
950 %split the choice of devided/undivided totally arbitrary
951             hexstreamfrom_devided=sink_todivide(1,:);
952             hexstreamfrom_undevided=source_todivide(1,:);
953             sink_todivide(1,:)=[];
954             source_todivide(1,:)=[];
955         end
956     elseif nargin==6
957         if (which1targeted==1)
958             if (k_6==1)
959                 divided_source=[source_todivide(1,1),T_target,...
960                 source_todivide(1,3:5),...
961                 (T_target-source_todivide(1,1))...
962                 *source_todivide(1,5),source_todivide(1,7);...
963                 T_target,source_todivide(1,2),...
964                 source_todivide(1,3:5),...
965                 (source_todivide(1,2)-T_target)*...
966                 source_todivide(1,5),source_todivide(1,7)];
967                 divided_sink=[sink_todivide(1,1),...
968                     (sink_todivide(1,1)+...
969                     divided_source(1,6)/sink_todivide(1,5)),...
970                     sink_todivide(1,3:5),-divided_source(1,6),...
971                     sink_todivide(1,7);...
972                     (sink_todivide(1,1)+...
973                     divided_source(1,6)/sink_todivide(1,5)),...
974                     sink_todivide(1,2),...
975                     sink_todivide(1,3:5),...
976                     sink_todivide(1,6)+divided_source(1,6),...
977                     sink_todivide(1,7)];
978               %actually both are divided by terminology is not changed
979                 if ((divided_sink(1,6)*divided_sink(2,6))<0)
980                 divided_source=[source_todivide(1,1),...
981                     source_todivide(1,1)-...
982                     sink_todivide(1,6)/source_todivide(1,5),...
983                     source_todivide(1,3:5),...
984                     -sink_todivide(1,6),source_todivide(1,7);...
985                     source_todivide(1,1)-...
986                     sink_todivide(1,6)/source_todivide(1,5),...
987                     source_todivide(1,2),source_todivide(1,3:5)...
988                     source_todivide(1,6)+sink_todivide(1,6),...
```

```
                    source_todivide(1,7)];
                    hexstreamfrom_undevided=sink_todivide(1,:);
                    sink_todivide(1,:)=[];
                else
                    hexstreamfrom_undevided=divided_sink(1,:);
                    sink_todivide(1,:)=divided_sink(2,:);
                end
                    hexstreamfrom_devided=divided_source(1,:);
                    source_todivide(1,:)=divided_source(2,:);
            elseif (k_6==2)
                divided_source=[T_target,source_todivide(1,2),...
                    source_todivide(1,3:5),...
                    -(source_todivide(1,2)-T_target)*...
                    source_todivide(1,5),source_todivide(1,7);...
                    source_todivide(1,1),...
                    T_target,source_todivide(1,3:5),...
                    -(T_target-source_todivide(1,1))*...
                    source_todivide(1,5),source_todivide(1,7)];
                divided_sink=[sink_todivide(1,2)+...
                    divided_source(1,6)/sink_todivide(1,5),...
                    sink_todivide(1,2),...
                    sink_todivide(1,3:5),-divided_source(1,6),...
                    sink_todivide(1,7);...
                    sink_todivide(1,1),sink_todivide(1,2)+...
                    divided_source(1,6)/sink_todivide(1,5),...
                    sink_todivide(1,3:5),...
                    sink_todivide(1,6)+divided_source(1,6),...
                    sink_todivide(1,7)];

                if ((divided_sink(1,6)*divided_sink(2,6))<0)
                divided_source=[source_todivide(1,2)-...
                    sink_todivide(1,6)/source_todivide(1,5),...
                    source_todivide(1,2),...
                    source_todivide(1,3:5),...
                    -sink_todivide(1,6),source_todivide(1,7);...
                    source_todivide(1,1),...
                    source_todivide(1,2)-...
                    sink_todivide(1,6)/source_todivide(1,5),...
                    source_todivide(1,3:5)...
                    source_todivide(1,6)+sink_todivide(1,6),...
                    source_todivide(1,7)];
                    hexstreamfrom_undevided=sink_todivide(1,:);
                    sink_todivide(1,:)=[];
                else
                    hexstreamfrom_undevided=divided_sink(1,:);
                    sink_todivide(1,:)=divided_sink(2,:);
                end
                    hexstreamfrom_devided=divided_source(1,:);
                    source_todivide(1,:)=divided_source(2,:);
            end
        end
        if (which1targeted==-1)
```

```
1041            if (k_6==1)
1042                divided_sink=[sink_todivide(1,1),T_target,...
1043                sink_todivide(1,3:5),...
1044                -(T_target-sink_todivide(1,1))*sink_todivide(1,5),...
1045                sink_todivide(1,7);...
1046                T_target,...
1047                sink_todivide(1,2),...
1048                sink_todivide(1,3:5),...
1049                -(sink_todivide(1,2)-T_target)*sink_todivide(1,5),...
1050                sink_todivide(1,7)];
1051                divided_source=[source_todivide(1,1),...
1052                    (source_todivide(1,1)-...
1053                    divided_sink(1,6)/source_todivide(1,5)),...
1054                    source_todivide(1,3:5),...
1055                    -divided_sink(1,6),source_todivide(1,7);...
1056                    (source_todivide(1,1)-...
1057                    divided_sink(1,6)/source_todivide(1,5)),...
1058                    source_todivide(1,2),source_todivide(1,3:5)...
1059                    source_todivide(1,6)+divided_sink(1,6),...
1060                    source_todivide(1,7)];
1061                if ((divided_source(1,6)*divided_source(2,6))<0)
1062                    divided_sink=[sink_todivide(1,1),...
1063                        sink_todivide(1,1)+...
1064                        source_todivide(1,6)/sink_todivide(1,5),...
1065                        sink_todivide(1,3:5),...
1066                        -source_todivide(1,6),...
1067                        sink_todivide(1,7);...
1068                        sink_todivide(1,1)+...
1069                        source_todivide(1,6)/sink_todivide(1,5),...
1070                        sink_todivide(1,2:5),...
1071                        sink_todivide(1,6)+source_todivide(1,6),...
1072                        sink_todivide(1,7)];
1073                    hexstreamfrom_devided=source_todivide(1,:);
1074                    source_todivide(1,:)=[];
1075                else
1076                    hexstreamfrom_devided=divided_source(1,:);
1077                    source_todivide(1,:)=divided_source(2,:);
1078                end
1079                hexstreamfrom_undevided=divided_sink(1,:);
1080                sink_todivide(1,:)=divided_sink(2,:);
1081            elseif (k_6==2)
1082                divided_sink=[T_target,sink_todivide(1,2),...
1083                sink_todivide(1,3:5),...
1084                (sink_todivide(1,2)-T_target)*sink_todivide(1,5),...
1085                sink_todivide(1,7);...
1086                sink_todivide(1,1),T_target,sink_todivide(1,3:5),...
1087                (T_target-sink_todivide(1,1))*sink_todivide(1,5),...
1088                sink_todivide(1,7)];
1089                divided_source=[source_todivide(1,2)-...
1090                    divided_sink(1,6)/source_todivide(1,5),...
1091                    source_todivide(1,2),source_todivide(1,3:5),...
1092                    -divided_sink(1,6),source_todivide(1,7);...
```

```
                    source_todivide(1,1),source_todivide(1,2)-...
                    divided_sink(1,6)/source_todivide(1,5),...
                    source_todivide(1,3:5)...
                    source_todivide(1,6)+divided_sink(1,6),...
                    source_todivide(1,7)];
                if ((divided_source(1,6)*divided_source(2,6))<0)
                    divided_sink=[sink_todivide(1,2)+...
                        source_todivide(1,6)/sink_todivide(1,5),...
                        sink_todivide(1,2:5),...
                        -source_todivide(1,6),...
                        sink_todivide(1,7);...
                        sink_todivide(1,1),...
                        sink_todivide(1,2)+...
                        source_todivide(1,6)/sink_todivide(1,5),...
                        sink_todivide(1,3:5),...
                        sink_todivide(1,6)+source_todivide(1,6),...
                        sink_todivide(1,7)];

                    hexstreamfrom_devided=source_todivide(1,:);
                    source_todivide(1,:)=[];
                else
                    hexstreamfrom_devided=divided_source(1,:);
                    source_todivide(1,:)=divided_source(2,:);
                end
                hexstreamfrom_undevided=divided_sink(1,:);
                sink_todivide(1,:)=divided_sink(2,:);
            end
        end
    end
end
end


function [source_tosplit,sink_tosplit]=...
       splitstream_sourcesink(obj,source_tosplit,sink_tosplit)
     if source_tosplit(1,5)>sink_tosplit(1,5)
         split_source=[source_tosplit(1,1:2),...
             sink_tosplit(1,5)/source_tosplit(1,4),...
             source_tosplit(1,4),sink_tosplit(1,5),...
             sign(source_tosplit(1,6))*sink_tosplit(1,5)*...
             (source_tosplit(1,2)-source_tosplit(1,1)),...
             source_tosplit(1,7);...
             source_tosplit(1,1:2),...
             (source_tosplit(1,5)-...
             sink_tosplit(1,5))/source_tosplit(1,4),...
             source_tosplit(1,4),...
             (source_tosplit(1,5)-sink_tosplit(1,5)),...
             sign(source_tosplit(1,6))*(source_tosplit(1,5)-...
             sink_tosplit(1,5))*(source_tosplit(1,2)-...
             source_tosplit(1,1)),source_tosplit(1,7)];
source_tosplit=[split_source;...
    source_tosplit(2:size(source_tosplit,1),:)];
```

```
1145      elseif source_tosplit(1,5)<sink_tosplit(1,5)
1146          split_sink=[sink_tosplit(1,1:2),...
1147              source_tosplit(1,5)/sink_tosplit(1,4),...
1148              sink_tosplit(1,4) source_tosplit(1,5),...
1149              sign(sink_tosplit(1,6))*source_tosplit(1,5)*...
1150              (sink_tosplit(1,2)-sink_tosplit(1,1)),...
1151              sink_tosplit(1,7);...
1152              sink_tosplit(1,1:2),...
1153              (sink_tosplit(1,5)-...
1154              source_tosplit(1,5))/sink_tosplit(1,4),...
1155              sink_tosplit(1,4),...
1156              (sink_tosplit(1,5)-source_tosplit(1,5)),...
1157              sign(sink_tosplit(1,6))*...
1158              (sink_tosplit(1,5)-source_tosplit(1,5))*...
1159              (sink_tosplit(1,2)-sink_tosplit(1,1)),...
1160              sink_tosplit(1,7)];
1161 % end
1162
1163  if 1==1
1164      sink_tosplit=[split_sink;sink_tosplit(2:size(sink_tosplit,1),:)];
1165  elseif 1==size(sink_tosplit,1)
1166      sink_tosplit=[sink_tosplit(1:(1-1),:);split_sink];
1167  else
1168      sink_tosplit=[sink_tosplit(1:(1-1),:);split_sink;...
1169          sink_tosplit((1+1):size(sink_tosplit,1),:)];
1170  end
1171      else
1172          %this case should not use this part
1173      end
1174 end
1175
1176 function [HEX,utilities]=build_Hex_sourcesink(obj,source,sink)
1177 %should use while since loopsize changes
1178 %create hex and remove those streams from stream matrix
1179 %m: Hex count
1180 %j_#: Sink Index
1181 %Always try to match first remaining stream of source and appropriate
1182 %sink. When a stream or part of a stream is used in a hex remove it
1183 %from stream matrices and change sink counter to avoid unchecked sink
1184 %streams (e.g. if you remove stream 3, you have to check stream 3 in
1185 %the next step since previous stream 4 became stream 3 after used
1186 %stream removal So it is necessary to decrease j_# when a stream is
1187 %checked and used in a HEX, but increase when it is checked but
1188 %criterias are not met to use it Increase the m when Hex connection
1189 %is established
1190 k_4=1*(sign(source(1,6))==1)+2*(sign(sink(1,6))==1);
1191 %1 above
1192 %2 below
1193 m=0;
1194 establishment_fail=0;
1195 z_1=2;
1196 z_2=2;
```

```
1197 if exist('dummy_utilityarray','var')
1198 delete(dummy_utilityarray)
1199 end
1200
1201
1202 interval4source_raw=[source(:,1);source(:,2)];
1203 interval4source_raw2=sort(unique(interval4source_raw),'descend');
1204 interval4source_raw3=[interval4source_raw2 interval4source_raw2+...
1205                                 (2*obj.Thalve)*((k_4==2)-(k_4==1))];
1206
1207 interval4sink_raw=[sink(:,1);sink(:,2)];
1208 interval4sink_raw2=sort(unique(interval4sink_raw),'descend');
1209 interval4sink_raw3=[interval4sink_raw2-(2*obj.Thalve)*...
1210                             ((k_4==2)-(k_4==1)) interval4sink_raw2 ];
1211
1212 interval_raw=unique([interval4source_raw3;interval4sink_raw3],'rows');
1213
1214 if (k_4==2)
1215 interval_final=sortrows(interval_raw,2,'descend');
1216 elseif (k_4==1)
1217 interval_final=sortrows(interval_raw,2,'ascend');
1218 end
1219
1220 security_iterationcount=0;
1221 while (size(source,1)~=0)&&(establishment_fail==0)&&...
1222                                         (security_iterationcount<1000)
1223
1224     src_mrg=1;
1225     while src_mrg<=size(source,1)
1226         src_mrg_2=1;
1227         while src_mrg_2<=size(source,1)
1228             if (src_mrg~=src_mrg_2)&&...
1229                     (source(src_mrg,7)==source(src_mrg_2,7))&&...
1230                     (source(src_mrg,1)==source(src_mrg_2,1))&&...
1231                     (source(src_mrg,2)==source(src_mrg_2,2))
1232               source(src_mrg,3)=source(src_mrg,3)+source(src_mrg_2,3);
1233               source(src_mrg,5)=source(src_mrg,5)+source(src_mrg_2,5);
1234               source(src_mrg,6)=source(src_mrg,6)+source(src_mrg_2,6);
1235               source(src_mrg_2,:)=[];
1236             else
1237             src_mrg_2=src_mrg_2+1;
1238             end
1239         end
1240         src_mrg=src_mrg+1;
1241     end
1242
1243     snk_mrg=1;
1244     while snk_mrg<=size(sink,1)
1245         snk_mrg_2=1;
1246         while snk_mrg_2<=size(sink,1)
1247             if (snk_mrg~=snk_mrg_2)&&...
1248                     (sink(snk_mrg,7)==sink(snk_mrg_2,7))&&...
```

```
                    (sink(snk_mrg,1)==sink(snk_mrg_2,1))&&...
                    (sink(snk_mrg,2)==sink(snk_mrg_2,2))
                sink(snk_mrg,3)=sink(snk_mrg,3)+sink(snk_mrg_2,3);
                sink(snk_mrg,5)=sink(snk_mrg,5)+sink(snk_mrg_2,5);
                sink(snk_mrg,6)=sink(snk_mrg,6)+sink(snk_mrg_2,6);
                sink(snk_mrg_2,:)=[];
            else
            snk_mrg_2=snk_mrg_2+1;
            end
        end
        snk_mrg=snk_mrg+1;
    end

    if k_4==2
        %energy negative in source below pinch so descend
    source=sortrows(source,[2,6],{'descend','ascend'});
    sink=sortrows(sink,[2,6],{'descend','ascend'});
    else
    source=sortrows(source,[1,5,6],{'ascend'});
    sink=sortrows(sink,[1,5,6],{'ascend'});
    end

[source,sink]=splitstream_sourcesink(obj,source,sink);

for rz_1=1:size(interval_final,1)
    if (k_4==2)
    re_index=((source(:,k_4)<=interval_final(rz_1,1)));
    elseif (k_4==1)
    re_index=((source(:,k_4)>=interval_final(rz_1,1)));
    end
if (prod(re_index,1)==1)&&(rz_1>=z_1)
z_1=rz_1;
end
end

T_source_target_raw=interval_final(z_1,1);
if k_4==2
checkJsource_index =(source(:,k_4)<=T_source_target_raw);
else
checkJsource_index =(source(:,k_4)>=T_source_target_raw);
end
if prod(checkJsource_index,1)==1
    T_sink_target_raw=interval_final(z_1,2);
    if k_4==2
        checkJsink_index =(sink(:,k_4)<=T_sink_target_raw);
    else
        checkJsink_index =(sink(:,k_4)>=T_sink_target_raw);
    end
    if prod(checkJsink_index,1)==1
        z_1=z_1+1;
        Temperature_Target_Source=interval_final(z_1,1);
        Target_check_1=1;
```

```
1301
1302     else
1303         Temperature_Target_Sink=T_sink_target_raw;
1304         Delta_Tt_sink=abs(sink(1,k_4)-Temperature_Target_Sink);
1305         T_source_proj=source(1,k_4)-((k_4==2)-(k_4==1))*Delta_Tt_sink;
1306         if (k_4==2)&&(T_source_proj<interval_final(z_1+1,1))
1307             Temperature_Target_Source=interval_final(z_1+1,1);
1308             Target_check_1=1;
1309         elseif(k_4==2)&&(Temperature_Target_Sink>=sink(1,1))
1310             Target_check_1=-1;
1311             Temperature_Target_Source=interval_final(z_1+1,1);
1312         elseif(k_4==2)&&(Temperature_Target_Sink<sink(1,1))
1313             Temperature_Target_Sink=sink(1,1);
1314             Target_check_1=-1;
1315         elseif(k_4==1)&&(T_source_proj>source(1,2))
1316             Temperature_Target_Source=source(1,2);
1317             Target_check_1=1;
1318         elseif(k_4==1)&&(T_source_proj>interval_final(z_1+1,1))
1319             Target_check_1=1;
1320             Temperature_Target_Source=interval_final(z_1+1,1);
1321         elseif(k_4==1)&&(Temperature_Target_Sink>=sink(1,2))
1322             Temperature_Target_Sink=sink(1,2);
1323             Target_check_1=-1;
1324         elseif(k_4==1)&&(Temperature_Target_Sink<sink(1,2))
1325             Target_check_1=-1;
1326         end
1327     end
1328
1329 else
1330     Temperature_Target_Source=T_source_target_raw;
1331     % %may be erased and remaining works
1332     Delta_Tt_source=abs(source(1,k_4)-Temperature_Target_Source);
1333     T_sink_proj=sink(1,k_4)-((k_4==2)-(k_4==1))*Delta_Tt_source;
1334     %
1335     if (size(sink,1)>=2)&&(size(source,1)>=2)
1336     if (k_4==1)&&(T_sink_proj<=sink(2,1))&&(source(2,1)-T_sink_proj<(2*ob
1337         Target_check_1=-1;
1338         Temperature_Target_Sink=source(2,1)-(2*obj.Thalve);
1339     elseif (k_4==2)&&(T_sink_proj>=sink(2,2))&&(T_sink_proj-source(2,2)<
1340         Target_check_1=-1;
1341         Temperature_Target_Sink=source(2,2)+(2*obj.Thalve);
1342     else
1343         Target_check_1=1;
1344     end
1345     else
1346     Target_check_1=1;
1347     end
1348
1349 end
1350
1351 if (Target_check_1==1)
1352 Temperature_Target=Temperature_Target_Source;
```

```
elseif (Target_check_1==-1)
Temperature_Target=Temperature_Target_Sink;
end

if ((source(1,k_4)*sign(source(1,6))+...
            (sink(1,k_4)*sign(sink(1,6))))>=(2*obj.Thalve))
           [stream_devidedforHEX,stream_undevided,source,sink]=...
               divide_streams_sourcesink(obj,source,sink,k_4,...
               Temperature_Target,Target_check_1);
security_iterationcount=security_iterationcount+1;
             if (sum(stream_devidedforHEX)~=0)&&...
                    (sum(stream_undevided)~=0)&&...
                    (sign(stream_devidedforHEX(1,6))==-1)
               m=m+1;
               HEX(1,:,m)=stream_devidedforHEX;
               HEX(2,:,m)=stream_undevided;
            elseif  (sum(stream_devidedforHEX)~=0)&&...
                    (sum(stream_undevided)~=0)&&...
                    (sign(stream_devidedforHEX(1,6))==1)
               m=m+1;
               HEX(2,:,m)=stream_devidedforHEX;
               HEX(1,:,m)=stream_undevided;
             end

else
establishment_fail=1;
end

        for src_ers=1:size(source,1)
            if source(src_ers,1)==source(src_ers,2)
            source(src_ers,:)=[];
            break
            else
            end
        end
        for snk_ers=1:size(sink,1)
            if sink(snk_ers,1)==sink(snk_ers,2)
            sink(snk_ers,:)=[];
            break
            else
            end
        end
end


utilities=[sink(:,1:5),-sink(:,6),sink(:,7)]; %utility=left sink
sink=[sink;zeros(1,size(sink,2))]; %avoid error in 1D sink
dummy=sum(sink);
utility=dummy(6);
if (isempty(source)==1)
    if (HEX(1,1,m)<(obj.Tpinch-obj.Thalve))
```

```
1405         if abs(utility-obj.coldDuty)<=obj.howaccurate
1406           disp('Heat Exchangers are created successfully below pinch')
1407         else
1408             disp('Work on buil_Hex function  belowpinch'),
1409         end
1410     elseif (HEX(1,2,m)>(obj.Tpinch-obj.Thalve))
1411         if abs(utility+obj.hotDuty)<=obj.howaccurate
1412           disp('Heat Exchangers are created successfully above pinch')
1413         else
1414             disp('Work on buil_Hex function abovepinch'),
1415         end
1416     else
1417     end
1418 else
1419     disp('Work on buil_Hex function'),
1420 end
1421 end
1422
1423
1424 function [HEXtoDraw]=DrawingNetworkHEX(obj,createdHEX,UtilitiestoDraw)
1425 adjusted_line_width=5./(max(createdHEX));
1426 HEXtoDraw=createdHEX;
1427 HEXtoDraw(:,3,:)=[];
1428 HEXtoDraw(:,4,:)=[];
1429 recount=[obj.StreamInfo(:,5),zeros(size(obj.StreamInfo,1),1),...
1430          zeros(size(obj.StreamInfo,1),1),...
1431          zeros(size(obj.StreamInfo,1),1)];
1432 for i_12=1:size(obj.StreamInfo,1)
1433     counter=0;
1434     for k_1=1:size(HEXtoDraw,3)
1435     counter=(HEXtoDraw(1,5,k_1)==i_12)*1+...
1436             (HEXtoDraw(2,5,k_1)==i_12)*1+counter;
1437     end
1438 %     for k_2=1:size(inUtility,1)
1439 %         counter=(inUtility(k_2,7)==i_12)*1+counter;
1440 %     end
1441     recount(i_12,2)=counter; %may be unnecessary to store this column
1442 end
1443 %leave 0.05 for distinction of splits and HEX lines
1444 clearance=0.025;
1445 verticalspaceforHEXs=0.25;
1446 verticalspaceforsplits=1-verticalspaceforHEXs-2*clearance;
1447
1448 for i_13=1:size(recount,1)
1449 %start from middle of the stream then add horizontal steps
1450 verticalstepforsplit=verticalspaceforsplits/(recount(i_13,2))*...
1451     (recount(i_13,2)~=1)+0; %+1 is to avoid overlapping hex and stream
1452 verticalstepforHEX=verticalspaceforHEXs/(recount(i_13,2))*...
1453     (recount(i_13,2)~=1)+0; %+1 is to avoid overlapping hex and stream
1454 countnum_forstream=0;
1455     for k_4=1:size(HEXtoDraw,3)
1456         if (i_13==HEXtoDraw(1,5,k_4))
```

```
1457 horizontalspaceforHEXs=(HEXtoDraw(1,2,k_4)-HEXtoDraw(1,1,k_4))./100;
1458 horizontalstepforHEX=horizontalspaceforHEXs/(recount(i_13,2))*...
1459         (recount(i_13,2)~=1)+0;
1460 %vertical position of stream
1461 HEXtoDraw(1,6,k_4)=HEXtoDraw(1,5,k_4)+(-0.25+clearance+...
1462     (countnum_forstream)*verticalstepforsplit)*...
1463     ((recount(i_13,2)~=1)&&(recount(i_13,2)~=0));
1464 %vertical position of HEX step
1465 HEXtoDraw(1,7,k_4)=HEXtoDraw(1,6,k_4)-0.25-...
1466     (countnum_forstream*verticalstepforHEX)*...
1467     ((recount(i_13,2)~=1)&&(recount(i_13,2)~=0));
1468 %horizontal position of HEX step
1469 HEXtoDraw(1,8,k_4)=(HEXtoDraw(1,1,k_4)+HEXtoDraw(1,2,k_4))/2+...
1470     (countnum_forstream)*horizontalstepforHEX*...
1471     ((recount(i_13,2)~=1)&&(recount(i_13,2)~=0));
1472 countnum_forstream=countnum_forstream+1;
1473         elseif (i_13==HEXtoDraw(2,5,k_4))
1474 horizontalspaceforHEXs=(HEXtoDraw(2,2,k_4)-HEXtoDraw(2,1,k_4))./100;
1475 horizontalstepforHEX=horizontalspaceforHEXs/(recount(i_13,2))*...
1476     (recount(i_13,2)~=1)+0;
1477         HEXtoDraw(2,6,k_4)=HEXtoDraw(2,5,k_4)+(-0.25+clearance+...
1478             (countnum_forstream)*verticalstepforsplit)*...
1479             ((recount(i_13,2)~=1)&&(recount(i_13,2)~=0));
1480         HEXtoDraw(2,8,k_4)=(HEXtoDraw(2,1,k_4)+...
1481             HEXtoDraw(2,2,k_4))/2-...
1482             countnum_forstream*horizontalstepforHEX*...
1483             ((recount(i_13,2)~=1)&&(recount(i_13,2)~=0));
1484         countnum_forstream=countnum_forstream+1;
1485         else
1486         end
1487     end
1488 end
1489 HEXtoDraw(2,7,:)=HEXtoDraw(1,7,:);
1490 for k_5=1:size(HEXtoDraw,3)
1491 plot([HEXtoDraw(1,1,k_5) HEXtoDraw(1,2,k_5)],...
1492      [HEXtoDraw(1,6,k_5) HEXtoDraw(1,6,k_5)],...
1493      'LineWidth',...
1494      obj.main_line_width*HEXtoDraw(1,3,k_5),...
1495      'Color','b');
1496 plot([HEXtoDraw(1,1,k_5) HEXtoDraw(1,1,k_5)],...
1497      [HEXtoDraw(1,5,k_5) HEXtoDraw(1,6,k_5)],':b');
1498 plot([HEXtoDraw(1,2,k_5) HEXtoDraw(1,2,k_5)],...
1499      [HEXtoDraw(1,5,k_5) HEXtoDraw(1,6,k_5)],':b');
1500 plot([HEXtoDraw(2,1,k_5) HEXtoDraw(2,2,k_5)],...
1501      [HEXtoDraw(2,6,k_5) HEXtoDraw(2,6,k_5)],...
1502      'LineWidth',...
1503      obj.main_line_width*HEXtoDraw(2,3,k_5),...
1504      'Color','r');
1505 plot([HEXtoDraw(2,1,k_5) HEXtoDraw(2,1,k_5)],...
1506      [HEXtoDraw(2,5,k_5) HEXtoDraw(2,6,k_5)],':r');
1507 plot([HEXtoDraw(2,2,k_5) HEXtoDraw(2,2,k_5)],...
1508      [HEXtoDraw(2,5,k_5) HEXtoDraw(2,6,k_5)],':r');
```

```
plot([(HEXtoDraw(1,1,k_5)+HEXtoDraw(1,2,k_5))/2,...
       HEXtoDraw(1,8,k_5)],...
      [HEXtoDraw(1,6,k_5) HEXtoDraw(1,7,k_5)],'m');
plot([HEXtoDraw(1,8,k_5) HEXtoDraw(2,8,k_5)],...
     [HEXtoDraw(1,7,k_5) HEXtoDraw(2,7,k_5)],'m');
plot([(HEXtoDraw(2,1,k_5)+HEXtoDraw(2,2,k_5))/2,...
       HEXtoDraw(2,8,k_5)],...
      [HEXtoDraw(2,6,k_5) HEXtoDraw(2,7,k_5)],'m');
end
end
function HexDrawCallBack_BP(obj,src,evt)
if ishandle(obj.NetworkBelowWithHexFigure)
    close(obj.NetworkBelowWithHexFigure)
end
obj.NetworkBelowWithHexFigure = figure('name',...
    'Network Grid Below Pinch with Heat Exchangers',...
    'numbertitle','off',...
    'ToolBar','none',...
    'MenuBar','None',...
    'Units','pixels',...
    'Position',[200 20 1600 1000],...
    'Resize','off');
%if only one cold below pinch
xminlimit_below=isvector(obj.ColdStreams_BelowPinch)*...
                        obj.ColdStreams_BelowPinch(1)+...
           (isvector(obj.ColdStreams_BelowPinch)==0)*...
                          min(obj.ColdStreams_BelowPinch);
obj.NetworkBelowWithHexAxes = axes('Units','normalized',...
    'Position',[50/1600 50/1000 1500/1600 920/1000],...
    'Parent',obj.NetworkBelowWithHexFigure,...
    'YLim',[0 size(obj.StreamInfo,1)+1],'YTick',...
    [1:size(obj.StreamInfo,1)],...
    'XLim',[xminlimit_below(1)-10 obj.Tpinch+obj.Thalve+10]);
    cla(obj.NetworkBelowWithHexAxes);
    axes(obj.NetworkBelowWithHexAxes);
    hold on
obj.ylabel_BP_NetworkGrid=annotation('textbox','LineStyle','none',...
                            'String','Stream Number',...
                            'Position',...
                            [10/1600 975/1000 300/1600 25/1000],...
                            'Units','normalized');
obj.xlabel_BP_NetworkGrid=annotation('textbox','LineStyle','none',...
                            'String','Temperature(^{\circ}C)',...
                            'Position',...
                            [780/1600 10/1000 1650/1600 25/1000],...
                            'Units','normalized');
for i_11=1:size(obj.HotStreams_BelowPinch,1)
    if(obj.HotStreams_BelowPinch(i_11,1)==...
            obj.HotStreams_BelowPinch(i_11,2))%Phase Change
    obj.Highest_T=obj.Tpinch+obj.Thalve;
    obj.Lowest_T=...
    min(obj.HotStreams_BelowPinch,obj.ColdStreams_BelowPinch);
```

```
    plot([max(obj.Highest_T(:,1:2)),min(obj.Lowest_T(:,1:2))],...
         [obj.HotStreams_BelowPinch(i_11,7),...
         obj.HotStreams_BelowPinch(i_11,7)],...
         ':k','Marker','<');
    hold on
    else
    plot(obj.HotStreams_BelowPinch(i_11,1:2),...
         [obj.HotStreams_BelowPinch(i_11,7),...
         obj.HotStreams_BelowPinch(i_11,7)],...
         ':k','Marker','<','MarkerSize',5);
    hold on
    end
end

for i_14=1:size(obj.ColdStreams_BelowPinch,1)
    if(obj.ColdStreams_BelowPinch(i_14,1)==...
            obj.ColdStreams_BelowPinch(i_14,2))%Phase Change
    obj.Highest_T=...
    max(obj.HotStreams_BelowPinch,obj.ColdStreams_BelowPinch);
    obj.Lowest_T=obj.Tpinch+obj.Thalve;
    plot([max(obj.Highest_T(:,1:2)),min(obj.Lowest_T(:,1:2))],...
         [obj.ColdStreams_BelowPinch(i_14,7),...
         obj.ColdStreams_BelowPinch(i_14,7)],...
        'Marker','>');
    hold on
    else
        plot(obj.ColdStreams_BelowPinch(i_14,1:2),...
             [obj.ColdStreams_BelowPinch(i_14,7),...
              obj.ColdStreams_BelowPinch(i_14,7)],...
             ':k','Marker','>','MarkerSize',5);
        hold on
    end
end

[Hexes_BelowPinchtoDraw]=...
        DrawingNetworkHEX(obj,obj.Hexes_BelowPinch,obj.coldutilities);
print(obj.NetworkBelowWithHexFigure,'4st_BelowPinchHEXGUI','-depsc2')
end
function HexDrawCallBack_AP(obj,src,evt)
if ishandle(obj.NetworkAboveWithHexFigure)
    close(obj.NetworkAboveWithHexFigure)
end

obj.NetworkAboveWithHexFigure = figure('name',...
    'Network Grid Above Pinch with Heat Exchangers',...
    'numbertitle','off','ToolBar','none','MenuBar','None',...
    'Units','pixels','Position',[200 20 1600 1000],...
    'Resize','off');
%if only one hot above pinch
xmaxlimit_below=isvector(obj.HotStreams_AbovePinch)*...
                        obj.HotStreams_AbovePinch(2)+...
           (isvector(obj.HotStreams_AbovePinch)==0)*...
```

```
                        max(obj.HotStreams_AbovePinch);
obj.NetworkAboveWithHexAxes = axes('Units','normalized',...
    'Position',[50/1600 50/1000 1500/1600 920/1000],...
    'Parent',obj.NetworkAboveWithHexFigure,...
    'YLim',...
    [0 size(obj.StreamInfo,1)+1],'YTick',...
    [1:size(obj.StreamInfo,1)],'XLim',...
    [obj.Tpinch-obj.Thalve-10 xmaxlimit_below+10]);
    hold on
    cla(obj.NetworkAboveWithHexAxes);
    axes(obj.NetworkAboveWithHexAxes);
obj.ylabel_AP_NetworkGrid=annotation('textbox','LineStyle','none',...
                            'String','Stream Number',...
                            'Position',...
                            [10/1600 975/1000 300/1600 25/1000],...
                            'Units','normalized');
obj.xlabel_AP_NetworkGrid=annotation('textbox','LineStyle','none',...
                            'String','Temperature(^{\circ}C)',...
                            'Position',...
                            [780/1600 10/1000 1650/1600 25/1000],...
                            'Units','normalized');
for i_11=1:size(obj.HotStreams_AbovePinch,1)
    if(obj.HotStreams_AbovePinch(i_11,1)==...
       obj.HotStreams_AbovePinch(i_11,2))%Phase Change
obj.Lowest_T=obj.Tpinch-obj.Thalve;
obj.Highest_T=max(obj.HotStreams_AbovePinch,...
                  obj.ColdStreams_AbovePinch);
plot([max(obj.Highest_T(:,1:2)) min(obj.Lowest_T(:,1:2))],...
     [obj.HotStreams_AbovePinch(i_11,7),...
      obj.HotStreams_AbovePinch(i_11,7)],':k','Marker','<');
      hold on
    else
plot(obj.HotStreams_AbovePinch(i_11,1:2),...
    [obj.HotStreams_AbovePinch(i_11,7),...
     obj.HotStreams_AbovePinch(i_11,7)],':k','Marker',...
     '<','MarkerSize',5);
        hold on
    end
end

for i_14=1:size(obj.ColdStreams_AbovePinch,1)
    if(obj.ColdStreams_AbovePinch(i_14,1)==...
                      obj.ColdStreams_AbovePinch(i_14,2))%Phase Change
obj.Lowest_T=obj.Tpinch-obj.Thalve;
obj.Highest_T=man(obj.HotStreams_AbovePinch,...
                  obj.ColdStreams_AbovePinch);
plot([max(obj.Highest_T(:,1:2)),min(obj.Lowest_T(:,1:2))],...
     [obj.ColdStreams_AbovePinch(i_14,7),...
      obj.ColdStreams_AbovePinch(i_14,7)],'Marker','>');
hold on
    else
plot(obj.ColdStreams_AbovePinch(i_14,1:2),...
```

```
1665      [obj.ColdStreams_AbovePinch(i_14,7),...
1666       obj.ColdStreams_AbovePinch(i_14,7)],':k','Marker',...
1667       '>','MarkerSize',5);
1668 hold on
1669     end
1670 end
1671
1672 [Hexes_AbovePinchtoDraw]=...
1673          DrawingNetworkHEX(obj,obj.Hexes_AbovePinch,obj.hotutilities);
1674 print(obj.NetworkAboveWithHexFigure,'4st_AbovePinchHEXGUI','-depsc2')
1675 end
1676 function NetworkGridDrawCallback(obj,src,evt)
1677     % Close the figure if exist
1678 if ishandle(obj.NetworkGridFigure)
1679     close(obj.NetworkGridFigure)
1680 end
1681 obj.NetworkGridFigure = figure('name','Network Grid',...
1682     'numbertitle','off','ToolBar','none','MenuBar','None',...
1683     'Units','pixels','Position',[20 50 1800 1000],'Resize','off');
1684 obj.NetworkGridAxes = axes('Units','normalized',...
1685     'Position',[50/1800 90/1000 1550/1800 860/1000],...
1686     'Parent',obj.NetworkGridFigure,...
1687     'YLim',[0 size(obj.StreamInfo,1)+1],...
1688     'YTick',[1:size(obj.StreamInfo,1)],...
1689     'XLim',[obj.allBreaks(1)-20 obj.allBreaks(end)+20]);
1690
1691 obj.xlabelNetworkGrid  = annotation('textbox','LineStyle','none',...
1692                           'String','Stream Number',...
1693                           'Position',...
1694                           [10/1800 960/1000 300/1800 25/1000],...
1695                           'Units','normalized');
1696 obj.ylabelNetworkGrid  = annotation('textbox','LineStyle','none',...
1697                           'String','Temperature(^{\circ}C)',...
1698                           'Position',...
1699                           [800/1800 40/1000 1650/1800 25/1000],...
1700                           'Units','normalized');
1701 hold on
1702 cla(obj.NetworkGridAxes);
1703 axes(obj.NetworkGridAxes);
1704         obj.Highest_T=max(obj.StreamInfo);
1705         obj.Lowest_T=min(obj.StreamInfo);
1706 m4lw=max(obj.StreamInfo);
1707 cp4lw=max(obj.StreamInfo);
1708 obj.main_line_width=4./(m4lw(3)*cp4lw(4));
1709 for i_9=1:size(obj.StreamInfo,1)
1710     if(obj.StreamInfo(i_9,1)==obj.StreamInfo(i_9,2))%Phase Change
1711         if obj.StreamInfo(i_9,6)==1
1712             plot([max(obj.Highest_T(1:2)) min(obj.Lowest_T(1:2))],...
1713                 [i_9 i_9],'LineWidth',obj.main_line_width*...
1714                 obj.StreamInfo(i_9,3)*obj.StreamInfo(i_9,4),...
1715                 'Marker','<','Color','r');
1716             hold on
```

```
        elseif obj.StreamInfo(i_9,6)==-1
            plot([max(obj.Highest_T(:,1:2)),...
                 min(obj.Lowest_T(:,1:2))],...
                 [i_9 i_9],'LineWidth',obj.main_line_width*...
                 obj.StreamInfo(i_9,3)*obj.StreamInfo(i_9,4),...
                'Marker','>','Color','b');
            hold on
        end
    else
        if obj.StreamInfo(i_9,6)==1
            plot(obj.StreamInfo(i_9,1:2), [i_9 i_9],...
                'LineWidth',obj.main_line_width*...
                 obj.StreamInfo(i_9,3)*obj.StreamInfo(i_9,4),...
                'Marker','<','MarkerSize',5,'Color','r');
            hold on
        elseif obj.StreamInfo(i_9,6)==-1
            plot(obj.StreamInfo(i_9,1:2), [i_9 i_9],...
                'LineWidth',obj.main_line_width*...
                 obj.StreamInfo(i_9,3)*obj.StreamInfo(i_9,4),...
                'Marker','>','MarkerSize',5,'Color','b');
            hold on
        end
    end
end

plot([obj.Tpinch+obj.Thalve obj.Tpinch+obj.Thalve],...
     [0.5 size(obj.hotinfo,1)+0.25],':r');
hold on
plot([obj.Tpinch-obj.Thalve obj.Tpinch-obj.Thalve],...
     [size(obj.hotinfo,1)+0.25 size(obj.hotinfo,1)+...
     0.25+size(obj.coldinfo,1)+0.25],':b');
plot([obj.Tpinch-obj.Thalve obj.Tpinch+obj.Thalve],...
     [size(obj.hotinfo,1)+0.25 size(obj.hotinfo,1)+0.25],':k');

%  print(datestr(now,'mmmm_dd_HH_MM'),'-dbmp16m')*********************


if isempty(obj.Hexes_BelowPinch)
obj.NOHexBuilt_BP = annotation('Parent',obj.NetworkGridFigure,...
    'textbox','String',...
    'No heat recovery below pinch for these inputs',...
    'Position',[60/1600 10/1000 150/1600 40/1000],...
    'Units','normalized','EdgeColor',0.9411*[0.5 0.5 0.5]);
else
obj.HexBuiltButton_BP = uicontrol('Parent',obj.NetworkGridFigure,...
    'Style','pushbutton','Units','normalized',...
    'Position',[60/1600 10/1000 150/1600 60/1000],...
    'String','Show HEXes Below Pinch',...
    'Callback',@(src,evt)HexDrawCallBack_BP(obj,src,evt),...
    'BusyAction','cancel', 'Interruptible','off');
end
if isempty(obj.Hexes_AbovePinch)
```

```
1769 obj.NOHexBuilt_AP = annotation('Parent',obj.NetworkGridFigure,...
1770     'textbox','String',...
1771     'No heat recovery above pinch for these inputs',...
1772     'Position',[1210/1600 10/1000 150/1600 40/1000],...
1773     'Units','normalized','EdgeColor',0.9411*[0.5 0.5 0.5]);
1774 else
1775 obj.HexBuiltButton_AP = uicontrol('Parent',obj.NetworkGridFigure,...
1776     'Style','pushbutton','Units','normalized',...
1777     'Position',[1210/1600 10/1000 150/1600 60/1000],...
1778     'String','Show HEXes Above Pinch',...
1779     'Callback',@(src,evt)HexDrawCallBack_AP(obj,src,evt),...
1780     'BusyAction','cancel', 'Interruptible','off');
1781 end
1782
1783 obj.hotcheck_title  = annotation('textbox','LineStyle','none',...
1784    'String','Mark invariable hot streams ',...
1785    'Position',...
1786    [1440/1600 720/1000 150/1600 150/1000],...
1787    'Units','normalized','fontsize',9);
1788 obj.checkHS1 = uicontrol('parent',obj.NetworkGridFigure,...
1789                 'Units','normalized',...
1790                 'Position',[1450/1600 800/1000 150/1600 40/1000],...
1791                 'Style','checkbox', 'string','1st Hot Stream');
1792            if size(obj.hotinfo,1)>=2
1793 obj.checkHS2 = uicontrol('parent',obj.NetworkGridFigure,...
1794                 'Units','normalized',...
1795                 'Position',[1450/1600 760/1000 150/1600 40/1000],...
1796                 'Style','checkbox', 'string','2nd Hot Stream');
1797                 if size(obj.hotinfo,1)>=3
1798 obj.checkHS3 = uicontrol('parent',obj.NetworkGridFigure,...
1799                 'Units','normalized',...
1800                 'Position',[1450/1600 720/1000 150/1600 40/1000],...
1801                 'Style','checkbox', 'string','3rd Hot Stream');
1802                     if size(obj.hotinfo,1)>=4
1803 obj.checkHS4 = uicontrol('parent',obj.NetworkGridFigure,...
1804                 'Units','normalized',...
1805                 'Position',[1450/1600 680/1000 150/1600 40/1000],...
1806                 'Style','checkbox', 'string','4th Hot Stream');
1807                         if size(obj.hotinfo,1)==5
1808 obj.checkHS5 = uicontrol('parent',obj.NetworkGridFigure,...
1809                 'Units','normalized',...
1810                 'Position',[1450/1600 640/1000 150/1600 40/1000],...
1811                 'Style','checkbox', 'string','5th Hot Stream');
1812                         else
1813                         end
1814                     else
1815                     end
1816                 else
1817                 end
1818            else
1819            end
1820
```

```
1821 obj.coldcheck_title  = annotation('textbox','LineStyle','none',...
1822    'String','Mark invariable cold streams ',...
1823    'Position',...
1824    [1440/1600 360/1000 150/1600 150/1000],...
1825    'Units','normalized','fontsize',9);
1826 obj.checkCS1 = uicontrol('parent',obj.NetworkGridFigure,...
1827                 'Units','normalized',...
1828                 'Position',[1450/1600 440/1000 150/1600 40/1000],...
1829                 'Style','checkbox', 'string','1st Cold Stream');
1830            if size(obj.coldinfo,1)>=2
1831 obj.checkCS2 = uicontrol('parent',obj.NetworkGridFigure,...
1832                 'Units','normalized',...
1833                 'Position',[1450/1600 400/1000 150/1600 40/1000],...
1834                 'Style','checkbox', 'string','2nd Cold Stream');
1835                 if size(obj.coldinfo,1)>=3
1836 obj.checkCS3 = uicontrol('parent',obj.NetworkGridFigure,...
1837                 'Units','normalized',...
1838                 'Position',[1450/1600 360/1000 150/1600 40/1000],...
1839                 'Style','checkbox', 'string','3rd Cold Stream');
1840                     if size(obj.coldinfo,1)>=4
1841 obj.checkCS4 = uicontrol('parent',obj.NetworkGridFigure,...
1842                 'Units','normalized',...
1843                 'Position',[1450/1600 320/1000 150/1600 40/1000],...
1844                 'Style','checkbox', 'string','4th Cold Stream');
1845                         if size(obj.coldinfo,1)==5
1846 obj.checkCS5 = uicontrol('parent',obj.NetworkGridFigure,...
1847                 'Units','normalized',...
1848                 'Position',[1450/1600 280/1000 150/1600 40/1000],...
1849                 'Style','checkbox', 'string','5th Cold Stream');
1850                         else
1851                         end
1852                     else
1853                     end
1854                 else
1855                 end
1856            else
1857            end
1858
1859 obj.OptimizationButton = uicontrol('Parent',obj.NetworkGridFigure,...
1860     'Style','pushbutton','Units','normalized',...
1861     'Position',[1450/1600 150/1000 110/1600 60/1000],...
1862     'String','Optimize',...
1863     'Callback',@(src,evt)OptimizationCallBack(obj,src,evt),...
1864     'BusyAction','cancel', 'Interruptible','off');
1865
1866 print(obj.NetworkGridFigure,'4st_1stepNetworkGUI','-depsc2')
1867 end
1868 function exergyCallBack(obj,src,evt)
1869 exergy_Calculate(obj)
1870 ExergyCompositeCurveDraw(obj)
1871 obj.GridButton = uicontrol('Parent',obj.pinchFigure,...
1872         'Style','pushbutton','Units','normalized',...
```

```
1873         'Position',[1300/1600 90/980 267/1600 50/980],...
1874         'String','Network',...
1875         'Callback',@(src,evt)NetworkGridDrawCallback(obj,src,evt),...
1876         'BusyAction','cancel', 'Interruptible','off',...
1877         'Enable','on');
1878 print(obj.pinchFigure,'4st_5stepPinchGUI','-depsc2')
1879 end
1880 function exergy_Calculate(obj)
1881
1882 obj.E_level_cold=1-(obj.property_Info(2)+273)./(obj.coldBreaks+273);
1883 obj.E_level_hot=1-(obj.property_Info(2)+273)./(obj.hotBreaks+273);
1884 obj.E_level_grand=1-(obj.property_Info(2)+273)./(obj.allBreaks+273);
1885
1886
1887 obj.exD_4recovery=trapz(obj.E_level_grand,obj.cascade);
1888
1889 %T_sun=6000K
1890 obj.ex_solarirradiation=obj.property_Info(3)*obj.property_Info(4)*...
1891                         (1-(4/3)*((obj.property_Info(2)+273)/6000)+...
1892                         (1/3)*(((obj.property_Info(2)+273)/6000)^4));
1893
1894 obj.ex_utilizedbysolarthermal=0;
1895 for i_15=1:size(obj.hotutilities,1)
1896
1897 obj.ex_utilizedbysolarthermal=obj.ex_utilizedbysolarthermal+...
1898           obj.hotutilities(i_15,6)-obj.hotutilities(i_15,5)*...
1899                                    (obj.property_Info(2)+273)*...
1900                            log((obj.hotutilities(i_15,2)+273)/...
1901                               (obj.hotutilities(i_15,1)+273));
1902 end
1903
1904 obj.exDL=(obj.ex_solarirradiation-obj.ex_utilizedbysolarthermal)*0.02;
1905 %0.3 very primitive approach to thermal exergy loss, in order to find
1906 %exact value T_reciever required. Either assume it or correct it
1907 %lit suggest 0.01 for oil 0.27 for air.
1908
1909 %solar thermal only used in heating so T_out term is high temperature,
1910 %considering negligable pressure drop for liquids ;
1911 obj.ex_DL_log(1)=obj.exD_4recovery;
1912 obj.ex_DL_log(2)=obj.exDL;
1913 obj.ex_DL_log(3)=obj.exD_4recovery+obj.exDL;
1914
1915 end
1916 function ExergyCompositeCurveDraw(obj)
1917     if ishandle(obj.NetworkGridFigure)
1918         close(obj.NetworkGridFigure)
1919     end
1920
1921 cla(obj.Ecc_Axes);
1922 axes(obj.Ecc_Axes);
1923 plot(obj.hotCoordinates,obj.E_level_hot,'r');
1924 hold on;
```

```
1925 plot(obj.coldCoordinates+obj.coldDuty,obj.E_level_cold,'b');
1926 legend('Hot','Cold','Location','northwest')
1927 cla(obj.GEccAxes);
1928 axes(obj.GEccAxes);
1929 plot(obj.cascade,obj.E_level_grand,'k');
1930
1931 %   print(datestr(now,'mm_dd_HH_MM'),'-dbmp16m')*******
1932
1933 % f_6CC=figure('Name','Exergy Composite Curve','NumberTitle',...
1934 %     'off','Position',[0 10 1000 980]);
1935 %     plot(obj.hotCoordinates,obj.E_level_hot,'r','LineWidth',2.5);
1936 %     hold on;
1937 %     plot(obj.coldCoordinates+obj.coldDuty,obj.E_level_cold,'b',...
1938 %          'LineWidth',3);
1939 %     xlabel('Enthalpy \it(kW)')
1940 %     ylabel('Carnot Factor, \eta_{c}')
1941 %     y_label_object = get(gca,'ylabel');
1942 %     py = get(y_label_object,'position');
1943 %     py(1) = 2*py(1) ;
1944 %     set(y_label_object,'position',py)
1945 %     set(gca,'fontsize',26,'FontName','Times New Roman')
1946 % %     print(f_6CC,...
1947 %strcat('4st_ECC',datestr(now,'mm_dd_HH_MM')),'-depsc2')
1948 % %print(f_6CC,'4st_ECC','-depsc2')
1949 % f_7CC=figure('Name','Grand Exergy Composite Curve','NumberTitle',...
1950 %     'off','Position',[0 10 800 980]);
1951 %     plot(obj.cascade,obj.E_level_grand,'k','LineWidth',1.5);
1952 %     hold on;
1953 %     xlabel('Enthalpy \it(kW)')
1954 %     ylabel('Carnot Factor, \eta_{c}')
1955 %     y_label_object = get(gca,'ylabel');
1956 %     py = get(y_label_object,'position');
1957 %     py(1) = 2*py(1) ;
1958 %     set(y_label_object,'position',py)
1959 %     set(gca,'fontsize',26,'FontName','Times New Roman')
1960 % %     print(f_7CC,...
1961 %     strcat('4st_GECC',datestr(now,'mm_dd_HH_MM')),'-depsc2')
1962 end
1963 function OptimizationCallBack(obj,src,evt)
1964 delete *.xlsx
1965 print(obj.NetworkGridFigure,'4st_2stepNetworkGUI','-depsc2')
1966 NOHEXtitle={'No HEX is build'};
1967 HEXBPtitle={'HEX Below Pinch'};
1968 xlswrite('HEXInfoComparison.xlsx',HEXBPtitle,'BelowPinch_OG','A1')
1969 if isempty(obj.Hexes_BelowPinch)
1970 xlswrite('HEXInfoComparison.xlsx',NOHEXtitle,...
1971                                     'BelowPinch_OG','A2')
1972 else
1973 for h_bp=1:size(obj.Hexes_BelowPinch,3)
1974 RangeTitleBP=strcat('A',num2str(4*h_bp-2));
1975 HEXBPtitle={strcat('HEX',num2str(h_bp))};
1976 RownoBP=num2str(4*h_bp-1);
```

```
xlswrite('HEXInfoComparison.xlsx',...
              HEXBPtitle,'BelowPinch_OG',RangeTitleBP)
xlswrite('HEXInfoComparison.xlsx',...
              obj.Hexes_BelowPinch(:,:,h_bp),'BelowPinch_OG',RownoBP)
end
end

HEXAPtitle={'HEX Above Pinch'};
xlswrite('HEXInfoComparison.xlsx',HEXAPtitle,'AbovePinch_OG','A1')
if isempty(obj.Hexes_AbovePinch)
xlswrite('HEXInfoComparison.xlsx',NOHEXtitle,...
                                    'AbovePinch_OG','A2')
else
for h_ap=1:size(obj.Hexes_AbovePinch,3)
RangeTitleAP=strcat('A',num2str(4*h_ap-2));
HEXAPtitle={strcat('HEX',num2str(h_ap))};
RownoAP=num2str(4*h_ap-1);
xlswrite('HEXInfoComparison.xlsx',...
              HEXAPtitle,'AbovePinch_OG',RangeTitleAP)
xlswrite('HEXInfoComparison.xlsx',...
              obj.Hexes_AbovePinch(:,:,h_ap),'AbovePinch_OG',RownoAP)
end

for h_hot=1:size(obj.hotutilities,1)
RangeTitlehotU=strcat('A',num2str(3*h_hot-2));
HotUtitle={strcat('Hot Utility',num2str(h_hot))};
RownoHotU=num2str(3*h_hot-1);
xlswrite('HEXInfoComparison.xlsx',...
              HotUtitle,'HotUtilities_OG',RangeTitlehotU)
xlswrite('HEXInfoComparison.xlsx',...
              obj.hotutilities(h_hot,:),'HotUtilities_OG',RownoHotU)
end

for h_cold=1:size(obj.coldutilities,1)
RangeTitlecoldU=strcat('A',num2str(3*h_cold-2));
coldUtitle={strcat('cold Utility',num2str(h_cold))};
RownocoldU=num2str(3*h_cold-1);
xlswrite('HEXInfoComparison.xlsx',...
              coldUtitle,'coldUtilities_OG',RangeTitlecoldU)
xlswrite('HEXInfoComparison.xlsx',...
            obj.coldutilities(h_cold,:),'coldUtilities_OG',RownocoldU)
end

end

obj.modify_check_hot=zeros(size(obj.hotinfo,1),1);
obj.modify_check_hot(1,1)=get(obj.checkHS1,'Value');
if size(obj.hotinfo,1)>=2
obj.modify_check_hot(2,1)=get(obj.checkHS2,'Value');
    if size(obj.hotinfo,1)>=3
obj.modify_check_hot(3,1)=get(obj.checkHS3,'Value');
        if size(obj.hotinfo,1)>=4
```

```
obj.modify_check_hot(4,1)=get(obj.checkHS4,'Value');
            if size(obj.hotinfo,1)==5
obj.modify_check_hot(5,1)=get(obj.checkHS5,'Value');
            else
            end
        else
        end
    else
    end
else
end
obj.modify_check_cold=zeros(size(obj.coldinfo,1),1);
obj.modify_check_cold(1,1)=get(obj.checkCS1,'Value');
if size(obj.coldinfo,1)>=2
obj.modify_check_cold(2,1)=get(obj.checkCS2,'Value');
    if size(obj.coldinfo,1)>=3
obj.modify_check_cold(3,1)=get(obj.checkCS3,'Value');
        if size(obj.coldinfo,1)>=4
obj.modify_check_cold(4,1)=get(obj.checkCS4,'Value');
            if size(obj.coldinfo,1)==5
obj.modify_check_cold(5,1)=get(obj.checkCS5,'Value');
            else
            end
        else
        end
    else
    end
else
end

obj.modify_check=zeros(10,1);
obj.modify_check(1:size(obj.StreamInfo,1),1)=...
                        [obj.modify_check_hot;obj.modify_check_cold];
obj.optimizationcheck=1;
realOG=obj.StreamInfo(:,1:4);
obj.ED_min_initial=obj.ex_DL_log(3);
obj.ED_min=obj.ex_DL_log(3);
obj.opt_inputs=realOG;

[obj.alternativesforstreams_all]=nested4opt(obj,realOG);
for h_alt=1:size(obj.alternativesforstreams_all,3)
RangeTitleAlt=strcat('A',num2str(10*h_alt-9));
Alttitle={strcat('Alternative',num2str(h_alt))};
RownoAlt=num2str(10*h_alt-8);
xlswrite('AllAlternatives.xlsx',...
              Alttitle,'Sheet1',RangeTitleAlt)
xlswrite('AllAlternatives.xlsx',...
          obj.alternativesforstreams_all(:,:,h_alt),'Sheet1',RownoAlt)
end

tic
nestedEDcheck(obj,realOG,obj.alternativesforstreams_all)
```

```
toc
%can not use hex info directly, that is the last alternative not the
%optimum inputs. Recall pinch,
obj.opt_hots=obj.opt_inputs(1:size(obj.hotinfo,1),:);
obj.opt_colds=obj.opt_inputs((size(obj.hotinfo,1)+1):end,:);
pinch_Calculate(obj,obj.opt_hots,obj.opt_colds)
hex_Calculate(obj)
exergy_Calculate(obj)

HEXBPtitle={'Modified HEX Below Pinch'};
xlswrite('HEXInfoComparison.xlsx',HEXBPtitle,...
                                    'BelowPinch_Modified','A1')
if isempty(obj.Hexes_BelowPinch)
xlswrite('HEXInfoComparison.xlsx',NOHEXtitle,...
                                    'BelowPinch_Modified','A2')
else
for h_bp_m=1:size(obj.Hexes_BelowPinch,3)
RangeTitleBP=strcat('A',num2str(4*h_bp_m-2));
HEXBPtitle={strcat('HEX',num2str(h_bp_m))};
RownoBP=num2str(4*h_bp_m-1);
xlswrite('HEXInfoComparison.xlsx',...
              HEXBPtitle,'BelowPinch_Modified',RangeTitleBP)
xlswrite('HEXInfoComparison.xlsx',obj.Hexes_BelowPinch(:,:,h_bp_m),...
                                    'BelowPinch_Modified',RownoBP)
end


end
HEXAPtitle={'Modified HEX Below Pinch'};
xlswrite('HEXInfoComparison.xlsx',HEXAPtitle,...
                                        'AbovePinch_Modified','A1')
if isempty(obj.Hexes_AbovePinch)
xlswrite('HEXInfoComparison.xlsx',NOHEXtitle,...
                                    'AbovePinch_Modified','A2')
else
for h_ap_m=1:size(obj.Hexes_AbovePinch,3)
RangeTitleAP=strcat('A',num2str(4*h_ap_m-2));
HEXAPtitle={strcat('HEX',num2str(h_ap_m))};
RownoAP=num2str(4*h_ap_m-1);
xlswrite('HEXInfoComparison.xlsx',...
              HEXAPtitle,'AbovePinch_Modified',RangeTitleAP)
xlswrite('HEXInfoComparison.xlsx',obj.Hexes_AbovePinch(:,:,h_ap_m),...
                                        'AbovePinch_Modified',RownoAP)
end

end

for h_hot_m=1:size(obj.hotutilities,1)
RangeTitlehotU=strcat('A',num2str(3*h_hot_m-2));
HotUtitle={strcat('Hot Utility',num2str(h_hot_m))};
RownoHotU=num2str(3*h_hot_m-1);
xlswrite('HEXInfoComparison.xlsx',...
```

```
2133              HotUtitle,'HotUtilities_Modified',RangeTitlehotU)
2134 xlswrite('HEXInfoComparison.xlsx',...
2135         obj.hotutilities(h_hot_m,:),'HotUtilities_Modified',RownoHotU)
2136 end
2137
2138 for h_cold_m=1:size(obj.coldutilities,1)
2139 RangeTitlecoldU=strcat('A',num2str(3*h_cold_m-2));
2140 coldUtitle={strcat('cold Utility',num2str(h_cold_m))};
2141 RownocoldU=num2str(3*h_cold_m-1);
2142 xlswrite('HEXInfoComparison.xlsx',...
2143              coldUtitle,'coldUtilities_Modified',RangeTitlecoldU)
2144 xlswrite('HEXInfoComparison.xlsx',...
2145     obj.coldutilities(h_cold_m,:),'coldUtilities_Modified',RownocoldU)
2146 end
2147
2148 obj.Percent_Redutction=100*(obj.ED_min_initial-obj.ED_min)/...
2149                                          obj.ED_min_initial;
2150 fprintf('Exergy destruction is reduced %d percent',...
2151                                  obj.Percent_Redutction);
2152 optimizedDrawingcallback(obj)
2153 end
2154 %--------------------------------MURAT_Adds;--------------------------
2155 function [alternative_every_stream]=nested4opt(obj,cont_info)
2156 alternative_every_stream=zeros(9,size(cont_info,2),size(cont_info,1));
2157 for s_1=1:size(cont_info,1)
2158     for s_2=1:9
2159     alternative_every_stream(s_2,:,s_1)=cont_info(s_1,:);
2160     end
2161 end
2162
2163 for s_3=1:size(cont_info,1)
2164     alt_indice_count=1;
2165     if obj.modify_check(s_3,1)==0
2166     for s_4=1:3
2167 Tlow=(s_4==1)*(cont_info(s_3,1))+...
2168       (s_4==2)*(cont_info(s_3,1)-...
2169                 (cont_info(s_3,2)-cont_info(s_3,1))/10)+...
2170       (s_4==3)*(cont_info(s_3,1)+...
2171                 (cont_info(s_3,2)-cont_info(s_3,1))/10);
2172 Thigh=(s_4==1)*(cont_info(s_3,2))+...
2173        (s_4==2)*(cont_info(s_3,2)-...
2174                 (cont_info(s_3,2)-cont_info(s_3,1))/10)+...
2175        (s_4==3)*(cont_info(s_3,2)+...
2176                 (cont_info(s_3,2)-cont_info(s_3,1))/10);
2177              for s_6=1:3
2178 mflow=(s_6==1)*(cont_info(s_3,3))+...
2179       (s_6==2)*(cont_info(s_3,3)*0.8)+...
2180       (s_6==3)*(cont_info(s_3,3)*1.2);
2181 alternative_every_stream(alt_indice_count,1,s_3)=Tlow;
2182 alternative_every_stream(alt_indice_count,2,s_3)=Thigh;
2183 alternative_every_stream(alt_indice_count,3,s_3)=mflow;
2184 alt_indice_count=alt_indice_count+1;
```

```
2185             end
2186     end
2187     else
2188     end
2189 end
2190 end
2191 function nestedEDcheck(obj,OG_SI,all_alt)
2192     Current_SI=OG_SI;
2193 % saybak=1;
2194 for p_1=1:(8*(obj.modify_check(1,1)~=1)+1)
2195     Current_SI(1,:)=all_alt(p_1,:,1);
2196     for p_2=1:(8*(obj.modify_check(2,1)~=1)+1)
2197         Current_SI(2,:)=all_alt(p_2,:,2);
2198         for p_3=1:(8*((obj.modify_check(3,1)~=1)&&...
2199                                                 (size(OG_SI,1)>=3))+1)
2200         if size(OG_SI,1)>=3
2201             Current_SI(3,:)=all_alt(p_3,:,3);
2202 [obj.ED_min,obj.opt_inputs]=...
2203 checkEDcallback(obj,Current_SI,3,obj.ED_min,obj.opt_inputs,OG_SI);
2204             for p_4=1:(8*((obj.modify_check(4,1)~=1)&&...
2205                                                 (size(OG_SI,1)>=4))+1)
2206             if size(OG_SI,1)>=4
2207                 Current_SI(4,:)=all_alt(p_4,:,4);
2208 [obj.ED_min,obj.opt_inputs]=...
2209 checkEDcallback(obj,Current_SI,4,obj.ED_min,obj.opt_inputs,OG_SI);
2210                 for p_5=1:(8*((obj.modify_check(5,1)~=1)&&...
2211                                                 (size(OG_SI,1)>=5))+1)
2212                 if size(OG_SI,1)>=5
2213                     Current_SI(5,:)=all_alt(p_5,:,5);
2214 [obj.ED_min,obj.opt_inputs]=...
2215 checkEDcallback(obj,Current_SI,5,obj.ED_min,obj.opt_inputs,OG_SI);
2216                     for p_6=1:(8*((obj.modify_check(6,1)~=1)&&...
2217                                                 (size(OG_SI,1)>=6))+1)
2218                     if size(OG_SI,1)>=6
2219                         Current_SI(6,:)=all_alt(p_6,:,6);
2220 [obj.ED_min,obj.opt_inputs]=...
2221 checkEDcallback(obj,Current_SI,6,obj.ED_min,obj.opt_inputs,OG_SI);
2222                         for p_7=1:(8*((obj.modify_check(7,1)~=1)&&...
2223                                                 (size(OG_SI,1)>=7))+1)
2224                         if size(OG_SI,1)>=7
2225                             Current_SI(7,:)=all_alt(p_7,:,7);
2226 [obj.ED_min,obj.opt_inputs]=...
2227 checkEDcallback(obj,Current_SI,7,obj.ED_min,obj.opt_inputs,OG_SI);
2228                             for p_8=1:(8*...
2229                     ((obj.modify_check(8,1)~=1)&&(size(OG_SI,1)>=8))+1)
2230                             if size(OG_SI,1)>=8
2231                                 Current_SI(8,:)=all_alt(p_8,:,8);
2232 [obj.ED_min,obj.opt_inputs]=...
2233 checkEDcallback(obj,Current_SI,8,obj.ED_min,obj.opt_inputs,OG_SI);
2234                                 for p_9=1:(8*...
2235                     ((obj.modify_check(9,1)~=1)&&(size(OG_SI,1)>=9))+1)
2236                                 if size(OG_SI,1)>=9
```

```
                                    Current_SI(9,:)=all_alt(p_9,:,9);
[obj.ED_min,obj.opt_inputs]=...
checkEDcallback(obj,Current_SI,9,obj.ED_min,obj.opt_inputs,OG_SI);
                              for p_10=1:(8*...
                                        (obj.modify_check(10,1)~=1)+1)
                              if size(OG_SI,1)>=10
                                  Current_SI(10,:)=all_alt(p_10,:,10);
[obj.ED_min,obj.opt_inputs]=...
checkEDcallback(obj,Current_SI,10,obj.ED_min,obj.opt_inputs,OG_SI);
                                  Current_SI(10,:)=OG_SI(10,:);
                              else
                              end
                              end
                                      Current_SI(9,:)=OG_SI(9,:);
                                  else
                                  end
                                  end
                                  Current_SI(8,:)=OG_SI(8,:);
                              else
                              end
                              end
                              Current_SI(7,:)=OG_SI(7,:);
                          else
                          end
                          end
                          Current_SI(6,:)=OG_SI(6,:);
                      else
                      end
                      end
                      Current_SI(5,:)=OG_SI(5,:);
                  else
                  end
                  end
                  Current_SI(4,:)=OG_SI(4,:);
              else
              end
              end
              Current_SI(3,:)=OG_SI(3,:);
          else
          end
          end
          Current_SI(2,:)=OG_SI(2,:);
      end
      Current_SI(1,:)=OG_SI(1,:);
end

end
%Attention this system doesnt work for non utilty for either end case
function [current_minimum,min_Inputs]=...
checkEDcallback(obj,current_inputs,currentindex,...
                        current_minimum,min_Inputs,original_user_inputs)
if size(original_user_inputs,1)==currentindex
```

```
2289     pinch_Calculate(obj,current_inputs(1:size(obj.hotinfo,1),:),...
2290         current_inputs((size(obj.hotinfo,1)+1):end,:))
2291     if (obj.hotDuty>obj.howaccurate)&&(obj.coldDuty>obj.howaccurate)
2292     hex_Calculate(obj)
2293     exergy_Calculate(obj)
2294     if (obj.ex_DL_log(3)<current_minimum)
2295         current_minimum=obj.ex_DL_log(3);
2296         min_Inputs=current_inputs;
2297     else
2298     end
2299     else
2300     end
2301
2302 else
2303 end
2304 end
2305 function optimizedDrawingcallback(obj)
2306 obj.FinalFigure = figure('name','Final analysis','numbertitle',...
2307     'off','ToolBar','none','MenuBar','None','Units','pixels',...
2308     'Position',[0 50 1600 980],'Resize','off');
2309 obj.hotFinalTable = uitable('Parent',obj.FinalFigure,...
2310     'Units','normalized',...
2311             'Data',obj.opt_hots,...
2312     'ColumnEditable',[false false false false false],...
2313     'Position',[1300/1600 810/980 265/1600 112/980],...
2314     'ColumnName',{'Tlow','Thigh','flowrate','"heat of change"'},...
2315     'ColumnWidth',{40,40,55,97});
2316 obj.coldFinalTable = uitable('Parent',obj.FinalFigure,...
2317     'Units','normalized',...
2318             'Data',obj.opt_colds,...
2319     'ColumnEditable',[false false false false false],...
2320     'Position',[1300/1600 600/980 267/1600 112/980],...
2321     'ColumnName',{'Tlow','Thigh','flowrate','"heat of change"'},...
2322     'ColumnWidth',{40,40,55,97});
2323 obj.hotFinalText = annotation('textbox',...
2324     'String',' OPTIMIZED HOT STREAMS',...
2325     'Color',[1 0 0],'FontWeight','bold',...
2326     'Position',[1320/1600 930/980 250/1600 25/980],...
2327     'Units','normalized','EdgeColor',0.9411*[1 1 1]);
2328 obj.coldFinalText = annotation('textbox',...
2329     'String','OPTIMIZED COLD STREAMS',...
2330     'Color',[0 0 1],'FontWeight','bold',...
2331     'Position',[1320/1600 720/980 250/1600 25/980],...
2332     'Units','normalized','EdgeColor',0.9411*[1 1 1]);
2333 obj.pinchAxesFinal = axes('Units','normalized',...
2334     'Position',[100/1600 550/980 500/1600 390/980],...
2335     'Parent',obj.FinalFigure);
2336 obj.pinch_titleFinal  = annotation('textbox','LineStyle','none',...
2337    'String','Composite Curve','FontWeight','bold',...
2338    'Position',...
2339    [270/1600 925/980 200/1600 40/980],'Units','normalized');
2340 obj.Gcc_AxesFinal = axes('Units','normalized',...
```

```
2341     'Position',[800/1600 550/980 360/1600 390/980],...
2342     'Parent',obj.FinalFigure);
2343 obj.Gcc_titleFinal  = annotation('textbox','LineStyle','none',...
2344    'String','Grand Composite Curve','FontWeight','bold',...
2345    'Position',[900/1600 925/980 200/1600 40/980],...
2346    'Units','normalized');
2347 obj.Ecc_AxesFinal = axes('Units','normalized','Position',...
2348     [100/1600 35/980 500/1600 390/980],'Parent',obj.FinalFigure);
2349 obj.Ecc_titleFinal = annotation('textbox','LineStyle','none',...
2350    'String','Exergy Composite Curve','FontWeight','bold',...
2351    'Position',[270/1600 420/980 200/1600 30/980],...
2352    'Units','normalized');
2353 obj.GEccAxesFinal = axes('Units','normalized',...
2354     'Position',[800/1600 35/980 360/1600 390/980],...
2355     'Parent',obj.FinalFigure);
2356 obj.Gecc_titleFinal = annotation('textbox','LineStyle','none',...
2357    'String','Grand Exergy Composite Curve','FontWeight','bold',...
2358    'Position',[900/1600 420/980 300/1600 30/980],...
2359    'Units','normalized');
2360 obj.xlabelpinchFinal = annotation('textbox','LineStyle','none',...
2361    'FontAngle','italic','String','Temperature(^{\circ}C)',...
2362    'Position',[5/1600 940/980 300/1600 30/980],...
2363    'Units','normalized');
2364 obj.ylabelpinchFinal = annotation('textbox','LineStyle','none',...
2365    'FontAngle','italic','String','Enthalpy(kW)',...
2366    'Position',[600/1600 260/500 400/1600 30/500],...
2367    'Units','normalized');
2368 obj.xlabelGccFinal = annotation('textbox','LineStyle','none',...
2369    'FontAngle','italic','String','Temperature(^{\circ}C)',...
2370    'Position',[700/1600 940/980 300/1600 30/980],...
2371    'Units','normalized');
2372 obj.ylabelGccFinal = annotation('textbox','LineStyle','none',...
2373    'FontAngle','italic','String','Enthalpy(kW)',...
2374    'Position',[1160/1600 260/500 400/1600 30/500],...
2375    'Units','normalized');
2376 obj.xlabelEccFinal = annotation('textbox','LineStyle','none',...
2377    'FontAngle','italic','String','Carnot Factor(\eta)',...
2378    'Position',[5/1600 420/980 300/1600 30/980],...
2379    'Units','normalized');
2380 obj.ylabelEccFinal = annotation('textbox','LineStyle','none',...
2381    'FontAngle','italic','String','Enthalpy(kW)',...
2382    'Position',[600/1600 0.1/500 400/1600 30/500],...
2383    'Units','normalized');
2384 obj.xlabelGEccFinal = annotation('textbox','LineStyle','none',...
2385    'FontAngle','italic','String','Carnot Factor(\eta)',...
2386    'Position',[700/1600 420/980 300/1600 30/980],...
2387    'Units','normalized');
2388 obj.ylabelGEccFinal = annotation('textbox','LineStyle','none',...
2389    'FontAngle','italic','String','Enthalpy(kW)',...
2390    'Position',[1160/1600 0.1/500 400/1600 30/500],...
2391    'Units','normalized');
2392
```

```
2393 cla(obj.pinchAxesFinal);
2394 axes(obj.pinchAxesFinal);
2395 plot(obj.hotCoordinates,obj.hotBreaks,'r');
2396 hold on;
2397 plot(obj.hotCoordinates,obj.hotBreaks-obj.Thalve,'--m');
2398 plot(obj.coldCoordinates+obj.coldDuty,obj.coldBreaks,'b');
2399 plot(obj.coldCoordinates+obj.coldDuty,...
2400                                       obj.coldBreaks+obj.Thalve,'--g');
2401 text(0.025,0.05,['Cold Utilities = ',num2str(obj.coldDuty)],...
2402                                                 'Units','normalized');
2403 text(0.7,.97,['Hot Utilities = ',num2str(obj.hotDuty)],'Units',...
2404                                                        'normalized');
2405 cla(obj.Gcc_AxesFinal);
2406 axes(obj.Gcc_AxesFinal);
2407 plot(obj.cascade,obj.allBreaks,'k');
2408
2409 cla(obj.Ecc_AxesFinal);
2410 axes(obj.Ecc_AxesFinal);
2411 plot(obj.hotCoordinates,obj.E_level_hot,'r');
2412 hold on;
2413 plot(obj.coldCoordinates+obj.coldDuty,obj.E_level_cold,'b');
2414
2415 cla(obj.GEccAxesFinal);
2416 axes(obj.GEccAxesFinal);
2417 plot(obj.cascade,obj.E_level_grand,'k');
2418 print(obj.FinalFigure,'4st_FinalGUI','-depsc2')
2419 end
2420     end
2421 end
```

A.2 Stream Class

```
% built for CHEN 4570 (CU Boulder) by Scott Rowe, all rights reserved
classdef stream < handle


%--------------------------MURAT_Comments;--------------------------
%
% Value Class: Default one, variables independent, changing in some
% will not affect the other
%
% Handle Class: It is a built in superclass, first line here is the
% example for notation. In previous one the data was copied then
% assigned to another variable but here it is e reference
% that points out assigned variable.
% Also in terms of inheritence purposes same super class of handle
% will be used for pinch class also
% --------------------------MURAT_Finishes;--------------------------


    properties


% --------------------------MURAT_Comments;--------------------------
%
% All of these properties can be considered in mol or kg basis.
% Besides, flow can be per time or total
%
% Phase shows whether or not phase change occurs in that interval as
% logical(0 or 1). This model requires designer to treat phase changin
% streams in a special way(My solution); Devide stream temperature
% continium into 3 sections which are before phase change during phase
% change and after phase change
% --------------------------MURAT_Finishes;--------------------------


        Tlo  = {};    % lowest  stream temperature "K"
        Thi  = {};    % highest stream temperature "K"
        flow = {};    % material flow ( "mol/sec" ) or ( "kg/sec" )
        heat = {};    % specific heat capacity ( "kJ/(mol*K)" )  or
                      % ( "J/(kg*K)" )
                      % or enthalpy of change ( "J/kg" )
        CP   = {};    % heat capacity  ( "J/sec*K" ) or ( "J/sec" )
        phase = [];
        hot = [];
    end
    methods


%     --------------------------MURAT_Comments;--------------------------
%     Object of obj holds all variables such as high and low temperatures,
%     heat capacities, flow rates etc. For gaining or loosing heat hot
%     variable is used.
%     --------------------------MURAT_Finishes;--------------------------
```

```
53
54
55          % inputs = [ Tlo Thi flow heat]
56          % hot = -1 if hot stream, +1 if cold stream
57          function obj = stream(inputs,hot)
58             obj.Tlo  = inputs(1);
59             obj.Thi  = inputs(2);
60             if inputs(1) > inputs(2)
61                  obj.Tlo  = inputs(2);
62                  obj.Thi  = inputs(1);
63             end
64             obj.flow = abs(inputs(3));     % coerce absolute flows
65             obj.heat = abs(inputs(4));     % coerce absolute energies
66             obj.CP = obj.flow*obj.heat;
67             obj.phase = 0;
68             if obj.Tlo == obj.Thi
69                 obj.phase = 1;
70             end
71             obj.hot = hot;
72          end
73          % return H, source/sink from stream on an arbitrary interval
74          % temps = [Tlower Tupper], extrenum of interval
75          % Tshift = negative # for hots, positive # for colds
76          %          absolute magnitude is halve the pinch temperature
77
78  % --------------------------MURAT_Comments;-------------------------
79  % nargin returns the number of function input arguments given in the
80  % call to the currently executing function
81  % First part in "H" definition is written instead of a new if/else
82  % section=logical
83  % Ts here expresses two element matrix with first two smallest
84  % temperatures. Conditional checks;
85  % This demand function calculates energies for each stream in
86  % temperature intervals. Loops constructed in way that; all streams're
87  % checked in related temperature interval
88  % --------------------------MURAT_Finishes;-------------------------
89
90
91  function H = demand(obj,Ts,Tshift)
92      H = 0;
93      if nargin == 3
94  Ts = Ts - Tshift;                               % unshift Ts
95  H  =     (~obj.phase)*(~(Ts(1) > obj.Thi))*(~(Ts(2) < obj.Tlo))*...
96           (min(Ts(2),obj.Thi) - max(Ts(1),obj.Tlo))*obj.flow*obj.heat...
97         + (obj.phase)*...
98            (Ts(2) == obj.Thi && Ts(1) == obj.Tlo)*...
99            (obj.flow*obj.heat);
100 % H = (sensible heat?)*(Ts includes stream?)*(Ts includes stream?)*...
101 %           (calculate sensible heat contribution)...
102 %          + (phase change?)*...
103 %           (Ts includes stream?)*(Ts includes stream?)*...
104 %           (phase heat)
```

```
105     elseif nargin == 2 % means Tshift not given
106 H  =    (~obj.phase)*(~(Ts(1) > obj.Thi))*(~(Ts(2) < obj.Tlo))*...
107         (min(Ts(2),obj.Thi) - max(Ts(1),obj.Tlo))*obj.flow*obj.heat...
108       + (obj.phase)*...
109         (Ts(2) == obj.Thi && Ts(1) == obj.Tlo)*...
110         (obj.flow*obj.heat);
111 % H =(sensible heat?)*(Ts includes stream?)*(Ts includes stream?)*...
112 %          (calculate sensible heat contribution)...
113 %        + (phase change?)*...
114 %          (Ts includes stream?)*(Ts includes stream?)*...
115 %          (phase heat)
116     else
117        % always a positive number
118        H =   (~obj.phase)*...
119              (obj.Thi - obj.Tlo)*obj.flow*obj.heat...
120            + (obj.phase)*...
121              (obj.flow*obj.heat);
122      % H =      (sensible heat?)*...
123      %          (calculate sensible heat contribution)...
124      %        + (phase change?)*...
125      %          (phase heat)
126     end
127     H = obj.hot*H;  % adjust the sign
128                     % hot streams surrender heat (neg #)
129                     % cold streams sink heat (pos #)
130 end
131     end
132 end
```

APPENDIX B

STREAM ALTERNATIVES TABLES

B.1 4 stream Alternative Tables

Table B.1: Alternative stream information: *Low temperature, high temperature, mass flow rate, specific heat*

Stream Number	T_{low}	T_{high}	$\dot{m}$	c_p
1	60	170	1.000	3.0
	60	170	0.800	3.0
	60	170	1.200	3.0
	49	159	0.800	3.0
	49	159	1.000	3.0
	49	159	1.200	3.0
	71	181	1.000	3.0
	71	181	0.800	3.0
	71	181	1.200	3.0
2	30	150	1.000	1.5
	30	150	1.000	1.5
	30	150	1.000	1.5
	30	150	1.000	1.5
	30	150	1.000	1.5
	30	150	1.000	1.5
	30	150	1.000	1.5
	30	150	1.000	1.5
	30	150	1.000	1.5

Continued on next page

Table B.1 – *Continued from previous page*

Stream Number	T_{low}	T_{high}	$\dot{m}$	c_p
	30	150	1.000	1.5
3	20	135	1.000	2.0
	20	135	0.800	3.0
	20	135	1.200	2.0
	8.5	123.5	1.000	3.0
	8.5	123.5	0.800	2.0
	8.5	123.5	1.200	3.0
	31.5	146.5	1.000	2.0
	31.5	146.5	0.800	3.0
	31.5	146.5	1.200	2.0
4	80	140	1.000	2.0
	80	140	0.800	3.0
	80	140	1.200	2.0
	74	134	0.800	3.0
	74	134	1.000	3.0
	74	134	1.200	3.0
	86	146	1.000	3.0
	86	146	0.800	3.0
	86	146	1.200	3.0

B.1.1 Case Study Alternative Tables

Table B.2: All of the alternative stream information: *Low temperature, high temperature, mass flow rate, specific heat*

Stream Number	T_{low}	T_{high}	$\dot{m}$	c_p
1	75	180	15	2
	75	180	15	2
	75	180	15	2
	75	180	15	2

Stream Number	T_{low}	T_{high}	$\dot{m}$	c_p
	75	180	15	2
	75	180	15	2
	75	180	15	2
	75	180	15	2
	75	180	15	2
2	60	240	20	2
	60	240	16	2
	60	240	24	2
	42	222	20	2
	42	222	16	2
	42	222	24	2
	78	258	20	2
	78	258	16	2
	78	258	24	2
3	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
4	120	260	7.5	2
	120	260	6	2
	120	260	9	2
	106	246	7.5	2
	106	246	6	2
	106	246	9	2
	134	274	7.5	2
	134	274	6	2

Stream Number	T_{low}	T_{high}	$\dot{m}$	c_p
	134	274	9	2
5	40	139	12.5	2
	40	139	10	2
	40	139	15	2
	30.1	129.1	12.5	2
	30.1	129.1	10	2
	30.1	129.1	15	2
	49.9	148.9	12.5	2
	49.9	148.9	10	2
	49.9	148.9	15	2
	80	190	10	2
	80	190	10	2
	80	190	10	2
	80	190	10	2
6	80	190	10	2
	80	190	10	2
	80	190	10	2
	80	190	10	2
	80	190	10	2

Table B.3: Selected alternative stream information: *Low temperature, high temperature, mass flow rate, specific heat*

Stream Number	T_{low}	T_{high}	$\dot{m}$	c_p
1	75	180	15	2
	75	180	15	2
	75	180	15	2
	75	180	15	2
	75	180	15	2
	75	180	15	2
	75	180	15	2

Stream Number	T_{low}	T_{high}	$\dot{m}$	c_p
	75	180	15	2
	75	180	15	2
2	60	240	20	2
	60	240	16	2
	60	240	24	2
	42	222	20	2
	42	222	16	2
	42	222	24	2
	78	258	20	2
	78	258	16	2
	78	258	24	2
3	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
	40	230	10	2
4	120	260	7.5	2
	120	260	6	2
	120	260	9	2
	106	246	7.5	2
	106	246	6	2
	106	246	9	2
	134	274	7.5	2
	134	274	6	2
	134	274	9	2
5	40	139	12.5	2
	40	139	10	2

Stream Number	T_{low}	T_{high}	$\dot{m}$	c_p
	40	139	15	2
	30.1	129.1	12.5	2
	30.1	129.1	10	2
	30.1	129.1	15	2
	49.9	148.9	12.5	2
	49.9	148.9	10	2
	49.9	148.9	15	2
	80	190	10	2
	80	190	10	2
	80	190	10	2
	80	190	10	2
6	80	190	10	2
	80	190	10	2
	80	190	10	2
	80	190	10	2
	80	190	10	2

APPENDIX C

HEX AND UTILITY INFORMATION FOR DIFFERENT SCENARIOS

C.1 Case Study Scenarios

Table C.1: Heat Exchanger Information Below Pinch for Case Study: 8.73% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	80	106	10	2	-520
	1	90	116	10	2	520
2	5	80	106	5	2	-260
	1	90	116	5	2	260
3	5	80	106	5	2	-260
	2	90	116	5	2	260
4	6	80	106	10	2	-520
	2	90	116	10	2	520
5	5	65	80	1	2	-30
	2	101	116	1	2	30
6	3	69	80	1	2	-22
	2	90	101	1	2	22
7	3	65	80	9	2	-270
	1	75	90	9	2	270
8	5	65	80	6	2	-180

Continued on next page

Table C.1 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	1	75	90	6	2	180
9	5	65	80	3	2	-90
	2	75	90	3	2	90
10	3	65	69	1	2	-8
	2	86	90	1	2	8
11	3	50	65	10	2	-300
	2	75	90	10	2	300
12	5	50	65	2	2	-60
	2	75	90	2	2	60
13	5	54	65	1	2	-22
	2	75	86	1	2	22
14	5	40	65	7	2	-350
	2	50	75	7	2	350
15	5	40	54	1	2	-28
	2	61	75	1	2	28
16	3	40	50	8	2	-160
	2	65	75	8	2	160
17	3	40	50	2	2	-40
	2	55	65	2	2	40
18	5	40	50	2	2	-40
	2	55	65	2	2	40

Table C.2: Heat Exchanger Information Above Pinch for Case Study: 8.73% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	106	139	7.5	2	-495
	1	116	149	7.5	2	495
2	3	106	139	7.5	2	-495
	1	116	149	7.5	2	495
3	3	106	139	2.5	2	-165
	2	116	149	2.5	2	165
4	6	106	139	10	2	-660
	2	116	149	10	2	660
5	5	106	139	3.5	2	-231
	2	116	149	3.5	2	231
6	5	106	137	6.5	2	-403
	1	149	180	6.5	2	403
7	5	137	139	6.5	2	-26
	1	149	151	6.5	2	26
8	4	139	170	2	2	-124
	1	149	180	2	2	124
9	4	139	170	5.5	2	-341
	2	149	180	5.5	2	341
10	3	139	170	10	2	-620
	2	149	180	10	2	620
11	6	139	170	0.5	2	-31
	2	149	180	0.5	2	31
12	6	139	168	6.5	2	-377
	1	151	180	6.5	2	377
13	6	139	159	3	2	-120

Continued on next page

Table C.2 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	2	180	200	3	2	120
14	6	159	170	3	2	-66
	2	180	191	3	2	66
15	6	168	170	6.5	2	-26
	2	180	182	6.5	2	26
16	4	170	190	3.5	2	-140
	2	180	200	3.5	2	140
17	4	170	188	4	2	-144
	2	182	200	4	2	144
18	3	170	188	2.5	2	-90
	2	182	200	2.5	2	90
19	3	170	179	3	2	-54
	2	191	200	3	2	54
20	3	170	190	4.5	2	-180
	2	200	220	4.5	2	180
21	6	170	190	10	2	-400
	2	200	220	10	2	400
22	3	179	190	1.5	2	-33
	2	200	211	1.5	2	33
23	3	179	190	1.5	2	-33
	2	211	222	1.5	2	33
24	3	188	190	2.5	2	-10
	2	220	222	2.5	2	10
25	4	188	190	4	2	-16
	2	220	222	4	2	16
26	4	190	192	7.5	2	-30
	2	220	222	7.5	2	30

Continued on next page

Table C.2 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
27	3	190	192	0.5	2	-2
	2	220	222	0.5	2	2

Table C.3: Case Study, 8.73% Exergy Destruction Reduction Scenario: Streams to be cooled by cold utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	2	42	55	2	2	-52
2	2	42	65	4	2	-184
3	2	42	61	1	2	-38
4	2	42	55	2	2	-52
5	2	42	50	7	2	-112

Table C.4: Case Study, 8.73% Exergy Destruction Reduction Scenario: Streams to be heated by hot Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	3	192	230	0.5	2	38
2	3	190	230	9.5	2	760
3	4	192	246	7.5	2	810

Table C.5: Heat Exchanger Information Below Pinch for Case Study: 13.72% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	3	80	120	10	2	-800
	1	90	130	10	2	800
2	5	80	120	5	2	-400
	1	90	130	5	2	400
3	5	80	120	5	2	-400
	2	90	130	5	2	400
4	6	80	120	10	2	-800
	2	90	130	10	2	800
5	5	65	80	1	2	-30
	2	115	130	1	2	30
6	3	65	80	1	2	-30
	2	100	115	1	2	30
7	3	70	80	1	2	-20
	2	90	100	1	2	20

Continued on next page

Table C.5 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
8	5	65	80	9	2	-270
	1	75	90	9	2	270
9	3	65	80	6	2	-180
	1	75	90	6	2	180
10	3	65	80	2	2	-60
	2	75	90	2	2	60
11	3	65	70	1	2	-10
	2	85	90	1	2	10
12	3	50	65	10	2	-300
	2	75	90	10	2	300
13	5	50	65	3	2	-90
	2	75	90	3	2	90
14	5	55	65	1	2	-20
	2	75	85	1	2	20
15	5	40	65	6	2	-300
	2	50	75	6	2	300
16	5	40	55	1	2	-30
	2	60	75	1	2	30
17	3	40	50	9	2	-180
	2	65	75	9	2	180
18	5	40	50	3	2	-60
	2	55	65	3	2	60
19	3	40	50	1	2	-20
	2	55	65	1	2	20

Table C.6: Heat Exchanger Information Above Pinch for Case Study: 13.72% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	120	139	7.5	2	-285
	1	130	149	7.5	2	285
2	3	120	139	7.5	2	-285
	1	130	149	7.5	2	285
3	3	120	139	2.5	2	-95
	2	130	149	2.5	2	95
4	6	120	139	10	2	-380
	2	130	149	10	2	380
5	5	120	139	3.5	2	-133
	2	130	149	3.5	2	133
6	5	120	139	6.5	2	-247
	1	149	168	6.5	2	247
7	4	139	170	7.5	2	-465
	1	149	180	7.5	2	465
8	3	139	170	1	2	-62
	1	149	180	1	2	62
9	3	139	170	9	2	-558
	2	149	180	9	2	558
10	6	139	170	7	2	-434
	2	149	180	7	2	434
11	6	139	151	3	2	-72
	1	168	180	3	2	72
12	6	151	158	3	2	-42
	1	168	175	3	2	42
13	6	158	170	0.5	2	-12

Continued on next page

Table C.6 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	1	168	180	0.5	2	12
14	6	158	163	2.5	2	-25
	1	175	180	2.5	2	25
15	6	163	168	0.5	2	-5
	1	175	180	0.5	2	5
16	6	163	170	2	2	-28
	2	180	187	2	2	28
17	6	168	170	0.5	2	-2
	2	180	182	0.5	2	2
18	4	170	190	7.5	2	-300
	2	180	200	7.5	2	300
19	3	170	190	6	2	-240
	2	180	200	6	2	240
20	3	170	188	0.5	2	-18
	2	182	200	0.5	2	18
21	3	170	183	2	2	-52
	2	187	200	2	2	52
22	3	170	190	1.5	2	-60
	2	200	220	1.5	2	60
23	6	170	190	10	2	-400
	2	200	220	10	2	400
24	3	183	190	2	2	-28
	2	200	207	2	2	28
25	3	188	190	0.5	2	-2
	2	200	202	0.5	2	2
26	4	190	212	2	2	-88
	2	200	222	2	2	88

Continued on next page

Table C.6 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
27	4	190	210	0.5	2	-20
	2	202	222	0.5	2	20
28	4	190	205	2	2	-60
	2	207	222	2	2	60
29	4	190	192	3	2	-12
	2	220	222	3	2	12
30	3	190	192	8.5	2	-34
	2	220	222	8.5	2	34

Table C.7: Case Study, 13.72% Exergy Destruction Reduction Scenario: Streams to be cooled by cold utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	2	42	55	1	2	-26
2	2	42	65	5	2	-230
3	2	42	60	1	2	-36
4	2	42	55	3	2	-78
5	2	42	50	6	2	-96

Table C.8: Case Study, 13.72% Exergy Destruction Reduction Scenario: Streams to be heated by hot Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	3	192	230	8.5	2	646
2	3	190	230	1.5	2	120
3	4	192	260	3	2	408
4	4	205	260	2	2	220
5	4	210	260	0.5	2	50
6	4	212	260	2	2	192

Table C.9: Heat Exchanger Information Below Pinch for Case Study: 18.23% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	5	80	106	15	2	-780
	1	90	116	15	2	780
2	3	80	106	10	2	-520
	2	90	116	10	2	520
3	6	80	106	10	2	-520
	2	90	116	10	2	520
4	5	65	80	15	2	-450
	1	75	90	15	2	450
5	3	65	80	10	2	-300

Continued on next page

Table C.9 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	2	75	90	10	2	300
6	5	50	65	10	2	-300
	2	75	90	10	2	300
7	3	40	65	10	2	-500
	2	50	75	10	2	500
8	5	40	65	5	2	-250
	2	50	75	5	2	250
9	5	40	50	5	2	-100
	2	65	75	5	2	100
10	5	40	50	5	2	-100
	2	55	65	5	2	100

Table C.10: Heat Exchanger Information Above Pinch for Case Study: 18.23% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	106	139	9	2	-594
	1	116	149	9	2	594
2	3	106	139	6	2	-396
	1	116	149	6	2	396
3	3	106	139	4	2	-264
	2	116	149	4	2	264
4	6	106	139	10	2	-660
	2	116	149	10	2	660
5	5	106	139	6	2	-396
	2	116	149	6	2	396

Continued on next page

Table C.10 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
6	5	106	137	9	2	-558
	1	149	180	9	2	558
7	5	137	139	6	2	-24
	1	149	151	6	2	24
8	5	137	139	3	2	-12
	2	149	151	3	2	12
9	4	139	170	9	2	-558
	2	149	180	9	2	558
10	3	139	170	8	2	-496
	2	149	180	8	2	496
11	3	139	168	2	2	-116
	2	151	180	2	2	116
12	6	139	168	1	2	-58
	2	151	180	1	2	58
13	6	139	168	6	2	-348
	1	151	180	6	2	348
14	6	139	159	3	2	-120
	2	180	200	3	2	120
15	6	159	170	3	2	-66
	2	180	191	3	2	66
16	3	168	170	2	2	-8
	2	180	182	2	2	8
17	6	168	170	7	2	-28
	2	180	182	7	2	28
18	4	170	190	5	2	-200
	2	180	200	5	2	200
19	4	170	188	4	2	-144

Continued on next page

Table C.10 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
	2	182	200	4	2	144
20	3	170	188	5	2	-180
	2	182	200	5	2	180
21	3	170	179	3	2	-54
	2	191	200	3	2	54
22	3	170	190	2	2	-80
	2	200	220	2	2	80
23	6	170	190	10	2	-400
	2	200	220	10	2	400
24	3	179	190	3	2	-66
	2	200	211	3	2	66
25	4	188	190	4	2	-16
	2	200	202	4	2	16
26	3	188	190	1	2	-4
	2	200	202	1	2	4
27	3	188	190	4	2	-16
	2	202	204	4	2	16
28	4	190	210	1	2	-40
	2	202	222	1	2	40
29	4	190	208	4	2	-144
	2	204	222	4	2	144
30	4	190	201	3	2	-66
	2	211	222	3	2	66
31	4	190	192	1	2	-4
	2	220	222	1	2	4
32	3	190	192	10	2	-40
	2	220	222	10	2	40

Continued on next page

Table C.10 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
33	4	192	194	1	2	-4
	2	220	222	1	2	4

Table C.11: Case Study, 18.23% Exergy Destruction Reduction Scenario: Streams to be cooled by cold utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	2	42	55	5	2	-130
2	2	42	50	15	2	-240

Table C.12: Case Study, 18.23% Exergy Destruction Reduction Scenario: Streams to be heated by hot Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	3	192	230	8.5	2	646
2	3	190	230	1.5	2	120
3	4	192	260	3	2	408
4	4	205	260	2	2	220
5	4	210	260	0.5	2	50

Table C.13: Heat Exchanger Information Below Pinch for Case Study: 31.83% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	80	106	10	2	-520
	1	90	116	10	2	520
2	5	80	106	5	2	-260
	1	90	116	5	2	260
3	5	80	106	5	2	-260
	2	90	116	5	2	260
4	6	80	106	10	2	-520
	2	90	116	10	2	520
5	5	65	80	1	2	-30
	2	101	116	1	2	30
6	3	69	80	1	2	-22
	2	90	101	1	2	22
7	3	65	80	9	2	-270
	1	75	90	9	2	270
8	5	65	80	6	2	-180
	1	75	90	6	2	180
9	5	65	80	3	2	-90
	2	75	90	3	2	90
10	3	65	69	1	2	-8
	2	86	90	1	2	8
11	3	50	65	10	2	-300
	2	75	90	10	2	300
12	5	50	65	2	2	-60
	2	75	90	2	2	60
13	5	54	65	1	2	-22

Continued on next page

Table C.13 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	2	75	86	1	2	22
14	5	40	65	7	2	-350
	2	50	75	7	2	350
15	5	40	54	1	2	-28
	2	61	75	1	2	28
16	3	40	50	8	2	-160
	2	65	75	8	2	160
17	3	40	50	2	2	-40
	2	55	65	2	2	40
18	5	40	50	2	2	-40
	2	55	65	2	2	40

Table C.14: Heat Exchanger Information Above Pinch for Case Study: 31.83% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	106	139	6	2	-396
	1	116	149	6	2	396
2	3	106	139	9	2	-594
	1	116	149	9	2	594
3	3	106	139	1	2	-66
	2	116	149	1	2	66
4	6	106	139	10	2	-660
	2	116	149	10	2	660
5	5	106	139	5	2	-330
	2	116	149	5	2	330

Continued on next page

Table C.14 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
6	5	106	137	5	2	-310
	1	149	180	5	2	310
7	5	137	139	5	2	-20
	1	149	151	5	2	20
8	4	139	170	5	2	-310
	1	149	180	5	2	310
9	4	139	170	1	2	-62
	2	149	180	1	2	62
10	3	139	170	10	2	-620
	2	149	180	10	2	620
11	6	139	170	5	2	-310
	2	149	180	5	2	310
12	6	139	168	5	2	-290
	1	151	180	5	2	290
13	6	168	170	5	2	-20
	2	180	182	5	2	20
14	4	170	190	6	2	-240
	2	180	200	6	2	240
15	3	170	190	5	2	-200
	2	180	200	5	2	200
16	3	170	188	5	2	-180
	2	182	200	5	2	180
17	6	170	190	10	2	-400
	2	200	220	10	2	400
18	3	188	190	5	2	-20
	2	200	202	5	2	20
19	4	190	212	1	2	-44

Continued on next page

Table C.14 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	2	200	222	1	2	44
20	4	190	210	5	2	-200
	2	202	222	5	2	200
21	3	190	192	10	2	-40
	2	220	222	10	2	40

Table C.15: Case Study, 31.83% Exergy Destruction Reduction Scenario: Streams to be cooled by cold utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	2	42	55	2	2	-52
2	2	42	65	4	2	-184
3	2	42	61	1	2	-38
4	2	42	55	2	2	-52
5	2	42	50	7	2	-112

Table C.16: Case Study, 31.83% Exergy Destruction Reduction Scenario: Streams to be heated by hot Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	3	192	230	10	2	760
2	4	210	246	5	2	360
3	4	212	246	1	2	68

Table C.17: Heat Exchanger Information Below Pinch for Case Study: 41.33% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg\cdot K)$	ΔH (kW)
1	5	80	106	15	2	-780
	1	90	116	15	2	780
2	3	80	106	10	2	-520
	2	90	116	10	2	520
3	6	80	106	10	2	-520
	2	90	116	10	2	520
4	5	65	80	15	2	-450
	1	75	90	15	2	450
5	3	65	80	10	2	-300
	2	75	90	10	2	300
6	5	50	65	10	2	-300
	2	75	90	10	2	300
7	3	40	65	10	2	-500
	2	50	75	10	2	500

Continued on next page

Table C.17 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
8	5	40	65	5	2	-250
	2	50	75	5	2	250
9	5	40	50	5	2	-100
	2	65	75	5	2	100
10	5	40	50	5	2	-100
	2	55	65	5	2	100

Table C.18: Heat Exchanger Information Above Pinch for Case Study: 41.33% Exergy Destruction Reduction Scenario

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	4	106	139	7.5	2	-495
	1	116	149	7.5	2	495
2	3	106	139	7.5	2	-495
	1	116	149	7.5	2	495
3	3	106	139	2.5	2	-165
	2	116	149	2.5	2	165
4	6	106	139	10	2	-660
	2	116	149	10	2	660
5	5	106	139	7.5	2	-495
	2	116	149	7.5	2	495
6	5	106	137	7.5	2	-465
	1	149	180	7.5	2	465
7	5	137	139	7.5	2	-30
	1	149	151	7.5	2	30
8	4	139	170	7.5	2	-465

Continued on next page

Table C.18 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
	2	149	180	7.5	2	465
9	3	139	170	10	2	-620
	2	149	180	10	2	620
10	6	139	170	2.5	2	-155
	2	149	180	2.5	2	155
11	6	139	168	7.5	2	-435
	1	151	180	7.5	2	435
12	6	168	170	7.5	2	-30
	2	180	182	7.5	2	30
13	4	170	190	7.5	2	-300
	2	180	200	7.5	2	300
14	3	170	190	5	2	-200
	2	180	200	5	2	200
15	3	170	188	5	2	-180
	2	182	200	5	2	180
16	6	170	188	2.5	2	-90
	2	182	200	2.5	2	90
17	6	170	190	7.5	2	-300
	2	200	220	7.5	2	300
18	6	188	190	2.5	2	-10
	2	200	202	2.5	2	10
19	3	188	190	5	2	-20
	2	200	202	5	2	20
20	4	190	212	5	2	-220
	2	200	222	5	2	220
21	4	190	210	2.5	2	-100
	2	202	222	2.5	2	100

Continued on next page

Table C.18 – *Continued from previous page*

HEX Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
22	3	190	210	5	2	-200
	2	202	222	5	2	200
23	3	190	192	5	2	-20
	2	220	222	5	2	20
24	3	192	194	2.5	2	-10
	2	220	222	2.5	2	10

Table C.19: Case Study, 41.33% Exergy Destruction Reduction Scenario: Streams to be cooled by cold utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	2	42	55	5	2	-130
2	2	42	50	15	2	-240

Table C.20: Case Study, 41.33% Exergy Destruction Reduction Scenario: Streams to be heated by hot Utilities

Utility Number	Stream Number	T_{low} (°C)	T_{high} (°C)	$\dot{m}(kg/s)$	$c_p(kJ/kg \cdot K)$	ΔH (kW)
1	3	194	230	2.5	2	180
2	3	192	230	2.5	2	190
3	4	210	246	2.5	2	180
4	3	210	230	5	2	200
5	4	212	246	5	2	340