Model Development and Annual Simulation of the Supercritical Carbon Dioxide Brayton Cycle for Concentrating Solar Power Applications

by

William Seidel

A thesis submitted in partial fulfillment of
the requirements for the degree of

Master of Science
(Mechanical Engineering)

at the
UNIVERSITY OF WISCONSIN – MADISON
2010

Approved by

_____

Professor Sanford A. Klein

_____

Professor Douglas T. Reindl

Date:_____

Abstract

The objective of this research is to assess the suitability of the Supercritical Carbon Dioxide (SCO_2) Brayton cycle for use as the power block for concentrating solar power (CSP) applications. A SCO_2 Brayton module was to be created for use in the National Renewable Energy Lab's (NREL) *Solar Advisor Model* (SAM) for this purpose.

Simplified models were developed for several different Brayton cycle configurations using the *Engineering Equation Solver* (EES) software. The performance of these cycles was validated against the literature and then the different configurations were compared against each other. The choice was made to pursue the Simple Brayton with Regeneration configuration for further, more detailed investigation due to its simplicity and high efficiency.

A detailed model of the Simple Brayton with Regeneration Cycle was developed in EES. The model expanded on the capabilities of the simplified system model, and focused primarily on more detailed heat exchanger modeling. The detailed model was further expanded to assess the feasibility of hybrid and air-cooled Brayton configurations to reduce water use.

The three Brayton cycle configurations were then implemented in FORTRAN for use in annual simulations performed in the TRNSYS transient energy system simulation software. Implementation was accomplished via a linear regression of sets of mapping simulations results from the detailed EES models. Linear regression of these mapping simulation results facilitated the creation of polynomial approximations of system performance as a function of molten salt and ambient air temperatures. The Brayton component was then simulated on an annual basis in the context of a concentrating solar power plant, using a TRNSYS deck configuration based on work by Wagner (2008).

Annual and monthly simulations were performed, and suggest that the SCO_2 Brayton cycle could present an attractive alternative to the Rankine cycle for CSP applications. Annual net efficiencies of 41.6% were found for the 'large' water-cooled configurations, while the hybrid-cooled configuration yielded moderate water savings and annual net efficiencies of 41.4% for the

'large' case. Both of these configurations resulted in greater annual efficiency and energy production than the Rankine cycle modeled by Wagner (2008) when coupled with the same solar field. Annual fossil fuel use was also reduced for both of these cycles when compared against the Rankine cycle.

The air-cooled cycle allowed for total elimination of water use, and annual net efficiencies of up to 39.7% for the 'large' case. Annual energy production for the air-cooled cycle was reduced when compared to the hybrid and water-cooled Brayton cycles, and the Rankine cycle, due to its lower peak capacity reducing its reliance on fossil fuel for thermal energy.

Acknowledgements

First thanks must go to NREL for their financial and intellectual support – without them this work would not have been possible. Mike Wagner's guidance, in the form of both his excellent thesis and his conversations with me, has been invaluable. I'd also like to thank Nate Blair, Craig Turchi, and all the other participants in my conference calls for their questions, comments, and insight throughout this process.

Professor Klein (first alphabetically amongst my advisors) deserves ample mention here, though due to space constraints I will be brief. Saying that he has a lightning-fast mind, a wealth of experience and an extreme ability to focus begins to quantify his abilities as an engineer. But it only begins. If I had a technical problem that needed solving, whatever the field, I'd be confident that he could do it. While he at times challenged my cardiovascular health (both through work-related stress and by once leaving me gasping in his wake when I tried to sociably cool down with him while he was jogging on the track…) I could not have asked for a more educationally productive advisor.

Professor Reindl (runner-up in my advisor's-names-alphabetical-rankings) was steadfast and brilliant in his ability to connect my abstract and perhaps out-of-touch thoughts to actual engineering practice. His insights into the operational realities of the 'real world' were critical for allowing me to produce a thesis that has useful applications in industry. Professor Reindl's constant availability via email (an attribute I bet he considers as much curse as blessing) was also extremely helpful during the late hours I have sometimes kept. I can only hope he continues to be somewhat available for problems that might present themselves to me in future work.

Professor Nellis, along with possessing an exquisitely dry wit that had me and my classmates in stitches during his Heat Transfer and Conduction classes, also manages to be quite the teacher! While my poor penmanship was at times over-challenged while attempting to keep up, his classes were immensely educational, to the extent that I would say any graduate student that misses them is making a mistake. I can't emphasize this point enough: they are *very* good.

Other people who deserve thanks for their contributions include: Matt, Bill, Greg ‘The Rock’ Marsicek, Mandy, Lukas, and Jonathan Goldstein. Ty I won’t thank, because I presume I’ll be seeing him around.

Table of Contents

List of Tables

List of Figures

Nomenclature

Abbreviations

PCHE	Printed Circuit Heat Exchanger
SCO_2	Supercritical Carbon Dioxide
CSP	Concentrating Solar Power
EES	Engineering Equation Solver (Software)
T-S Diagram	Temperature-Entropy Diagram
NREL	National Renewable Energy Lab
SAM	Solar Advisor Model
TRNSYS	Transient Energy System Simulation Tool (Software)
FORTRAN	Formula Translation (Software coding language)
HX	Heat Exchanger
LTR	Low Temperature Regenerator
HTR	High Temperature Regenerator

Variables

UA	Conductance
T	Temperature
ε	Heat exchanger effectiveness
$\dot{q}$	Heat transfer rate
i	Enthalpy
$\dot{m}$	Mass flow rate
N	Number of nodes
$\dot{C}$	Capacitance rate
NTU	Number of transfer units
C_R	Ratio of minimum capacitance rate to maximum capacitance rate

s	Entropy
w	Specific work
η	Efficiency
P	Pressure
r_T	Turbine pressure ratio
r_C	Compressor pressure ratio
rpr	$rpr = \frac{(r_C - 1)}{(r_T - 1)}$
P_{int}	Split expansion cycle intermediate turbine outlet pressure
f	$f = \frac{p_{\text{int}} - p_{low}}{p_{high} - p_{low}}$ for split expansion cycle
c_p	Specific heat
Nu	Nusselt Number
G	Mass flow rate per cross section area [kg/m^2-s]
D	Diameter
D_h	Hydraulic diameter
th	Thickness of heat exchanger web
W_{hot}	Heat exchanger hot side channel width
W_{cold}	Heat exchanger cold side channel width
$T_{H,B}$	Hot side bulk temperature
$T_{H,W}$	Hot side wall temperature
$T_{C,B}$	Cold side bulk temperature
$T_{C,W}$	Cold side wall temperature
N_{ch}	Number of heat exchanger channels
Δx	Heat exchanger node length
f_{fd}	Fully developed friction factor
h	Heat transfer coefficient
k	Thermal conductivity
ff	Fouling factor
C	Form loss coefficient

ΔP	Pressure drop
v	Fluid velocity
ρ	Fluid density
$AREA_{HX}$	Air-cooler heat exchanger area
$AREA_{fr}$	Frontal area of air-cooler
L	Air-cooler length
α	Ratio of air-cooler heat exchanger area to total volume
$\Delta T_{approach}$	Approach temperature
$\dot{W}_{net,normalized}$	Net normalized power
$T_{water,in}$	Cooling water inlet temperature
$T_{water,out}$	Cooling water outlet temperature
$T_{salt,in}$	Molten salt inlet temperature
$T_{salt,out}$	Molten salt outlet temperature
$T_{air,ambient}$	Ambient dry bulb temperature

Subscripts

H	Hot
C	Cold
i	Node i
max	Maximum
min	Minimum
in	inlet
out	outlet
outs	isentropic outlet

1 Introduction

1.1 Background

Increasingly expensive and unpredictable energy sources, along with the enormous impact of fossil fuel use on the global environment have spurred widespread interest in renewable energy. Direct conversion of solar radiation into electricity via photovoltaic panels has long been viewed as an attractive technology, but its high cost, its inherently transient production, and difficulties associated with grid-integration have limited its growth.

Large scale solar-thermal power plants work by concentrating solar radiation to heat a heat transfer fluid. This heat transfer fluid is then used to power a heat engine in much the same way as a fossil fuel power plant. Large scale solar-thermal power plants can utilize high temperature storage to levelized power production during intermittent solar periods or after hours of darkness. The economy of scale and the ability to add storage increases the attractiveness of solar-thermal power production.

Work for this thesis has focused on central receiver plants. These plants are composed of an array of mirrors (heliostats) which focus incident solar radiation on a central 'power tower' as shown in Figure 1-1. Heat transfer fluid is pumped through the power tower and then used to drive a conventional power cycle (Feierabend, 2009).

Figure 1-1: Central Receiver Solar Thermal Power Plants – Abengoa's PS10 and PS20 Plants near Seville, Spain (http://upload.wikimedia.org/wikipedia/commons/2/22/PS20andPS10.jpg)

At the present time, all currently operational central receiver plants use power cycles based on the steam Rankine cycle to extract heat from the heating fluid and produce electricity. This thesis investigates the potential of an alternative technology to the Rankine power cycle - a supercritical carbon dioxide Brayton (SCO_2) cycle.

The Brayton cycle is a type of gas-turbine cycle with many variations which are discussed in Chapter 2 of this thesis. The cycle, in its simplest practical form, is represented in the schematic equipment and temperature-entropy diagram shown in Figure 1-2 and Figure 1-3, respectively.

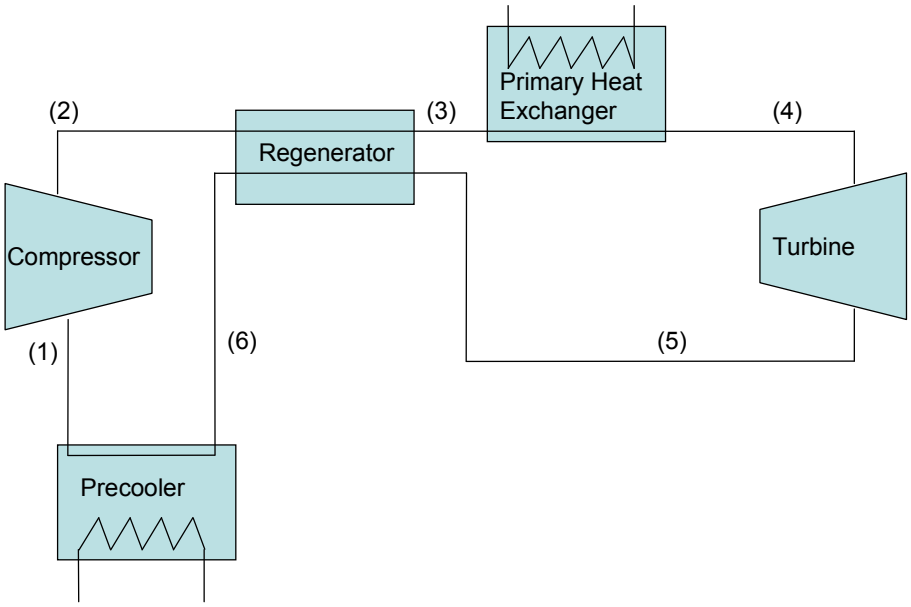

Figure 1-2: Schematic Equipment Diagram of Simple Brayton Cycle w/Regeneration

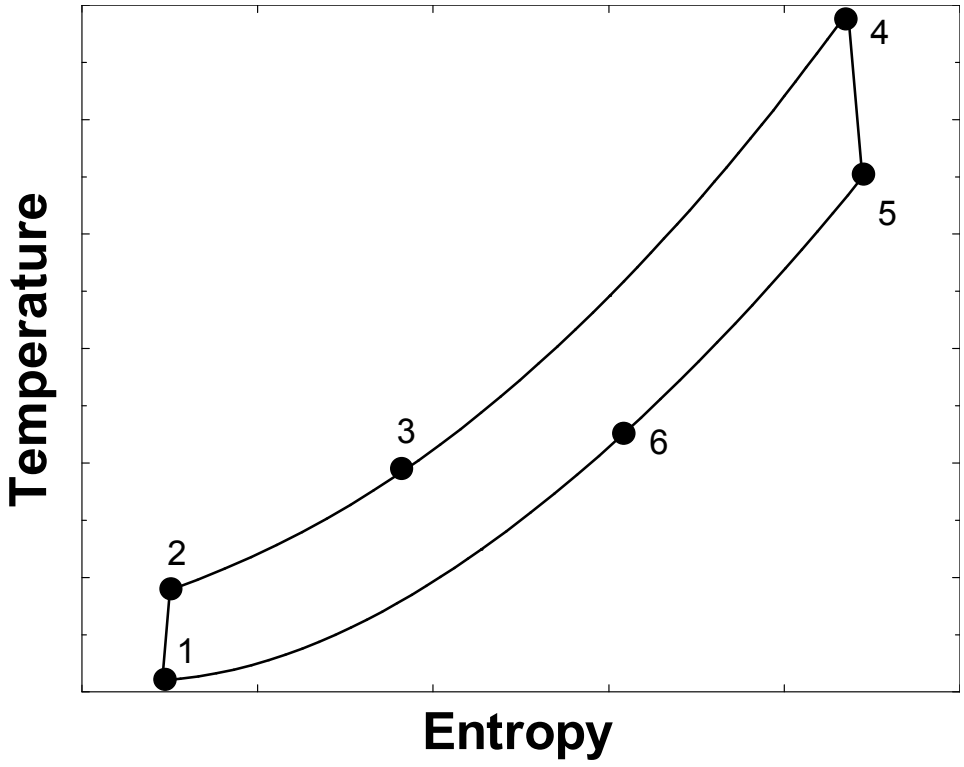

Figure 1-3: Temperature-Entropy Diagram of Simple Brayton w/Regeneration Cycle

The supercritical CO_2 Brayton cycle is of interest both to the solar energy and nuclear energy communities. One characteristic that is driving interest in the SCO_2 cycle is the working fluid operates at or near its critical point (at 304.13 K and 7.38 MPa) where it has extremely high density (Dostal, 2006). The density of carbon dioxide as a function of temperature for a range of pressures at and above the critical point is shown below in Figure 1-4.

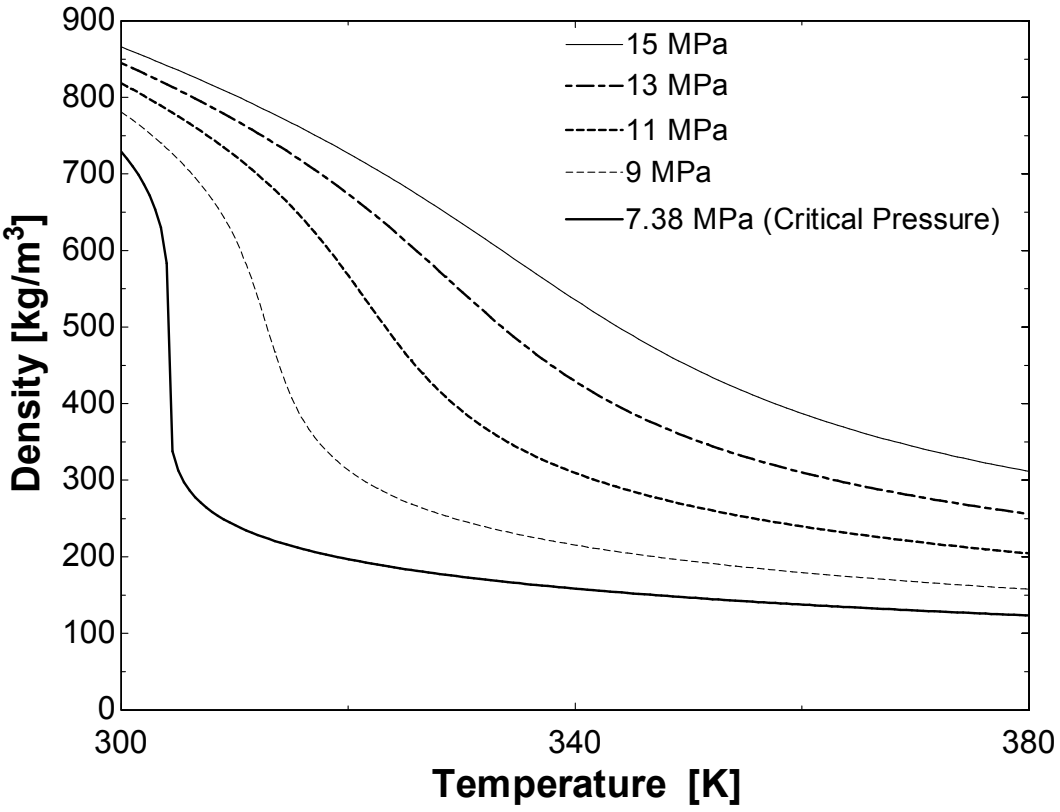

Figure 1-4: CO_2 Density Near Critical Point (Klein, 2010)

High fluid density offers the potential to reduce both compressor power and size; thereby, leading to greater efficiency than an ideal-gas Brayton cycle and greater power density when compared to a superheated Rankine cycle (Hoang, 2009). These characteristics allow for smaller, less expensive equipment, and could make

The SCO_2 Brayton cycle is also seen as attractive due to its heat rejection characteristics. Since the Brayton cycle rejects heat across a range of relatively high temperatures, unlike the Rankine

cycle, there is potential for novel heat rejection strategies, including dry and hybrid cooling. While reductions in water use via dry or hybrid cooling have traditionally come at the expense of performance both in terms of power and efficiency, water conservation is such a critical issue in the arid areas where CSP is a viable technology, any potential reduction is valuable.

1.2 Project Motivation

The National Renewable Energy Lab (NREL) has developed a simulation program designed to facilitate the analysis of solar technologies. The 'Solar Advisor Model' (SAM) allows users to quickly perform schematic-level technical and economic analyses of a potential project. Users can assess whether a given environmental and financial environment is appropriate for the deployment of a number of solar technologies, including solar heating, Concentrating Solar Power (CSP), and photovoltaic (Feierabend, 2009).

The primary objective of this thesis is to create a model capable of predicting the performance of an SCO_2 Brayton cycle for integration into SAM. This model provides an alternative to the Rankine model produced by Wagner (2008). The SCO_2 Brayton model has been implemented as a component in TRNSYS (Klein, 2010) (a transient simulation engine that performs system performance calculations for SAM). This TRNSYS component is written in FORTRAN, and it is based on a cycle performance map created using a more detailed model developed in Engineering Equation Solver (EES).

This thesis discusses the work undertaken to create the SCO_2 Brayton cycle model in EES, use that cycle model to create a performance map, implement that map as a TRNSYS component written in FORTRAN, and finally to use that TRNSYS component to perform annual simulations. The TRNSYS component representing the SCO_2 Brayton cycle will now be incorporated into SAM to allow energy professionals to assess its suitability to their future projects.

1.3 Structure of Thesis

Another goal of this project is to assess the suitability of the Supercritical Carbon Dioxide (SCO_2) Brayton cycle for use as the power block for concentrating solar power (CSP) applications. The assessment is based on analysis performed using cycle models developed as part of this research to simulate the SCO_2 cycle performance over a range of operating conditions. Model results are compared with other results published in the open literature (see Chapter 2). The SCO_2 model is then coupled with existing CSP system components and simulations to evaluate potential performance gains over traditional steam power cycles for operating conditions expected during annual operation.

Chapters 2 and 3 address the methods employed to model the SCO_2 Brayton cycle. Discussion proceeds from investigation into multiple Brayton configurations using a simplified methodology. A specific configuration is then chosen for more in-depth modeling, and the more realistic model is then discussed. Correlations used for heat transfer coefficients and pressure drops are discussed along with turbomachinery, and cycle controls.

Chapter 4 discusses the motivation and implementation of the air-cooled and hybrid-cooled Brayton cycle models. Design considerations specific to these cycles are presented.

Chapter 5 provides an examination of how various design parameters for the SCO_2 Brayton cycle affect performance. The chapter discusses the potential trade-offs between power and efficiency that must be struck in choosing a cycle design. Finally the design parameters for three cycles (one water, one air, and one hybrid-cooled) are presented along with their performance metrics.

Chapter 6 focuses on the implementation of the SCO_2 Brayton model as a TRNSYS component, and on the annual performance of the SCO_2 Brayton cycle in a CSP application. Performance comparisons between the water, hybrid, and air-cooled cycles are made in terms of annual and monthly net energy production, net efficiency, and cooling water requirements. The cycles are also compared against results from Wagner's (2008) work simulating the Rankine cycle.

Chapter 7 summarizes conclusions and recommendations drawn from the work composing this

thesis. The chapter also puts forth potential areas of future work.

2 Simplified Modeling Methodology

2.1 Modeling Work Outline

The goal of this project is to produce a SCO_2 Brayton component for use in annual simulations with the TRNSYS simulation program (Klein, 2010). This goal is pursued by creating a detailed model of a suitable SCO_2 cycle configuration in EES (Klein, 2010) and then simplifying the results from the detailed model to a computationally-simpler performance map for implementation in TRNSYS. The intent of the performance map is to allow the performance of a SCO_2 Brayton cycle to be robustly and efficiently executed in TRNSYS. This approach is similar to that used by Wagner (2008).

This chapter focuses on developing the necessary component models to simulate the performance of a water-cooled Brayton cycle. Subsequent modifications to the model are made to investigate air-cooled and hybrid-cooled Brayton cycles (see Chapter 5). First a simplified SCO_2 Brayton model is created to analyze various cycle configurations. Based on the results of the simplified cycle analysis, a cycle configuration was selected and the development of detailed component models for incorporating into a detailed cycle model commenced. The detailed cycle model development is discussed in Chapter 3. Full code listings for the simplified model are found in Appendix A, and full code listings for the detailed model are found in Appendix E.

2.2 Simplified Model

A simplified model of a SCO_2 Brayton cycle provides a basis for quickly comparing different cycle configurations. The simplified models created here are compared to calculations reported by Dostal (2009). Since experimental data are not presently available to validate one or more of the models developed, agreement with Dostal's results is sought to provide a degree of validity to the present model development effort.

A schematic for a simple Brayton cycle with regeneration is shown in Figure 2-1.

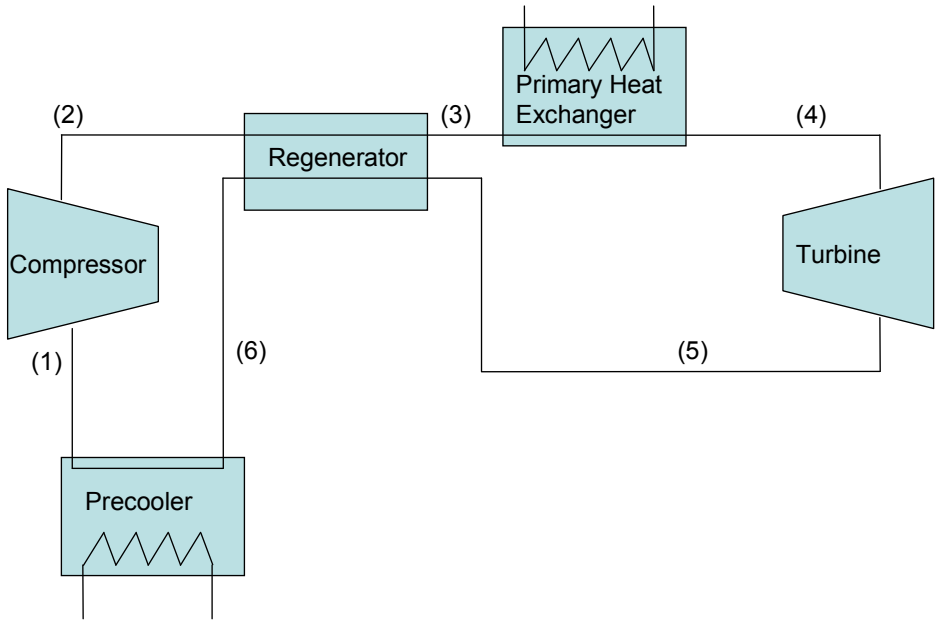

Figure 2-1: Simple Brayton Cycle With Regeneration

The regeneration cycle is composed of five components, a primary heat exchanger (PHX), turbine, regenerator, precooler, and compressor. Models are developed for each of these components as discussed in the sections that follow.

Heat Exchangers (PHX, Regenerator, Precooler)

Significant variability of CO_2 properties around the critical point necessitates nodalization of the heat exchanger models (Dostal, 2004; Hoang, 2009; Kao, 2009). The heat exchangers are modeled with *N* nodes. Selection of the number of nodes to be used in each heat exchanger is discussed later in this section. A schematic of the nodalization scheme is shown in Figure 2-2.

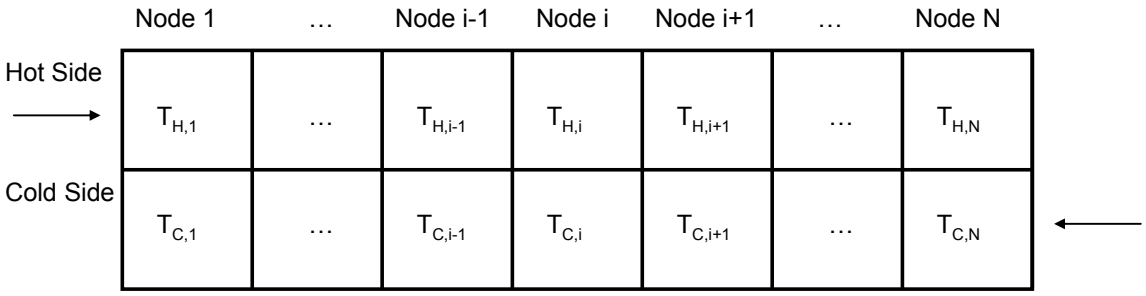

Figure 2-2: Heat Exchanger Nodalization Schematic

Figure 2-2 presents a schematic view of a heat exchanger divided into, *N* nodes, with each node being modeled as an independent heat exchanger. By using a sufficient number of nodes, the model can accurately account for the dependence of the fluid properties of CO_2 on temperature and pressure as it passes through the heat exchanger.

The heat exchanger size can be defined either by specifying a fixed effectiveness and calculating the necessary conductance (UA) values for each node, specifying a total UA value for the entire heat exchanger and calculating the corresponding effectiveness, or by simply specifying two inlet temperatures and one outlet temperature. Initial model validation was performed by specifying the CO_2 outlet temperatures for the precooler and primary heat exchanger, and by fixing the effectiveness of the regenerator. UA values were specified for each heat exchanger during the analysis work comparing various Brayton configurations.

The hot and cold sides of the exchanger will represent different fluids depending on which heat exchanger is being modeled. Energy is absorbed into the cycle in the primary heat exchanger where the "hot side" fluid is molten salt (60% $NaNO_3$ 40% KNO_3), and the "cold side fluid" is CO_2. In the regenerator, both "hot" and "cold side" fluids are CO_2. In the precooler, heat is rejected from the cycle where the "hot side" fluid is CO_2 and the "cold side" fluid is cooling water.

For the purpose of developing a simplified model here, the salt and water streams are not explicitly modeled. Rather, the CO_2 outlet temperatures from the precooler and primary heat

exchanger are simply fixed. For the rest of the simple modeling discussed in this chapter the heating and cooling fluids are replaced with very high flow rates of air. This was done to ensure high capacitance rates on the heating and cooling sides of the primary heat exchanger and precooler, respectively. Salt and cooling water are used in the detailed model, described in depth in Chapter 3.

The overall effectiveness of a heat exchanger is defined as the ratio of actual heat transfer to the maximum possible heat transfer through the heat exchanger, were the heat exchanger infinitely large. This is shown in equation 2-1.

$$\varepsilon = \frac{\dot{q}}{q_{\max}} \qquad 2\text{-}1$$

where ε is the effectiveness of the overall heat exchanger, $\dot{q}$ is the actual heat transfer through the heat exchanger, and $\dot{q}_{\max}$ is the maximum possible heat transfer through the heat exchanger.

An energy balance on the hot side of the heat exchanger shows the total heat transfer can be calculated in terms of the enthalpy difference between hot side inlet and outlet and the mass flow rate of the hot side fluid as shown in equation 2-2.

$$\dot{q} = \dot{m}_H \left(i_{H,1} - i_{H,N} \right) \qquad 2\text{-}2$$

where $i_{H,1}$ and $i_{H,N}$ are the specific enthalpies of the hot side fluid at inlet (node 1) and outlet (node *N*), $\dot{q}$ is the heat transfer rate, and $\dot{m}_H$ is the mass flow rate of the hot side working fluid. Specific enthalpy values for carbon dioxide are calculated using EES's built in property database.

The enthalpy of the cold side inlet fluid ($i_{C,N}$) is calculated in the same manner as the hot side inlet and outlet enthalpies. The cold side outlet enthalpy is calculated knowing the cold side inlet enthalpy, the overall rate of heat transfer, and the cold side mass flow rate as related in equation

2-3.

$$i_{C,1} = i_{C,N} + \frac{\dot{q}}{\dot{m}_C} \qquad 2\text{-}3$$

where $i_{C,1}$ is the enthalpy of the cold side fluid at the outlet, $i_{C,N}$ is the enthalpy of the cold side fluid at the inlet, and $\dot{m}_C$ is the mass flow rate of the cold side fluid.

The temperature of the cold side outlet is calculated from its enthalpy and pressure using EES's built in temperature function property data. At this point, the heat exchanger function has calculated the temperatures and enthalpies at the cold and hot side inlet and outlet temperatures and the total heat transfer through the heat exchanger.

Rather than physically discretizing the heat exchanger into an equal number of physically-sized sections, the present heat exchanger model divides the total heat transfer by the number of sections (nodes), and assigns an equal amount of heat transfer to each node (while allowing the *UA* of each node to vary) following a methodology outlined in Klein and Nellis (2009). It uses the heat transfer in each section to calculate changes in the specific enthalpy of each stream as shown in equations 2-4 and 2-5, which is later used to calculate the temperature at each node.

$$i_{H,i} = i_{H,i-1} + \frac{\dot{q}}{N \cdot \dot{m}_H} \qquad 2\text{-}4$$

$$i_{C,i} = i_{C,i-1} + \frac{\dot{q}}{N \cdot \dot{m}_C} \qquad 2\text{-}5$$

where N is the number of nodes used in the heat exchanger, and the subscript 'i' refers to the number of each particular node. The heat exchanger model then calculates the temperature at each node i knowing the specific enthalpy and pressure.

This heat exchanger model can calculate the temperature and enthalpy differences between each node on the hot and cold sides of the heat exchanger, as shown in equations 2-6 through 2-9.

Refer to Figure 2-2 to illustrate node nomenclature.

$$\Delta T_{H,I} = T_{H,i} - T_{H,i+1} \quad \text{2-6}$$

$$\Delta T_{C,i} = T_{C,i} - T_{C,i+1} \quad \text{2-7}$$

$$\Delta i_{H,i} = i_{H,i} - i_{H,i+1} \quad \text{2-8}$$

$$\Delta i_{C,i} = i_{C,i} - i_{C,i+1} \quad \text{2-9}$$

where Δ represents a difference, subscripts '*c*' and '*h*' represent the cold and hot sides respectively, and the subscript '*i*' represents the index of the node being calculated.

The hot side and cold side capacitance rates are calculated on the basis of fluid mass flow rate and the ratio of enthalpy and temperature differences at each node (where this ratio represents the average heat capacity for the node). The calculation for the hot side is shown below as equation 2-10 and the method used for the cold side is identical but not shown.

$$\dot{C}_{H,i} = \dot{m}_H \cdot \frac{\Delta i_{H,i}}{\Delta T_{H,i}} \quad \text{2-10}$$

where $\dot{C}_{H,i}$ is the capacitance of the hot side fluid at node i.

The effectiveness of each heat exchanger node is then calculated based on its relationship with the overall heat transfer rate, the minimum capacitance rate for the fluids at that node, and the temperature difference between those fluids as shown in equation 2-11.

$$\dot{q} = \sum \varepsilon_i \dot{C}_{\min,i} (T_{H,i} - T_{C,i+1}) \quad \text{2-11}$$

where ε_i is the effectiveness of heat exchanger node *i,* $\dot{C}_{\min,i}$ is the minimum capacitance rate between the hot and cold fluids at node *i,* $T_{H,i}$ is the temperature of the hot fluid at node *i* and $T_{C,i+1}$ is the temperature of the cold fluid at node *i+1*.

The UA value for each node is calculated by means of equation 2-12.

$$UA_i = NTU_i \cdot \min(\dot{C}_{H,i}, \dot{C}_{C,i}) \qquad 2\text{-}12$$

where NTU_i is the 'number of transfer units' for each node, and it is calculated (for the counterflow heat exchangers examined in this thesis) by means of equation 2-13 (Nellis, Klein 2009).

$$NTU_i = \frac{1 - \exp[-NTU \cdot (1 - C_R)]}{1 - C_R \cdot \exp[-NTU \cdot (1 - C_R)]} \qquad 2\text{-}13$$

where $C_R = \frac{\dot{C}_{\min}}{\dot{C}_{\max}}$.

The above equations are implemented in EES in the form of a function (a full code listing for the function appears in Appendix A). The function has arrays containing the temperature of the two fluids at each node in the heat exchanger, and the heat exchanger analysis is complete.

The number of nodes, *N,* is variable but set to 10 for the precooler, a value that provides results that correlate well with higher-node simulations. This is shown in Figure 2-3, where cycle efficiency is plotted against pressure ratio while the number of nodes is varied.

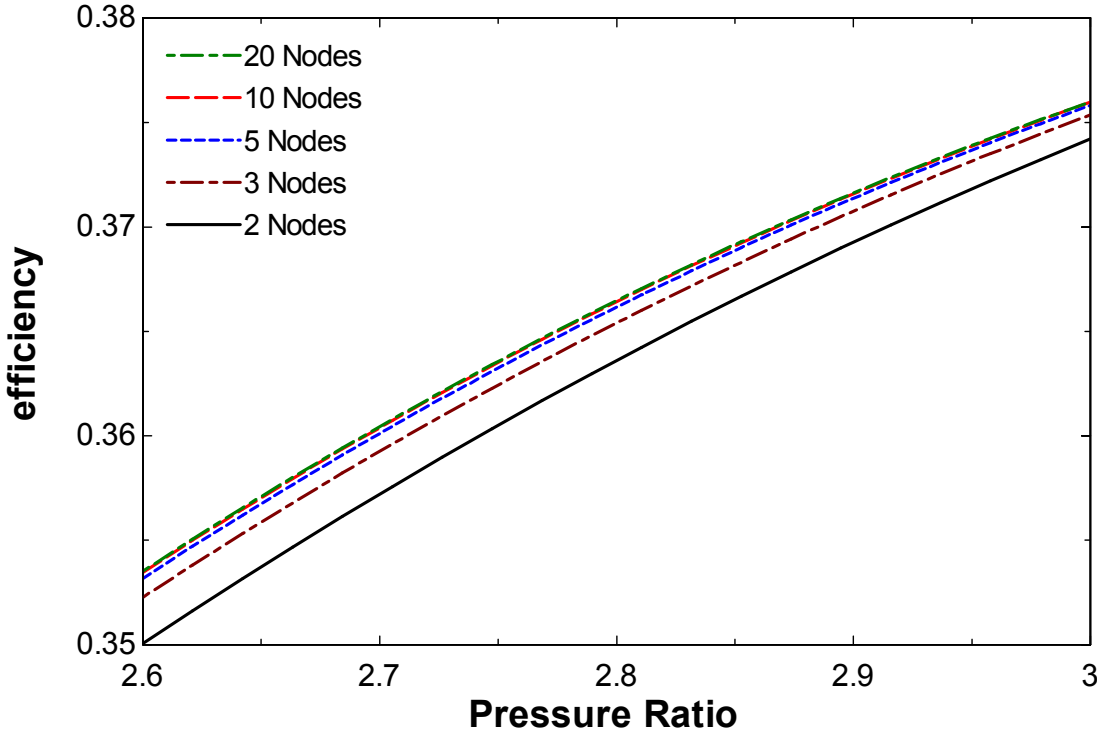

Figure 2-3: Specific Work v. Pressure Ratio at Various HX Node Numbers

As the number of nodes increases the efficiency begin to converge. 10 and 20 nodes produce very similar results, and for the scope of this project are considered to be nearly identical. Ten nodes have been found to be sufficient for the precooler, and fewer nodes are needed for the other heat exchangers because they operate at states for which the properties are less sensitive to temperature.

The regenerator and primary heat exchanger are represented with 2 nodes each, due to greater distance from the critical point of CO_2, as suggested by Cox (2009).

Turbine

The turbine is modeled by specifying the inlet fluid temperature and pressure, the outlet pressure, and the isentropic efficiency of the turbine. The turbine function calculates the entropy and enthalpy of the fluid entering the turbine (s_{in} and i_{in}) based on the known inlet temperature and pressure. It then calculates an isentropic exit entropy condition (s_{outs}) that has an outlet entropy equal to the inlet entropy as shown in equation 2-14.

$$s_{outs} = s_{in} \qquad 2\text{-}14$$

where s_{outs} is the isentropic outlet entropy, and s_{in} is the inlet entropy. The outlet enthalpy can then be found using the specified outlet pressure P_{out} (i_{outs}) and the specific entropy value s_{outs}. The outlet enthalpy value is then used to determine the specific work for an isentropic turbine (w_s) as shown in equation 2-15.

$$w_s = i_{in} - i_{outs} \qquad 2\text{-}15$$

where i_{outs} is the outlet enthalpy from an isentropic turbine and w_s is the work per unit mass produced by the isentropic turbine.

This isentropic work is related to the actual turbine work with a constant turbine efficiency as shown in equation 2-16.

$$w = w_s \cdot \eta \qquad 2\text{-}16$$

where w is the actual specific work, and η is the isentropic efficiency of the turbine. The actual work can, in turn, be used to determine the actual enthalpy leaving the turbine using equation 2-17.

$$i_{out} = i_{in} - w \qquad 2\text{-}17$$

where i_{out} is the actual enthalpy after the turbine. With this value, and EES's built in temperature function and data, the actual temperature after the turbine (T_{out}) can be found using the enthalpy i_{out} and the outlet pressure, P_{out}.

At this point, the turbine exit temperature and specific turbine work are known.

Compressor

The compressor is modeled using a methodology identical to that employed with the turbine function - with one difference. The relationship between ideal work and actual work of the compressor are defined as given in equation 2-18.

$$w = \frac{w_s}{\eta} \qquad 2\text{-}18$$

where η, is the efficiency of the compressor (modeled as 89% to coincide with Dostal's (2009) assumptions). Otherwise all calculations are carried out in the same manner as in the turbine model.

2.3 Simplified Model Validation

The simplified model was compared with the results previously presented by Dostal (2009). Assumptions used in the present simplified model, as well as by Dostal, and are as follows:

Cycle:	*Simple Brayton with Regeneration*
Turbine Efficiency:	*90%*
Compressor Efficiency:	*89%*
Regenerator Effectiveness:	*95%*
Turbine Inlet Temperature:	*550°C*
Compressor Inlet Temperature:	*32°C*
Pressure Losses Neglected	

The simplified model was run across a range of pressure ratios (defined by the compressor outlet pressure divided by the turbine outlet pressure) at several different compressor outlet pressures. Figure 2-4 and Figure 2-5 show the thermal efficiency of the Brayton Cycle as reported by Dostal and as predicted by the simplified model.

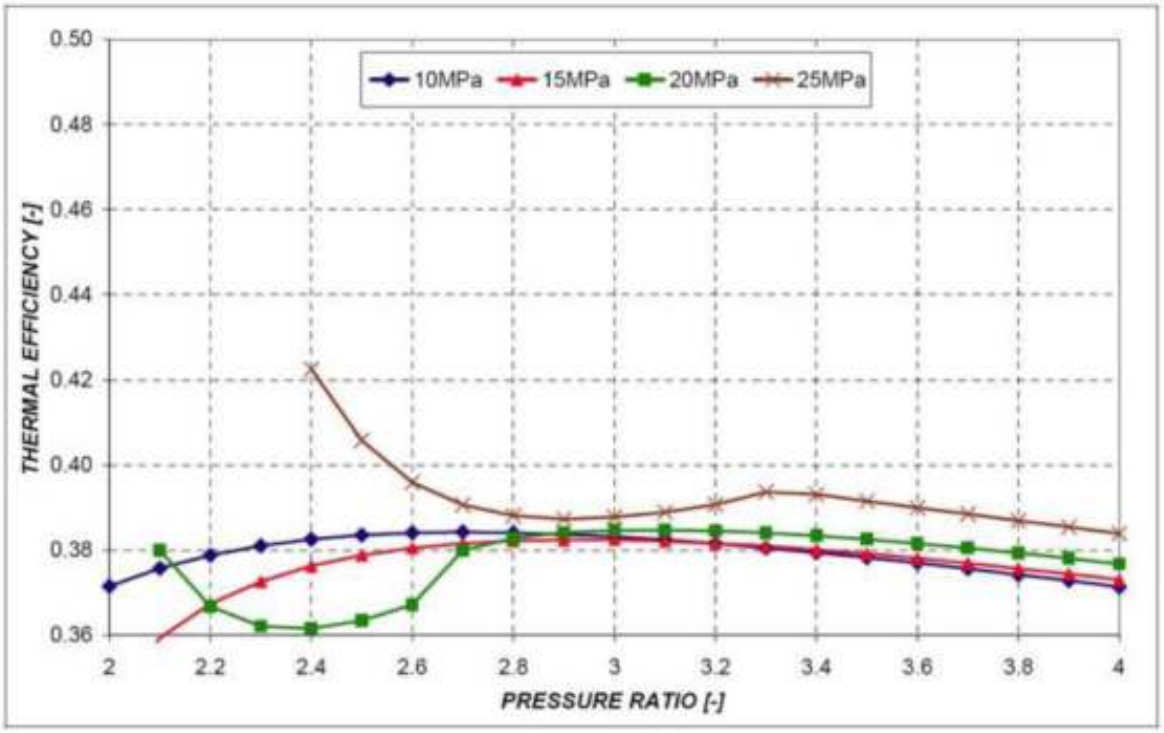

Figure 2-4: Thermal Efficiency v/Pressure Ratio and Compressor Outlet Pressure (Dostal 2009)

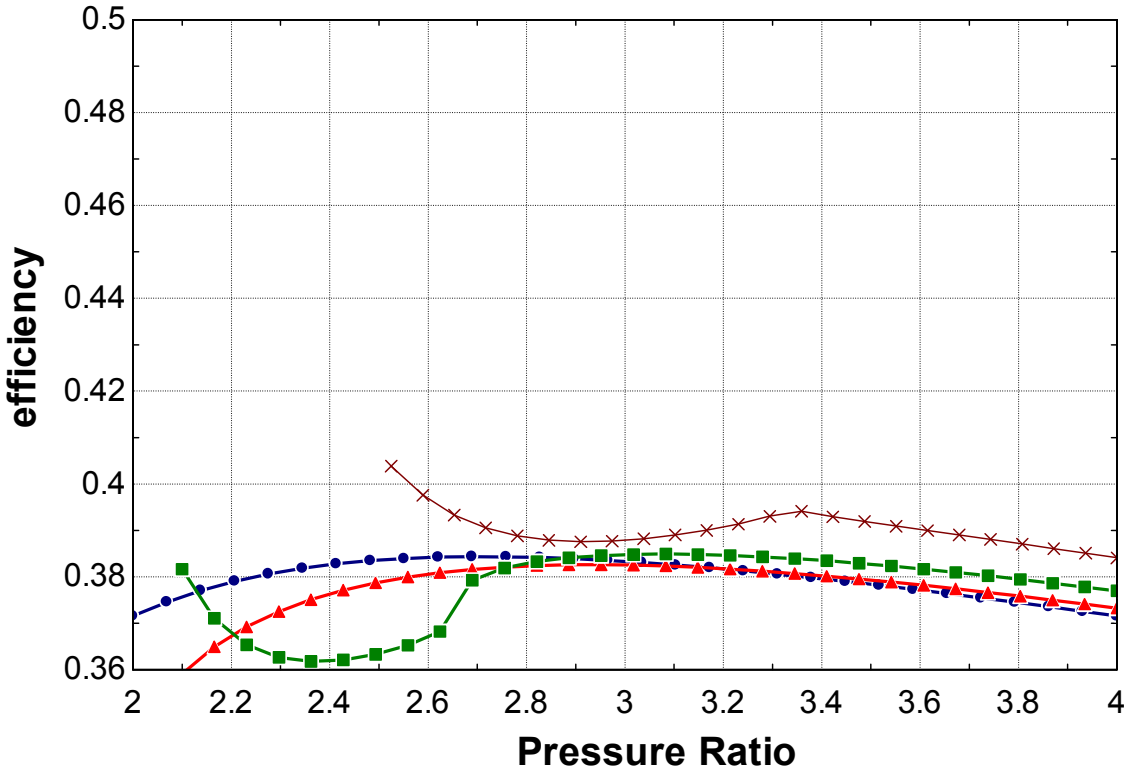

Figure 2-5: Thermal Efficiency v/Pressure Ratio and Compressor Outlet Pressure Calculated with the Simplified Model

Close agreement between these two plots provides some degree of confidence the present model is able to predict the results of Dostal. Also, all further analysis is based on a 25 MPa compressor outlet pressure, since that condition results in the greatest efficiency.

Dostal (2009) investigated several potential Brayton configurations across a range of operating conditions. A good deal of initial modeling focused on recreating these cycle models and choosing which cycle to move forward with. The next section discusses this work.

2.4 Cycle Comparison

A variety of Brayton Cycle configurations exist, with varying degrees of complexity. A single cycle had to be chosen for further investigation, and so models of the different cycles proposed by Dostal were developed evaluated. Their relative performance was balanced against their complexity in choosing which to model in more detail. The Simple Brayton with Regeneration, Recompression, Precompression, and Split Expansion cycles were examined. The Simple Brayton Cycle with Regeneration has been discussed already, and the three other Brayton configurations are outlined below.

Precompression Cycle

The Precompression Cycle employs both a high and low temperature regenerator, with an additional compressor situated between the two regenerators on the low pressure side of the cycle. By increasing the pressure of the working fluid before the low temperature regenerator, the heating side fluid in that heat exchanger increases in temperature, helping to avoid a pinch point (a point in a heat exchanger where the hot and cold streams approach the same temperature, reducing heat transfer) and facilitating further heat transfer. (Dostal, 2009)

The Precompression Cycle defines two pressure ratios, the pressure ratio across the turbine, r_T, and the pressure ratio across the primary compressor, r_C. These two values define *rpr*, as shown in equation 2-19.

$$rpr = \frac{(r_C - 1)}{(r_T - 1)} \qquad 2\text{-}19$$

This value is held at 0.8, as recommended by Dostal. Figure 2-6 and Figure 2-7 show schematic and T-S diagrams for the Precompression cycle, respectively. The EES code for the Precompression model is provided in Appendix B.

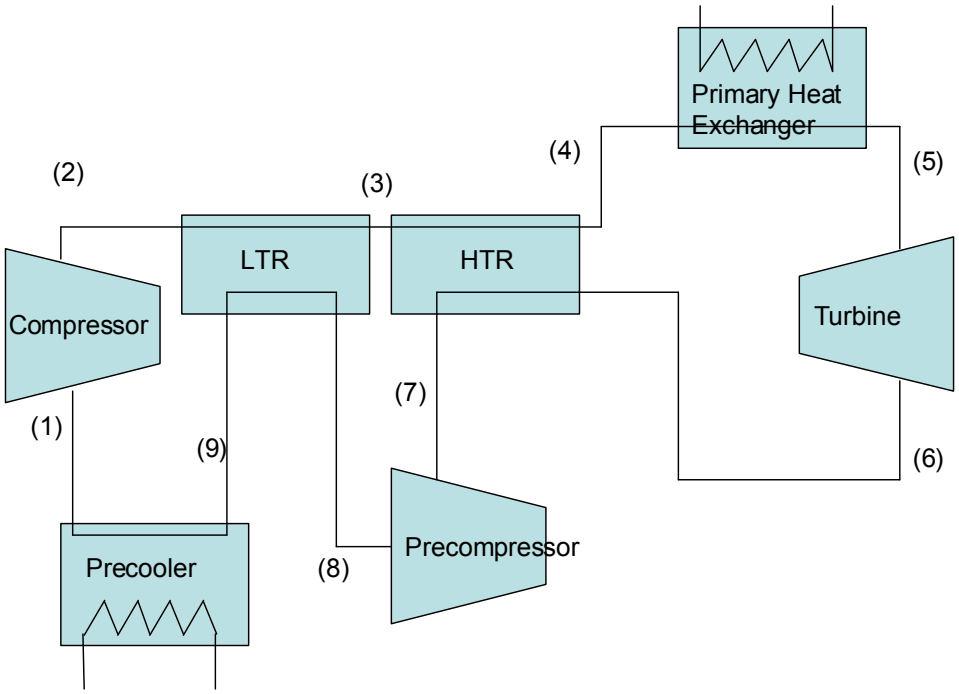

Figure 2-6: Schematic Diagram of Precompression Brayton Cycle

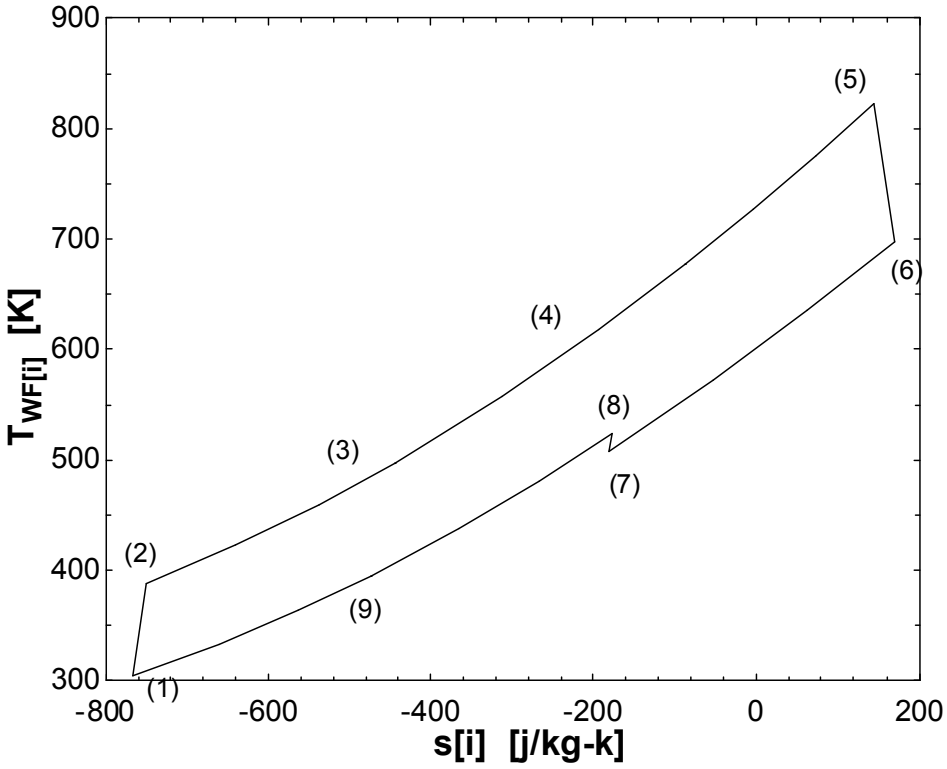

Figure 2-7: T-S Diagram of Precompression Brayton Cycle, with State Points

Recompression Cycle

The Recompression Cycle is similar to the Precompression Cycle – but with the second compressor placed after both regenerators. The flow splits at this compressor – with a portion continuing towards to the primary compressor, and a portion entering the recompressing compressor and exiting between the two regenerators on the high pressure side of the cycle. The mass split fraction is set to let the recompressor outlet conditions match those of the fluid exiting the low temperature regenerator on the high pressure side of the cycle. This cycle changes the capacitance rate by varying the mass flow through the low temperature regenerator, helping to a pinch point. (Dostal, 2009)

Figure 2-8 and Figure 2-9 show schematic and T-S diagrams for the Recompression cycle, respectively. The EES code for the Recompression model is provided in Appendix C.

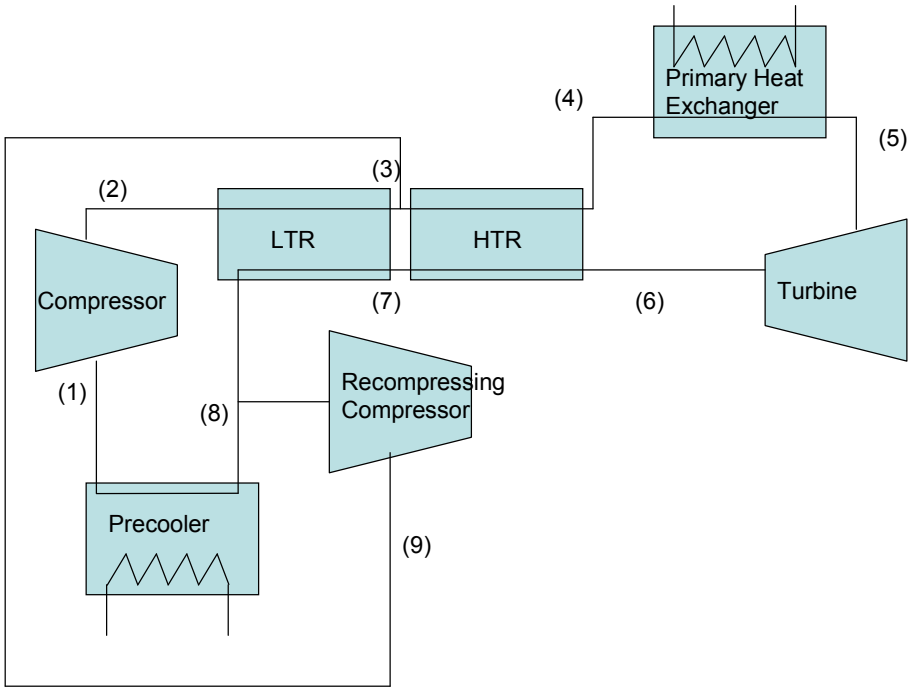

Figure 2-8: Schematic Diagram of Recompression Brayton Cycle

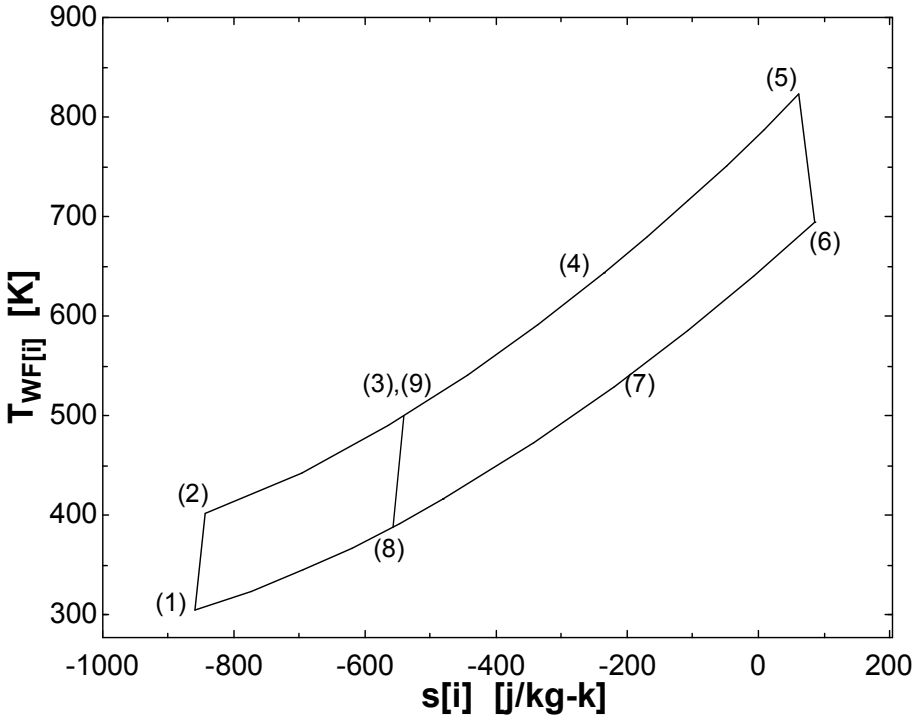

Figure 2-9: T-S Diagram of Recompression Brayton Cycle, with State Points

Split Expansion Cycle

The Split Expansion Cycle is, effectively, identical to the Recompression Cycle with the addition of a single secondary turbine between the high temperature regenerator and primary heat exchanger on the high pressure side of the cycle. This addition allows the cycle to extract work for the secondary turbine while reducing the highest temperature achieved in the primary heat exchanger, which can be useful if materials constraints exist. The ratio of the magnitudes of the pressure drop through the secondary turbine and the pressure increase through the primary compressor is defined by Dostal (2009) as shown in equation 2-20, below.

$$f = \frac{p_{\text{int}} - p_{low}}{p_{high} - p_{low}} \qquad 2\text{-}20$$

The higher the value of *f* that is specified the closer the Split Expansion Cycle will be to the Recompression Cycle, and the higher its efficiency. The lower the value of *f,* the lower the maximum temperature in the primary heat exchanger will be. For this analysis f was fixed at 0.5 since Dostal found that a value as low as 0.5 resulted in an efficiency loss of only about ~1.5%.

Figure 2-10 and Figure 2-11 show schematic and T-S diagrams for the split expansion cycle, respectively. The EES code for the Split Expansion model is provided in Appendix D.

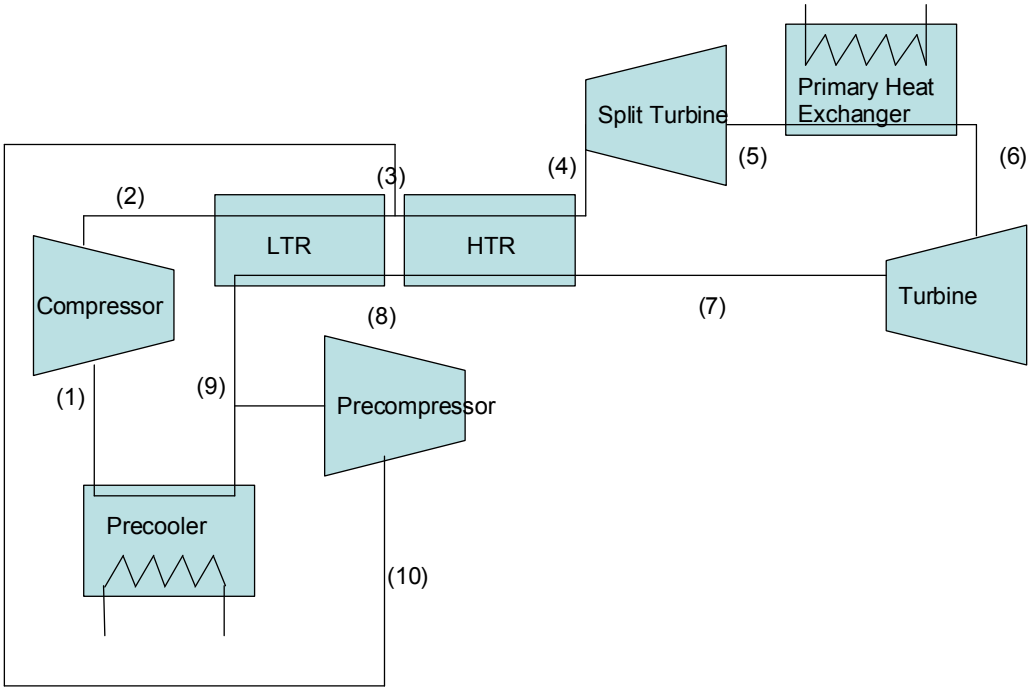

Figure 2-10: Schematic Diagram of Split Expansion Brayton Cycle

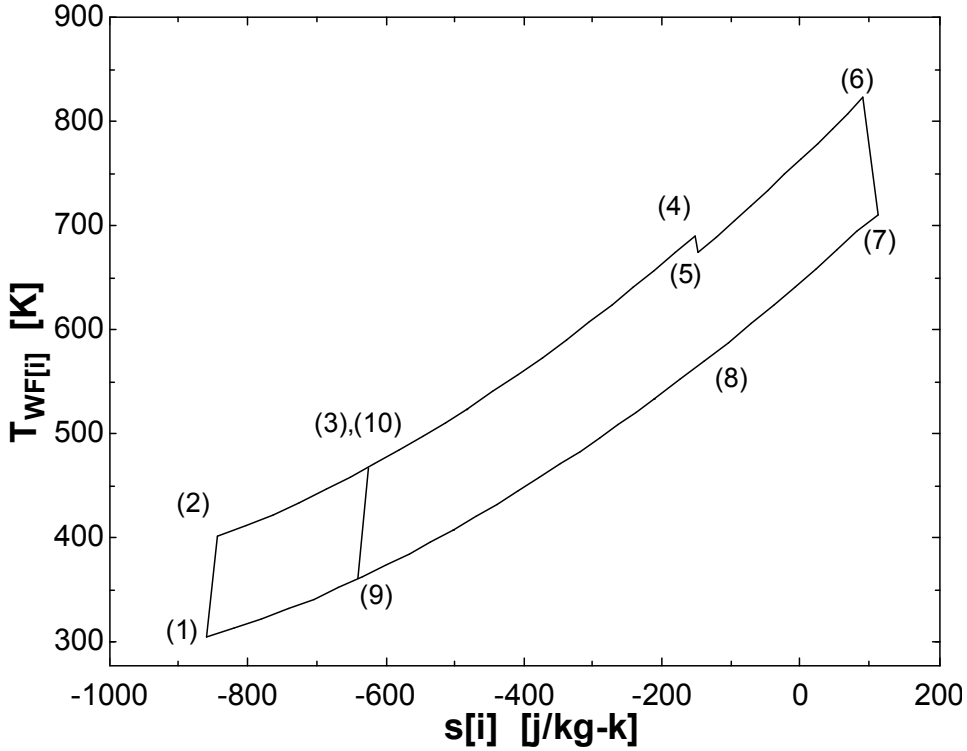

Figure 2-11: T-S Diagram of Split Expansion Brayton Cycle, with State Points

Cycle Performance Comparison

The four Brayton configurations were compared across a range of pressure ratios. The total UA for all heat exchanger in each configuration was kept constant and the distribution of that UA was optimized for power (turbine power less compressor power) for one comparison and efficiency (net power divided by the heat transfer rate in the primary heat exchanger) for the other. Optimization was performed by retaining the cycle assumptions outlined at the beginning of section 2.3, but without fixing the distribution of total UA between the heat exchangers. Instead the distribution is numerically optimized using EES's "Min/Max" functionality.

Figure 2-12 shows an efficiency comparison of the four cycles with the same total UA available, and the distribution of that UA optimized at each operating condition for efficiency.

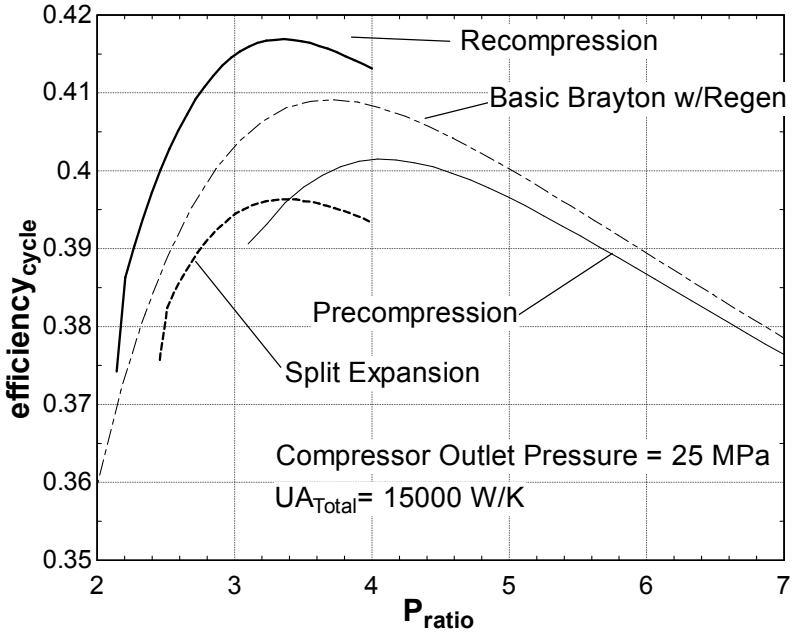

Figure 2-12: Efficiency Comparison with Fixed Total UA

The Recompression cycle has the highest overall efficiency with the Simple Brayton with Regeneration performing nearly as well. These two cycles both offer simplicity and high efficiency. The Basic Brayton offers slightly greater simplicity at a slightly reduced efficiency.

Figure 2-13 shows the power production of each cycle with UA distribution optimized for power

production.

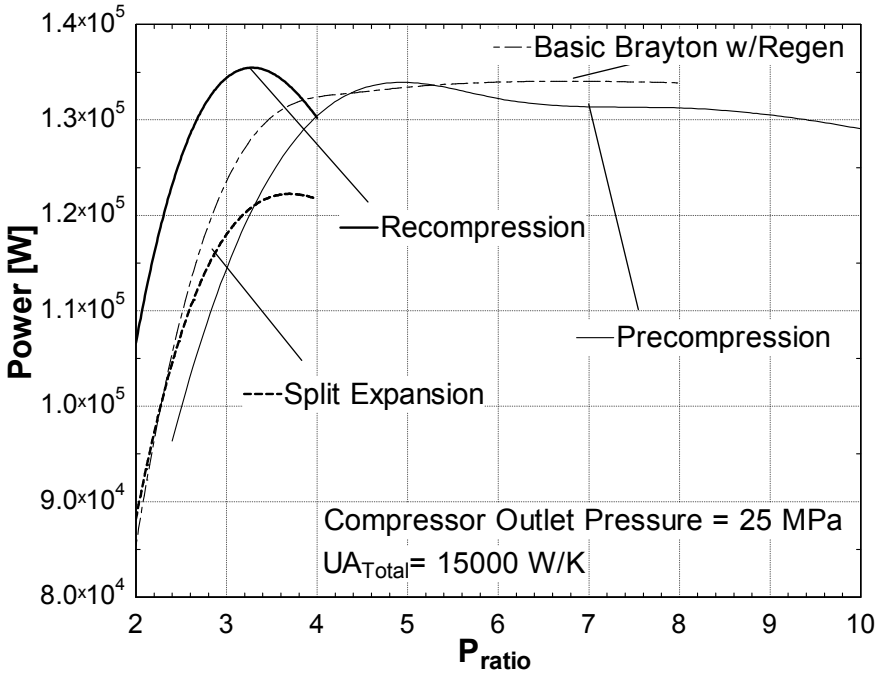

Figure 2-13: Power Comparison with Fixed Total UA

As with efficiency, the Recompression cycle performs best, though the Basic Brayton and Precompression cycles are close behind. The Split Expansion cycle lags significantly in terms of peak power production.

2.5 Cycle Selection

Fixing the total available *UA* for all heat exchangers and optimizing the cycles for efficiency or power yielded a meaningful comparison of the various Brayton configurations. This comparison revealed that the Recompression and Simple Brayton with Regeneration cycles provide the best performance of the four cycles examined. The Recompression provides the best efficiency and power production across a range of pressure ratios. The Simple Brayton with Regeneration cycle provides comparable, if slightly reduced, performance but is a simpler design.

The Simple Brayton with Regeneration was selected for further analysis due to its simplicity and high performance. This choice agrees with the conclusion reached by Turchi (2009).

3 Detailed Modeling Methodology

Chapter 2 discussed the modeling methodology behind the simplified SCO_2 Brayton model implemented in EES (Klein, 2010). That simplified model serves as the basis for a more detailed SCO_2 Brayton cycle model that is the focus of Chapter 3. The differences between the two models fall into two categories.

The first category encompasses the heat exchangers, which are no longer simply modeled on a conductance (UA) basis, but instead are represented physically as Printed Circuit Heat Exchangers (PCHEs) with calculated heat transfer coefficients and pressure drops. Also, the hot fluid in the primary heat exchanger, and the cold fluid in the precooler are no longer modeled as high capacitance rate streams. Instead they are modeled as finite capacitance streams assuming molten salt and water, respectively.

The second category is cycle controls. Whereas the simplified model allows the compressor and turbine inlet temperatures to vary as a function of heat exchanger effectiveness, the detailed model fixes the compressor inlet CO_2 temperature by demanding a specific cooling water temperature from an external cooling tower. The detailed model also fixes the CO_2 flow rate at the compressor inlet on a volumetric basis, whereas a mass basis was used in the simplified model.

Changes in both of these categories are discussed in detail in the sections below. For full listing of the EES code for the detailed model please refer to Appendix E.

3.1 Detailed Heat Exchangers

Printed Circuit Heat Exchangers

Modeling the power cycle requires simulation of several heat exchangers. Traditional shell and tube heat exchangers are not cost-effective due to the extremely high pressure difference

between the hot and cold side of the regenerator and the relatively low heat transfer effectiveness of the configuration. To meet the design pressure requirements, shell and tube heat exchangers would require extremely thick walls (Dostal, 2004). An alternative heat exchanger design considered here is the PCHE. Printed Circuit Heat Exchangers (PCHEs) offer high strength and tolerance of high temperatures (Southall, 2009). PCHEs are created by chemically etching flow channels into flat metal plates. Channels can be manufactured at any size above 0.1 mm diameter, with 2.0 mm diameter being a common baseline size (Dostal, 2004). The metal plates are then diffusion bonded creating a chemically consistent block of metal with flow channels. A PCHE section is shown in Figure 3-1.

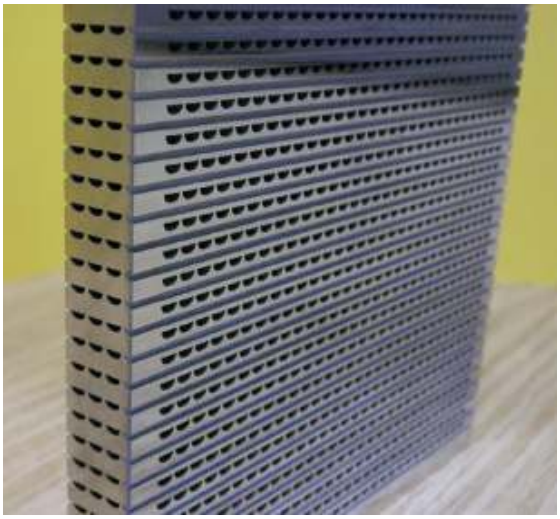

Figure 3-1: Printed Circuit Heat Exchanger (PCHE) Section (Southall, 2009)

The conditions that would be imposed on the PCHE heat exchangers by the SCO_2 Brayton cycle are well within the operating range for these heat exchangers, which at 25MPa includes temperatures up to 950 K (Dostal, 2004).

Counter-flow PCHEs were modeled with the hemispherical channels approximated by square channels as shown in Figure 3-2. A counter-flow configuration was chosen for ease of simulation and maximum performance.

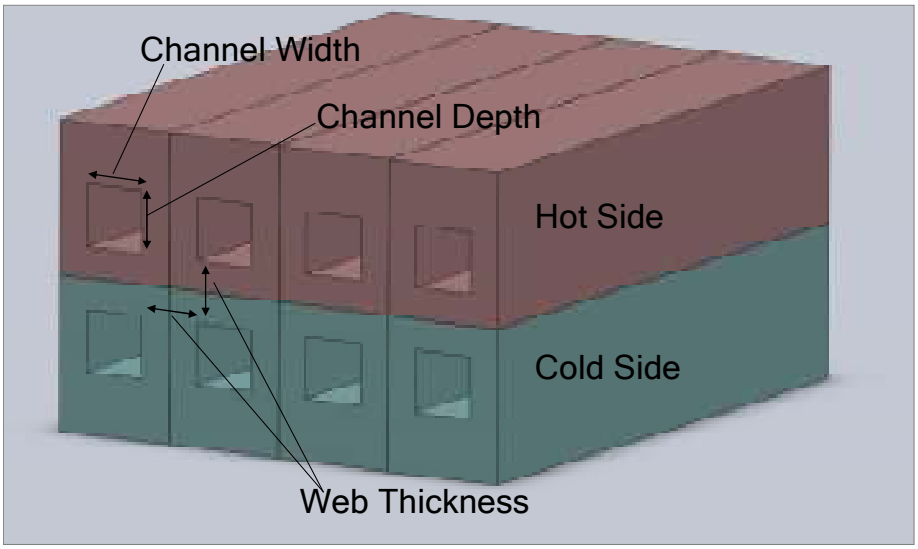

Figure 3-2: Printed Circuit Heat Exchanger Geometry as Modeled

Approximating the hemispherical channels as square is acceptable because flow remains turbulent in all heat exchangers. Heat exchanger performance is not sensitive to the shape of a channel's cross section while flow remains turbulent.

Molten Salt in the Primary Heat Exchanger

The simplified heat exchanger model described in Chapter 2 relied upon very high capacitance rates of the heating fluid in the primary heat exchanger. The detailed model uses molten salt with a composition of 60% $NaNO_3$, 40% KNO_3 to agree with Wagner (2008) at flow rates that can be expected in actual operation. The use of molten salt necessitates a slightly different methodology for calculating the enthalpy change through the primary heat exchanger. This is necessary because while EES can directly calculate the specific enthalpy of air and water, this feature is not available for molten salt. An inlet specific enthalpy is calculated by multiplying the inlet temperature and the average of the specific heat evaluated at the inlet temperature and outlet temperature, as shown in equation 3-1.

$$i_{H,in} = T_{H,in} \cdot \frac{c_p(T_{H,in}) + c_p(T_{H,out})}{2} \qquad 3\text{-}1$$

where $i_{H,in}$ is the inlet enthalpy of the molten salt. The same method is used to calculate the outlet enthalpy of the salt:

$$i_{H,out} = T_{H,out} \cdot \frac{c_p(T_{H,in}) + c_p(T_{H,out})}{2} \qquad 3\text{-}2$$

The assumption that the specific heat of the molten salt is a linear function of temperature is reasonable, based on the relationship between temperature and salt specific heat shown in Figure 3-3 for a temperature range consistent with the primary heat exchanger.

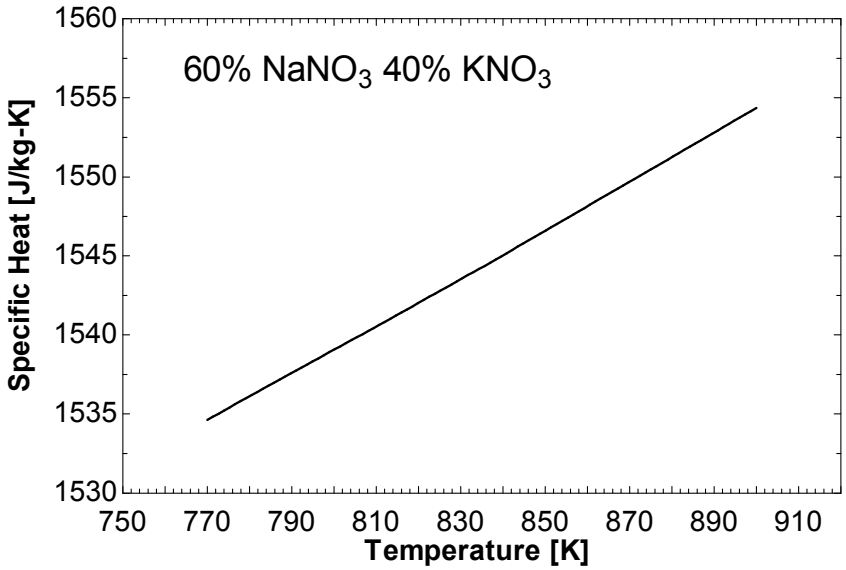

Figure 3-3: Specific Heat v. Temperature for Molten Salt

This modification of the methodology behind calculating enthalpy is all that is required to adapt the primary heat exchanger model for use with molten salt, the rest of the primary heat exchanger model remains identical to that used in the simplified model described in Chapter 2.

Cooling Water in the Precooler

As in the primary heat exchanger, the simplified precooler model (see Chapter 2) assumed that the fluid exchanging heat with CO_2 was a fluid provided at high capacitance rate. The detailed precooler models the secondary side of the heat exchanger assuming a finite capacitance rate

fluid (water). There are no required changes for the precooler model to take into account this change – aside from referring to the cooling fluid as *'water'* and not *'air'* in the EES code.

Heat Transfer Coefficients

While the simplified SCO_2 Brayton model approximates the heat exchangers by means of setting their conductance (UA), the detailed model calculates the overall conductance by fixing the heat exchanger area and calculating the overall heat transfer coefficient by combining the heat transfer coefficients on the carbon dioxide-side and the secondary fluid side along with the resistance of the heat exchanger wall. Calculation of heat transfer coefficients in these heat exchangers for supercritical CO_2 relied upon the Gnielinski correlation (Nellis and Klein, 2009) with properties evaluated at the film temperature (the average between the bulk temperature and wall temperature). Kruizenga (2010) showed this method to be superior to using the Gnielinski correlation with properties evaluated at the bulk temperature. Figure 3-4 shows data in the form of Nusselt Number ratios calculated at different axial locations in an experimental PCHE precooler (Kruizenga 2010). The Nusselt number ratio represents the ratio of Nusselt number calculated to the experimentally-measured Nusselt number for two different cases. In one case (open square symbols), the calculated Nusselt number evaluates properties at the bulk temperature while the second case (solid square symbols) evaluates properties at the film temperature. The bulk temperature is defined as the mean temperature of the flow at a particular axial position. The film temperature is defined as the mean of the bulk temperature and the wall temperature.

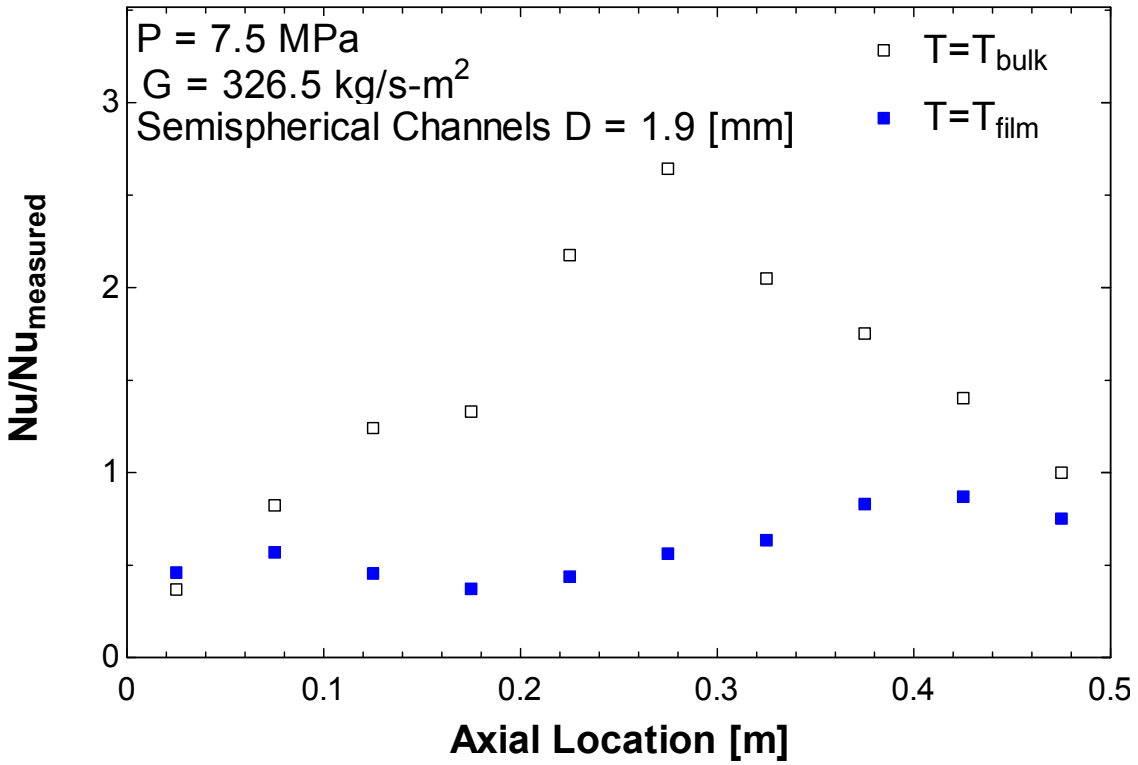

Figure 3-4: Ratio of Calculated and Measured Nusselt Numbers Evaluated Using Gnielinski Correlation at Bulk and Film Temperatures v. Axial Location

Kruizenga found that the standard Gnielinski correlation with properties evaluated at the bulk temperature greatly over predicts heat transfer when bulk conditions are near the critical point. Evaluating fluid properties at the film temperature (the average of the bulk and wall temperatures) mitigates this effect, and no longer results in over prediction. Over prediction is due to the Gnielinski correlation's dependence on the Prandtl number (and therefore on specific heat), which increases drastically around the critical point. This relationship is shown in Figure 3-5.

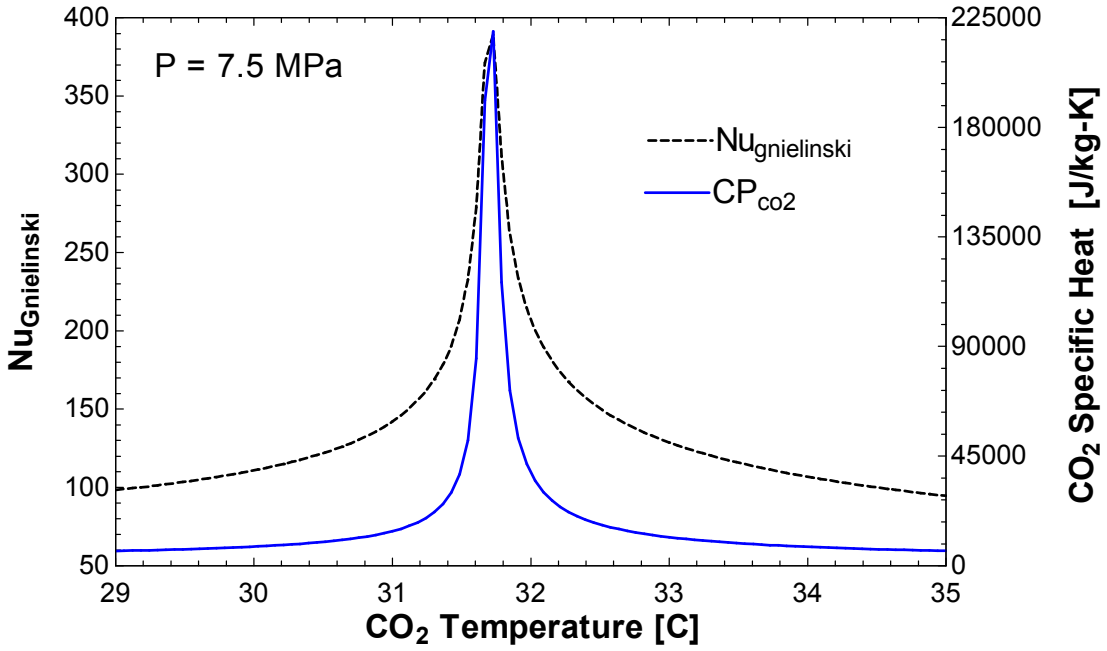

Figure 3-5: Calculated Nusselt Number and CO_2 Specific Heat v. Temperature

In general, the heat capacity of CO_2 will increase dramatically as it approaches the critical point. Because the data plotted above is at a pressure slightly higher than the critical pressure, the heat capacity reaches its maximum at 31.7 C which is slightly higher than the critical temperature of 30.98 C. The increase in specific heat decreases the rate of change of the bulk temperature with respect to axial position. Although the wall of the heat exchanger continues to cool steadily as the cooling water moves axially through the heat exchanger, the bulk temperature of the CO_2 decreases much more slowly and instead stays close to the critical temperature. While the actual heat transfer is occurring in the cooler CO_2 near the wall, the Gnielinski correlation traditionally depends on fluid properties evaluated at the bulk temperature very close to the critical point. The correlation relies upon an artificially inflated specific heat and this causes the correlation to over-predict the heat transfer coefficient. By evaluating CO_2 properties at the film temperature, the artificial specific heat peak is avoided, and the heat transfer coefficient is not over predicted. Further information on heat transfer calculations in supercritical CO_2 can be found in Kruizenga (2010), including an improved correlation which was not implemented for this thesis.

To calculate the film temperature in the heat exchanger model it is necessary to perform an energy balance on each node of the heat exchanger. The heat exchanger geometry is shown in

Figure 3-6.

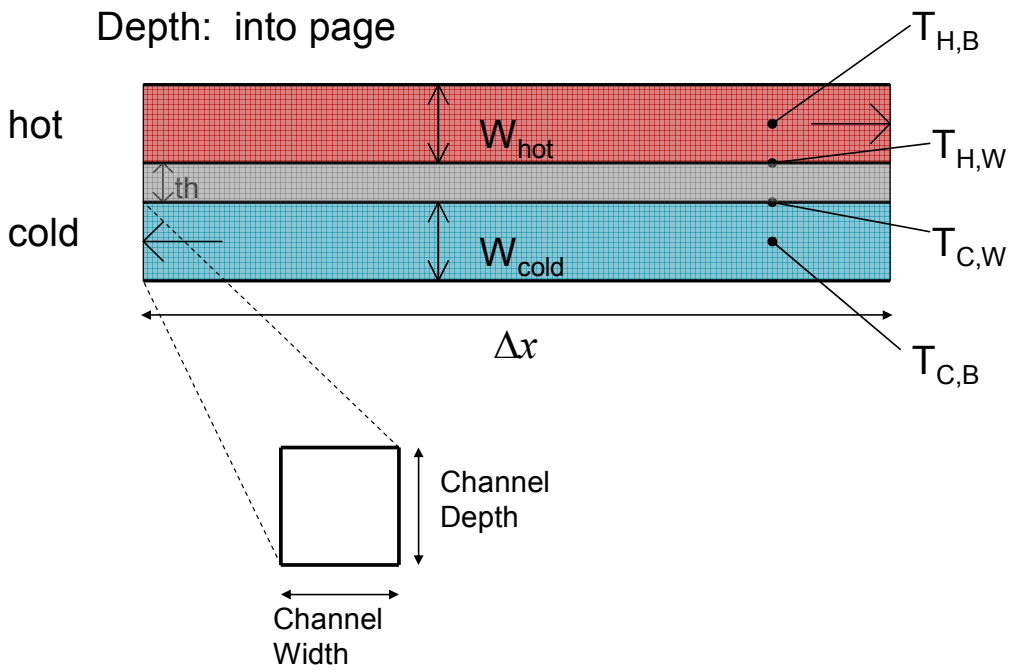

Figure 3-6: Heat Exchanger Geometry for Energy Balance

where $T_{H,B}$ and $T_{H,W}$ represent the hot side bulk and wall temperatures, respectively. Similarly, $T_{C,B}$ and $T_{C,W}$ represent the cold side bulk and wall temperatures. The heat exchanger web thickness is represented by *th*, and the channel widths of the hot and cold side are given as W_{hot} and W_{cold}. To simplify the heat transfer calculations and increase speed, it is assumed that the temperature of the heat exchanger web is constant in each section and equal to the average temperature of the hot and cold side bulk temperatures. Figure 3-7 shows a resistance network representative of heat transfer in a titanium regenerator node with 2mm channel widths and 2mm web thickness.

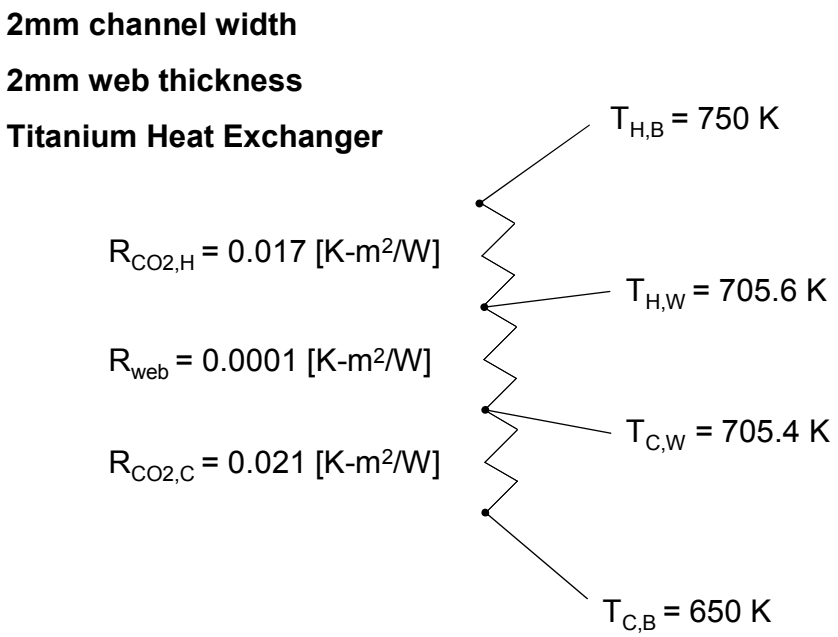

Figure 3-7: Representative Resistance Network for PCHE

The thermal resistance of the heat exchanger web is two orders of magnitude less than that of the CO_2 on either side of the heat exchanger, and the temperature variation in the web is small compared to the temperature variation in the CO_2. The wall temperature is also close to the average of the hot and cold side bulk temperatures.

At steady state, an energy balance requires the heat transfer from the cold fluid to the web to be equal to the heat transfer from the hot fluid to the web. These quantities are calculated as shown in equations 3-3 and 3-4, respectively.

$$\dot{Q} = h_H A(T_{H,B} - T_{H,W}) \qquad 3\text{-}3$$

$$\dot{Q} = h_C A(T_{C,W} - T_{C,B}) \qquad 3\text{-}4$$

where A is the heat transfer area defined by equation 3-5.

$$A = W_{ch} \cdot \Delta x \cdot N_{ch} \qquad 3\text{-}5$$

W_{ch} is the width of the channels, N_{ch} is the number of channels, and Δx is the length of the node. The heat transfer coefficients for the hot and cold sides are represented as h_C and h_H respectively. The heat transfer coefficients are calculated using the Gnielinski correlation for Nusselt number in equation 3-6.

$$Nu_{D_h} = \frac{\left(\frac{f_{fd}}{8}\right)\left(\mathrm{Re}_{D_h} - 1000\right)\mathrm{Pr}}{1 + 12.7\left(\mathrm{Pr}^{2/3} - 1\right)\sqrt{\frac{f_{fd}}{8}}} \qquad 3\text{-}6$$

The Gnielinski correlation is found in Nellis and Klein (2009), and in this case properties were evaluated at the film temperature, as discussed above. Re_{D_h} is the Reynolds number defined by the hydraulic diameter of the channel, which in turn is defined in equation 3-7.

$$D_h = \frac{2 \cdot W_{ch} \cdot D_{ch}}{W_{ch} + D_{ch}} \qquad 3\text{-}7$$

where W_{ch} is the width of the channel, and D_{ch} is the height of the channel. Since the channels are assumed to be square – this formula is simplified to equation 3-8.

$$D_h = W_{ch} \qquad 3\text{-}8$$

The fully developed friction factor (f_{fd}) is calculated from the Petukhov correlation (Nellis and Klein, 2009) shown in equation 3-9.

$$f_{fd} = \frac{1}{\left[.79\ln(\mathrm{Re}_{D_h}) - 1.64\right]^2} \qquad 3\text{-}9$$

With the Nusselt number calculated from the Gnielinski correlation, the heat transfer coefficients for the hot and cold channels are calculated using equation 3-10.

$$h = \frac{Nu_{D_h} \cdot k}{D_h} \qquad 3\text{-}10$$

where k is the thermal conductivity of the fluid evaluated at the film temperature.

Using equations 3-10, 3-3, 3-4 and the assumption that the wall temperature is uniform and equal to the average of the hot and cold side temperatures, all relevant temperatures and heat transfer coefficients are calculated. The physical size of the heat exchanger can then be determined, as discussed below.

Just as in the simplified SCO_2 Brayton model, the heat exchanger is divided into N nodes, each of which are assigned the same amount of heat transfer. The heat exchanger model then calculates the UA required for each node. In the detailed model, however, the physical size of each node is calculated. With the number and width of the channels specified and fixed for each side of the heat exchanger, the length of each node is dictated by the heat transfer coefficients on each side of the heat exchanger, the thermal conductivity of the heat exchanger metal, the heat exchanger web thickness, and a fouling factor. The length of a node is given below.

$$\Delta x = \frac{UA_i \cdot \left(\frac{1}{h_{H,i}} + \frac{1}{h_{C,i}} + \frac{th}{k_m} + ff \right)}{W_{ch} \cdot N_{ch}} \qquad 3\text{-}11$$

where $h_{H,\mathrm{i}}$ and $h_{C,\mathrm{i}}$ are the heat transfer coefficients on the hot and cold side of the heat exchanger, respectively. The heat exchanger web thickness is given by th, and the thermal conductivity of the heat exchanger metal is k_m. The total fouling factor for the heat exchanger (fixed throughout the detailed model as 0.00035 m^2-K/W (www.engineeringpage.com)) is ff, the width of each channel is W_{th} and the number of channels is N_{ch}.

With the physical size and geometry of the heat exchangers calculated, it is possible to calculate the pressure drop due to flow through the heat exchanger.

Pressure Drop

The pressure drops due to flow through the heat exchangers were neglected in the simplified model, but are calculated in the detailed model in order to better characterize the overall cycle performance. Neglecting the pressure drops could lead to the selection of heat exchanger geometries that requires fluid to flow through extremely small passages at high velocity in order to encourage high convective heat transfer coefficients. In reality, however, this type of heat exchanger would result in a large pressure drop, degrading cycle performance either by decreasing turbine power, increasing compressor power, or increasing parasitic salt or water pumping power. Characterizing realistic heat exchanger designs requires consideration for both heat transfer and fluid pressure drops. The pressure drops due to flow through the heat exchangers are broken into two categories. The first category is *form losses*, which represent the irrecoverable pressure drops due to geometry changes such as entrance and exit effects from the heat exchanger channels. The second category is friction losses.

Form losses are calculated using equation 3-12 (Dostal, 2004).

$$\Delta P_{form} = C \cdot \rho \frac{v^2}{2} \qquad 3\text{-}12$$

where C is the form loss coefficient, ρ is the density of the fluid at the relevant location (heat exchanger entrance for entrance losses, and heat exchanger exit for exit losses), and v is the fluid velocity at the relevant location. The entrance form loss coefficient was fixed at 0.5 for the entrance and 1.0 for the exit for straight channels (Dostal, 2004).

Friction losses are calculated at each heat exchanger node using the fully developed friction factor calculated from the Petukhov correlation for smooth walls (Nellis and Klein, 2009), shown below as equation 3-13.

$$f_{fd} = \frac{1}{\left[.79\ln(\mathrm{Re}_{D_h}) - 1.64\right]^2} \qquad 3\text{-}13$$

From the definition of the friction factor (Nellis and Klein, 2009), the pressure drop across each node is calculated using equation 3-14.

$$\Delta P_i = f_{fd,i} \cdot \left(\frac{\Delta x_i}{W_{ch}}\right) \cdot \rho_i \cdot \left(\frac{v_i^2}{2}\right) \qquad 3\text{-}14$$

where the subscript *i* represents a quantity evaluated at a specific node, and *v* represents the velocity of the fluid. Equations 3-12, 3-13, and 3-14 allow the detailed heat exchanger to calculate pressures at each node.

3.2 Cycle Control

Compressor Inlet Temperature Control

The simplified model features fixed high capacitance rates in the primary heat exchanger and precooler, respectively. This simplification is not appropriate for real world applications for two reasons. First, the inherently transient nature of solar applications will require the cycle to operate with varying molten salt temperatures at the primary heat exchanger inlet. Second, changing salt and cooling water temperatures would mean the CO_2 temperature at the compressor inlet would vary significantly. This variation could result in very large CO_2 density changes and at the very least would greatly impact performance. In the worst case, these large density changes could damage the compressor. The detailed model uses a control method which addresses both of these issues, allowing for transient operation as the salt temperature changes and avoiding substantial density changes at the compressor inlet.

The mass flow rate of molten salt through the primary heat exchanger, and the mass flow rate of cooling water through the precooler are both fixed. While the temperature of the molten salt

changes, the temperature of the CO_2 at the compressor inlet is fixed by controlling the temperature of the cooling water. Figure 3-8 illustrates this control strategy.

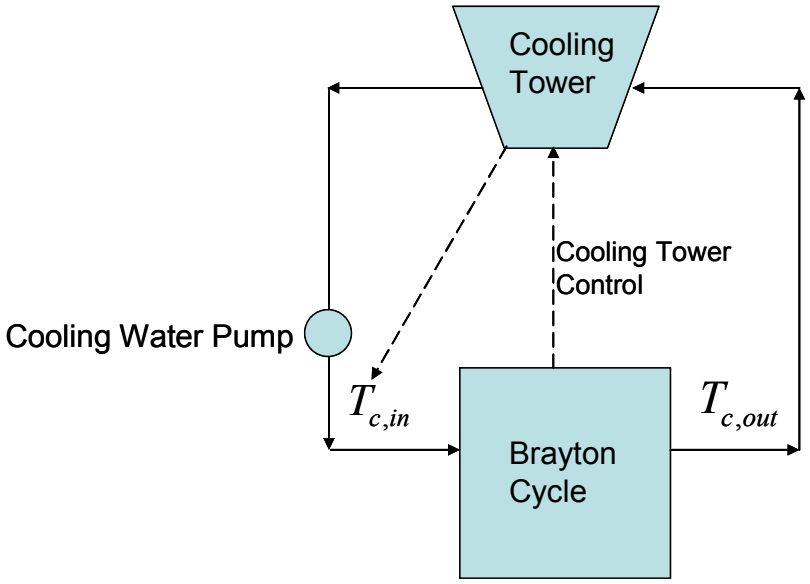

Figure 3-8: Compressor Inlet Temperature Control

With the mass flow rates of cooling water and molten salt fixed, the Brayton cycle model need only control the cooling water temperature to fix the CO_2 compressor inlet temperature. This is accomplished by controlling the cooling tower (modeled externally to the EES SCO_2 Brayton model to allow for transient simulation in TRNSYS) to output cooling water at a specific temperature.

Working Fluid Flow Rate

The simplified model fixes the mass flow rate of CO_2 through the cycle. This simplification, however, does not account for the significant density changes that occur at the compressor inlet. A fixed mass flow rate of CO_2 therefore corresponds to a greatly varying volumetric flow rate through the compressor. This does not accurately reflect a fixed compressor size, and more closely models a situation where the size of the compressor is different at each operating condition.

The model makes a first-order approximation of a fixed compressor size by fixing the CO_2 volumetric flow rate at the compressor inlet. This method is not perfect, but operates under a more reasonable assumption than fixing the CO_2 mass flow rate.

Now that the detailed model has been introduced, chapter 4 will discuss the design process which led to the selection of the design parameters for the SCO_2 Brayton Cycle.

4 Alternate Cooling Strategy Modeling Methodology

With Concentrating Solar Power (CSP) installations most well suited for arid desert environments with consistently clear skies, water consumption is of primary economic and environmental concern. Because the supercritical CO_2 Brayton cycle rejects heat across a range of temperatures, its heat rejection system can potentially be configured for air-cooling or hybrid water/air cooling. A hybrid-cooled Brayton power block can be designed to take advantage of the large temperature difference between CO_2 exiting the regenerator and ambient air dry bulb to accomplish a significant fraction of required cycle heat rejection; thereby, reducing water consumption associated with traditional power cycles. Given sufficiently cool ambient dry bulb temperature conditions, it would also be possible to completely avoid wet cooling, and so create an air-cooled Brayton cycle.

This chapter explains the modifications made to the detailed model presented in Chapter 3 to accommodate simulating the performance of a hybrid-cooled or air-cooled Brayton cycle. The methodology behind the modeling of a CO_2 air-cooler is explored, along with the implementation of the air-cooler as part of the hybrid-cooled and air-cooled Brayton cycles.

At the end of this chapter, the modeling methodology underpinning the water-cooled, hybrid-cooled, and air-cooled Brayton cycles will have been explained. Chapter 5 contains an explanation of the methodology used to select the designed parameters of each of these cycles, along with a comparison of their performance at a fixed design point.

4.1 Air-cooler Modeling Methodology

The primary modification to the detailed water-cooled model made to assess hybrid and air-cooled cycles is the addition of an air-cooler. The air-cooler is modeled as a fin-tube heat exchanger with a geometry taken from EES's compact heat exchanger library (Klein, 2010), shown in Figure 4-1.

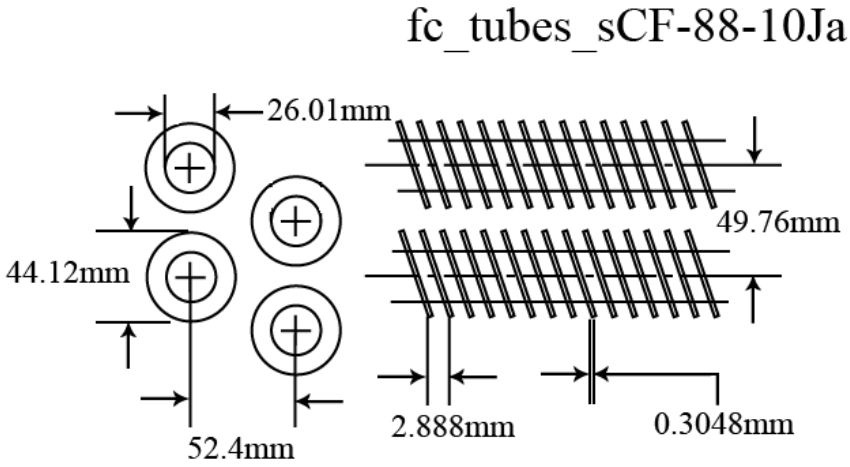

Figure 4-1: Air-cooler Geometry (Klein, 2010)

The geometry type 'fc_tubes_sCF-88-10Ja' corresponds to finned circular tubes with surface CF-8.8-1.0J (Klein, 2010). CO_2 passes through the tubes of the heat exchangers – with ambient air blowing across the fins. The heat exchanger area is defined in equation 4-1 as:

$$Area_{HX} = L \cdot Area_{fr} \cdot \alpha \qquad 4\text{-}1$$

where L is the length of the heat exchanger, $Area_{fr}$ is the frontal area of the heat exchanger, and α is the ratio of the heat exchanger area to the total volume of the heat exchanger (a parameter defined by the geometry of the heat exchanger). A diagram of this relationship is shown in Figure 4-2.

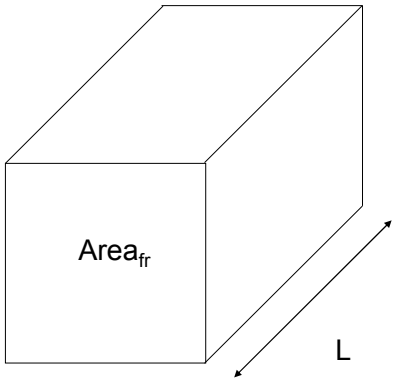

Figure 4-2: Aircooler Geometry Schematic

$Area_{HX}$, in turn, is used to calculate the UA as shown in equation 4-2.

$$UA = Area_{HX} \cdot h \qquad 4\text{-}2$$

where h represents the heat transfer coefficient on the air-side of the heat exchanger, as calculated by EES' built in compact heat exchanger function – which in turn employs correlations for the Colburn j factor for the air-side of the heat exchanger as originally developed by Kays and London (1984). Note that the resistances to heat transfer defined by the tube walls and CO_2 are neglected as they are assumed to be small compared to the air-side resistance.

Fan power is calculated using equation 4-3.

$$power_{fan} = \frac{\Delta P \cdot \dot{m}_{air}}{\rho_{air} \cdot \eta_{fan}} \qquad 4\text{-}3$$

with ρ_{air} representing the density of air at an average temperature between inlet and outlet conditions, ΔP representing the air-side pressure drop calculated by EES' built in compact heat exchanger routine (based on correlations developed by Kays and London (1984) for friction factor), and η_{fan} represents fan efficiency (assumed to be fixed at 50% throughout this thesis).

The CO_2 pressure drop is calculated by means of the Petukhov correlation for friction factor in smooth tubes (Nellis and Klein, 2009) shown in equation 4-4.

$$f_{fd} = \frac{1}{\left[.79\ln(\mathrm{Re}_{D_h}) - 1.64\right]^2} \qquad 4\text{-}4$$

From the definition of the friction factor (Nellis and Klein, 2009), the pressure drop is calculated using equation 4-5.

$$\Delta P = f_{fd} \cdot \left(\frac{L}{D}\right) \cdot \rho \cdot \left(\frac{v^2}{2}\right) \qquad 4\text{-}5$$

where v represents the velocity of the CO_2 , D represents the diameter of the tube, and L represents the length of the air-cooler (the aspect ratio of the air-cooler is kept constant as its size changes, but a typical value is 1 meter).

The cooler air inlet temperature and pressure are set by ambient conditions. The CO_2 inlet temperature, pressure and mass flow rate are set by conditions at the regenerator outlet. The area and length of the heat exchanger are specified by the user. If the air-cooler outlet CO_2 temperature is set, the air-cooler model will calculate the necessary fan power at a given ambient temperature to reject heat from the cycle. If the fan power is specified, the air-cooler model will calculate the resulting CO_2 outlet temperature.

4.2 Hybrid-Cooled Model

Air-cooler Implementation

The hybrid-cooled Brayton cycle is cooled using the same methodology as was employed for the water-cooled model (shown in schematic form in Figure 4-3) but with the addition of an air-cooler before the precooler.

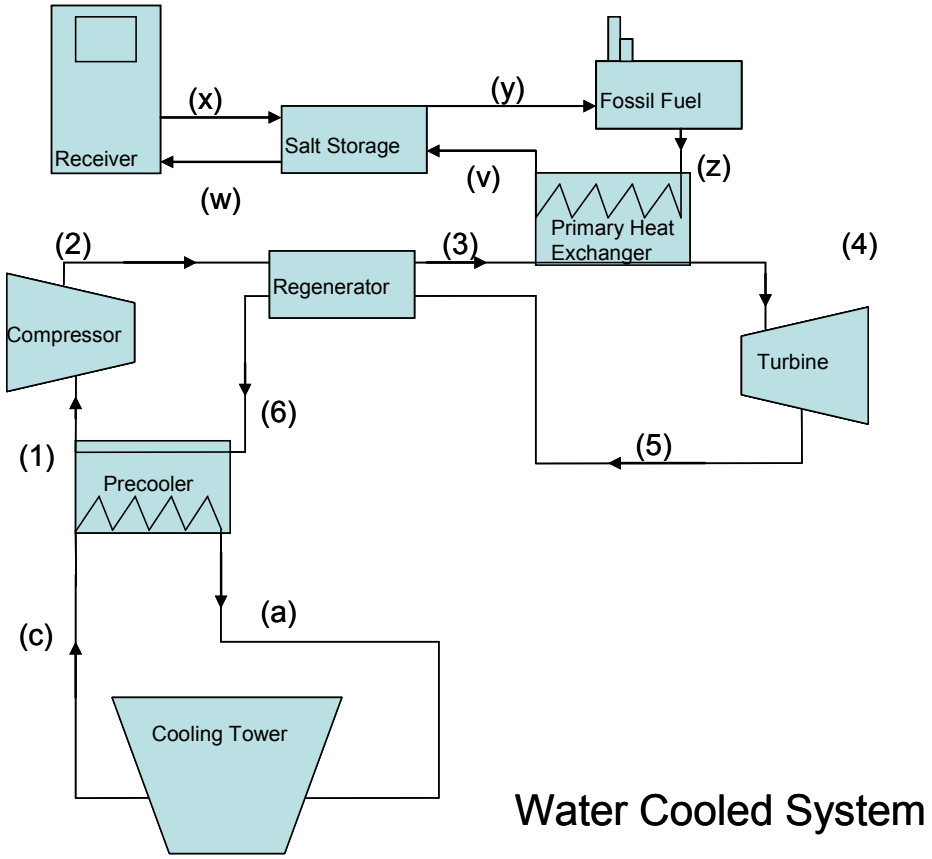

Figure 4-3: Water-Cooled Brayton Equipment Schematic

Heat rejection in the hybrid-cooled Brayton cycle is accomplished with the addition of an air-cooler upstream of the water-cooled precooler as shown in Figure 4-4 .

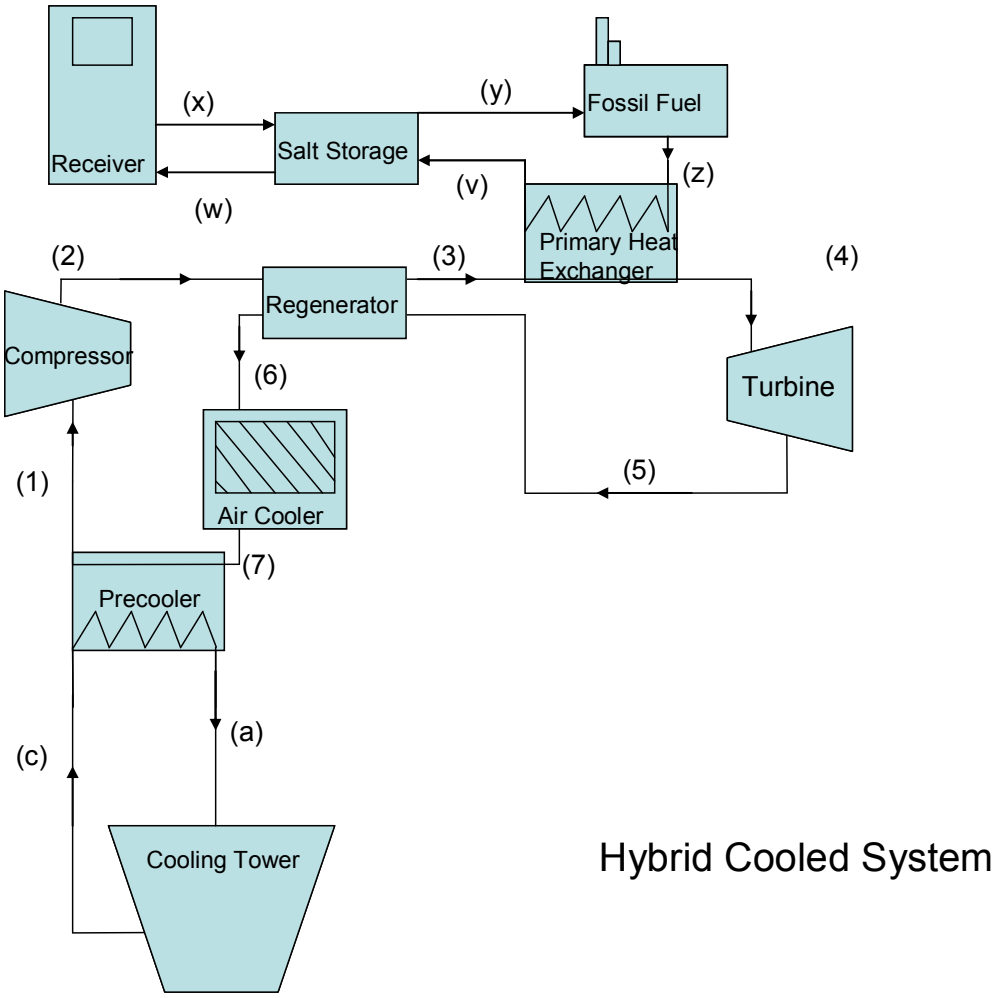

Figure 4-4: Hybrid-Cooled Brayton Equipment Schematic

The air-cooler cools the CO_2 before it enters the precooler. This location was chosen to take advantage of the maximum temperature difference between CO_2 and the ambient air.

The control methodology employed for the hybrid-cooled cycle is identical to that employed for the water-cooled cycle (see Chapter 3 for details on water-cooled control) with the exception that the added air-cooler operates at a fixed fan power regardless of ambient temperature. During cool weather, the temperature of the CO_2 at the precooler inlet is significantly reduced. This in turn reduces the load on the cooling tower, which also reduces water usage. When ambient temperatures are high, however, the cooling provided by the air-cooler is insufficient and the cooling tower must be used to provide the required rate of heat rejection.

Limitations

Although rejecting heat from the CO2 occurs across a wide range of temperatures, the specific heat of CO_2 spikes dramatically near its critical point. Also, as CO_2 is cooled towards its critical point it experiences a rapid increase in density. These behaviors are shown in Figure 4-5 for a pressure of 8 MPa, representative of a nominal pressure ratio of 3.1 (for more details on pressure ratio selection see Chapter 5).

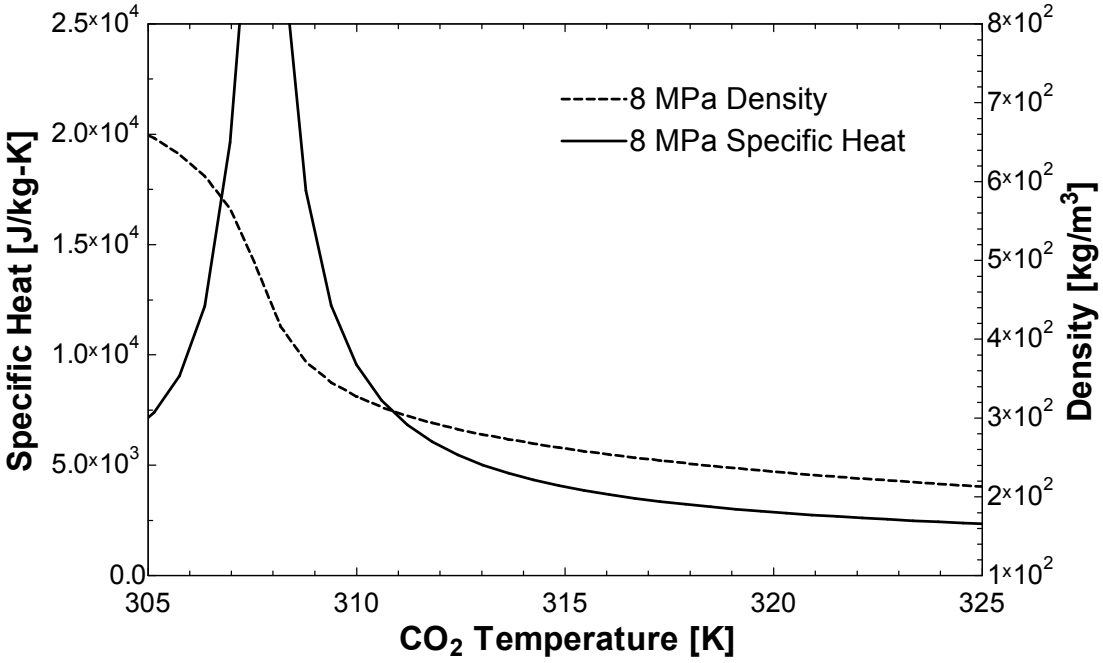

Figure 4-5: CO_2 Properties near Critical Point

To reach the high densities at the compressor inlet that are required for good cycle performance, it is necessary to cool the CO_2 through a massive spike in specific heat. Although cooling occurs across a wide range of temperatures, the majority of heat rejection occurs across a much narrower range of relatively low temperatures. In this sense, heat rejection characteristics for the SCO_2 Brayton cycle at typical pressure ratios (~3) are similar to that of a conventional Rankine cycle, in that the majority of heat is rejected at a narrow range of temperatures. This minimizes the potential benefit of hybrid cooling to water consumption at traditional pressure ratios. Figure 4-6 shows the efficiency of a hybrid-cooled cycle across a range of air-cooler volume ratios (defined as the ratio of the air-cooler volume to the total volume, including air-cooler, of all heat exchangers) at several ambient temperatures, with the percentage of cooling load met by air

listed for each ambient temperature. For details the sensitivity of the hybrid-cooled Brayton cycle to design parameters, see Chapter 5, which focuses on cycle design and performance. Note that 'approach' is defined by the minimum temperature difference between the ambient air and the CO_2 being cooled, and that '% AC' in Figure 4-6 refers to the percentage of total cooling accomplished by air. Also note that for the hybrid-cooled cycle shown in Figure 4-6 the air-cooler outlet temperature is fixed at 310 K with a minimum approach of 5 K, unless the ambient temperature exceeds 305 K. When the ambient temperature exceeds 305 K the air-cooler maintains a 5 K approach by allowing the air-cooler outlet temperature to increase.

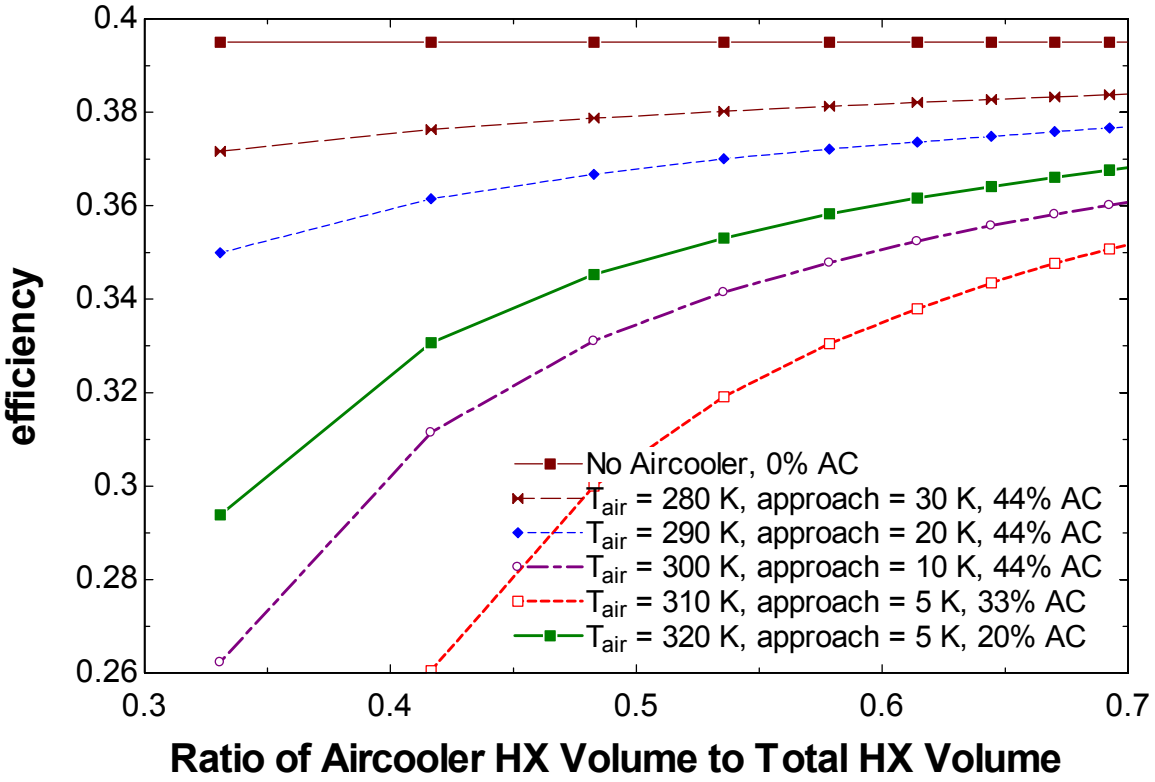

Figure 4-6: Representative Hybrid-Cooled Brayton Performance Across a Range of Ambient Temperatures and Air-cooler Sizes

Performance is significantly reduced for all ambient temperatures, with less than half of the cooling load met by air. At higher ambient temperatures, reasonable performance is only obtained with an air-cooler volume that is very large relative to the total volume of all heat exchangers in the cycle.

The effect of the dramatic specific heat spike might be avoided, however, by increasing the pressure of the CO_2 at the compressor inlet. This can be accomplished (with a fixed compressor outlet pressure) by reducing the cycle's pressure ratio. Figure 4-7 shows the relationship between CO_2 specific heat and density for a variety of temperatures and pressures.

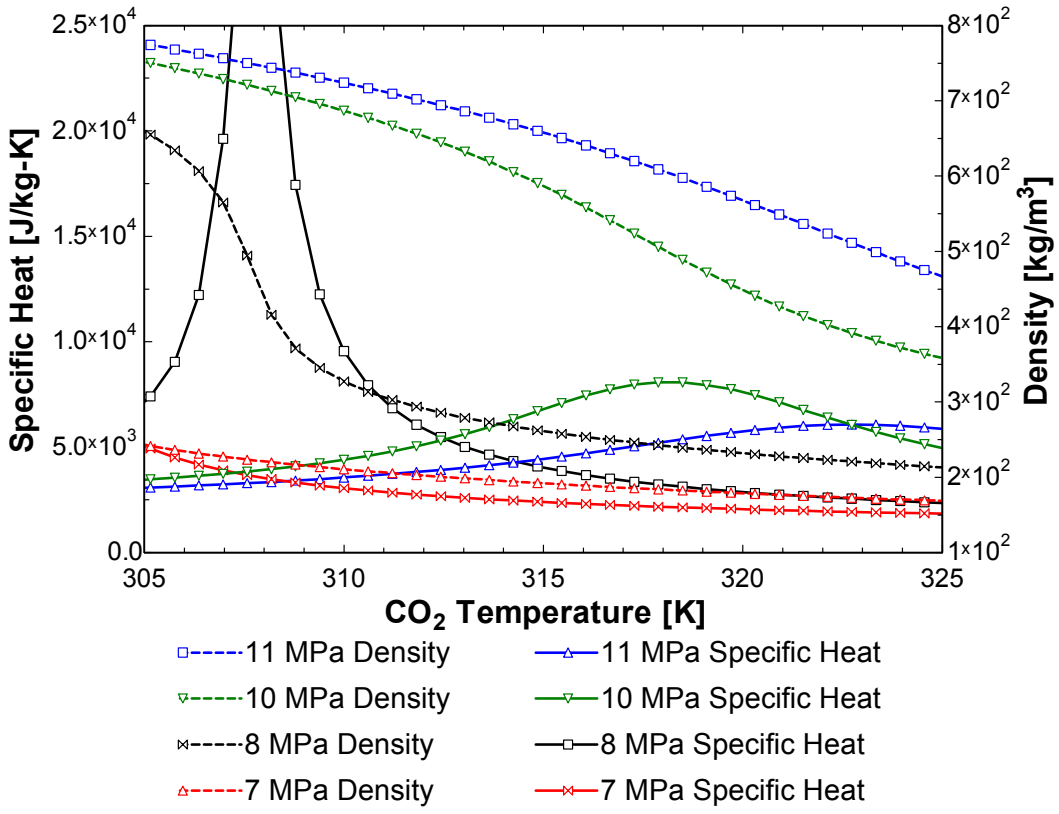

Figure 4-7: CO_2 Properties Across a Range of Pressures

At pressures significantly above the critical pressure for CO_2, the temperature-dependent density change is less abrupt. Also, the specific heat peak is flattened and shifted to a higher temperature. It is possible to achieve high (though somewhat reduced) densities at higher compressor inlet temperatures, while avoiding a drastic spike in specific heat. This would allow an air-cooler to perform a greater proportion of the total cooling. While overall thermodynamic efficiency would be degraded by the reduced pressure ratio, water savings could be significant.

If a pressure ratio was to be chosen that shifted the specific heat peak to a high enough temperature, it would be possible to design a cycle that operates entirely via dry cooling. This

concept is the focus of the next section, where the modeling considerations behind the air-cooled Brayton cycle are investigated.

4.3 Air-Cooled Model

The air-cooled Brayton is shown in schematic form in Figure 4-8.

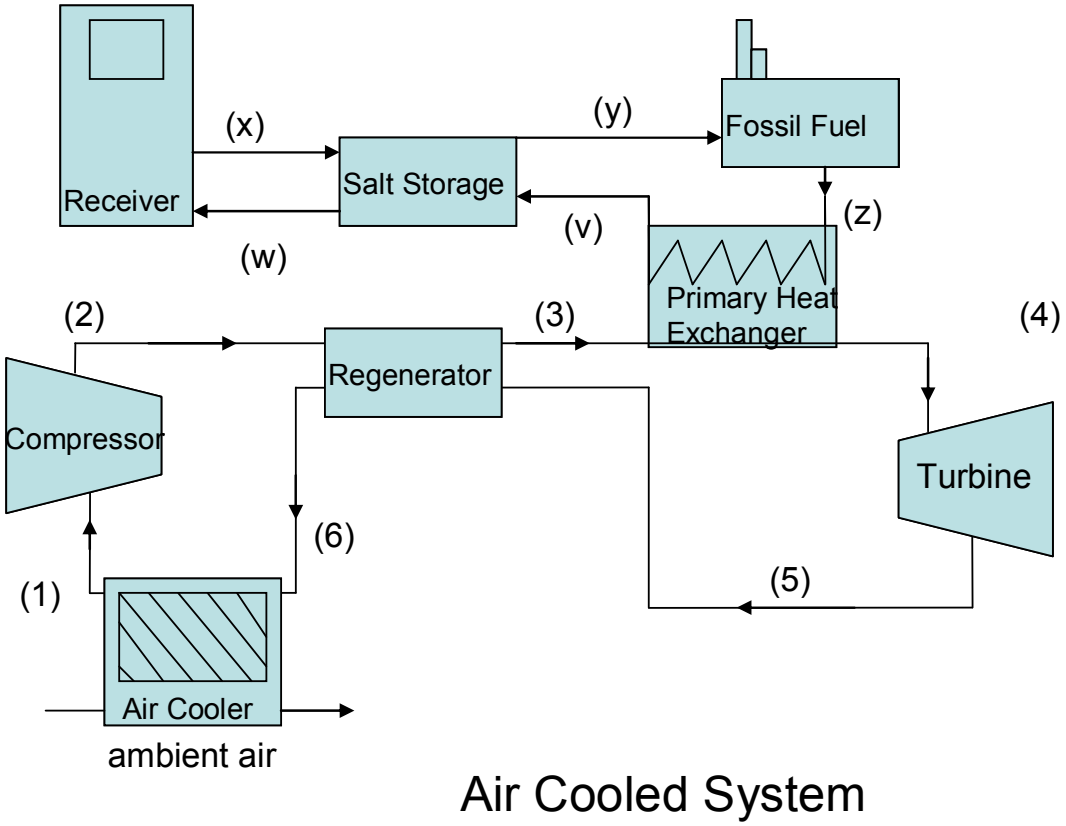

Figure 4-8: Air-Cooled Brayton Equipment Schematic

In an air-cooled cycle, the precooler is an air-cooler, allowing the cycle to run without any water use. The control strategy for the air-cooled cycle is similar to that used in the water-cooled cycle, with the compressor inlet temperature fixed. The compressor inlet temperature is fixed by controlling the fan power to the air-cooler. When ambient temperatures are low, the air-cooler requires a reduced level of fan power to reject heat from the cycle and reach the compressor inlet set temperature. At higher ambient temperatures, fan power requirements increase and become more significant. If the ambient temperature is within a user-specified minimum approach

temperature of the design compressor inlet temperature, the compressor inlet temperature is set to the ambient temperature plus the minimum approach temperature. This is done to address situations where unreasonably large fan powers are required to meet the design compressor inlet temperature, or situations where reaching the design compressor inlet temperatures is impossible (where the ambient temperature exceeds the design compressor inlet temperature). The logic used to specify the compressor inlet temperature is shown in equation 4-6.

$$T_{comp,inlet} = \max(T_{comp,inlet,design}, T_{ambient} + \Delta T_{approach}) \qquad 4\text{-}6$$

where $T_{comp,inlet,design}$ is the design compressor inlet temperature, $T_{ambient}$ is the ambient temperature, and $\Delta T_{approach}$ is the approach temperature. Figure 4-9 shows the impact of ambient temperature on performance for the air-cooled cycle with several different air-cooler sizes.

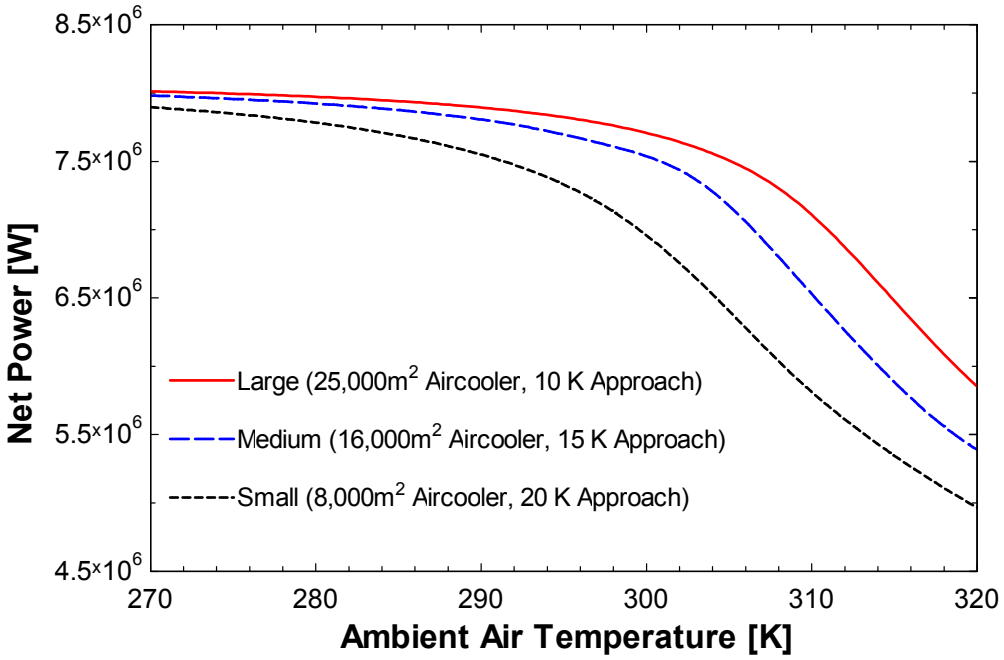

Figure 4-9: Air-Cooled Brayton Performance v. Ambient Temperature

Net power degrades as the ambient temperature increases, due to increased fan power requirements and higher compressor inlet temperatures. A larger air-cooler reduces required air velocities, pressure drops, and fan power requirements. This in turn reduces sensitivity to increased ambient temperatures.

Chapter 5 discusses the design and performance characteristics of the air-cooled Brayton cycle in more detail, along with discussions of the water and hybrid-cooled Brayton cycles.

Full code listings for the hybrid-cooled and air-cooled cycles can be found in Appendices F and G respectively.

5 Cycle Design

Chapters 2 and 3 introduced the simplified and detailed SCO_2 Brayton models. Chapter 4 discussed modifications made to assess the feasibility of dry and hybrid cooling. This chapter examines the methodology behind choosing the design parameters for the water, hybrid, and dry cooled SCO_2 Brayton cycles for annual performance evaluation.

5.1 Cycle Design vs. Cycle Optimization

As the SCO_2 model became more detailed, it became clear that optimization with the goal of maximizing cycle efficiency (or power) is not a simple task. One aspect of the complexity arises when considering what variables to include in the optimization: heat exchanger channel sizes (6 variables), overall heat exchanger allocation (3 variables), heat exchanger aspect ratios (3 variables), nominal cycle pressures/ratios (2 variables), etc. Quickly, the number of variables in the optimization rises, which greatly complicates the analysis to determine the combinations that will maximize efficiency (or power).

To illustrate issues that arise in the optimization, consider the prospect of optimizing the heat exchangers for efficiency-only. If left unconstrained, the optimization process will tend to drive heat exchangers to very small channel sizes and very long and thin heat exchangers. As the heat exchanger channels become long and narrow, the pressure drop through the heat exchanger increases dramatically, leading to a reduction in the pressure and density of CO_2 at the compressor inlet. Since the mass flow rate of CO_2 through the cycle is based on a fixed

volumetric flow rate at the compressor inlet, heat exchangers 'optimized' for efficiency will likely yield less than desirable performance on a power basis. In the present analysis the volumetric flow rate of CO_2 through the compressor is fixed to approximate fixing the size of the compressor as the design point changes. In the case of very long and thin heat exchangers, these heat exchangers will reduce the mass flow rate of CO_2 through the cycle, which greatly reduces the cycle's power output. Clearly, there has to be a balance between a cycle design that seeks to maximize efficiency while providing a reasonable level of power production capability. While in an actual plant design this balance would be struck by optimizing cost, the scope of analysis presented in this thesis does not include any economic analysis.

The source for cycle cooling or heat rejection also complicates optimization. The current water-cooled SCO_2 cycle model assumes the compressor has a fixed inlet CO_2 temperature controlled by varying the temperature of the cooling water (set external to the Brayton model by varying cooling tower fan power). Cooling fan power is therefore external to this EES analysis.

These two issues have led to abandoning the idea of optimization. Instead the design procedure relies on previous findings in the more narrow power block optimization studies that were performed previously. The goal here is to make reasonable design choices that strike a balance between power and efficiency, while not requiring unreasonably low cooling water temperatures.

5.2 Relative Sensitivity of Cycle Performance to Designed Parameters

Table 5-1 shows a baseline cycle configuration, with a listing of those parameters that are fixed and those parameters that are later selected for improving the system performance. The performance characteristics are also listed.

Table 5-1: Baseline Cycle Parameters

Water-Cooled: Baseline (Medium HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*1166*	
Regenerator Area [m^2]	*1166*	
Precooler Area [m^2]	*1166*	
PHX Salt Side Channel Width/Depth [mm]	*2*	
PHX CO2 Side Channel Width/Depth [mm]	*2*	
Regenerator Hot Side Channel Width/Depth [mm]	*2*	
Regenerator Cold Side Channel Width/Depth [mm]	*2*	
Precooler CO2 Side Channel Width/Depth [mm]	*2*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*5000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*5000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*5000*	
Cooling Water Design Demand Temperature [K]	**300.3**	Performance
Cooling Water Outlet Temperature [K]	**308.7**	
Power [MW]	**11.75**	
Efficiency	**40.18%**	

Cycle design is accomplished by individually tuning each parameter in sequence to achieve a design. The order in which the parameters are varied is based on the relative sensitivity of the cycle to each parameter, proceeding from the most sensitive parameter to the least. The following section briefly examines the relative sensitivity of the cycle to each parameter across a small range. This section is not meant to evaluate the complex behavior of the cycle in response to varying each parameter – but just the relative sensitivity of the cycle to each parameter. Details on how the cycle responds to each changing each parameter are found in Section 5.3.

Note that "power" refers to "net power," where *net power* is defined as the gross power produced by the turbine less the power required by the compressor, pumping high temperature salt, and cooling water as given by equation 5-1.

$$\dot{W}_{net} = \dot{W}_{turbine} - \dot{W}_{compressor} - \dot{W}_{pumping,salt} - \dot{W}_{pumping,water} \qquad 5\text{-}1$$

Efficiency is defined as the ratio of net power to the total heat transfer rate supplied by the high temperature salt to the CO_2 in the primary heat exchanger, as shown in equation 5-2.

$$\eta = \frac{\dot{W}_{net}}{\dot{Q}_{PHX}} \qquad 5\text{-}2$$

Total Heat Exchanger Area

A reasonable value for total heat exchanger area of the precooler, heater, and regenerator had to be chosen. This choice was made by evaluating the performance of the cycle at a variety of total heat exchanger areas and selecting a point at which performance improvements come only at the cost of very large increases in heat exchanger area. Figure 5-1 shows cycle performance as a function of total heat exchanger area normalized by the chosen design value (3500 m^2 for the ~10 MW design plant, also used as the design total heat exchanger area in subsequent sections).

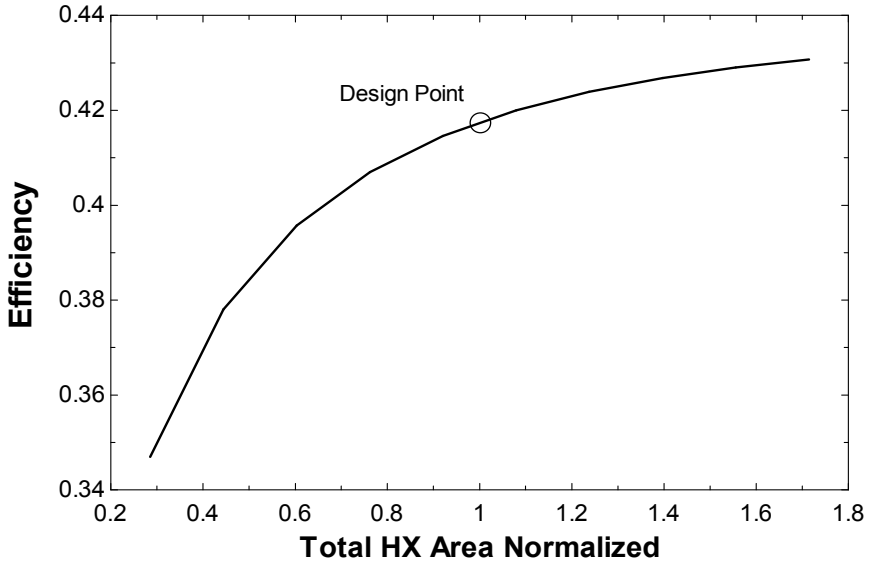

Figure 5-1: cycle performance as a function of total HX area

Significant increases in heat exchanger area beyond the design value result in only increasingly small performance improvements. Note that the heat exchanger distribution fractions are those used as the design values.

Pressure Ratio

Figure 5-2 shows the power and efficiency of the cycle as a function of pressure ratio. A design pressure ratio of 2.9 was chosen because it provided high efficiency and net power. Note that the pressure ratio refers to the ratio between the compressor and turbine outlet pressures, and that the compressor outlet pressure is always fixed at 25 MPa for maximum performance (as shown in Chapter 2) while staying within operating conditions acceptable for PCHE heat exchangers (Dostal, 2009). With a fixed compressor outlet pressure, an increased pressure ratio corresponds to a reduced turbine outlet pressure.

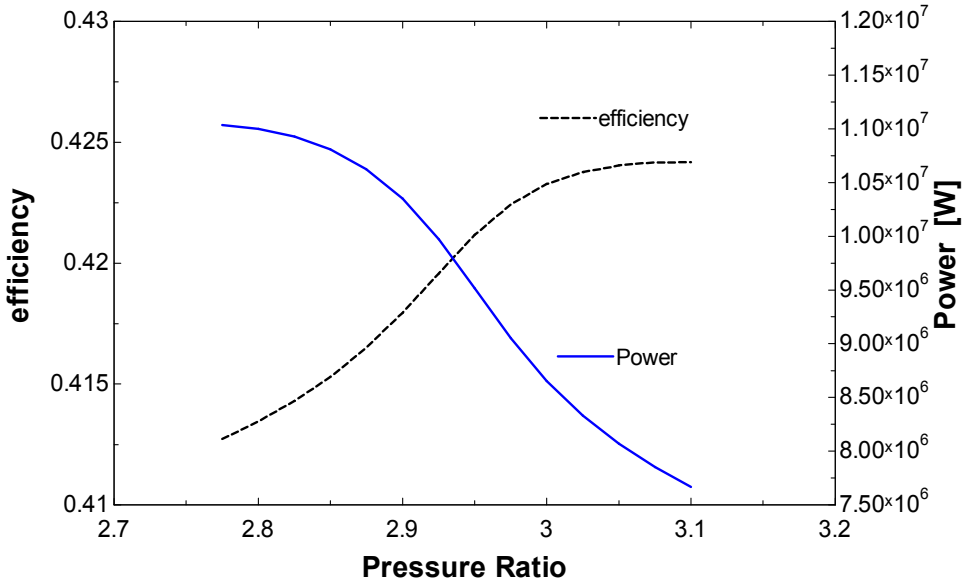

Figure 5-2: cycle performance as a function of pressure ratio

Note that increasing pressure ratios provide greater efficiency but the reduced low side pressure leads to reduced working fluid mass flow rate and cycle power. This improved efficiency is, principally, a function of the reduced working fluid mass flow rate due to increased pressure drops. The cycle's efficiency is relatively sensitive to pressure ratio, with a pressure ratio decrease from 3.0 to 2.8 resulting in an absolute efficiency decrease of greater than 1%.

Heat Exchanger Channel Size

Each heat exchanger has two sides, and each side has channels that can be sized independently. Figure 5-3 shows cycle efficiency as a function of the cold side regenerator channel width, which is representative of cycle sensitivity to channel width.

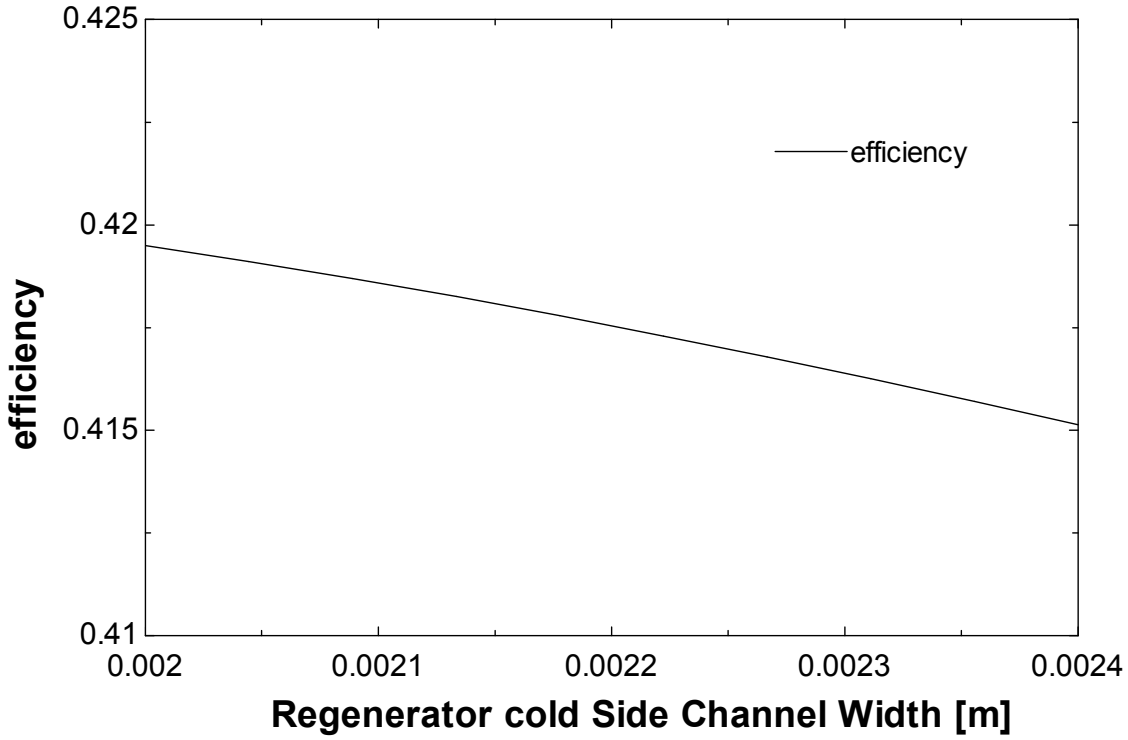

Figure 5-3: Cycle performance as a function of cold side regenerator channel width

Note that cycle performance is much less sensitive to channel width when compared with pressure ratio, with a 3% change in channel width resulting in an absolute efficiency impact of less than 0.1%.

Heat Exchanger Area Distribution

At a fixed total heat exchanger area, the distribution of that area between each heat exchanger is chosen to achieve a balance between efficiency and power. Figure 5-4 shows the sensitivity of cycle performance to the percentage of total heat exchanger area that is allocated to the precooler.

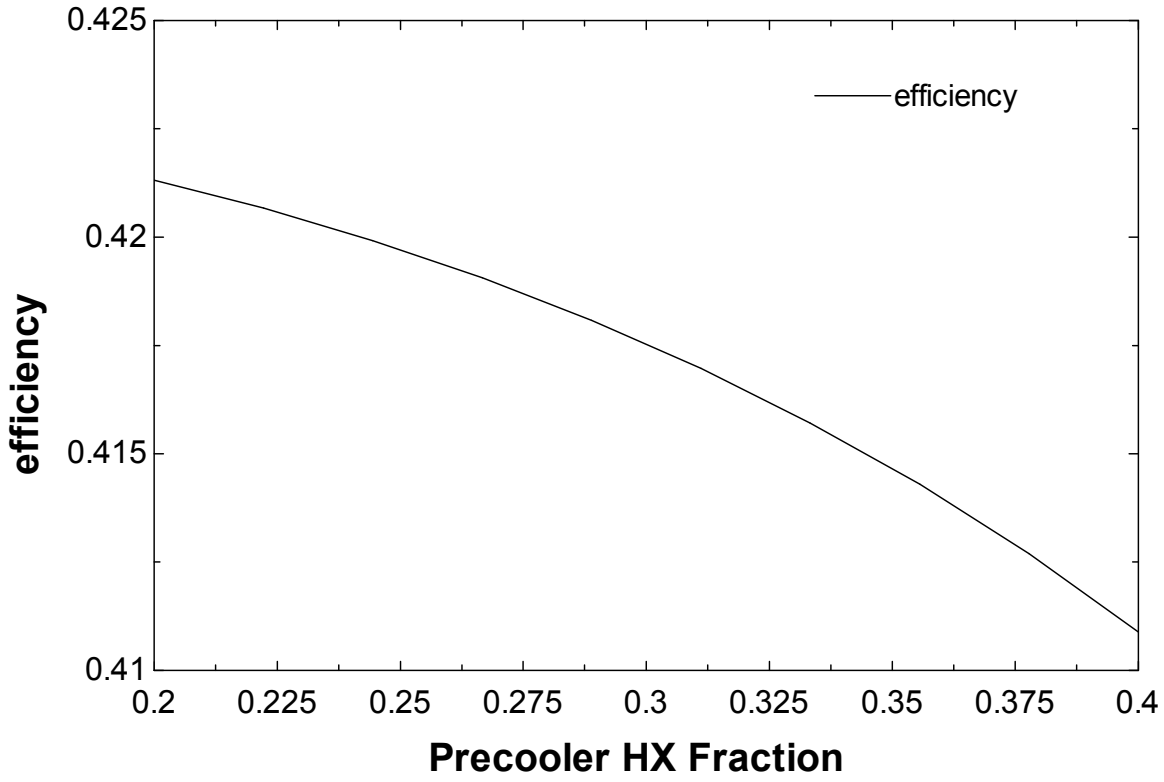

Figure 5-4: Cycle performance as a function of precooler area fraction

The cycle sensitivity to the precooler fraction ratio is relatively small, with a change of (+/-) ~3% in precooler fraction resulting in an absolute efficiency impact of less than 0.04%.

Heat Exchanger Aspect Ratio

Each heat exchanger has an overall *aspect ratio* here is defined as the number of channels per meter of heat exchanger length. A larger value corresponds to a shorter, wider heat exchanger with reduced channel velocities which leads to lower channel pressure drops and lower channel heat transfer coefficients. A well-chosen heat exchanger aspect ratio will balance pressure drops with heat transfer effectiveness for good cycle performance. Figure 5-6 shows the relationship between cycle efficiency and the aspect ratio of the regenerator.

.

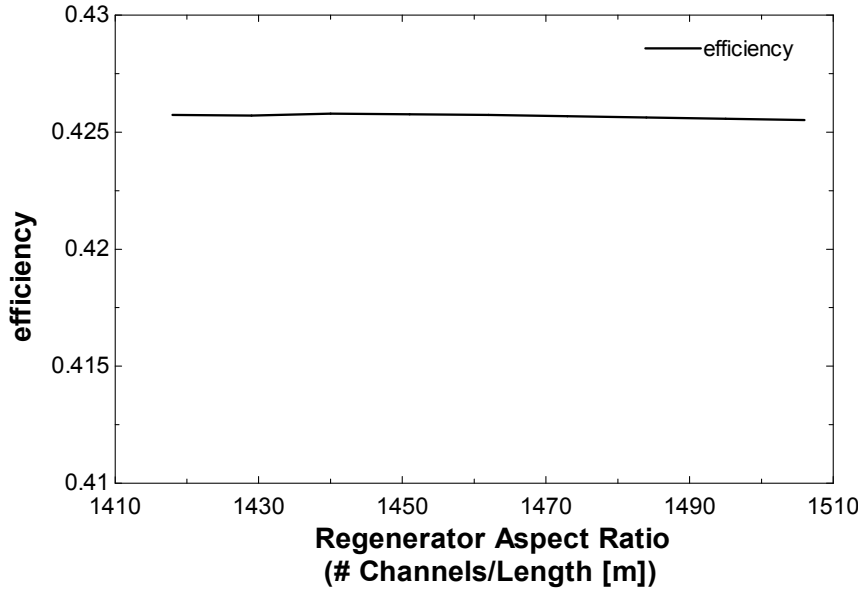

Figure 5-5

Figure 5-6: Cycle performance as a function of regenerator channel aspect ratio

Sensitivity to heat exchanger aspect ratio is relatively small, with a change of 3% in aspect ratio resulting in an absolute efficiency impact of less than 0.03%.

The relative sensitivity of cycle performance to design parameters is summarized in

Table 5-2.

Table 5-2: Summary of Cycle Performance Sensitivity to Design Parameters

Sensitivity Analysis @ Design Point		
Variable	Partial Derivative of Efficiency with Respect to Variable	% of Uncertainty wrt Efficiency
Pressure Ratio	0.04026	92.84%
Primary Heat Exchanger Salt Side Channel Size [m]	0.8611 [m^-1]	0.00%
Primary Heat Exchanger CO2 Side Channel Size [m]	-2.091 [m^-1]	0.16%
Regenerator Hot Side Channel Size [m]	-3.317 [m^-1]	0.24%
Regenerator Cold Side Channel Size [m]	-11 [m^-1]	3.99%
Precooler CO2 Side Channel Size [m]	-0.5639 [m^-1]	0.00%
Precooler Water Side Channel Size [m]	7.567 [m^-1]	1.56%
Regenerator Area Fraction	0.0215	0.64%
Primary Heat Exchanger Aspect Ratio [#/m]	-6.593e-8 [m]	0.00%
Regenerator Aspect Ratio [#/m]	-1.982e-6 [m]	0.43%
Precooler Aspect Ratio [#/m]	4.23e-7 [m]	0.13%

Cycle efficiency is most sensitivity to pressure ratio, followed by heat exchanger channel size, heat exchanger area distribution, and heat exchanger aspect ratio, respectively. This is the order in which the parameters were designed.

5.3 Performance Impact of Individual Parameters

When designing the cycle as a whole, it is useful to understand the impact that varying a single parameter has on cycle power and efficiency. The following section examines how changing each parameter impacts operation, and should be useful to anyone seeking to understand how varying a given parameter will impact performance. This section is divided into two parts, one of which applies to the water-cooled and hybrid cooled Brayton cycles, and one that applies to the air-cooled Brayton cycle.

Note that all parameters are varied one at a time with the remaining parameters fixed at the design values, which can be found later in the report in Table 5-3 (for water-cooled) and Table 5-4 (for air-cooled). Since parameters are varied one at a time, interactions between all the parameters are not captured. This section is simply meant to show the changes in cycle operation that result from changing each parameter. Overall cycle design must concern itself

with choosing a set of parameters that produce good performance while considering interactions (i.e. changing multiple parameters at once).

This section starts by examining how changing the pressure ratio of the system (with a fixed compressor outlet pressure of 25 MPa) impacts the cycle. The section proceeds from there into work done to investigate the impact of channel width, and then into the impact of heat exchanger area distribution. Finally the section examines the impact of heat exchanger aspect ratio (how long and thin, or short and wide the heat exchangers are).

5.3.1 Water/Hybrid Cooled Brayton

Pressure Ratio

The choice of pressure ratio largely depends on the goal of the cycle. A higher pressure ratio yields greater efficiency. Figure 5-7 shows the relationship between pressure ratio and cycle performances.

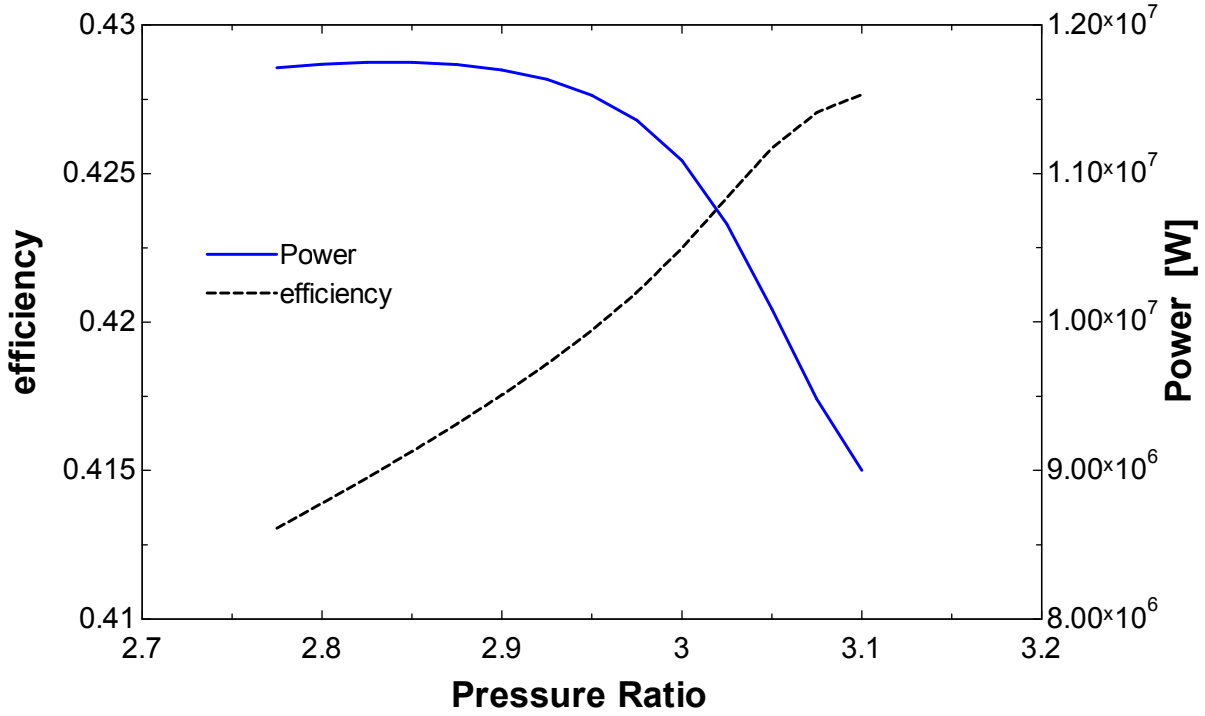

Figure 5-7: cycle performance as a function of pressure ratio

While higher pressure ratios improve efficiency, this comes at the expense of power. This is a result of reduced working fluid mass flow rate due to lower CO_2 density at the compressor inlet. This relationship is shown in Figure 5-8.

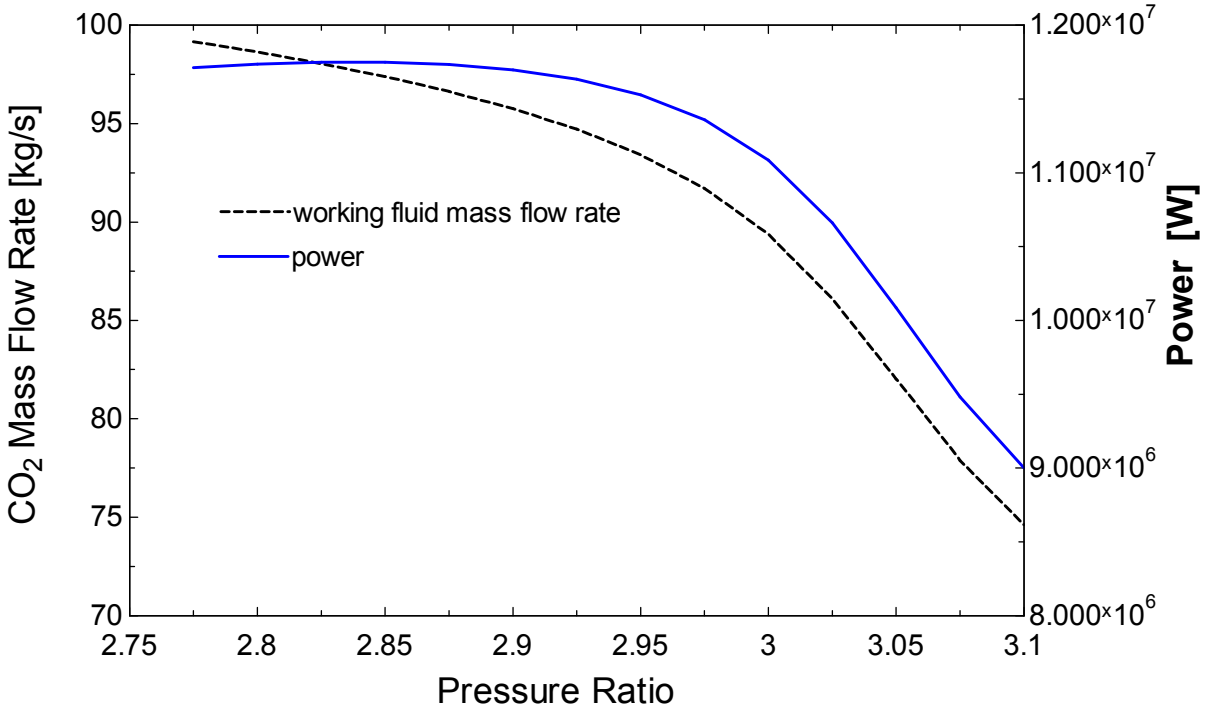

Figure 5-8: CO_2 Mass Flow Rate and Power v. Pressure Ratio

The relatively minor efficiency benefits of increasing the pressure ratio are primarily a function of reduced working fluid mass flow rate, relative to heat exchanger size. The efficiency gains should not be given too much weight, since the same gains could be realized by decreasing the compressor volumetric flow rate, which would also decrease power.

Primary Heat Exchanger Salt Side Channel Width/Depth

Channel sizes on the salt side of the primary heat exchanger must be designed to balance the pumping power required to pump salt through smaller channels against the improved heat transfer coefficients provided by increased flow rate. Figure 5-9 shows the relationship between primary heat exchanger channel size on the salt side and cycle performance.

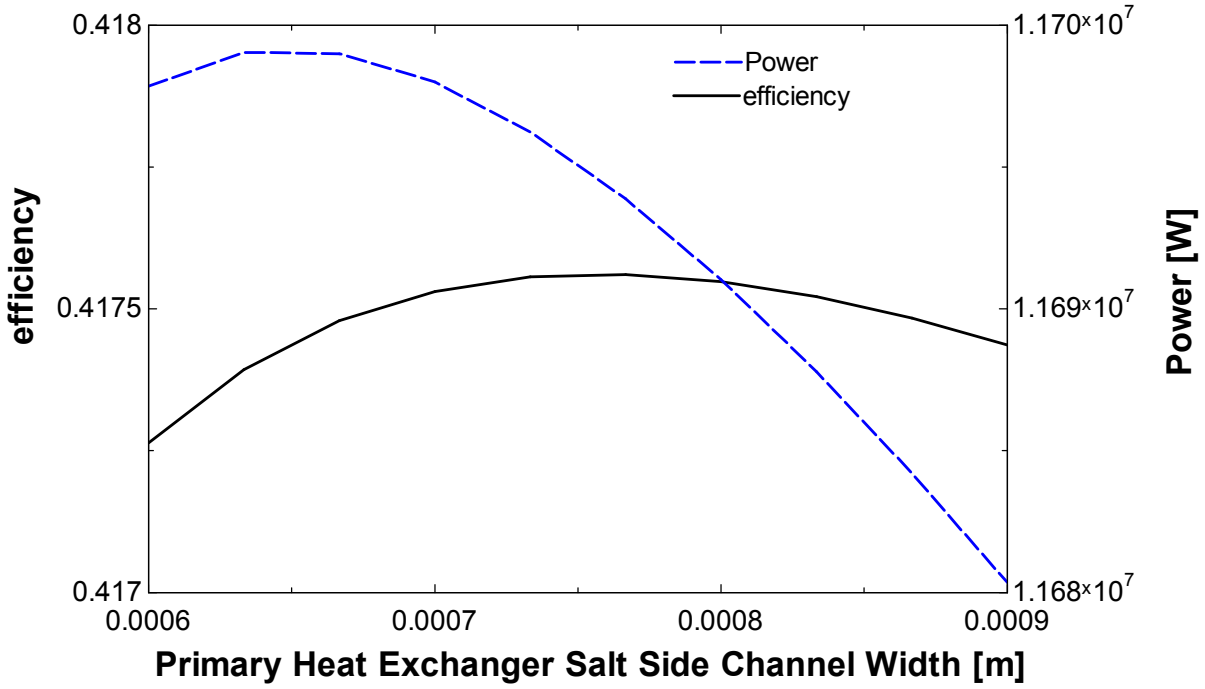

Figure 5-9: Cycle Performance v. PHX Salt Side Channel Size

Either power or efficiency might be optimized for this parameter, depending on design goals.

Primary Heat Exchanger CO_2 Side Channel Width/Depth

Choosing the primary heat exchanger's CO_2 side channel size involves balancing increased pressure losses against increased heat transfer coefficient. Smaller channels produce higher turbine temperatures and lower turbine pressures. Figure 5-10 shows the performance impact of these relationships.

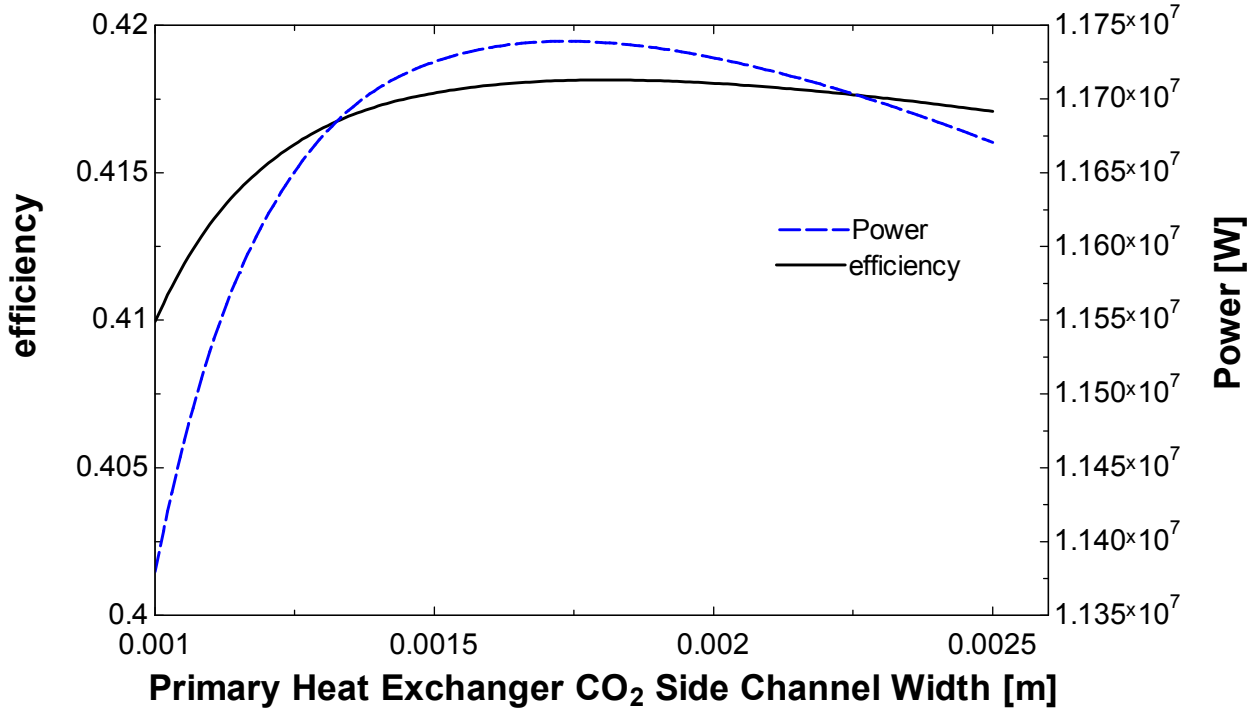

Figure 5-10: Cycle Performance v. PHX CO_2 Side Channel Size

Either power or efficiency might be optimized for this parameter, depending on design goals.

Regenerator Hot Side Channel Width/Depth

As with the CO_2 side of the Primary Heat Exchanger, selection of the channel size on the hot side of the regenerator involves balancing pressure drops against enhanced heat transfer. The relationship between cycle performance and this parameter is shown in Figure 5-11.

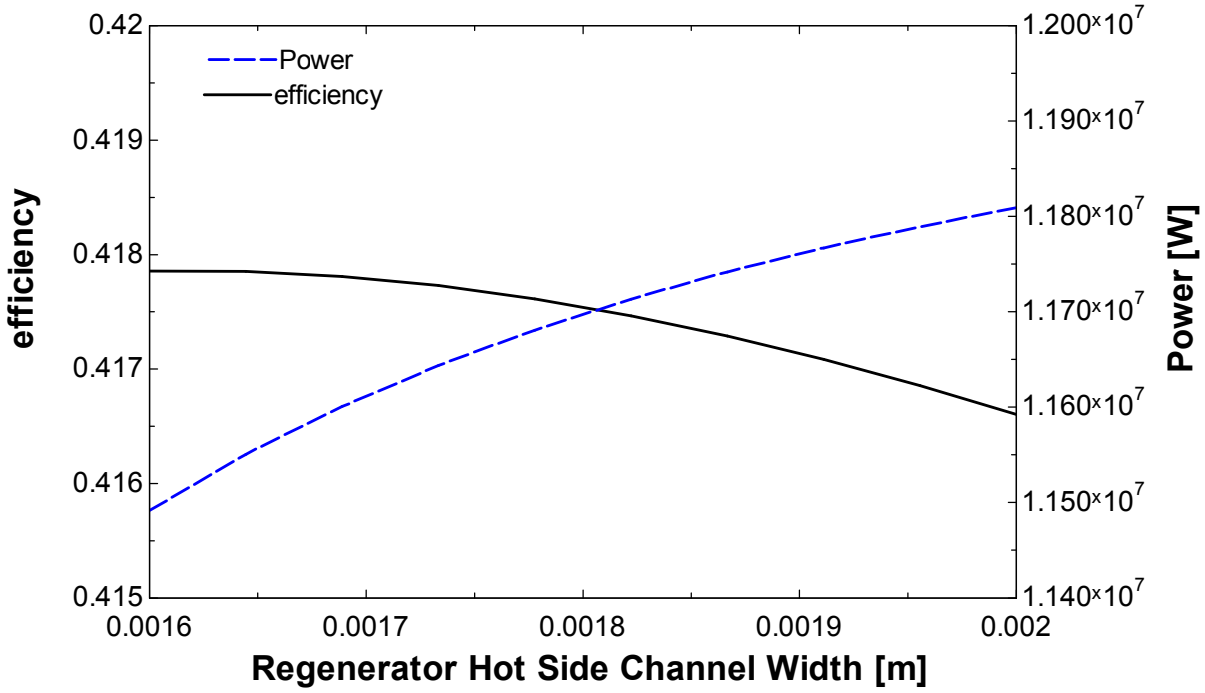

Figure 5-11: Cycle Performance v. Regenerator Hot Side Channel Size

Note that optimization for power requires the use of large channels which would benefit from decreased pressure drops while greatly decreasing regenerator effectiveness. Optimization for efficiency, on the other hand, entails the use of smaller channels that enhance regenerator effectiveness at the cost of greater pressure drops and reduced CO_2 mass flow rate. These impacts must be taken into account when choosing a value for this parameter.

Regenerator Cold Side Channel Width/Depth

Selection of the regenerator's cold side channel size requires that enhanced regenerator effectiveness be balanced against increased pressure drop. Smaller channels will provide enhanced heat transfer and therefore improved regeneration but also increased pressure drops and therefore reduced CO_2 density at the compressor inlet. Reduced compressor inlet density increases compressor work, thereby degrading power and efficiency. The relationship between channel size and performance is shown in Figure 5-12.

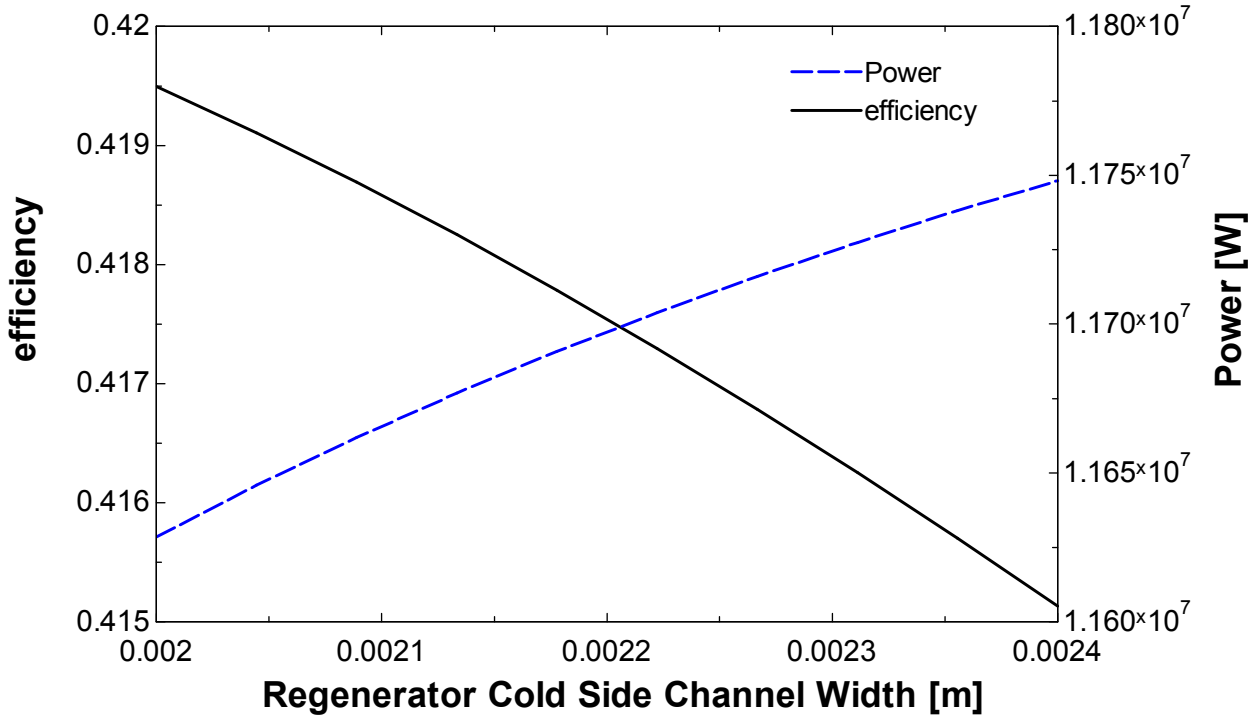

Figure 5-12: Cycle Performance v. Regenerator Cold Side Channel Size

Designing for efficiency requires use of very small channels, due to increased heat transfer coefficients in the regenerator. Designing for peak power, on the other hand, would require the use of larger channels with degraded heat transfer and regenerator effectiveness. As with the hot side channel size of the regenerator, power continues to increase with larger sizes as efficiency falls because the larger channels results in reduced pressure drops which increases turbine power. Heat transfer in the regenerator, on the other hand, is reduced, which results in degraded regenerator effectiveness and cycle efficiency. This relationship is shown in Figure 5-13.

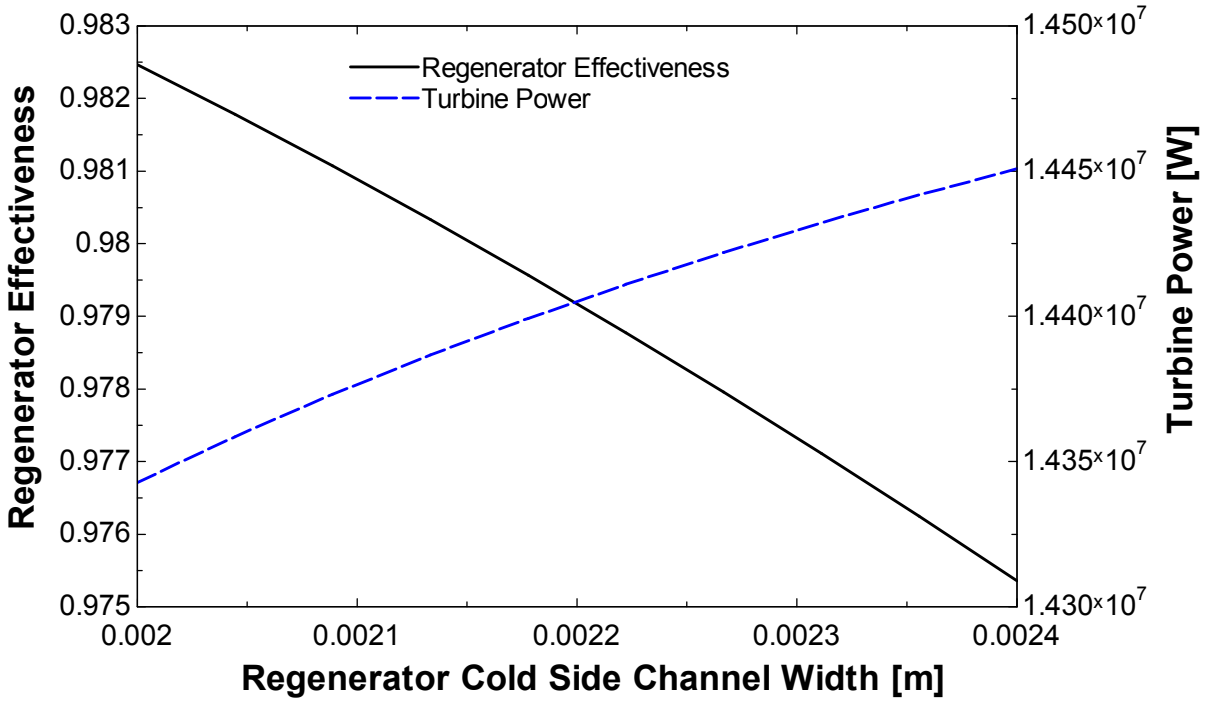

Figure 5-13: Regenerator Effectiveness and Turbine Power v. Regenerator Cold Side Channel Width

The response of regenerator effectiveness, and turbine power to changing the size of the regenerator cold side channels mirror those of the cycle's power and efficiency shown in Figure 5-12.

Precooler CO_2 Side Channel Width/Depth

Selection of heat exchanger channel size on the CO_2 side of the precooler requires balancing the impact of pressure drops on working fluid density at the compressor inlet versus the impact of improved or degraded heat transfer in the precooler. Figure 5-14 shows the performance impact of varying this parameter.

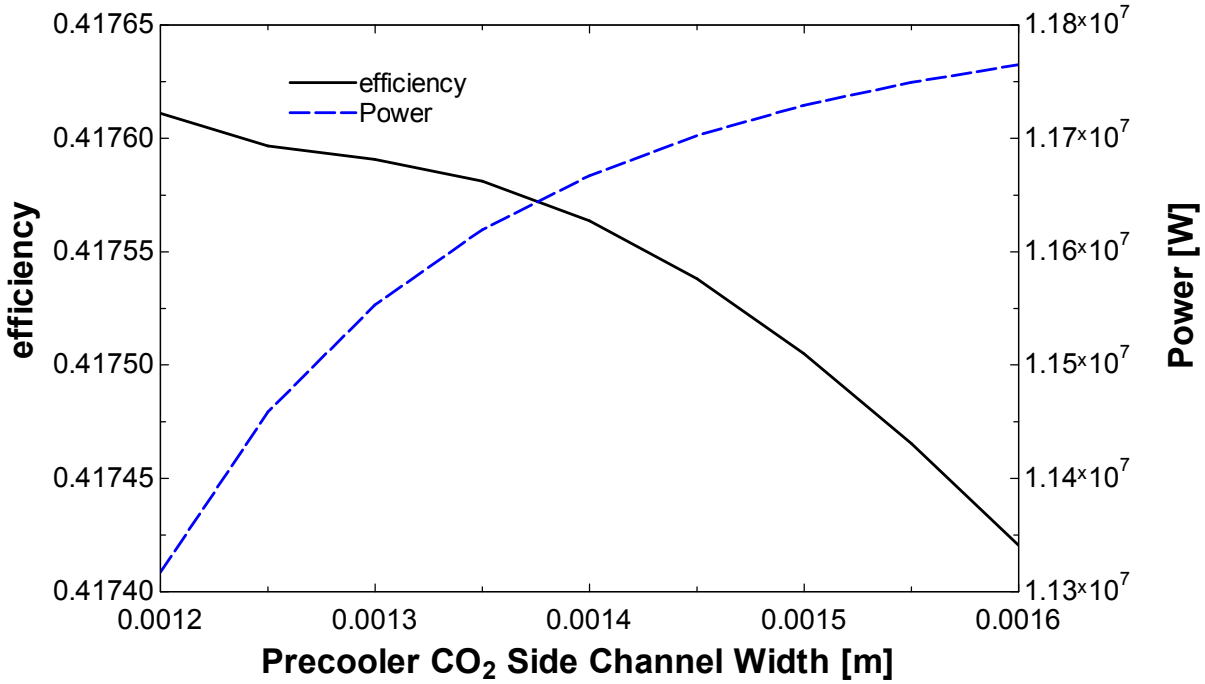

Figure 5-14: Cycle Performance v. Precooler CO_2 Side Channel Size

Increasing channel size reduces pressure drops – and therefore improves working fluid density at the compressor inlet. This increases the working fluid mass flow rate – increasing power. It should be noted, however, that the parasitic power use of the cooling tower is not accounted for in the Brayton module itself, apart from the module demanding lower water temperatures when the effectiveness of the precooler is degraded. This relationship is shown in Figure 5-15. Note that the lack of an apparent efficiency or power maximum is due to the cooling tower being modeled externally to EES. If the effectiveness of the regenerator is degraded – the EES model can simply demand colder cooling water. Power increases asymptotically as channel width increases due to decreased pressure drops in the CO_2 increasing the density and mass flow rate of CO_2 at the compressor inlet. The slight increase in efficiency at smaller channel sizes is due only to the decrease in CO_2 mass flow rate increasing the relative size of the cycle's heat exchangers.

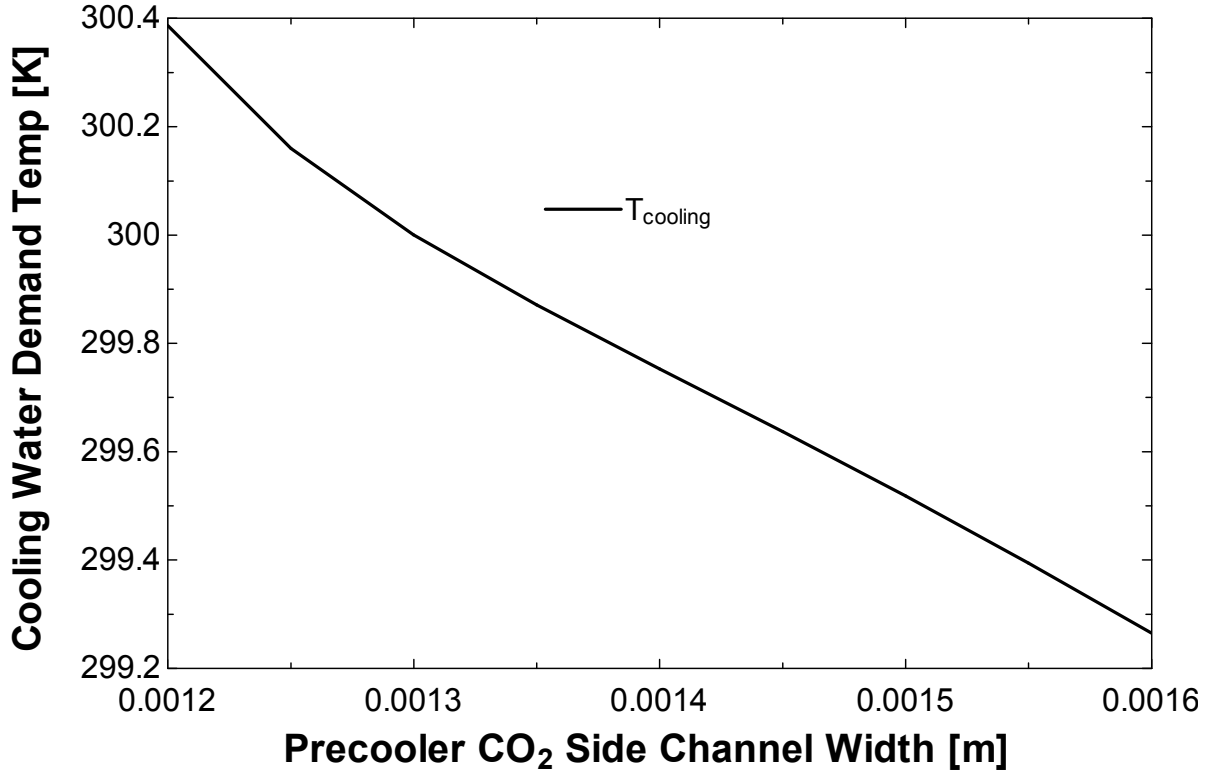

Figure 5-15: Cooling Water Demand Temperature v. Precooler CO_2 Side Channel Size

As the precooler channel sizes increase the fluid velocity and heat transfer of CO_2 in the precooler both decrease. To meet the design CO_2 temperature at the compressor inlet, therefore, the inlet temperature of cooling water must be reduced. Any design of the channel size on the precooler CO_2 side, therefore, should restrict itself to values that do not require unreasonably low cooling water temperatures.

Precooler Water Side Channel Width/Depth

Selection of the water side channel size in the precooler is primarily a function of balancing water pumping power against improved heat transfer (and accompanying increased cooling water demand temperatures). Figure 5-16 shows the relationship between channel size and cycle performance.

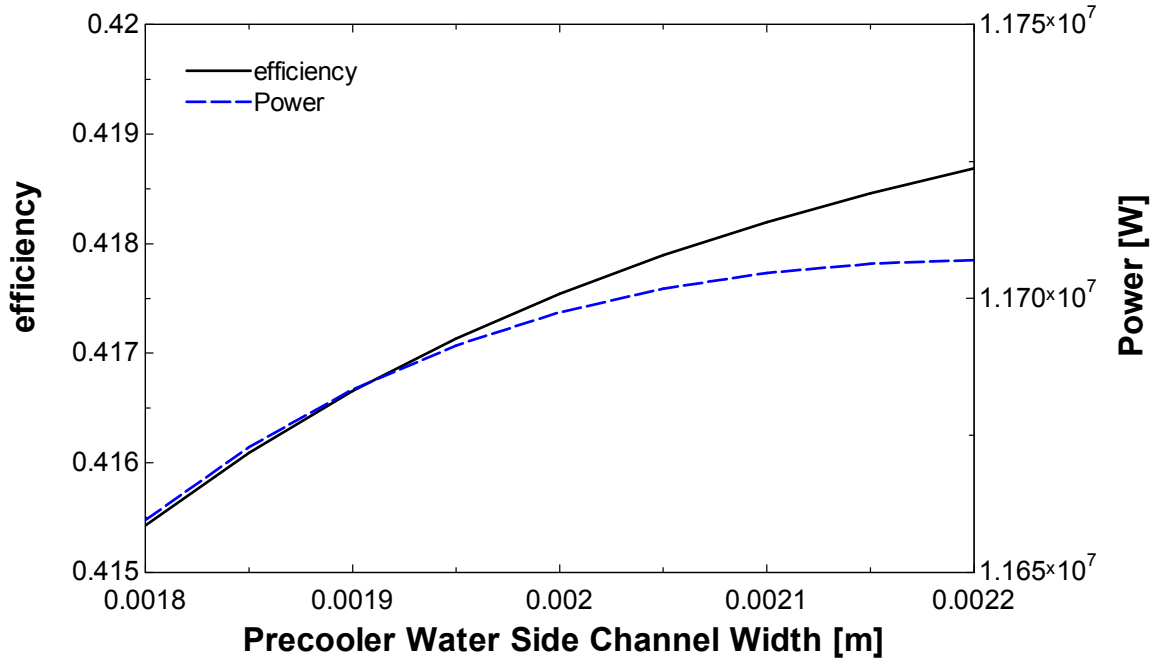

Figure 5-16: Cycle Performance v. Precooler Water Side Channel Size

Efficiency continues to improve as the water side precooler channel width increases because of reduced pumping power requirements, and the fact that with the cooling tower modeled externally to EES the Brayton cycle simply demands colder cooling water as the effectiveness of the precooler decreases.

Power, on the other hand, eventually decreases. This is due to the decreased effectiveness of the heat exchanger slowing the rate of cooling in the CO_2; this in turn increases pressure drops on the CO_2 side of the precooler. Increased pressure drops coincide with decreased compressor inlet densities for the CO_2. This relationship is shown in Figure 5-17.

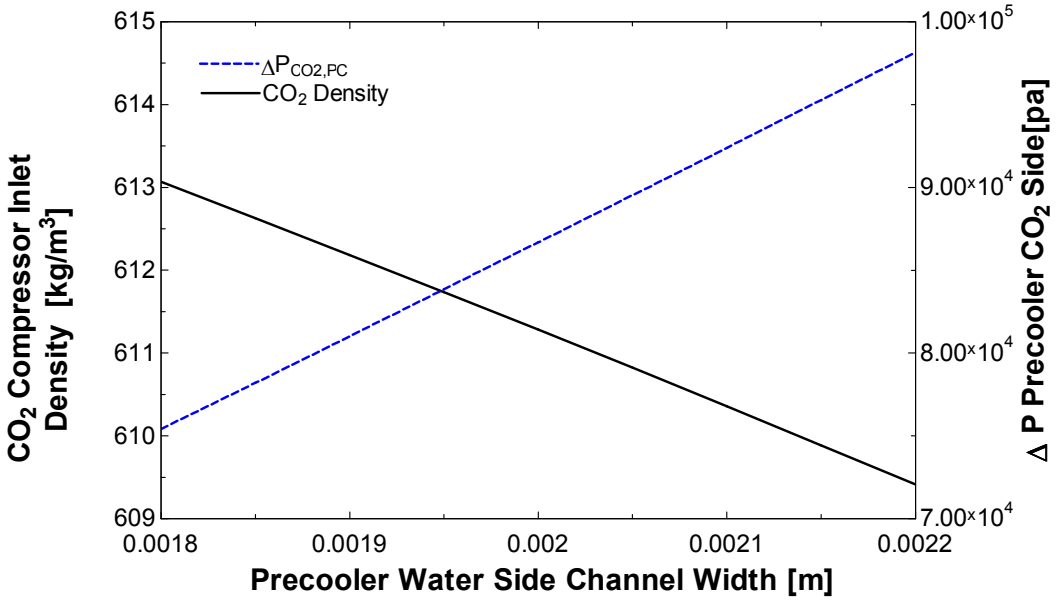

Figure 5-17: Precooler CO_2 Side Pressure Drops and Outlet Densities v. Precooler Water Side Channel Size

Another impact of increasing the water-side channel size is that decreased effectiveness in the precooler necessitates lower temperature cooling water, as shown in Figure 5-18.

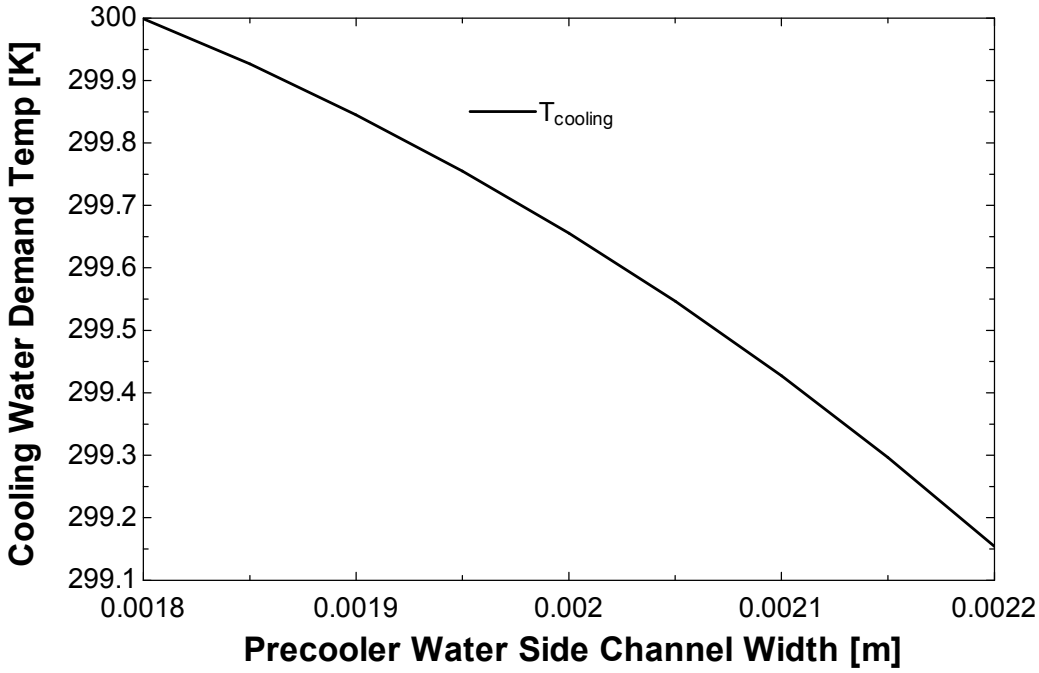

Figure 5-18: Cooling Water Demand Temperature v. Precooler Water Side Channel Size

Since cooling tower energy use is not accounted for in this model, selection must ensure that the

required cooling water inlet temperature is not pushed too low.

Heat Exchanger Area Distribution

Optimum heat exchanger area distribution requires a balance be struck between the effectiveness of each of the heat exchangers. Figure 5-19 shows the relationship between cycle performance and the fraction of total heat exchanger area allocated to the regenerator.

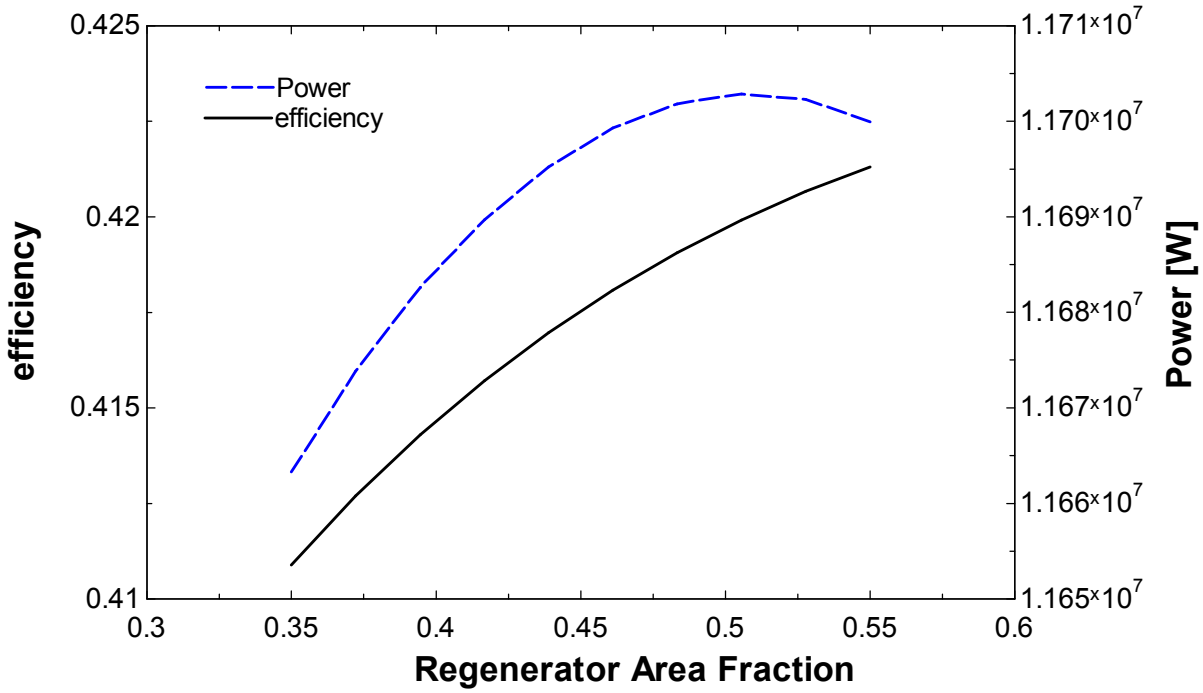

Figure 5-19: Cycle Performance vs. Regenerator Area Fraction

Increased regenerator size drives improved performance, though it must be noted that selection of this parameter must be restricted to ranges that allow for a reasonable cooling water inlet temperatures, since cooling tower use is external to this analysis. The relationship between this temperature and regenerator area fraction is shown in Figure 5-20. Note that a regenerator is necessary for peak power production, because without one the cycle simply rejects a great deal of heat into the precooler and is unable to entirely make up for this by taking more heat from the molten salt.

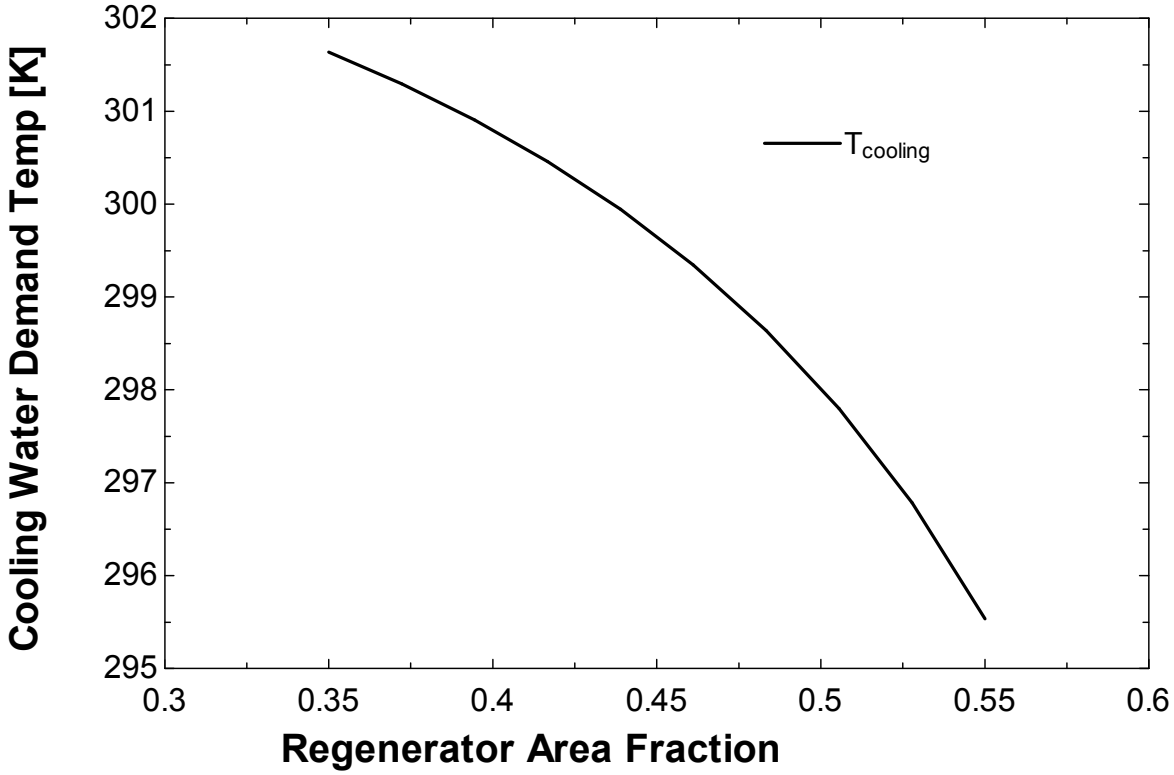

Figure 5-20: Cooling Water Demand Temperature vs. Regenerator Area Fraction

Colder cooling water is required to meet the design compressor inlet CO_2 temperature as precooler area is reduced to accommodate a larger regenerator.

Primary Heat Exchanger Aspect Ratio

Selecting the aspect ratio of the primary heat exchanger requires the balancing the impact of pressure drops against enhanced heat transfer. The relationship between aspect ratio and cycle performance is shown in Figure 5-21 below.

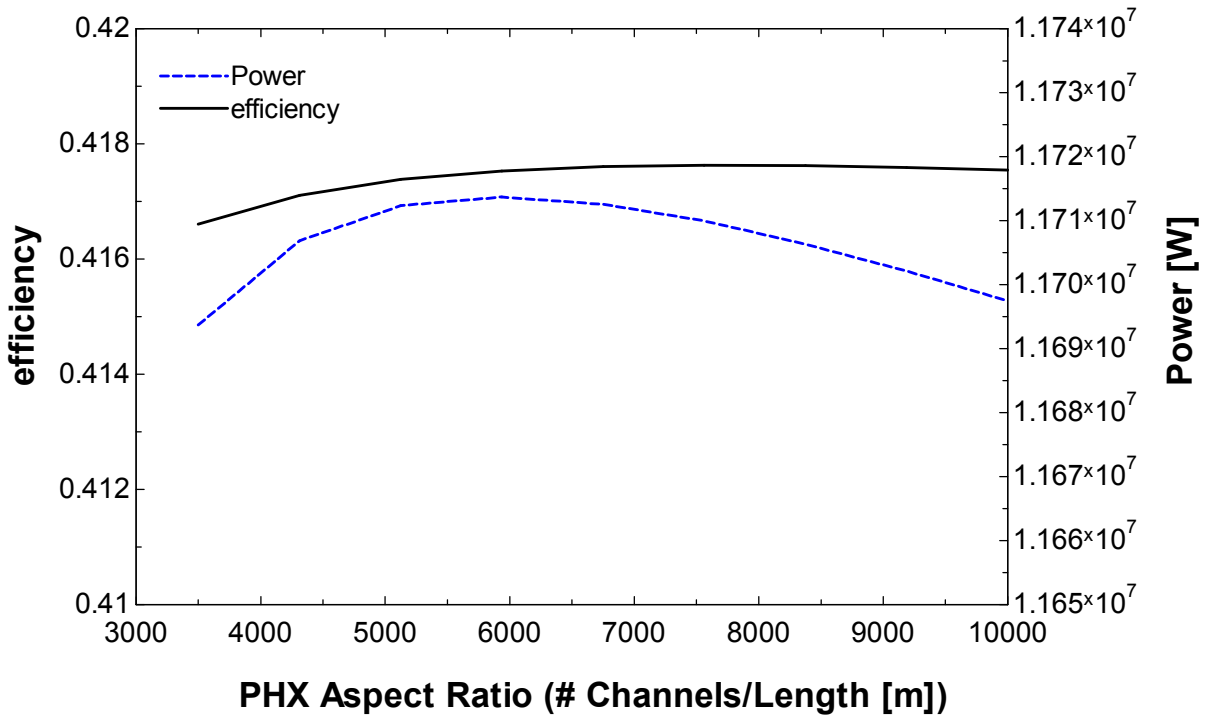

Figure 5-21: Cycle Performance vs. PHX Aspect Ratio

Designing for efficiency requires the use of high aspect ratios (corresponding to a very short and wide heat exchanger). Peak power is reached at a somewhat lower aspect ratio, corresponding to a longer and thinner heat exchanger.

Regenerator Aspect Ratio

Selecting the aspect ratio (number of channels in parallel per meter of length) of the regenerator is a matter of balancing the enhanced heat transfer created by fewer, longer channels against the greater pressure drops this entails. Figure 5-22 shows the performance impact of varying this parameter.

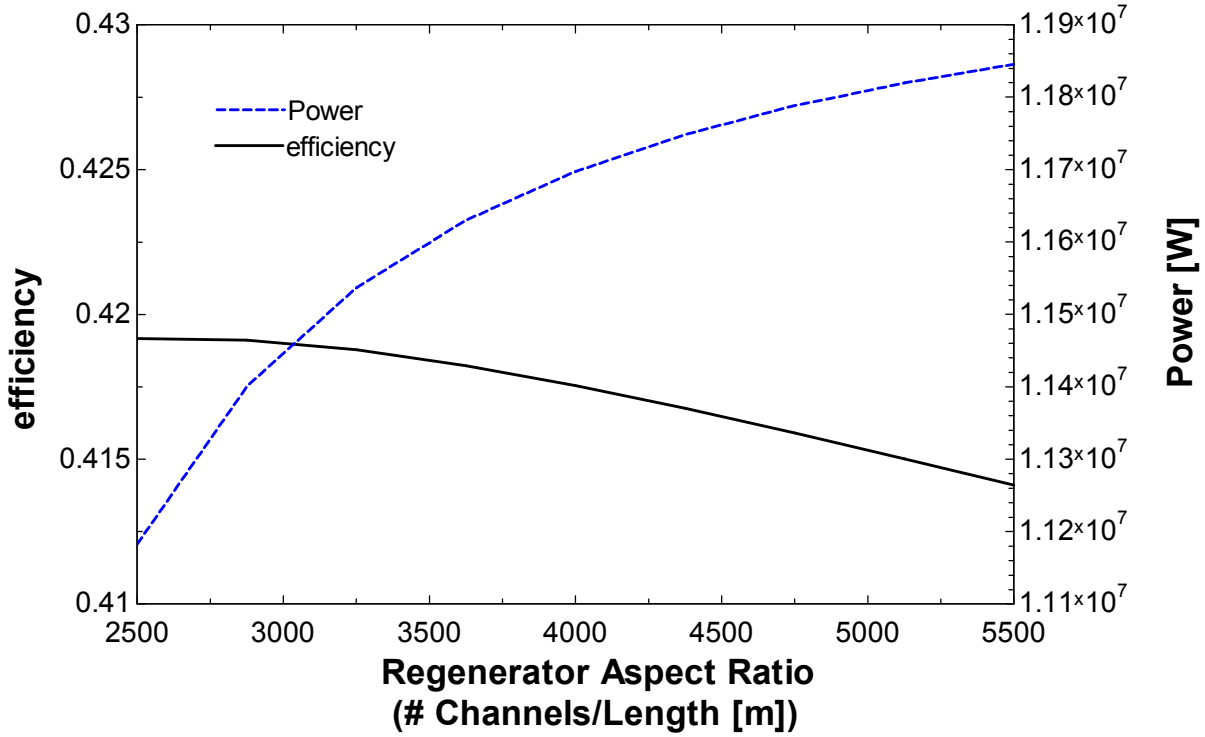

Figure 5-22: Cycle Performance v. Regenerator Aspect Ratio

Peak efficiency is reached at an aspect ratio that balances pressure drop against regenerator effectiveness. Note that power continues to increase with very short and wide heat exchangers, due to the decreasing pressure drop in the regenerator corresponding to a greater compressor inlet pressure and therefore increased density. This relationship is shown in Figure 5-23.

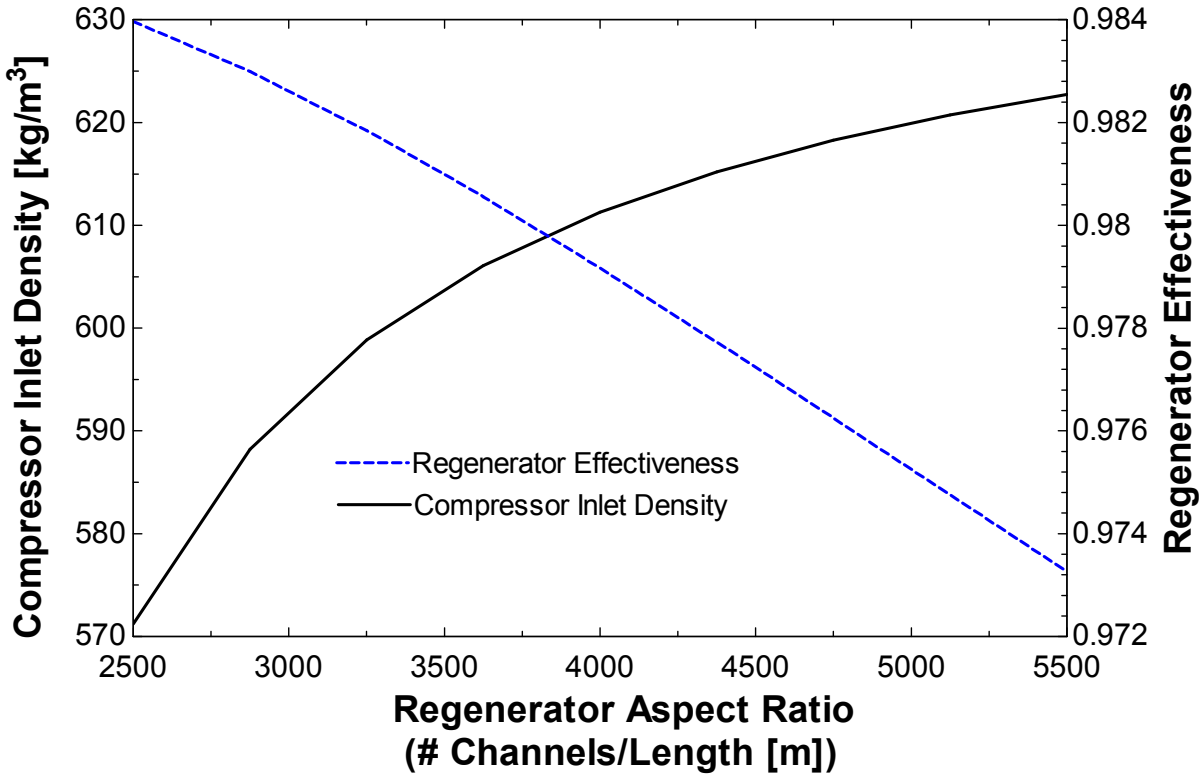

Figure 5-23: Compressor Inlet Density and Regenerator Effectiveness v. Regenerator Aspect Ratio

Note that the density of CO_2 at the compressor inlet continues to increase at higher and higher regenerator aspect ratios. This increased density drives increased power at very high aspect ratios – despite degrading regenerator effectiveness. The regenerator aspect ratio impacts the required cooling water inlet temperature. This relationship is shown in Figure 5-24.

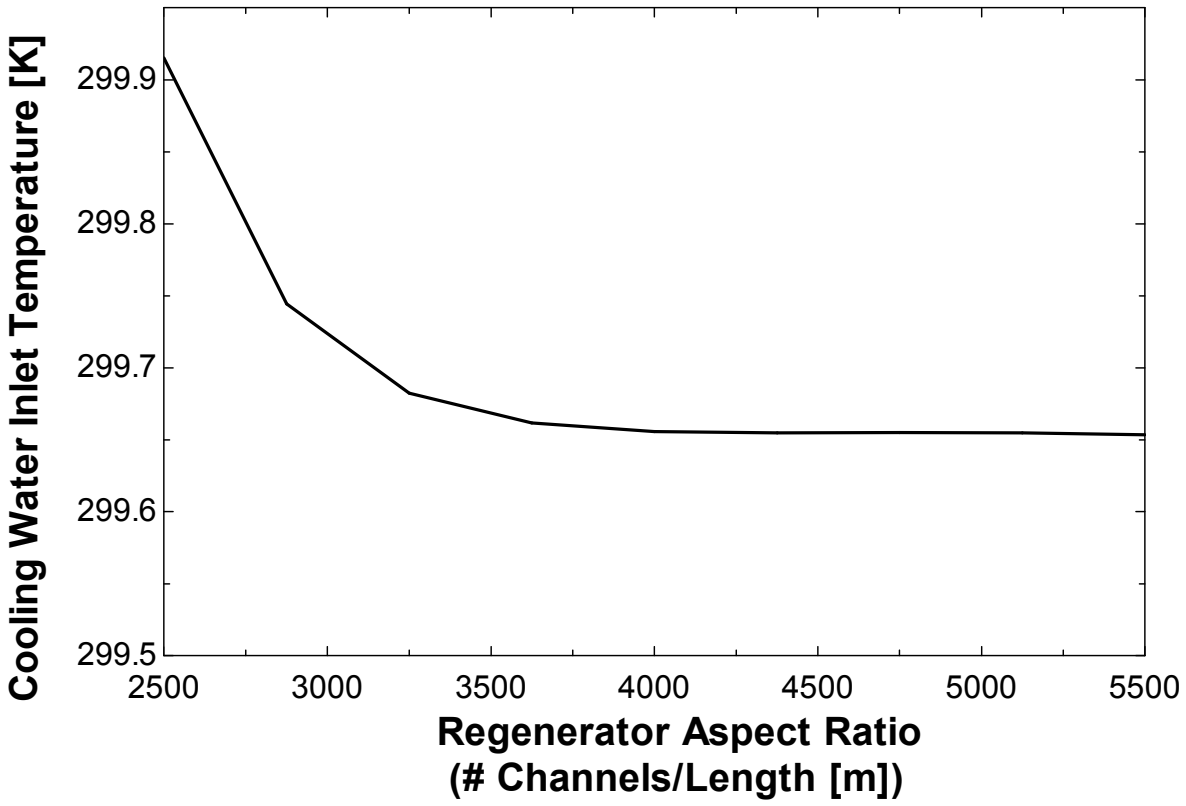

Figure 5-24: Cooling Water Demand Temperature v. Regenerator Aspect Ratio

The required cooling temperature increases sharply at low aspect ratios. This is due to increased flow velocity in the regenerator enhancing heat transfer on both its hot and cold side. This in turn improves regenerator effectiveness leaving less heat transfer to be accomplished by the precooler to meet the design CO_2 compressor inlet temperature. These relationships are shown in Figure 5-25.

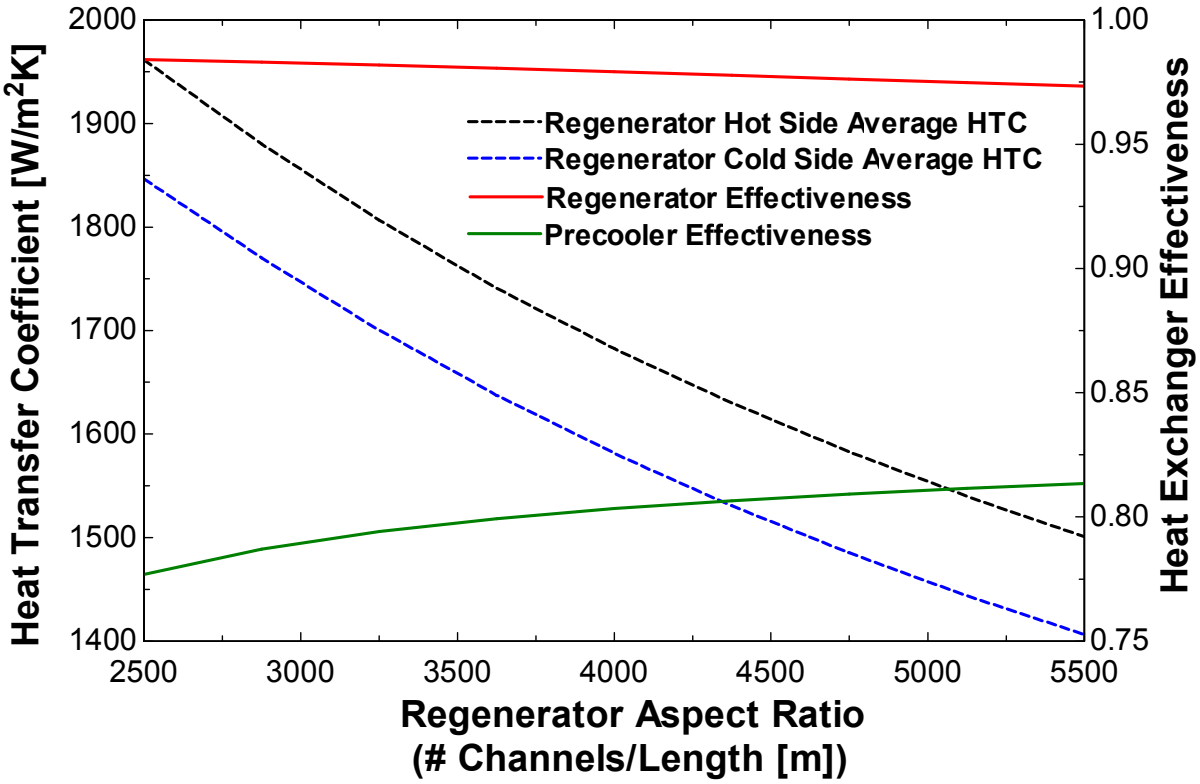

Figure 5-25: Regenerator Heat Transfer Coefficients / Precooler and Regenerator Effectiveness v. Regenerator Aspect Ratio

Improved regenerator performance decreases the cooling load that must be met by the precooler. With a reduced precooler load the cycle can demand warmer cooling water as shown in Figure 5-24.

Precooler Heat Exchanger Aspect Ratio

As with other aspect ratio selections pressure drops must be balanced against decreased heat transfer in the heat exchanger. The relationship between precooler aspect ratio and cycle performance is shown in Figure 5-26.

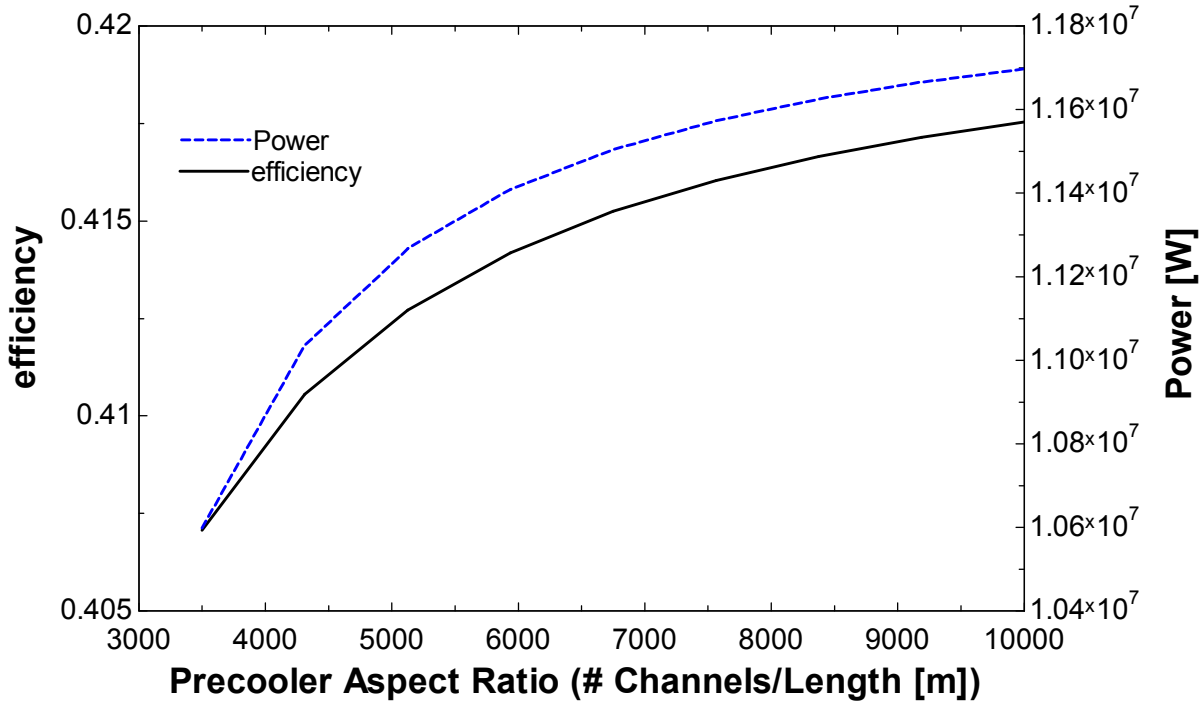

Figure 5-26: Cycle Performance v. Precooler Aspect Ratio

Extremely large aspect ratios give peak performance both in efficiency and power. This is, however, largely due to the fact that decreased heat exchanger effectiveness is being made up for by simply demanding lower temperature cooling water. This relationship is shown in Figure 5-27.

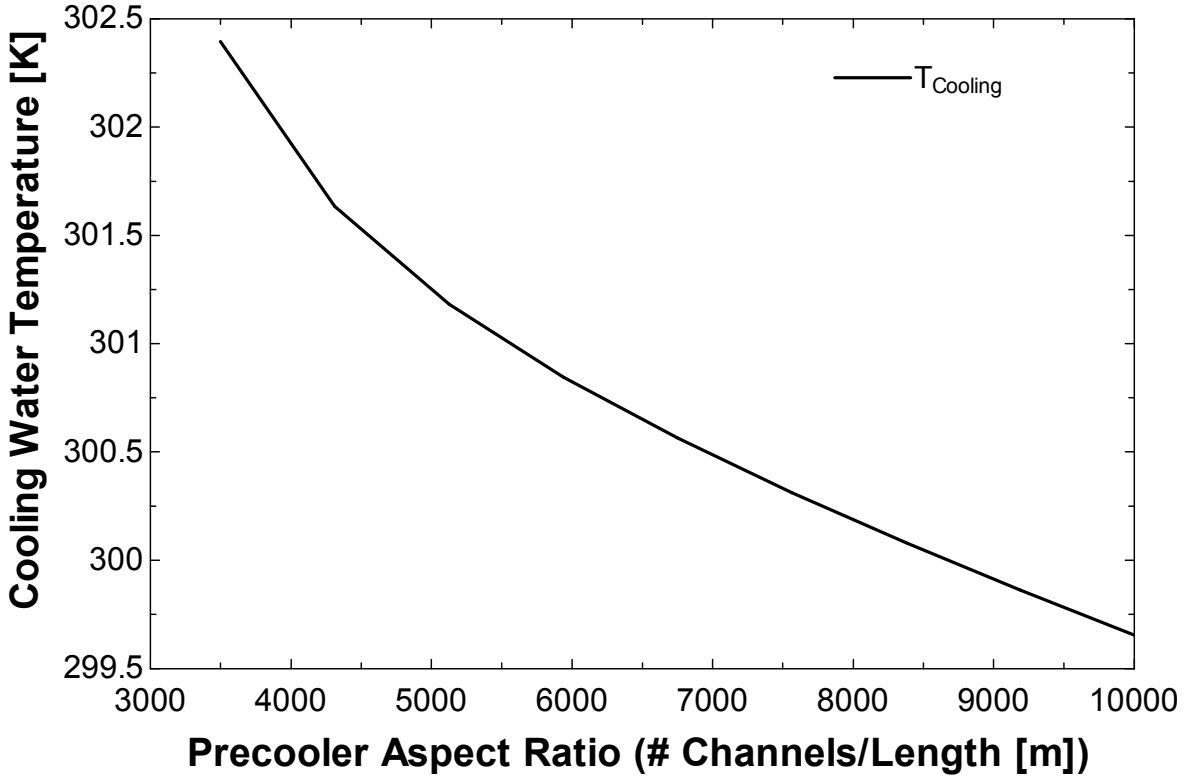

Figure 5-27: Cooling Water Demand Temperature v. Precooler Aspect Ratio

Since cooling tower calculations are external to this analysis, we will constrain our selection of aspect ratios to a range that does not require unreasonably low cooling water temperatures.

Now that the impact of each parameter on cycle performance has been examined, it is clear that designing a cycle is a matter of balancing the impact of each parameter on net power, net efficiency, and cooling water demand temperature. A well-designed plant will provide the required net power at a high efficiency, and without demanding cooling water at an unreasonably low temperature. The next section of this chapter addresses the impact of varying design parameters in the air-cooled Brayton cycle.

5.3.2 Air-Cooled Brayton

Pressure Ratio

The compressor inlet pressure of the air-cooled Brayton cycle is a very important parameter since it dictates the temperature at which specific heat peaks, and therefore dictates the highest

ambient temperature at which heat rejection can reasonably be achieved via dry cooling, while still obtaining favorable compressor inlet densities. Figure 5-28 shows the relationship between temperature and density at a variety of pressures.

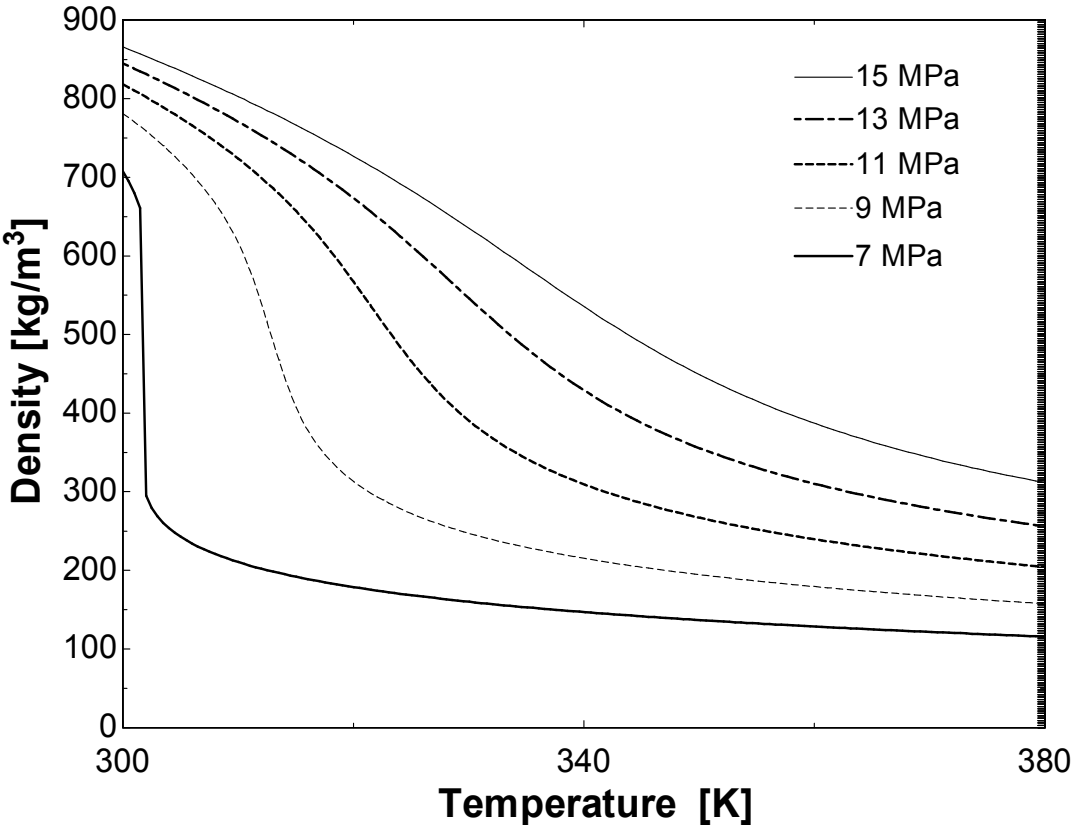

Figure 5-28: CO_2 Density v. Temperature

For higher pressures the temperatures required to reach attractive densities are higher, allowing for easier heat rejection to relatively high ambient temperatures. Figure 5-29 shows the relationship between specific heat and temperature at a variety of pressures.

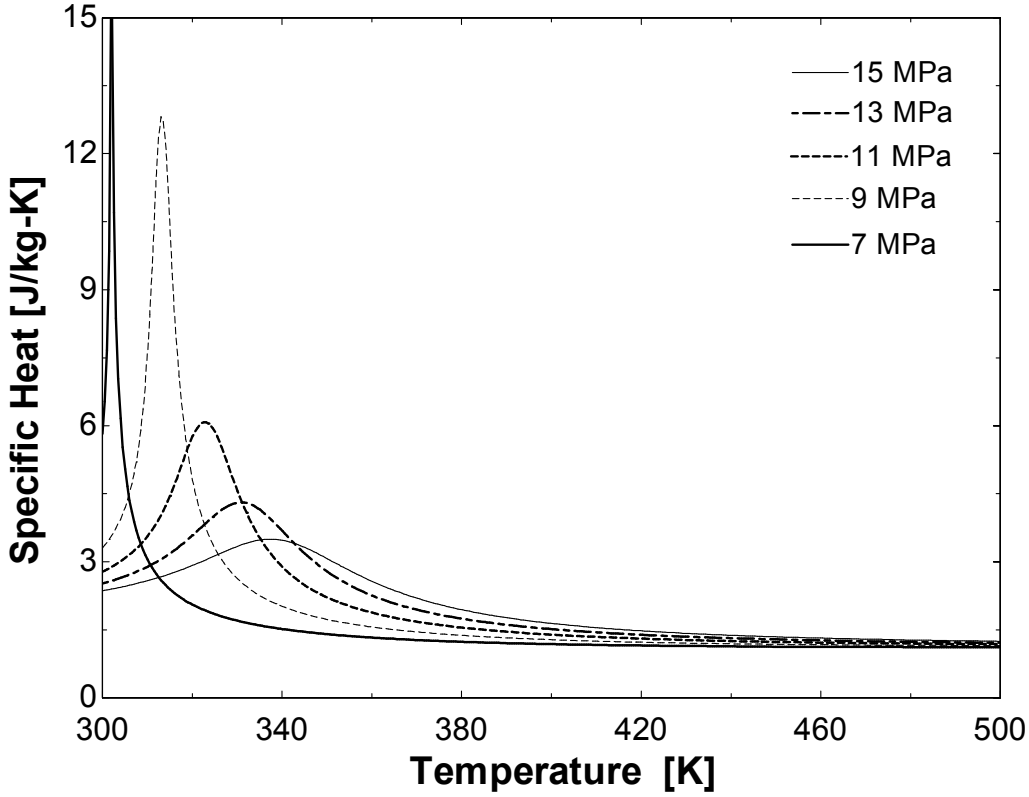

Figure 5-29: CO_2 Specific Heat v. Temperature

At 7 MPa, for example, the specific heat reaches a sharp peak at ~305 K. Rejecting heat to ambient temperatures above 305K is not possible and much lower ambient temperatures would be required in practice. At 11 MPa, on the other hand, specific heat peaks at ~325 K. This behavior could allow dry heat rejection at ambient conditions reaching as high as 320 K.

When selecting a pressure ratio a balance must be struck between providing high CO_2 densities at the compressor inlet while still allowing heat rejection to occur at relatively high temperatures. Figure 5-30 shows the relationship between pressure ratio and net efficiency at a variety of ambient temperatures, with the compressor outlet pressure fixed at 25 MPa.

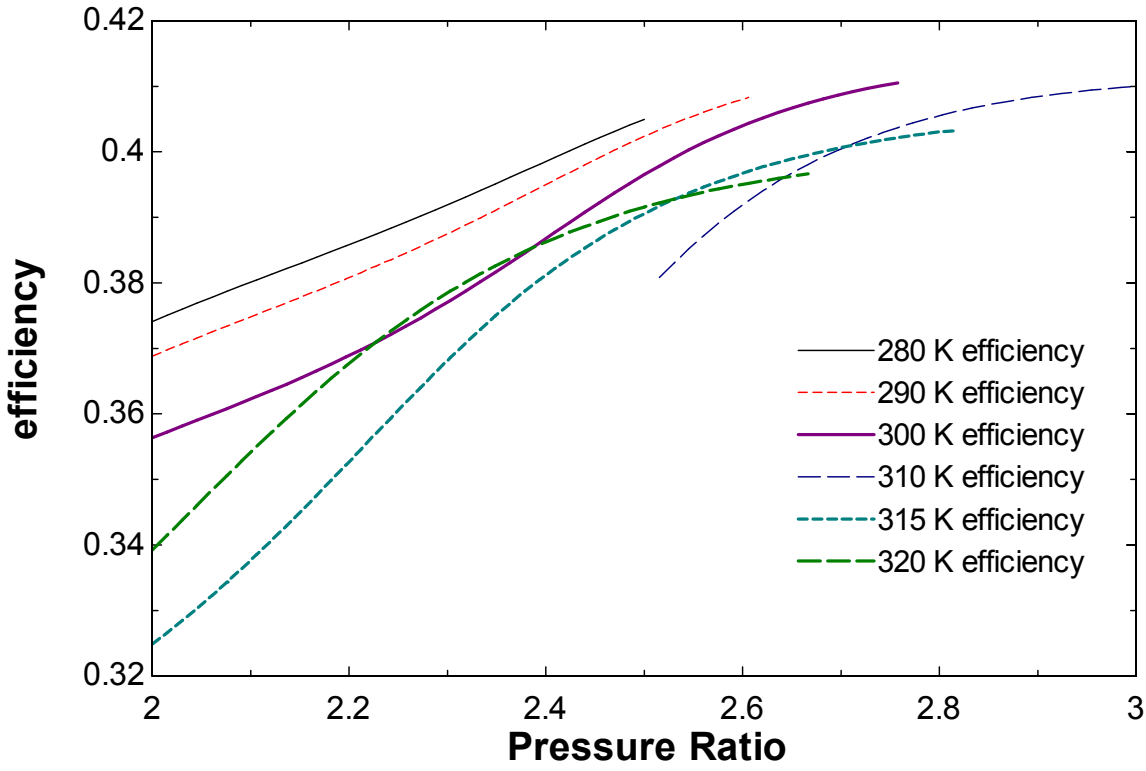

Figure 5-30: Net Efficiency variation as a function of Pressure Ratio with ambient dry bulb temperature as a parameter.

Note, however, that the enhanced efficiency we see at higher pressure ratios are due to decreased pressure at the compressor inlet reducing the mass flow rate of CO_2 through the cycle. This in turn increases the relative size of all the heat exchangers, improving the efficiency of the cycle while greatly degrading power.

Figure 5-31 shows the relationship between power and pressure ratio at the same range of temperatures. Net power is a more meaningful metric for comparison because it reflects the impact of reduced working fluid mass flow rate at higher pressure ratios.

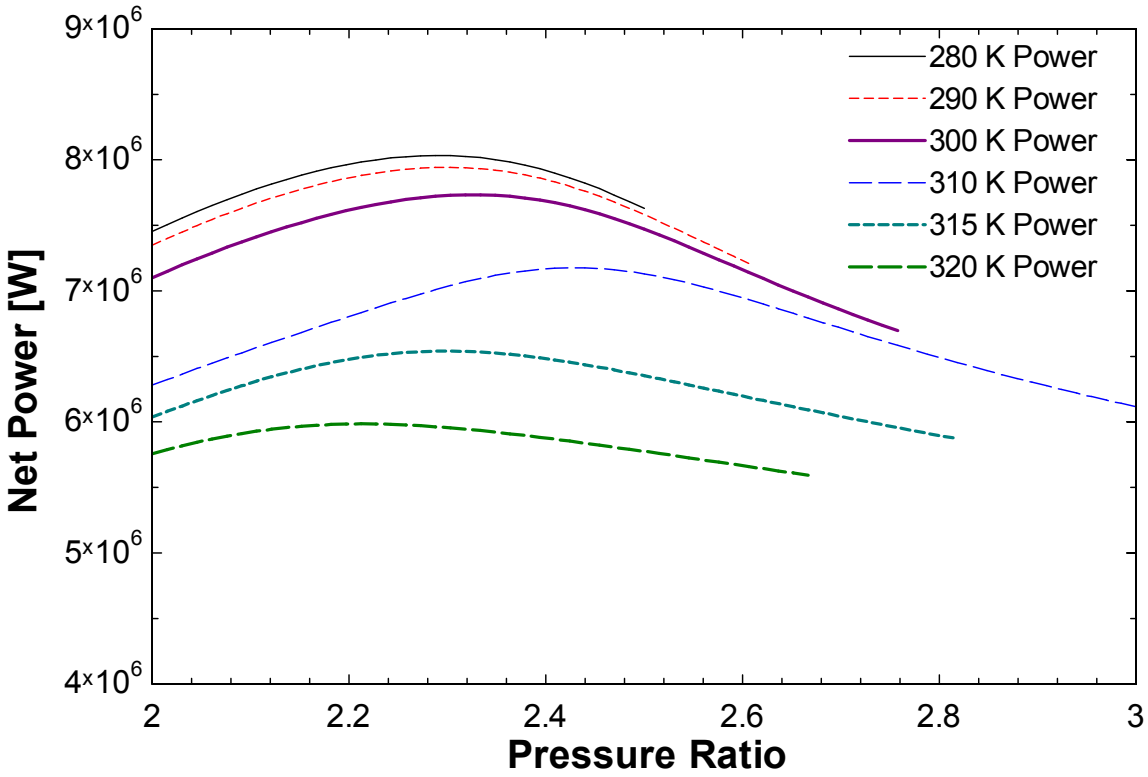

Figure 5-31: Net Power v. Pressure Ratio

A pressure ratio can be selected by choosing a representative ambient temperature and then striking a balance between power and efficiency.

Primary Heat Exchanger Salt Side Channel Width/Depth

Selection of the salt-side channel size in the primary heat exchanger is a question of balancing heat transfer effectiveness against parasitic salt pumping power. Figure 5-32 shows cycle performance as a function of this parameter.

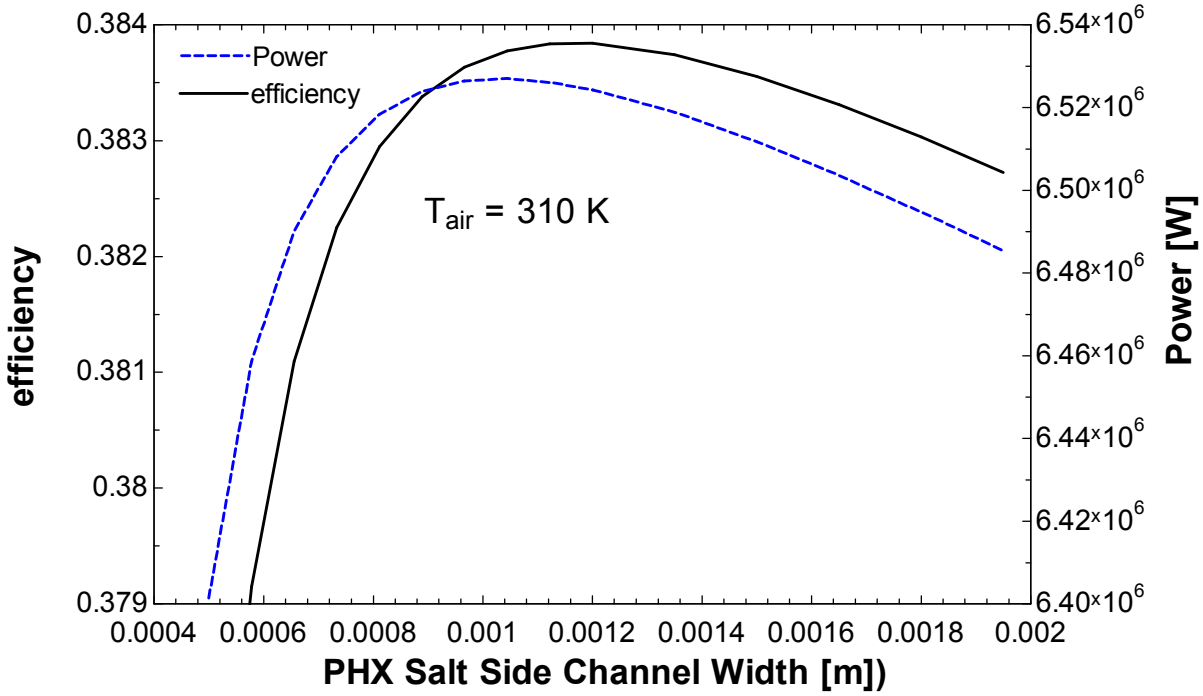

Figure 5-32: Cycle Performance v. PHX Salt Side Channel Size

Smaller channels provide enhanced heat transfer – but also greater pumping power requirements. Increased pumping power degrades net power and efficiency. This behavior is shown in Figure 5-33.

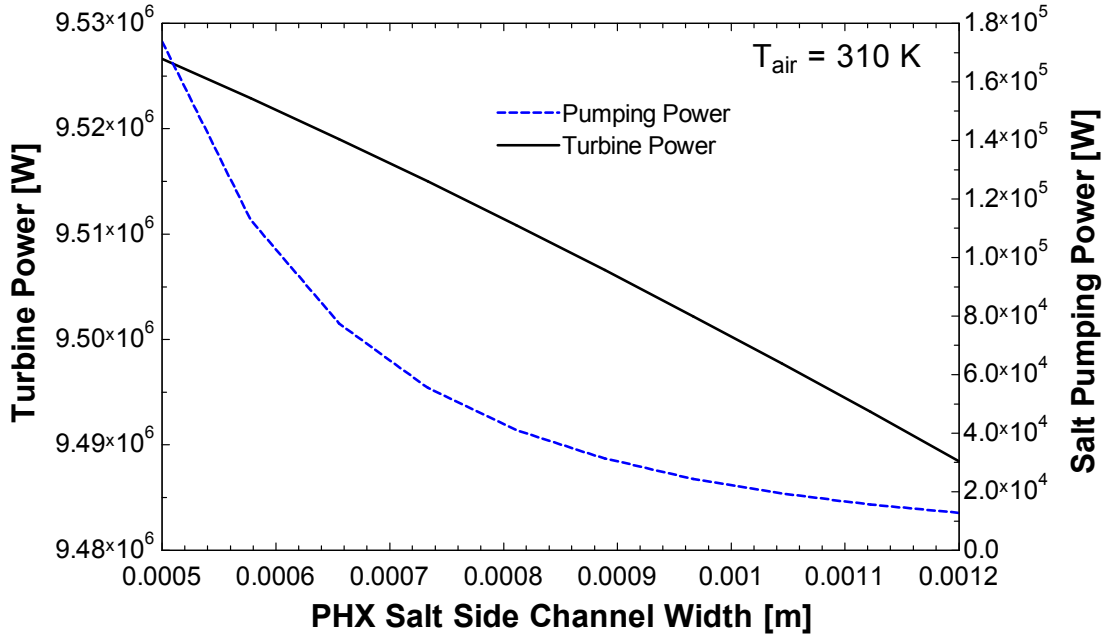

Figure 5-33: Salt Pumping Power and Turbine Power v. PHX Salt Side Channel Size

Smaller channels increase salt pumping power due to greater salt pressure drops, and also increase turbine power by increasing heat transfer and CO_2 temperatures at the turbine inlet.

Primary Heat Exchanger CO_2 Side Channel Width/Depth

The CO_2 side channels of the Primary Heat Exchanger must balance pressure drops against heat transfer coefficient. The relationship between cycle performance and channel size is shown in Figure 5-34.

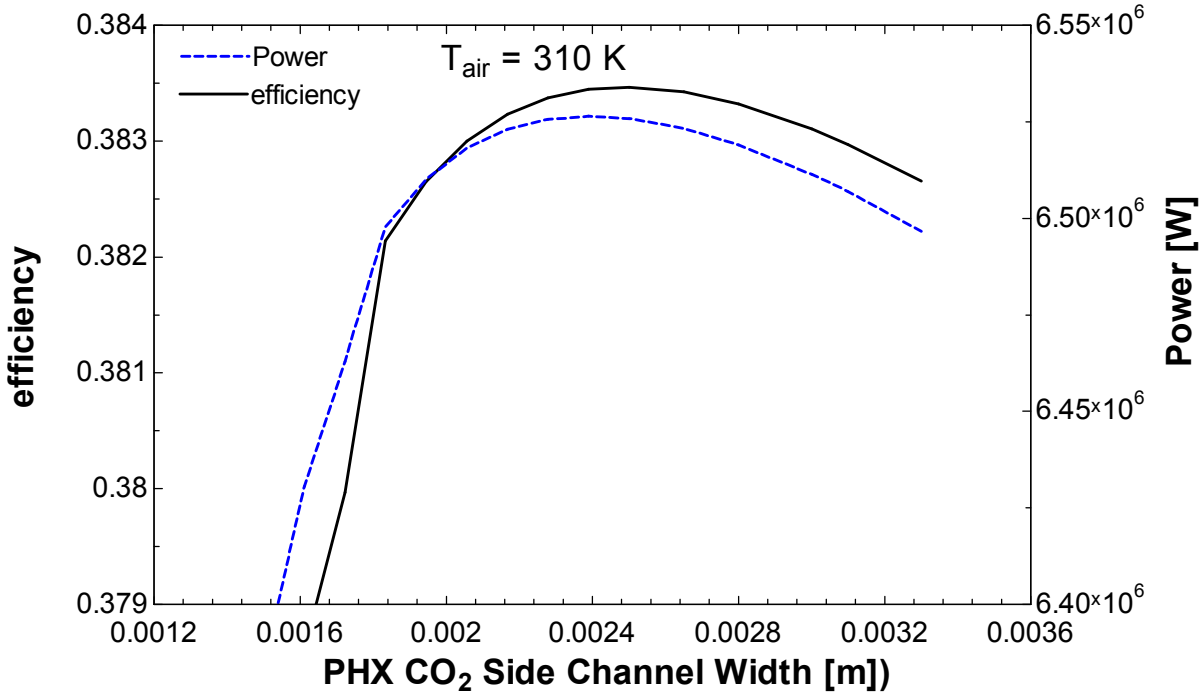

Figure 5-34: Cycle Performance v. PHX CO_2 Side Channel Size

Larger channels have degraded heat transfer performance but also reduced pressure drops – allowing for a greater pressure drop across the turbine. The channel size can be optimized for either peak power or efficiency.

Regenerator Hot Side Channel Width/Depth

Selection of the hot side channel size in the regenerator requires balancing pressure drops and their effect on compressor inlet density against the effectiveness of the regenerator. Figure 5-35

shows cycle performance as a function of Regenerator hot side channel width.

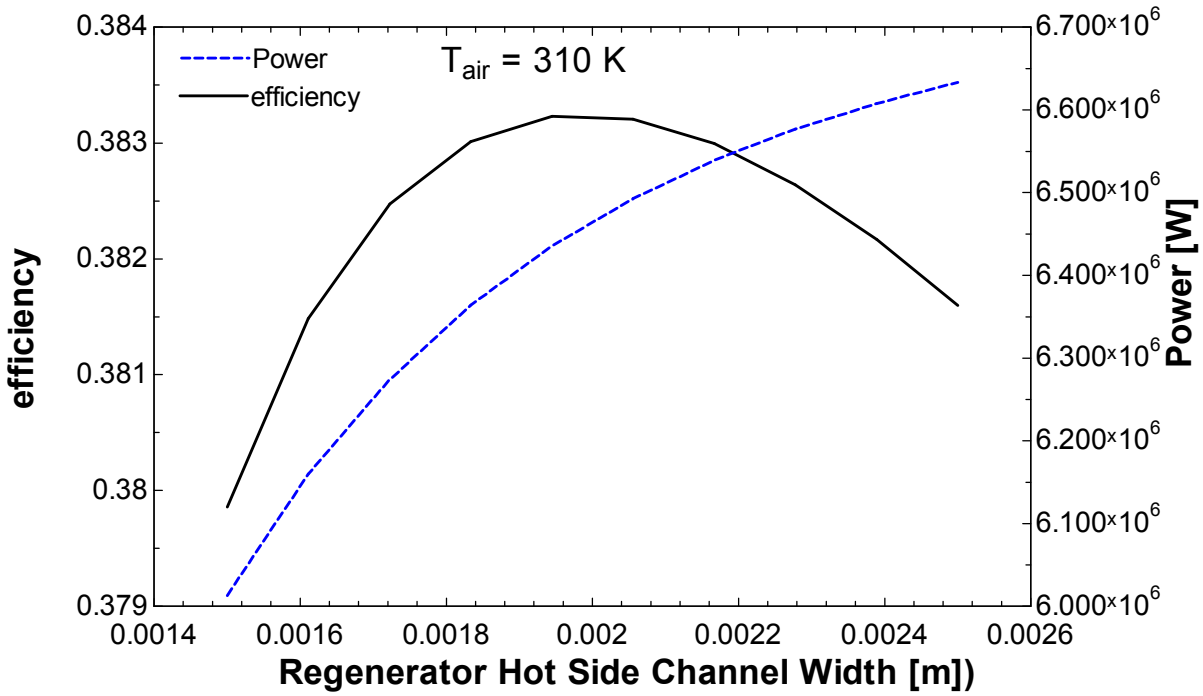

Figure 5-35: Cycle Performance v. Regenerator Hot Side Channel Size

Note that while efficiency peaks near a 2mm channel width, power continues to increase at larger channel widths. This is because very large channels, while greatly degrading regenerator effectiveness, have very small pressure drops. These minimal pressure drops allow for higher compressor inlet densities – and therefore a greater CO_2 mass flow rate and greater power.

Regenerator Cold Side Channel Width/Depth

As with the regenerator's hot side channels, selection of the cold size channel parameters involves balancing the impact of enhanced heat transfer against increased pressure drops. The relationship between regenerator cold side channel size and performance is shown in Figure 5-36.

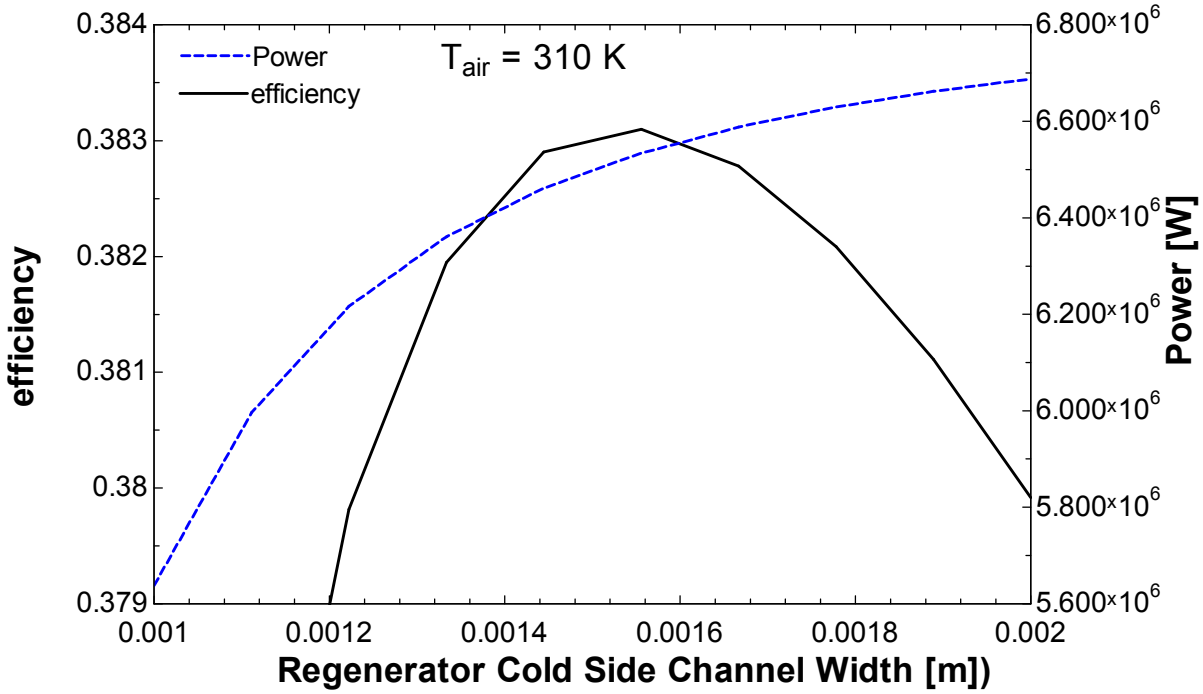

Figure 5-36: Cycle Performance v. Regenerator Cold Side Channel Size

Smaller channels increase regenerator effectiveness by enhancing heat transfer – but at the cost of increased pressure drops. These greater pressure drops decrease the pressure drop across the turbine – decreasing its output. Efficiency can be optimized by balancing these two impacts. Peak power, however, is only reached at an extremely large channel size with correspondingly degraded regenerator effectiveness and cycle efficiency.

Heat Exchanger Area Distribution

The air-cooled cycle only has two compact heat exchangers, the primary heat exchanger and the regenerator. Cycle performance can be tuned by varying the fraction of total heat exchanger area that is dedicated to each heat exchanger. The performance of the cycle as a function of the fraction of heat exchanger dedicated to the regenerator (with the remainder dedicated to the primary heat exchanger) is shown in Figure 5-37.

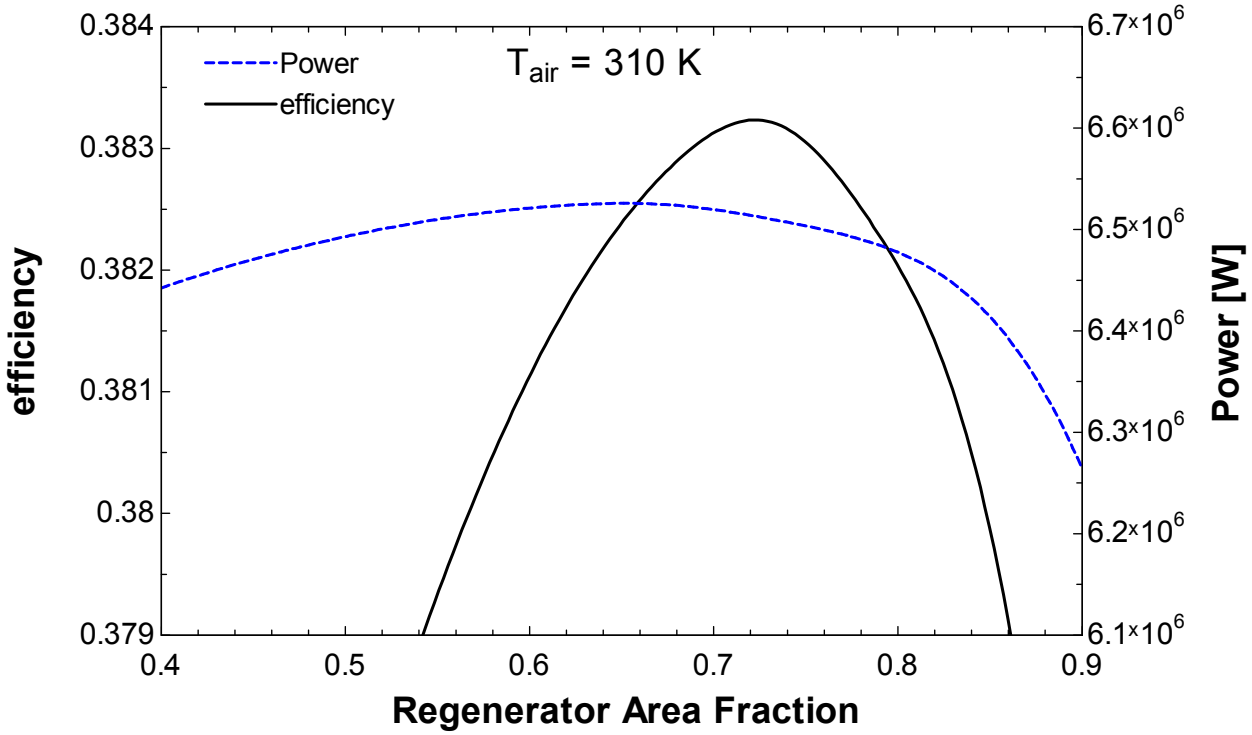

Figure 5-37: Cycle Performance v. Regenerator Fraction

Greater regeneration and reduced pressure drops through the regenerator's hot side drive improve performance at higher regenerator fractions. At very high regenerator fractions both efficiency and power are degraded, since heat exchanger area is moved from the primary heat exchanger to the regenerator. This reduces turbine inlet temperatures for the CO_2 and therefore turbine power output. This parameter can be optimized for either power or efficiency depending on design goals.

Primary Heat Exchanger Aspect Ratio

Selecting the aspect ratio of the primary heat exchanger requires heat transfer be balanced against pressure drops. Performance as a function of PHX aspect ratio is shown in Figure 5-38.

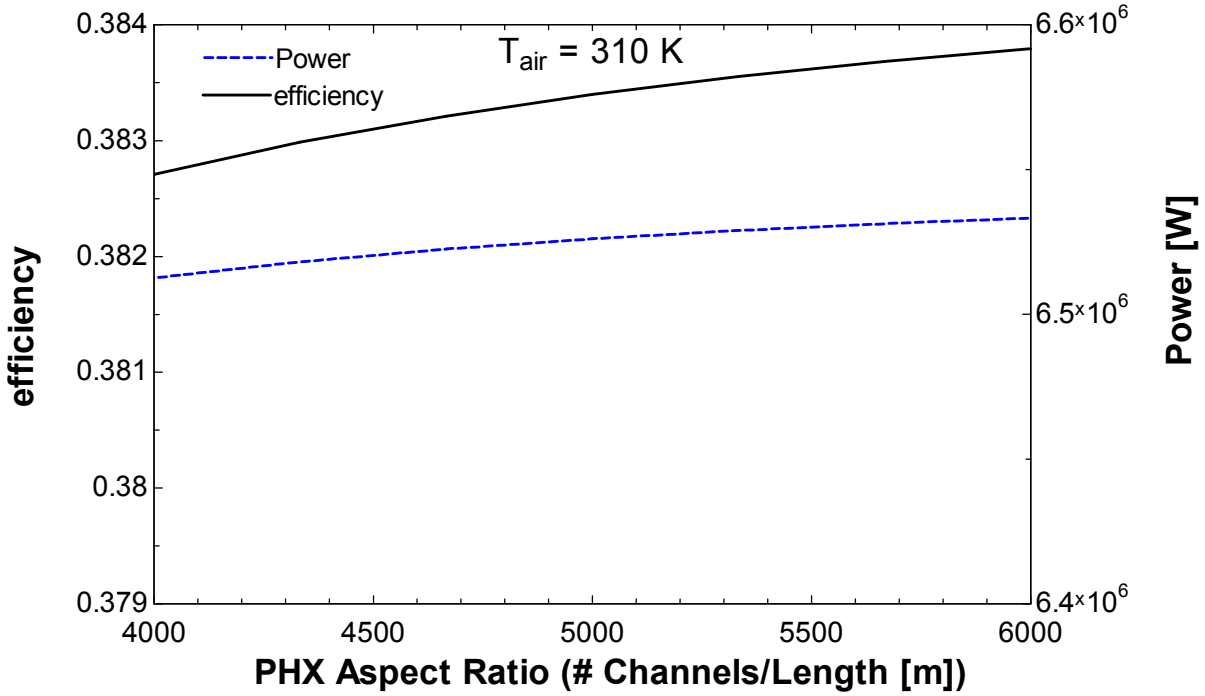

Figure 5-38: Cycle Performance v. PHX Aspect Ratio

Larger aspect ratios (corresponding to a shorter, wider heat exchanger) produce reduced heat transfer coefficients due to reduced CO_2 velocity in the primary heat exchanger. Large aspect ratios also result in lower pressure drops.

Regenerator Aspect Ratio

Selecting the aspect ratio of the regenerator (as with the primary heat exchanger) is a matter of balancing pressure drops against heat transfer effectiveness. Figure 5-39 shows the relationship between cycle performance and this parameter.

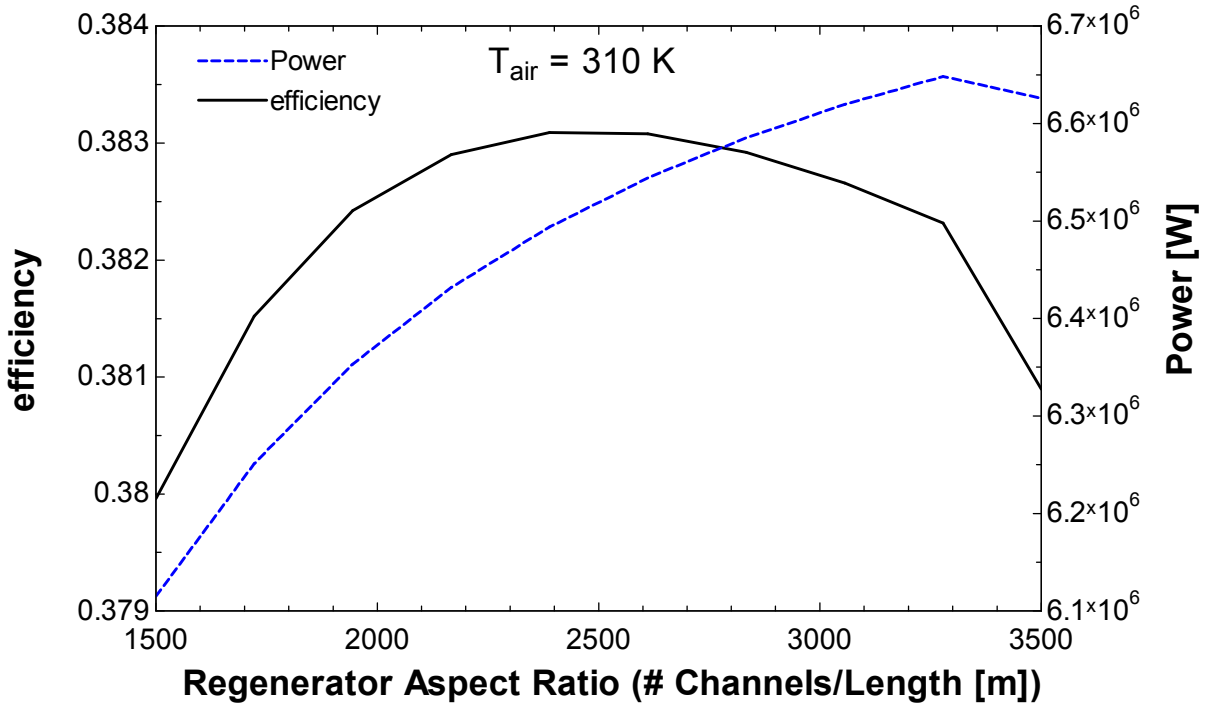

Figure 5-39: Cycle Performance v. Regenerator Aspect Ratio

Smaller aspect ratios (corresponding to a longer, thinner heat exchanger) provider higher CO_2 velocities and enhanced heat transfer, but also greater pressure drops. Larger aspect ratios reduce pressure drops and provide greater compressor inlet density, this in turn increases power input. A balance can be struck to optimize either efficiency or power, depending on design goals.

5.4 Designed Cycle Performance

The water-cooled, air-cooled, and hybrid-cooled Brayton cycles were selected using the methodology explained above, that is by choosing each parameter in turn, starting with the most sensitive parameters and moving to the least sensitive. This process yielded the following cycle configurations. Note that the air, water, and hybrid-cooled Brayton cycles were each modeled with design-sized heat exchangers, and also with over and under-sized heat exchangers. Performance results for the over and under-sized cases are found in Appendix H.

5.4.1 Designed Water-Cooled Cycle Performance

The designed cycle has the characteristics laid out in Table 5-3.

Table 5-3: Cycle Parameters for Designed Water-Cooled Brayton

Water-Cooled: Designed (Medium HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*875*	
Regenerator Area [m^2]	*1575*	
Precooler Area [m^2]	*1050*	
PHX Salt Side Channel Width/Depth [mm]	*0.71*	
PHX CO2 Side Channel Width/Depth [mm]	*2.3*	
Regenerator Hot Side Channel Width/Depth [mm]	*1.8*	
Regenerator Cold Side Channel Width/Depth [mm]	*2.2*	
Precooler CO2 Side Channel Width/Depth [mm]	*1.4*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*10000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*4000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*10000*	
Cooling Water Design Demand Temperature [K]	**299.7**	Performance
Cooling Water Outlet Temperature [K]	**307.5**	
Power [MW]	**11.7**	
Efficiency	**41.75%**	

When compared against the baseline case shown in Table 5-1, we see slightly improved efficiency along with slightly decreased power, and a slightly reduced cooling water temperature rise.

5.4.2 Designed Air-Cooled Cycle Performance

The designed cycle has characteristics laid out in Table 5-4, with ambient air temperatures of 310 K (37 C), which corresponds to a summer afternoon in Daggett, CA.

Table 5-4: Cycle Parameters for Designed Air-Cooled Brayton

Air-Cooled: Designed (Design HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Design Salt Inlet Temperature [K]	900	
Design Air Temperature [K]	310	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	325	
Aircooler Area [m^2]	16000	
Minimum Air-CO2 Approach Temperature [K]	15	
Design Pressure Ratio	2.33	Designed Parameters
Primary Heat Exchanger Area [m^2]	700	
Regenerator Area [m^2]	1633	
Fan Power @Air = 310 K [kW]	386	
PHX Salt Side Channel Width/Depth [mm]	1.2	
PHX CO2 Side Channel Width/Depth [mm]	2.4	
Regenerator Hot Side Channel Width/Depth [mm]	1.95	
Regenerator Cold Side Channel Width/Depth [mm]	1.5	
PHX Aspect Ratio [# Channels/Length [m]]	4500	
Regenerator Aspect Ratio [# Channels/Length [m]]	2500	
Power [MW]	**6.52**	Performance
Efficiency	**38.3%**	

When compared to the water-cooled Brayton described in Table 5-3, we see a greatly reduced power and moderately reduced efficiency. Degraded performance must be balanced against the potential advantages of heat rejection without the use of water.

5.4.3 Designed Hybrid-Cooled Cycle Performance

The designed cycle has the characteristics indicated in Table 5-5, which are identical to those values used in the water-cooled cycle, with the addition of an air-cooler. When compared against the water-cooled Brayton shown in Table 5-3, we see significant degradation in efficiency and power. The benefits of the hybrid-cooled Brayton lie in the reduced cooling water temperature rise and higher cooling water demand temperature, both of which correspond to lower water and energy use in the cooling tower.

Table 5-5: Cycle Parameters for Designed Hybrid-Cooled Brayton

Hybrid-Cooled: Designed (Medium HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Design Air Temperature [K]	310	
Aircooler Fan Power [kW]	63.5	
Aircooler Area [m^2]	16000	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*875*	
Regenerator Area [m^2]	*1575*	
Precooler Area [m^2]	*1050*	
PHX Salt Side Channel Width/Depth [mm]	*0.71*	
PHX CO2 Side Channel Width/Depth [mm]	*2.3*	
Regenerator Hot Side Channel Width/Depth [mm]	*1.8*	
Regenerator Cold Side Channel Width/Depth [mm]	*2.2*	
Precooler CO2 Side Channel Width/Depth [mm]	*1.4*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*10000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*4000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*10000*	
Cooling Water Design Demand Temperature [K]	**300.7**	Performance
Cooling Water Outlet Temperature [K]	**306.8**	
Power [MW]	**11.8**	
Efficiency	**41.60%**	

Now that three cycle designs have been chosen, the next chapter examines how the cycles perform on an annual basis, and how their performance compares to the Rankine cycle modeled by Wagner (2008).

6 Implementation of SCO_2 Brayton Model in TRNSYS

Chapters 2, 3, 4, and 5 discuss the development and implementation of design point models of the SCO_2 Brayton Cycle. The design point models allow evaluation of cycle performance at a fixed design point but they are not amenable for use in predicting the cycle performance over longer time periods (e.g. annual operation). To facilitate the simulation of the SCO_2 Brayton cycle operation over an annual basis, a more computationally expedient cycle model is needed. The design point model is exercised over a range of operating conditions and its corresponding performance curve fit to arrive at a "map" suitable for implementing in TRNSYS (Klein, 2010) to simulate long term cycle performance.

Implementation of the Brayton cycle curve fit model as a component in TRNSYS for evaluation with existing CSP components was based on a method developed by Wagner (2008), who created a Rankine cycle curve fit model to investigate its annual performance. Wagner created a detailed system model in EES and then implemented curve-fits of this detailed model into a component in TRNSYS.

In this chapter, the methodology used to map the performance of the SCO_2 Brayton design point model described in Chapters 2, 3, and 4 into TRNSYS as a component is discussed, along with its associated operating/control logic. The TRNSYS deck used to perform annual simulations is presented and both monthly and annual results for the air-cooled, hybrid-cooled, and water-cooled cycles are compared.

6.1 TRNSYS Implementation

Performance models for the water-cooled, hybrid-cooled, and air-cooled SCO_2 Brayton cycles are implemented in FORTRAN as curve fits. These curve fits are based on performance maps for each cycle configuration. The performance, expressed in terms of power, efficiency, and cooling water demand temperature, of the water-cooled cycle is mapped as a function of salt inlet

temperature, and the hybrid and air-cooled cycles are mapped as a function of salt inlet temperature and ambient air temperature. The curve fits do not depend on flow rates of molten salt or cooling water because the design point model only considered fixed salt and cooling water flow rates.

The SCO_2 Brayton cycle curve fit model in FORTRAN is composed of polynomials and simple logical operators. This section examines the logic behind the code.

Polynomial Operations

The demand salt and water flow rates are calculated by normalized values for the design-sized plant (the design-sized plant is the plant detailed in Chapter 5 of this thesis), a plant with the capacity ratio of 1 will demand salt and cooling water at a flow rate identical to that of the design-sized plant, whereas a plant with a capacity ratio of 2 will demand twice that flow rate. Once a capacity ratio is specified, the flow rates of water and salt are fixed. The salt flow rate for the design-sized plant (a capacity ratio of 1) is fixed at 150 kg/s, chosen because it is the default mass flow rate of molten salt through the receiver used in Wagner's thesis (2008). This flow rate is fixed because the cycle's performance is not sensitive to salt mass flow rate. The relationship between efficiency and salt mass flow rate for the design-sized water-cooled configuration is shown in Figure 6-1.

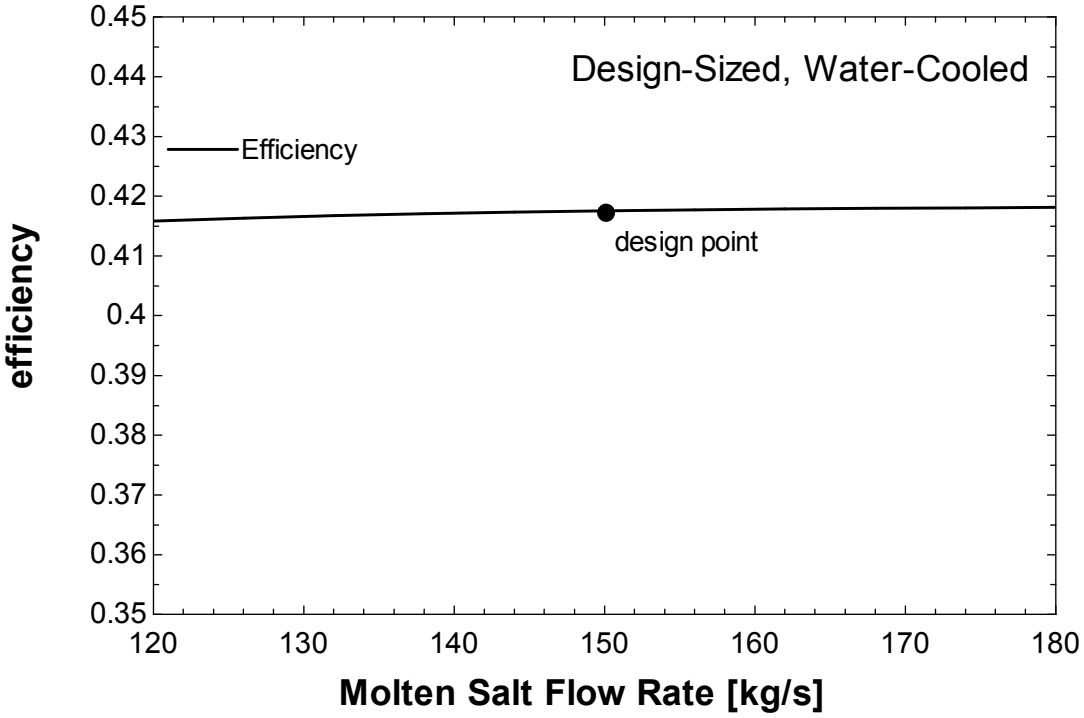

Figure 6-1: Cycle Efficiency vs. Salt Mass Flow Rate

Cycle efficiency is only minimally sensitive to salt mass flow rate.

Likewise, the cooling water mass flow rate is fixed (for a capacity ratio of 1) at 500 kg/s, a value chosen to produce a reasonable water temperature rise across the precooler (7.7 C). This parameter is also fixed due to the low sensitivity of cycle performance to cooling water mass flow rate, shown in Figure 6-2.

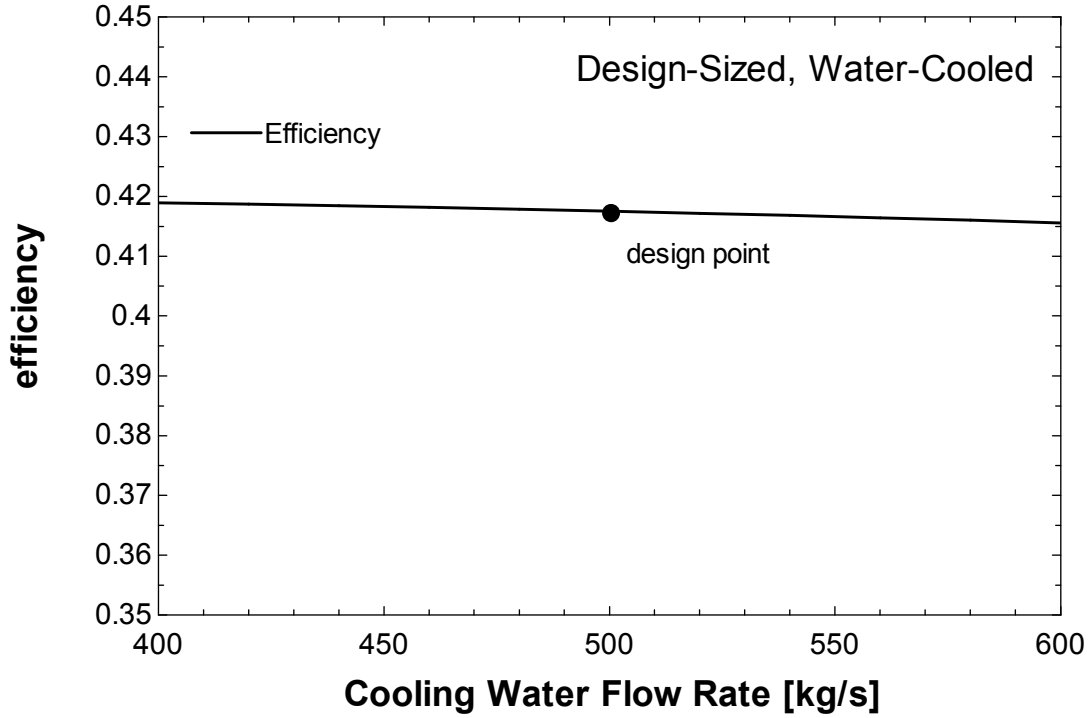

Figure 6-2: Cycle Efficiency vs. Cooling Water Mass Flow Rate

Cycle efficiency is only minimally sensitive to cooling water mass flow rate.

The demand inlet cooling water temperature, cycle power, water outlet temperature, and salt outlet temperature are then calculated by means of polynomial curve fits developed from the analysis results from the EES model. These polynomials are of the form shown in equations 6-1 through 6-4 and the polynomials themselves are shown later in the chapter, in equations 6-11 through 6-17 . For the water-cooled cycle, the performance mapping is solely a function of salt inlet temperature, since with the flow rates of molten salt and water fixed, and the cooling water inlet temperature an output, the salt temperature is the only remaining independent variable for a fixed plant configuration. Note that $W_{net,normalized}$ is normalized by power output at the design point.

$$\dot{W}_{net,normalized} = f(T_{salt,in}, power_{rated}) \quad 6\text{-}1$$

$$T_{water,in} = f(T_{salt,in}) \quad 6\text{-}2$$

$$T_{water,out} = f(T_{salt,in}) \quad 6\text{-}3$$

$$T_{salt,out} = f(T_{salt,in}) \quad 6\text{-}4$$

For the hybrid-cooled Brayton cycle plant, the performance (in terms of power, efficiency, and cooling water demand temperature) is a function of the inlet salt temperature and the ambient dry bulb air temperature, assuming fixed water and salt flow rates (discussed above and listed, along with all other relevant cycle parameters for each Brayton configuration and size, in Appendix H). Dependence on ambient dry bulb temperature is due to changing air-cooler performance at different dry bulb temperatures. Polynomials of the forms shown in equations 6-5 through 6-8 represent the performance of the hybrid-cooled Brayton cycle.

$$\dot{W}_{net,normalized} = f(T_{salt,in}, power_{rated}, T_{air,ambient}) \quad 6\text{-}5$$

$$T_{water,in} = f(T_{salt,in}, T_{air,ambient}) \quad 6\text{-}6$$

$$T_{water,out} = f(T_{salt,in}, T_{air,ambient}) \quad 6\text{-}7$$

$$T_{salt,out} = f(T_{salt,in}, T_{air,ambient}) \quad 6\text{-}8$$

The air-cooled cycle is represented by polynomials of a similar form but without any dependence on cooling water inlet or outlet temperature. The form of the required polynomials are given in equations 6-9 and 6-10.

$$\dot{W}_{net,normalized} = f(T_{salt,in}, power_{rated}, T_{air,ambient}) \quad 6\text{-}9$$

$$T_{salt,out} = f(T_{salt,in}, T_{air,ambient}) \quad 6\text{-}10$$

These polynomials are used to calculate cycle performance during the periods when the cycle is operating. For more details on the calculation of these polynomials refer to the next section of this chapter, which deals with the linear regressions used to develop the polynomials.

Operating Hours

Outside of the plant's user-defined operating hours, the Brayton module does not change the temperature of the water. It also sets back the demand temperature for heating salt and cooling water to values that will not require cooling tower operation, and will only require natural gas heating in very rare cases to avoid salt freezing.

If, however, the plant has molten salt available at a very high temperature (above the design salt inlet temperature of 900 K) at a time outside the operating hours, it will continue to operate and produce electricity. This is done to avoid excessively high salt temperatures during periods where the plant is unable to convert the solar resource into electricity at a sufficient rate. This condition occurs when the solar resource is large enough such that the molten salt storage tank is still above the design salt inlet temperature at scheduled plant shutdown.

Cooling Water Failure

When the cooling water inlet temperature is above the temperature demanded by the power block, the Brayton plant shuts down. Unlike times outside of operating hours, however, it continues to demand water at operating temperatures – and will start producing power again once cooling water temperatures are low enough. In reality a plant would likely operate at a reduced capacity, but the operation of the design point model (detailed in Chapter 4) is based on a fixed compressor inlet CO_2 temperature. As such, the cooling water must be provided at the required temperature to achieve this operating condition. Shutting down the plant when this condition occurs is a conservative measure, and sets a lower bound on plant performance.

6.2 Development of Performance Maps by Linear Regression

The design point models for each of the three cycles (water-cooled, hybrid-cooled, and air-cooled) described in Chapters 4 and 5 were each run across the range of independent variables shown in Table 6-1. The curve fits developed in this Chapter are assumed to be valid across this range of conditions.

Table 6-1: Independent Variable Mapping Range

	Lower Range	Upper Range
Molten Salt Temperature [K]	700	970
Dry Bulb Temperature [K]	270	320

The simulation results were used to create polynomial curve fits via linear regression to quantify cycle performance across the range of independent variables, as discussed earlier in this chapter. This section discussed the linear regressions performed for the water-cooled cycle. Note that the regressions discussed in-depth this section are for the design-sized water-cooled cycle, a summary of regression results for all cycle configurations and plant sizes can be found in Table 6-6, and the full polynomials can be found in Appendix I. The methodology shown in this section, however, is used for all Brayton configurations and sizes.

The performance of the water-cooled cycle is a function of the molten salt inlet temperature as discussed in Chapter 3. This discusses the mapping of each dependent variable via linear regression.

Normalized Net Power

Net power is normalized by the design power, all plant parameters are fixed at the design values discussed in Chapter 5 Normalized power results as a function of inlet salt temperature are shown in Figure 6-3.

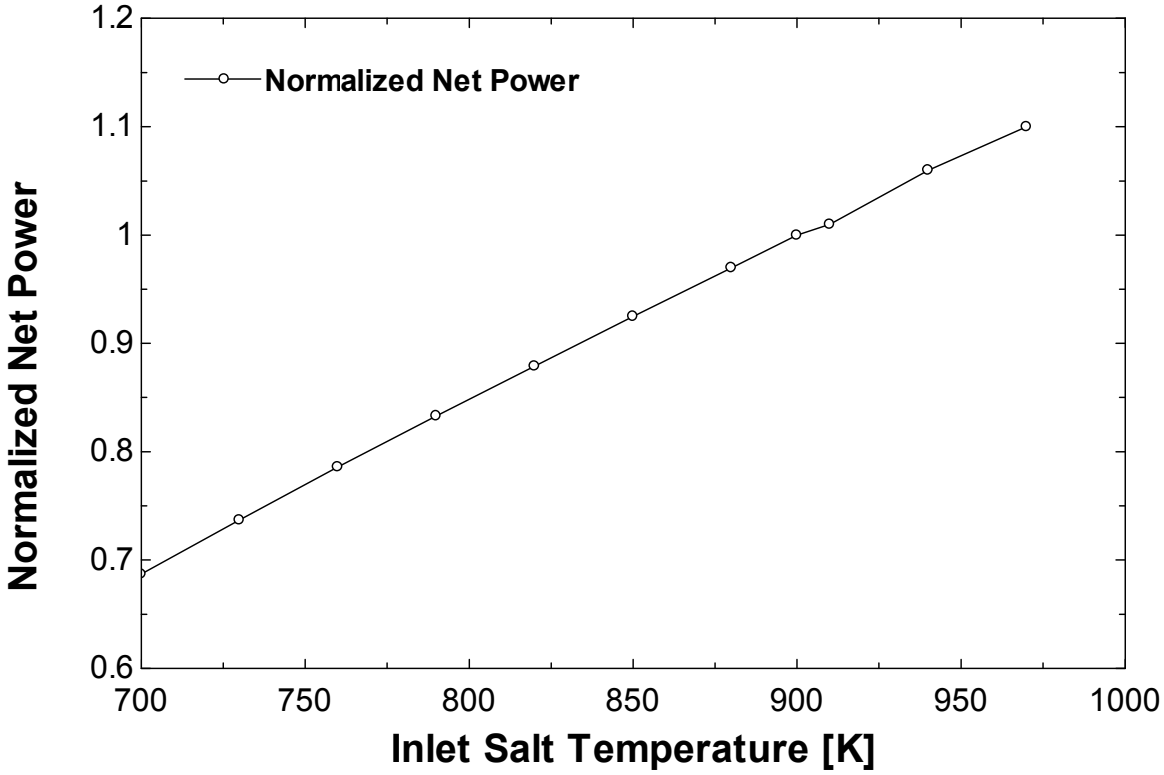

Figure 6-3: Simulated Results for Normalized Net Power vs. Inlet Salt Temperature

Increased inlet salt temperatures cause higher CO_2 temperatures at the turbine inlet, increasing power output. A linear regression of the mapping runs for normalized net power yields a polynomial of the form shown in equation 6-11.

$$\dot{W}_{net,normalized} = a_0 + a_1 \cdot T_{salt,in} \qquad 6\text{-}11$$

The root mean square (rms) error associated with this approximation is 3.5e-3, suggesting robust approximation across the range of salt temperatures investigated.

For the air-cooled and hybrid-cooled cycles, which are sensitive to the ambient dry bulb temperature, the equation for power is of the form shown in 6-12.

$$\dot{W}_{net,normalized} = a_0 + a_1 \cdot T_{salt,in} + a_2 \cdot T_{salt,in}{}^2 + a_3 \cdot T_{air,ambient} + a_4 \cdot T_{air,ambient}{}^2 \qquad 6\text{-}12$$

The coefficients for each configuration and each size are listed in Table 6-2.

Table 6-2: Normalized Power Curve Fit Coefficients

Curve Fit Coefficients - Normalized Power						
		a0	a1	a2	a3	a4
water-cooled	small	-3.95E-01	1.55E-03	N/A	N/A	N/A
	design	-3.77E-01	1.53E-03	N/A	N/A	N/A
	large	-3.64E-01	1.51E-03	N/A	N/A	N/A
hybrid-cooled	small	-3.87E-01	1.57E-03	N/A	-7.67E-05	N/A
	design	-3.81E-01	1.53E-03	N/A	-1.63E-06	N/A
	large	-3.58E-01	1.52E-03	N/A	-3.45E-05	N/A
air-cooled	small	-1.64E+01	4.97E-03	-1.82E-06	1.09E-01	-2.00E-04
	design	-1.80E+01	2.52E-03	-3.98E-07	1.24E-01	-2.22E-04
	large	-1.51E+01	8.36E-04	5.93E-07	1.07E-01	-1.90E-04

Cooling Water Inlet Temperature

The cooling water inlet temperature demanded by the cycle as a function of molten salt inlet temperature is shown in Figure 6-4.

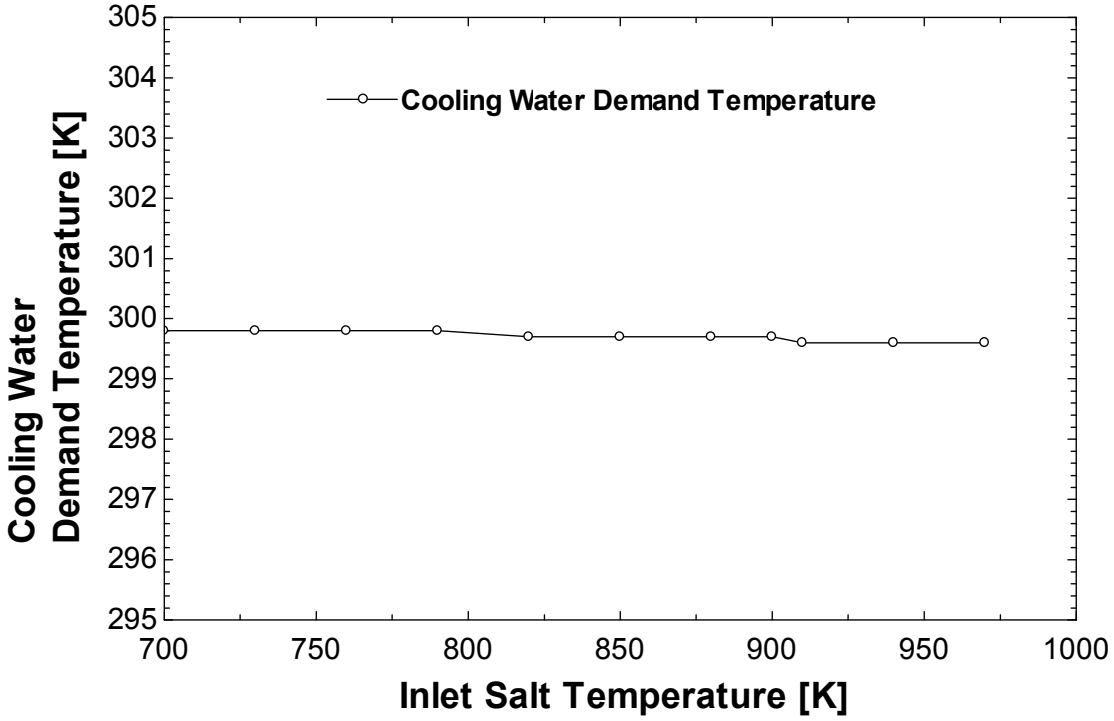

Figure 6-4: Simulated Results for Cooling Water Demand Temperature vs. Inlet Salt Temperature

The demanded cooling water temperature decreases slowly as inlet salt temperature increases. A linear regression of the mapping runs for the demanded cycle cooling water inlet temperature yields a polynomial of the form shown in equation 6-13.

$$T_{water,in} = b_0 + b_1 \cdot T_{salt,in} \qquad 6\text{-}13$$

The rms error associated with this approximation is 3.3e-2 [K], suggesting robust approximation across the range of salt temperatures investigated.

For the air-cooled and hybrid-cooled cycles, which are sensitive to the ambient dry bulb temperature, the equation for cooling water inlet temperature is of the form shown in 6-14.

$$T_{water,in} = b_0 + b_1 \cdot T_{salt,in} + b_2 \cdot T_{salt,in}^{\ 2} + b_3 \cdot T_{air,ambient} + b_4 \cdot T_{air,ambient}^{\ 2} \qquad 6\text{-}14$$

The coefficients for each configuration and each size are listed in Table 6-3.

Table 6-3: Cooling Water Inlet Temperature Curve Fit Coefficients

Curve Fit Coefficients - Cooling Water Inlet Temperature						
		b0	b1	b2	b3	b4
water-cooled	small	2.97E+02	-1.47E-03	N/A	N/A	N/A
	design	3.00E+02	-8.72E-04	N/A	N/A	N/A
	large	3.03E+02	-5.06E-04	N/A	N/A	N/A
hybrid-cooled	small	6.76E+02	6.77E-04	-7.73E-07	-2.40E+00	3.82E-03
	design	5.58E+02	-8.69E-04	4.92E-08	-1.60E+00	2.60E-03
	large	4.78E+02	1.55E-04	-4.00E-07	-1.12E+00	1.78E-03
air-cooled	small	N/A	N/A	N/A	N/A	N/A
	design	N/A	N/A	N/A	N/A	N/A
	large	N/A	N/A	N/A	N/A	N/A

Cooling Water Outlet Temperature

The cooling water outlet temperature as a function of salt inlet temperature is shown in Figure 6-5.

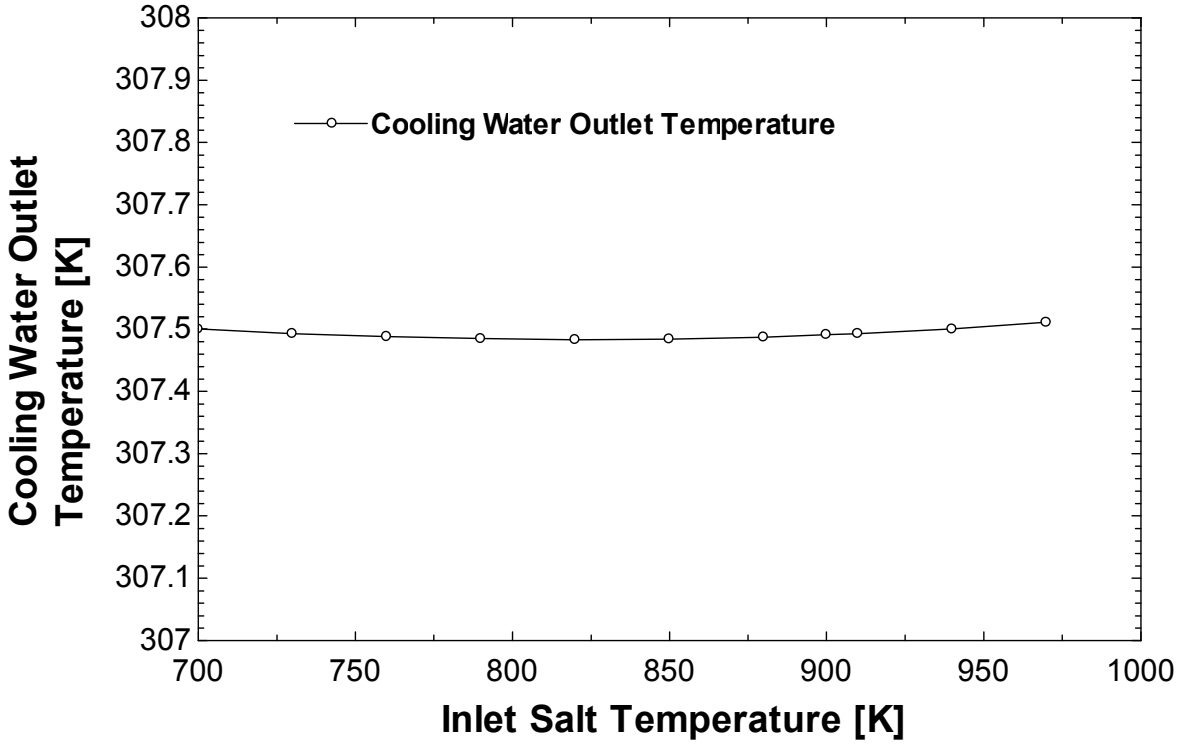

Figure 6-5: Simulated Results for Cooling Water Outlet Temperature vs. Inlet Salt Temperature

A linear regression of the mapping runs for cooling water outlet temperature yields a polynomial of the form shown in equation 6-15.

$$T_{water,out} = c_0 + c_1 \cdot T_{salt,in} + c_2 \cdot T_{salt,in}{}^2 \qquad 6\text{-}15$$

The rms error associated with this approximation is 4.97e-4 [K], suggesting robust approximation across the range of salt temperatures investigated (700 - 970).

For the air-cooled and hybrid-cooled cycles, which are sensitive to the ambient dry bulb temperature, the equation for cooling water outlet temperature is of the form shown in 6-16.

$$T_{water,out} = c_0 + c_1 \cdot T_{salt,in} + c_2 \cdot T_{salt,in}{}^2 + c_3 \cdot T_{air,ambient} + c_4 \cdot T_{air,ambient}{}^2 \qquad 6\text{-}16$$

The coefficients for each configuration and each size are listed in Table 6-4.

Table 6-4: Cooling Water Outlet Temperature Curve Fit Coefficients

Curve Fit Coefficients - Cooling Water Outlet Temperature						
		c0	c1	c2	c3	c4
water-cooled	small	3.05E+02	-2.71E-03	1.69E-06	N/A	N/A
	design	3.08E+02	-1.98E-03	1.21E-06	N/A	N/A
	large	3.11E+02	-1.50E-03	7.75E-07	N/A	N/A
hybrid-cooled	small	4.89E+02	-1.72E-03	7.29E-07	-1.19E+00	1.93E-03
	design	3.82E+02	-1.08E-03	2.03E-07	-4.97E-01	8.32E-04
	large	3.01E+02	-5.52E-04	N/A	2.76E-02	N/A
air-cooled	small	N/A	N/A	N/A	N/A	N/A
	design	N/A	N/A	N/A	N/A	N/A
	large	N/A	N/A	N/A	N/A	N/A

Molten Salt Outlet Temperature

The salt outlet temperature as a function of salt inlet temperature is shown in Figure 6-6.

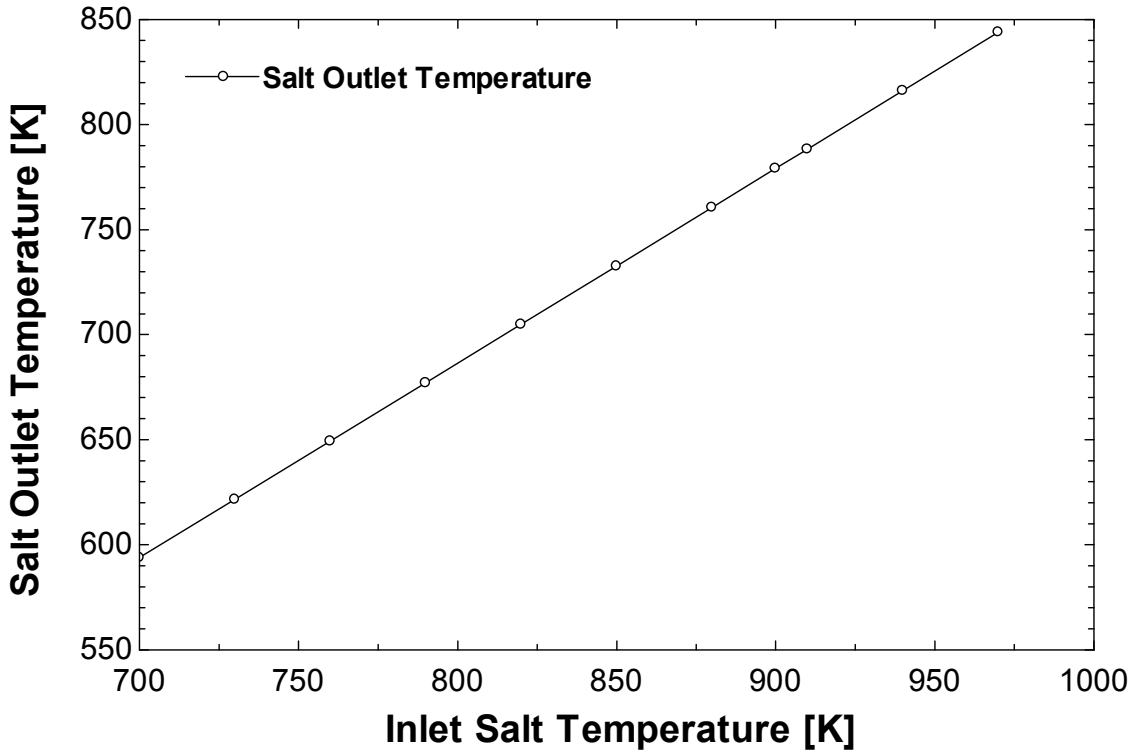

Figure 6-6: Simulated Results for Salt Outlet Temperature vs. Inlet Salt Temperature

The salt outlet temperature increases almost linearly with inlet salt temperature, and a linear regression of the mapping runs for molten salt outlet temperature yields a polynomial of the form shown in equation 6-17.

$$T_{salt,out} = d_0 + d_1 \cdot T_{salt,in} \qquad 6\text{-}17$$

The rms error associated with this approximation is 9.4e-2 [K], suggesting robust approximation across the range of salt temperatures investigated.

For the air-cooled and hybrid-cooled cycles, which are sensitive to the ambient dry bulb temperature, the equation for cooling water outlet temperature is of the form shown in 6-18 .

$$T_{water,out} = d_0 + d_1 \cdot T_{salt,in} + d_2 \cdot T_{salt,in}^{\ 2} + d_3 \cdot T_{air,ambient} + d_4 \cdot T_{air,ambient}^{\ 2} \qquad 6\text{-}18$$

The coefficients for each configuration and each size are listed in Table 6-5.

Table 6-5: Cooling Salt Outlet Temperature Curve Fit Coefficients

Curve Fit Coefficients - Salt Outlet Temperature						
		d0	d1	d2	d3	d4
water-cooled	small	-5.18E+01	9.22E-01	N/A	N/A	N/A
	design	-5.48E+01	9.27E-01	N/A	N/A	N/A
	large	-5.74E+01	9.30E-01	N/A	N/A	N/A
hybrid-cooled	small	-5.39E+01	9.22E-01	N/A	8.06E-03	N/A
	design	-5.46E+01	9.25E-01	N/A	3.01E-17	N/A
	large	-5.88E+01	9.30E-01	N/A	4.92E-03	N/A
air-cooled	small	1.70E+03	9.98E-01	2.85E-05	-1.25E+01	2.21E-02
	design	1.59E+03	9.46E-01	4.10E-06	-1.15E+01	2.03E-02
	large	1.45E+03	8.68E-01	5.19E-05	-1.02E+01	1.78E-02

\

The same linear regression methodology is performed for all mapped variables for each cycle. The results of this analysis are summarized in Table 6-6, which lists the number of points regressed for each configuration, the range of the independent variables investigated, as well as the order and rms error for each regression.

Table 6-6: Linear Regression Summary

	Water-Cooled (small)	Water-Cooled (design)	Water-Cooled (large)	Hybrid-Cooled (small)	Hybrid-Cooled (design)	Hybrid-Cooled (large)	Air-Cooled (small)	Air-Cooled (design)	Air-Cooled (large)
Number of Points Regressed	11	11	11	84	84	84	84	84	84
Salt Temperature Range [K]	700-970	700-970	700-970	700-970	700-970	700-970	700-970	700-970	700-970
Air Temperature Range [K]	N/A	N/A	N/A	270-320	270-320	270-320	270-320	270-320	270-320
Normalized Power Linear Regression Order	1st	1st	1st	1st	1st	1st	2nd	2nd	2nd
Cooling Water Inlet Temp Linear Regression Order	1st	1st	1st	2nd	2nd	2nd	N/A	N/A	N/A
Cooling Water Outlet Temp Linear Regression Order	2nd	2nd	2nd	2nd	2nd	1st	N/A	N/A	N/A
Salt Outlet Temp Linear Regression Order	1st	1st	1st	1st	1st	1st	2nd	2nd	2nd
Normalized Power rms	3.98E-03	3.53E-03	3.00E-03	3.32E-03	3.18E-03	3.00E-03	2.93E-02	1.82E-02	2.55E-02
Cooling Water Inlet Temp rms [K]	2.90E-02	3.29E-02	2.84E-02	4.41E-01	2.88E-01	1.92E-01	N/A	N/A	N/A
Cooling Water Outlet Temp rms [K]	1.30E-03	4.97E-04	3.22E-04	2.14E-01	8.55E-02	3.37E-02	N/A	N/A	N/A
Salt Outlet Temp rms [K]	9.89E-02	9.41E-02	1.10E-01	9.48E-02	8.42E-02	1.06E-01	2.98E+00	1.80E+00	3.52E+00

With the SCO_2 Brayton Cycle implemented as a TRNSYS type using the polynomials and logic described in the sections above, it is then simulated in TRNSYS using the deck described in the next section.

6.3 Implementing Brayton TRNSYS Type in CSP Test Deck

The SCO_2 Brayton TRNSYS type is represented by the TRNSYS type information flow diagram shown in Figure 6-7.

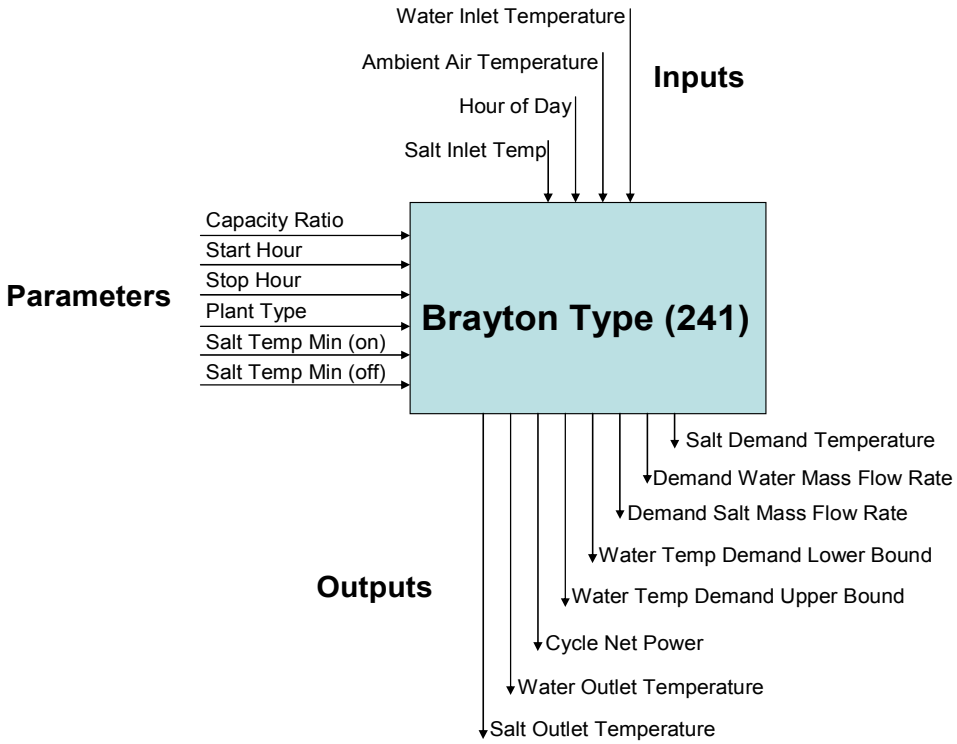

Figure 6-7: Brayton Type Information Flow Diagram

The user sets the capacity ratio and plant type in accordance with Table 6-7 , along with the minimum salt temperature demanded by the cycle during operations, and the minimum salt temperature demanded during non-operating hours (to prevent freezing).

Table 6-7: Plant Type Numbers

Plant Type	Plant Name	Total HX Area (not including air-cooler) [m^2]	Aircooler Area [m^2]	Minimum Air Approach [K]	Fixed Air-cooler Fan Power [kW]	Design Net Power [MW]
0	Water-Cooled (small)	2450	N/A	N/A	N/A	11.4
1	Water-Cooled (design)	3500	N/A	N/A	N/A	11.7
2	Water-Cooled (large)	5250	N/A	N/A	N/A	11.9
3	Hybrid-Cooled (small)	2450	16000	N/A	50	11.3
4	Hybrid-Cooled (design)	3500	16000	N/A	50	11.8
5	Hybrid-Cooled (large)	5250	16000	N/A	50	11.9
6	Air-Cooled (small)	2333	8000	20	N/A	5.8
7	Air-Cooled (design)	2333	16000	15	N/A	6.5
8	Air-Cooled (large)	2333	25000	10	N/A	7.1

With the SCO_2 Brayton parameters set, the power block component is simulated in the context of a TRNSYS deck composed of CSP components with a configuration based on work by Wagner (2008). These components are arranged as shown in Figure 6-8.

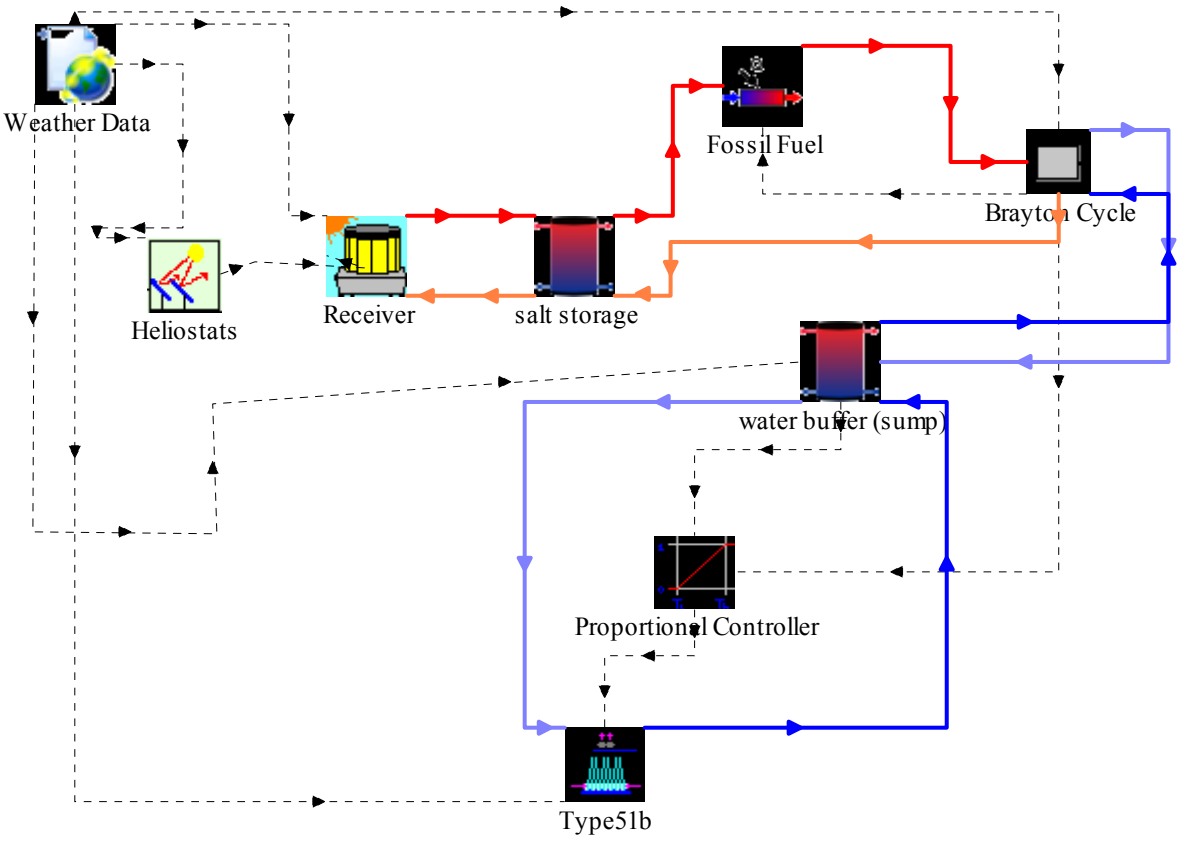

Figure 6-8: TRNSYS Deck Schematic

Parameters for the test deck are given in Table 6-8 and remain fixed throughout all annual analysis performed in TRNSYS. These parameters were based on work by Wagner (2008) and Patnode (2006).

Table 6-8: Fixed TRNSYS Test Deck Parameters

	Plant Value	Notes	Component
Weather File	Dagget, California	Wagner (2008)	Weather
Number of Reciever Panels	24	Wagner (2008)	Receiver
Reciever Diameter [m]	6.2	Wagner (2008)	
Tower Height [m]	76.2	Wagner (2008)	
Receiver Outlet Salt Temperature [C]	700		
Plant Start Hour	10:00 AM	Wagner (2008)	SCO2 Brayton
Plant Stop Hour	11:00 PM	Wagner (2008)	
Non-Operating Salt Demand Temperature [C]	257	Wagner (2008)	
Operating Salt Demand Temperature [C]	550		
Salt Storage Volume [m^3]	15000		Salt Storage Tank
Salt Storage Tank Type	Stratified (20 nodes)		
Cooling Tower Design Flow Rate [m^3/s]	3600	Patnode (2006)	Cooling Tower
Cooling Tower Design Fan Power [kW]	984	Patnode (2006)	

The receiver outlet temperature of 700°C was chosen to ensure that, when mixed with cooler salt returning from the power block, the salt storage tank could provide salt at the 623°C (900 K) design molten salt inlet temperature for the SCO_2 Brayton cycle.

The salt demand temperature during non-operating hours was chosen as a very low value that would allow the salt to remain molten (Wagner, 2008). The demand temperature during operating hours was chosen as 550°C, a value that limits the percentage of thermal energy from fossil fuel at ~25% for the design-sized water-cooled Brayton cycle. Specifying a higher temperature would improve annual efficiency and energy production for the plant, but would require more fossil fuel use, whereas choosing a lower temperature would degrade efficiency and power production but reduce fossil fuel use.

The size of the salt storage tank in Table 6-4 was chosen to allow sufficient thermal capacitance and maintain the temperature of molten salt supply to the Brayton cycle to remain close to the design value. Smaller values allowed the temperature of the molten salt to vary dramatically, degrading annual performance. The relationship between annual efficiency and molten salt storage volume (normalized by plant capacity) is shown in Figure 6-9.

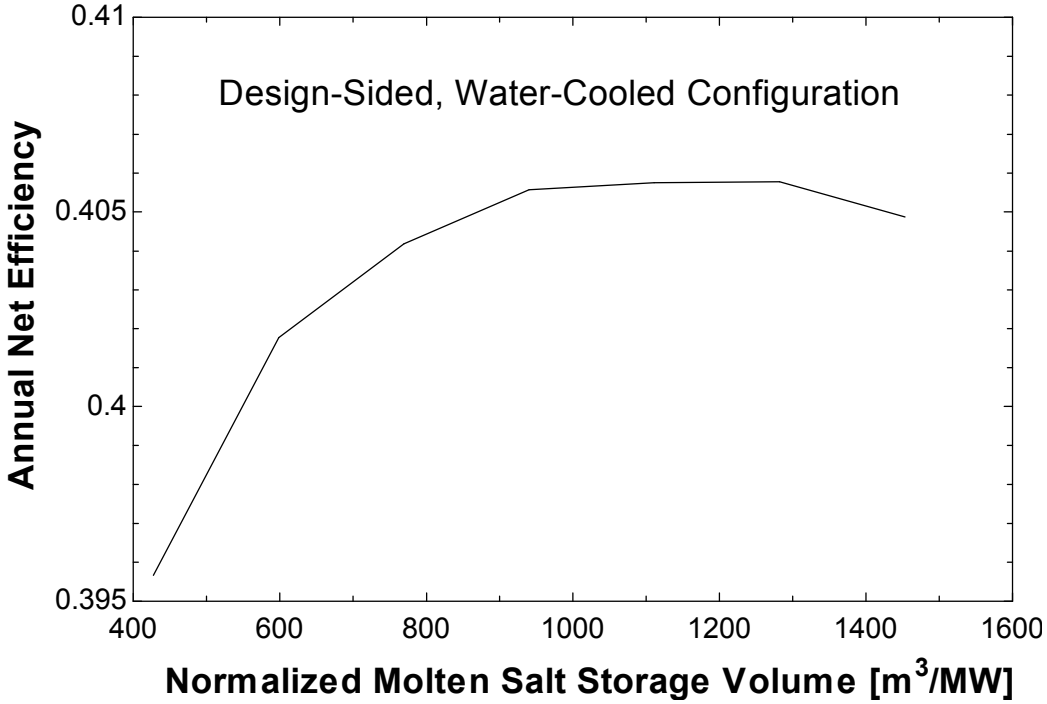

Figure 6-9: Annual Efficiency v. Salt Storage Volume

These results support Wagner's (2008) conclusion that for a 11 MW CSP plant a 1500m^3 storage tank would be insufficient. A normalized tank size of 1300m^3 per MW of plant capacity was found to offer peak annual efficiency, and a tank size of 15000m^3 was chosen for the design-sized water-cooled plant to correspond with this optimum. The tank size was kept constant for all plant configurations to allow for a valid performance comparison..

Note that eventually adding more storage degrades annual performance. This effect is due to the increased thermal losses from the storage tank, as shown in Figure 6-10. Note that plotted losses are normalized by plant capacity, and so have units of hours (MWh normalized by MW).

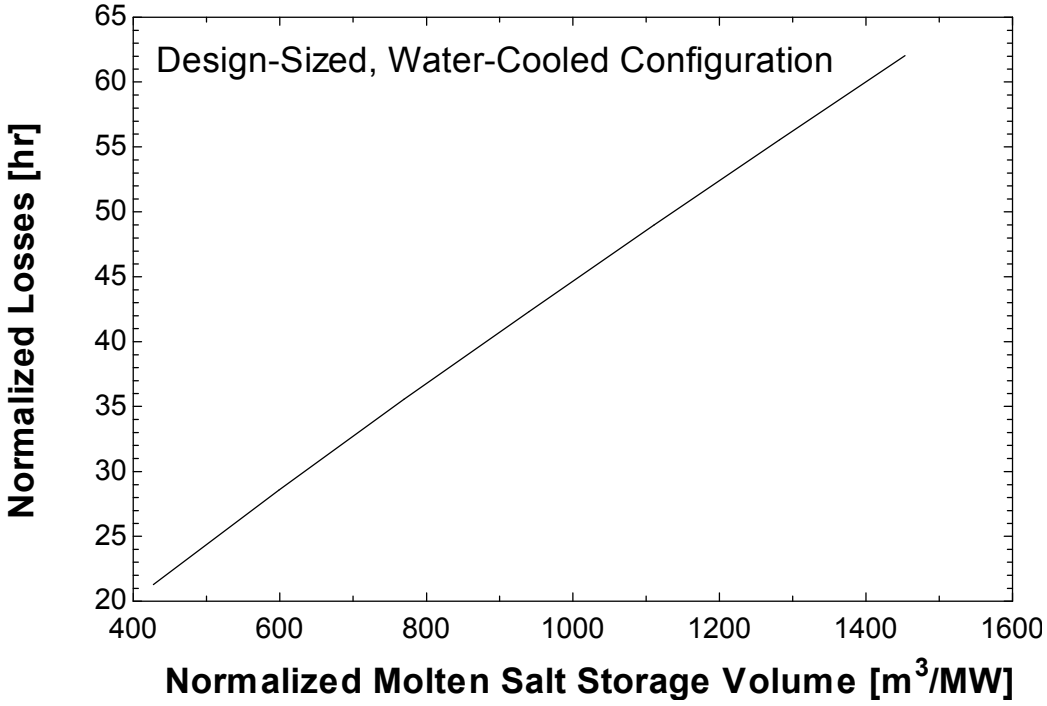

Figure 6-10: Normalized Thermal Losses from Storage v. Normalized Molten Salt Storage Volume

With increased tank size, the added surface area eventually allows great enough thermal losses to degrade annual efficiency.

Salt Side Operation

The Brayton type outputs two salt minimum temperatures – the higher is for plant operating hours, the lower for off-operation hours to prevent freezing. These temperatures must be met either by solar thermal energy or if that proves insufficient by natural gas boost from the 'Fossil Fuel' module. The Brayton cycle's salt pump is assumed to operate only during periods during which power is being produced, the flow rate is fixed at 100 kg/s for the design-sized plant.

Once heat is extracted from the salt in the Brayton cycle it is returned to the salt storage tank. Cool salt from the tank is then pumped through the central receiver (the tower that heats the molten salt by absorbing radiation concentrated by the heliostat field), with the flow rate through the receiver controlled to maintain a receiver outlet temperature for the salt of 700°C.

Hot salt from the receiver is in turn pumped to the salt storage tank – and hot salt from the tank is then pumped to the Brayton cycle, and if its temperature is below the minimum demand temperature for the cycle it is heated in the fossil fuel booster.

Water Side Operation

The Brayton type outputs a pair of demand temperatures for cooling water, the first being the temperature the cycle requires for operation and a second being one degree (K) colder. These two temperatures are used to set the upper and lower bounds for a proportional controller that governs fan power in the cooling tower, and are calculated by means of the curve fits discussed earlier in this chapter. The input signal to the proportional controller is the cooling water tank temperature. By this control mechanism the cooling tower fan power is modulated to maintain the required cooling water inlet temperature for operation.

If the cooling tower is unable to provide cool enough water the cycle will temporarily stop producing power. During this time, both the salt and cooling water will continue to circulate without temperature change through the power block. The cycle will resume production when the cooling water temperature is again inside bounds.

During operation, the cooling water temperature will rise as it flows through the precooler, with a design temperature rise of 7.7 K. The warm return water leaves the cycle and flows back to the cooling water storage tank. Warm water from the storage tank is in turn cooled by the tower (using the proportional control scheme described above) and returned to the storage tank.

6.4 Annual Results

The creation of a SCO_2 Brayton TRNSYS type, and the integration of that TRNSYS type into the CSP test deck described in the preceding section allows for annual simulations. Annual simulations were run with the test deck conditions laid out in Table 6-8 for each plant type in Table 6-7. The performance of the different SCO_2 Brayton configurations are compared in terms of annual electricity production, annual net efficiency (including parasitic fan power and salt pumping losses), annual water use (the amount of water evaporated in the cooling tower), and

the percentage of thermal energy from solar. The load factor is also listed, defined in equation 6-19.

$$LoadFactor = \frac{Energy_{net}}{Capacity \cdot Hours_{operation}} \qquad 6\text{-}19$$

These results are found in Table 6-9, alongside results from Wagner's Rankine-based CSP simulation.

Table 6-9: Annual Simulation Results

Plant Type	Plant Name	Annual Electricity Production [GWh]	Annual Net Efficiency	Annual Water Use [litres]	Annual Solar Themal Energy Use [GWh]	Annual Fossil Themal Energy Use [GWh]	% of Solar Energy from Solar	Load Factor
0	Water-Cooled (small)	49.9	38.7%	1.98E+08	98.3	30.6	76.3%	88.8%
1	Water-Cooled (design)	53.4	40.5%	1.30E+08	98.3	33.4	74.6%	89.3%
2	Water-Cooled (large)	54.6	41.6%	9.50E+07	98.3	33	74.9%	89.8%
3	Hybrid-Cooled (small)	51.4	38.8%	1.37E+08	98.3	34.3	74.1%	89.0%
4	Hybrid-Cooled (design)	53.7	40.3%	9.76E+07	98.3	34.8	73.9%	89.1%
5	Hybrid-Cooled (large)	54.4	41.4%	7.84E+07	98.3	33.1	74.8%	89.5%
6	Air-Cooled (small)	39.5	37.3%	N/A	98.6	7.3	93.1%	118.3%
7	Air-Cooled (design)	41.8	39.4%	N/A	98.6	7.6	92.8%	113.0%
8	Air-Cooled (large)	42.4	39.7%	N/A	98.6	8.3	92.2%	106.8%
N/A	Rankine (Wagner, 2008)	50.9	36.1%	*Not Available*	100.16	40.86	71.0%	*100% (Calculated by Seidel)*

These annual results are discussed below, starting with the water-cooled results and proceeding to the hybrid-cooled and air-cooled results.

Water-Cooled

As the size of the water-cooled plant's heat exchangers increased (from 70% of design for the 'small' case to 150% of design for the 'large' case) the annual power production increased, alongside annual efficiency. All three sizes were significantly more efficient than the Rankine plant (despite the Rankine model neglecting parasitic cooling tower fan power), with the 'design' and 'large' cases also producing more power. All three cases require less fossil fuel usage than the Rankine cycle, due to their higher thermal efficiencies.

Water use in the cooling tower is reduced with larger heat exchangers, because a larger precooler allows the cycle to demand cooling water at a higher temperature.

Hybrid-Cooled

The hybrid-cooled Brayton's annual power production and net efficiency are very similar to that of the water-cooled Brayton. The 'small' and 'design' cases actually produce slightly more power on an annual basis. This is due to the larger-diameter (compared to the precooler channels) CO_2 tubes in the air-cooler, which cool the relatively warm and low density CO_2 before its entry into the precooler, whereas CO_2 in the water-cooled cycle is forced immediately through the much smaller precooler channels (see Chapter 4 for more information on the modeling methodology behind the hybrid-cooled Brayton). The hybrid-cooled configuration therefore has smaller pressure drops through the precooler than the water-cooled configuration, which results in a slight power improvement. The water cooled cycle doesn't use larger channels because that would drastically degrade heat transfer at the cooler, higher density end of the precooler where CO_2 velocity would be reduced. The hybrid cycle benefits here because it effectively has a '2-node' converging channel heat exchanger. Of course it also benefits because it has the same sized precooler as the water-cooled cycle, and also an additional air-cooler. The benefit of decreased pressure drops for the hybrid-cooled configuration is somewhat counter-balanced by the parasitic requirements of the air-cooler's fans, which result in slightly degraded performance when comparing the 'large' water-cooled and hybrid-cooled configurations.

Annual Water use, when compared against the water-cooled configuration, is reduced by 30.8% for the 'small' case, 24.9% for the 'design' case, and 17.5% for the 'large' case. Reductions are significant across the size range, but are most pronounced in plants with under-sized heat exchangers which in the absence of air-cooling would require very low cooling water temperatures to meet the design CO_2 compressor inlet temperature (see Chapters 3 and 4 for information on cycle control).

Air-Cooled

The design power of the air-cooled cycle is significantly less than that of either the water-cooled or hybrid-cooled configurations (see Table 6-7 for a comparison between the plant types). This results in the cycle being able to convert less heat from the solar field, and therefore less reliance upon fossil fuel generation.

Annual power production is significantly reduced for all air-cooled configurations, while efficiency (though lower than that of equivalent hybrid and water-cooled SCO_2 Brayton cycles) is still significantly higher than the Rankine reference case for all sizes. Water consumption, of course, is completely eliminated.

The annual load factor for all three air-cooled configurations exceeds 100%, this is due to the reference power for the plant being defined with ambient air temperatures of 310 K. Since the average temperature for Daggett is below 310 K, the plant often operates at a power output above the design value.

6.5 Monthly Results

While annual results give a good summary of cycle performance, monthly results can provide further insight. This is particularly true because a CSP plant is likely to be most profitable during periods of peak electricity prices during warm summer months. This section compares performance for the three plant configurations (at design sizes) during the months of January (winter), May (shoulder season), and August (summer).

Monthly results for January are shown in Table 6-10.

Table 6-10: January Simulation Results

Plant Type	Plant Name	January Electricity Production [GWh]	January Net Efficiency	January Water Use [litres]	January Solar Themal Energy Use [GWh]	January Fossil Themal Energy Use [GWh]	January % of Solar Energy from Solar	January Load Factor	January Operating Hours
0	Water-Cooled (small)	4.37	43.6%	5.50E+06	5.8	4.22	57.9%	88.3%	434
1	Water-Cooled (design)	4.51	45.2%	1.70E+06	5.8	4.18	58.1%	88.8%	434
2	Water-Cooled (large)	4.61	46.3%	7.33E+04	5.8	4.15	58.3%	89.3%	434
3	Hybrid-Cooled (small)	4.37	43.4%	1.22E+05	5.8	4.26	57.7%	89.1%	434
4	Hybrid-Cooled (design)	4.54	45.0%	0.00E+00	5.8	4.29	57.5%	88.7%	434
5	Hybrid-Cooled (large)	4.6	46.1%	0.00E+00	5.8	4.17	58.2%	89.1%	434
6	Air-Cooled (small)	2.98	40.7%	N/A	5.82	1.5	79.5%	118.4%	434
7	Air-Cooled (design)	3.13	42.6%	N/A	5.81	1.54	79.0%	111.0%	434
8	Air-Cooled (large)	3.17	42.8%	N/A	5.81	1.6	78.4%	102.9%	434

All plants show excellent efficiency in January due to low ambient temperatures, which result in minimal parasitic fan use in the cooling tower. Water use is also extremely low, and even non-existent for two hybrid-cooled cases.

Electricity production and the fraction of thermal energy provided by solar energy, however, are both low compared to annual values due to reduced solar flux during winter.

Monthly results for May are shown in Table 6-11.

Table 6-11: May Simulation Results

Plant Type	Plant Name	May Electricity Production [GWh]	May Net Efficiency	May Water Use [litres]	May Solar Themal Energy Use [GWh]	May Fossil Themal Energy Use [GWh]	May % of Solar Energy from Solar	May Load Factor	May Operating Hours
0	Water-Cooled (small)	4.45	39.5%	1.50E+07	10	1.26	88.8%	89.9%	434
1	Water-Cooled (design)	4.59	40.9%	1.17E+07	10	1.22	89.1%	90.4%	434
2	Water-Cooled (large)	4.7	42.0%	8.85E+06	10	1.18	89.4%	91.0%	434
3	Hybrid-Cooled (small)	4.42	39.2%	1.28E+07	9.98	1.3	88.5%	90.1%	434
4	Hybrid-Cooled (design)	4.61	41.2%	9.80E+06	9.97	1.23	89.0%	90.0%	434
5	Hybrid-Cooled (large)	4.68	41.9%	7.48E+06	9.98	1.19	89.3%	90.6%	434
6	Air-Cooled (small)	3.78	37.8%	N/A	10	0	100.0%	122.5%	532
7	Air-Cooled (design)	4.04	40.4%	N/A	10	0	100.0%	118.4%	525
8	Air-Cooled (large)	4.12	41.2%	N/A	10	0	100.0%	113.1%	513

Efficiency is significantly reduced for the hybrid-cooled and water-cooled cycles when compared against January results, due to increased fan power requirements in the cooling tower. Increased ambient temperatures cause water use to increase dramatically when compared to January, most notably in the ‘small’ hybrid-cooled and water-cooled configurations. Efficiency for the air-cooled cycles is significantly reduced when compared against January results. This

reduction in efficiency is due to increased ambient temperatures requiring much more significant fan usage in the air-cooler.

Electricity production is higher for all cycles, when compared against January results. This is due to much greater solar flux, which also decreases the proportion of thermal energy converted from fossil fuels.

Monthly results for August are shown in Table 6-12.

Table 6-12: August Simulation Results

Plant Type	Plant Name	August Electricity Production [GWh]	August Net Efficiency	August Water Use [litres]	August Solar Themal Energy Use [GWh]	August Fossil Themal Energy Use [GWh]	August % of Solar Energy from Solar	August Load Factor	August Operating Hours
0	Water-Cooled (small)	3.68	38.2%	3.53E+07	9.64	0	100.0%	93.0%	347
1	Water-Cooled (design)	4.54	39.7%	1.85E+07	9.62	1.81	84.2%	89.4%	434
2	Water-Cooled (large)	4.65	40.8%	1.16E+07	9.63	1.78	84.4%	90.0%	434
3	Hybrid-Cooled (small)	4.32	37.6%	2.40E+07	9.62	1.87	83.7%	88.1%	434
4	Hybrid-Cooled (design)	4.57	39.6%	1.45E+07	9.62	1.92	83.4%	89.2%	434
5	Hybrid-Cooled (large)	4.64	40.7%	1.02E+07	9.62	1.78	84.4%	89.8%	434
6	Air-Cooled (small)	3.59	37.2%	N/A	9.65	0	100.0%	114.6%	540
7	Air-Cooled (design)	3.82	39.6%	N/A	9.65	0	100.0%	111.5%	527
8	Air-Cooled (large)	3.85	39.9%	N/A	9.65	0	100.0%	107.4%	505

As ambient temperatures increase, efficiency for the water-cooled cycle is further degraded due to increased parasitic fan power in the cooling tower. The hybrid-cooled cycles also face reduced efficiency. The air-cooled cycles have lower efficiency during August when compared to May, due to warmer ambient conditions, and greater air-cooler parasitic fan power requirements. Efficiency for the air-cooled SCO_2 Brayton during August is only slightly better than the annual efficiency for the Rankine cycle as modeled by Wagner (2008).

Water use increases during August when compared to May for all (non air-cooled) cycles. This increase is due to increased ambient temperatures, and is most notable in the ‘small’ water-cooled configuration.

Slightly decreased solar resource, when compared to May, results in only reduced electricity production for all cycles. The one exception to this is the ‘small’ water-cooled cycle, for which

the low cooling water temperatures it requires necessitate greatly increased fan use in the cooling tower as ambient temperatures increase. Electricity production for the 'small' water-cooled cycle is much lower in August, despite comparable solar resource in both months.

Now that annual and seasonal simulations of the three SCO_2 Brayton configurations have been performed at various sizes, Chapter 7 examines these results and draws conclusions on the viability of the configurations for CSP application. Chapter 7 also recommends areas for future work in quantifying the applicability of the SCO_2 Brayton cycle.

7 Conclusions and Recommendations for Future Work

The goal of this project was to assess the suitability of the supercritical carbon dioxide (SCO_2) Brayton cycle for power conversion applications in a Concentrating Solar Power plant. Annual and monthly simulations have been performed, with results discussed in Chapter 6 of this thesis. This chapter will summarize conclusions on the suitability of each SCO_2 Brayton configuration. Recommendations for future work will also be presented.

7.1 Conclusions

7.1.1 Water-Cooled SCO_2 Brayton

The water-cooled configuration provides excellent performance when compared against the Rankine cycle (Wagner, 2008). Annual power production (53.4 GWh) and efficiency (40.5%) for the design-sized Brayton exceed that of the Rankine cycle. Notably, annual cooling water use is reduced by ~27% in the 'large' heat exchanger case (the case where all heat exchangers are sized as 150% the size of the design heat exchangers) when compared against the 'design' heat exchanger case (the case with heat exchangers sized using the design process outlined in Chapter 5). This reduction in water use is due to the 'large' case being able to operate with significantly warmer cooling water, allowing a greater proportion of cooling to occur in the cooling tower without recourse to evaporation.

Monthly results show high efficiency (when compared against the Rankine cycle modeled by Wagner (2008)) and stable power production for the 'large' and 'design' cases during all times of year, though efficiency is best during the cool winter months when parasitic fan power requirements are lowest. The 'small' case performs well during the winter and spring, but performance degrades significantly during the summer due to its demand for very cold water from the cooling tower.

The water-cooled SCO_2 Brayton Cycle has the potential to perform consistently well all year, particularly with sufficiently large heat exchangers, which limit the cycle's sensitivity to high ambient dry bulb temperatures.

7.1.2 Hybrid-Cooled SCO_2 Brayton

Annual power production and efficiency for the hybrid-cooled configuration are similar to those of the water-cooled configuration. The additional air-cooler, however, substantially reduces annual water use (for details on water use reduction, see Chapter 6, sections 4 and 5). As with the water-cooled configuration, increasing the relative size of the heat exchangers significantly reduces water use by allowing the cycle to operate with higher cooling water temperatures.

Monthly results show that the hybrid-cooled configuration performs well during all seasons, though peak efficiency occurs during the cool winter months.

Though employing a hybrid-cooled configuration appears to be technically feasible method to reduce water consumption, the viability of this approach will have to be assessed by economic analysis. The performance of the 'large' water-cooled cycle, for example, is better than that of the 'design' hybrid-cooled cycle in terms of annual power production, efficiency, and water consumption. The 'large' water-cooled cycle also surpasses the 'design' hybrid-cooled cycle in all performance metrics during August. Economic analysis would answer the question of whether money would be best spent in purchasing an air-cooler, or by simply increasing the size of the other heat exchangers.

7.1.3 Air-Cooled SCO_2 Brayton

Annual efficiency for the air-cooled configuration, even for the 'small' air-cooled cycle, exceeds that of the Rankine cycle modeled by Wagner (2008).

Monthly results show the air-cooled cycle, predictably, is most efficient during the cool winter months, with efficiency dropping from 42.6% to 39.6% from January to August for the 'design' air-cooled plant as ambient dry bulb temperatures increase. This could be a significant drawback economically if electricity prices and therefore the incentive for peak performance are greatest during the summer months.

The air-cooled Brayton cycle is a technically feasible method of eliminating water consumption while maintaining reasonable efficiency during all periods of the year. Whether the air-cooled cycle is financially reasonable must be determined by economic analysis that balances degraded efficiency and power production against eliminated water use.

7.2 Recommendations for Future Work

7.2.1 Detailed Turbomachinery

The current model uses a simple isentropic efficiency to model the turbomachinery, and fixes the volumetric flow rate of CO_2 through the compressor to approximate a fixed compressor size. A more complex turbomachinery model could allow for better characterization of performance, particularly at off-design conditions.

7.2.2 Fixed Power Optimization

The analysis conducted in this thesis was based on a cycle design approach which sought to choose plant parameters to provide good power and good efficiency. A more meaningful approach to plant optimization might be to fix the net power of the plant, and vary plant parameters to optimize efficiency at this design power. This approach, when combined with economic constraints, would provide a more meaningful performance comparison between different SCO_2 Brayton configurations.

7.2.3 Economic Analysis

While this thesis established the theoretical possibility of coupling a SCO_2 Brayton power block with a CSP plant, economic analysis would be necessary to establish the real world viability of such a system. Cost information for heat exchangers, heliostats, central receivers, cooling towers, water, and salt storage would be necessary to accurately assess the plant. Seasonal variations in electricity prices would also be useful in assessing how seasonal performance variations impact the potentially profitability of such a venture.

Integrating economic analysis into the SCO_2 Brayton model would also allow for more meaningful plant optimization, with the objective being the lowest possible electricity cost.

References

Cox, Timothy L. 2009. Preliminary Results of Heat Transfer Experiments with Supercritical Carbon in the Small System Test Loop. Proceedings of $SCCO_2$ Power Cycle Symposium

Dostal, Vaclav. 2004. A Supercritical Carbon Dioxide Cycle for Next Generation Nuclear Reactors. PhD Thesis, Massachusetts Institute of Technology.

Dostal, Vaclav et al. 2006. High Performance Supercritical Carbon Dioxide Cycle for Next-Generation Nuclear Reactors. Nuclear Technology, Volume 154, June 2006.

Dostal, Vaclav. Kulhanek, Martin. 2009. Research on Supercritical Carbon Dioxide Cycles in The Czech Republic. Proceedings of $SCCO_2$ Power Cycle Symposium 2009.

Feierabend, Lukas. 2009. Thermal Model Development and Simulation of Cavity-Type Solar Central Receiver Systems. UW-Madison MS Thesis.

Hoang, Hiep T. 2009. Thermodynamic Study of a Supercritical CO_2 Brayton Cycle Concept. Proceedings of $SCCO_2$ Power Cycle Symposium 2009.

Kao, Shih-Ping et al. 2009. Dynamic Simulation and Control of a Supercritical CO_2 Power Conversion System for Small Light Water Reactor Applications. Proceedings of $SCCO_2$ Power Cycle Symposium 2009.

Kays, W.M., & London, A.L. 1984. Compact Heat Exchangers, 3rd ed., McGraw Hill, New York,

Klein, Nellis. 2009. Heat Transfer. Cambridge University Press

Klein, S.A. 2010, EES – Engineering Equation Solver. F-Chart Software. http://www.fchart.com/

Klein, S.A. 2010, TRNSYS – Transient Energy System Simulation Tool. TESS. http://www.trnsys.com/

Kruizenga, Alan M. 2010. Heat Transfer and Pressure Drop Measurements in Prototypic Heat Exchangers for the Supercritical Carbon Dioxide Brayton Power Cycles. UW-Madison MS Thesis.

Liao, S.M. Zhao, T.S. 2002. An Experimental Investigation of Convection Heat Transfer to Supercritical Carbon Dioxide in Miniature Tubes.

Southall, David. 2009. Diffusion Bonding in Compact Heat Exchangers. Proceedings of $SCCO_2$ Power Cycle Symposium 2009.

Turchi, Craig S. 2009. Supercritical CO_2 for Application in Concentrating Solar Power Systems. Proceedings of $SCCO_2$ Power Cycle Symposium 2009.

Wagner, Michael. 2008. Simulation and Predictive Performance Modeling of Utility-Scale Central Receiver System Power Plants. UW-Madison MS Thesis.

Appendix A: Simplified Brayton w/Regeneneration EES Code

"these groups of contants are laid out to allow for different numbers of nodes. By commenting out all but one group we can select either 2, 5, 10, or 20 nodes."

```
$Constant N#=20
$Constant N1#=21                                    "N+1"
$Constant N2#=22                                    "N+2"
$Constant N3#=23                                    "N+3"
$Constant N4#=24                                    "N+4"
$Constant N22#=42                                   "2*N+2"
$Constant N23#=43                                   "2*N+3"
$Constant N24#=44                                   "2*N+4"
$Constant N34#=64                                   "3*N+4"
$Constant N35#=65                                   "3*N+5"
$Constant N45#=85                                   "4*N+5"
$Constant N46#=86                                   "4*N+6"

{$Constant N#=10
$Constant N1#=11                                    "N+1"
$Constant N2#=12                                    "N+2"
$Constant N3#=13                                    "N+3"
$Constant N4#=14                                    "N+4"
$Constant N22#=22                                   "2*N+2"
$Constant N23#=23                                   "2*N+3"
$Constant N24#=24                                   "2*N+4"
$Constant N34#=34                                   "3*N+4"
$Constant N35#=35                                   "3*N+5"
$Constant N45#=45                                   "4*N+5"
$Constant N46#=46                                   "4*N+6"}

{$Constant N#=5
$Constant N1#=6                                     "N+1"
$Constant N2#=7                                     "N+2"
$Constant N3#=8                                     "N+3"
$Constant N4#=9                                     "N+4"
$Constant N22#=12                                   "2*N+2"
$Constant N23#=13                                   "2*N+3"
$Constant N24#=14                                   "2*N+4"
$Constant N34#=19                                   "3*N+4"
$Constant N35#=20                                   "3*N+5"
$Constant N45#=25                                   "4*N+5"
$Constant N46#=26                                   "4*N+6"}


$UnitSystem SI MASS RAD PA K J
$Tabstops 0.2 0.4 0.6 3.5 in
{$Constant N#=2
$Constant N1#=3                                     "N+1"
$Constant N2#=4                                     "N+2"
$Constant N3#=5                                     "N+3"
$Constant N4#=6                                     "N+4"
$Constant N22#=6                                    "2*N+2"
$Constant N23#=7                                    "2*N+3"
$Constant N24#=8                                    "2*N+4"
$Constant N34#=10                                   "3*N+4"
$Constant N35#=11                                   "3*N+5"
$Constant N45#=13                                   "4*N+5"
$Constant N46#=14                                   "4*N+6"}
{
$Constant N#=3
$Constant N1#=4                                     "N+1"
$Constant N2#=5                                     "N+2"
$Constant N3#=6                                     "N+3"
$Constant N4#=7                                     "N+4"
$Constant N22#=8                                    "2*N+2"
```

```
$Constant N23#=9                                                    "2*N+3"
$Constant N24#=10                                                   "2*N+4"
$Constant N34#=13                                                   "3*N+4"
$Constant N35#=14                                                   "3*N+5"
$Constant N45#=17                                                   "4*N+5"
$Constant N46#=18                                                   "4*N+6"}


Subprogram Compressor(eff,WF$,T_in,P_in,P_out:T_out,W)

    h_in=enthalpy(WF$,T=T_in,P=P_in)                                "calculate enthalpy pre-compressor from
temp/pressure"
    s_in=entropy(WF$,T=T_in,P=P_in)                                 "calculate entropy pre-compressor from
temp/pressure"
    s_outs=s_in "s2s represents the entropy that would exist after the compressor if it were operating isentropically"
    h_outs=enthalpy(WF$,s=s_outs,P=P_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
    Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
    W=Ws/eff "the amount of work required - taking into account efficiency"
    W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
    T_out=temperature(WF$,P=P_out,h=h_out) "the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"


end

Subprogram Turbine(eff,WF$,T_in,P_in,P_out:T_out,W)

    h_in=enthalpy(WF$,T=T_in,P=P_in)                                "calculate enthalpy pre-compressor from
temp/pressure"
    s_in=entropy(WF$,T=T_in,P=P_in)                                 "calculate entropy pre-compressor from
temp/pressure"
    s_outs=s_in "s4s represents the entropy that would exist after the compressor if it were operating isentropically"
    h_outs=enthalpy(WF$,s=s_outs,P=p_out)                           "the enthalpy after the compressor, if the
compressor were purely isentropic"
    Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
    W=Ws*eff "the amount of work produced - taking into account efficiency"
    W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
    T_out=temperature(WF$,P=P_out,h=h_out) "the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"

end

module
PHX(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,eff_overall,eff_te
st[1..N#],x[1..N1#],UA_total,delta_T_HC[1])

                                                                    "Inputs"
W=35 [cm]*convert(cm,m)                                             "width of heat exchanger"
N_ch=100 [-]  "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                         "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                         "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                         "thickness of plate"


duplicate i=1,N1#
    P_H[i]=P_H_in                                                   "pressure distribution (constant)"
    P_C[i]=P_C_in
end

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                               "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                             "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                      "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                               "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                        "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                        "temperature of cold outlet fluid"

N=N#                                                                "number of sub-heat exchangers"
duplicate i=1,N
```

```
      q_dot[i]=i*q_dot/N                                                    "total heat transfer rate"
end


                                                                            "Obtain temperature distribution"
T_H[1]=T_H_in                                                               "hot-side inlet temperature"
T_C[1]=T_C_out                                                              "cold-side outlet temperature"
i_H[1]=i_H_in                                                               "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                              "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                                     "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                              "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                                      "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                              "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                            "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                          "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                      "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                      "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                                         "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"

      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]                 "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
      eff_test[i]=q_dot_node[i]/q_dot_max[i]                                "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))           "effectiveness of sub-heat exchanger"

      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU') "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                               "conductance in sub-heat exchanger"
end

"determine length of each sub-heat exchanger"
x[1]=0.00 [m] "starting position of 1st sub-heat exchanger"

duplicate i=1,N
      call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
      h_H[i], h_H_H[i], dPHdx[i])  "hot-side local heat transfer coefficient"
      call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
      h_C[i], h_C_H[i], dPCdx[i]) "cold-side local heat transfer coefficient"
      k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)  "metal conductivity at local average temperature"
      DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W) "length of sub-heat exchanger"
      x[i+1]=x[i]+DELTAx[i]
end

UA_total=sum(UA[i],i=1,N)

duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end
```

```
"Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))

end

module
Regenerator(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,eff_over
all,eff_test[1..N#],x[1..N1#],UA_total,delta_T_HC[1])

W=35 [cm]*convert(cm,m)                                          "width of heat exchanger"
N_ch=100 [-]  "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                      "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                      "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                      "thickness of plate"


duplicate i=1,N1#
    P_H[i]=P_H_in                                                "pressure distribution (constant)"
    P_C[i]=P_C_in
end

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                            "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                          "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                   "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                            "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                     "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                     "temperature of cold outlet fluid"

N=N#                                                             "number of sub-heat exchangers"
duplicate i=1,N
    q_dot[i]=i*q_dot/N                                           "total heat transfer rate"
end


                                                                 "Obtain temperature distribution"
T_H[1]=T_H_in                                                    "hot-side inlet temperature"
T_C[1]=T_C_out                                                   "cold-side outlet temperature"
i_H[1]=i_H_in                                                    "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                   "cold-side outlet enthalpy"

duplicate i=2,(N+1)
    i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                            "energy balance on hot-side of each sub-heat
exchanger"
    T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                     "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
    i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                            "energy balance on cold-side of each sub-heat
exchanger"
    T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                     "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                 "Apply effectiveness-NTU solution"
duplicate i=1,N
    delta_T_H[i]=T_H[i]-T_H[i+1]  "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
    delta_T_C[i]=T_C[i]-T_C[i+1]
    delta_i_H[i]=(i_H[i]-i_H[i+1])
    delta_i_C[i]=(i_C[i]-i_C[i+1])
    C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])             "hot-side capacitance rate"
    C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])             "cold-side capacitance rate"
end
```

```
duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1]"by breaking the delta T/delta enthalpy values into their own variables I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"

    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i] "this is the maximum amount of heat that can be transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]                    "this ratio of the maximum heat possible for transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))    "effectiveness of sub-heat exchanger"

    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU') "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                   "conductance in sub-heat exchanger"
end

"determine length of each sub-heat exchanger"
x[1]=0.00 [m] "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                               "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                               "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                  "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)  "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

UA_total=sum(UA[i],i=1,N)

duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

                                                              "Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))   "This definition of effectiveness is taken from dostal's thesis page 74"

end

module
Precooler(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,eff_overall,eff_test[1..N#],x[1..N1#],UA_total,delta_T_HC[1])

                                                              "Inputs"
W=35 [cm]*convert(cm,m)                                       "width of heat exchanger"
N_ch=100 [-]                                                  "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                   "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                   "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                   "thickness of plate"


duplicate i=1,N1#
    P_H[i]=P_H_in                                             "pressure distribution (constant)"
    P_C[i]=P_C_in
end

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                         "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                       "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                         "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                  "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                  "temperature of cold outlet fluid"

N=N#                                                          "number of sub-heat exchangers"
duplicate i=1,N
    q_dot[i]=i*q_dot/N                                        "total heat transfer rate"
end
```

```
                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in                                                       "hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
    i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                               "energy balance on hot-side of each sub-heat
exchanger"
    T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                        "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
    i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                               "energy balance on cold-side of each sub-heat
exchanger"
    T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                        "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
    delta_T_H[i]=T_H[i]-T_H[i+1]                                    "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
    delta_T_C[i]=T_C[i]-T_C[i+1]
    delta_i_H[i]=(i_H[i]-i_H[i+1])
    delta_i_C[i]=(i_C[i]-i_C[i+1])
    C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                "hot-side capacitance rate"
    C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                "cold-side capacitance rate"
end



duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1]                                   "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N#  "this is the heat transfer through the node of the heat exchanger"

    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]           "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]                          "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))     "effectiveness of sub-heat exchanger"

    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                         "conductance in sub-heat exchanger"
end

                                                                    "determine length of each sub-heat exchanger"
x[1]=0.00 [m]                                                       "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                                     "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                                     "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                        "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)       "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

UA_total=sum(UA[i],i=1,N)

duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

                                                                    "Effectiveness calculation"
```

```
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))

end


                                                                    "Overall Program Control"

                                                                    "Constants"
L_H=10        "length of heating HX"
L_C=10        "length of cooling HX"
L_R=10        "length of regenerator HX"
C$='air_ha'   "cooling fluid"
H$='air_ha'   "heating fluid"
WF$='carbondioxide'                                                 "working fluid"


$ifnot ParametricTable
P_high=25000000 [pa]                                                "high pressure point"
{P_low=5000000 [pa]}                                                "low pressure point"
P_ratio=3.2
$endif
P_H=1000000[Pa]                                                     "hot fluid pressure"
P_C=1000000[Pa]                                                     "cold fluid pressure"
P_ratio=P_high/P_low

"in this case the inlet temp of hot fluid is commented out and the temp at the end of the primary heat exchanger is specified"
T_H_in   =900 [k]                                                   "heating fluid inlet temp"
T_C_in=295 [k]                                                      "cooling fluid inlet temp"
m_dot_wf=1[kg/s]                                                    "mass flow rate of working fluid"
m_dot_h=30 [kg/s]                                                   "mass flow rate of heating fluid"
m_dot_c=30 [kg/s]                                                   "mass flow rate of cooling fluid"


T2=T_WF[N46#]

T5=T_WF[N23#]

{eff_r=.95}                                                         "regenerator effectiveness"
turbine_eff=.9                                                      "turbine efficiency"
comp_eff=.89"compressor efficiency"

UA_regenerator =5000                                                "set regenerator UA"
UA_precooler=5000[W/K]
UA_PHX=5000 [W/K]

{UA_total = 15000 [W/K]}
UA_Total = UA_regenerator+UA_precooler+UA_PHX

Call
PHX(L_H,H$,T_H_in,T_H_out,P_H,WF$,m_dot_h,T_WF[N1#],p_high,m_dot_wf:T_H[1..N1#],T_WF[N2#..N22#],q_dot,eff_h,eff_node_h[1..N#],x_h[1..N1#],UA_PHX,T_PHX_Pinch) "this HX represents heat transfer from heat source to WF after regenerator"

Call
Regenerator(L_R,WF$,T5,T6,P_low,WF$,m_dot_wf,T2,p_high,m_dot_wf:T_WF[N24#..N34#],T_WF[1..N1#],q_dot_reg,eff_r,eff_node_r[1..N#],x_r[1..N1#],UA_regenerator,T_RG_Pinch) "this heat exchanger represents the regenerator"


Call Turbine (turbine_eff,WF$,T_WF[N22#],P_high,P_low:T_WF[N23#],W_turbine)"this turbine extracts energy from the working fluid"

Call
Precooler(L_C,WF$,T_WF[N34#],T1,P_low,C$,m_dot_wf,T_C_in,P_C,m_dot_c:T_WF[N35#..N45#],T_C[1..N1#],q_dot_cooling,eff_c,eff_node_c[1..N#],x_c[1..N1#],UA_precooler,T_PC_Pinch) "this heat exchanger involves the working fluid after the turbine and the cooling fluid from the cooling tower or whatnot"

Call Compressor(comp_eff,WF$,T_WF[N45#],P_low,P_high:T_WF[N46#],W_comp) "the compressor takes the fluid from the heat rejection heat exchanger to the regenerator"

work_net=m_dot_wf*(W_turbine+W_comp)                                "net work done by the cycle"
```

```
efficiency=work_net/q_dot                                   "this section of code records the efficiency of the cycle in terms of turbine/compressor work and heat transfer to the system in the primary heat exchanger"


efficiency_test=1-(q_dot_cooling/q_dot)

comp_inlet_density=density(WF$,T=T_WF[N45#],P=P_low)        {working fluid density at compressor inlet}
flowrate_compressor_inlet = m_dot_wf/comp_inlet_density     "volumetric flowrate of working fluid at compressor inlet"

duplicate i=1,N22# {this section sets up the array recording the entropy of the working fluid at each point}
    s[i]=entropy(WF$,T=T_WF[i],P=P_high)
    h[i]=enthalpy(WF$,T=T_WF[i],P=P_high)
end

duplicate i=N23#,N45#
    s[i]=entropy(WF$,T=T_WF[i],P=P_low)
h[i]=enthalpy(WF$,T=T_WF[i],P=P_high)
end

s[N46#]=entropy(WF$,T=T_WF[N46#],P=P_high)
h[N46#]=enthalpy(WF$,T=T_WF[N46#],P=P_high)


duplicate i=1,N1#                                           {this segment sets up heat exchanger node distances for plotting temperature v. node data for hot and cold sides of the heat exchanger}
    x_r_reverse[-i+N2#]=x_r[i]                              {this transposes the lengths of the nodes into reverse order in the right location in the arrays for graphing purposes}
    T_R_H[i]=T_WF[i+N23#] {this is the temperature of the fluid going through the hot side of the regenerator}
    T_R_C[i]=T_WF[i] {this is the temperature of the fluid going through the cold side of the regenerator}
    count[i]=i
end
```

Appendix B: Precompression EES Code

```
$UnitSystem SI MASS RAD PA K J
$Tabstops 0.2 0.4 0.6 3.5 in
"these groups of contants are laid out to allow for different numbers of nodes. By commenting out all but one group we can select either 2, 5, 10, or 20 nodes."

{$Constant N#=20
$Constant N1#=21                    "N+1"
$Constant N2#=22                    "N+2"
$Constant N3#=23                    "N+3"
$Constant N4#=24                    "N+4"
$Constant N22#=42                   "2*N+2"
$Constant N23#=43                   "2*N+3"
$Constant N24#=44                   "2*N+4"
$Constant N34#=64                   "3*N+4"
$Constant N35#=65                   "3*N+5"
$Constant N45#=85                   "4*N+5"
$Constant N46#=86                   "4*N+6"}

{$Constant N#=3
$Constant N1#=4                     "N+1"
$Constant N2#=5                     "N+2"
$Constant N3#=6                     "N+3"
$Constant N4#=7                     "N+4"
$Constant N22#=8                    "2*N+2"
$Constant N23#=9                    "2*N+3"
$Constant N24#=10                   "2*N+4"
$Constant N33#=12                   "3*N+3"
$Constant N34#=13                   "3*N+4"
$Constant N35#=14                   "3*N+5"
$Constant N44#=16                   "4*N+4"
$Constant N45#=17                   "4*N+5"
$Constant N46#=18                   "4*N+6"
$Constant N55#=20                   "5*N+5"
$Constant N56#=21                   "5*N+6"
$Constant N66#=24                   "6*N+6"
$Constant N67#=25                   "6*N+7"}


$Constant N#=10
$Constant N1#=11                    "N+1"
$Constant N2#=12                    "N+2"
$Constant N3#=13                    "N+3"
$Constant N4#=14                    "N+4"
$Constant N22#=22                   "2*N+2"
$Constant N23#=23                   "2*N+3"
$Constant N24#=24                   "2*N+4"
$Constant N33#=33                   "3*N+3"
$Constant N34#=34                   "3*N+4"
$Constant N35#=35                   "3*N+5"
$Constant N44#=44                   "4*N+4"
$Constant N45#=45                   "4*N+5"
$Constant N46#=46                   "4*N+6"
$Constant N55#=55                   "5*N+5"
$Constant N56#=56                   "5*N+6"
$Constant N66#=66                   "6*N+6"
$Constant N67#=67                   "6*N+7"

{$Constant N#=5
$Constant N1#=6                     "N+1"
$Constant N2#=7                     "N+2"
$Constant N3#=8                     "N+3"
$Constant N4#=9                     "N+4"
$Constant N22#=12                   "2*N+2"
$Constant N23#=13                   "2*N+3"
```

```
$Constant N24#=14                                                    "2*N+4"
$Constant N34#=19                                                    "3*N+4"
$Constant N35#=20                                                    "3*N+5"
$Constant N45#=25                                                    "4*N+5"
$Constant N46#=26                                                    "4*N+6"}

{$Constant N#=2
$Constant N1#=3                                                      "N+1"
$Constant N2#=4                                                      "N+2"
$Constant N3#=5                                                      "N+3"
$Constant N4#=6                                                      "N+4"
$Constant N22#=6                                                     "2*N+2"
$Constant N23#=7                                                     "2*N+3"
$Constant N24#=8                                                     "2*N+4"
$Constant N34#=10                                                    "3*N+4"
$Constant N35#=11                                                    "3*N+5"
$Constant N45#=13                                                    "4*N+5"
$Constant N46#=14                                                    "4*N+6"}


Subprogram Precompressor(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                               "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                                "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in "s2s represents the entropy that would exist after the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=P_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
      Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
      W=Ws/eff                                                       "the amount of work required - taking into
account efficiency"
      W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)                         "the temperature after the compressor defined
by the actual enthalpy and pressure after the compressor"


end

Subprogram Compressor(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                               "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                                "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in "s2s represents the entropy that would exist after the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=P_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
      Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
      W=Ws/eff                                                       "the amount of work required - taking into
account efficiency"
      W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)                         "the temperature after the compressor defined
by the actual enthalpy and pressure after the compressor"


end

Subprogram Turbine(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                               "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                                "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in "s4s represents the entropy that would exist after the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=p_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
      Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
      W=Ws*eff "the amount of work produced - taking into account efficiency"
      W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out) "the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"
```

```
end

Module
HTR(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total)

                                                                    "Inputs"
W=35 [cm]*convert(cm,m)                                             "width of heat exchanger"
N_ch=100 [-] "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                         "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                         "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                         "thickness of plate"


duplicate i=1,N1#
      P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
      P_C[i]=P_C_in
end

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                               "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                             "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                      "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                               "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                        "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                        "temperature of cold outlet fluid"

N=N#                                                                "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                            "total heat transfer rate"
end

                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in                                                       "hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                             "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                       "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                             "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                      "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])              "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])              "cold-side capacitance rate"
end


duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from
going to zero and messing up solving"
```

```
    q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]                    "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]                                   "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))              "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')          "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                                  "conductance in sub-heat exchanger"
end

                                                                             "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                                   "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                                              "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                                              "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                                 "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)                "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

UA_total=sum(UA[i],i=1,N)

err=abs(x[N+1]-L)/L                                                          "objective function"


duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"

eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))

end

Module
LTR(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total)

                                                                             "Inputs"
W=35 [cm]*convert(cm,m)                                                      "width of heat exchanger"
N_ch=100 [-]                                                                 "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                                  "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                                  "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                                  "thickness of plate"


duplicate i=1,N1#
    P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
    P_C[i]=P_C_in
end

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                                        "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                                      "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                               "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                                        "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                                 "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                                 "temperature of cold outlet fluid"

N=N#                                                                         "number of sub-heat exchangers"
duplicate i=1,N
    q_dot[i]=i*q_dot/N                                                       "total heat transfer rate"
end

                                                                             "Obtain temperature distribution"
```

```
T_H[1]=T_H_in                                                    "hot-side inlet temperature"
T_C[1]=T_C_out                                                   "cold-side outlet temperature"
i_H[1]=i_H_in                                                    "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                   "cold-side outlet enthalpy"

duplicate i=2,(N+1)
    i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                            "energy balance on hot-side of each sub-heat
exchanger"
    T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                     "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
    i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                            "energy balance on cold-side of each sub-heat
exchanger"
    T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                     "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                 "Apply effectiveness-NTU solution"
duplicate i=1,N
    delta_T_H[i]=T_H[i]-T_H[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
    delta_T_C[i]=T_C[i]-T_C[i+1]
    delta_i_H[i]=(i_H[i]-i_H[i+1])
    delta_i_C[i]=(i_C[i]-i_C[i+1])
    C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])             "hot-side capacitance rate"
    C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])             "cold-side capacitance rate"
end



duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from
going to zero and messing up solving"
    q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]        "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]                       "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))  "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                      "conductance in sub-heat exchanger"
end

                                                                 "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                       "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                                  "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                                  "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                     "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)    "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

UA_total=sum(UA[i],i=1,N)

err=abs(x[N+1]-L)/L                                              "objective function"


duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
```

```
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))
end

Module
Precooler(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_over
all,eff_test[1..N#],x[1..N1#],UA_total)

                                                                    "Inputs"
W=35 [cm]*convert(cm,m)                                             "width of heat exchanger"
N_ch=100 [-]                                                        "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                         "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                         "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                         "thickness of plate"


duplicate i=1,N1#
      P_H[i]=P_H_in     "this represents the pressure drop at the inlet of the heat exchanger"
      P_C[i]=P_C_in
end

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                               "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                             "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                      "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                               "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                        "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                        "temperature of cold outlet fluid"

N=N#                                                                "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                            "total heat transfer rate"
end

                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in                                                       "hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                             "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                       "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                             "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                      "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])              "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])              "cold-side capacitance rate"
end


duplicate i=1,N
```

```
    delta_T_HC[i]=T_H[i]-T_C[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from
going to zero and messing up solving"
    q_dot_node[i]=q_dot/N#   "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]                    "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]                                   "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))              "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')          "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                                  "conductance in sub-heat exchanger"
end

                                                                             "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                                   "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                                              "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                                              "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                                 "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)                "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

UA_total=sum(UA[i],i=1,N)

err=abs(x[N+1]-L)/L                                                          "objective function"


duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

                                                                             "Effectiveness calculation"

eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))
end

Module
PHX(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total)

                                                                             "Inputs"
W=35 [cm]*convert(cm,m)                                                      "width of heat exchanger"
N_ch=100 [-]                                                                 "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                                  "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                                  "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                                  "thickness of plate"


duplicate i=1,N1#
    P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
    P_C[i]=P_C_in
end

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                                        "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                                      "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                               "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                                        "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                                 "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                                 "temperature of cold outlet fluid"

N=N#                                                                         "number of sub-heat exchangers"
duplicate i=1,N
    q_dot[i]=i*q_dot/N                                                       "total heat transfer rate"
end
```

```
                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in                                                       "hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
    i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                               "energy balance on hot-side of each sub-heat
exchanger"
    T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                        "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
    i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                               "energy balance on cold-side of each sub-heat
exchanger"
    T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                        "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
    delta_T_H[i]=T_H[i]-T_H[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
    delta_T_C[i]=T_C[i]-T_C[i+1]
    delta_i_H[i]=(i_H[i]-i_H[i+1])
    delta_i_C[i]=(i_C[i]-i_C[i+1])
    C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                 "hot-side capacitance rate"
    C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                 "cold-side capacitance rate"
end



duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1]                                   "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]           "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]                          "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))     "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                         "conductance in sub-heat exchanger"
end

                                                                    "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                          "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                                     "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                                     "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                        "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)       "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

UA_total=sum(UA[i],i=1,N)

err=abs(x[N+1]-L)/L                                                 "objective function"


duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
```

```
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))
end


"Overall Program Control"

                                                                    "Constants"
L_H=10        "length of heating HX"
L_C=10        "length of cooling HX"
L_R=10        "length of regenerator HX"
C$='air_ha'   "cooling fluid"
H$='air_ha'   "heating fluid"
WF$='carbondioxide'                                                 "working fluid"


$ifnot ParametricTable
P_high=25000000 [pa]                                                "high pressure point"
P_ratio = 3                                                         "pressure ratio"
rpr=.

$endif
P_H=1000000[Pa]                                                     "hot fluid pressure"
P_C=1000000[Pa]                                                     "cold fluid pressure"
P_ratio=P_high/P_low

T_H_in   =900 [k]                                                   "heating fluid inlet temp"
T_C_in=295 [k]                                                      "cooling fluid inlet temp"
m_dot_wf=1[kg/s]                                                    "mass flow rate of working fluid"
m_dot_h=20 [kg/s]                                                   "mass flow rate of heating fluid"
m_dot_c=20 [kg/s]                                                   "mass flow rate of cooling fluid"

turbine_eff=.9 "turbine efficiency"
comp_eff=.89"compressor efficiency"


{UA_HTR=5000 [w/k]                                                  "UA of Heat Exchangers"
UA_LTR=5000 [w/k]
UA_PHX=3000 [w/k]
UA_Precooler=5000 [w/k]}

UA_Total = UA_HTR+UA_LTR+UA_PHX+UA_Precooler
UA_total = 15000 [w/k]

T2=T_WF[N1#]
T3=T_WF[N22#]
T4=T_WF[N33#]
{T4=823 [k]
T9=305 [k]}

P_ratio_RC=(P_high/P_recomp)

rpr=(p_ratio_rc-1)/(p_ratio-1)  "rpr as defined on page 5 of dostale [2009]"

f*(t5-t1)=(t6-t1)                                                   "temperature factor 'f' is defined on page 5 of
dostal [2009]"
{f=0.2}



Call
PHX(L_H,H$,T_H_in,T_H_out,P_H,WF$,m_dot_h,T3,p_high,m_dot_wf:T_H[1..N1#],T_WF[N23#..N33#],q_dot_h,err_h,eff_h,eff_no
de_h[1..N#],x_h[1..N1#],UA_PHX)                                     "primary heat exchanger"
Call
LTR(L_R,WF$,T7,T8,P_low,WF$,m_dot_wf,T1,p_high,m_dot_wf:T_WF[N45#..N55#],T_WF[1..N1#],q_dot_r_l,err_r_l,eff_r_l,eff_nod
e_r_l[1..N#],x_r_l[1..N1#],UA_LTR)                                  "low temp regenerator"
Call
HTR(L_R,WF$,T5,T6,P_low,WF$,m_dot_wf,T2,p_high,m_dot_wf:T_WF[N34#..N44#],T_WF[N2#..N22#],q_dot_r_h,err_r_h,eff_r_h,
eff_node_r_h[1..N#],x_r_h[1..N1#],UA_HTR)"high temp regen"
```

```
Call
Precooler(L_R,WF$,T8,T9,P_recomp,WF$,m_dot_wf,T_C_in,P_C,m_dot_c:T_WF[N56#..N66#],T_C[1..N1#],q_dot_c,err_c,eff_c,eff
_node_c[1..N#],x_c[1..N1#],UA_precooler)                    "precooler"
Call Precompressor(comp_eff,WF$,T6,P_low,P_recomp:T7,W_recomp)          "recompressing compressor"
Call Compressor(comp_eff,WF$,T9,P_recomp,P_high:T1,W_compressor)        "primary compressor"
Call Turbine(turbine_eff,WF$,T4,P_high,P_low:T5,W_turbine)

work_comp_total=w_recomp*m_dot_wf+W_compressor*m_dot_wf "the total work done by the compressors, weighted by mass flow
rate"
work_turbine_total=W_turbine*m_dot_wf  "total work done by the turbine, weighted by mass flow rate"

efficiency_cycle= (work_turbine_total+work_comp_total)/q_dot_h "this defines the efficiency of the cycle as a whole"

work_net = work_turbine_total+work_comp_total

duplicate i=1,N33#
      s[i]=entropy(WF$,T=T_WF[i],P=P_high)
end

duplicate i=N34#,N44#
      s[i]=entropy(WF$,T=T_WF[i],P=P_low)
end

duplicate i=N45#,N55#
      s[i]=entropy(WF$,T=T_WF[i],P=P_recomp)
end

duplicate i=N56#,N66#
      s[i]=entropy(WF$,T=T_WF[i],P=P_recomp)
end

s[N67#]=entropy(WF$,P=P_high,T=T_WF[1])                     "closing the loop for graphing purposes"
T_WF[N67#]=T_WF[1]

duplicate i=1,N1#                                           "for plotting the temp distribution in LTR
regenerator"
      T_LTR[12-i] = T_WF[44+i]
end

duplicate i=1,N1#
      T_delta_LTR[i]=T_LTR[i]-T_WF[i]
end
```

Appendix C: Recompression EES Code

```
$UnitSystem SI MASS RAD PA K J
$Tabstops 0.2 0.4 0.6 3.5 in

"these groups of contants are laid out to allow for different numbers of nodes. By commenting out all but one group we can select either 2, 5, 10, or 20 nodes."

{$Constant N#=20
$Constant N1#=21                    "N+1"
$Constant N2#=22                    "N+2"
$Constant N3#=23                    "N+3"
$Constant N4#=24                    "N+4"
$Constant N22#=42                   "2*N+2"
$Constant N23#=43                   "2*N+3"
$Constant N24#=44                   "2*N+4"
$Constant N34#=64                   "3*N+4"
$Constant N35#=65                   "3*N+5"
$Constant N45#=85                   "4*N+5"
$Constant N46#=86                   "4*N+6"}

$Constant N#=10
$Constant N1#=11                    "N+1"
$Constant N2#=12                    "N+2"
$Constant N3#=13                    "N+3"
$Constant N4#=14                    "N+4"
$Constant N22#=22                   "2*N+2"
$Constant N23#=23                   "2*N+3"
$Constant N24#=24                   "2*N+4"
$Constant N33#=33                   "3*N+3"
$Constant N34#=34                   "3*N+4"
$Constant N35#=35                   "3*N+5"
$Constant N44#=44                   "4*N+4"
$Constant N45#=45                   "4*N+5"
$Constant N46#=46                   "4*N+6"
$Constant N55#=55                   "5*N+5"
$Constant N56#=56                   "5*N+5"
$Constant N65#=65                   "6*N+5"
$Constant N66#=66                   "6*N+6"


{$Constant N#=5
$Constant N1#=6                     "N+1"
$Constant N2#=7                     "N+2"
$Constant N3#=8                     "N+3"
$Constant N4#=9                     "N+4"
$Constant N22#=12                   "2*N+2"
$Constant N23#=13                   "2*N+3"
$Constant N24#=14                   "2*N+4"
$Constant N34#=19                   "3*N+4"
$Constant N35#=20                   "3*N+5"
$Constant N45#=25                   "4*N+5"
$Constant N46#=26                   "4*N+6"}


{$Constant N#=2
$Constant N1#=3                     "N+1"
$Constant N2#=4                     "N+2"
$Constant N3#=5                     "N+3"
$Constant N4#=6                     "N+4"
$Constant N22#=6                    "2*N+2"
$Constant N23#=7                    "2*N+3"
$Constant N24#=8                    "2*N+4"
$Constant N34#=10                   "3*N+4"
```

```
$Constant N35#=11                                                    "3*N+5"
$Constant N45#=13                                                    "4*N+5"
$Constant N46#=14                                                    "4*N+6"}


Subprogram Compressor(eff,WF$,T_in,P_in,P_out:T_out,W)

    h_in=enthalpy(WF$,T=T_in,P=P_in)                                 "calculate enthalpy pre-compressor from
temp/pressure"
    s_in=entropy(WF$,T=T_in,P=P_in)                                  "calculate entropy pre-compressor from
temp/pressure"
    s_outs=s_in "s2s represents the entropy that would exist after the compressor if it were operating isentropically"
    h_outs=enthalpy(WF$,s=s_outs,P=P_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
    Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
    W=Ws/eff "the amount of work required - taking into account efficiency"
    W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
    T_out=temperature(WF$,P=P_out,h=h_out) "the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"


end

Subprogram Turbine(eff,WF$,T_in,P_in,P_out:T_out,W)

    h_in=enthalpy(WF$,T=T_in,P=P_in)                                 "calculate enthalpy pre-compressor from
temp/pressure"
    s_in=entropy(WF$,T=T_in,P=P_in)                                  "calculate entropy pre-compressor from
temp/pressure"
    s_outs=s_in "s4s represents the entropy that would exist after the compressor if it were operating isentropically"
    h_outs=enthalpy(WF$,s=s_outs,P=p_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
    Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
    W=Ws*eff "the amount of work produced - taking into account efficiency"
    W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
    T_out=temperature(WF$,P=P_out,h=h_out) "the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"

end
Module
PHX(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total,DELTA_T_HC[1])

"Inputs"
W=35 [cm]*convert(cm,m)                                              "width of heat exchanger"
N_ch=100 [-]                                                         "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                          "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                          "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                          "thickness of plate"


duplicate i=1,N1#
    P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
    P_C[i]=P_C_in
end

    "this calculation takes into account the outlet pressure drop"
    "this calculation isn't using variable density values yet - this needs to be added"

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                                "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                              "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                       "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                                "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                         "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                         "temperature of cold outlet fluid"

N=N#                                                                 "number of sub-heat exchangers"
duplicate i=1,N
    q_dot[i]=i*q_dot/N                                               "total heat transfer rate"
end
```

```
                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in                                                       "hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
    i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                               "energy balance on hot-side of each sub-heat
exchanger"
    T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                        "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
    i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                               "energy balance on cold-side of each sub-heat
exchanger"
    T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                        "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
    delta_T_H[i]=T_H[i]-T_H[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
    delta_T_C[i]=T_C[i]-T_C[i+1]
    delta_i_H[i]=(i_H[i]-i_H[i+1])
    delta_i_C[i]=(i_C[i]-i_C[i+1])
    C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                 "hot-side capacitance rate"
    C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                 "cold-side capacitance rate"
end



duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1] "by breaking the delta T/delta enthalpy values into their own variabl  es I can stop them from
going to zero and messing up solving"
    q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]           "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i] "this ratio of the maximum heat possible for transfer to actual heat transfer defines node
effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))     "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                         "conductance in sub-heat exchanger"
end

UA_total=sum(UA[i],i=1,N)

                                                                    "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                          "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                                     "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                                     "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                        "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)       "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

err=abs(x[N+1]-L)/L                                                 "error function - for HX length optimization
routine"


duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end
```

```
                                                                    "Effectiveness calculation"

eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))
end

Module
Precooler(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_over
all,eff_test[1..N#],x[1..N1#],UA_total,DELTA_T_HC[10])

W=35 [cm]*convert(cm,m)                                             "width of heat exchanger"
N_ch=100 [-]                                                        "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                         "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                         "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                         "thickness of plate"


duplicate i=1,N1#
      P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
      P_C[i]=P_C_in
end

                                                                    "this calculation takes into account the outlet
pressure drop"
                                                                    "this calculation isn't using variable density
values yet - this needs to be added"

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                               "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                             "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                      "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                               "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                        "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                        "temperature of cold outlet fluid"

N=N#                                                                "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                            "total heat transfer rate"
end

                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in                                                       "hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                       "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                              "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                       "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                              "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                       "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                  "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
```

```
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                    "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                    "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                                      "by breaking the delta T/delta enthalpy values
into their own variabl  es I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]              "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
      eff_test[i]=q_dot_node[i]/q_dot_max[i]                             "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))         "effectiveness of sub-heat exchanger"
      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')    "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                            "conductance in sub-heat exchanger"
end

UA_total=sum(UA[i],i=1,N)

                                                                         "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                               "starting position of 1st sub-heat exchanger"

duplicate i=1,N
      call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
      h_H[i], h_H_H[i], dPHdx[i])                                        "hot-side local heat transfer coefficient"
      call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
      h_C[i], h_C_H[i], dPCdx[i])                                        "cold-side local heat transfer coefficient"
      k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                           "metal conductivity at local average temperature"
      DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)          "length of sub-heat exchanger"
      x[i+1]=x[i]+DELTAx[i]
end

err=abs(x[N+1]-L)/L                                                      "error function - for HX length optimization
routine"


duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

                                                                         "Effectiveness calculation"


eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))  "This definition of effectiveness is taken from dostal's
thesis page 74"

end

Module
HTR(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total,DELTA_T_HC[10])

W=35 [cm]*convert(cm,m)                                                  "width of heat exchanger"
N_ch=100 [-]                                                             "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                              "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                              "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                              "thickness of plate"


duplicate i=1,N1#
      P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
      P_C[i]=P_C_in
end

                "this calculation takes into account the outlet pressure drop"
                "this calculation isn't using variable density values yet - this needs to be added"
```

```
i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                    "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                  "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                           "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                    "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                             "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])             "temperature of cold outlet fluid"

N=N#                                                     "number of sub-heat exchangers"
duplicate i=1,N
    q_dot[i]=i*q_dot/N                                   "total heat transfer rate"
end


                                                         "Obtain temperature distribution"
T_H[1]=T_H_in                                            "hot-side inlet temperature"
T_C[1]=T_C_out                                           "cold-side outlet temperature"
i_H[1]=i_H_in                                            "hot_side inlet enthalpy"
i_C[1]=i_C_out                                           "cold-side outlet enthalpy"

duplicate i=2,(N+1)
    i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                    "energy balance on hot-side of each sub-heat
exchanger"
    T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])             "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
    i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                    "energy balance on cold-side of each sub-heat
exchanger"
    T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])             "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                         "Apply effectiveness-NTU solution"
duplicate i=1,N
    delta_T_H[i]=T_H[i]-T_H[i+1]                         "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
    delta_T_C[i]=T_C[i]-T_C[i+1]
    delta_i_H[i]=(i_H[i]-i_H[i+1])
    delta_i_C[i]=(i_C[i]-i_C[i+1])
    C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])     "hot-side capacitance rate"
    C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])     "cold-side capacitance rate"
end



duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1]                        "by breaking the delta T/delta enthalpy values
into their own variabl  es I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]  "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]               "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))  "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])              "conductance in sub-heat exchanger"
end

UA_total=sum(UA[i],i=1,N)


                                                         "determine length of each sub-heat exchanger"
x[1]=0 [m]                                               "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                          "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
```

```
    h_C[i], h_C_H[i], dPCdx[i])                                   "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                      "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)     "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

err=abs(x[N+1]-L)/L                                               "error function - for HX length optimization
routine"


duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

                                                                  "Effectiveness calculation"


eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))  "This definition of effectiveness is taken from dostal's
thesis page 74"

end

Module
LTR(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total,DELTA_T_HC[10])


W=35 [cm]*convert(cm,m)                                           "width of heat exchanger"
N_ch=100 [-]                                                      "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                       "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                       "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                       "thickness of plate"


duplicate i=1,N1#
    P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
    P_C[i]=P_C_in
end

            "this calculation takes into account the outlet pressure drop"
            "this calculation isn't using variable density values yet - this needs to be added"

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                             "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                           "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                    "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                             "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                      "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                      "temperature of cold outlet fluid"

N=N#                                                              "number of sub-heat exchangers"
duplicate i=1,N
    q_dot[i]=i*q_dot/N                                            "total heat transfer rate"
end

                                                                  "Obtain temperature distribution"
T_H[1]=T_H_in                                                     "hot-side inlet temperature"
T_C[1]=T_C_out                                                    "cold-side outlet temperature"
i_H[1]=i_H_in                                                     "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                    "cold-side outlet enthalpy"

duplicate i=2,(N+1)
    i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                             "energy balance on hot-side of each sub-heat
exchanger"
    T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                      "temperature leaving hot-side of each sub-heat
exchanger"
end
```

```
duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                                  "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                           "temperature leaving cold-side of each sub-heat
exchanger"
end



                                                                         "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                       "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                   "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                   "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                                      "by breaking the delta T/delta enthalpy values
into their own variabl  es I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]              "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
      eff_test[i]=q_dot_node[i]/q_dot_max[i]                             "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))        "effectiveness of sub-heat exchanger"
      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')    "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                            "conductance in sub-heat exchanger"
end

UA_total=sum(UA[i],i=1,N)

                                                                         "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                               "starting position of 1st sub-heat exchanger"

duplicate i=1,N
      call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
      h_H[i], h_H_H[i], dPHdx[i])                                        "hot-side local heat transfer coefficient"
      call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
      h_C[i], h_C_H[i], dPCdx[i])                                        "cold-side local heat transfer coefficient"
      k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                           "metal conductivity at local average temperature"
      DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)          "length of sub-heat exchanger"
      x[i+1]=x[i]+DELTAx[i]
end

err=abs(x[N+1]-L)/L                                                      "error function - for HX length optimization
routine"


duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

                                                                         “Effectiveness calculation"



eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))

end
                                                                         "Overall Program Control"

"Constants"
L_H=10                                                                   "length of heating HX"
L_C=10                                                                   "length of cooling HX"
L_R=10                                                                   "length of regenerator HX"
```

```
C$='air_ha'                                                    "cooling fluid"
H$='air_ha'                                                    "heating fluid"
WF$='carbondioxide'                                            "working fluid"


$ifnot ParametricTable
P_high=25000000 [pa]                                           "high pressure point"
{P_low=5357000 [pa]}                                           "low pressure point"
P_ratio =2.5                                                   "pressure ratio"
$endif
P_H=10000000[pa]                                               "hot fluid pressure"
P_C=10000000[Pa]                                               "cold fluid pressure"
P_ratio=P_high/P_low

"in this case the inlet temp of hot fluid is commented out and the temp at the end of the primary heat exchanger is specified"
T_H_in   =900 [k]                                              "heating fluid inlet temp"
{T_H_out=700 [k]}                                              "heating fluid outlet temp"
T_C_in=295 [k]                                                 "cooling fluid inlet temp"
{T_C_out=300[k]}
m_dot_wf=1[kg/s]                                               "mass flow rate of working fluid"
{recomp_fraction=0.08}                                         "fraction of mass flow rate diverted to
recompressor"
m_dot_comp=m_dot_wf*(1-recomp_fraction)                        "mass flow rate to compressor"
m_dot_recomp=m_dot_wf*recomp_fraction                          "mass flow rate to recompressor"
m_dot_h=30 [kg/s]                                              "mass flow rate of heating fluid"
m_dot_c=30 [kg/s]                                              "mass flow rate of cooling fluid"


{eff_r_l=.95                                                   "low temp regenerator effectiveness"
eff_r_h=.95}                                                   "high temp regen effectiveness"



{UA_HTR=5000 [w/k]                                             "UA of regenerators set for when regenerator
effectiveness is not set"
UA_LTR=5000 [w/k]                                              "UA of primary heat exchanger and precooler"
UA_PHX=3000 [w/k]
UA_precooler=5000 [w/k]}

UA_total=UA_HTR+UA_LTR+UA_PHX+UA_Precooler
UA_total=15000 [w/k]

UA_PHX_fraction = UA_PHX/UA_total
UA_HTR_fraction = UA_HTR/UA_total
UA_LTR_fraction = UA_LTR/UA_total
UA_PC_fraction = UA_Precooler/UA_total

turbine_eff=.9                                                 "turbine efficiency"
comp_eff=.89                                                   "compressor efficiency"

T2=T_WF[11]
T3=T_WF[22]
T4=T_WF[33]

{T4=823 [k]
T8=305 [k]}

Call
PHX(L_H,H$,T_H_in,T_H_out,P_H,WF$,m_dot_h,T3,p_high,m_dot_wf:T_H[1..N1#],T_WF[N23#..N33#],q_dot_h,err_h,eff_h,eff_no
de_h[1..N#],x_h[1..N1#],UA_PHX,T_PINCH_PHX)"primary heat exchanger"
Call
LTR(L_R,WF$,T6,T7,P_low,WF$,m_dot_wf,T1,p_high,m_dot_comp:T_WF[N45#..N55#],T_WF[1..N1#],q_dot_r_l,err_r_l,eff_r_l,eff_
node_r_l[1..N#],x_r_l[1..N1#],UA_LTR,T_PINCH_LTR)  "low temp regenerator"
Call
HTR(L_R,WF$,T5,T6,P_low,WF$,m_dot_wf,T2,p_high,m_dot_wf:T_WF[N34#..N44#],T_WF[N2#..N22#],q_dot_r_h,err_r_h,eff_r_h,
eff_node_r_h[1..N#],x_r_h[1..N1#],UA_HTR,T_PINCH_HTR)"high temp regen"
Call
Precooler(L_R,WF$,T7,T8,P_low,WF$,m_dot_comp,T_C_in,P_C,m_dot_c:T_WF[N56#..N66#],T_C[1..N1#],q_dot_c,err_c,eff_c,eff_
node_c[1..N#],x_c[1..N1#],UA_precooler,T_PINCH_PC)             "precooler"
Call Compressor(comp_eff,WF$,T7,P_low,P_high:T2,W_recomp)      "recompressing compressor"
```

```
Call Compressor(comp_eff,WF$,T8,P_low,P_high:T1,W_compressor)        "primary compressor"
Call Turbine(turbine_eff,WF$,T4,P_high,P_low:T5,W_turbine)          "main turbine"


work_comp_total=w_recomp*m_dot_recomp+W_compressor*m_dot_comp        "the total work done by the compressors,
weighted by mass flow rate"
work_turbine_total=W_turbine*m_dot_wf                               "total work done by the turbine, weighted by
mass flow rate"
efficiency_cycle= (work_turbine_total+work_comp_total)/q_dot_h      "this defines the efficiency of the cycle as a
whole"

work_net = work_comp_total+work_turbine_total
```

Appendix D: Split Expansion EES Code

"these groups of contants are laid out to allow for different numbers of nodes. By commenting out all but one group we can select either 2, 5, 10, or 20 nodes."

```
{$Constant N#=20
$Constant N1#=21                    "N+1"
$Constant N2#=22                    "N+2"
$Constant N3#=23                    "N+3"
$Constant N4#=24                    "N+4"
$Constant N22#=42                   "2*N+2"
$Constant N23#=43                   "2*N+3"
$Constant N24#=44                   "2*N+4"
$Constant N34#=64                   "3*N+4"
$Constant N35#=65                   "3*N+5"
$Constant N45#=85                   "4*N+5"
$Constant N46#=86                   "4*N+6"}

$Constant N#=10
$Constant N1#=11                    "N+1"
$Constant N2#=12                    "N+2"
$Constant N3#=13                    "N+3"
$Constant N4#=14                    "N+4"
$Constant N22#=22                   "2*N+2"
$Constant N23#=23                   "2*N+3"
$Constant N24#=24                   "2*N+4"
$Constant N33#=33                   "3*N+3"
$Constant N34#=34                   "3*N+4"
$Constant N35#=35                   "3*N+5"
$Constant N44#=44                   "4*N+4"
$Constant N45#=45                   "4*N+5"
$Constant N46#=46                   "4*N+6"
$Constant N55#=55                   "5*N+5"
$Constant N56#=56                   "5*N+5"
$Constant N65#=65                   "6*N+5"
$Constant N66#=66                   "6*N+6"
$Constant N67#=67                   "6*N+7"

{$Constant N#=5
$Constant N1#=6                     "N+1"
$Constant N2#=7                     "N+2"
$Constant N3#=8                     "N+3"
$Constant N4#=9                     "N+4"
$Constant N22#=12                   "2*N+2"
$Constant N23#=13                   "2*N+3"
$Constant N24#=14                   "2*N+4"
$Constant N34#=19                   "3*N+4"
$Constant N35#=20                   "3*N+5"
$Constant N45#=25                   "4*N+5"
$Constant N46#=26                   "4*N+6"}


$UnitSystem SI MASS RAD PA K J
$Tabstops 0.2 0.4 0.6 3.5 in
{$Constant N#=2
$Constant N1#=3                     "N+1"
$Constant N2#=4                     "N+2"
$Constant N3#=5                     "N+3"
$Constant N4#=6                     "N+4"
$Constant N22#=6                    "2*N+2"
$Constant N23#=7                    "2*N+3"
$Constant N24#=8                    "2*N+4"
$Constant N34#=10                   "3*N+4"
$Constant N35#=11                   "3*N+5"
$Constant N45#=13                   "4*N+5"
$Constant N46#=14                   "4*N+6"}
```

```
Subprogram Compressor(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                                  "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                                   "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in "s2s represents the entropy that would exist after the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=P_out)                             "the enthalpy after the compressor, if the
compressor were purely isentropic"
      Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
      W=Ws/eff                                                          "the amount of work required - taking into
account efficiency"
      W=h_in-h_out  "the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)                            "the temperature after the compressor defined
by the actual enthalpy and pressure after the compressor"


end

Subprogram Recompressor(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                                  "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                                   "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in                                                       "s2s represents the entropy that would exist after
the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=P_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
      Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
      W=Ws/eff                                                          "the amount of work required - taking into
account efficiency"
      W=h_in-h_out  "the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)                            "the temperature after the compressor defined
by the actual enthalpy and pressure after the compressor"


end

Subprogram Turbine(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                                  "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                                   "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in                                                       "s4s represents the entropy that would exist after
the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=p_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
      Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
      W=Ws*eff "the amount of work produced - taking into account efficiency"
      W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out) "the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"

end

Subprogram Secondaryturbine(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                                  "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                                   "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in "s4s represents the entropy that would exist after the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=p_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
      Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
      W=Ws*eff "the amount of work produced - taking into account efficiency"
      W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)                            "the temperature after the compressor defined
by the actual enthalpy and pressure after the compressor"
```

```
end

Module
PHX(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total)
                                                                    "Inputs"
W=35 [cm]*convert(cm,m)                                             "width of heat exchanger"
N_ch=100 [-]                                                        "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                         "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                         "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                         "thickness of plate"

duplicate i=1,N1#
      P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
      P_C[i]=P_C_in
end


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                               "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                             "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                      "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                               "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                        "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                        "temperature of cold outlet fluid"

N=N#                                                                "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                            "total heat transfer rate"
end

                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in "                                                     hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                             "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                       "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                             "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                      "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                  "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])              "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])              "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                                 "by breaking the delta T/delta enthalpy values
into their own variabl  es I can stop them from going to zero and messing up solving"
```

```
    q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]                    "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]                                   "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))               "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')          "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                                  "conductance in sub-heat exchanger"
end

UA_total=sum(UA[i],i=1,N)

                                                                             "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                                   "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                                              "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                                              "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                                 "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)                "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

err=abs(x[N+1]-L)/L                                                          "objective function"


duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))
end

Module
HTR(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total)
                                                                             "Inputs"
W=35 [cm]*convert(cm,m)                                                      "width of heat exchanger"
N_ch=100 [-]                                                                 "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                                  "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                                  "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                                  "thickness of plate"

duplicate i=1,N1#
    P_H[i]=P_H_in    "this represents the pressure drop at the inlet of the heat exchanger"
    P_C[i]=P_C_in
end


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                                        "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                                      "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                               "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                                        "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                                 "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                                 "temperature of cold outlet fluid"

N=N#                                                                         "number of sub-heat exchangers"
duplicate i=1,N
    q_dot[i]=i*q_dot/N                                                       "total heat transfer rate"
end

                                                                             "Obtain temperature distribution"
T_H[1]=T_H_in "                                                              hot-side inlet temperature"
T_C[1]=T_C_out                                                               "cold-side outlet temperature"
i_H[1]=i_H_in                                                                "hot_side inlet enthalpy"
```

```
i_C[1]=i_C_out                                                    "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                           "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                    "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                           "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                    "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                  "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])            "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])            "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                               "by breaking the delta T/delta enthalpy values
into their own variabl  es I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]       "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
      eff_test[i]=q_dot_node[i]/q_dot_max[i]                      "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))  "effectiveness of sub-heat exchanger"
      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                     "conductance in sub-heat exchanger"
end

UA_total=sum(UA[i],i=1,N)

                                                                  "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                        "starting position of 1st sub-heat exchanger"

duplicate i=1,N
      call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
      h_H[i], h_H_H[i], dPHdx[i])                                 "hot-side local heat transfer coefficient"
      call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
      h_C[i], h_C_H[i], dPCdx[i])                                 "cold-side local heat transfer coefficient"
      k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                    "metal conductivity at local average temperature"
      DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)   "length of sub-heat exchanger"
      x[i+1]=x[i]+DELTAx[i]
end

err=abs(x[N+1]-L)/L                                               "objective function"


duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

                                                                  "Effectiveness calculation"

eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))

end
```

```
Module
LTR(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_overall,eff
_test[1..N#],x[1..N1#],UA_total)
                                                                    "Inputs"
W=35 [cm]*convert(cm,m)                                             "width of heat exchanger"
N_ch=100 [-]                                                        "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                         "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                         "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                         "thickness of plate"

duplicate i=1,N1#
      P_H[i]=P_H_in "this represents the pressure drop at the inlet of the heat exchanger"
      P_C[i]=P_C_in
end


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                               "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                             "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                      "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                               "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                        "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                        "temperature of cold outlet fluid"

N=N#                                                                "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                            "total heat transfer rate"
end

                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in "                                                     hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                             "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                       "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                             "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                      "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                  "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])              "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])              "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                                 "by breaking the delta T/delta enthalpy values
into their own variabl  es I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N# "this is the heat transfer through the node of the heat exchanger"
```

```
      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]                    "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
      eff_test[i]=q_dot_node[i]/q_dot_max[i]                                   "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))              "effectiveness of sub-heat exchanger"
      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')          "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                                  "conductance in sub-heat exchanger"
end

UA_total=sum(UA[i],i=1,N)

                                                                               "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                                     "starting position of 1st sub-heat exchanger"

duplicate i=1,N
      call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
      h_H[i], h_H_H[i], dPHdx[i])                                              "hot-side local heat transfer coefficient"
      call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
      h_C[i], h_C_H[i], dPCdx[i])                                              "cold-side local heat transfer coefficient"
      k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                                 "metal conductivity at local average temperature"
      DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)                "length of sub-heat exchanger"
      x[i+1]=x[i]+DELTAx[i]
end

err=abs(x[N+1]-L)/L                                                            "objective function"


duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

                                                                               "Effectiveness calculation"

eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))

end

Module
Precooler(L,H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c:T_H[1..N1#],T_C_reverse[1..N1#],q_dot,err,eff_over
all,eff_test[1..N#],x[1..N1#],UA_total)
                                                                               "Inputs"
W=35 [cm]*convert(cm,m)                                                        "width of heat exchanger"
N_ch=100 [-]                                                                   "number of channel pairs"
th_H=2.2 [mm]*convert(mm,m)                                                    "channel width on hot-side"
th_C=2.2 [mm]*convert(mm,m)                                                    "channel width on cold-side"
th_m=0.5 [mm]*convert(mm,m)                                                    "thickness of plate"

duplicate i=1,N1#
      P_H[i]=P_H_in                    "this represents the pressure drop at the inlet of the heat exchanger"
      P_C[i]=P_C_in
end


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                                          "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                                        "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                                 "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                                          "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                                   "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N1#])                                   "temperature of cold outlet fluid"

N=N#                                                                           "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                                       "total heat transfer rate"
end


                                                                               "Obtain temperature distribution"
T_H[1]=T_H_in "                                                                hot-side inlet temperature"
T_C[1]=T_C_out                                                                 "cold-side outlet temperature"
```

```
i_H[1]=i_H_in                                                    "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                   "cold-side outlet enthalpy"

duplicate i=2,(N+1)
    i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                            "energy balance on hot-side of each sub-heat
exchanger"
    T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                     "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
    i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                            "energy balance on cold-side of each sub-heat
exchanger"
    T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                     "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                 "Apply effectiveness-NTU solution"
duplicate i=1,N
    delta_T_H[i]=T_H[i]-T_H[i+1]                                 "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
    delta_T_C[i]=T_C[i]-T_C[i+1]
    delta_i_H[i]=(i_H[i]-i_H[i+1])
    delta_i_C[i]=(i_C[i]-i_C[i+1])
    C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])             "hot-side capacitance rate"
    C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])             "cold-side capacitance rate"
end



duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1]                                "by breaking the delta T/delta enthalpy values
into their own variabl  es I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N#   "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]        "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff_test[i]=q_dot_node[i]/q_dot_max[i]                       "this ratio of the maximum heat possible for
transfer to actual heat transfer defines node effectiveness"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))  "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                      "conductance in sub-heat exchanger"
end

UA_total=sum(UA[i],i=1,N)

                                                                 "determine length of each sub-heat exchanger"
x[1]=0 [m]                                                       "starting position of 1st sub-heat exchanger"

duplicate i=1,N
    call DuctFlow_local(H$,(T_H[i]+T_C[i])/2,p_H[i],m_dot_H/N_ch,th_H,W,x[i]+0.001 [m],0 [-]: &
    h_H[i], h_H_H[i], dPHdx[i])                                  "hot-side local heat transfer coefficient"
    call DuctFlow_local(C$,(T_H[i]+T_C[i])/2,p_C[i],m_dot_C/N_ch,th_C,W,x[N+1]-x[i]+0.001 [m],0 [-]: &
    h_C[i], h_C_H[i], dPCdx[i])                                  "cold-side local heat transfer coefficient"
    k_m[i]=k_('Titanium', (T_H[i]+T_C[i])/2)                     "metal conductivity at local average temperature"
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(2*N_ch*W)    "length of sub-heat exchanger"
    x[i+1]=x[i]+DELTAx[i]
end

err=abs(x[N+1]-L)/L                                              "objective function"


duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

                                                                 "Effectiveness calculation"

eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in)) "This definition of effectiveness is taken from dostal's
thesis page 74"
```

```
end

                                                                    "Overall Program Control"

                                                                    "Constants"
L_H=10        "length of heating HX"
L_C=10        "length of cooling HX"
L_R=10        "length of regenerator HX"
C$='air_ha'   "cooling fluid"
H$='air_ha'   "heating fluid"
WF$='carbondioxide'                                                 "working fluid"

p_intermediate=p_low+f*(P_high-P_Low)                               "definition of 'f', the split expansion factor. The
higher the value (the closer to the recompression cycle) the better efficiency is - but a lower value takes stress off the heat source by
reducing the pressure in it. This could be useful if material concerns are an issue"
f=0.5

$ifnot ParametricTable
P_high=25000000 [pa]                                                "high pressure point"
{P_low=5000000 [pa]}                                                "low pressure point"
P_ratio = 3
$endif
P_H=1000000[Pa]                                                     "hot fluid pressure"
P_C=1000000[Pa]                                                     "cold fluid pressure"
P_ratio=P_high/P_low

T_H_in   =900 [k]                                                   "heating fluid inlet temp"
T_C_in=295 [k]                                                      "cooling fluid inlet temp"
m_dot_wf=1[kg/s]                                                    "mass flow rate of working fluid"
{recomp_fraction=0.08}                                              "fraction of mass flow rate diverted to
recompressor"
m_dot_comp=m_dot_wf*(1-recomp_fraction)                             "mass flow rate to compressor"
m_dot_recomp=m_dot_wf*recomp_fraction                               "mass flow rate to recompressor"
m_dot_h=30 [kg/s]                                                   "mass flow rate of heating fluid"
m_dot_c=30 [kg/s]                                                   "mass flow rate of cooling fluid"

{eff_r_l=.95                                                        "low temp regenerator effectiveness"
eff_r_h=.95}                                                        "high temp regen effectiveness"
turbine_eff=.9                                                      "turbine efficiency"
comp_eff=.89                                                        "compressor efficiency"


"UA values for constant UA cases"
{UA_HTR=5000
UA_LTR=5000
UA_PHX=3000
UA_Precooler=5000 }

UA_Total=UA_HTR+UA_LTR+UA_PHX+UA_Precooler
UA_TOTAL = 15000 [w/k]


T2=T_WF[N1#]
T3=T_WF[N22#]
T5=T_WF[N33#]

{T5=823 [k]                                                         "High and low temperatures in cycle"
T9=305 [k]}

Call
PHX(L_H,H$,T_H_in,T_H_out,P_H,WF$,m_dot_h,T4,p_intermediate,m_dot_wf:T_H[1..N1#],T_WF[N23#..N33#],q_dot_h,err_h,eff_
h,eff_node_h[1..N#],x_h[1..N1#],UA_PHX)
Call
LTR(L_R,WF$,T7,T8,P_low,WF$,m_dot_wf,T1,p_high,m_dot_comp:T_WF[N45#..N55#],T_WF[1..N1#],q_dot_r_l,err_r_l,eff_r_l,eff_
node_r_l[1..N#],x_r_l[1..N1#],UA_LTR)
Call
HTR(L_R,WF$,T6,T7,P_low,WF$,m_dot_wf,T2,p_high,m_dot_wf:T_WF[N34#..N44#],T_WF[N2#..N22#],q_dot_r_h,err_r_h,eff_r_h,
eff_node_r_h[1..N#],x_r_h[1..N1#],UA_HTR)
```

```
Call
Precooler(L_R,WF$,T8,T9,P_low,WF$,m_dot_comp,T_C_in,P_C,m_dot_c:T_WF[N56#..N66#],T_C[1..N1#],q_dot_c,err_c,eff_c,eff_
node_c[1..N#],x_c[1..N1#],UA_precooler)
Call Recompressor(comp_eff,WF$,T8,P_low,P_high:T2,W_recomp)
Call Compressor(comp_eff,WF$,T9,P_low,P_high:T1,W_compressor)
Call Turbine(turbine_eff,WF$,T5,P_intermediate,P_low:T6,W_turbine)
Call SecondaryTurbine(turbine_eff,WF$,T3,P_high,P_intermediate:T4,W_turbine_secondary)

work_comp_total=w_recomp*m_dot_recomp+W_compressor*m_dot_comp        "the total work done by the compressors,
weighted by mass flow rate"
work_turbine_total=W_turbine*m_dot_wf +W_Turbine_secondary*m_dot_wf        "total work done by the turbine, weighted by
mass flow rate"

work_net = work_comp_total+work_turbine_total

efficiency_cycle= (work_turbine_total+work_comp_total)/q_dot_h        "this defines the efficiency of the cycle as a
whole"

duplicate i=1,N22#
    s[i] = entropy(WF$,T=T_WF[i],P=P_High)
end

duplicate i=N23#,N33#
    s[i] = entropy(WF$,T=T_WF[i],P=P_intermediate)
end

duplicate i=N34#,N66#
    s[i] = entropy(WF$,T=T_WF[i],P=P_low)
end
T_WF[N67#]=T_WF[1]        "closing the loop for plotting purposes"
s[N67#]=s[1]
```

Appendix E: Detailed Water-Cooled EES Code

```
$UnitSystem SI MASS RAD PA K J
$Tabstops 0.2 0.4 0.6 3.5 in

$Constant N_CHX# =10 "number of nodes in complex HX"
$Constant N_SHX# = 2 "number of nodes in simple HX"

$Constant N_CHX1# = 11 "number of nodes in complex HX plus 1"
$Constant N_SHX1# = 3 "number of nodes in simple HX plus 1"

$Bookmark Ductflow_N
procedure DuctFlow_N_(Re, Pr, L\D, Aspect, relRough: Nusselt_T,Nusselt_H, f)
      if (Re> 2300) then {turbulent flow}
          Call PipeFlow_Turbulent(Re, Pr, L\D, RelRough : Nusselt_T, f)
          Nusselt_H=Nusselt_T
      else {laminar flow <2300}
          Call DuctFlow_laminar(Re, Pr, L\D, Aspect: Nusselt_T, Nusselt_H, f)
      Endif
end
$Bookmark DuctFlow_

procedure DuctFlow_( Fluid$, T, P, m_dot, H, W, L, RelRough: h_T, h_H, DELTAP, Nusselt_T,  f, Re,V_dot)
{$DuctFlow
This function returns the heat transfer coefficient in [W/m^2-K] and friction factor [-] inside a  rectangular.    Note:  EES must be set to SI units.  T and P must be provided in the units specified in the EES UnitSystem dialog.

Inputs
Fluid$ is a string constant or variable with the name of a fluid in the EES database
T is the temperature (in C or K) that properties are evaluated at.
P is the absolute average pressure of the fluid in the pipe.  The pressure can be Pa, kPa, bar or MPa, depending upon the EES units setting.
m_dot is the flow rate in kg/s.
W is the duct width in m
H is the duct height in m
L is the duct length in m
RelRough is the ratio of the roughness to the tube diameter  <0.05

Outputs  (Only the first output is required)
h_T is the heat transfer coefficient in W/m^2-K assuming that the duct wall is isothermal - this is a lower bound on h
h_H is the heat transfer coefficient in W/m^2-K assuming that the duct wall is exposed to a constant heat flux - this is an upper bound on h
DELTAP is the pressure drop across the tube in the EES pressure setting
Nusselt_T is the Nusselt number corresponding to a constant duct wall temperature
f is the friction factor
Re is Reynold's number

Notes
This function determines Re, Pr and L/D from input data and then calls DuctFlow_N to obtain f and Nusselt}
call IF_unit_check(1:Eng, Uh$,UL$,UMF$,UP$,UT$)
call GetPropsInt(Fluid$,T,P:rho, mu, k, Pr)
D_h=4*W*H/(2*W+2*H)  {hydraulic diameter}
Aspect=H/W
if (Aspect>1) then Aspect=W/H
A=W*H  {cross-sectional area}
Re=(m_dot*D_h)/(A*mu)
call DuctFlow_N_(Re,Pr,L/D_h, Aspect,RelRough: Nusselt_T, Nusselt_H, f)
h_T=Nusselt_T*k/D_h  {based on constant temperature boundary}
h_H=Nusselt_H*k/D_h {based on constant heat flux boundary}
V_dot=m_dot/(A*rho) "velocity"
DELTAP=f*L/D_h*(1/2)*rho*V_dot^2*convert(Pa,UP$)
tree=ENg
END
$Bookmark Channel_Flow
```

```
module
channel_flowPH(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DEL
TAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

      T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
      T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"

      call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
      call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

      D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
      D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"


end

module
channel_flowRG(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DE
LTAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

      T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
      T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"

      call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
      call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

      D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
      D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"


end

module
channel_flowPC(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DEL
TAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

      T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
      T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"

      call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
      call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

      D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
      D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"


end
```

```
$Bookmark Compressor
Subprogram Compressor(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)"calculate enthalpy pre-compressor from temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)"calculate entropy pre-compressor from temp/pressure"
      s_outs=s_in"s2s represents the entropy that would exist after the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=P_out)"the enthalpy after the compressor, if the compressor were purely isentropic"
      Ws=h_in-h_outs"the work the compressor requires, if the compressor were purely isentropic"
      W=Ws/eff"the amount of work required - taking into account efficiency"
      W=h_in-h_out"the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)"the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"


end
$Bookmark Turbine
Subprogram Turbine(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)"calculate enthalpy pre-compressor from temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)"calculate entropy pre-compressor from temp/pressure"
      s_outs=s_in"s4s represents the entropy that would exist after the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=p_out)"the enthalpy after the compressor, if the compressor were purely isentropic"
      Ws=h_in-h_outs"the work the compressor requires, if the compressor were purely isentropic"
      W=Ws*eff"the amount of work produced - taking into account efficiency"
      W=h_in-h_out"the amount of actual work relates to the actual difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)"the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"

end
$Bookmark PHX
 module
PHX(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_C,W_H,N_CH_C:T_H[1..N_SHX1#],T_C_reverse[1..N_
SHX1#],q_dot,eff_overall,UA_total,delta_T_HC[2],Area,P_C[1..3],Length_HX,H_bar_H,H_bar_C,DELTAP_H,DELTAP_C,V_H[1],V_
C[1],Re_h[1],Re_C[1],Volume,pumping_power_hot,N_CH_H)

th_m=1.9 [mm]*convert(mm,m)                                   "thickness of plate"
M$ = 'titanium'
D_C = W_C
D_H = W_H

rough=0


i_H_in=T_H_in*(c_(H$,T=T_H_in) +c_(H$,T=T_H_out))/2           "enthalpy of hot inlet fluid"
i_H_out=T_H_out*(c_(H$,T=T_H_in) +c_(H$,T=T_H_out))/2         "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                "total heat transfer rate"


i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                         "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                  "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_SHX1#])              "temperature of cold outlet fluid"

N=N_SHX#   "number of sub-heat exchangers"
N_plus_1=N+1

duplicate i=1,N
      q_dot[i]=i*q_dot/N                                      "total heat transfer rate"
end


                                                              "Obtain temperature distribution"
T_H[1]=T_H_in                                                 "hot-side inlet temperature"
T_C[1]=T_C_out                                                "cold-side outlet temperature"
i_H[1]=i_H_in                                                 "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H) "energy balance on hot-side of each sub-heat exchanger"
      T_H[i]=i_H[i]/c_(H$,T=T_H[i])"temperature leaving hot-side of each sub-heat exchanger"
```

```
end

duplicate i=2,(N+1)
    i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C) "energy balance on cold-side of each sub-heat exchanger"
    T_C[i]=temperature(C$,h=i_C[i],P=p_C[i]) "temperature leaving cold-side of each sub-heat exchanger"
end

"Apply effectiveness-NTU solution"
duplicate i=1,N
    delta_T_H[i]=T_H[i]-T_H[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
    delta_T_C[i]=T_C[i]-T_C[i+1]
    delta_i_H[i]=(i_H[i]-i_H[i+1])
    delta_i_C[i]=(i_C[i]-i_C[i+1])
    C_dot_H[i]*(delta_T_H[i]) =m_dot_H*(delta_i_H[i])                    "hot-side capacitance rate"
    C_dot_C[i]*(delta_T_C[i]) =m_dot_C*(delta_i_C[i])                    "cold-side capacitance rate"
end

duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1]                                        "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N"this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i] "this is the maximum amount of heat that can be transfered through
node of the heat exchanger"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))          "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')      "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                              "conductance in sub-heat exchanger"
end
"determine length of each sub-heat exchanger"
x[1]=1e-5 [m] "starting position of 1st sub-heat exchanger"

fouling_factor = 0.00035 [m^2-K/W]                                       "fouling factor"

duplicate i=1,N
    DELTAx[i]=UA[i]*(fouling_factor+1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_C*N_CH_C)  "length of sub-heat exchanger"
    call
channel_flowPH(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,ro
ugh,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])

    k_m[i]=k_(M$, (T_H[i]+T_C[i])/2) "metal conductivity at local average temperature"
    x[i+1]=x[i]+DELTAx[i]

end

duplicate i=1,N-1 "all pressure calculations except the first and last node"

    P_H[i+1] = P_H[i] - DELTA_P_H[i]
    P_C[i+1] = P_C[i] - DELTA_P_C[i]

end

P_H[1]=P_H_in -DELTA_P_H_entrance
P_C[1]=P_C_in - DELTA_P_C_entrance

P_H[N_SHX1#] = P_H[N] - DELTA_P_H[N] - DELTA_P_H_exit
P_C[N_SHX1#] = P_C[N] - DELTA_P_C[N] - DELTA_P_C_exit
C_entrance = 0.5 "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*rho_(H$,T=T_H_in)*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*rho_(H$,T=T_H[N])*V_H[N]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N],P=P_C[N])*V_C[N]^2

duplicate i=1,N
    H_H_weighting[i]=H_H[i]*DELTAx[i] "this section calculates weighting factors for whole-HX calculation"
    H_C_weighting[i]=H_C[i]*DELTAx[i]
end
```

```
H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX

DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]

Length_HX = x[N+1] "total length of HX"

UA_total=sum(UA[i],i=1,N)

N_CH_C = N_CH_H*(W_H/W_C)  "number of channels on cold side defined by ratio of channel widths"

Volume = N_CH_C*(2*W_C+2*th_m)*(th_m+D_C)*Length_HX                    "HX volume"

pumping_power_hot = DELTAP_H*m_dot_H/(rho_(H$,T=(T_H[N+1]+T_H[1])/2))

Area = N_CH_C*D_C*Length_HX

duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - T_C_in*c_(H$,T=T_C_in))  "This definition of effectiveness is taken from dostal's thesis page
74"

end

$Bookmark Regenerator
module
Regenerator(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_H,W_C,N_CH_H:T_H[1..N_SHX1#],T_C_revers
e[1..N_SHX1#],q_dot,eff_overall,UA_total,delta_T_HC[2],Area,P_H[1..N_SHX1#],P_C[1..N_SHX1#],Length_HX,H_bar_H,H_bar_C,
DELTAP_H,DELTAP_C,V_H[1],V_C[1],Re_h[1],Re_C[2],Volume,N_CH_C)

th_m=1.9[mm]*convert(mm,m)                                             "thickness of plate"
M$ = 'titanium'
D_C = W_C
D_H = W_H
rough=0

i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                                  "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                                "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                         "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                                  "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                           "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_SHX#+1])                      "temperature of cold outlet fluid"

N=N_SHX#   "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                               "total heat transfer rate"
end

                                                                       "Obtain temperature distribution"
T_H[1]=T_H_in                                                          "hot-side inlet temperature"
T_C[1]=T_C_out                                                         "cold-side outlet temperature"
i_H[1]=i_H_in                                                          "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                         "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H) "energy balance on hot-side of each sub-heat exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i]) "temperature leaving hot-side of each sub-heat exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C) "energy balance on cold-side of each sub-heat exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i]) "temperature leaving cold-side of each sub-heat exchanger"
end
                                                                       "Apply effectiveness-NTU solution"
```

```
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                    "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                    "cold-side capacitance rate"
end

duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N "this is the heat transfer through the node of the heat exchanger"
      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]              "this is the maximum amount of heat that can be transfered through node of the heat exchanger"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))  "effectiveness of sub-heat exchanger"
      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU') "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])  "conductance in sub-heat exchanger"
end

"determine length of each sub-heat exchanger"
x[1]=1e-5 [m]  "starting position of 1st sub-heat exchanger"

duplicate i=1,N
      DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_H*N_CH_H) "length of sub-heat exchanger"
      call channel_flowRG(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,rough,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])
      k_m[i]=k_(M$, (T_H[i]+T_C[i])/2)  "metal conductivity at local average temperature"
      x[i+1]=x[i]+DELTAx[i]


end


duplicate i=1,N-1 "all pressure calculations except the first and last node"

      P_H[i+1] = P_H[i] - DELTA_P_H[i]
      P_C[i+1] = P_C[i] - DELTA_P_C[i]

end

P_H[1]=P_H_in - DELTA_P_H_entrance
P_C[1]=P_C_in- DELTA_P_C_entrance


P_H[N+1] = P_H[N] - DELTA_P_H[N] -DELTA_P_H_exit
P_C[N+1] = P_C[N] - DELTA_P_C[N] - DELTA_P_C_exit
C_entrance = 0.5 "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*density(H$,T=T_H_in,P=P_H[1])*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*density(H$,T=T_H[N_SHX#],P=P_H[N_SHX#])*V_H[N_SHX#]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N_SHX#],P=P_C[N_SHX#])*V_C[N_SHX#]^2

duplicate i=1,N

      H_H_weighting[i]=H_H[i]*DELTAx[i] "this section calculates weighting factors for whole-HX calculation"
      H_C_weighting[i]=H_C[i]*DELTAx[i]
end

H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX


DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]
```

```
Length_HX = x[N+1] "total length of HX"

Volume = N_CH_H*(2*W_H+2*th_m)*(th_m+D_H)*Length_HX                    "HX volume"

UA_total=sum(UA[i],i=1,N)

N_CH_C = N_CH_H*(W_H/W_C) "number of channels on cold side defined by ratio of channel widths"

Area = N_CH_H*D_H*Length_HX
duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in)) "This definition of effectiveness is taken from dostal's
thesis page 74"

end

$Bookmark Precooler
module
Precooler(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_H,W_C,N_CH_H:T_H[1..N_CHX1#],T_C_reverse[1
..N_CHX1#],q_dot,eff_overall,UA_total,delta_T_HC[1],Area,P_H[1..N_CHX1#],Length_HX,H_bar_H,H_bar_C,DELTAP_H,DELTAP
_C,V_H[1],V_C[1],Re_H[1],Re_C[1],pumping_power_cold,N_CH_C)

th_m=1.9 [mm]*convert(mm,m) "thickness of plate"
M$ = 'titanium'
D_C = W_C
D_H = W_H
rough=0


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                                 "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[N_CHX1#])                         "enthalpy of hot outlet fluid"
q_dot=m_dot_H*(i_H_in-i_H_out)                                        "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=P_C_in)                                 "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                          "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_CHX1#])                      "temperature of cold outlet fluid"

N=N_CHX#   "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                              "total heat transfer rate"
end

                                                                      "Obtain temperature distribution"
T_H[1]=T_H_in                                                         "hot-side inlet temperature"
T_C[1]=T_C_out                                                        "cold-side outlet temperature"
i_H[1]=i_H_in                                                         "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                        "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)  "energy balance on hot-side of each sub-heat exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])  "temperature leaving hot-side of each sub-heat exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C) "energy balance on cold-side of each sub-heat exchanger"
      T_C[i]=temperature(C$,h=i_C[i],p=p_c[i])  "temperature leaving cold-side of each sub-heat exchanger"
end


                                                                      "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                    "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
```

```
    C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                    "hot-side capacitance rate"
    C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                    "cold-side capacitance rate"
end

duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1]                                      "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]              "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))        "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU') "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                            "conductance in sub-heat exchanger"
end

"determine length of each sub-heat exchanger"
x[1]=1e-5 [m]  "starting position of 1st sub-heat exchanger"

fouling_factor = 0.00035 [m^2-K/W]

duplicate i=1,N
    DELTAx[i]=UA[i]*(fouling_factor+1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_H*N_CH_H) "length of sub-heat exchanger"
    call
channel_flowPC(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,ro
ugh,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])
    k_m[i]=k_(M$, (T_H[i]+T_C[i])/2) "metal conductivity at local average temperature"
    x[i+1]=x[i]+DELTAx[i]


end

duplicate i=1,N-1 "all pressure calculations except the first and last node"

    P_H[i+1] = P_H[i] -DELTA_P_H[i]
    P_C[i+1] = P_C[i]-DELTA_P_C[i]

end


P_H[1]=P_H_in - DELTA_P_H_entrance
P_C[1]=P_C_in- DELTA_P_C_entrance


P_H[N_CHX1#] = P_H[N_CHX#] - DELTA_P_H[N_CHX#] -DELTA_P_H_exit
P_C[N_CHX1#] = P_C[N_CHX#] - DELTA_P_C[N_CHX#] - DELTA_P_C_exit
C_entrance = 0.5 "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*density(H$,T=T_H_in,P=P_H[1])*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*density(H$,T=T_H[N_CHX#],P=P_H[N_CHX#])*V_H[N_CHX#]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N_CHX#],P=P_C[N_CHX#])*V_C[N_CHX#]^2


duplicate i=1,N

    H_H_weighting[i]=H_H[i]*DELTAx[i] "this section calculates weighting factors for whole-HX calculation"
    H_C_weighting[i]=H_C[i]*DELTAx[i]
end

H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX


DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]

Length_HX = x[N+1] "total length of HX"
```

```
UA_total=sum(UA[i],i=1,N)

Area = N_CH_H*D_H*Length_HX

N_CH_C = N_CH_H*(W_C/W_H) "number of channels on cold side defined by ratio of channel widths"

duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))  "This definition of effectiveness is taken from dostal's thesis page 74"


pumping_power_cold= DELTAP_C*m_dot_C/density(C$,T=(T_C[N+1]+T_C[1])/2,P=(P_C[1]+P_c[N+1])/2)

end


$bookmark Main
"Overall Program Control"
"Constants"
C$='water'                                                    "cooling fluid"
H$='Salt (60% NaNO3, 40% KNO3)'                               "heating fluid"
WF$='carbondioxide'                                           "working fluid"

Area_norm =1

N1 = 1          "Regenerator Cold Side Inlet"
N2 = 1+N_SHX# "Regenerator Cold Side outlet"
N3 = 2+N_SHX#  "PHX Inlet"
N4 = 2+2*N_SHX# "PHX Outlet/Turbine Inlet"
N5 = 3+2*N_SHX# "Turbine Outlet"
N6 = 4+2*N_SHX#  "Regenerator Hot Side inlet"
N7 = 4+3*N_SHX# "Regenerator Hot Side Outlet"
N8 = 5+ 3*N_SHX# "Precooler Inlet"
N9  = 5+ 3*N_SHX#+N_CHX# "Precooler Outlet/Compressor inlet"
N10 = 6+ 3*N_SHX#+N_CHX# "Compressor Outlet"
N_PC_1 = N_CHX#+1


channel_length_ratio_PHX =(N_ch_PHX_C/Length_PHX)*(m_dot_wf_scaled/m_dot_wf)"ratio of number of channels in hx to length of hx - fixing this value fixed aspect ratio of hx as hx area changes"
channel_length_ratio_RG =(N_ch_RG_C/Length_RG)*(m_dot_wf_scaled/m_dot_wf)
channel_length_ratio_PC =(N_ch_PC_C/Length_PC)*(m_dot_wf_scaled/m_dot_wf)

T_WF[N9] = 307.5 "setting compressor inlet temp of co2 dictates required temperature of cooling water to precooler"
m_dot_blend =0
$ifnot ParametricTable
{T_C_in=295[k] "cooling fluid inlet temp"}
T_H_in   =900 [k]                                             "heating fluid inlet temp"
m_dot_cool_scaled=500 [kg/s]
m_dot_h_scaled=150 [kg/s]
{area_total_scaled=5250 [m^2]                                 "large"}
area_total_scaled=3500 [m^2]                                  "medium"
{area_total_scaled=2450 [m^2]                                 "small"}
"CONVERGING AT SINGLE RUN SETTINGS"

fraction_pc=0.3 "heat exchanger distribution"
fraction_rg=.45


$endif

channel_length_ratio_PC=10000 "hx aspect ratio"
channel_length_ratio_PHX= 10000
channel_length_ratio_RG=4000
```

```
{T_C_cool = 296 "inlet temperature of blended water to precooler"}
h_C_in = enthalpy(water,T=T_C_in, P=P_C) "inlet enthalpy of water from cooling tower"
h_C_out = enthalpy(water,T=T_C[N_CHX1#],P=P_C) "enthalpy of water exiting cycle"
h_C_cool = enthalpy(water,T=T_C_cool,P=P_C) "enthalpy of water entering cycle"

m_dot_cool_scaled=m_dot_tower+m_dot_blend "cooling water mass flow rate to precooler is the sum of mass flow rate from the cooling tower and from the blended back precooler outlet water"

m_dot_cool_scaled*h_C_cool=m_dot_blend*h_c_out+m_dot_tower*h_c_in "energy balance on water at blending point"

P_high=25000000 [pa] "high pressure point"
{P_low=5000000 [pa]} "low pressure point"

P_ratio=2.9
m_dot_normalized = 1
W_channel_PHX_C=0.0023 [m] "all channels opt"
W_channel_PHX_H=0.0007119[m]
W_channel_RG_C=0.0022[m]
W_channel_RG_H=0.0018 [m]
W_channel_PC_C=0.002[m]
W_channel_PC_H=0.001442 [m]


P_H=10000000[Pa]                                                    "hot fluid pressure"
P_C=10000000[Pa]                                                    "cold fluid pressure"
P_ratio=P_high/P_low

flowrate_compressor_inlet=0.15 [m^3/s]

T2=T_WF[N10]
T5=T_WF[N5]

turbine_eff=.9 "turbine efficiency"
comp_eff=.89"compressor efficiency"

fraction_PHX_opt = area_PHX/area_total "for optimization - these values can't go out of physical bounds"
fraction_PC_opt = area_PC/(area_total-Area_PHX) "for optimization - these values can't go out of physical bounds"


fraction_PHX = area_PHX/area_total
fraction_RG = area_RG/area_total
fraction_PC=area_PC/area_total

area_total = area_PC+area_phx+area_RG
area_total_scaled = area_total*(m_dot_wf_scaled/m_dot_wf)

{UA_total = 15000 [W/K]}
UA_Total = UA_regenerator+UA_precooler+UA_PHX

DELTA_T_salt = T_H_in - T_H_out

Call
PHX(H$,T_H_in,T_H_out,P_H,WF$,m_dot_h,T_WF[N2],P_WF[N2],m_dot_wf,W_channel_PHX_C,W_channel_PHX_H,N_ch_PHX_C:T_H[N1..N2],T_WF[N3..N4],q_dot,eff_h,UA_PHX,T_PHX_Pinch,Area_PHX,P_WF[N3..N4],Length_PHX,H_bar_H_PHX,H_bar_C_PHX,DELTAP_H_PHX,DELTAP_C_PHX,V_PHX_H,V_PHX_C,Re_PHX_H,Re_PHX_C,Volume_PHX,pumping_power_salt,N_ch_PHX_H) "this HX represents heat transfer from heat source to WF after regenerator"

Call
Regenerator(WF$,T5,T6,P_low,WF$,m_dot_wf,T2,p_high,m_dot_wf,W_channel_RG_H,W_channel_RG_C,N_ch_RG_H:T_WF[N6..N7],T_WF[1..N2],q_dot_reg,eff_r,UA_regenerator,T_RG_Pinch,Area_RG,P_WF[N6..N7],P_WF[1..N2],Length_RG,H_bar_H_RG,H_bar_C_RG,DELTAP_H_RG,DELTAP_C_RG,V_RG_H,V_RG_C,Re_RG_H,Re_RG_C,Volume_RG,N_ch_RG_C)"this heat exchanger represents the regenerator"

Call Turbine (turbine_eff,WF$,T_WF[N4],P_WF[N4],P_low:T_WF[N5],W_turbine) "this turbine extracts energy from the working fluid"

Call
Precooler(WF$,T_WF[N7],T1,P_WF[N7],C$,m_dot_wf,T_C_cool,P_C,m_dot_cool,W_channel_PC_H,W_channel_PC_C,N_ch_PC_H:T_WF[N8..N9],T_C[1..N_PC_1],q_dot_cooling,eff_c,UA_precooler,T_PC_Pinch,Area_PC,P_WF[N8..N9],Length_PC,H_bar_H_P
```

```
C,H_bar_C_PC,DELTAP_H_PC,DELTAP_C_PC,V_PC_H,V_PC_C,Re_PC_H,Re_PC_C,pumping_power_water,N_ch_PC_C) "this heat exchanger involves the working fluid after the turbine and the cooling fluid from the cooling tower or whatnot"

Call Compressor(comp_eff,WF$,T_WF[N9],P_WF[N9],P_high:T_WF[N10],W_comp) "the compressor takes the fluid from the heat rejection heat exchanger to the regenerator"

work_net=m_dot_wf*(W_turbine+W_comp) - pumping_power_water - pumping_power_salt "net work done by the cycle"

efficiency=work_net/q_dot "this section of code records the efficiency of the cycle in terms of turbine/compressor work and heat transfer to the system in the primary heat exchanger"



comp_inlet_density=density(WF$,T=T_WF[N9],P=P_WF[N9])                    {working fluid density at compressor inlet}
flowrate_compressor_inlet = (m_dot_wf/comp_inlet_density)*(m_dot_wf_scaled/1 [kg/s])  "volumetric flowrate of working fluid at compressor inlet"

duplicate i=1,N4                                                         {this section sets up the array recording the entropy of the working fluid at each point}
      s[i]=entropy(WF$,T=T_WF[i],P=P_high)
      h[i]=enthalpy(WF$,T=T_WF[i],P=P_high)
end

duplicate i=N5,N9
      s[i]=entropy(WF$,T=T_WF[i],P=P_low)
h[i]=enthalpy(WF$,T=T_WF[i],P=P_high)
end

s[N10]=entropy(WF$,T=T_WF[N10],P=P_high)
h[N10]=enthalpy(WF$,T=T_WF[N10],P=P_high)


"PLANT SIZE SCALING"


Power_scaled=m_dot_wf_scaled*work_net/m_dot_wf "comment out to determine m_dot_wf_scaled"
turbine_power_scaled = w_turbine*m_dot_wf_scaled
comp_power_scaled=w_comp*m_dot_wf_scaled
m_dot_wf_scaled=100*m_dot_wf
area_phx_scaled=m_dot_wf_scaled*area_phx/m_dot_wf
area_rg_scaled=m_dot_wf_scaled*area_rg/m_dot_wf
area_pc_scaled=m_dot_wf_scaled*area_pc/m_dot_wf

pumping_power_water_scaled = pumping_power_water*(m_dot_wf_scaled/m_dot_wf)
pumping_power_salt_scaled = pumping_power_salt*(m_dot_wf_scaled/m_dot_wf)

m_dot_cool_scaled = m_dot_cool*(m_dot_wf_scaled/m_dot_wf)
m_dot_h_scaled = m_dot_h*(m_dot_wf_scaled/m_dot_wf)

T_C_out = T_C[N_CHX1#]
```

Appendix F: Hybrid-Cooled EES Code

```
$UnitSystem SI MASS RAD PA K J
$Tabstops 0.2 0.4 0.6 3.5 in


$Constant N_CHX# =10 "number of nodes in complex HX"
$Constant N_SHX# = 2 "number of nodes in simple HX"

$Constant N_CHX1# = 11 "number of nodes in complex HX plus 1"
$Constant N_SHX1# = 3 "number of nodes in simple HX plus 1"

{$Constant SN#=2                                    "number of nodes in simpler HXs"
$Constant SN1#=3                                    "number of nodes in simpler HXs"
$Constant CN1#=11                                   "number of nodes in complex HXs +1"

$Constant N#=10                                     "number of nodes in complex HXs"
$Constant N1#=3                                     "N+1"
$Constant N2#=4                                     "N+2"
$Constant N22#=6                                    "2*N+2"
$Constant N23#=7                                    "2*N+3"
$Constant N24#=8                                    "2*N+4"
$Constant N34#=10                                   "3*N+4"
$Constant N35#=11                                   "3*N+5"
$Constant N45#=21                                   "4*N+5"
$Constant N46#=22}


$Bookmark aircooler
subprogram
aircooler(HX_area,T_air_in,T_co2_in,m_dot_co2,P_co2_in,L,power_fan:T_air_out,T_co2_out,P_co2_out,q_dot,approach,A_fr,volu
me,m_dot_air)


      P_air =1 [atm]*convert(atm,pa)
{{    L=3 [m]}
      LoverA = 9/20000
      LoverA=(L^2)/HX_area}
      fan_efficiency = 0.5

      N_tubes = A_fr/(0.0524 [m]*0.0498 [m])        "density of tubes per cross sectional area"
      T_air_avg = (T_air_in + T_air_out)/2
      T_co2_avg = (T_co2_in+T_co2_out)/2

      P_co2_avg = (P_Co2_in +P_co2_out)/2

      C_dot_air = CP(air,T=T_air_avg)*m_dot_air
      C_dot_co2=CP(carbondioxide,T=T_co2_avg,p=p_co2_avg)*m_dot_co2

      C_min = min(C_dot_air,C_dot_co2)

      NTU = UA/C_min

      q_dot = c_dot_co2*(T_co2_in-T_co2_out)
      q_dot = c_dot_air*(T_air_out-T_air_in)


      eff = q_dot/q_dot_max

      q_dot_max = C_min*(T_co2_in-T_air_in)

      eff=HX('counterflow', NTU, C_dot_co2, C_dot_air, 'epsilon')

      Call CHX_geom_finned_tube('fc_tubes_sCF-88-10Ja': D_o, fin_pitch, D_h, fin_thk, sigma, alpha, A_fin\A)
      Call CHX_h_finned_tube('fc_tubes_sCF-88-10Ja', m_dot_air, A_fr, 'air',T_air_avg, P_air:h)
      Call CHX_DELTAp_finned_tube('fc_tubes_sCF-88-10Ja', m_dot_air, A_fr,L, 'air', T_air_in, T_air_out, P_air: DELTAp)
```

```
    D_tube_i = 26.01 [mm]*convert(mm,m)

    HX_area = L*A_fr*alpha
    UA  = HX_area*1/((1/h)+(1/h_T))

    rho_air = density('air',P=P_air,T=T_air_avg)
    power_fan = DELTAp*m_dot_air/(rho_air*fan_efficiency)

    "for CO2 pressure drops"
    call PipeFlow('air',T_co2_avg,P_co2_in,m_dot_co2/N_tubes,D_tube_i,L,0:h_T, h_H ,DELTAP_co2, Nusselt_T, f, Re)

    P_co2_out = P_Co2_in  -DELTAP_co2

    approach = T_Co2_out - T_air_in

    volume = A_fr*L
end
$Bookmark Ductflow_N


procedure DuctFlow_N_(Re, Pr, L\D, Aspect, relRough: Nusselt_T,Nusselt_H, f)

    if (Re> 2300) then {turbulent flow}
        Call PipeFlow_Turbulent(Re, Pr, L\D, RelRough : Nusselt_T, f)
        Nusselt_H=Nusselt_T
    else {laminar flow <2300}
        Call DuctFlow_laminar(Re, Pr, L\D, Aspect: Nusselt_T, Nusselt_H, f)
    Endif
end
$Bookmark DuctFlow_

procedure DuctFlow_( Fluid$, T, P, m_dot, H, W, L, RelRough: h_T, h_H, DELTAP, Nusselt_T,  f, Re,V_dot)
{$DuctFlow
This function returns the heat transfer coefficient in [W/m^2-K] and friction factor [-] inside a  rectangular.    Note:  EES must be set to SI units.  T and P must be provided in the units specified in the EES UnitSystem dialog.

Inputs
Fluid$ is a string constant or variable with the name of a fluid in the EES database
T is the temperature (in C or K) that properties are evaluated at.
P is the absolute average pressure of the fluid in the pipe.  The pressure can be Pa, kPa, bar or MPa, depending upon the EES units setting.
m_dot is the flow rate in kg/s.
W is the duct width in m
H is the duct height in m
L is the duct length in m
RelRough is the ratio of the roughness to the tube diameter  <0.05

Outputs  (Only the first output is required)
h_T is the heat transfer coefficient in W/m^2-K assuming that the duct wall is isothermal - this is a lower bound on h
h_H is the heat transfer coefficient in W/m^2-K assuming that the duct wall is exposed to a constant heat flux - this is an upper bound on h
DELTAP is the pressure drop across the tube in the EES pressure setting
Nusselt_T is the Nusselt number corresponding to a constant duct wall temperature
f is the friction factor
Re is Reynold's number

Notes
This function determines Re, Pr and L/D from input data and then calls DuctFlow_N to obtain f and Nusselt}
call IF_unit_check(1:Eng, Uh$,UL$,UMF$,UP$,UT$)
call GetPropsInt(Fluid$,T,P:rho, mu, k, Pr)
D_h=4*W*H/(2*W+2*H)  {hydraulic diameter}
Aspect=H/W
if (Aspect>1) then Aspect=W/H
A=W*H  {cross-sectional area}
Re=(m_dot*D_h)/(A*mu)
{if (Re<=0) then Call Error('Re in DuctFlow must be greater than 0,  The value is XXXA1',Re)}
call DuctFlow_N_(Re,Pr,L/D_h, Aspect,RelRough: Nusselt_T, Nusselt_H, f)
h_T=Nusselt_T*k/D_h  {based on constant temperature boundary}
h_H=Nusselt_H*k/D_h {based on constant heat flux boundary}
```

```
V_dot=m_dot/(A*rho) "velocity"
DELTAP=f*L/D_h*(1/2)*rho*V_dot^2*convert(Pa,UP$)
tree=ENg
END
$Bookmark Channel_Flow
module
channel_flowPH(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DEL
TAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

      T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
      T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"

      call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
      call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

      D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
      D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"


end

module
channel_flowRG(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DE
LTAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

      T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
      T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"

      call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
      call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

      D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
      D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"


end

module
channel_flowPC(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DEL
TAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

      T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
      T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"

      call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
      call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

      D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
```

```
      D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"


end

$Bookmark Compressor
Subprogram Compressor(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                    "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                     "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in                                         "s2s represents the entropy that would exist after
the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=P_out)               "the enthalpy after the compressor, if the
compressor were purely isentropic"
      Ws=h_in-h_outs                                      "the work the compressor requires, if the
compressor were purely isentropic"
      W=Ws/eff                                            "the amount of work required - taking into
account efficiency"
      W=h_in-h_out                                        "the amount of actual work relates to the actual
difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)              "the temperature after the compressor defined
by the actual enthalpy and pressure after the compressor"


end
$Bookmark Turbine
Subprogram Turbine(eff,WF$,T_in,P_in,P_out:T_out,W)

      h_in=enthalpy(WF$,T=T_in,P=P_in)                    "calculate enthalpy pre-compressor from
temp/pressure"
      s_in=entropy(WF$,T=T_in,P=P_in)                     "calculate entropy pre-compressor from
temp/pressure"
      s_outs=s_in                                         "s4s represents the entropy that would exist after
the compressor if it were operating isentropically"
      h_outs=enthalpy(WF$,s=s_outs,P=p_out)               "the enthalpy after the compressor, if the
compressor were purely isentropic"
      Ws=h_in-h_outs                                      "the work the compressor requires, if the
compressor were purely isentropic"
      W=Ws*eff                                            "the amount of work produced - taking into
account efficiency"
      W=h_in-h_out                                        "the amount of actual work relates to the actual
difference in enthalpy"
      T_out=temperature(WF$,P=P_out,h=h_out)              "the temperature after the compressor defined
by the actual enthalpy and pressure after the compressor"

end
$Bookmark PHX
 module
PHX(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_C,W_H,N_CH_C:T_H[1..N_SHX1#],T_C_reverse[1..N_
SHX1#],q_dot,eff_overall,UA_total,delta_T_HC[2],Area,P_C[1..3],Length_HX,H_bar_H,H_bar_C,DELTAP_H,DELTAP_C,V_H[1],V_
C[1],Re_h[1],Re_C[1],Volume,pumping_power_hot,N_CH_H)

th_m=1.9 [mm]*convert(mm,m)                               "thickness of plate"
M$ = 'titanium'
D_C = W_C
D_H = W_H

rough=0


i_H_in=T_H_in*(c_(H$,T=T_H_in) +c_(H$,T=T_H_out))/2       "enthalpy of hot inlet fluid"
i_H_out=T_H_out*(c_(H$,T=T_H_in) +c_(H$,T=T_H_out))/2     "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                            "total heat transfer rate"


i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                     "enthalpy of cold inlet fluid"
```

```
i_C_out=i_C_in+q_dot/m_dot_C                                    "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_SHX1#])                "temperature of cold outlet fluid"

N=N_SHX#   "number of sub-heat exchangers"
N_plus_1=N+1

duplicate i=1,N
      q_dot[i]=i*q_dot/N                                        "total heat transfer rate"
end

                                                                "Obtain temperature distribution"
T_H[1]=T_H_in                                                   "hot-side inlet temperature"
T_C[1]=T_C_out                                                  "cold-side outlet temperature"
i_H[1]=i_H_in                                                   "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                  "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                         "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=i_H[i]/c_(H$,T=T_H[i])                             "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                          "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                  "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                              "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]*(delta_T_H[i]) =m_dot_H*(delta_i_H[i])          "hot-side capacitance rate"
      C_dot_C[i]*(delta_T_C[i]) =m_dot_C*(delta_i_C[i])          "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                             "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N                                     "this is the heat transfer through the node of the
heat exchanger"
      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]     "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))  "effectiveness of sub-heat exchanger"
      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU') "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                   "conductance in sub-heat exchanger"
end

                                                                "determine length of each sub-heat exchanger"
x[1]=1e-5 [m]                                                   "starting position of 1st sub-heat exchanger"


fouling_factor = 0.00035 [m^2-K/W]                              "placeholder value from
http://www.engineeringpage.com/technology/thermal/fouling_factors.html"


duplicate i=1,N
      DELTAx[i]=UA[i]*(fouling_factor+1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_C*N_CH_C) "length of sub-heat exchanger"
      call
channel_flowPH(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,ro
ugh,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])
      k_m[i]=k_(M$, (T_H[i]+T_C[i])/2)                          "metal conductivity at local average temperature"
```

```
    x[i+1]=x[i]+DELTAx[i]

end

duplicate i=1,N-1 "all pressure calculations except the first and last node"

    P_H[i+1] = P_H[i] - DELTA_P_H[i]
    P_C[i+1] = P_C[i] - DELTA_P_C[i]

end

P_H[1]=P_H_in -DELTA_P_H_entrance
P_C[1]=P_C_in - DELTA_P_C_entrance

P_H[N_SHX1#] = P_H[N] - DELTA_P_H[N] - DELTA_P_H_exit
P_C[N_SHX1#] = P_C[N] - DELTA_P_C[N] - DELTA_P_C_exit
C_entrance = 0.5 "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*rho_(H$,T=T_H_in)*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*rho_(H$,T=T_H[N])*V_H[N]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N],P=P_C[N])*V_C[N]^2

duplicate i=1,N

    H_H_weighting[i]=H_H[i]*DELTAx[i]                    "this section calculates weighting factors for
whole-HX calculation"
    H_C_weighting[i]=H_C[i]*DELTAx[i]
end

H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX

DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]

Length_HX = x[N+1]                                       "total length of HX"

UA_total=sum(UA[i],i=1,N)

N_CH_C = N_CH_H*(W_H/W_C)                                "number of channels on cold side defined by
ratio of channel widths"

Volume = N_CH_C*(2*W_C+2*th_m)*(th_m+D_C)*Length_HX      "HX volume"

pumping_power_hot = DELTAP_H*m_dot_H/(rho_(H$,T=(T_H[N+1]+T_H[1])/2))

Area = N_CH_C*D_C*Length_HX

duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end

                                                         "Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - T_C_in*c_(H$,T=T_C_in))

end
$Bookmark Regenerator
module
Regenerator(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_H,W_C,N_CH_H:T_H[1..N_SHX1#],T_C_revers
e[1..N_SHX1#],q_dot,eff_overall,UA_total,delta_T_HC[2],Area,P_H[1..N_SHX1#],P_C[1..N_SHX1#],Length_HX,H_bar_H,H_bar_C,
DELTAP_H,DELTAP_C,V_H[1],V_C[1],Re_h[1],Re_C[2],Volume,N_CH_C)
                                                         "Inputs"
th_m=1.9[mm]*convert(mm,m)                               "thickness of plate"
```

```
M$ = 'titanium'
D_C = W_C
D_H = W_H
rough=0


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                                   "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                                 "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                          "total heat transfer rate"


i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                                   "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                            "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_SHX#+1])                       "temperature of cold outlet fluid"

N=N_SHX#   "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                                "total heat transfer rate"
end

                                                                        "Obtain temperature distribution"
T_H[1]=T_H_in                                                           "hot-side inlet temperature"
T_C[1]=T_C_out                                                          "cold-side outlet temperature"
i_H[1]=i_H_in                                                           "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                          "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                                 "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                          "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                                  "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                          "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                        "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                      "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])                  "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])                  "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                                     "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N                                             "this is the heat transfer through the node of the
heat exchanger"
      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]             "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))       "effectiveness of sub-heat exchanger"
      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU') "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                           "conductance in sub-heat exchanger"
end

                                                                        "determine length of each sub-heat exchanger"
x[1]=1e-5 [m]                                                           "starting position of 1st sub-heat exchanger"
```

```
duplicate i=1,N
      DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_H*N_CH_H)  "length of sub-heat exchanger"
      call
channel_flowRG(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,ro
ugh,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])
      k_m[i]=k_(M$, (T_H[i]+T_C[i])/2)                                    "metal conductivity at local average temperature"
      x[i+1]=x[i]+DELTAx[i]


end


duplicate i=1,N-1 "all pressure calculations except the first and last node"

      P_H[i+1] = P_H[i] - DELTA_P_H[i]
      P_C[i+1] = P_C[i] - DELTA_P_C[i]

end

P_H[1]=P_H_in- DELTA_P_H_entrance
P_C[1]=P_C_in- DELTA_P_C_entrance


P_H[N+1] = P_H[N] - DELTA_P_H[N] -DELTA_P_H_exit
P_C[N+1] = P_C[N] - DELTA_P_C[N] - DELTA_P_C_exit
C_entrance = 0.5 "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*density(H$,T=T_H_in,P=P_H[1])*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*density(H$,T=T_H[N_SHX#],P=P_H[N_SHX#])*V_H[N_SHX#]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N_SHX#],P=P_C[N_SHX#])*V_C[N_SHX#]^2

duplicate i=1,N

      H_H_weighting[i]=H_H[i]*DELTAx[i]                                   "this section calculates weighting factors for
whole-HX calculation"
      H_C_weighting[i]=H_C[i]*DELTAx[i]
end

H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX


DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]

Length_HX = x[N+1]                                                        "total length of HX"

Volume = N_CH_H*(2*W_H+2*th_m)*(th_m+D_H)*Length_HX                       "HX volume"

UA_total=sum(UA[i],i=1,N)

N_CH_C = N_CH_H*(W_H/W_C) "number of channels on cold side defined by ratio of channel widths"

Area = N_CH_H*D_H*Length_HX
duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

                                                                          "Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))
end

$Bookmark Precooler
module
Precooler(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_H,W_C,N_CH_H:T_H[1..N_CHX1#],T_C_reverse[1
```

```
..N_CHX1#],q_dot,eff_overall,UA_total,delta_T_HC[1],Area,P_H[1..N_CHX1#],Length_HX,H_bar_H,H_bar_C,DELTAP_H,DELTAP
_C,V_H[1],V_C[1],Re_H[1],Re_C[1],pumping_power_cold,N_CH_C,volume)
                                                                    "Inputs"
th_m=1.9 [mm]*convert(mm,m)                                         "thickness of plate"
M$ = 'titanium'
D_C = W_C
D_H = W_H
rough=0


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                               "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[N_CHX1#])                       "enthalpy of hot outlet fluid"
q_dot=m_dot_H*(i_H_in-i_H_out)                                      "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=P_C_in)                               "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                        "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_CHX1#])                    "temperature of cold outlet fluid"

N=N_CHX#   "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                            "total heat transfer rate"
end

                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in                                                       "hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                             "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                      "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                              "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],p=p_c[i])                      "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]                                  "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])              "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])              "cold-side capacitance rate"
end

duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1]                                 "by breaking the delta T/delta enthalpy values
into their own variables I can stop them from going to zero and messing up solving"
      q_dot_node[i]=q_dot/N                                         "this is the heat transfer through the node of the
heat exchanger"
      q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]         "this is the maximum amount of heat that can be
transfered through node of the heat exchanger"
      eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))   "effectiveness of sub-heat exchanger"
      NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU') "NTU required by sub-heat exchanger"
      UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                       "conductance in sub-heat exchanger"
end

                                                                    "determine length of each sub-heat exchanger"
x[1]=1e-5 [m]                                                       "starting position of 1st sub-heat exchanger"
```

```
fouling_factor = 0.00035 [m^2-K/W]
duplicate i=1,N
    DELTAx[i]=UA[i]*(fouling_factor+1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_H*N_CH_H)  "length of sub-heat exchanger"
    call
channel_flowPC(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,ro
ugh,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])
    k_m[i]=k_(M$, (T_H[i]+T_C[i])/2)                                    "metal conductivity at local average temperature"
    x[i+1]=x[i]+DELTAx[i]


end

duplicate i=1,N-1                                                       "all pressure calculations except the first and last
node"

    P_H[i+1] = P_H[i] -DELTA_P_H[i]
    P_C[i+1] = P_C[i] - DELTA_P_C[i]

end


P_H[1]=P_H_in - DELTA_P_H_entrance
P_C[1]=P_C_in - DELTA_P_C_entrance


P_H[N_CHX1#] = P_H[N_CHX#] - DELTA_P_H[N_CHX#] -DELTA_P_H_exit
P_C[N_CHX1#] = P_C[N_CHX#] - DELTA_P_C[N_CHX#] - DELTA_P_C_exit
C_entrance = 0.5                                                        "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*density(H$,T=T_H_in,P=P_H[1])*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*density(H$,T=T_H[N_CHX#],P=P_H[N_CHX#])*V_H[N_CHX#]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N_CHX#],P=P_C[N_CHX#])*V_C[N_CHX#]^2


duplicate i=1,N

    H_H_weighting[i]=H_H[i]*DELTAx[i]                                   "this section calculates weighting factors for
whole-HX calculation"
    H_C_weighting[i]=H_C[i]*DELTAx[i]
end

H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX


DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]

Length_HX = x[N+1]                                                      "total length of HX"


UA_total=sum(UA[i],i=1,N)

Area = N_CH_H*D_H*Length_HX

N_CH_C = N_CH_H*(W_C/W_H) "number of channels on cold side defined by ratio of channel widths"

duplicate i=1,N+1
    T_C_reverse[i]=T_C[N+2-i]
end
                                                                        "Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))

Volume = N_CH_C*(2*W_C+2*th_m)*(th_m+D_C)*Length_HX                     "HX volume"
pumping_power_cold= DELTAP_C*m_dot_C/density(C$,T=(T_C[N+1]+T_C[1])/2,P=(P_C[1]+P_c[N+1])/2)

end
```

```
$bookmark Main
                                                    "Overall Program Control"
                                                    "Constants"
C$='water'                                          "cooling fluid"
H$='Salt (60% NaNO3, 40% KNO3)'                     "heating fluid"
WF$='carbondioxide'                                 "working fluid"

N1 = 1                                              "Regenerator Cold Side Inlet"
N2 = 1+N_SHX#                                       "Regenerator Cold Side outlet"
N3 = 2+N_SHX#                                       "PHX Inlet"
N4 = 2+2*N_SHX#                                     "PHX Outlet/Turbine Inlet"
N5 = 3+2*N_SHX#                                     "Turbine Outlet"
N6 = 4+2*N_SHX#                                     "Regenerator Hot Side inlet"
N7 = 4+3*N_SHX#                                     "Regenerator Hot Side Outlet"
N8 = 5+ 3*N_SHX#                                    "aircooler outlet"
N9 =6+ 3*N_SHX#                                     "Precooler Inlet"
N10 = 6+ 3*N_SHX#+N_CHX#                            "Precooler Outlet/Compressor inlet"
N11 = 7+ 3*N_SHX#+N_CHX#                            "Compressor Outlet"
N_PC_1 = N_CHX#+1


channel_length_ratio_PHX =(N_ch_PHX_C/Length_PHX)*(m_dot_wf_scaled/m_dot_wf)"ratio of number of channels in hx to length
of hx - fixing this value fixed aspect ratio of hx as hx area changes"
channel_length_ratio_RG =(N_ch_RG_C/Length_RG)*(m_dot_wf_scaled/m_dot_wf)
channel_length_ratio_PC =(N_ch_PC_C/Length_PC)*(m_dot_wf_scaled/m_dot_wf)

channel_length_ratio_PC=10000                       "opt"
channel_length_ratio_PHX=10000                      "not OPT"
channel_length_ratio_RG=4000                        " not OPT"



fraction_pc = 0.3                                   " OPT"
fraction_rg = 0.45                                  " OPT"


volume_nonaircooler = (volume_rg+volume_pc+volume_Phx)*(m_dot_wf_scaled/m_dot_wf) "total volume of heat exchangers -
neglecting aircooler"

volume_ratio = volume_aircooler/(volume_nonaircooler+volume_aircooler) "ratio of air cooler volume to overall HX volume"

T_WF[N7] = T_AC_in                                  "inlet temp to aircooler"
T_WF[N8] = T_PC_in                                  "inlet temp to precooler"
T_WF[N8] = max(T_WF[N10]+approach,T_air_in+approach)
{T_WF[N8]=330}
power_aircool=50000[W] "fixed fanpower"


T_WF[N10] = 307.5
m_dot_blend = 0
L_aircooler=1.034 "for medium (16000m2) aircooler size"
$ifnot ParametricTable
{T_C_in=300 "cooling fluid inlet temp"}
T_H_in   =970 [k]                                   "heating fluid inlet temp"
m_dot_cool_scaled=500 [kg/s]
m_dot_h_scaled=150 [kg/s]
T_air_in = 270 [K]
HX_area_aircooler=16000 [m^2]
area_total_scaled =5250
$endif
P_high=25000000 [pa]                                "high pressure point"
{P_low=5000000 [pa]}                                "low pressure point"
P_ratio=2.9
m_dot_normalized = 1
W_channel_PHX_C=0.0023 [m]                          "all channels opt"
W_channel_PHX_H=0.0007119[m]
W_channel_RG_C=0.0022[m]
W_channel_RG_H=0.0018 [m]
```

```
W_channel_PC_C=0.002[m]
W_channel_PC_H=0.001442 [m]

h_C_in = enthalpy(water,T=T_C_in, P=P_C) "inlet enthalpy of water from cooling tower"
h_C_out = enthalpy(water,T=T_C[N_CHX1#],P=P_C) "enthalpy of water exiting cycle"
h_C_cool = enthalpy(water,T=T_C_cool,P=P_C) "enthalpy of water entering cycle"
m_dot_cool_scaled=m_dot_tower+m_dot_blend "cooling water mass flow rate to precooler is the sum of mass flow rate from the cooling tower and from the blended back precooler outlet water"
m_dot_cool_scaled*h_C_cool=m_dot_blend*h_c_out+m_dot_tower*h_c_in "energy balance on water at blending point"


q_air_ratio=q_dot_aircooler/(q_dot_aircooler+q_dot_cooling*100) "ratio of air cooling to overall cooling"


P_H=10000000[Pa]                                                    "hot fluid pressure"
P_C=10000000[Pa]                                                    "cold fluid pressure"
P_ratio=P_high/P_low


m_dot_wf=m_dot_wf_design                                            "mass flow rate of working fluid"
flowrate_compressor_inlet=0.15 [m^3/s]

T2=T_WF[N11]
T5=T_WF[N5]

turbine_eff=.9                                                      "turbine efficiency"
comp_eff=.89"compressor efficiency"

fraction_PHX_opt = area_PHX/area_total                              "for optimization - these values can't go out of physical bounds"
fraction_PC_opt = area_PC/(area_total-Area_PHX)                     "for optimization - these values can't go out of physical bounds"


fraction_PHX = area_PHX/area_total
fraction_RG = area_RG/area_total
fraction_PC=area_PC/area_total

area_total = area_PC+area_phx+area_RG
area_total_scaled = area_total*(m_dot_wf_scaled/m_dot_wf)

UA_Total = UA_regenerator+UA_precooler+UA_PHX

DELTA_T_salt = T_H_in - T_H_out

Call
PHX(H$,T_H_in,T_H_out,P_H,WF$,m_dot_h,T_WF[N2],P_WF[N2],m_dot_wf,W_channel_PHX_C,W_channel_PHX_H,N_ch_PHX_C:T_H[N1..N2],T_WF[N3..N4],q_dot,eff_h,UA_PHX,T_PHX_Pinch,Area_PHX,P_WF[N3..N4],Length_PHX,H_bar_H_PHX,H_bar_C_PHX,DELTAP_H_PHX,DELTAP_C_PHX,V_PHX_H,V_PHX_C,Re_PHX_H,Re_PHX_C,Volume_PHX,pumping_power_salt,N_ch_PHX_H) "this HX represents heat transfer from heat source to WF after regenerator"

Call
Regenerator(WF$,T5,T6,P_low,WF$,m_dot_wf,T2,p_high,m_dot_wf,W_channel_RG_H,W_channel_RG_C,N_ch_RG_H:T_WF[N6..N7],T_WF[1..N2],q_dot_reg,eff_r,UA_regenerator,T_RG_Pinch,Area_RG,P_WF[N6..N7],P_WF[1..N2],Length_RG,H_bar_H_RG,H_bar_C_RG,DELTAP_H_RG,DELTAP_C_RG,V_RG_H,V_RG_C,Re_RG_H,Re_RG_C,Volume_RG,N_ch_RG_C)"this heat exchanger represents the regenerator"

Call Turbine (turbine_eff,WF$,T_WF[N4],P_WF[N4],P_low:T_WF[N5],W_turbine) "this turbine extracts energy from the working fluid"

Call
aircooler(HX_area_aircooler,T_air_in,T_WF[N7],m_dot_wf_scaled,P_WF[N7],L_aircooler,power_aircool:T_air_out,T_WF[N8],P_WF[N8],q_dot_aircooler,air_approach,A_fr_aircooler,volume_aircooler,m_dot_air)

Call
Precooler(WF$,T_WF[N8],T1,P_WF[N8],C$,m_dot_wf,T_C_cool,P_C,m_dot_cool,W_channel_PC_H,W_channel_PC_C,N_ch_PC_H:T_WF[N9..N10],T_C[1..N_PC_1],q_dot_cooling,eff_c,UA_precooler,T_PC_Pinch,Area_PC,P_WF[N9..N10],Length_PC,H_bar_H_PC,H_bar_C_PC,DELTAP_H_PC,DELTAP_C_PC,V_PC_H,V_PC_C,Re_PC_H,Re_PC_C,pumping_power_water,N_ch_PC_C,volume_pc) "this heat exchanger involves the working fluid after the turbine and the cooling fluid from the cooling tower or whatnot"
```

```
Call Compressor(comp_eff,WF$,T_WF[N10],P_WF[N10],P_high:T_WF[N11],W_comp) "the compressor takes the fluid from the heat rejection heat exchanger to the regenerator"

work_net=m_dot_wf*(W_turbine+W_comp) - pumping_power_water - pumping_power_salt - power_aircool*(m_dot_wf/m_dot_wf_scaled) "net work done by the cycle"

efficiency=work_net/q_dot "this section of code records the efficiency of the cycle in terms of turbine/compressor work and heat transfer to the system in the primary heat exchanger"


comp_inlet_density=density(WF$,T=T_WF[N10],P=P_WF[N10])                    {working fluid density at compressor inlet}
flowrate_compressor_inlet = (m_dot_wf/comp_inlet_density)*(m_dot_wf_scaled/1 [kg/s])  "volumetric flowrate of working fluid at compressor inlet"

duplicate i=1,N4                                                           {this section sets up the array recording the entropy of the working fluid at each point}
      s[i]=entropy(WF$,T=T_WF[i],P=P_high)
      h[i]=enthalpy(WF$,T=T_WF[i],P=P_high)
end

duplicate i=N5,N10
      s[i]=entropy(WF$,T=T_WF[i],P=P_low)
h[i]=enthalpy(WF$,T=T_WF[i],P=P_high)
end

s[N11]=entropy(WF$,T=T_WF[N11],P=P_high)
h[N11]=enthalpy(WF$,T=T_WF[N11],P=P_high)


"PLANT SIZE SCALING"

Power_scaled=m_dot_wf_scaled*work_net/m_dot_wf
turbine_power_scaled = w_turbine*m_dot_wf_scaled
comp_power_scaled=w_comp*m_dot_wf_scaled
m_dot_wf_scaled=100*m_dot_wf
area_phx_scaled=m_dot_wf_scaled*area_phx/m_dot_wf
area_rg_scaled=m_dot_wf_scaled*area_rg/m_dot_wf
area_pc_scaled=m_dot_wf_scaled*area_pc/m_dot_wf

pumping_power_water_scaled = pumping_power_water*(m_dot_wf_scaled/m_dot_wf)
pumping_power_salt_scaled = pumping_power_salt*(m_dot_wf_scaled/m_dot_wf)

m_dot_cool_scaled = m_dot_cool*(m_dot_wf_scaled/m_dot_wf)
m_dot_h_scaled = m_dot_h*(m_dot_wf_scaled/m_dot_wf)

T_C_out = T_C[N_CHX1#]
```

Appendix G: Air-Cooled EES Code

```
$UnitSystem SI MASS RAD PA K J
$Tabstops 0.2 0.4 0.6 3.5 in
$Constant N_CHX# =10 "number of nodes in complex HX"
$Constant N_SHX# = 2 "number of nodes in simple HX"
$Constant N_CHX1# = 11 "number of nodes in complex HX plus 1"
$Constant N_SHX1# = 3 "number of nodes in simple HX plus 1"

$Bookmark aircooler
subprogram
aircooler(HX_area,T_air_in,T_co2_in,m_dot_air,m_dot_co2,P_co2_in,L:T_air_out,T_co2_out,P_co2_out,power_fan,q_dot,approach
,A_fr,volume)


      P_air =1 [atm]*convert(atm,pa)
{     L=3 [m]}
{     LoverA = 9/20000
      LoverA=(L^2)/HX_area}
      fan_efficiency = 0.5

      N_tubes = A_fr/(0.0524 [m]*0.0498 [m]) "density of tubes per cross sectional area"
      T_air_avg = (T_air_in + T_air_out)/2
      T_co2_avg = (T_co2_in+T_co2_out)/2

      P_co2_avg = (P_Co2_in +P_co2_out)/2

      C_dot_air = CP(air,T=T_air_avg)*m_dot_air
      C_dot_co2=CP(carbondioxide,T=T_co2_avg,p=p_co2_avg)*m_dot_co2

      C_min = min(C_dot_air,C_dot_co2)

      NTU = UA/C_min

      q_dot = c_dot_co2*(T_co2_in-T_co2_out)
      q_dot = c_dot_air*(T_air_out-T_air_in)


      eff = q_dot/q_dot_max

      q_dot_max = C_min*(T_co2_in-T_air_in)

      eff=HX('counterflow', NTU, C_dot_co2, C_dot_air, 'epsilon')

      Call CHX_geom_finned_tube('fc_tubes_sCF-88-10Ja': D_o, fin_pitch, D_h, fin_thk, sigma, alpha, A_fin\A)
      Call CHX_h_finned_tube('fc_tubes_sCF-88-10Ja', m_dot_air, A_fr, 'air',T_air_avg, P_air:h)
      Call CHX_DELTAp_finned_tube('fc_tubes_sCF-88-10Ja', m_dot_air, A_fr,L, 'air', T_air_in, T_air_out, P_air: DELTAp)

      D_tube_i = 26.01 [mm]*convert(mm,m)

      HX_area = L*A_fr*alpha
      UA  = HX_area*h

      rho_air = density('air',P=P_air,T=T_air_avg)
      power_fan = DELTAp*m_dot_air/(rho_air*fan_efficiency)

      "for CO2 pressure drops"
      call PipeFlow('carbondioxide',T_co2_avg,P_co2_in,m_dot_co2/N_tubes,D_tube_i,L,0:h_T, h_H ,DELTAP_co2, Nusselt_T, f,
Re)

      P_co2_out = P_Co2_in  -DELTAP_co2

      approach = T_Co2_out - T_air_in

      volume = A_fr*L
end
```

```
$Bookmark aircooler2
subprogram
aircooler2(HX_area,T_air_in,T_co2_in,m_dot_air,m_dot_co2,P_co2_in,L:T_air_out,T_co2_out,P_co2_out,power_fan,q_dot,approac
h,A_fr,volume)


      P_air =1 [atm]*convert(atm,pa)
{     L=3 [m]}
{     LoverA = 9/20000
      LoverA=(L^2)/HX_area}
      fan_efficiency = 0.5

      N_tubes = A_fr/(0.0524 [m]*0.0498 [m])                    "density of tubes per cross sectional area"
      T_air_avg = (T_air_in + T_air_out)/2
      T_co2_avg = (T_co2_in+T_co2_out)/2

      P_co2_avg = (P_Co2_in +P_co2_out)/2

      C_dot_air = CP(air,T=T_air_avg)*m_dot_air
      C_dot_co2=CP(carbondioxide,T=T_co2_avg,p=p_co2_avg)*m_dot_co2

      C_min = min(C_dot_air,C_dot_co2)

      NTU = UA/C_min

      q_dot = c_dot_co2*(T_co2_in-T_co2_out)
      q_dot = c_dot_air*(T_air_out-T_air_in)


      eff = q_dot/q_dot_max

      q_dot_max = C_min*(T_co2_in-T_air_in)

      eff=HX('counterflow', NTU, C_dot_co2, C_dot_air, 'epsilon')

      Call CHX_geom_finned_tube('fc_tubes_sCF-88-10Ja': D_o, fin_pitch, D_h, fin_thk, sigma, alpha, A_fin\A)
      Call CHX_h_finned_tube('fc_tubes_sCF-88-10Ja', m_dot_air, A_fr, 'air',T_air_avg, P_air:h)
      Call CHX_DELTAp_finned_tube('fc_tubes_sCF-88-10Ja', m_dot_air, A_fr,L, 'air', T_air_in, T_air_out, P_air: DELTAp)

      D_tube_i = 26.01 [mm]*convert(mm,m)

      HX_area = L*A_fr*alpha
      UA  = HX_area*h

      rho_air = density('air',P=P_air,T=T_air_avg)
      power_fan = DELTAp*m_dot_air/(rho_air*fan_efficiency)

      "for CO2 pressure drops"
      call PipeFlow('carbondioxide',T_co2_avg,P_co2_in,m_dot_co2/N_tubes,D_tube_i,L,0:h_T, h_H ,DELTAP_co2, Nusselt_T, f,
Re)

      P_co2_out = P_Co2_in  -DELTAP_co2

      approach = T_Co2_out - T_air_in

      volume = A_fr*L
end

$Bookmark aircooler3
subprogram
aircooler3(HX_area,T_air_in,T_co2_in,m_dot_air,m_dot_co2,P_co2_in,L:T_air_out,T_co2_out,P_co2_out,power_fan,q_dot,approac
h,A_fr,volume)


      P_air =1 [atm]*convert(atm,pa)
{     L=3 [m]}
{     LoverA = 9/20000
      LoverA=(L^2)/HX_area}
      fan_efficiency = 0.5
```

```
    N_tubes = A_fr/(0.0524 [m]*0.0498 [m])                    "density of tubes per cross sectional area"
    T_air_avg = (T_air_in + T_air_out)/2
    T_co2_avg = (T_co2_in+T_co2_out)/2

    P_co2_avg = (P_Co2_in +P_co2_out)/2
    C_dot_air = CP(air,T=T_air_avg)*m_dot_air
    C_dot_co2=CP(carbondioxide,T=T_co2_avg,p=p_co2_avg)*m_dot_co2
    C_min = min(C_dot_air,C_dot_co2)

    NTU = UA/C_min

    q_dot = c_dot_co2*(T_co2_in-T_co2_out)
    q_dot = c_dot_air*(T_air_out-T_air_in)


    eff = q_dot/q_dot_max

    q_dot_max = C_min*(T_co2_in-T_air_in)

    eff=HX('counterflow', NTU, C_dot_co2, C_dot_air, 'epsilon')

    Call CHX_geom_finned_tube('fc_tubes_sCF-88-10Ja': D_o, fin_pitch, D_h, fin_thk, sigma, alpha, A_fin\A)
    Call CHX_h_finned_tube('fc_tubes_sCF-88-10Ja', m_dot_air, A_fr, 'air',T_air_avg, P_air:h)
    Call CHX_DELTAp_finned_tube('fc_tubes_sCF-88-10Ja', m_dot_air, A_fr,L, 'air', T_air_in, T_air_out, P_air: DELTAp)

    D_tube_i = 26.01 [mm]*convert(mm,m)

    HX_area = L*A_fr*alpha
    UA  = HX_area*h

    rho_air = density('air',P=P_air,T=T_air_avg)
    power_fan = DELTAp*m_dot_air/(rho_air*fan_efficiency)

    "for CO2 pressure drops"
    call PipeFlow('carbondioxide',T_co2_avg,P_co2_in,m_dot_co2/N_tubes,D_tube_i,L,0:h_T, h_H ,DELTAP_co2, Nusselt_T, f,
Re)

    P_co2_out = P_Co2_in  -DELTAP_co2

    approach = T_Co2_out - T_air_in

    volume = A_fr*L
end

$Bookmark Ductflow_N


procedure DuctFlow_N_(Re, Pr, L\D, Aspect, relRough: Nusselt_T,Nusselt_H, f)

    if (Re> 2300) then {turbulent flow}
        Call PipeFlow_Turbulent(Re, Pr, L\D, RelRough : Nusselt_T, f)
        Nusselt_H=Nusselt_T
    else {laminar flow <2300}
        Call DuctFlow_laminar(Re, Pr, L\D, Aspect: Nusselt_T, Nusselt_H, f)
    Endif
end
$Bookmark DuctFlow_

procedure DuctFlow_( Fluid$, T, P, m_dot, H, W, L, RelRough: h_T, h_H, DELTAP, Nusselt_T,  f, Re,V_dot)
{$DuctFlow
This function returns the heat transfer coefficient in [W/m^2-K] and friction factor [-] inside a  rectangular.    Note:  EES must be set
to SI units.  T and P must be provided in the units specified in the EES UnitSystem dialog.

Inputs
Fluid$ is a string constant or variable with the name of a fluid in the EES database
T is the temperature (in C or K) that properties are evaluated at.
P is the absolute average pressure of the fluid in the pipe.  The pressure can be Pa, kPa, bar or MPa, depending upon the EES
units setting.
m_dot is the flow rate in kg/s.
```

```
W is the duct width in m
H is the duct height in m
L is the duct length in m
RelRough is the ratio of the roughness to the tube diameter  <0.05

Outputs  (Only the first output is required)
h_T is the heat transfer coefficient in W/m^2-K assuming that the duct wall is isothermal - this is a lower bound on h
h_H is the heat transfer coefficient in W/m^2-K assuming that the duct wall is exposed to a constant heat flux - this is an upper
bound on h
DELTAP is the pressure drop across the tube in the EES pressure setting
Nusselt_T is the Nusselt number corresponding to a constant duct wall temperature
f is the friction factor
Re is Reynold's number

Notes
This function determines Re, Pr and L/D from input data and then calls DuctFlow_N to obtain f and Nusselt}
call IF_unit_check(1:Eng, Uh$,UL$,UMF$,UP$,UT$)
call GetPropsInt(Fluid$,T,P:rho, mu, k, Pr)
D_h=4*W*H/(2*W+2*H)  {hydraulic diameter}
Aspect=H/W
if (Aspect>1) then Aspect=W/H
A=W*H  {cross-sectional area}
Re=(m_dot*D_h)/(A*mu)
{if (Re<=0) then Call Error('Re in DuctFlow must be greater than 0,  The value is XXXA1',Re)}
call DuctFlow_N_(Re,Pr,L/D_h, Aspect,RelRough: Nusselt_T, Nusselt_H, f)
h_T=Nusselt_T*k/D_h  {based on constant temperature boundary}
h_H=Nusselt_H*k/D_h {based on constant heat flux boundary}
V_dot=m_dot/(A*rho) "velocity"
{if (Eng=0) then DELTAP=f*L/D_h*(1/2)*rho*V_dot^2*convert(Pa,UP$) else DELTAP=f*L/D_h*(1/2)*rho*V_dot^2*convert(lbm/ft-
hr^2,UP$)}
DELTAP=f*L/D_h*(1/2)*rho*V_dot^2*convert(Pa,UP$)
tree=ENg
END
$Bookmark Channel_Flow
module
channel_flowPH(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DEL
TAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

      T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
      T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"

      call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
      call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

      D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
      D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"


end

module
channel_flowRG(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DE
LTAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

      T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
      T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"
```

```
    call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
    call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

    D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
    D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"



end

module
channel_flowPC(H$,C$,M$,L,W_C,W_H,D_C,D_H,T_H_B,T_C_B,P_H,P_C,th,m_dot_H,m_dot_C,rough_H,rough_C:H_H,H_C,DEL
TAP_H,DELTAP_C,V_h,V_c,Re_h,Re_C,f_h,f_c)

T_wall_avg=(T_H_B+T_C_B)/2
T_H_W=T_wall_avg
T_C_W=T_wall_avg

    T_H_film = (T_H_B+T_H_W)/2 "the film temperature for the hot and cold side are calculated as the average of wall and bulk
temperatures, and used for ductflow to calculate..."
    T_C_film = (T_C_B+T_C_W)/2 "...fluid properties"

    call DuctFlow_(H$,T_H_film,P_H,m_dot_H,D_H,W_H,L,Rough_H:h_T_h, h_H ,DELTAP_h, Nu_T_h,f_H, Re_h,V_h) "hot side
ductflow call"
    call DuctFlow_(C$,T_C_film,P_C,m_dot_C,D_C,W_C,L,Rough_C:h_T_c, h_C ,DELTAP_c, Nu_c_h,f_c, Re_c,V_c) "cool side
ductflow call"

    D_H_h = 4*(W_H*D_H)/(2*(W_H+D_H)) "hydraulic diameter of hot side"
    D_H_c = 4*(W_c*D_C)/(2*(W_C+D_C)) "hydraulic diameter of cold side"



end

$Bookmark Compressor
Subprogram Compressor(eff,WF$,T_in,P_in,P_out:T_out,W)

    h_in=enthalpy(WF$,T=T_in,P=P_in)                    "calculate enthalpy pre-compressor from
temp/pressure"
    s_in=entropy(WF$,T=T_in,P=P_in)                     "calculate entropy pre-compressor from
temp/pressure"
    s_outs=s_in                                         "s2s represents the entropy that would exist after
the compressor if it were operating isentropically"
    h_outs=enthalpy(WF$,s=s_outs,P=P_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
    Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
    W=Ws/eff                                            "the amount of work required - taking into
account efficiency"
    W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
    T_out=temperature(WF$,P=P_out,h=h_out) "the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"


end
$Bookmark Turbine
Subprogram Turbine(eff,WF$,T_in,P_in,P_out:T_out,W)

    h_in=enthalpy(WF$,T=T_in,P=P_in)                    "calculate enthalpy pre-compressor from
temp/pressure"
    s_in=entropy(WF$,T=T_in,P=P_in)                     "calculate entropy pre-compressor from
temp/pressure"
    s_outs=s_in "s4s represents the entropy that would exist after the compressor if it were operating isentropically"
    h_outs=enthalpy(WF$,s=s_outs,P=p_out) "the enthalpy after the compressor, if the compressor were purely isentropic"
    Ws=h_in-h_outs "the work the compressor requires, if the compressor were purely isentropic"
    W=Ws*eff "the amount of work produced - taking into account efficiency"
    W=h_in-h_out "the amount of actual work relates to the actual difference in enthalpy"
    T_out=temperature(WF$,P=P_out,h=h_out) "the temperature after the compressor defined by the actual enthalpy and pressure
after the compressor"
```

```
end
$Bookmark PHX
 module
PHX(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_C,W_H,N_CH_C:T_H[1..N_SHX1#],T_C_reverse[1..N_
SHX1#],q_dot,eff_overall,UA_total,delta_T_HC[2],Area,P_C[1..3],Length_HX,H_bar_H,H_bar_C,DELTAP_H,DELTAP_C,V_H[1],V_
C[1],Re_h[1],Re_C[1],Volume,pumping_power_hot,N_CH_H)

th_m=1.9 [mm]*convert(mm,m)                                   "thickness of plate"
M$ = 'titanium'
D_C = W_C
D_H = W_H

rough=0


i_H_in=T_H_in*(c_(H$,T=T_H_in) +c_(H$,T=T_H_out))/2           "enthalpy of hot inlet fluid"
i_H_out=T_H_out*(c_(H$,T=T_H_in) +c_(H$,T=T_H_out))/2         "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                "total heat transfer rate"


i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                         "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                  "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_SHX1#])              "temperature of cold outlet fluid"

N=N_SHX#                                                      "number of sub-heat exchangers"
N_plus_1=N+1

duplicate i=1,N
      q_dot[i]=i*q_dot/N                                      "total heat transfer rate"
end

                                                              "Obtain temperature distribution"
T_H[1]=T_H_in                                                 "hot-side inlet temperature"
T_C[1]=T_C_out                                                "cold-side outlet temperature"
i_H[1]=i_H_in                                                 "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                       "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=i_H[i]/c_(H$,T=T_H[i])                           "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                        "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                              "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]   "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]*(delta_T_H[i]) =m_dot_H*(delta_i_H[i])        "hot-side capacitance rate"
      C_dot_C[i]*(delta_T_C[i]) =m_dot_C*(delta_i_C[i])        "cold-side capacitance rate"
end



duplicate i=1,N
      delta_T_HC[i]=T_H[i]-T_C[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from
going to zero and messing up solving"
      q_dot_node[i]=q_dot/N "this is the heat transfer through the node of the heat exchanger"
```

```
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]                    "this is the maximum amount of heat that can be transfered through node of the heat exchanger"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))              "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU') "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])                                  "conductance in sub-heat exchanger"
end

                                                                             "determine length of each sub-heat exchanger"
x[1]=1e-5 [m]                                                                "starting position of 1st sub-heat exchanger"


fouling_factor = 0.00035 [m^2-K/W]


duplicate i=1,N
    DELTAx[i]=UA[i]*(fouling_factor+1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_C*N_CH_C) "length of sub-heat exchanger"
    call channel_flowPH(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,rough,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])
    k_m[i]=k_(M$, (T_H[i]+T_C[i])/2)                                         "metal conductivity at local average temperature"
    x[i+1]=x[i]+DELTAx[i]
end


duplicate i=1,N-1 "all pressure calculations except the first and last node"

    P_H[i+1] = P_H[i] - DELTA_P_H[i]
    P_C[i+1] = P_C[i]  - DELTA_P_C[i]

end

P_H[1]=P_H_in -DELTA_P_H_entrance
P_C[1]=P_C_in - DELTA_P_C_entrance

P_H[N_SHX1#] = P_H[N] - DELTA_P_H[N] - DELTA_P_H_exit
P_C[N_SHX1#] = P_C[N] - DELTA_P_C[N] - DELTA_P_C_exit
C_entrance = 0.5 "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*rho_(H$,T=T_H_in)*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*rho_(H$,T=T_H[N])*V_H[N]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N],P=P_C[N])*V_C[N]^2

duplicate i=1,N

    H_H_weighting[i]=H_H[i]*DELTAx[i] "this section calculates weighting factors for whole-HX calculation"
    H_C_weighting[i]=H_C[i]*DELTAx[i]
end

H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX

DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]


Length_HX = x[N+1]                                                           "total length of HX"


UA_total=sum(UA[i],i=1,N)

N_CH_C = N_CH_H*(W_H/W_C) "number of channels on cold side defined by ratio of channel widths"

Volume = N_CH_C*(2*W_C+2*th_m)*(th_m+D_C)*Length_HX                          "HX volume"

pumping_power_hot = DELTAP_H*m_dot_H/(rho_(H$,T=(T_H[N+1]+T_H[1])/2))

Area = N_CH_C*D_C*Length_HX
```

```
duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - T_C_in*c_(H$,T=T_C_in))

end
$Bookmark Regenerator
module
Regenerator(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_H,W_C,N_CH_H:T_H[1..N_SHX1#],T_C_revers
e[1..N_SHX1#],q_dot,eff_overall,UA_total,delta_T_HC[2],Area,P_H[1..N_SHX1#],P_C[1..N_SHX1#],Length_HX,H_bar_H,H_bar_C,
DELTAP_H,DELTAP_C,V_H[1],V_C[1],Re_h[1],Re_C[2],Volume,N_CH_C)

th_m=1.9[mm]*convert(mm,m)                                          "thickness of plate"
M$ = 'titanium'
D_C = W_C
D_H = W_H
rough=0


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                               "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[1])                             "enthalpy of hot outlet fluid"

q_dot=m_dot_H*(i_H_in-i_H_out)                                      "total heat transfer rate"


i_C_in=enthalpy(C$,T=T_C_in,P=p_C[1])                               "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                        "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_SHX#+1])                   "temperature of cold outlet fluid"

N=N_SHX#                                                            "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                            "total heat transfer rate"
end

                                                                    "Obtain temperature distribution"
T_H[1]=T_H_in                                                       "hot-side inlet temperature"
T_C[1]=T_C_out                                                      "cold-side outlet temperature"
i_H[1]=i_H_in                                                       "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                      "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                             "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                      "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                              "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],P=p_C[i])                      "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                                    "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]  "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])              "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])              "cold-side capacitance rate"
end
```

```
duplicate i=1,N
    delta_T_HC[i]=T_H[i]-T_C[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]  "this is the maximum amount of heat that can be transfered through node of the heat exchanger"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))  "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])  "conductance in sub-heat exchanger"
end

                                                  "determine length of each sub-heat exchanger"
x[1]=1e-5 [m]                                     "starting position of 1st sub-heat exchanger"


duplicate i=1,N
    DELTAx[i]=UA[i]*(1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_H*N_CH_H)  "length of sub-heat exchanger"
    call channel_flowRG(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,rough,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])
    k_m[i]=k_(M$, (T_H[i]+T_C[i])/2)  "metal conductivity at local average temperature"
    x[i+1]=x[i]+DELTAx[i]


end


duplicate i=1,N-1 "all pressure calculations except the first and last node"

    P_H[i+1] = P_H[i] - DELTA_P_H[i]
    P_C[i+1] = P_C[i] - DELTA_P_C[i]

end

P_H[1]=P_H_in- DELTA_P_H_entrance
P_C[1]=P_C_in- DELTA_P_C_entrance


P_H[N+1] = P_H[N] - DELTA_P_H[N] -DELTA_P_H_exit
P_C[N+1] = P_C[N] - DELTA_P_C[N] - DELTA_P_C_exit
C_entrance = 0.5 "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*density(H$,T=T_H_in,P=P_H[1])*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*density(H$,T=T_H[N_SHX#],P=P_H[N_SHX#])*V_H[N_SHX#]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N_SHX#],P=P_C[N_SHX#])*V_C[N_SHX#]^2

duplicate i=1,N

    H_H_weighting[i]=H_H[i]*DELTAx[i] "this section calculates weighting factors for whole-HX calculation"
    H_C_weighting[i]=H_C[i]*DELTAx[i]
end

H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX


DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]

Length_HX = x[N+1]                                "total length of HX"

Volume = N_CH_H*(2*W_H+2*th_m)*(th_m+D_H)*Length_HX  "HX volume"

UA_total=sum(UA[i],i=1,N)
```

```
N_CH_C = N_CH_H*(W_H/W_C) "number of channels on cold side defined by ratio of channel widths"

Area = N_CH_H*D_H*Length_HX
duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))
end

$Bookmark Precooler
module
Precooler(H$,T_H_in,T_H_out,P_H_in,C$,m_dot_h,T_C_in,P_C_in,m_dot_c,W_H,W_C,N_CH_H:T_H[1..N_CHX1#],T_C_reverse[1
..N_CHX1#],q_dot,eff_overall,UA_total,delta_T_HC[1],Area,P_H[1..N_CHX1#],Length_HX,H_bar_H,H_bar_C,DELTAP_H,DELTAP
_C,V_H[1],V_C[1],Re_H[1],Re_C[1],pumping_power_cold,N_CH_C,volume)
th_m=1.9 [mm]*convert(mm,m)                                    "thickness of plate"
M$ = 'titanium'
D_C = W_C
D_H = W_H
rough=0


i_H_in=enthalpy(H$,T=T_H_in,P=p_H[1])                          "enthalpy of hot inlet fluid"
i_H_out=enthalpy(H$,T=T_H_out,P=p_H[N_CHX1#])                  "enthalpy of hot outlet fluid"
q_dot=m_dot_H*(i_H_in-i_H_out)                                 "total heat transfer rate"

i_C_in=enthalpy(C$,T=T_C_in,P=P_C_in)                          "enthalpy of cold inlet fluid"
i_C_out=i_C_in+q_dot/m_dot_C                                   "enthalpy of cold outlet fluid"
T_C_out=temperature(C$,h=i_C_out,P=p_C[N_CHX1#])               "temperature of cold outlet fluid"

N=N_CHX#   "number of sub-heat exchangers"
duplicate i=1,N
      q_dot[i]=i*q_dot/N                                       "total heat transfer rate"
end

                                                               "Obtain temperature distribution"
T_H[1]=T_H_in                                                  "hot-side inlet temperature"
T_C[1]=T_C_out                                                 "cold-side outlet temperature"
i_H[1]=i_H_in                                                  "hot_side inlet enthalpy"
i_C[1]=i_C_out                                                 "cold-side outlet enthalpy"

duplicate i=2,(N+1)
      i_H[i]=i_H[i-1]-q_dot/(N*m_dot_H)                        "energy balance on hot-side of each sub-heat
exchanger"
      T_H[i]=temperature(H$,h=i_H[i],P=p_H[i])                 "temperature leaving hot-side of each sub-heat
exchanger"
end

duplicate i=2,(N+1)
      i_C[i]=i_C[i-1]-q_dot/(N*m_dot_C)                         "energy balance on cold-side of each sub-heat
exchanger"
      T_C[i]=temperature(C$,h=i_C[i],p=p_c[i])                 "temperature leaving cold-side of each sub-heat
exchanger"
end


                                                               "Apply effectiveness-NTU solution"
duplicate i=1,N
      delta_T_H[i]=T_H[i]-T_H[i+1]  "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going
to zero and messing up solving"
      delta_T_C[i]=T_C[i]-T_C[i+1]
      delta_i_H[i]=(i_H[i]-i_H[i+1])
      delta_i_C[i]=(i_C[i]-i_C[i+1])
      C_dot_H[i]=m_dot_H*(delta_i_H[i])/(delta_T_H[i])          "hot-side capacitance rate"
      C_dot_C[i]=m_dot_C*(delta_i_C[i])/(delta_T_C[i])          "cold-side capacitance rate"
end

duplicate i=1,N
```

```
    delta_T_HC[i]=T_H[i]-T_C[i+1] "by breaking the delta T/delta enthalpy values into their own variables I can stop them from going to zero and messing up solving"
    q_dot_node[i]=q_dot/N "this is the heat transfer through the node of the heat exchanger"
    q_dot_max[i]=min(C_dot_H[i],C_dot_C[i])*delta_T_HC[i]  "this is the maximum amount of heat that can be transfered through node of the heat exchanger"
    eff[i]=q_dot/(N*MIN(C_dot_H[i],C_dot_C[i])*(delta_T_HC[i]))  "effectiveness of sub-heat exchanger"
    NTU[i]=HX('counterflow', eff[i], C_dot_H[i], C_dot_C[i], 'NTU')  "NTU required by sub-heat exchanger"
    UA[i]=NTU[i]*MIN(C_dot_H[i],C_dot_C[i])  "conductance in sub-heat exchanger"
end

                                                                    "determine length of each sub-heat exchanger"
x[1]=1e-5 [m]                                                       "starting position of 1st sub-heat exchanger"

fouling_factor = 0.00035 [m^2-K/W]

duplicate i=1,N
    DELTAx[i]=UA[i]*(fouling_factor+1/h_H[i]+th_m/k_m[i]+1/h_C[i])/(D_H*N_CH_H) "length of sub-heat exchanger"
    call channel_flowPC(H$,C$,M$,Deltax[i],W_C,W_H,D_C,D_H,T_H[i],T_C[i],P_H[i],P_C[i],th_m,m_dot_H/N_CH_H,m_dot_C/N_CH_C,rough,rough:H_H[i],H_C[i],DELTA_P_H[i],DELTA_P_C[i],V_h[i],V_c[i],Re_H[i],Re_C[i],f_h[i],f_c[i])
    k_m[i]=k_(M$, (T_H[i]+T_C[i])/2)  "metal conductivity at local average temperature"
    x[i+1]=x[i]+DELTAx[i]


end

duplicate i=1,N-1 "all pressure calculations except the first and last node"

    P_H[i+1] = P_H[i] {-DELTA_P_H[i]}
    P_C[i+1] = P_C[i]{ - DELTA_P_C[i]}

end


P_H[1]=P_H_in {- DELTA_P_H_entrance}
P_C[1]=P_C_in {- DELTA_P_C_entrance}


P_H[N_CHX1#] = P_H[N_CHX#]{ - DELTA_P_H[N_CHX#] -DELTA_P_H_exit}
P_C[N_CHX1#] = P_C[N_CHX#] {- DELTA_P_C[N_CHX#] - DELTA_P_C_exit}
C_entrance = 0.5 "entrance and exit form loss coefficients"
C_exit = 1

DELTA_P_H_entrance = (C_entrance/2)*density(H$,T=T_H_in,P=P_H[1])*V_H[1]^2
DELTA_P_C_entrance = (C_entrance/2)*density(C$,T=T_C_in,P=P_C[1])*V_C[1]^2

DELTA_P_H_exit = (C_exit/2)*density(H$,T=T_H[N_CHX#],P=P_H[N_CHX#])*V_H[N_CHX#]^2
DELTA_P_C_exit = (C_exit/2)*density(C$,T=T_C[N_CHX#],P=P_C[N_CHX#])*V_C[N_CHX#]^2


duplicate i=1,N

    H_H_weighting[i]=H_H[i]*DELTAx[i] "this section calculates weighting factors for whole-HX calculation"
    H_C_weighting[i]=H_C[i]*DELTAx[i]
end

H_bar_H=sum(H_H_weighting[i],i=1,N)/Length_HX
H_bar_C=sum(H_C_weighting[i],i=1,N)/Length_HX


DELTAP_H = P_H[1]-P_H[N+1]
DELTAP_C = P_C[1]-P_C[N+1]

Length_HX = x[N+1]                                                  "total length of HX"


UA_total=sum(UA[i],i=1,N)

Area = N_CH_H*D_H*Length_HX
```

```
N_CH_C = N_CH_H*(W_C/W_H) "number of channels on cold side defined by ratio of channel widths"

duplicate i=1,N+1
      T_C_reverse[i]=T_C[N+2-i]
end

"Effectiveness calculation"
eff_overall= (i_h_in-i_h_out)/(i_h_in - enthalpy(H$,P=P_H_in,T=T_C_in))
Volume = N_CH_C*(2*W_C+2*th_m)*(th_m+D_C)*Length_HX                    "HX volume"
pumping_power_cold= DELTAP_C*m_dot_C/density(C$,T=(T_C[N+1]+T_C[1])/2,P=(P_C[1]+P_c[N+1])/2)

end


$bookmark Main
                                                                       "Overall Program Control"
                                                                       "Constants"
C$='water'                                                             "cooling fluid"
H$='Salt (60% NaNO3, 40% KNO3)'                                        "heating fluid"
WF$='carbondioxide'                                                    "working fluid"

N1 = 1                                                                 "Regenerator Cold Side Inlet"
N2 = 1+N_SHX#                                                          "Regenerator Cold Side outlet"
N3 = 2+N_SHX#                                                          "PHX Inlet"
N4 = 2+2*N_SHX#                                                        "PHX Outlet/Turbine Inlet"
N5 = 3+2*N_SHX#                                                        "Turbine Outlet"
N6 = 4+2*N_SHX#                                                        "Regenerator Hot Side inlet"
N7 = 4+3*N_SHX#                                                        "Regenerator Hot Side Outlet"
N8 = 5+ 3*N_SHX#                                                       "aircooler outlet"
N9 =6+ 3*N_SHX#                                                        "aircooler2 outlet"
N10 = N9+1   "aircooler3 outlet"
N11=N10+1   "compressor outlet"
N_PC_1 = N_CHX#+1

channel_length_ratio_PHX =(N_ch_PHX_C/Length_PHX)*(m_dot_wf_scaled/m_dot_wf)  "ratio of number of channels in hx to
length of hx - fixing this value fixed aspect ratio of hx as hx area changes"
channel_length_ratio_RG =(N_ch_RG_C/Length_RG)*(m_dot_wf_scaled/m_dot_wf)



fraction_PHX=0.3 "opt"


volume_nonaircooler = (volume_rg+volume_Phx)*(m_dot_wf_scaled/m_dot_wf) "total volume of heat exchangers - neglecting
aircooler"

volume_ratio = volume_aircooler/(volume_nonaircooler+volume_aircooler) "ratio of air cooler volume to overall HX volume"

T_WF[N7] = T_AC_in "inlet temp to aircooler"
T_WF[N8] = T_PC_in "inlet temp to precooler"
T_WF[N10]=max(320,T_air_in+approach)

approach = 10                                                          "for large aircooler"
{approach=15                                                           "for medium aircooler"}
{approach=20                                                           "for small aircooler"}

L_aircooler=1.2 "for large (2500m2) aircooler size"
{L_aircooler=1.034 "for medium (16000m2) aircooler size"}
{L_aircooler=0.8208  "for small (8000m2) aircooler size"}
aircooler_AR = L_aircooler/sqrt(A_fr_aircooler)                        "ratio of aircooler length to its height/width"
{aircooler_AR=0.249 }

channel_length_ratio_PHX=4500                                          "opt"
channel_length_ratio_RG=2500                                           "opt"

$ifnot ParametricTable
T_H_in   =970 [k]                                                      "heating fluid inlet temp"
m_dot_cool_scaled=500 [kg/s]
m_dot_h_scaled=150 [kg/s]
```

```
"CONVERGING AT SINGLE RUN SETTINGS"
T_air_in = 270 [K]
{m_dot_air = 200 [kg/s]}

HX_area_aircooler=25000 [m^2]                                   "for larege aircooler"
{HX_area_aircooler=16000 [m^2]                                  "for medium aircooler"}
{HX_area_aircooler=8000 [m^2]                                   "for small aircooler"}

area_total_scaled =2333                                         "for medium aircooler"
P_ratio=2.333                                                   "OPT FOR POWER"

$endif


P_high=25000000 [pa]                                            "high pressure point"
m_dot_normalized = 1
W_channel_PHX_C=0.0021 [m]                                      " OPT"
W_channel_PHX_H=0.0008348 [m]                                   " OPT"
W_channel_RG_C=0.001532[m]                                      " OPT"
W_channel_RG_H=0.002119 [m]                                     " OPT"
N_aircoolers =3
power_aircool2=power_aircool
power_aircool3=power_aircool
q_air_ratio=1 "ratio of air cooling to overall cooling"

P_H=10000000[Pa]                                                "hot fluid pressure"
P_C=10000000[Pa]                                                "cold fluid pressure"
P_ratio=P_high/P_low
                                                                "in this case the inlet temp of hot fluid is
commented out and the temp at the end of the primary heat exchanger is specified"
{T_H_out=700 [k]}                                               "heating fluid outlet temp"
{T_C_out=300[k]}
{m_dot_wf_design=1[kg/s]}
m_dot_wf=m_dot_wf_design                                        "mass flow rate of working fluid"
flowrate_compressor_inlet=0.15 [m^3/s]

T2=T_WF[N11]
T5=T_WF[N5]

turbine_eff=.9                                                  "turbine efficiency"
comp_eff=.89                                                    "compressor efficiency"

fraction_PHX_opt = area_PHX/area_total                          "for optimization - these values can't go out of
physical bounds"

fraction_PHX = area_PHX/area_total
fraction_RG = area_RG/area_total

area_total = area_phx+area_RG
area_total_scaled = area_total*(m_dot_wf_scaled/m_dot_wf)

UA_Total = UA_regenerator+UA_PHX

DELTA_T_salt = T_H_in - T_H_out

Call
PHX(H$,T_H_in,T_H_out,P_H,WF$,m_dot_h,T_WF[N2],P_WF[N2],m_dot_wf,W_channel_PHX_C,W_channel_PHX_H,N_ch_PHX
_C:T_H[N1..N2],T_WF[N3..N4],q_dot,eff_h,UA_PHX,T_PHX_Pinch,Area_PHX,P_WF[N3..N4],Length_PHX,H_bar_H_PHX,H_bar_
C_PHX,DELTAP_H_PHX,DELTAP_C_PHX,V_PHX_H,V_PHX_C,Re_PHX_H,Re_PHX_C,Volume_PHX,pumping_power_salt,N_ch
_PHX_H) "this HX represents heat transfer from heat source to WF after regenerator"

Call
Regenerator(WF$,T5,T6,P_low,WF$,m_dot_wf,T2,p_high,m_dot_wf,W_channel_RG_H,W_channel_RG_C,N_ch_RG_H:T_WF[N6.
.N7],T_WF[1..N2],q_dot_reg,eff_r,UA_regenerator,T_RG_Pinch,Area_RG,P_WF[N6..N7],P_WF[1..N2],Length_RG,H_bar_H_RG,H
_bar_C_RG,DELTAP_H_RG,DELTAP_C_RG,V_RG_H,V_RG_C,Re_RG_H,Re_RG_C,Volume_RG,N_ch_RG_C)"this heat
exchanger represents the regenerator"

Call Turbine (turbine_eff,WF$,T_WF[N4],P_WF[N4],P_low:T_WF[N5],W_turbine)   "this turbine extracts energy from the working
fluid"
```

```
m_dot_air=m_dot_air1+m_dot_air2+m_dot_air3

Call
aircooler(HX_area_aircooler/n_aircoolers,T_air_in,T_WF[N7],m_dot_air1,m_dot_wf_scaled,P_WF[N7],L_aircooler:T_air_out,T_WF[
N8],P_WF[N8],power_aircool,q_dot_aircooler,air_approach,A_fr_aircooler,volume_aircooler)

Call
aircooler2(HX_area_aircooler/n_aircoolers,T_air_in,T_WF[N8],m_dot_air2,m_dot_wf_scaled,P_WF[N8],L_aircooler:T_air_out2,T_W
F[N9],P_WF[N9],power_aircool2,q_dot_aircooler2,air_approach2,A_fr_aircooler2,volume_aircooler2)

Call
aircooler3(HX_area_aircooler/n_aircoolers,T_air_in,T_WF[N9],m_dot_air3,m_dot_wf_scaled,P_WF[N9],L_aircooler:T_air_out3,T_W
F[N10],P_WF[N10],power_aircool3,q_dot_aircooler3,air_approach3,A_fr_aircooler3,volume_aircooler3)


Call Compressor(comp_eff,WF$,T_WF[N10],P_WF[N10],P_high:T_WF[N11],W_comp) "the compressor takes the fluid from the
heat rejection heat exchanger to the regenerator"

work_net=m_dot_wf*(W_turbine+W_comp)  - pumping_power_salt -
(power_aircool+power_aircool2+power_aircool3)*(m_dot_wf/m_dot_wf_scaled)"net work done by the cycle"

efficiency=work_net/q_dot "this section of code records the efficiency of the cycle in terms of turbine/compressor work and heat
transfer to the system in the primary heat exchanger"

comp_inlet_density=density(WF$,T=T_WF[N10],P=P_WF[N10])                    {working fluid density at compressor inlet}
flowrate_compressor_inlet = (m_dot_wf/comp_inlet_density)*(m_dot_wf_scaled/1 [kg/s])  "volumetric flowrate of working fluid at
compressor inlet"

duplicate i=1,N4 {this section sets up the array recording the entropy of the working fluid at each point}
      s[i]=entropy(WF$,T=T_WF[i],P=P_high)
      h[i]=enthalpy(WF$,T=T_WF[i],P=P_high)
end

duplicate i=N5,N10
      s[i]=entropy(WF$,T=T_WF[i],P=P_low)
      h[i]=enthalpy(WF$,T=T_WF[i],P=P_low)
end

s[N11]=entropy(WF$,T=T_WF[N11],P=P_high)
h[N11]=enthalpy(WF$,T=T_WF[N11],P=P_high)


"PLANT SIZE SCALING"

Power_scaled=m_dot_wf_scaled*work_net/m_dot_wf                    "comment out to determine m_dot_wf_scaled"
turbine_power_scaled = w_turbine*m_dot_wf_scaled
comp_power_scaled=w_comp*m_dot_wf_scaled
m_dot_wf_scaled=100*m_dot_wf
area_phx_scaled=m_dot_wf_scaled*area_phx/m_dot_wf
area_rg_scaled=m_dot_wf_scaled*area_rg/m_dot_wf

pumping_power_salt_scaled = pumping_power_salt*(m_dot_wf_scaled/m_dot_wf)

m_dot_h_scaled = m_dot_h*(m_dot_wf_scaled/m_dot_wf)
```

Appendix H: Design Point Model Parameters and Results

Water-Cooled Results

Water-Cooled: Designed (Small HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*612.5*	
Regenerator Area [m^2]	*1102*	
Precooler Area [m^2]	*735*	
PHX Salt Side Channel Width/Depth [mm]	*0.71*	
PHX CO2 Side Channel Width/Depth [mm]	*2.3*	
Regenerator Hot Side Channel Width/Depth [mm]	*1.8*	
Regenerator Cold Side Channel Width/Depth [mm]	*2.2*	
Precooler CO2 Side Channel Width/Depth [mm]	*1.4*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*10000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*4000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*10000*	
Cooling Water Design Demand Temperature [K]	**295.8**	Performance
Cooling Water Outlet Temperature [K]	**303.9**	
Power [MW]	**11.4**	
Efficiency	**40.32%**	

Water-Cooled: Designed (Design HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*875*	
Regenerator Area [m^2]	*1575*	
Precooler Area [m^2]	*1050*	
PHX Salt Side Channel Width/Depth [mm]	*0.71*	
PHX CO2 Side Channel Width/Depth [mm]	*2.3*	
Regenerator Hot Side Channel Width/Depth [mm]	*1.8*	
Regenerator Cold Side Channel Width/Depth [mm]	*2.2*	
Precooler CO2 Side Channel Width/Depth [mm]	*1.4*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*10000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*4000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*10000*	
Cooling Water Design Demand Temperature [K]	**299.7**	Performance
Cooling Water Outlet Temperature [K]	**307.5**	
Power [MW]	**11.7**	
Efficiency	**41.75%**	

Water-Cooled: Designed (Large HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*1313*	
Regenerator Area [m^2]	*2363*	
Precooler Area [m^2]	*1575*	
PHX Salt Side Channel Width/Depth [mm]	*0.71*	
PHX CO2 Side Channel Width/Depth [mm]	*2.3*	
Regenerator Hot Side Channel Width/Depth [mm]	*1.8*	
Regenerator Cold Side Channel Width/Depth [mm]	*2.2*	
Precooler CO2 Side Channel Width/Depth [mm]	*1.4*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*10000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*4000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*10000*	
Cooling Water Design Demand Temperature [K]	**302.5**	Performance
Cooling Water Outlet Temperature [K]	**310.2**	
Power [MW]	**11.9**	
Efficiency	**42.83%**	

Hybrid-Cooled Results

Hybrid-Cooled: Designed (Small HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Design Air Temperature [K]	310	
Aircooler Fan Power [kW]	50	
Aircooler Area [m^2]	16000	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*612.5*	
Regenerator Area [m^2]	*1102*	
Precooler Area [m^2]	*735*	
PHX Salt Side Channel Width/Depth [mm]	*0.71*	
PHX CO2 Side Channel Width/Depth [mm]	*2.3*	
Regenerator Hot Side Channel Width/Depth [mm]	*1.8*	
Regenerator Cold Side Channel Width/Depth [mm]	*2.2*	
Precooler CO2 Side Channel Width/Depth [mm]	*1.4*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*10000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*4000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*10000*	
Cooling Water Design Demand Temperature [K]	**297.8**	Performance
Cooling Water Outlet Temperature [K]	**303.9**	
Power [MW]	**11.3**	
Efficiency	**40.14%**	

Hybrid-Cooled: Designed (Design HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Design Air Temperature [K]	310	
Aircooler Fan Power [kW]	50	
Aircooler Area [m^2]	16000	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*875*	
Regenerator Area [m^2]	*1575*	
Precooler Area [m^2]	*1050*	
PHX Salt Side Channel Width/Depth [mm]	*0.71*	
PHX CO2 Side Channel Width/Depth [mm]	*2.3*	
Regenerator Hot Side Channel Width/Depth [mm]	*1.8*	
Regenerator Cold Side Channel Width/Depth [mm]	*2.2*	
Precooler CO2 Side Channel Width/Depth [mm]	*1.4*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*10000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*4000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*10000*	
Cooling Water Design Demand Temperature [K]	**300.7**	Performance
Cooling Water Outlet Temperature [K]	**306.8**	
Power [MW]	**11.8**	
Efficiency	**41.60%**	

Hybrid-Cooled: Designed (Large HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Cooling Water Mass Flow Rate [kg/s]	500	
Design Salt Inlet Temperature [K]	900	
Design Air Temperature [K]	310	
Aircooler Fan Power [kW]	50	
Aircooler Area [m^2]	16000	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	307.5	
Design Pressure Ratio	*2.9*	Designed Parameters
Primary Heat Exchanger Area [m^2]	*1313*	
Regenerator Area [m^2]	*2363*	
Precooler Area [m^2]	*1575*	
PHX Salt Side Channel Width/Depth [mm]	*0.71*	
PHX CO2 Side Channel Width/Depth [mm]	*2.3*	
Regenerator Hot Side Channel Width/Depth [mm]	*1.8*	
Regenerator Cold Side Channel Width/Depth [mm]	*2.2*	
Precooler CO2 Side Channel Width/Depth [mm]	*1.4*	
Precooler Water Side Channel Width/Depth [mm]	*2*	
PHX Aspect Ratio [# Channels/Length [m]]	*10000*	
Regenerator Aspect Ratio [# Channels/Length [m]]	*4000*	
Precooler Aspect Ratio [# Channels/Length [m]]	*10000*	
Cooling Water Design Demand Temperature [K]	**303.2**	Performance
Cooling Water Outlet Temperature [K]	**309.1**	
Power [MW]	**11.9**	
Efficiency	**42.65%**	

Air-Cooled Results

Air-Cooled: Designed (Small HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Design Salt Inlet Temperature [K]	900	
Design Air Temperature [K]	310	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	325	
Aircooler Area [m^2]	8000	
Minimum Air-CO2 Approach Temperature [K]	20	
Design Pressure Ratio	2.33	Designed Parameters
Primary Heat Exchanger Area [m^2]	700	
Regenerator Area [m^2]	1633	
Fan Power @Air = 310 K [kW]	386	
PHX Salt Side Channel Width/Depth [mm]	1.2	
PHX CO2 Side Channel Width/Depth [mm]	2.4	
Regenerator Hot Side Channel Width/Depth [mm]	1.95	
Regenerator Cold Side Channel Width/Depth [mm]	1.5	
PHX Aspect Ratio [# Channels/Length [m]]	4500	
Regenerator Aspect Ratio [# Channels/Length [m]]	2500	
Power [MW]	**5.81**	Performance
Efficiency	**38.4%**	

Air-Cooled: Designed (Design HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Design Salt Inlet Temperature [K]	900	
Design Air Temperature [K]	310	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	325	
Aircooler Area [m^2]	16000	
Minimum Air-CO2 Approach Temperature [K]	15	
Design Pressure Ratio	2.33	Designed Parameters
Primary Heat Exchanger Area [m^2]	700	
Regenerator Area [m^2]	1633	
Fan Power @Air = 310 K [kW]	386	
PHX Salt Side Channel Width/Depth [mm]	1.2	
PHX CO2 Side Channel Width/Depth [mm]	2.4	
Regenerator Hot Side Channel Width/Depth [mm]	1.95	
Regenerator Cold Side Channel Width/Depth [mm]	1.5	
PHX Aspect Ratio [# Channels/Length [m]]	4500	
Regenerator Aspect Ratio [# Channels/Length [m]]	2500	
Power [MW]	**6.52**	Performance
Efficiency	**38.3%**	

Air-Cooled: Designed (Large HX)		
Compressor Outlet Pressure [MPa]	25	Fixed Parameters
CO2 Compressor Inlet Flowrate [m3/s]	0.15	
60% NaNO3 40% KNO3 Mass Flow Rate [kg/s]	150	
Design Salt Inlet Temperature [K]	900	
Design Air Temperature [K]	310	
Turbine Efficiency [%]	90.00%	
Compressor Efficiency [%]	89.00%	
Compressor Inlet Design CO2 Temperature [K]	325	
Aircooler Area [m^2]	25000	
Minimum Air-CO2 Approach Temperature [K]	10	
Design Pressure Ratio	2.33	Designed Parameters
Primary Heat Exchanger Area [m^2]	700	
Regenerator Area [m^2]	1633	
Fan Power @Air = 310 K [kW]	386	
PHX Salt Side Channel Width/Depth [mm]	1.2	
PHX CO2 Side Channel Width/Depth [mm]	2.4	
Regenerator Hot Side Channel Width/Depth [mm]	1.95	
Regenerator Cold Side Channel Width/Depth [mm]	1.5	
PHX Aspect Ratio [# Channels/Length [m]]	4500	
Regenerator Aspect Ratio [# Channels/Length [m]]	2500	
Power [MW]	**7.1**	Performance
Efficiency	**35.9%**	

Appendix I: FORTRAN Code

```
SUBROUTINE TYPE241 (TIME,XIN,OUT,T,DTDT,PAR,INFO,ICNTRL,*)


C    TRNSYS acess functions (allow to acess TIME etc.)
      USE TrnsysConstants
      USE TrnsysFunctions

C-------------------------------------------------------------------------------------------------------
------------------------
C    REQUIRED BY THE MULTI-DLL VERSION OF TRNSYS
      !DEC$ATTRIBUTES DLLEXPORT :: TYPE241                              !SET THE CORRECT TYPE NUMBER
HERE
C-------------------------------------------------------------------------------------------------------
------------------------
C-------------------------------------------------------------------------------------------------------
------------------------
C    TRNSYS DECLARATIONS
      IMPLICIT NONE!REQUIRES THE USER TO DEFINE ALL VARIABLES BEFORE USING THEM

        DOUBLE PRECISION XIN   !THE ARRAY FROM WHICH THE INPUTS TO THIS TYPE WILL BE RETRIEVED
        DOUBLE PRECISION OUT   !THE ARRAY WHICH WILL BE USED TO STORE THE OUTPUTS FROM THIS TYPE
        DOUBLE PRECISION TIME  !THE CURRENT SIMULATION TIME - YOU MAY USE THIS VARIABLE BUT DO NOT
SET IT!
        DOUBLE PRECISION PAR   !THE ARRAY FROM WHICH THE PARAMETERS FOR THIS TYPE WILL BE
RETRIEVED
        DOUBLE PRECISION STORED !THE STORAGE ARRAY FOR HOLDING VARIABLES FROM TIMESTEP TO
TIMESTEP
        DOUBLE PRECISION T             !AN ARRAY CONTAINING THE RESULTS FROM THE DIFFERENTIAL
EQUATION SOLVER
        DOUBLE PRECISION DTDT  !AN ARRAY CONTAINING THE DERIVATIVES TO BE PASSED TO THE DIFF.EQ.
SOLVER
        INTEGER*4 INFO(15)             !THE INFO ARRAY STORES AND PASSES VALUABLE INFORMATION TO
AND FROM THIS TYPE
        INTEGER*4 NP,NI,NOUT,ND        !VARIABLES FOR THE MAXIMUM NUMBER OF
PARAMETERS,INPUTS,OUTPUTS AND DERIVATIVES
        INTEGER*4 NPAR,NIN,NDER        !VARIABLES FOR THE CORRECT NUMBER OF
PARAMETERS,INPUTS,OUTPUTS AND DERIVATIVES
        INTEGER*4 IUNIT,ITYPE  !THE UNIT NUMBER AND TYPE NUMBER FOR THIS COMPONENT
        INTEGER*4 ICNTRL               !AN ARRAY FOR HOLDING VALUES OF CONTROL FUNCTIONS WITH THE
NEW SOLVER
        INTEGER*4 NSTORED              !THE NUMBER OF VARIABLES THAT WILL BE PASSED INTO AND OUT
OF STORAGE
        CHARACTER*3 OCHECK             !AN ARRAY TO BE FILLED WITH THE CORRECT VARIABLE TYPES FOR
THE OUTPUTS
        CHARACTER*3 YCHECK             !AN ARRAY TO BE FILLED WITH THE CORRECT VARIABLE TYPES FOR
THE INPUTS
C-------------------------------------------------------------------------------------------------------
------------------------

C-------------------------------------------------------------------------------------------------------
------------------------
C    USER DECLARATIONS - SET THE MAXIMUM NUMBER OF PARAMETERS (NP), INPUTS (NI),
C    OUTPUTS (NOUT), AND DERIVATIVES (ND) THAT MAY BE SUPPLIED FOR THIS TYPE
      PARAMETER (NP=6,NI=4,NOUT=9,ND=0,NSTORED=0)
C-------------------------------------------------------------------------------------------------------
------------------------

C-------------------------------------------------------------------------------------------------------
------------------------
C    REQUIRED TRNSYS DIMENSIONS
      DIMENSION XIN(NI),OUT(NOUT),PAR(NP),YCHECK(NI),OCHECK(NOUT),
     1   STORED(NSTORED),T(ND),DTDT(ND)
      INTEGER NITEMS
C-------------------------------------------------------------------------------------------------------
------------------------
```

```
C-----------------------------------------------------------------------------------------------------
-----------------------
C    ADD DECLARATIONS AND DEFINITIONS FOR THE USER-VARIABLES HERE

C     Parameters
      DOUBLE PRECISION  cap_ratio,time_start,time_stop,hx_area
      DOUBLE PRECISION plant_type,ua_aircooler,salt_temp_min
      DOUBLE PRECISION salt_temp_off


C    Inputs
      DOUBLE PRECISION T_salt_in,hour,T_water_in
      DOUBLE PRECISION T_air_in

C    Outputs / Local variables
      DOUBLE PRECISION power_cycle,T_salt_out,T_water_out,m_dot_salt_ref
      DOUBLE PRECISION m_dot_water_ref,power_ref,m_dot_salt_norm
      DOUBLE PRECISION m_dot_water_norm,power_rated,eta,T_water_in_low
      DOUBLE PRECISION m_dot_water,m_dot_salt,T_water_in_high
      DOUBLE PRECISION T_salt_demand
C-----------------------------------------------------------------------------------------------------
-----------------------
C       READ IN THE VALUES OF THE PARAMETERS IN SEQUENTIAL ORDER
      cap_ratio = PAR(1)
      time_start = PAR(2)
      time_stop = PAR(3)
      plant_type = PAR(4)
      salt_temp_min = PAR(5)
      salt_temp_off = PAR(6)

C-----------------------------------------------------------------------------------------------------
-----------------------
C    RETRIEVE THE CURRENT VALUES OF THE INPUTS TO THIS MODEL FROM THE XIN ARRAY IN SEQUENTIAL
ORDER

      T_salt_in = XIN(1) + 273.15 !convert to K
      hour = XIN(2)
      T_air_in = XIN(3) + 273.15
      T_water_in = XIN(4) + 273.15

C-----------------------------------------------------------------------------------------------------
-----------------------
C    SET THE VERSION INFORMATION FOR TRNSYS
      IF(INFO(7).EQ.-2) THEN
          INFO(12)=16
          RETURN 1
        ENDIF
C-----------------------------------------------------------------------------------------------------
-----------------------

C-----------------------------------------------------------------------------------------------------
-----------------------
C    DO ALL THE VERY LAST CALL OF THE SIMULATION MANIPULATIONS HERE
      IF (INFO(8).EQ.-1) THEN
          RETURN 1
        ENDIF
C-----------------------------------------------------------------------------------------------------
-----------------------

C-----------------------------------------------------------------------------------------------------
-----------------------
C    PERFORM ANY 'AFTER-ITERATION' MANIPULATIONS THAT ARE REQUIRED HERE
C    e.g. save variables to storage array for the next timestep
      IF (INFO(13).GT.0) THEN
          NITEMS=0
C         STORED(1)=... (if NITEMS > 0)
C        CALL setStorageVars(STORED,NITEMS,INFO)
          RETURN 1
        ENDIF
C
```

```
C-------------------------------------------------------------------------------------------------
----------------------

C-------------------------------------------------------------------------------------------------
----------------------
C    DO ALL THE VERY FIRST CALL OF THE SIMULATION MANIPULATIONS HERE
      IF (INFO(7).EQ.-1) THEN

C       SET SOME INFO ARRAY VARIABLES TO TELL THE TRNSYS ENGINE HOW THIS TYPE IS TO WORK
         INFO(6)=NOUT
         INFO(9)=1
          INFO(10)=0  !STORAGE FOR VERSION 16 HAS BEEN CHANGED

C       SET THE REQUIRED NUMBER OF INPUTS, PARAMETERS AND DERIVATIVES THAT THE USER SHOULD SUPPLY
IN THE INPUT FILE
C       IN SOME CASES, THE NUMBER OF VARIABLES MAY DEPEND ON THE VALUE OF PARAMETERS TO THIS
MODEL....
         NIN=NI
          NPAR=NP
          NDER=ND

C       CALL THE TYPE CHECK SUBROUTINE TO COMPARE WHAT THIS COMPONENT REQUIRES TO WHAT IS
SUPPLIED IN
C       THE TRNSYS INPUT FILE
          CALL TYPECK(1,INFO,NIN,NPAR,NDER)

C       SET THE NUMBER OF STORAGE SPOTS NEEDED FOR THIS COMPONENT
         NITEMS=0
C          CALL setStorageSize(NITEMS,INFO)

C       RETURN TO THE CALLING PROGRAM
         RETURN 1

      ENDIF
C-------------------------------------------------------------------------------------------------
----------------------

C-------------------------------------------------------------------------------------------------
----------------------
C    DO ALL OF THE INITIAL TIMESTEP MANIPULATIONS HERE - THERE ARE NO ITERATIONS AT THE INTIAL
TIME
      IF (TIME .LT. (getSimulationStartTime() +
     . getSimulationTimeStep()/2.D0)) THEN

C       SET THE UNIT NUMBER FOR FUTURE CALLS
         IUNIT=INFO(1)
         ITYPE=INFO(2)

C       CHECK THE PARAMETERS FOR PROBLEMS AND RETURN FROM THE SUBROUTINE IF AN ERROR IS FOUND
C         IF(...) CALL TYPECK(-4,INFO,0,"BAD PARAMETER #",0)

C       PERFORM ANY REQUIRED CALCULATIONS TO SET THE INITIAL VALUES OF THE OUTPUTS HERE
C               Cycle power output
      out(5) =23

C       PERFORM ANY REQUIRED CALCULATIONS TO SET THE INITIAL STORAGE VARIABLES HERE
         NITEMS=0
C          STORED(1)=...

C       PUT THE STORED ARRAY IN THE GLOBAL STORED ARRAY
C         CALL setStorageVars(STORED,NITEMS,INFO)

C       RETURN TO THE CALLING PROGRAM
         RETURN 1

      ENDIF
C-------------------------------------------------------------------------------------------------
----------------------

C-------------------------------------------------------------------------------------------------
----------------------
```

```
C    *** ITS AN ITERATIVE CALL TO THIS COMPONENT ***
C-----------------------------------------------------------------------------------------------------
-----------------------


C-----------------------------------------------------------------------------------------------------
-----------------------
C    RETRIEVE THE VALUES IN THE STORAGE ARRAY FOR THIS ITERATION
C      NITEMS=
C       CALL getStorageVars(STORED,NITEMS,INFO)
C      STORED(1)=
C-----------------------------------------------------------------------------------------------------
-----------------------
C-----------------------------------------------------------------------------------------------------
-----------------------
C    CHECK THE INPUTS FOR PROBLEMS
C      IF(...) CALL TYPECK(-3,INFO,'BAD INPUT #',0,0)
C       IF(IERROR.GT.0) RETURN 1
C-----------------------------------------------------------------------------------------------------
-----------------------
C-----------------------------------------------------------------------------------------------------
-----------------------
C    *** PERFORM ALL THE CALCULATION HERE FOR THIS MODEL. ***
C-----------------------------------------------------------------------------------------------------
-----------------------
      ! under-sized HX water
      IF (plant_type == 0) THEN
      power_ref = 1.13721E7            !reference power of EES-designed plant
      m_dot_salt_ref = 360000*1.5      !reference salt flow of EES-designed plant
      m_dot_water_ref = 500*3600       !reference waterflow of EES-designed plant
      power_rated= cap_ratio*power_ref !ratio of plant cap to ref plant

      m_dot_salt=m_dot_salt_ref*cap_ratio

      m_dot_salt_norm = m_dot_salt/(m_dot_salt_ref*cap_ratio)

      power_cycle=(-3.94842841E-01+1.54738067E-03*T_salt_in)*power_rated

      eta=6.86950698E-02+3.70498836E-04*T_salt_in

      T_water_in_high =2.97118219E+02-1.47031431E-03*T_salt_in -273.15 !water temp demand from
tower

      T_water_in_low = T_water_in_high - 1      !lower range demand

      T_water_out=3.04961084E+02-2.71305267E-03*T_salt_in+
     1 1.68520524E-06*T_salt_in**2- 273.15

      T_salt_out=-5.17505483E+01+9.22269139E-01*T_salt_in- 273.15

      m_dot_water_norm=1

      m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio

      T_salt_demand = salt_temp_min


      ENDIF
   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!   ! design HX water

      IF (plant_type ==1) THEN
      power_ref = 11697400              !reference power of EES-designed plant
      m_dot_salt_ref = 360000*1.5       !reference salt flow of EES-designed plant
      m_dot_water_ref = 500*3600        !reference waterflow of EES-designed plant
      power_rated= cap_ratio*power_ref  !ratio of plant cap to ref plant

      m_dot_salt=m_dot_salt_ref*cap_ratio

      m_dot_salt_norm = m_dot_salt/(m_dot_salt_ref*cap_ratio)

      power_cycle=(-3.76870780E-01+1.52784633E-03*T_salt_in)*power_rated
```

```
      eta=6.99647497E-02+3.85120838E-04*T_salt_in     !placeholder value

      T_water_in_high =3.00442317E+02-8.71944121E-04*T_salt_in -273.15 !water temp demand from
tower

      T_water_in_low = T_water_in_high - 1            !lower range demand

      T_water_out=3.08294756E+02-1.97969880E-03*T_salt_in+
     1 1.20726815E-06*T_salt_in**2 - 273.15

      T_salt_out=-5.47800077E+01+9.26564128E-01*T_salt_in - 273.15

      m_dot_water_norm=1

      m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio

      T_salt_demand = salt_temp_min
      ENDIF

      !!!!!!!!!!!!!!!!!!!!!! ! large HX water

      IF (plant_type == 2) THEN
      power_ref = 1.19393E7             !reference power of EES-designed plant
      m_dot_salt_ref = 360000*1.5       !reference salt flow of EES-designed plant
      m_dot_water_ref = 500*3600        !reference waterflow of EES-designed plant
      power_rated= cap_ratio*power_ref  !ratio of plant cap to ref plant

      m_dot_salt=m_dot_salt_ref*cap_ratio

      m_dot_salt_norm = m_dot_salt/(m_dot_salt_ref*cap_ratio)

      power_cycle=(-3.63702561E-01+1.51337601E-03*T_salt_in)*power_rated

      eta=6.83884692E-02+3.98908847E-04*T_salt_in !placeholder value

      T_water_in_high= 302.980384 -5.06402794E-04*T_salt_in -273.15 !water temp demand from tower

      T_water_in_low = T_water_in_high - 1      !lower range demand

      T_water_out=3.10892541E+02-1.49657841E-03*T_salt_in
     1 +7.74568649E-07*T_salt_in**2- 273.15

      T_salt_out=-5.74054917E+01+9.30160196E-01*T_salt_in - 273.15

      m_dot_water_norm=1

      m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio

      T_salt_demand = salt_temp_min
      ENDIF
      !!!!!!!!!!!!!!!!!!!!!!!!!!!!   ! small HX hybrid

      IF (plant_type ==3) THEN
      power_ref = 11345300              !reference power of EES-designed plant
      m_dot_salt_ref = 360000*1.5       !reference salt flow of EES-designed plant
      m_dot_water_ref = 500*3600        !reference waterflow of EES-designed plant
      power_rated= cap_ratio*power_ref  !ratio of plant cap to ref plant

      m_dot_salt=m_dot_salt_ref*cap_ratio

      m_dot_salt_norm = m_dot_salt/(m_dot_salt_ref*cap_ratio)

      power_cycle=(-3.87190942E-01+1.56648804E-03*T_salt_in
     1 -7.67346939E-05*T_air_in)*power_rated

      eta=6.71209527E-02+3.70995336E-04*T_salt_in-
     1 1.69387755E-06*T_air_in     !placeholder value


      T_water_in_high = 6.75563195E+02+6.76996311E-04*T_salt_in-
```

```
1 7.72761988E-07*T_salt_in**2-2.40376786E+00*T_air_in
1 +3.82015306E-03*T_air_in**2-273.15  !water temp demand from tower

 T_water_in_low = T_water_in_high - 1 !lower range demand

 T_water_out= 4.88496359E+02-1.71520894E-03*T_salt_in+
1 7.29018962E-07*T_salt_in**2-1.19049235E+00*T_air_in
1 +1.92729592E-03*T_air_in**2- 273.15

 T_salt_out= -5.39088020E+01+9.21584797E-01*T_salt_in
1 +8.06122449E-03*T_air_in- 273.15

 m_dot_water_norm=1

 m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio

 T_salt_demand = salt_temp_min
 ENDIF

 !!!!!!!!!!!!!!!!!!!!!!!!!!!!   ! design HX hybrid

 IF (plant_type ==4) THEN
 power_ref = 11791300             !reference power of EES-designed plant
 m_dot_salt_ref = 360000*1.5      !reference salt flow of EES-designed plant
 m_dot_water_ref = 500*3600       !reference waterflow of EES-designed plant
 power_rated= cap_ratio*power_ref !ratio of plant cap to ref plant

 m_dot_salt=m_dot_salt_ref*cap_ratio

 m_dot_salt_norm = m_dot_salt/(m_dot_salt_ref*cap_ratio)

 power_cycle=(-3.81176549E-01+1.53306539E-03*T_salt_in-
1 1.63265306E-06*T_air_in)*power_rated

 eta=6.93145281E-02+3.84278059E-04*T_salt_in-
1 2.10037701E-20*T_air_in    !placeholder value


 T_water_in_high = 5.57934273E+02-8.68579148E-04*T_salt_in+
1 4.92294484E-08*T_salt_in**2-1.63349745E+00*T_air_in+
1 2.59821429E-03*T_air_in**2-273.15 !water temp demand from tower

 T_water_in_low = T_water_in_high - 1      !lower range demand

 T_water_out= 3.81585129E+02-1.08007850E-03*T_salt_in+
1 2.02613711E-07*T_salt_in**2-4.96581633E-01*T_air_in
1 +8.31632653E-04*T_air_in**2- 273.15

 T_salt_out= -5.46265456E+01+9.24785650E-01*T_salt_in
1 +3.00866103E-17*T_air_in- 273.15

 m_dot_water_norm=1

 m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio

 T_salt_demand = salt_temp_min
 ENDIF
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!   ! large HX hybrid

 IF (plant_type ==5) THEN
 power_ref = 11899200              !reference power of EES-designed plant
 m_dot_salt_ref = 360000*1.5       !reference salt flow of EES-designed plant
 m_dot_water_ref = 500*3600        !reference waterflow of EES-designed plant
 power_rated= cap_ratio*power_ref  !ratio of plant cap to ref plant

 m_dot_salt=m_dot_salt_ref*cap_ratio

 m_dot_salt_norm = m_dot_salt/(m_dot_salt_ref*cap_ratio)

 power_cycle=(-3.57694544E-01+1.51867123E-03*T_salt_in-
1 3.44897959E-05*T_air_in)*power_rated
```

```
 eta=6.67650738E-02+3.99632827E-04*T_salt_in
1 -2.22448980E-06*T_air_in    !placeholder value


 T_water_in_high = 4.78302798E+02+1.54773473E-04*T_salt_in-
1 3.99978366E-07*T_salt_in**2-1.11684694E+00*T_air_in+
1 1.78061224E-03*T_air_in**2-273.15   !water temp demand from tower

 T_water_in_low = T_water_in_high - 1 !lower range demand

 T_water_out= 3.01074667E+02-5.52495782E-04*T_salt_in
1 +2.75510204E-02*T_air_in- 273.15

 T_salt_out=-5.88321305E+01+9.29933016E-01*T_salt_in
1 +4.91836735E-03*T_air_in - 273.15

 m_dot_water_norm=1

 m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio

 T_salt_demand = salt_temp_min
 ENDIF
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ! air cooled small
 IF (plant_type == 6) THEN
 power_ref = 5.81241E6            !reference power of EES-designed plant
 m_dot_salt_ref = 360000*1.5      !reference salt flow of EES-designed plant
 m_dot_water_ref = 0*3600         !reference waterflow of EES-designed plant
 power_rated= cap_ratio*power_ref !ratio of plant cap to ref plant

 m_dot_salt=m_dot_salt_ref*cap_ratio

 power_cycle=(-1.63796751E+01+4.97489628E-03*T_salt_in-
1 1.81632371E-06*T_salt_in**2+1.08480637E-01*T_air_in-
1 1.99987245E-04*T_air_in**2)*power_rated

 eta=2.65277971E+00+1.05635011E-03*T_salt_in
1 -3.61599356E-07*T_salt_in**2-1.93877406E-02*T_air_in
1 +3.19756378E-05*T_air_in**2

 T_water_out=20

 T_salt_out=1.69589591E+03+9.98064519E-01*T_salt_in-
1 2.85010351E-05*T_salt_in**2-1.24683265E+01*T_air_in
1 +2.21020408E-02*T_air_in**2 -273.15 !converting to C


 T_water_in_high = 40
 T_water_in_low = T_water_in_high - 1    !lower range demand

 m_dot_water_norm=1
 m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio
 T_salt_demand = salt_temp_min
 ENDIF
 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ! air cooled medium
 IF (plant_type == 7) THEN
 power_ref = 6.52E6               !reference power of EES-designed plant
 m_dot_salt_ref = 360000*1.5      !reference salt flow of EES-designed plant
 m_dot_water_ref = 0*3600         !reference waterflow of EES-designed plant
 power_rated= cap_ratio*power_ref !ratio of plant cap to ref plant

 m_dot_salt=m_dot_salt_ref*cap_ratio

 power_cycle=(-1.80308062E+01+2.51700885E-03*T_salt_in-
1 3.98254247E-07*T_salt_in**2+1.24133163E-01*T_air_in
1 -2.22500000E-04*T_air_in**2)*power_rated

 eta=7.82154583E-01+1.23444289E-03*T_salt_in-
1 4.82803956E-07*T_salt_in**2-7.05464056E-03*T_air_in+
1 1.11195153E-05*T_air_in**2
```

```
T_water_out=20

T_salt_out= 1.58512827E+03+9.45716996E-01*T_salt_in+
1 4.09714035E-06*T_salt_in**2-1.14710739E+01*T_air_in
1 +2.02181122E-02*T_air_in**2- 273.15  !converting to C


T_water_in_high = 40
T_water_in_low = T_water_in_high - 1    !lower range demand

m_dot_water_norm=1
m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio

T_salt_demand = salt_temp_min
ENDIF

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! ! air cooled large
IF (plant_type == 8) THEN
power_ref = 7.10E+06 !reference power of EES-designed plant
m_dot_salt_ref = 360000*1.5  !reference salt flow of EES-designed plant
m_dot_water_ref = 0*3600 !reference waterflow of EES-designed plant
power_rated= cap_ratio*power_ref !ratio of plant cap to ref plant

m_dot_salt=m_dot_salt_ref*cap_ratio

power_cycle=(-1.50819253E+01+8.35592549E-04*T_salt_in+
1 5.92606649E-07*T_salt_in**2+1.06770510E-01*T_air_in
1 -1.90051020E-04*T_air_in**2)*power_rated

eta=4.84823004E-01+1.17148629E-03*T_salt_in-
1 4.46766731E-07*T_salt_in**2-4.63197143E-03*T_air_in
1 +6.64183673E-06*T_air_in**2

T_water_out=20

T_salt_out=1.44892618E+03+8.68428469E-01*T_salt_in
1 +5.19623130E-05*T_salt_in**2-1.01885459E+01*T_air_in
1 +1.77678571E-02*T_air_in**2- 273.15
!converting to C

T_water_in_high =40
T_water_in_low = T_water_in_high - 1    !lower range demand

m_dot_water_norm=1
m_dot_water = m_dot_water_norm*m_dot_water_ref*cap_ratio
!convert kg/s to kg/hr
T_salt_demand = salt_temp_min
ENDIF



IF (T_salt_in < 700) THEN
  power_cycle = 0
  eta = 0
  T_salt_out = T_salt_in - 273.15
  T_water_out = T_water_in - 273.15
  T_salt_demand = salt_temp_min
ENDIF

IF (T_water_in > T_water_in_high + 273.15) THEN
  power_cycle = 0
  eta = 0
  T_salt_out = T_salt_in - 273.15
  T_water_out = T_water_in - 273.15
  T_salt_demand = salt_temp_off
ENDIF
IF (hour < time_start) THEN

  IF(T_salt_in < 900) THEN
```

```
            T_water_in_high =30
            T_water_in_low = 28    !lower range demand
            power_cycle = 0
            eta = 0
            T_salt_out = T_salt_in - 273.15
            T_water_out = T_water_in - 273.15
            T_salt_demand = salt_temp_off
          ENDIF

        ENDIF
        IF (hour > time_stop) THEN

          IF(T_salt_in < 900) THEN

            T_water_in_high =30
            T_water_in_low = 28    !lower range demand
            power_cycle = 0
            eta = 0
            T_salt_out = T_salt_in - 273.15
            T_water_out = T_water_in - 273.15
            T_salt_demand = salt_temp_off
          ENDIF

        ENDIF


C-----------------------------------------------------------------------------------------------------
-----------------------

C-----------------------------------------------------------------------------------------------------
-----------------------
C-----------------------------------------------------------------------------------------------------
-----------------------
C     SET THE STORAGE ARRAY AT THE END OF THIS ITERATION IF NECESSARY
C       NITEMS=
C       STORED(1)=
C        CALL setStorageVars(STORED,NITEMS,INFO)
C-----------------------------------------------------------------------------------------------------
-----------------------
C-----------------------------------------------------------------------------------------------------
-----------------------
C     REPORT ANY PROBLEMS THAT HAVE BEEN FOUND USING CALLS LIKE THIS:
C       CALL MESSAGES(-1,'put your message here','MESSAGE',IUNIT,ITYPE)
C       CALL MESSAGES(-1,'put your message here','WARNING',IUNIT,ITYPE)
C       CALL MESSAGES(-1,'put your message here','SEVERE',IUNIT,ITYPE)
C       CALL MESSAGES(-1,'put your message here','FATAL',IUNIT,ITYPE)
C-----------------------------------------------------------------------------------------------------
-----------------------
C-----------------------------------------------------------------------------------------------------
-----------------------
C     SET THE OUTPUTS FROM THIS MODEL IN SEQUENTIAL ORDER AND GET OUT

C                Cycle power output
                OUT(1)=T_salt_out
                OUT(2)=T_water_out
                OUT(3)=power_cycle
                OUT(4)=eta
                OUT(5)=T_water_in_high
          OUT(6)=T_water_in_low
          OUT(7)=m_dot_salt
          OUT(8)=m_dot_water
          OUT(9)=T_salt_demand
C-----------------------------------------------------------------------------------------------------
-----------------------
C     EVERYTHING IS DONE - RETURN FROM THIS SUBROUTINE AND MOVE ON
       RETURN 1
       END
C-----------------------------------------------------------------------------------------------------
-----------------------
```