Modeling and Analysis of Hybrid Geothermal-Solar Thermal Energy Conversion Systems

by

Andrew David Greenhut

B.S., Mechanical Engineering (2006)
Massachusetts Institute of Technology

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY 2010

Signature of Author.....
Department of Mechanical Engineering
December 15, 2009

Certified by.....
Jefferson W. Tester
H.P. Meissner Professor of Chemical Engineering
Thesis Supervisor

Certified by.....
Ahmed F. Ghoniem
Ronald C. Crane (1972) Professor of Mechanical Engineering
ME Faculty Reader

Accepted by.....
David E. Hardt
Chairman, Committee for Graduate Students
Ralph E. and Eloise F Cross Professor of Mechanical Engineering

Modeling and Analysis of Hybrid Geothermal-Solar Thermal Energy Conversion Systems

by

Andrew David Greenhut

Submitted to the Department of Mechanical Engineering
on December 15, 2009 in Partial Fulfillment of the
Requirements for the Degree of Master of Science in
Mechanical Engineering

Abstract

Innovative solar-geothermal hybrid energy conversion systems were developed for low enthalpy geothermal resources augmented with solar energy. The goal is to find cost-effective hybrid power cycles that take advantage of the potential synergies of solar thermal and geothermal resources. One aspect is to determine the hybrid configuration that yields the highest annualized electricity generation. The levelized cost of electricity (LCOE) is estimated using equipment costing rules of thumb developed from Aspen HTFS and Aspen ICARUS software and from other sources.

Detailed models for the hybrid solar-geothermal system were developed using Aspen Plus and Aspen Dynamics. Turbine flexibility relative to vapor flow rate, temperature and pressure variations was analyzed. In one scenario, a parametric steady-state study was carried out to examine the performance over the range of conditions resulting from diurnal and seasonal variations. The results of the diurnal and seasonal parametric studies were grossly weighted to approximate a typical year in Nevada, and these results led to an estimate of the annualized electricity generation. In another scenario, a dynamic model was selected from possible "greenfield" hybrid systems and used to examine the transient performance for a typical January day and a typical July day in Nevada. The dynamic model approximates the thermal inertial of the heat exchangers and the working fluids in the exchangers, solar collectors, piping and storage tanks. The dynamic model is driven with forcing functions for solar input and ambient temperature to approximate the typical winter and summer days.

In all cases, solar energy was found to come at a higher cost per kW capacity than geothermal when the cost of geothermal wells was not considered. However, including well costs had an effect of evening out the levelized cost of electricity. Model complexity increased as more solar heat was added to existing geothermal systems, which suggests that moving a higher exergy heat source down to a lower exergy heat source is difficult, especially given the transient nature of the solar resource. The models developed in this thesis demonstrate the design decisions and complex dynamic behavior inherent in this type of hybrid system.

Thesis Supervisor: Jefferson W. Tester
Title: H.P. Meissner Professor of Chemical Engineering

Acknowledgments

There are many people who have made invaluable contributions to my research and me over the past several years. It would be impossible to thank you all here individually but that does not mean this support has gone unnoticed. Specifically, I would like to acknowledge Prof. Jefferson Tester, my thesis advisor, for his leadership, guidance, and friendship. I would also like to thank Prof. Randall Field and Prof. Ron DiPippo for all their efforts in patiently listening, teaching, and guiding me throughout this project. In addition, I want to thank my mechanical engineering advisor, Prof. Ahmed Ghoniem, for our stimulating conversations and his help throughout my MIT masters program. I truly appreciate all of your support.

Finally, I am deeply grateful for the encouragement and love of my family and especially my wife, Gita.

Table of Contents

List of Figures

List of Tables

Chapter 1

1. Introduction

1.1. Project Overview and Motivation

At this very moment, our species is conducting the biggest experiment in our history, and our planet is the test specimen. The question of this experiment is: "what happens to our planet when we perturb our atmosphere with a step function increase in carbon dioxide and other greenhouse gases?" Of course, we are not intentionally running this experiment, but rather we are all conducting it by virtue of the way we use energy resources. The fact is, most of our energy comes from carbon-based sources (known collectively as fossil fuels), and the use of them increases the concentration of carbon dioxide into our atmosphere. In addition, our emissions of methane, nitrous oxide, and other halocarbons further exacerbate this problem.

In order to end this experiment, our global community must make bold decisions. One of these is to remove carbon from our energy sources. Moving to non-carbon or carbon-neutral sources of energy, such as nuclear, solar, geothermal, wind, wave, tidal, and/or biomass can end our dependence on carbon-based fuels. However, harvesting these sources of energy will require new infrastructure and methods that currently are on average more expensive than carbon-based sources. In addition, many of these other sources of energy come with unique constraints that we are not accustomed to working with, which further add to costs of development.

Other than greenhouse gas abatement, there are other reasons to transition to a new energy system. Our current energy system is unsustainable since we will eventually run out of fossil fuels. If we do not figure out a way to power modern society by other means, then economies will suffer as we pay a higher and higher price for energy. Some believe this will lead to political instability and wars. From the United States perspective, there are other internal reasons to move away from fossil fuels. For example, reducing our use of oil would allow us to important less from the Middle East region, and reducing our burning of coal would reduce local pollution of harmful chemicals into our environment. In additon, we could move to a new energy system that uses energy more efficiently, whether it is fossil-based or not. This would save us energy and money.

This thesis aims to explore new technologies in renewable energy with the hope that it will accelerate adoption of carbon-free sources of energy. Specifically, this thesis focuses on two sources: geothermal and solar energy. The question to be answered is: "Can these two sources of energy be combined to produce a single more efficient and lower cost hybrid system than individually?" In other words, are there synergies to a geothermal-solar hybrid energy conversion system, and should we pursue this technology? Since this question is very broad, this thesis will focus on specific geothermal and solar energy conversion methods that are assumed to be most suitable, and then analyze two possible hybrid scenarios within this scope. These analyses will range from steady state snapshots of models to a fully dynamic model, with minute-to-minute fluctuations. The results here will offer insights on hybrid geothermal-solar energy conversion

systems, and general design implications will be extracted. Finally, alternative hybrid systems that were not analyzed in this thesis, but potentially better, will be suggested for future research.

1.2. Introduction to Solar and Geothermal Energy

This thesis examines geothermal energy (heat from inside the Earth), and solar energy (radiation from the sun). The term geothermal energy usually refers to the heat inside the Earth stored in hot water, or hydrothermal resources. However, it can also refer to other categories of heat inside the Earth, namely conduction-dominated heat in dry rock, geopressured systems, and coproduced fluids from oil and gas wells. Alternatively, geothermal energy can sometimes refer to ground source heat pumps, where energy on the surface is exchanged with shallow wells dug into the ground. This thesis focuses on hydrothermal resources, which are by far the most utilized form of geothermal energy in the world. Chapter 2 will discuss how to convert this resource into useful forms of energy and the specific technology that will be modeled in the hybrid system.

Solar energy refers to energy that comes directly from the sun's radiation. This is not to be confused with indirect forms of solar energy, such as wind and biomass, which are harnessed in other ways. The two main ways to utilize solar energy are with photovoltaic devices and through thermal heat collection. Photovoltaic devices absorb photons from the sun, which directly excite a flow of electrons to generate electricity. Solar heat can be used for general space heating and cooling, or it can be concentrated into a heat transfer fluid, which operates a thermodynamic cycle to convert heat into electricity. This thesis will look at solar thermal energy conversion methods for electricity production.

It is no coincidence these two resources were selected for this study. Besides both being renewable and carbon-free sources of energy, they both have high potential in similar geographic areas. In the United States, areas of high geothermal and solar energy potential intersect in the west and southwest mainland, as seen in **Figures 1.1** and **1.2**.

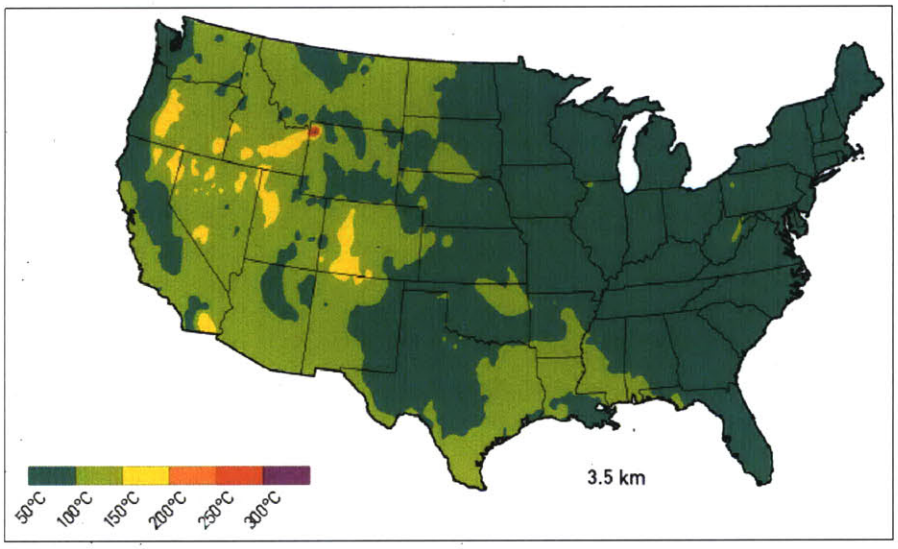


Figure 1.1 Average temperature at depth of 3.5 km [53].

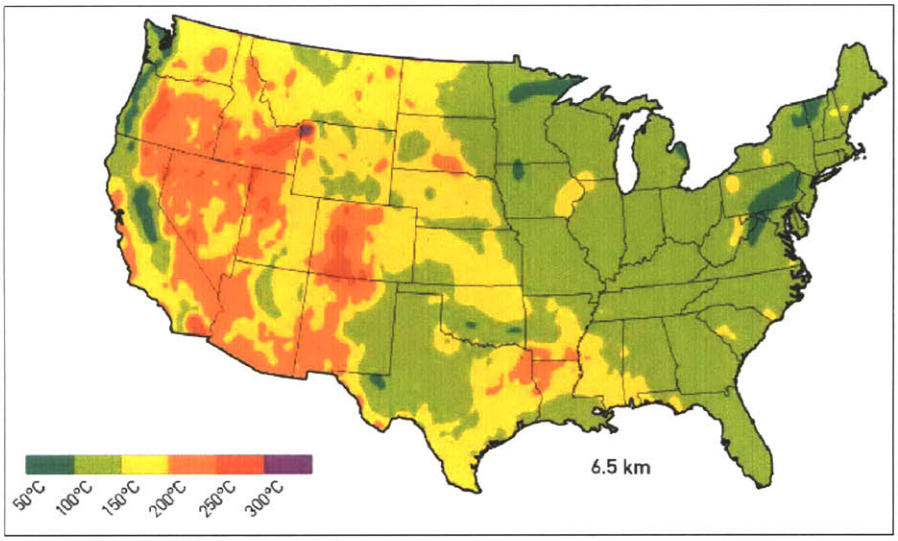


Figure 1.2 Average temperature at depth of 6.5 km [53].

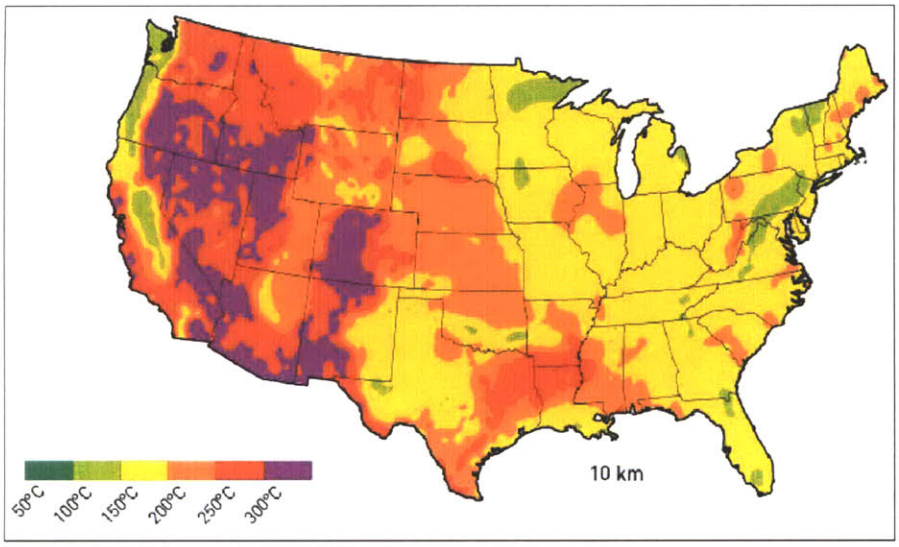


Figure 1.3 Average temperature at depth of 10 km [53].

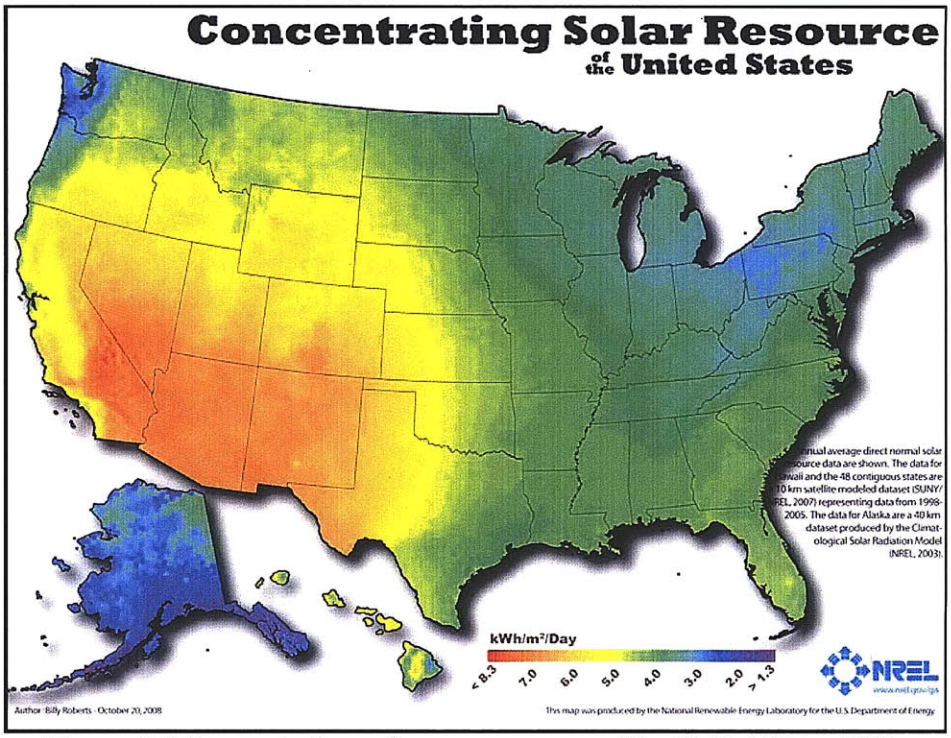


Figure 1.4 Concentrating solar resource map of the United States [36].

Because both solar thermal and geothermal energy come in the form of heat, they both are converted to electricity through a thermodynamic cycle. It turns out that these two resources are both best utilized through a Rankine cycle because of the magnitude of their temperatures. However, different types of Rankine cycles are better designed for different temperature resources. For higher temperatures above 200°C, water is typically used as the working fluid for the Rankine cycle. For lower temperatures, an organic fluid is typically used.

The laws of thermodynamics tell us that the lower the temperature of a resource, the lower the availability or exergy or power producing potential which leads to a lower maximum thermal efficiency that can be realized –Then, it can be said that a solar resource at 400°C has considerably more power producing potential than a geothermal resource at 200°C. There is a caveat, however, which is that the sun is not always shining. In fact, the amount of energy is radiates on average, or the insolation, is a function of time (or position in the sky). Therefore, the usefulness of solar energy is constantly changing. In order to mitigate this issue, storage of heat is employed to extend the effective hours of solar heat into the night. However, this comes at a cost, and this thesis will examine hybrids with different amounts of solar storage to see what the tradeoffs are to this approach.

In contrast to solar heat, geothermal energy is available all the time. Geothermal reservoir temperatures may vary from too low (< 120°C) for economic electricity production, to upwards of 300°C. The challenge is finding geothermal hot spots, where high temperatures are found in relatively shallow depths. In all cases, the resource available at a given location is always limited

by the size and productivity of the reservoir. Thus, it is challenging to expand existing geothermal power plant capacity when you locally reach the limit of your resource.

1.3. Thesis Objectives

The objective of this thesis is to develop innovative solar-geothermal hybrid energy conversion system models for low enthalpy geothermal resources augmented with solar energy. A low enthalpy geothermal resource means that the brine, or water from the Earth, comes out at a temperature between 100-150°C, which is considered too low for economic electricity production. By augmenting the thermodynamic quality of the brine with solar energy, higher performance should be possible. The goal is to find cost-effective hybrid power cycles that take advantage of the potential synergies of solar thermal and geothermal resources. These synergies may be related to the differing diurnal cycles of the resources, as well as to using the lower enthalpy (or exergy) geothermal energy to enhance the power generation of higher enthalpy (or exergy) solar energy, or *vice versa*. Such combinations would be desirable because there are many locations in the U.S. and world with low enthalpy geothermal resources, and reasonable quality solar thermal resources.

1.3.1. Basis for hybridization

The proposition for hybrid geothermal and solar thermal energy conversion systems are based on several potential synergies that exist between them. These include (but are not limited to):

- Ability to Meet Demand Better: Geothermal power plants, especially those in water scarce areas, commonly use air-cooled condensers instead of water condensers or wet cooling towers and suffer a decrease in power produced during the day because of the increase in ambient temperature. This increase in temperature reduces the plant's ability to reject heat to the environment and reduces overall efficiency. Solar thermal energy can be used to boost power plant performance during the day because that is when the sun is shining the most.

- Resource Location: As indicated earlier, high-potential geothermal and good solar thermal resources are located in similar geographic areas in the U.S. In addition, these resources also intersect in other parts of the world, such as Central America, Indonesia, and the Mediterranean region.

- Financial Mitigation: A hybrid system can mitigate the high cost of solar projects with the low cost of geothermal projects. As solar technology improves, this balance may shift, but the knowledge gained from working with both technologies will allow one to be in a better position when that time comes.

- Equipment Sharing: Because both solar thermal and geothermal resources are developed with similar energy conversion methods, there is the potential for both energy sources to share common equipment, such as turbines, condensers, and heat exchangers. This allows more equipment to run full time even though the sun is intermittent.

- Boosting Power to Existing Plants: Geothermal power plants can be limited by the underground resource in a particular area. However, solar thermal technology can boost the

performance of existing hardware and take advantage of existing transmission infrastructure built for the original plant.

- Ability to Capture Economic Incentives: Different economic incentives are available for different technologies. By combining geothermal and solar technology, hybrid systems can qualify for more forms of economic support.

- Publicity: Geothermal and solar power are both renewable sources of energy, however they are only a very small percentage of total energy produced in the world. Hybrid systems with only renewable energy can capture attention and create a buzz for the industry.

1.4. Approach

The analyses conducted in this thesis can be divided into three main sections:

1. Chapters 2 and 3: Describes existing technology for geothermal (Ch. 2) and solar thermal (Ch. 3) energy. These chapters prepare the reader for the analyses performed in later chapters. In particular, these chapters describe the energy conversion methods modeled in this thesis.

2. Chapters 4 and 5: Each chapter focuses on one of two scenarios. Chapter 4 focuses on hybrid systems in which solar energy augments *existing* geothermal power plants. Chapter 5 focuses on new hybrid systems

3. Chapters 6 and 7: Discusses the applications of hybrid systems and their effectiveness. These chapters look at the results from the hybrid models and draw conclusions about their predictions. These chapters also discuss some of the assumptions built into the models, and how these have affected the results.

1.4.1. Modeling

This is a breakdown of the steps conducted in each scenario. The first few steps are similar for both scenarios because these steps are for conceptual design and selection. The main difference between analysis in scenario A and analysis in scenario B is that scenario A stays in steady-state while scenario B goes to dynamic models. In one sense, this means the modeling for scenario B is more advanced. Also, note that the same software packages are used for both scenarios, such experience from scenario A was carried over to scenario B.

Scenario A:

- Several power cycle configurations were evaluated in terms of both thermodynamic and economic metrics.
- Accurate pure component property data and equation of state models were utilized where possible using the Aspen Plus platform for mass and energy balance calculations.
- Turbine flexibility relative to vapor flow rate, temperature, pressure variations was analyzed.
- Detailed steady-state models for the solar-geothermal system were developed.
- A parametric steady-state study was carried out to examine the performance over the range of conditions resulting from diurnal and seasonal variations. The results of the diurnal and seasonal parametric studies were grossly weighted to approximate a typical year near Reno, Nevada, and these results led to an estimate of the annualized electricity generation.
- Analysis focused on determining the hybrid configuration that yields the highest annualized amount of electricity generation. The levelized cost of electricity (LCOE) was estimated.

Scenario B:

- Several power cycle configurations were evaluated in terms of both thermodynamic and economic metrics.
- Accurate pure component property data and equation of state models were utilized where possible using the Aspen Plus platform for mass and energy balance calculations.
- Turbine flexibility relative to variations in working fluid flow rate, temperature, and pressure was analyzed.
- Detailed steady-state models for the solar-geothermal system were developed.
- In this scenario of a new plant, one working fluid was selected, R134a, and one type of binary cycle was chosen, a supercritical-pressure cycle (the purpose of these selections is discussed in chapter 2)
- A dynamic model was created based on the selected steady-state hybrid system, and was used to examine the transient performance for a typical January day and a typical July day.
- The dynamic model approximates the thermal inertial of the heat exchangers and the working fluids in the exchangers, solar collectors, piping and storage tanks. The model does not represent the moment of inertia of the turbine, since that time scale is not relevant to this study.
- The dynamic model is driven with forcing functions for solar input and ambient temperature to approximate the typical winter and summer days.
- Analysis focused on determining the hybrid configuration that yields the highest annualized electricity generation. The levelized cost of electricity (LCOE) was estimated using equipment costing rules of thumb developed from Aspen HTFS V7.1 and Aspen ICARUS V7.1 software and from other sources [38].

Chapter 2

2. Geothermal Energy Conversion

2.1 Overview

Geothermal energy is under our feet, everywhere. On average, heat flows through the crust to the surface at a rate of 5.9×10^{-2} W/m^2 [55]. For centuries, humans have known about this resource in the form of hot aquifers or geysers, but it was not until 1904, in Larderello, Italy, that we started to harness the energy for electricity. In places like Larderello, hot magma inside the Earth penetrates close to the surface, heating the local area (sometimes the magma reaches the surface in the form of volcanoes). If water is present deep in the ground, it will be heated to high temperatures and possibly turned into steam. This hot brine may stay trapped in the Earth, or it may escape to the surface in the form of geysers, fumaroles, hot springs, or solfataras.

Human have become very good at harnessing the energy in hot brines, known as hydrothermal geothermal energy. We have an installed world electrical generating capacity of over 10,000 MW_e, and a non-electric capacity of more than 100,000 MWt (direct heating and cooling) [53]. Based on the temperature and permeability and depth of a reservoir, we assign a "quality" to that resource, where "high quality" refers to high temperature, high permeable, and relatively shallow resources (<3 km deep). In addition, the geochemical composition of a resource plays a role with how easily a resource can be utilized, where low salinity and low concentrations of dissolved non-condensable gases are easier to exploit. Therefore, the economics of power production look more favorable when exploiting the highest quality resource available. Unfortunately, many opportunities remain undeveloped, even though they may be economically favorable. For example, we know that the hydrothermal resource base to 10km depth is between 2,400 – 9,600 Exajoules (1EJ = 10^{18} J), yet today's hydrothermal systems rarely go deeper than 3km [53] and all the possible resources at 3km are not developed.

2.2 Geothermal Systems Considered in this Thesis

2.2.1 Binary Plants

For low-grade geothermal reservoirs considered in this thesis, the common type of power plant to build is a Rankine cycle that employs an organic working fluid. These are commonly called binary cycle power plants. Today, binary plants are the most widely used type of geothermal power plant with 162 units in operation, generating 373 MW of power in 17 countries, and making up 32% of all geothermal units, but only 4% of the total power [10]. **Figure 2.1** shows a schematic of the basic binary plant.

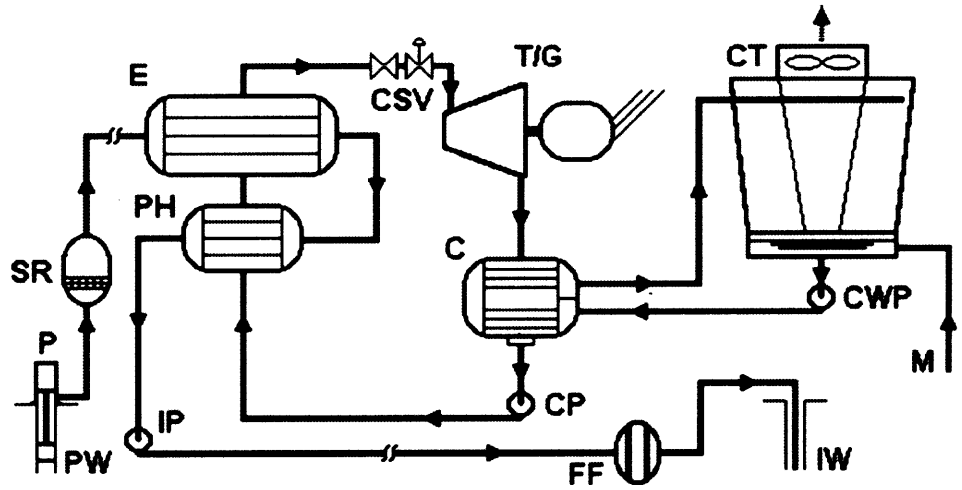


Figure 2.1: Simplified schematic of a basic binary geothermal power plant [10].

The thermodynamic process is quite simple. Brine from the ground comes up from the Production well (P) and passes through two heat exchangers before returning to the ground. The brine in the evaporator (E) provides the latent heat to boil the working fluid (WF), and the brine in the preheater (PH) provides the sensible heat to bring the WF to the boiling temperature. Therefore in a binary cycle, the brine remains a liquid at all times (under enough pressure so it doesn't boil). This prevents minerals inside the brine from depositing on the inside surface of the pipes, and prevents any non-condensable gases in the brine from being released to the atmosphere.

In the WF loop, the WF boils in the evaporator and then passes through an organic turbine, where it expands and transfers energy to a generator to produce electricity. The WF is then condensed with an air or water cooled condenser (the hybrid models later all use air-cooled condensers because water is limited in the western U.S.) and pumped back up to a higher pressure. The change of state of the WF can be seen in **Figure 2.2**.

Notice that there is no risk of the WF expanding into a wet region from states 1 to 2. This is because of the special characteristic of organic fluids, where the saturated vapor curve has a positive slope. This means that organic fluids do not have to be superheated before going to a turbine, which lowers the cost of a binary system. Also note that the critical temperature of organic fluids tend to be between 100-200°C, which indicates that supercritical power cycles are feasible for organic fluids.

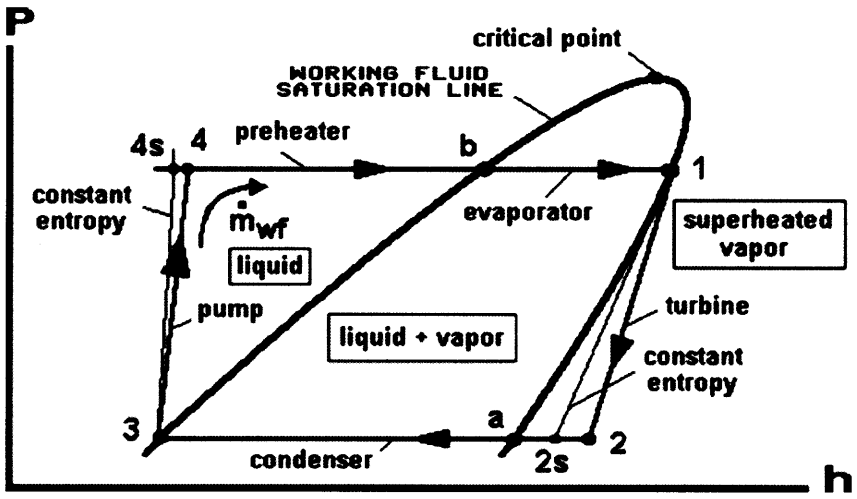


Figure 2.2: Pressure-enthalpy diagraph for a basic binary plant [10].

One concern of binary working fluids is that they can be strong greenhouse gases (GHG) if released into the atmosphere. Although they are not intended to be released, their widespread use can potentially increase GHG emissions through mishandling. This means that certain binary fluids, especially the refrigerants, will likely be banned in future international agreements (Some have already been banned – i.e. R-12 and R-114).

2.2.2 Binary Cycle Supporting Literature

In 2008, Tester et. al. [54] completed an unpublished study on the "Utilization of Low-Enthalpy Geothermal Fluids to Produce Electric Power." The main objective of that study was to "evaluate power conversion options for geothermal applications and to propose designs for an innovative, cost-effective binary power plant for geothermal applications." Several power cycles and working fluids were analyzed for thermodynamic and system performance with brine temperatures between 100-200°C. The study identified best case working fluids for different power cycles given a range of geothermal fluid temperatures.

The working fluids considered in this study were chosen based on earlier studies [25] and existing plants. In general, candidate fluids have critical temperatures and pressures far lower than water [10]. This makes it also feasible to consider supercritical cycles.

The working fluids considered are:

- Propane (C_3H_8) (e.g., East Mesa Magmamax bottoming loop)
- Isobutane (i-C_4H_{10}) (e.g., EM Magmamax main loop)
- Normal pentane (n-C_5H_{12}) (e.g., Miravalles Unit 5)
- Isopentane (i-C_5H_{12}) (in use today)
- Cyclopentane (C_5H_{10})
- Toluene (C_7H_8)
- Isobutane-isopentane mixtures (e.g., 90% - 10%)

- Water-ammonia (H_20-NH_3) (e.g., Kalina plants)
- Refrigerant R-134a (e.g., Chena Hot Springs)
- Refrigerant R-245fa
- Refrigerant R-32

The two major power cycles modeled in this study were:

- Supercritical basic binary w/ heat regenerator, if feasible
- Subcritical basic binary w/ heat regenerator, if feasible

Figure 2.3 below shows the Aspen Plus model created by [54] to test various WF in a supercritical cycle. In this cycle, there is no preheater. This is because a supercritical cycle does not go through a phase change, and two heat exchangers are not necessary. Instead, one heat exchanger can be used to transfer all the heat from the brine to the WF.

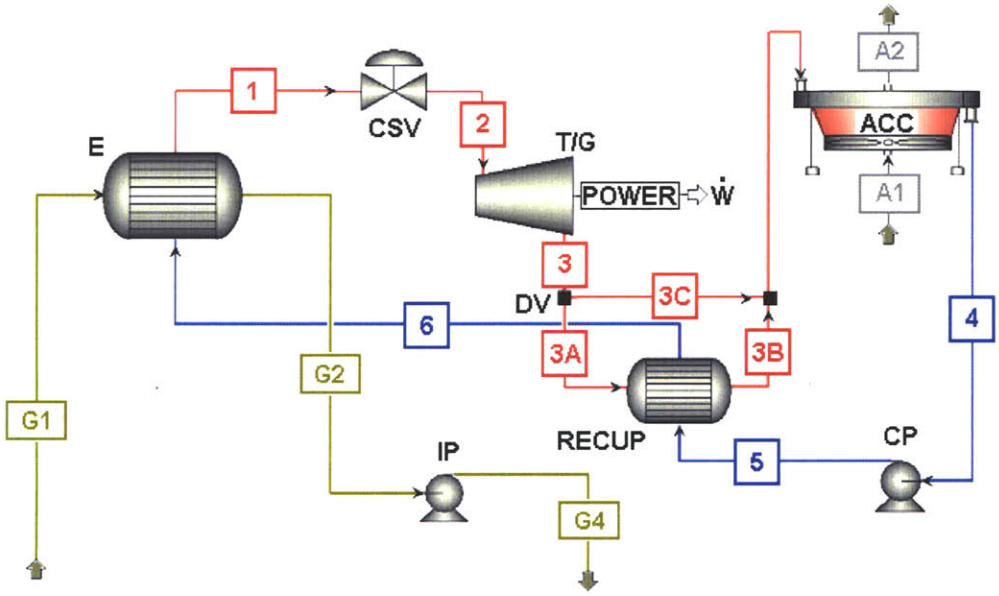


Figure 2.3: Supercritical binary plant, shown with recuperator. The diverter valve (DV) is a computer artifact that directs the full WF flow to either state 3A when a recuperator is feasible or to state 3C when one is not feasible [54].

Figures 2.4 and **2.5** show results from Tester et. al. Utilization efficiency is on the y-axis and geothermal production temperature on the x-axis. The utilization efficiency is based on the second law of thermodynamics, and is given by the ratio of net power output to the exergy of the brine coming into the plant, viz.

$$\eta_u = \dot{W}_{net}/\dot{E}_{in}\,, \tag{2.1}$$

Where the geofluid exergy, $\dot{E_{in}}$, is given by:

$$\dot{E}_{in} = \dot{m}[h_{in} - h_0 - T_0(s_{in} - s_0)]\,. \tag{2.2}$$

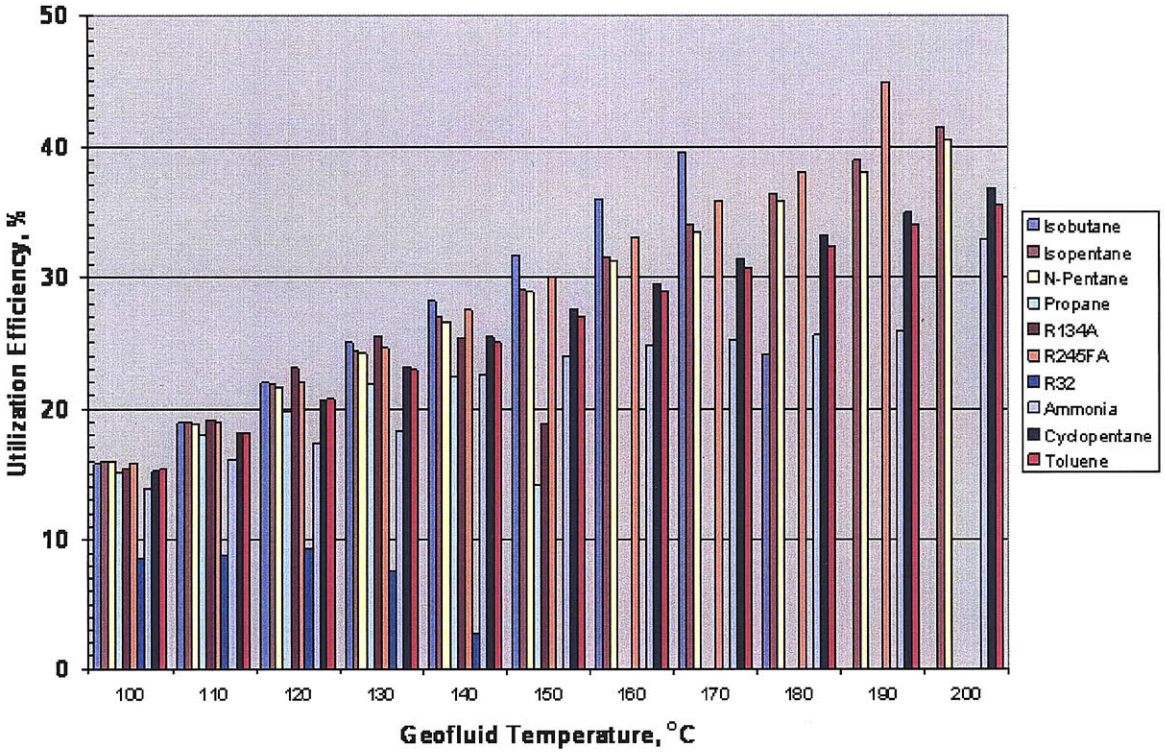


Figure 2.4: Utilization efficiency (%) for subcritical cycles as a function of geofluid temperature (°C) for ten different candidate working fluids [54].

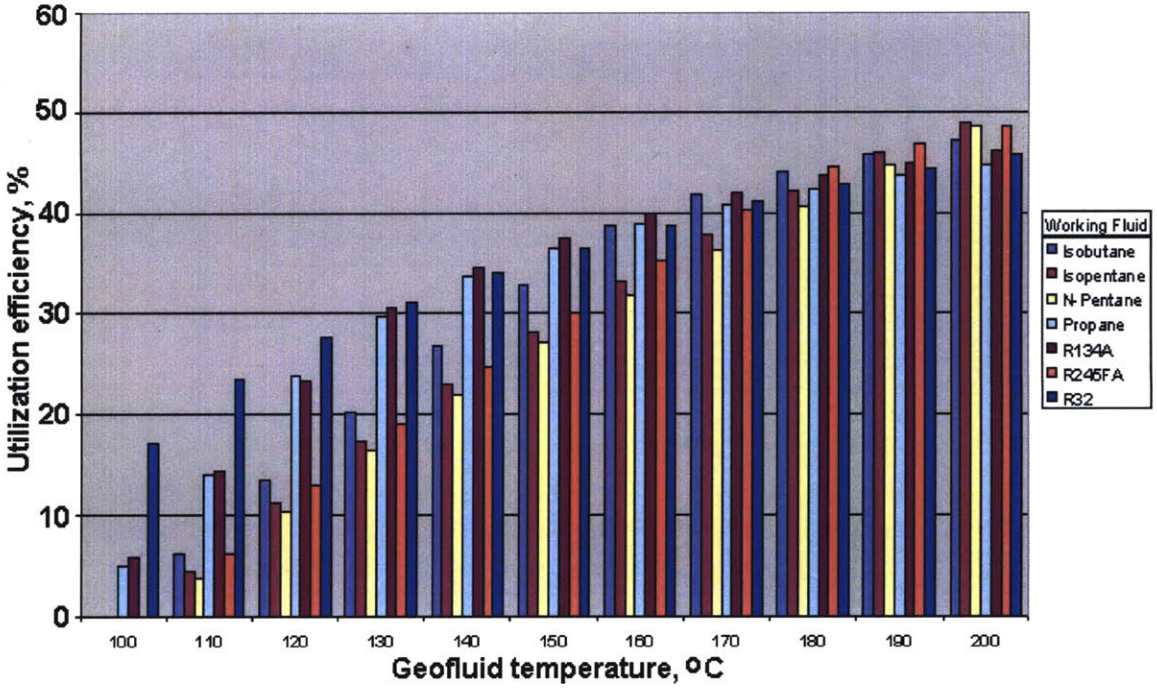


Figure 2.5: Utilization efficiency (%) for supercritical cycles as a function of geofluid temperature (°C) for seven candidate working fluids [54].

One power cycle and working fluid combination was identified for further study based on its superior performance under a range of temperatures. This was the supercritical binary cycle with R-134a as the WF. This cycle was selected for further analysis in scenario B of this thesis.

2.2.3 Single-Flash Steam Power Plants

If solar heat is added to the low enthalpy geothermal brine, the hybrid system can potentially be turned into a single-flash steam power plant. Single-flash plants account for about 32% of all geothermal plants, and constitute over 42% of the total installed geothermal power capacity in the world [10]. In this type of plant, brine from the ground undergoes a flashing process in which the pressure of the brine is throttled down, and then the steam and concentrated liquid brine is separated into two different paths. The concentrated liquid brine is reinjected into the ground, and the steam is used to power a steam turbine. **Figure 2.6** shows a simple schematic of a single-flash power plant.

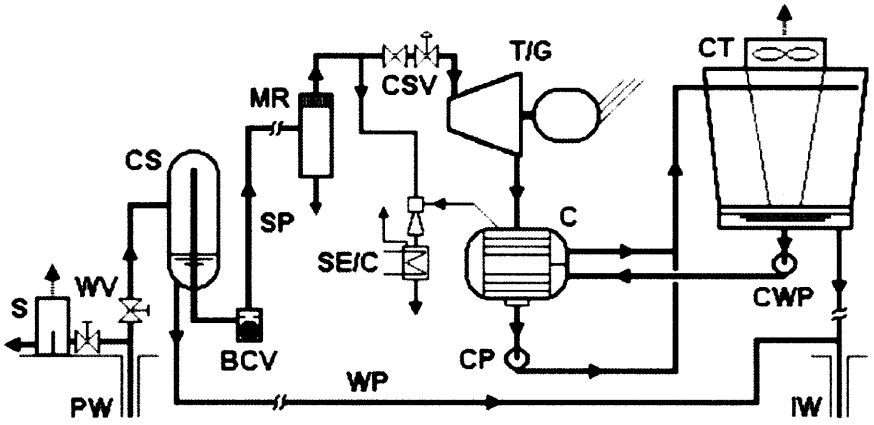


Figure 2.6: Simplified single-flash power plant schematic [10].

From **Figure 2.7**, the process of a simple single-flash power plant can be seen on a temperature-entropy diagram. From state 1, the brine is flashed to state 2 in a nearly enthalpic path. The separator splits the liquid brine to state 3 and the steam to state 4. Unlike the organic fluids in the binary cycle, water has a negative slope on its saturated vapor curve. This means that when the steam expands through the turbine from state 4 to 5, some vapor turns into liquid. This reduced the performance of a steam turbine. To prevent this from happening, the steam must be superheated so that it expands only in the superheated vapor region. It has been proposed to use fossil fuels for this superheat (Ron DiPippo, 1978), however transporting necessary fuel to remote geothermal plants may be costly. Instead, consider a solar-geothermal hybrid system where solar energy is used to provide superheat for the steam. The interesting question to ask is whether the cost of harnessing solar power just for superheating steam is worth the thermodynamic benefit. Another idea would be to use solar power to create more steam, as was done in Handal, 2007 [2].

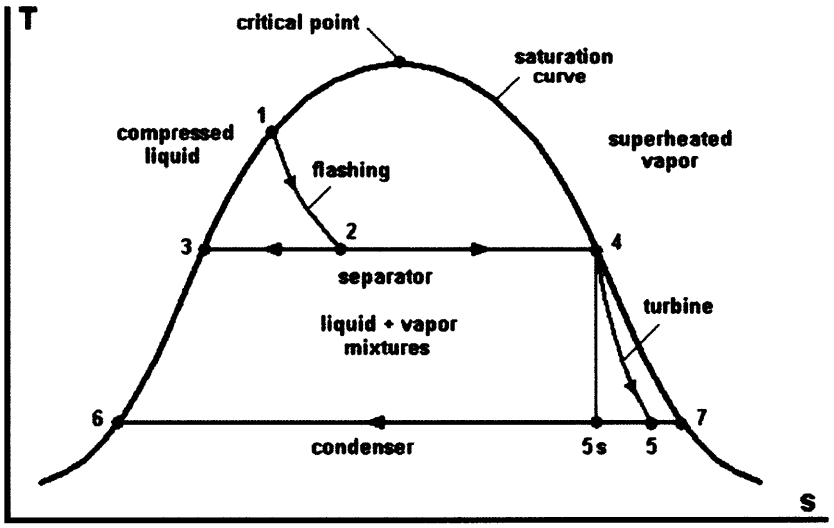


Figure 2.7: Temperature-entropy state diagram for single-flash plants [10].

Chapter 3

3. Concentrating Solar Thermal Energy Conversion

3.1 Overview

Solar thermal energy is heat energy from the sun, not to be confused with the ambient air temperature. The sun can be radiating just as much heat in January as in July depending on the location and time of day. By concentrating this radiation, heat can be collected at sufficiently high temperatures to make electricity.

The devices uses to concentrate the sun's radiation are called concentrating solar collectors There are four types that have been demonstrated with at minimum large scale prototypes. These four types are the parabolic dish, the parabolic trough collector, the power tower (heliostat field focused on a central receiver) and the compact linear Fresnel reflector. **Figures 3.1** to **3.4** show these designs in action. The arrows show the path of the sun's heat and how they concentrate on a point on line. Only trough collectors have been deployed in commercial service so far.

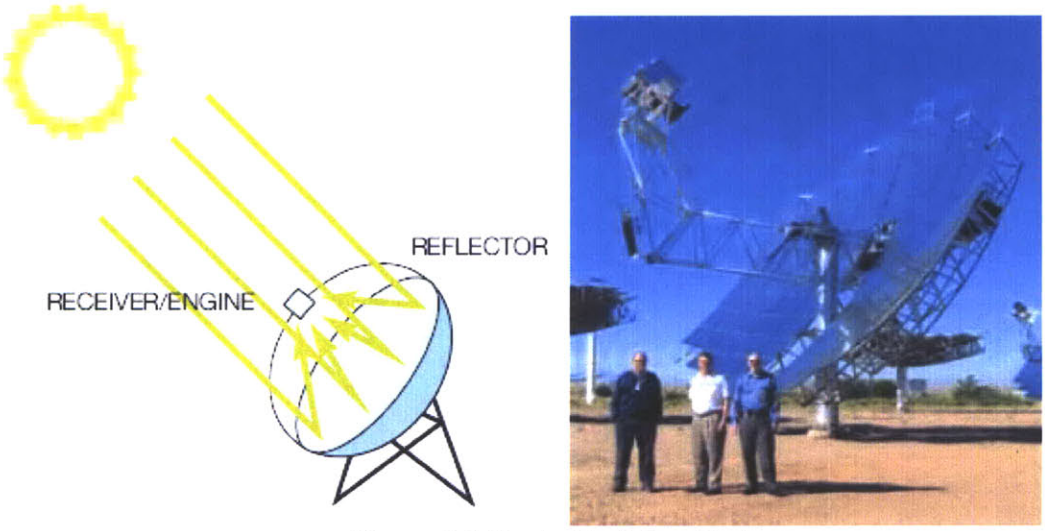


Figure 3.1 Parabolic dish collectors.

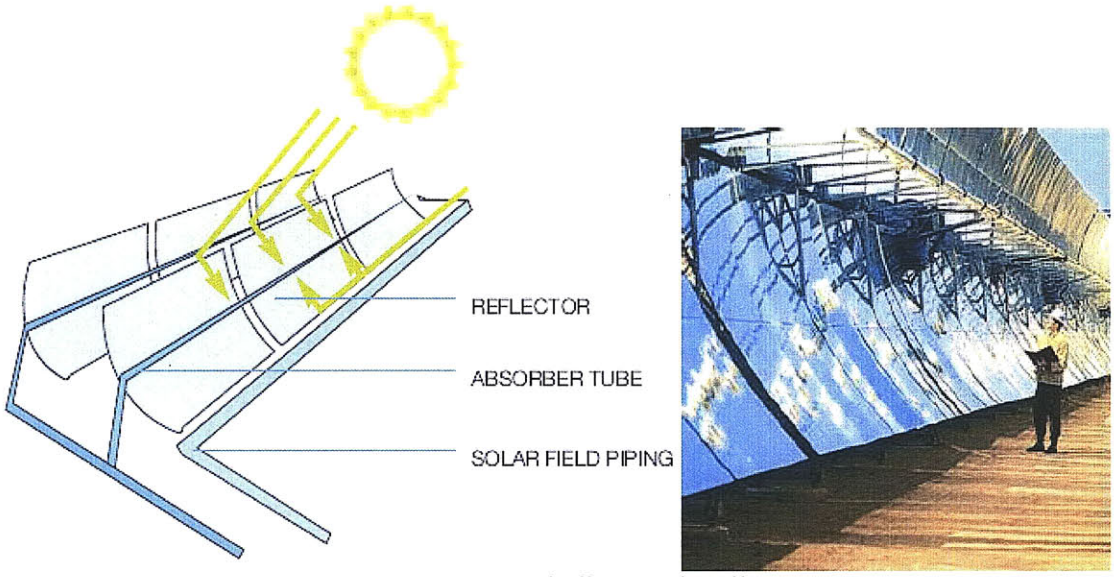


Figure 3.2 Parabolic trough collectors.

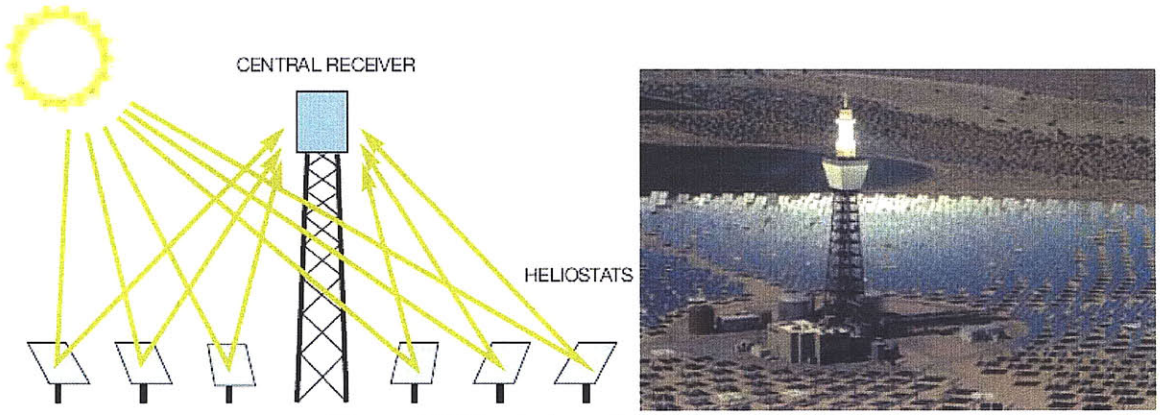


Figure 3.3 Central receiver with heliostats.

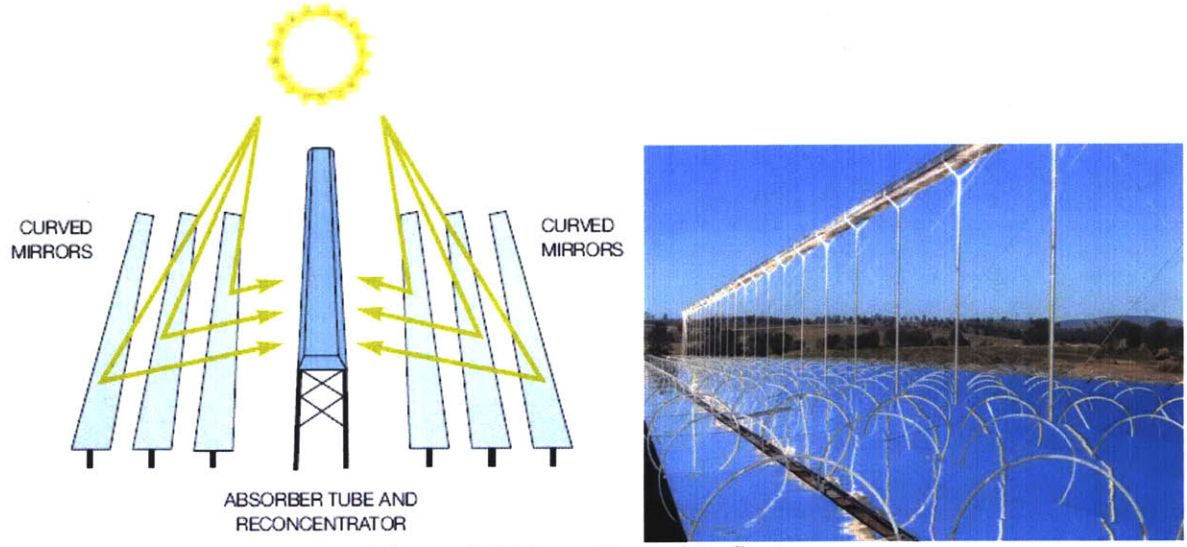


Figure 3.4 Linear Fresnel Reflector

Solar thermal collectors have been known for at least 100 years, but historically their appeal has been extremely low due to the high cost of collector technology. In the 1980s and early 1990, the company Luz famously built 9 parabolic trough collector solar thermal plants in the Mojave Desert, CA with the help of government incentives for solar technology (collectively known as SEGS). Since then, development of new plants has been nearly non-existent until very recently with a new resurgence in government assistance. Today, just over 500MW of generating power is supplied by solar thermal power, and another 1000MW is in construction [35]. Beyond that, several GW of power is being proposed and planned in the next decade. The majority of this development is with parabolic trough collectors.

Table 3.1 summarizes the operating characteristics of different solar collectors. The most mature technology is the parabolic trough, with over 20 years operating in California. The other technologies have only been constructed in the last few years, but hold promise. For example, compact linear Fresnel reflects are much simpler to build, and are potentially cheaper on a large scale, however it is too soon to tell if they will grow to compete with the parabolic trough.

Table 3.1 Operating characterisics of CSP technology [54, 34].

CSP Technology	Concentration Ratio	Tracking Requirement	Operating Temperature	Ave. solar to electric efficiency	Unit size range	Installed Capacity 2009
Power Tower	500-1,000	2-axis heliostats	400-600°C	12-18%	30-200 MWe	40 MW
Parabolic Trough	10-100	1-axis reflector	100-400+°C	8-12%	30-100 MWe	500 MW
Dish-Engines	600-3,000	2-axis	600-1,500°C	15-30%	5-50 kWe	0.5 MW
CLFR	<100	1-axis reflector	100-300°C	<10%	1-50 MWe	5 MW

Cost is an important concern to selecting a collector for the hybrid system. Unfortunately, this is very hard to predict with accuracy. Any industry estimates that are available are for the specific power plant designs and make it difficult to isolate the collector costs only. Still, reports by SunLab and the DOE [47] estimate parabolic troughs to cost the least in the short term. CLFR may yet cost less, but they will not be considered in this thesis because of their relatively new entry.

Another important characteristic of these collectors is the operating temperatures. If the collectors operate at a temperature much hotter than the geofluid, the mismatch may be thermodynamically suboptimal. From this point of view, parabolic trough collectors and CLFRs seem to be the best choice.

Given the temperatures of the geofluid, the preferred option might be to use cheaper, lower-temperature collectors, including evacuated flat-plate, parabolic trough with water, and other linear Fresnel. At 150°C, the evacuated-tube flat-plate collectors are a few percent more efficient than conventional parabolic trough collectors (see **Figure 3.5**). These collector types are in the research and development phase and can be considered as future market options for the hybrid plant (for more information, see [45]). However, early cost indications show there is not much to save by switching from parabolic troughs to flat-plate collectors.

3.2 Solar Thermal Systems Considered in this Thesis

This thesis will consider state-of-the-art parabolic trough collectors for the hybrid system. To develop our understanding of this technology, an operational CSP plant and the collectors will be examined.

Figure 3.6 shows a schematic of Andasol-1, a power plant built in 2008 by ACS/Cobra Group and Solar Millennium Group. The power plant can be divided into three subsystems, the solar field, the power block, and the thermal storage system (optional). Each system has a different working fluid. The solar field uses a synthetic oil heat transfer fluid, such as Therminol VP-1, the power block uses steam, and the thermal storage uses a form of liquid salt. Thermal storage will be discussed further in the next section, however to explain this design, the reason you cannot store heat in the synthetic oil is because the vapor pressures are high and a large pressurized tank would be very expensive.

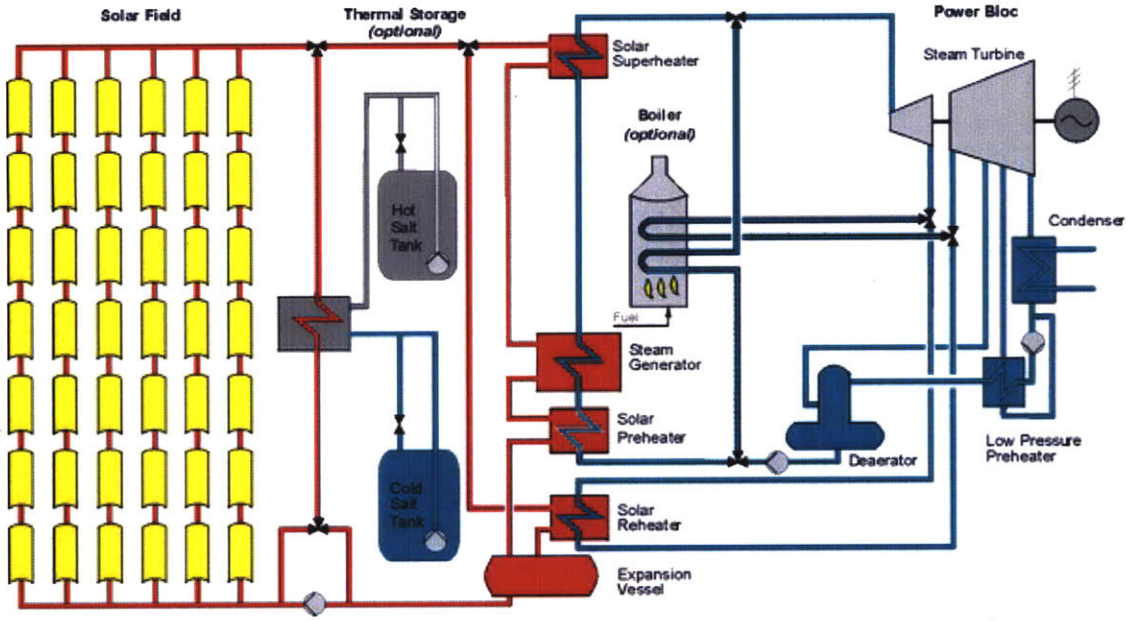


Figure 3.5 Parabolic trough power plant with storage schematic: Andasol-1.

The heat transfer fluid (HTF) in the solar field will heat to nearly 400°C and transfer that heat to a steam cycle at different points. Note that there is an optional fossil fuel powered boiler to supplement the solar energy when the sun sets and thermal storage is depleted for the night.

3.2.1 Parabolic Trough Collectors

Parabolic trough collectors comprise of 4 subsystems: the stationary structure and rotating troughs, the mirrors on the surface of the troughs, the heat collection elements (HCE) that receive the concentrated radiation and contain the HTF, and the control system that determines the 1-axis position of the trough at any time of the year. For most systems, the structure and controls are relatively standard engineering with costs to build being the main differentiator. However, the parabolic trough and HCE design have a significant effect on the overall collector performance.

Figure 3.6 shows a cross-sectional diagram of the trough (5) and HCE (3). Incident radiation (1) comes from the sun and reflects towards the parabola focal point (2). A structure holds the HCE in a fixed position relative to the trough (4). The radius of the parabola at any location is denoted by r, and is called the "mirror" radius. The corresponding angle it forms with the receiver support bracket is denoted by φ. The mirror radius and angle are related by the following equation [6]:

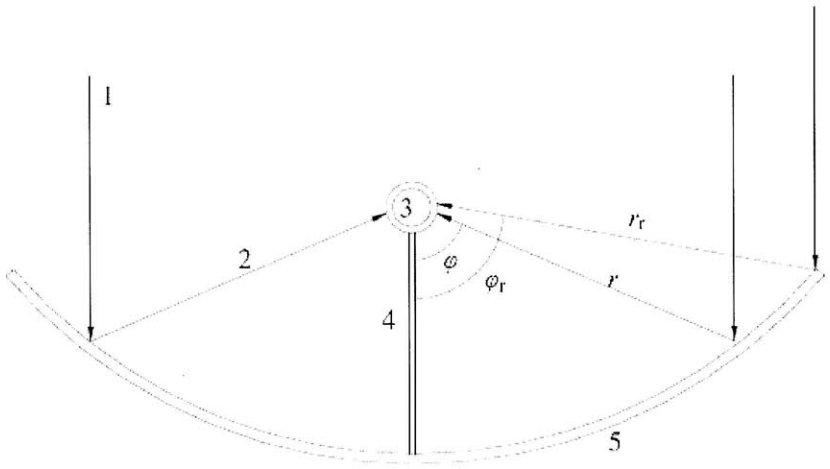
$$r = \frac{2f}{1+\cos\varphi}, \tag{3.1}$$

where f is the constant focal length.

The collector's effectiveness can generally be measure by the concentration ratio:

$$C = \frac{\text{flux density at receiver}}{\text{flux density at aperture}} = \frac{E}{{}^{s}E_b}, \tag{3.2}$$

where C is the concentration ratio, ${}^{s}E_b$ is the direct solar irradiance, and E is the total direct, diffuse, and ambient radiation. C measures the density of radiation intensity at the receiver relative to the density that reaches the collector. A typical concentration ratio for a parabolic trough, as indicated in **Table 3.1**, is between 10 and 100.



1. Incident Beam Radiation	4. Receiver Tube Support Bracket
2. Reflected Beam Radiation	5. Parabolic Trough Reflector
3. Receiver Tube	

Figure 3.6 Cross-Sectional View of Parabolic Trough Solar Collector (Not to Scale) [6].

To measure the efficiency of a solar collector, a more precise equation is provided by Winter, et. al.[57] viz.:

$$\eta_c = \frac{Q_s}{{}^{s}E_b} = (\alpha\tau)F + (\alpha\varepsilon)F\frac{\sigma T_c^4}{C\,{}^{s}E_b} - (\epsilon\bar{\rho})F\frac{\sigma T^4}{C\,{}^{s}E_b} - U_L F\frac{(T-T_a)}{C\,{}^{s}E_b}, \tag{3.3}$$

where ${}^{s}E_b$ is the direct radiation, C is the concentration ratio, σ is the Stefan-Boltzmann constant, 5.67×10^{-8} W/m^{2}K^4, F is a fin factor which is close to 1 for a well designed receiver, T is the HTF temperature, T_c is the cover temperature, T_a is the ambient temperature, U_L is the heat loss coefficient, Q_s is the heat transferred to the HTF, and η_c is the collector efficiency. This equation explains the conversion of radiation into heat using four terms. The first term, $(\alpha\tau)F$,describes the absorption of radiation into the HCE, where $(\alpha\tau)$ is a complex function of the absorber and the cover material. The second term, $(\alpha\varepsilon)F\frac{\sigma T_c^4}{C\,{}^{s}E_b}$, describes the emission from the hot cover to the absorber. The third term, $(\epsilon\bar{\rho})F\frac{\sigma T^4}{C\,{}^{s}E_b}$, describes heat loss due to radiation through the aperture of the receiver. Finally the last term, $U_L F\frac{(T-T_a)}{C\,{}^{s}E_b}$, describes heat loss due to convection from the

receiver to the ambiance. In each of these four terms, the symbols in parenthesis [e.g. $(\epsilon\bar{\rho})$] represent functions of the HCE materials, where α is the absorptivity, ε is the emissivity, τ transmissivity, and ρ is the backscattering.

Figure 3.7 provides a graphical representation of collector efficiency as a function of HTF temperature for a range of collector types. In this chart, convective and conductive heat losses are neglected, and radiative losses are assumed to dominate. Note that parabolic troughs maintain a near constant efficiency until approximately 400°C.

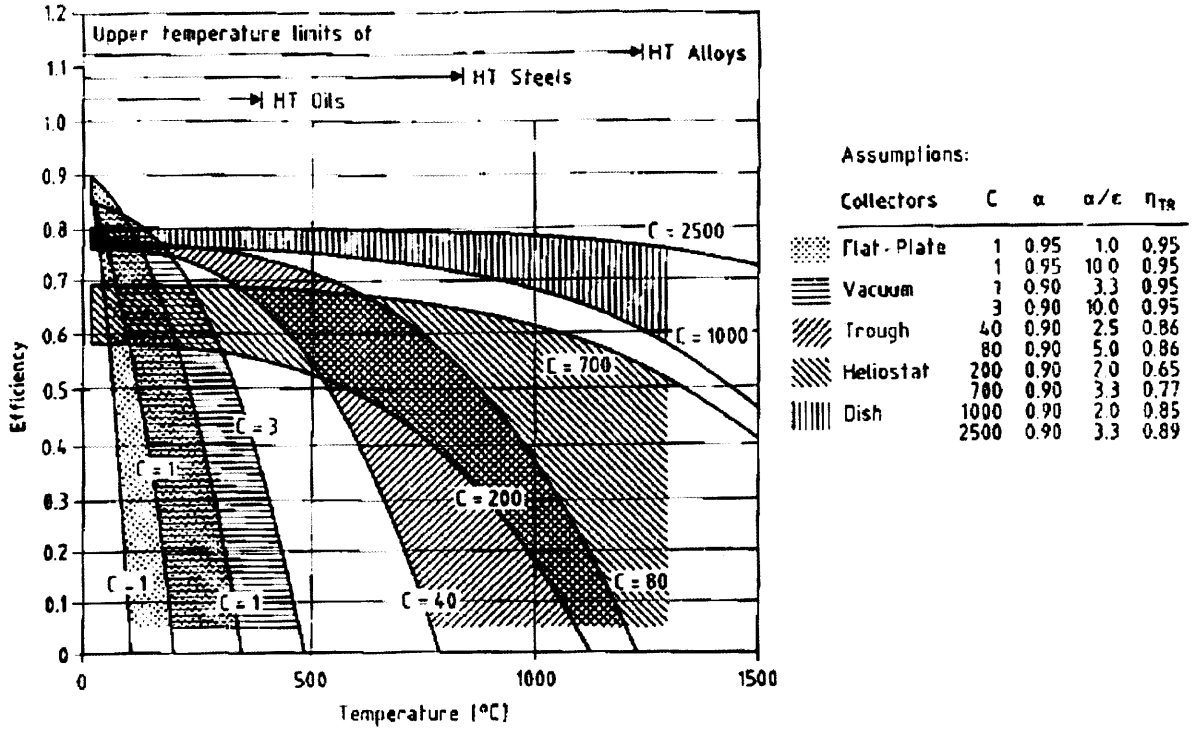


Figure 3.7 Efficiency of collector type vs. HTF temperature [57].

When calculating the average heat that reaches the HTF, it is easier to work with the local insolation value, given by the equation:

$$Q_s = IA\eta, \tag{3.4}$$

where Q_s is the heat added from the sun, I is the solar insolation, η is the collector efficiency, and A is the area.

The original Luz heat collection elements used in the SEGS plants is illustrated in **Figure 3.8**. This model has been improved upon by 2nd generation companies Schott and Solel, but the basic design remains the same. A stainless steel absorber tube is surrounded by an evacuated space and then a glass envelope. This design absorbs a lot of radiation and reduces convective heat losses. Every 4m are bellows that bind the steel-glass assembly together. There is an evacuation nozzle, which allows the vacuum to be maintained, and hydrogen getters are added to absorb hydrogen that

diffuses into the evacuated annulus. The stainless steel tube is coated with a selective absorber surface and the evacuated glass tube is coated with an antireflective (AR) coating. This increases the amount of solar radiation that reaches the HTF. In addition, if the receiver loses its vacuum and is exposed to air, an evaporable barium getter will change from silver to white.

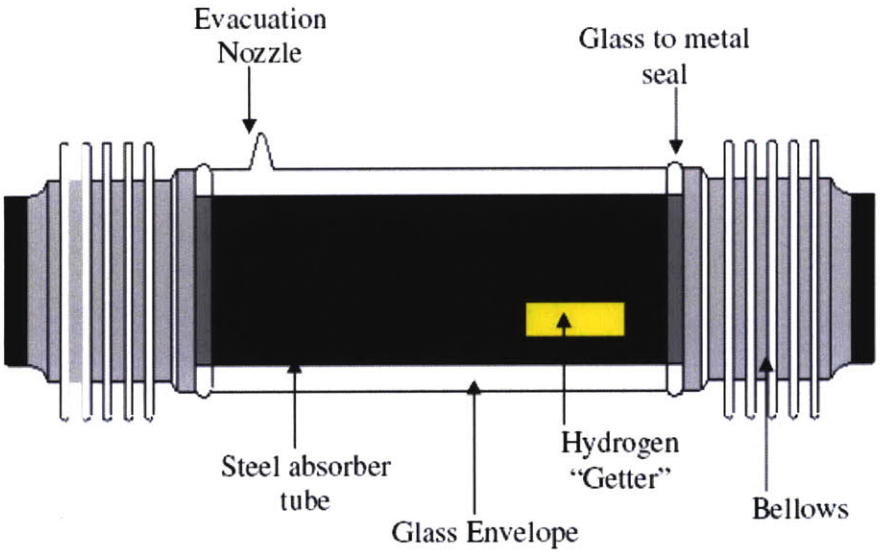


Figure 3.8 Illustration of a solar collector heat collection element [6].

Convection, conduction, and radiation are the modes of heat transfer in the HCE, with radiation dominating. However, because temperatures are so high in the HTF, one must consider every contribution of heat transfer to understand what losses can be predicted. **Figure 3.9** shows an illustration of the modes of heat transfer occurring. In later analysis, these heat losses will be re-introduced from an experimental point of view for accurate modeling.

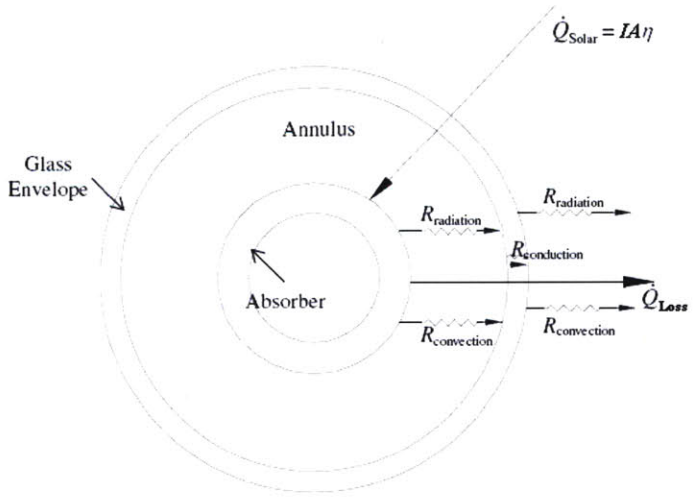


Figure 3.9 Heat transfer schematic of a heat collection element [6].

These HCE are expensive, so reliability is a big priority. Unfortunately, these highly precise components can fail in several ways and are currently being addressed by suppliers Solel and Schott. The biggest causes of failure are vacuum leaks due to a break in the glass to metal seal, and hydrogen permiating into the vacuum. Both of these failures significantly increase conductive heat losses and reduce HTF temperature in the receiver. The seal is hard to design because it must compensate for the different expansion of glass and metal as they are heated. Hydrogen is also a big problem because it comes from the decomposition of the HTF (therminol-VP1) at higher temperatures, which then permeates through the steel. Removing this hydrogen adequately requires better getter designs.

New receivers show an improvement in collector performance [44]. These receivers have improved on the design of seals and bellows, and have developed superior absorption coatings for the steel and AR coatings for the glass. The latest models, which have been thoroughly tested by NREL, are shown below. Schott's latest receiver is the PTR 70, and Solel now makes the UVAC 2008.

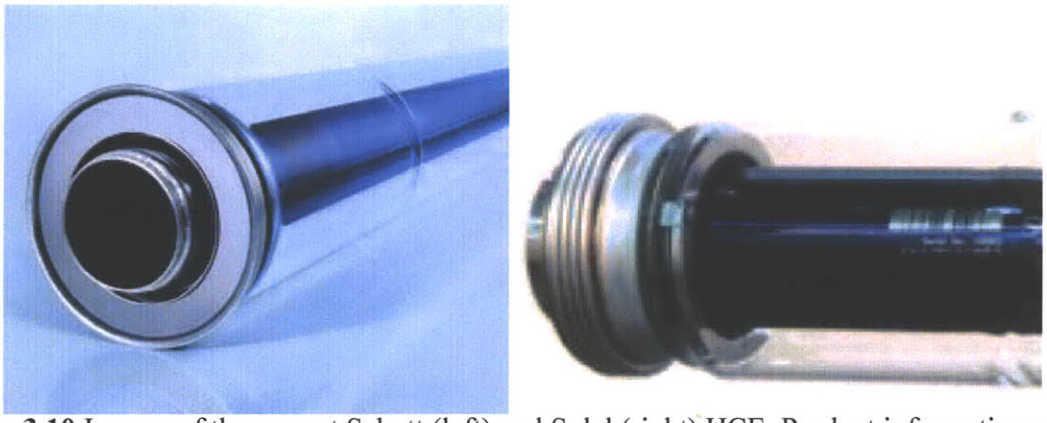


Figure 3.10 Images of the newest Schott (left) and Solel (right) HCE. Product information can be found on their website [47, 51].

3.2.2 Thermal Energy Storage Options

Thermal energy storage allows a solar plant to produce power after the sun sets and when electricity demand is high. Storage can increase a plant's capacity factor from 25% without thermal storage to up to 70% or more with it [17]. Storage may also be used to buffer the transient nature of solar radiation during the day as a result of changing sun angles, passing clouds, etc, thereby regulating the heat delivered to the power bloc. The challenge with thermal energy storage is that there is no standard practice for accomplishing this task. There are currently several design paths being researched and tested, with no clear leader at this time. These are:

- Two-tank direct with oil
- Two-tank Indirect
- Single-tank thermocline
- Two-Tank Direct with molten salt
- Solid thermal media
- Phase-change materials

For a thorough review of existing storage options and how they meet requirements of a solar thermal power plant, the reader is referred to Herrmann & Kearney, 2002. This thesis will model a two-tank direct mode of storage for thermodynamic purposes, but will consider the costs of two-tank indirect storage. This is because two-tank direct modes are easier to model and accomplish the same goal of energy storage, but two-tank indirect modes are more economically feasible.

Chapter 4

4. Scenario Plant A

4.1 Objective

The objective of scenario plant A was to develop innovative solar-geothermal hybrid energy conversion systems for existing low enthalpy geothermal power plants by augmenting them with solar energy. The goal was to find hybrid solutions that take advantage of the potential synergies of solar thermal and geothermal power cycles.

4.1.1 Approach

A single geothermal power plant was selected for evaluation based on data available from ENEL. This plant was modeled with the Aspen Plus platform and validated against real plant data to confirm accuracy. Accurate pure component property data and equation of state models were utilized for mass and energy balance calculations.

Next, several hybrid power cycle configurations were proposed and evaluated in which solar heating was optimally integrated with Plant A. These hybrid configurations utilized solar energy from parabolic troughs, as discussed in the previous section. The constraints on this hybrid system were the existing geothermal plant hardware and the specified land area for solar thermal collection.

A parametric steady-state study was completed to generate annual power generation profiles for the most promising hybrid configurations. The results of these studies were combined with actual insolation rates and ambient temperature measurements over a typical year at the location of the existing plant in order to estimate the proposed configurations' annualized electricity generation.

The study focused on determining the hybrid configuration that yielded the highest annualized electricity generation for the lowest cost. This was measured by the levelized cost of electricity (LCOE), which was estimated using equipment costs for the solar collectors.

4.2 Simulation Studies of Existing Plant

The first step in conducting this study was to develop a simulation in Aspen Plus that could model the operation of the existing binary power plant in its current configuration. As such, actual operating data was used as the basis for the model. For privacy reasons, this data will not be presented in this thesis.

The existing geothermal plant was modeled with three different bases:

- Original Design Basis: Original design mode of the plant (on paper);

- Current Operating Basis: Current operating mode of the plant. This is different from the original design due to equipment changes and changes in the geothermal resource temperature;
- Reference Case: Future operating mode with artesian well brine at 270°F (132.2°C). This model is used as the basis for the analysis of all the hybrid systems.

4.2.1 Model for Original Plant Design

The original geothermal plant is a dual cycle binary plant with isopentane as the binary working fluid. It has been operating since 1989. A process flow diagram of the original geothermal plant is shown in **Figure 4.1**. The flowsheet model shows a single unit. There are a total of seven parallel units, each with a Level 1 and Level 2 cycle. The implicit assumption in this model is that the seven parallel units receive an equal distribution of brine flow and that they all perform identically.

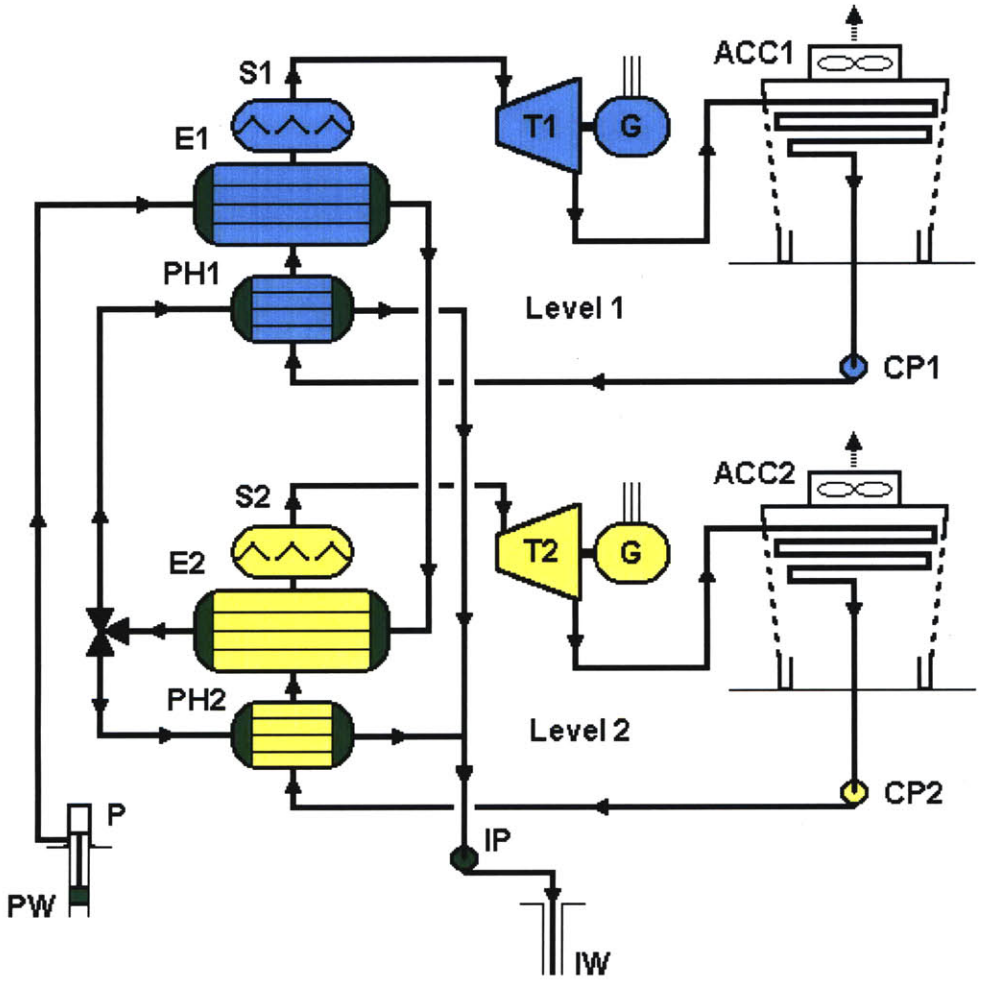


Figure 4.1 Original plant configuration of a single unit.

(Source: Ron DiPippo, personal communication)

Each level consists of a preheater, an evaporator, a turbine and generator, an air cooled condenser, and a recirculating pump. The Level 1 vaporizer (E1) receives the incoming hot brine and uses this higher quality heat to operate at a higher pressure than Level 2. The design basis pressure ratio for the Level 1 turbine is 11.1. The Level 1 turbine is a 2-stage Ormat impulse turbine.

The brine exiting the Level 1 vaporizer provides lower quality heat to the Level 2 vaporizer (E2). Therefore the Level 2 system operates at a substantially lower pressure with a turbine pressure ratio of 5.77 for the design basis. The Level 2 turbine is a 1-stage Ormat impulse turbine.

From the Level 2 evaporator, the brine splits to feed both the Level 1 and Level 2 preheaters before returning to the ground.

The temperature and pressure results for each stream in the design basis flow sheet model are provided in the **Figure 4.2** for Level 1 and **Figure 4.3** for Level 2.

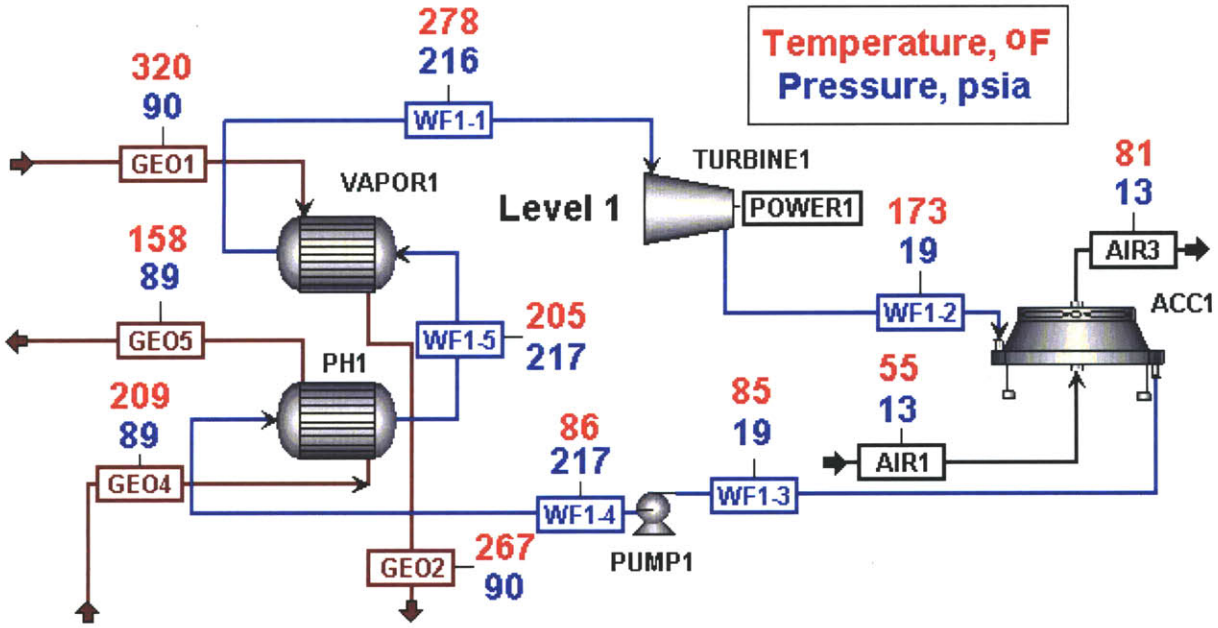


Figure 4.2 Temperature and pressure results for original geothermal plant Level 1.

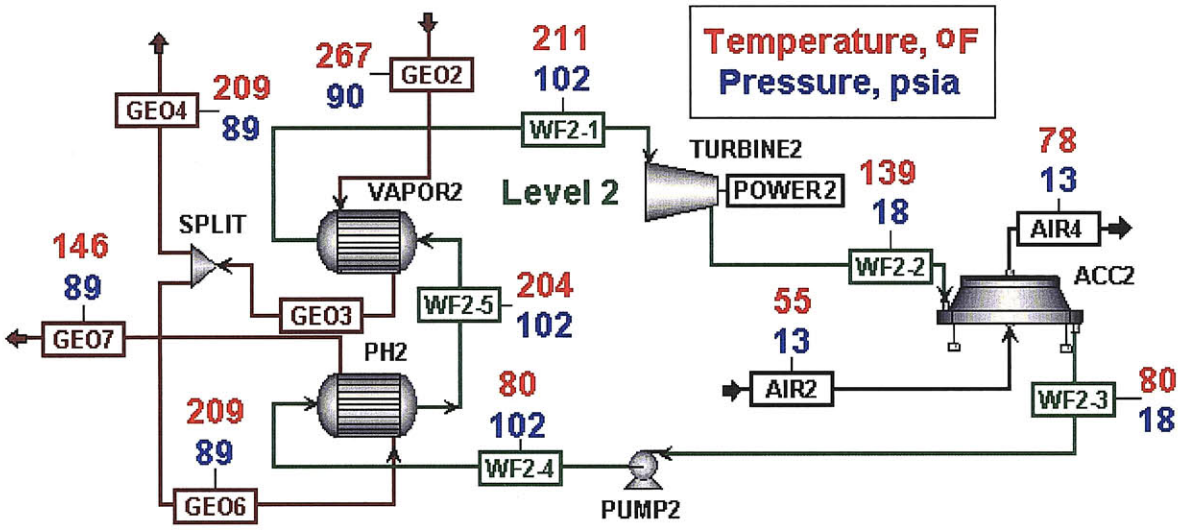


Figure 4.3 Temperature and pressure results for original geothermal plant Level 2.

4.2.2 Flowsheet Modeling Assumptions

Ideally, the representation of the equipment should be the same for all modes, however, there were specific exceptions that prevented the design mode equipment performance parameters from being used in modeling of the operating plant. For example, the design basis indicates turbine isentropic efficiencies of approximately 85%. This high turbine efficiency is not compatible with the turbine performance data from the operating plant. There was also a fundamental difference between the design basis and the operating plant in terms of vaporizer superheating of the working fluid. The design basis did not superheat the isopentane, but the plant operating strategy is to superheat by about 5°F (2.7°C).

For the models of the current geothermal plant operations and for the reference geothermal plant, **Table 4.1** gives all the assumptions that were made.

1. The ACC unit operates to condense the working fluid to saturated liquid.
2. The turbine exit pressure is determined based on the ability of the ACC to reject heat at the given ambient temperature. At lower ambient temperatures, the turbines may be choked, thereby limiting the flow rate through them.
3. The turbine exit pressure is not permitted to fall below 13.0 psia (0.896 bara) because plant operating procedures avoid creating a vacuum in the ACC as a precaution against air in-leakage. The 13 psia (0.896 bara) pressure was selected as the limit because 13.0 was the lowest pressure value found in the plant data set. The plant will turn off fans as needed to ensure that the 13 psia limit is not violated. The model represents this strategy as a continuum by reducing the effective UA of the ACC unit as needed to satisfy all the constraints.
4. The mass flow rate of isopentane is determined by the model to achieve 5°F (2.7°C) of superheat for the working fluid leaving the vaporizers.

Table 4.1 Aspen Plus simulation assumptions.

Equipment Item	Model Parameter[1]	Parmeter value	Basis
Turbine Level 1	Isentropic Efficiency	79.5%	Average from plant performance data
Turbine Level 2	Isentropic Efficiency	74.2%	Average from plant performance data
Generators	Efficiency	95%	Assumed value
Gear box	Efficiency	98%	Assumed value
ACC Level 1	UA value	342,893 W/K	Determined from plant performance data
ACC Level 2	UA value	369,270 W/K	Determined from plant performance data
ACC Level 1	Pressure drop	1.35 psi (0.093 bar)	Calculated with Aspen Acol+ software
ACC Level 2	Pressure drop	1.10 psi (0.0758 bar)	Calculated with Aspen Acol+ software
Vaporizer Level 1	UA value	430,059 W/K	Determined from design basis
Vaporizer Level 2	UA value	474229 W/K	Determined from design basis
Vaporizer Level 1	Pressure drop	0.4 psi (0.028 bar)	Design basis
Vaporizer Level 2	Pressure drop	0.3 psi (0.021 bar)	Design basis

Based on operating data from the existing plant, the flow through the turbines was found to be sonically choked. The mass flow rate is proportional to the absolute pressure and inversely proportional to the square root of the absolute temperature of the i-C5 entering the turbine, i.e.

$$\dot{m}_{iC5} = K\frac{P_2}{\sqrt{T_2}}\,, \tag{4.1}$$

where K is a collection of flow constants [41]. Equation (4.1) governing the choked mass flow is scientifically validated for a perfect gas, and it is assumed that the isopentane behaves sufficiently like a perfect gas (this is because isopentane is superheated before entering the turbine, thereby approaching a perfect gas) to allow the use of this equation. Furthermore, the equation was calibrated using the given design data to determine the flow constants which should overcome any problem with the perfect gas assumption. In general, the coefficient K in eq. (4.1) depends on the

[1] Model parameters are defined as follows: Isentropic Efficiency $\eta_{i,e} = \frac{W_{actual}}{W_{isentropic}}$, where the actual turbine work is less than the work of an isentropic expansion. Similarly, the efficiency of the generators and gear box reduces the actual work that is converted from the turbine rotation into electricity. UA value [W/K] is equal to U*A, where U is the overall heat transfer coefficient of the heat exchanger [W/m^2-K], and A is the heat transfer area [m^2]. Press drop defines the drop in pressure across the length of a heat exchanger. In these assumptions, only pressure drop in the organic working fluid is considered.

turbine geometry (discharge coefficient and discharge hole cross-sectional area), and the perfect gas properties of the working fluid, including the isentropic index (i.e., the ratio of the specific heat at constant pressure to the specific heat at constant volume), the ideal gas constant, and the gas compressibility factor of a real gas at P and T. However, lacking sufficient information regarding the turbine geometry, a method of calibrating the equation was adapted to the design data provided by Ormat, the original plant designers. From the design specifications provided, the pressure and temperature of the i-C5 at the turbine inlet were given. The design mass flow rate was found using a first law energy balance on the vaporizer using the design properties for brine and i-C5 together with a thermodynamic database. Thus the following values were obtained for the design flow rates of i-C5:

- 146,314 lbm/h (18.44 kg/s) (Level 1)
- 172,846 lbm/h (21.78 kg/s) (Level 2).

Then eq. (4.1) was used to calculate the coefficient K for the two levels, giving the following results:

$$\dot{m}_{iC5} = 18{,}412\frac{P_2}{\sqrt{T_2}} \qquad \text{(Level 1)} \qquad (4.2)$$

$$\dot{m}_{iC5} = 43{,}874\frac{P_2}{\sqrt{T_2}} \qquad \text{(Level 2).} \qquad (4.3)$$

The units in these equations are automatically correct; i.e., when the pressure P is in psia, the mass flow rate is in lbm/h; the absolute temperature T in degrees Rankine is the temperature in degrees Fahrenheit plus 459.67.

For the solar-geothermal hybrid configurations, the vaporizer pressures are expected to deviate substantially from the typical pressures measured in the plant during past operations. Therefore to determine the pressure exiting the vaporizers, the turbine choke flow equations 4.2 and 4.3 were implemented in the hybrid plant models to ensure that the flow-temperature-pressure relationship is always maintained.

The *P-V-T* (volumetric) thermodynamic properties for isopentane were based on the Benedict-Webb-Rubin-Starling (BWRS) equation of state (EOS); the primary source for the EOS parameters is the Design Institute for Physical Properties (DIPPR). The Steam Tables were used for water properties.

The BWRS property method is comparable to Peng-Robinson and Redlich-Kwong-Soave for phase equilibrium calculations, but is more accurate for liquid molar volume and enthalpy. It is suited for reduced temperatures (T/T_{cr}) as low as 0.3 and reduced densities (ρ/ρ_{cr}) as great as 3.0.

Based on the elevation of the plant, the ambient air pressure was set to 12.75 psia (0.879 bar) for all flowsheet simulations.

As an assist in visualizing the existing plant, it is useful to see the plant in its original configuration (see **Figure 4.4**) and in its current configuration (see **Figure 4.5**). A single brine header feeds all

seven units and a single brine return header collects the cooled brine from all units (see **Figure 4.6**). There is little open area in the immediate vicinity of the units, requiring that any equipment needed to transform it into a hybrid solar-geothermal plant will need to be sited either on the adjacent parcel of land or in the area between the plant and that field.

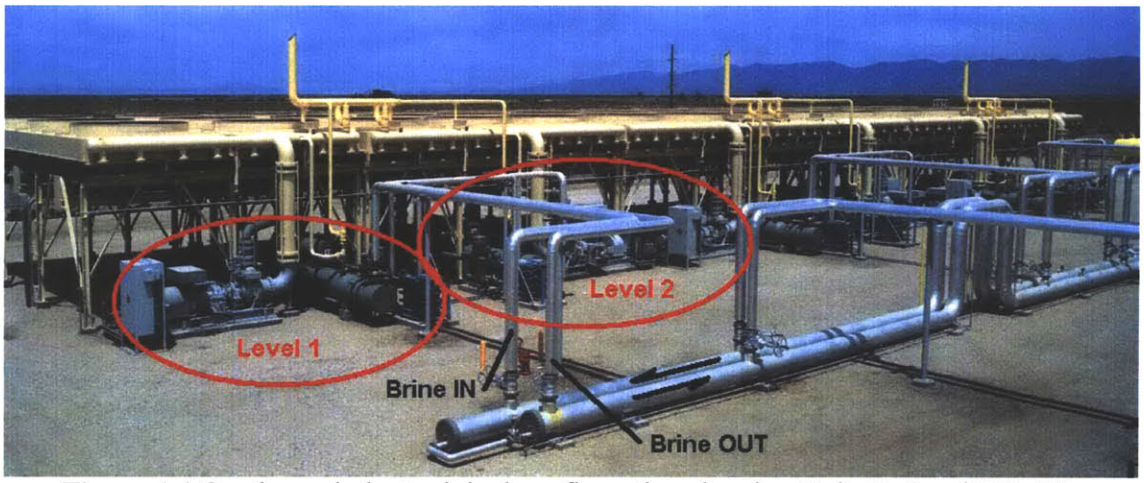


Figure 4.4 Geothermal plant original configuration showing Units 7, 6 and 5 (L-R).

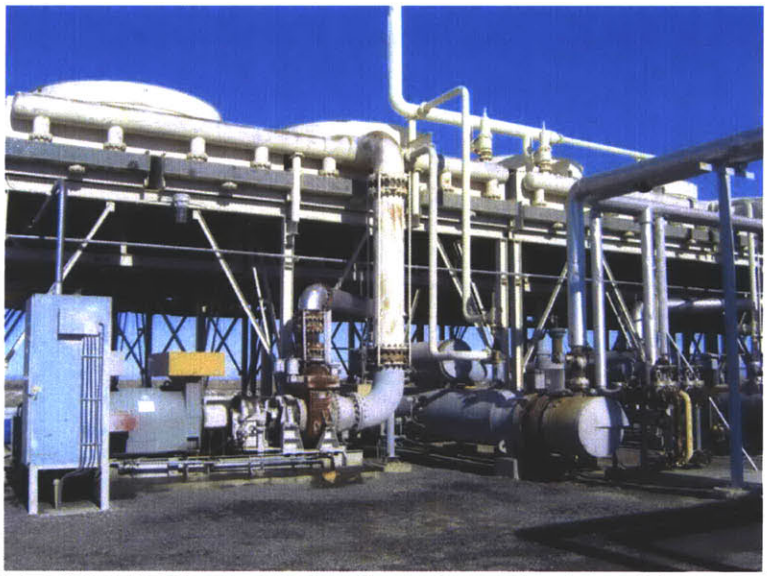


Figure 4.5 Level 1 of Unit 7. Note vertical yellow pipe to the right of the vaporizer; this bypass replaced the preheater. [Photos by R. DiPippo, 12/16/2008.]

In the current arrangement, the brine is divided among the seven units as evenly as possible (although without control). For each unit, the brine passes in order through the vaporizer of the Level 1 cycle, the vaporizer of the Level 2 cycle, and then enters the return header in preparation for reinjection. In the original configuration, the brine passed through two preheaters in parallel for each level before entering the return header. Since the preheaters have been removed, the actual operation is shown in the simplified schematic flow diagram, **Figure 4.6**.

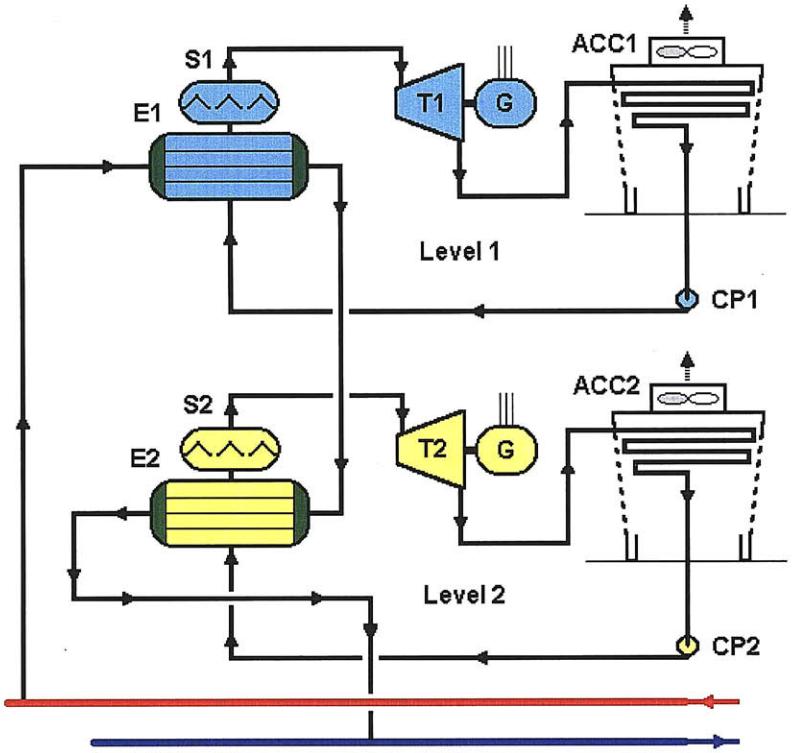


Figure 4.6 Schematic flow diagram for current unit operation of geothermal plant.

(Source: Ron DiPippo, personal communication)

4.2.3 Model for Current Plant Configuration

The model of the original plant configuration was modified to remove the preheaters. The preheaters were titanium plate-and-frame exchangers. These exchangers were problematic with leaking seals and very high maintenance costs. Therefore these preheaters have been removed from operation. The vaporizers now function as both preheaters and boilers. The vaporizers have not been modified for this new service. The lack of preheaters creates a problem in trying to model the vaporizers since they now must preheat the isopentane as well as boil it.

The resulting MIT flowsheet model of the existing plant is shown in **Figure 4.7**. Two closed isopentane power cycles are served by geofluid (brine) that passes through vaporizers first in Level 1 (upper section) and then in Level 2 (lower section).

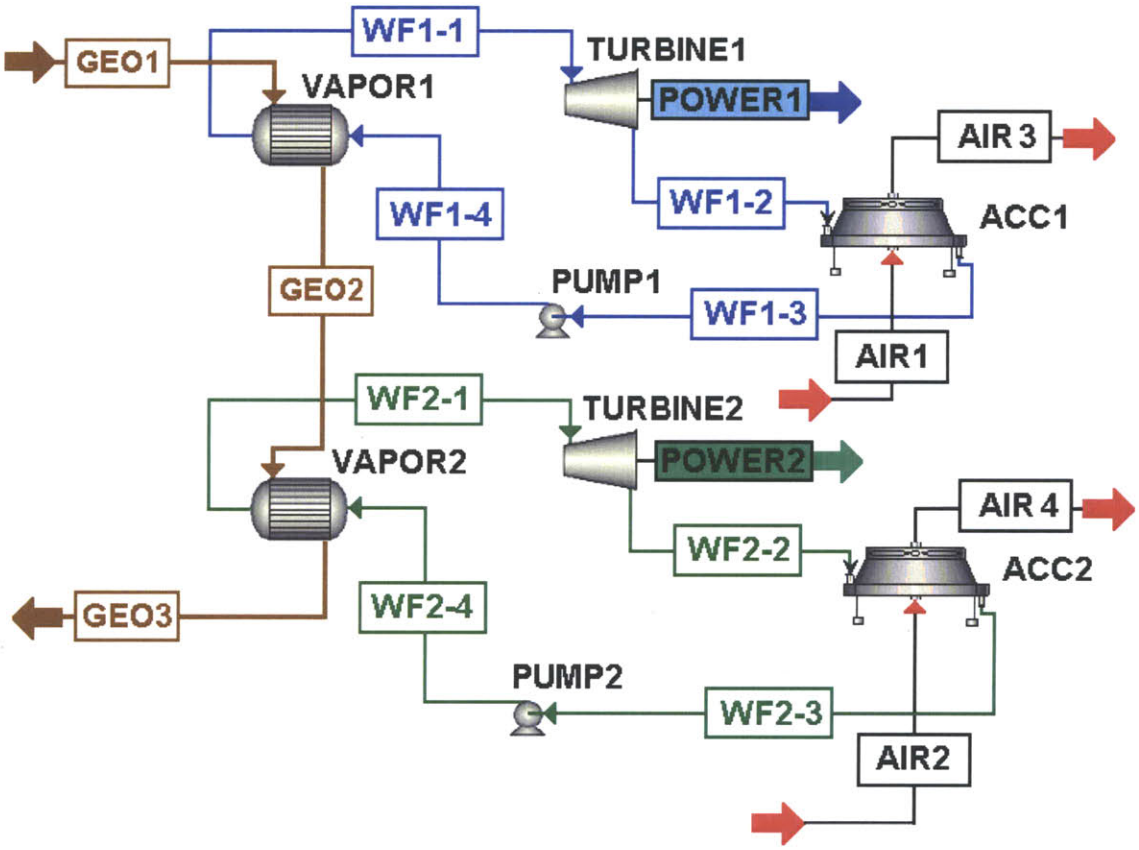


Figure 4.7 Modified configuration of a single unit – Aspen Plus model.

The model is based on simultaneously solving a set of equations and conditions that describe the physical operation of the plant:

(1) Heat balance (First Law of thermodynamics) on the vaporizer for each level to ensure that the model responds to changes in brine flow;
(2) Sizing of the vaporizers to ensure that the model responds to changes in the brine temperature;
(3) Choked flow equation for the turbines to satisfy the relationship among turbine inlet pressure, inlet temperature and mass flow rate to represent the physical limits of the turbine nozzle; and
(4) Maintenance of 5°F (2.7°C) of superheat at the vaporizer outlets to ensure the temperature and pressure are consistent with the operating specifications.

Condition (1) is equivalent to using:

$$\dot{m}_{iC5}\left(h_{2,iC5} - h_{1,iC5}\right) = \dot{m}_b\left(h_{in,b} - h_{out,b}\right). \tag{4.4}$$

Condition (2) requires satisfying the following equation:

$$\dot{Q} = UA\ x\ LMTD\ . \tag{4.5}$$

where U is the overall heat transfer coefficient for the vaporizer, A is the total area available for heat transfer in the vaporizer, and $LMTD$ is the cumulative zonal analysis log-mean temperature difference between the brine and the i-C5.

Condition (3) requires satisfying the choked flow equations given earlier:

$$\dot{m}_{iC5} = 18{,}412\frac{P_2}{\sqrt{T_2}} \qquad \text{(Level 1)} \tag{4.2}$$

$$\dot{m}_{iC5} = 43{,}874\frac{P_2}{\sqrt{T_2}} \qquad \text{(Level 2).} \tag{4.3}$$

It must be emphasized that the model uses a lumped parameter approach for the behavior of the vaporizers that now perform double duty as well as condition of choked flow through both turbines in each unit. The efficacy of these somewhat heuristic assumptions is borne out in the closeness of the model's performance compared to the actual plant performance.

The absence of purpose-built preheaters and the reduced temperature of the feed brine have substantially reduced the power generation capacity of the current plant. The MIT model of the original flow sheet was modified to eliminate the preheaters and then validated against the data collected from the plant taken in September 2007, December 2007, March 2008 and June 2008, spanning a very wide range of ambient conditions. The ambient temperature covered the range from 25 to 85°F (-3.89 to 29.4°C) while the brine flow rate varied over a range of 4540-5980 GPM (286.4-377.2 kg/s). To ensure consistency and completeness of the data set, only data for days when all equipment was operating were used for this analysis. This resulted in a total of 36 data points available for this analysis.

As expected, the most important independent variable for this model is the ambient air temperature. This is because the ambient temperature has a substantial impact on the turbine pressure ratio and the ambient temperature has a large natural seasonal variability. By comparison, the brine flow rate was not very interesting. The brine flow for the available plant data set was relatively consistent and therefore brine flow was set to the average flow rate of 794.3 GPM (50.1 kg/s).

The brine feed temperatures ranged from 291.1 to 299.1°F (143.94 to 148.39°C) for the plant data set. This temperature variation is important because this temperature is relatively close to the boiling temperature of 259°F (126.1°C) for vaporizer 1. On the coldest recorded days (December 2007 and March 2008), the brine temperature is always high, averaging 298°F (147.78°C). The brine temperature was generally lower on the warmer days in September 2007 and in June 2008.

To gain confidence in the Aspen Plus model for the existing plant, many simulation runs were carried out; **Figure 4.8** shows the results of the simulation. The gross power (in MW) from the entire 7-unit plant is plotted against the ambient temperature, which is a critical parameter affecting the performance of any air-cooled binary plant. The model follows the actual

performance quite well, capturing the flattening of the performance curve at very low ambient temperatures. At temperatures below 45°F (7.2°C), the power generation starts to level off. This is because the turbine outlet pressures are limited by the 13 psia (0.896 bara) lower limit (ambient air pressure).

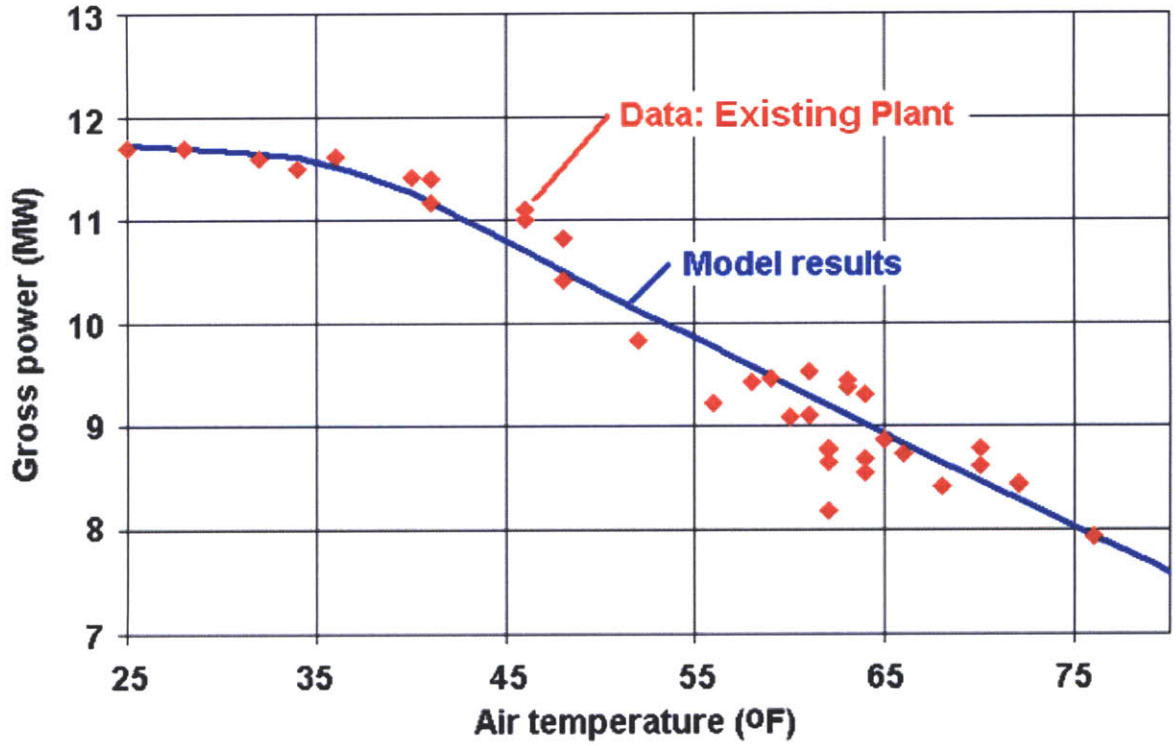


Figure 4.8 Gross plant power output vs. ambient temperature: model vs. plant data.

4.2.4 Model for Reference Geothermal Plant

The Reference Geothermal plant was modeled using the same simultaneous set of equations as described in **Section 4.2.3**. The only differences are that the brine specifications were set to 270°F (132.2°C) and 750 GPM (47.3 kg/s) per unit to reflect that only low-temperature and low-flow artesian well brine will be available moving forward. The brine temperature equation was not used in the reference geothermal plant model. The reference plant model was used as the basis for all the hybrid analysis. The results of this reference plant model are presented in the context of the hybrid systems in **Section 4.4** below.

4.3 Conceptual Hybrid Designs and Metrics for Comparing Options

This section presents and discusses five hybrid solar-geothermal systems that were studied. The main constraint on the study was that the existing plant had to be incorporated into the hybrid arrangement essentially in its current configuration. Furthermore, use could be made of a portion of the roughly 40-acre (161874 m^2) site immediately adjacent to the plant on its west side for the field of solar collectors.

The five conceptual designs are called:

1. Isopentane superheat concept
2. Brine preheat concept
3. Brine recirculation concept
4. Brine preheat-recirculation concept
5. Brine reheat concept.

4.3.1 Isopentane Superheat Concept

One possible approach to integrating solar energy into the existing plant is to use the solar collector to raise the temperature of the isopentane before it enters the turbines, thereby increasing the working fluid exergy and making it possible to increase the power from the turbines while reducing the i-C5 flow rate. If this can be accomplished, a reduction in the parasitic pumping power would result. This is particularly attractive on summer days when net power from the plant is critically limited and parasitic power grows to a large fraction of the gross power. The concept is shown in **Figure 4.9**.

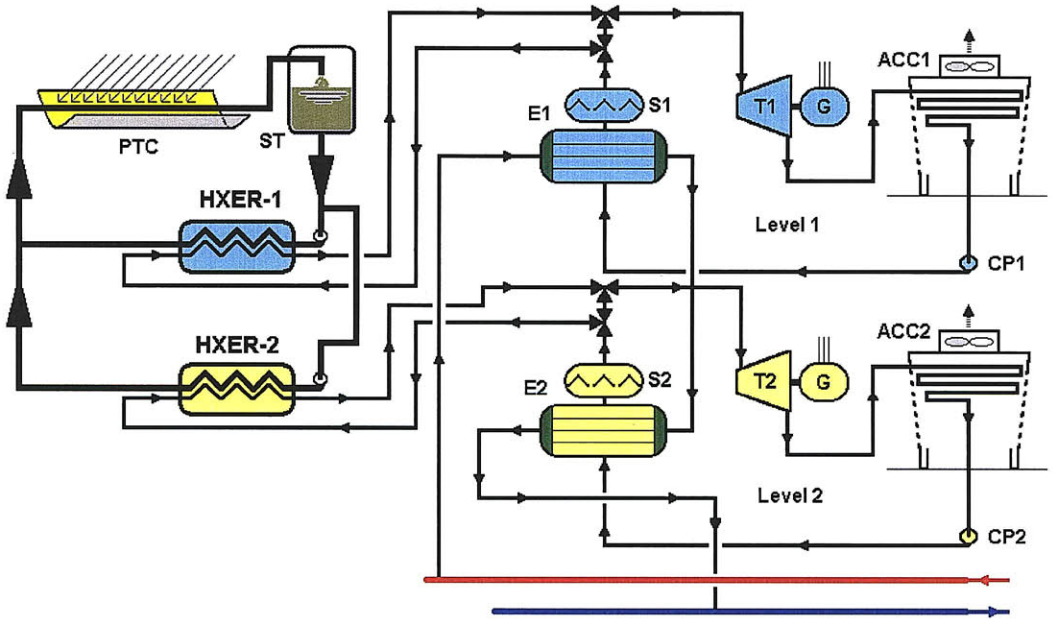


Figure 4.9 Solar-superheated isopentane concept.

(Source: Ron DiPippo, personal communication)

The figure depicts superheating for both turbines in a given unit, but the concept can be demonstrated by applying the concept to just the Level 1 turbine in, say, Unit 7, the closest to the solar field. Several 3-way valves and new piping will need to be added to the existing plant equipment to accommodate the solar heat when it is available. Naturally, the entire solar collector system is new.

The thermodynamic effect of increasing the superheat of the i-C5 is shown in **Figure 4.10**.

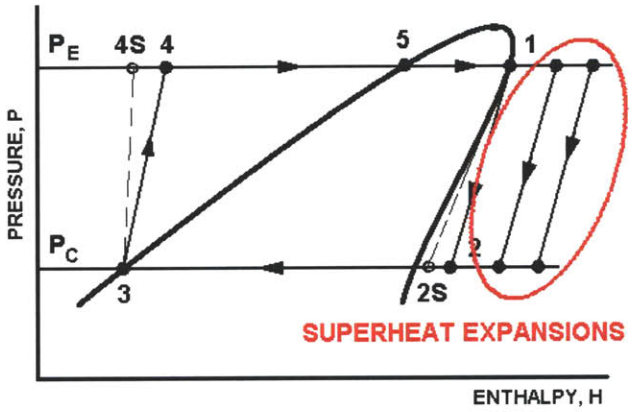


Figure 4.10 Pressure-enthalpy process diagram showing superheat at turbine inlet.

(Source: Ron DiPippo, personal communication)

The process line from 1-2 is the expansion from a saturated (or slightly superheated) turbine inlet state, essentially the current situation. The two additional expansion lines depict two possible expansions when the inlet state is significantly superheated. The inlet and outlet pressure levels will remain essentially the same, but slightly lower due to pressure losses in the new piping and heat exchangers. This pressure loss could be compensated by additional boost from the feed pumps.

4.3.2 Brine Preheat Concept

In the brine preheat system, the geofluid is boosted from its current wellhead temperature of 270°F (132.2°C) to the design temperature of 320°F (160°C) by means of the solar heat input. **Figure 4.11** shows how this takes place.

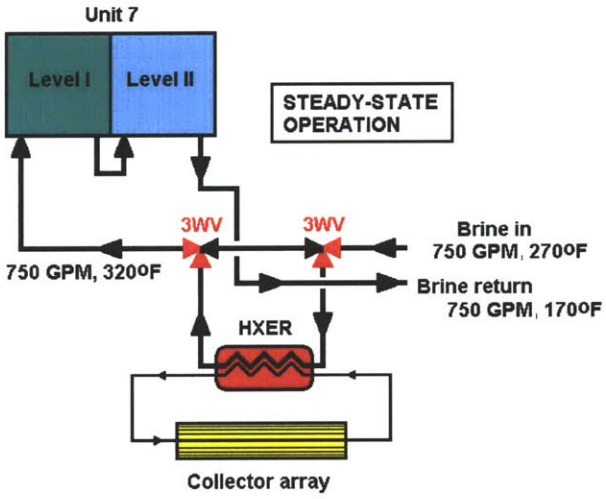


Figure 4.11 Brine preheat solar-geothermal hybrid concept.

(Source: Ron DiPippo, personal communication)

The heat transfer fluid (HTF) circulating between the concentrating parabolic trough collectors and the solar-geo heat exchanger (HXER) imparts thermal energy to the brine before it enters the vaporizer of Level 1 of Unit 7. Unit 7 is chosen in this example because it is closest to the proposed solar collector field than the other six units. The flow rate of 750 GPM (47.3 kg/s) is what is expected to be available from the artesian wells (one-half of the total flow of 1500 GPM) (94.6 kg/s). After the brine passes through both levels, it is sent to reinjection.

The objective of this arrangement is to restore the unit to its design inlet brine conditions, namely, 320°F (160°C) and 846 GPM (53.4 kg/s). While the temperature can be achieved, the flow rate is still about 100 GPM (6.31 kg/s) below design.

4.3.3 Brine Recirculation Concept

The objective of the brine recirculation system is to boost the brine temperature to 320°F (160°C) while increasing the flow rate through the units by about 18% relative to design [1000 GPM (63.09) vs. 846 GPM (53.37 kg/s) design]. This is an attempt to overcome the poor vaporizer performance in the absence of the preheaters. **Figure 4.12** shows the flow diagram.

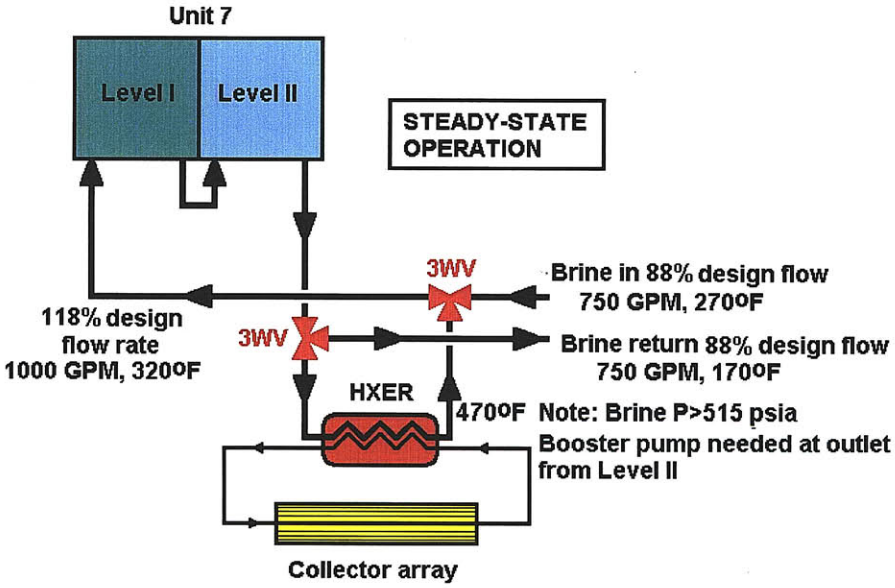


Figure 4.12 Brine recirculation solar-geothermal hybrid concept.

(Source: Ron DiPippo, personal communication)

Steady-state operation is assumed in the diagram. Initially, the brine flows at 750 GPM (47.31 kg/s) and 270°F (132.2°C) into the unit. When is leaves the unit, it is directed to the solar-geo heat exchanger (HXER) where is heats up to 470°F (243.3°C). It is kept under sufficient pressure to prevent boiling. The hot brine is then mixed with fresh incoming brine to form an inlet stream of 1000 GPM (63.09 kg/s) and 320°F (160°C). Once steady conditions are achieved, 250 GPM (15.77 kg/s) of brine will be passing through the HXER and 750 GPM (47.31 kg/s) of brine leaving the

unit will be sent to reinjection. It will be necessary to control the 3-way valves during start-up until the desired conditions are met.

4.3.4 Brine Preheat-Recirculation Concept

The two previous conceptual designs can be combined as seen in **Figure 4.13** in what is called the brine preheat-recirculation system. Here the brine is raised to 320°F (160°C) as it approaches to the unit. There is less heat duty on the HXER in this case (relative to the pure recirculation system) since the leaving brine is raised only to 320°F (160°C) in the HXER so that it mixes isothermally with the preheated brine, thus avoiding exergy loss due from dissimilar temperature mixing. The flow rates are the same as in the recirculation case.

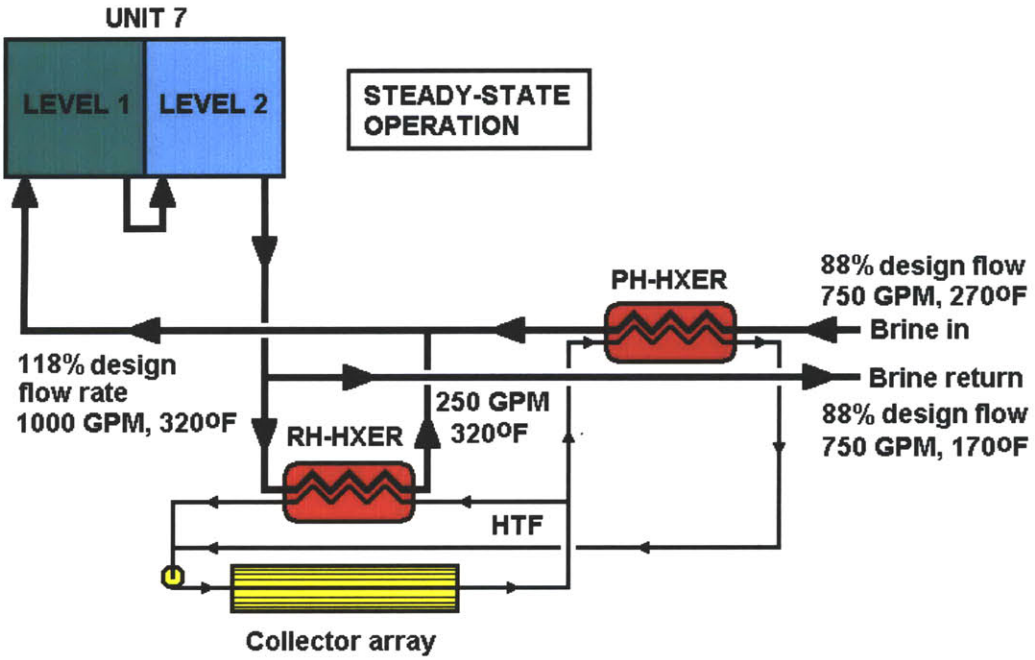


Figure 4.13 Brine preheat-recirculation solar-geothermal hybrid concept.

(Source: Ron DiPippo, personal communication)

4.3.5 Brine Cascade Reheat Concept

Another way to incorporate solar energy into the existing plant is the brine reheat concept is shown in **Figure 4.14**. Since the solar array will have the capability of restoring the brine temperature to its original value (or higher) after the brine serves both levels of any particular unit, the brine can be diverted before it enters the brine return header to the solar-geo heat exchanger HXER. The brine temperature can either be restored to 270°F (132.2°C) or boosted to the design value of 320°F (160°C) for the second unit. The second unit would generate an amount of power somewhat higher than that of the first unit. Since Units 7 and 6 are the closest to the available land that would host the solar collectors, these two units are shown as likely candidates for this hybrid approach in **Figure 4.14**.

Under this scheme the benefit of the solar input would simply be the output from the second unit that would otherwise have required a mass flow of brine equal to that feeding the first unit. In effect, two units could be run with the same brine flow rate (47.31 kg/s) that is now used for only one unit. This concept could be repeated in a cascaded sequence as many times as feasible for the rest of the existing units, provided a large enough collector array could be built.

This concept, like the previous ones, does not require a very high temperature solar collector. Thus for this demonstration, flat plate hot water solar collectors might serve as preheaters in the solar array, with concentrating parabolic troughs providing only the higher temperature finishing touch. This would be potentially less expensive than gathering the solar heat solely by parabolic troughs.

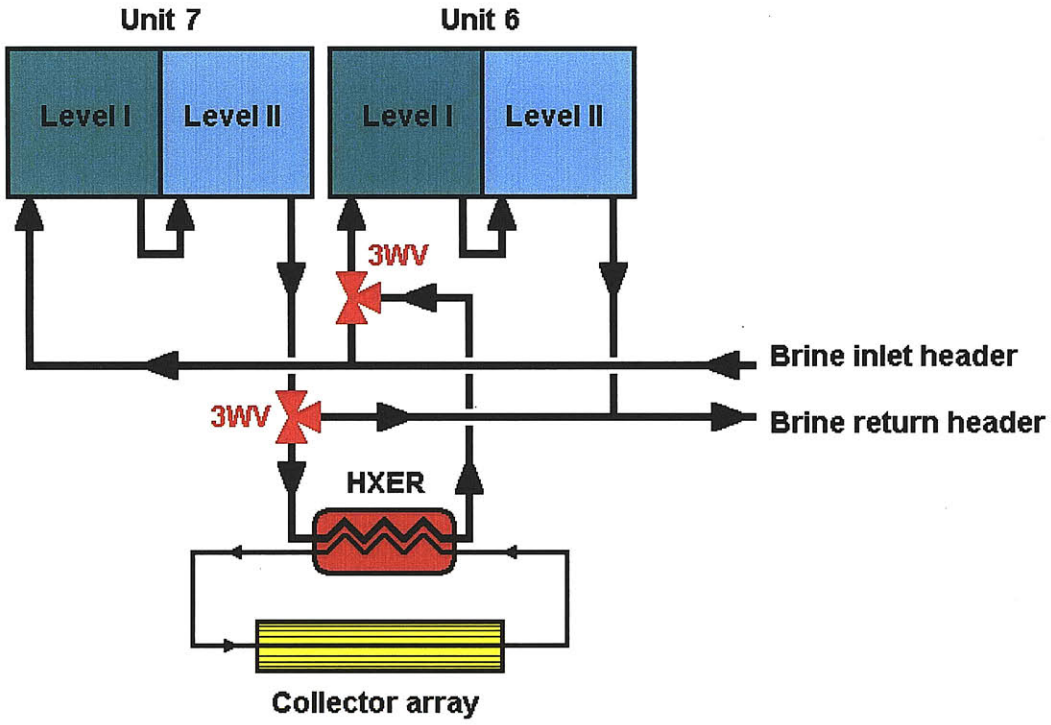


Figure 4.14 Brine reheat solar-geothermal hybrid concept.

(Source: Ron DiPippo, personal communication)

4.3.6 Solar Collector Assumptions

The solar field parameters were used in modeling the size, cost, and operating performance of the hybrid systems. The following assumptions were used:

- The collectors are conventional parabolic trough.
- Collector efficiency: 70%. Note that the general equation for collector efficiency is given by Winter et. al. [57] (3.3). 70% is the predicted collector efficiency for a parabolic trough with a HTF at 400°C when including radiative and convective losses.
- The cost per unit area of collector: $\$250/m^2$ for the base case economic analysis.
- The solar collector area is calculated using (3.4)

4.3.7 Metrics for Comparing Hybrid Designs

The basis for selecting one hybrid design over another will involve a combination of thermodynamic performance improvement coupled with the cost of the new equipment to accomplish the design. Both the gain in power output when the sun is shining and the extra electrical energy generated are important performance measures. A practical aspect in the selection process concerns the need to keep the geothermal plant in continuous operation, both under hybrid and stand-alone conditions. It is impractical to shut in the wells serving the plant or to shut down a unit when the sun is not shining.

The best criterion for comparing the thermodynamic performance is the utilization efficiency, η_u. This factor is based on the Second Law of thermodynamics and can be used for any power generating system, whether it operates cyclically or as a sequence of processes. It uses the exergy (or available work) as the basis for the performance assessment.

In its general form as applied to a simple system operating under steady, open conditions, the rate of exergy of a fluid at a given state 1 is given by:

$$\dot{E}_1 = \dot{m}[h_1 - h_0 - T_0(s_1 - s_0)]\,, \tag{4.6}$$

where $\dot{m}$ is the mass flow rate, h is the enthalpy, s is the entropy, and T_0 is the dead-state temperature. This equation may be applied directly in this form to calculate the exergy of the brine as it enters the power plant.

The exergy of the solar energy input is not as direct. Ideally one would like to calculate the exergy associated with the electromagnetic solar radiation but this is a controversial subject on which there is no general agreement. One can, however, base his calculations on the exergy of the heat transfer fluid (HTF) that circulates through the solar collectors, using the value of exergy that the HTF possesses when it leaves the solar field and enters the solar-geo heat exchanger, shown in the figures as HXER. This can be found from the following equation:

$$\dot{E}_{HTF} = \dot{m}_{HTF} c_{p,HTF}[T_{1,HTF} - T_0 - T_0 \ln \tfrac{T_{1,HTF}}{T_0}]\,. \tag{4.7}$$

This assumes a constant specific heat at constant pressure for the HTF.

One may also consider the change in the exergy of the HTF as it passes through the HXER. This is the rate of exergy that the HTF releases when in thermal contact with either the brine (reheat concept) or the i-C5 (superheat concept) and is given by:

$$\Delta\dot{E}_{HTF} = \dot{m}_{HTF} c_{p,HTF}[T_{1,HTF} - T_{2,HTF} - T_0 \ln \tfrac{T_{1,HTF}}{T_{2,HTF}}]\,. \tag{4.8}$$

Since exergy is not a conserved quantity, a fraction of the exergy released by the HTF will be destroyed by various irreversibilities within the HXER.

There are several possible definitions of utilization efficiency that are appropriate for our task. The first is the one for a basic geothermal power plant, namely:

$$\eta_{u,GEO} = \frac{\dot{W}_{net}}{\dot{E}_{GEO}}. \tag{4.9}$$

For a hybrid plant in which there are two distinct forms of exergy input, the definition that works best is:

$$\eta_{u,HYB} = \frac{\dot{W}_{net,HYB}}{\dot{E}_{GEO}+\dot{E}_{HTF}}. \tag{4.10}$$

In choosing between alternative hybrid systems, it is often useful to consider incremental utilization efficiencies. Where a solar system is added to an existing geothermal plant, it is logical to examine the incremental benefit of the additional solar input. Thus, the incremental improvement in performance due to the solar input should be determined and compared to the amount to the exergy of the solar input. The resulting equation is:

$$\eta_{u,i-solar} = \frac{\dot{W}_{net,HYB}-\dot{W}_{net,GEO}}{\dot{E}_{HTF}}. \tag{4.11}$$

The numerator represents the net power that can be ascribed to the solar input.

Another possible measure of performance is the ratio of the incremental output from the solar to the change in exergy of the HTF. This shows how much of the exergy released by the HTF shows up in the output of the hybrid plant. The equation is:

$$\eta_{u,i-solar} = \frac{\dot{W}_{net,HYB}-\dot{W}_{net,GEO}}{\Delta\dot{E}_{HTF}}. \tag{4.12}$$

One variation on eq. (4.12) is the global exergetic efficiency, or the ratio of the incremental output from the solar to the change in exergy of the HTF plus the change in exergy of the brine. The equation is:

$$\eta_{u,HYB} = \frac{\dot{W}_{net,HYB}-\dot{W}_{net,GEO}}{\Delta\dot{E}_{HTF}+\Delta\dot{E}_{GEO}}. \tag{4.13}$$

Another variation on eq. (4.12) is the functional geothermal utilization efficiency, calculated as the ratio of net work from the hybrid plant to the change in exergy of the brine. This shows how much exergy released by the brine shows up in the output of the hybrid plant. The equation is:

$$\eta_{u,GEO} = \frac{\dot{W}_{net,HYB}}{\Delta\dot{E}_{GEO}}. \tag{4.14}$$

The last of the efficiency equations is the familiar thermal efficiency:

$$\eta_{th,HYB} = \frac{\dot{W}_{net,HYB}}{\dot{Q}_{WF}} . \quad (4.15)$$

In this case the denominator is the heat delivered to the cycle working fluid.

These measures are combined with the cost of the additional systems to arrive at a thermodynamic-economic optimum design. Thus, the capital expense (CapEx) needed to generate a kilowatt of rated or installed power can be used as a criterion that combines thermodynamic performance and economics. Also, the levelized cost to generate a kilowatt-hour of electricity is an equally, if not more important economic factor.

4.4 Simulation Studies of Solar-Geothermal Hybrid Designs

The results of the performance analyses for the candidate solar-geothermal hybrid conceptual designs are presented in this section. One system, the isopentane superheat system, turned out to be a non-viable candidate; it is treated first in **Section 4.4.1**. The results of the design study for the other four systems are presented together in **Section 4.4.2**. Based on the design results, one system was eliminated from further study, and **Section 4.4.3** presents the operating results for the three viable candidate systems.

4.4.1 Superheat Hybrid Cycle Results

The MIT proposed isopentane superheat configuration as described in **Section 4.3.1** was modeled in Aspen Plus. In this configuration the solar heat is used to provide superheat to the working fluid between the vaporizer and the turbine. Because the solar energy is providing the superheating, the vaporizer operating strategy was changed to generate saturated vapor. This strategy allows the vaporizer to maximize performance without consuming any brine heat for superheating. The MIT model of this flow sheet was shown in **Figure 4.3.4**.

When the working fluid is superheated by the solar heat transfer fluid, the working fluid temperature, pressure and mass flow must remain in balance in accordance with the choke flow eqs. (4.2) and (4.3). This balance is achieved simultaneously in the vaporizer, solar heater and turbine, as the back pressure is propagated upstream from the turbine. Therefore the increase in temperature is strongly moderated by the resulting decrease in mass flow through the turbine due to choking as shown for Level 1 in **Figure 4.15**. The consequence is that the overall gross power increases moderately with superheating as shown in **Figure 4.16.**

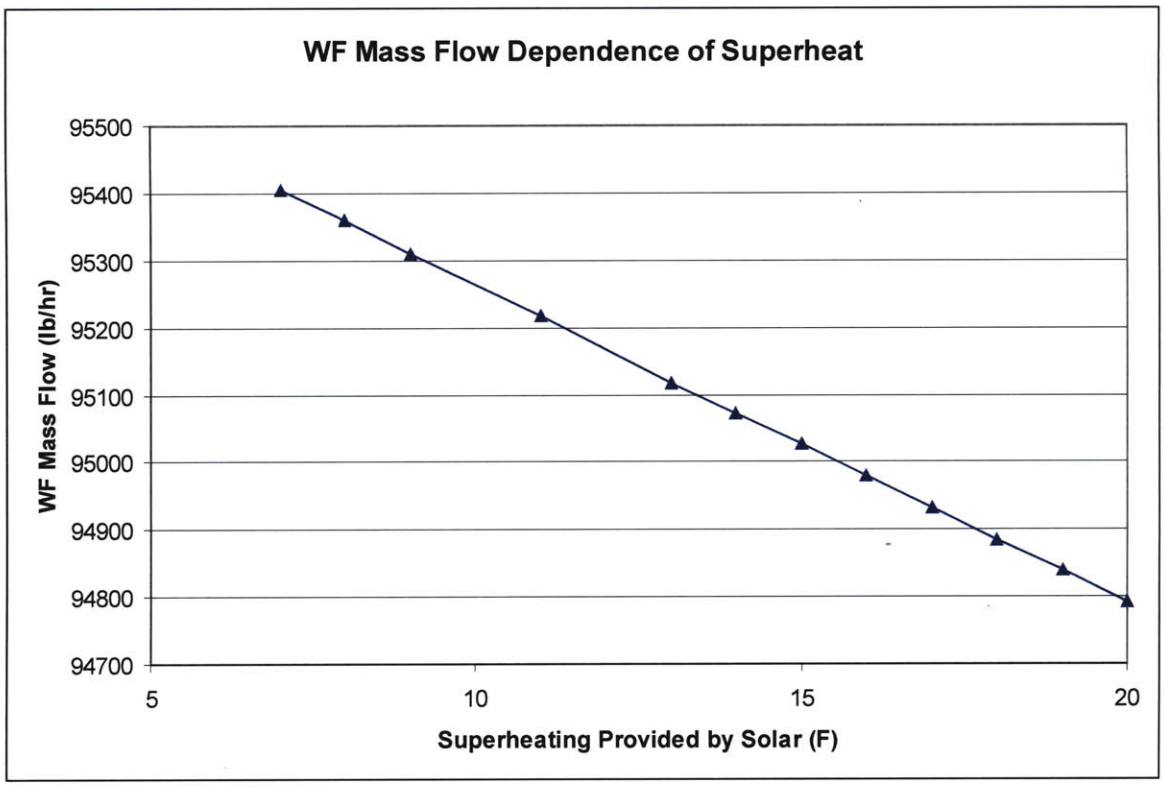


Figure 4.15 Level 1 working fluid mass flow versus degrees superheat.

The main finding is that the superheat option has lower utilization efficiency than the reheat configuration. More important is the fact that the potential for significant solar power generation from the superheat configuration is very constrained by the operating limits of the existing equipment. The superheat configuration will encounter temperature limits if the solar power is used to generate more than 0.20 MW per unit, whereas the reheat configuration can boost the gross power generation of a single unit by 0.57 MW without exceeding the nominal pressure and temperature for the original Ormat design basis.

The fundamental shortcoming of the superheat case is that the solar heat is not being used to increase the working fluid flow and therefore substantial improvements in power generation cannot be achieved. Power generation with the geothermal plant is currently very limited by the low quality brine at 270°F (132.2°C). Therefore the solar heat must be applied in a manner that will compensate for this bottleneck in the process. Potential alternative configurations are described in the following section of the report.

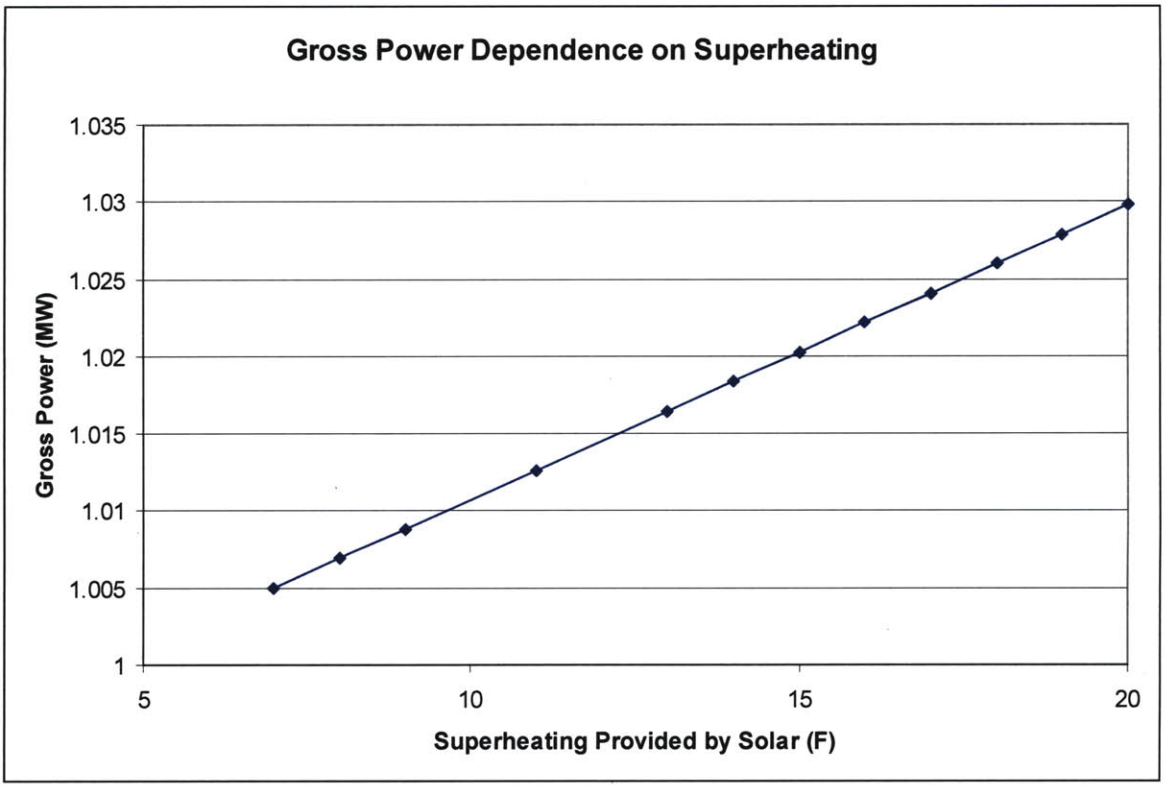


Figure 4.16 Gross power generation for the superheat hybrid configuration.

4.4.2 Hybrid Cycle Design Results for Viable Candidate Systems

The following hybrid systems that were described in **Sections 4.3.2 to 4.3.5** were analyzed using Aspen Plus and the results are presented in this section:

- Brine Preheat concept (Preheat)
- Brine Recirculation concept (Recirculation)
- Brine Preheat-Recirculation concept (Preheat-Recirc)
- Brine Cascade Reheat concept (Cascade Reheat).

The results are based on the performance of the hybrid systems compared to the reference geothermal plant described in **Section 4.2.4**, that is, a geothermal plant with an incoming brine temperature of 270°F (132.2°C) and a volumetric flow rate of 750 GPM (47.31 kg/s).

The results of the design studies are presented in the composite **Table 4.2**. This table includes the power generated, working fluid conditions, and overall efficiency values for each system, for an assumed ambient temperature of 94°F (34.4°C). Included are the Reference Case results against which the hybrid results are compared. The power benefits of the hybrid systems over the Reference Case are denoted as "Incremental". The efficiency values are compared graphically in **Figure 4.17**.

The cascade reheat configuration as described in **Section 4.3.5** was modeled in Aspen Plus to represent two units. When solar energy is available, these units operate in series with the brine from the first unit reheated by solar energy before entering in the second unit. The results presented in this section for the cascade reheat configuration pertain to the performance of the second unit heated by solar energy, where this is feasible.

Table 4.2 Summary of Design Analysis for Reference Case and Four Hybrid Systems.

Parameter	Reference	Recirculation	Cascade Reheat	Preheat	Preheat-Recirc
Thermal eff.	0.046	0.061	0.055	0.061	0.064
Utilization eff.	0.190	0.134	0.136	0.155	0.146
Solar incremental eff.	n/a	0.104	0.224	0.127	0.110
Solar functional inc. eff.	n/a	0.114	0.236	0.151	0.124
Global exergetic eff.	n/a	0.088	0.119	0.091	0.095
Solar collector area (acres)	n/a	3.51 (142 m^2)	5.72 (23148 m^2)	1.99 (8053 m^2)	3.43 (13881 m^2)
Solar collector cost (US$)	n/a	$3,551,408	$5,783,656	$2,014,837	$3,472,645
Level 1					
WF mass flow (kg/s)	11.86	18.44	17.24	17.24	18.19
Vaporizer Inlet temp (°C)	50.8	61.1	59.3	59.3	60.7
Vaporizer inlet press (bar)	9.35	14.97	13.93	13.93	14.75
Turbine inlet temp (°C)	115.2	139.7	135.7	135.7	138.8
Turbine inlet press (bar)	9.32	14.94	13.9	13.9	14.72
Turbine outlet temp (°C)	81.9	98.1	95.4	95.4	97.5
Turbine outlet press (bar)	2.14	2.81	2.68	2.68	2.78
Level 2					
WF mass flow (lbm/h)	14.58	21.92	18.17	18.17	21.73
Vaporizer inlet temp (°C)	51.9	61.2	56.5	56.5	61
Vaporizer inlet press (bar)	4.63	7.13	5.84	5.84	7.06
Turbine inlet temp (°C)	83.5	102.3	93.4	93.4	101.9
Turbine inlet press (bar)	4.61	7.11	5.82	5.82	7.04
Turbine outlet temp (°C)	68.6	82.5	75.8	75.8	82.2
Turbine outlet press (bar)	2.21	2.84	2.5	2.5	2.83
Unit Power					
Gross Power (kW)	753.3	1380.1	1186.3	1185.7	1359.5
Parasitic Load (kW)	265.2	374.5	304.2	296.6	312.5
Net Power (kW)	488.1	1005.6	882.1	889.2	1047.0
Incremental Power (kW)	0.0	517.5	882.1	401.0	558.9

The Preheat-Recirculation system generates the largest net power, namely, 1047 kW, which is more than twice the net power generated by the geothermal reference plant. In fact, all four hybrid systems generate much more power than the non-hybrid system.

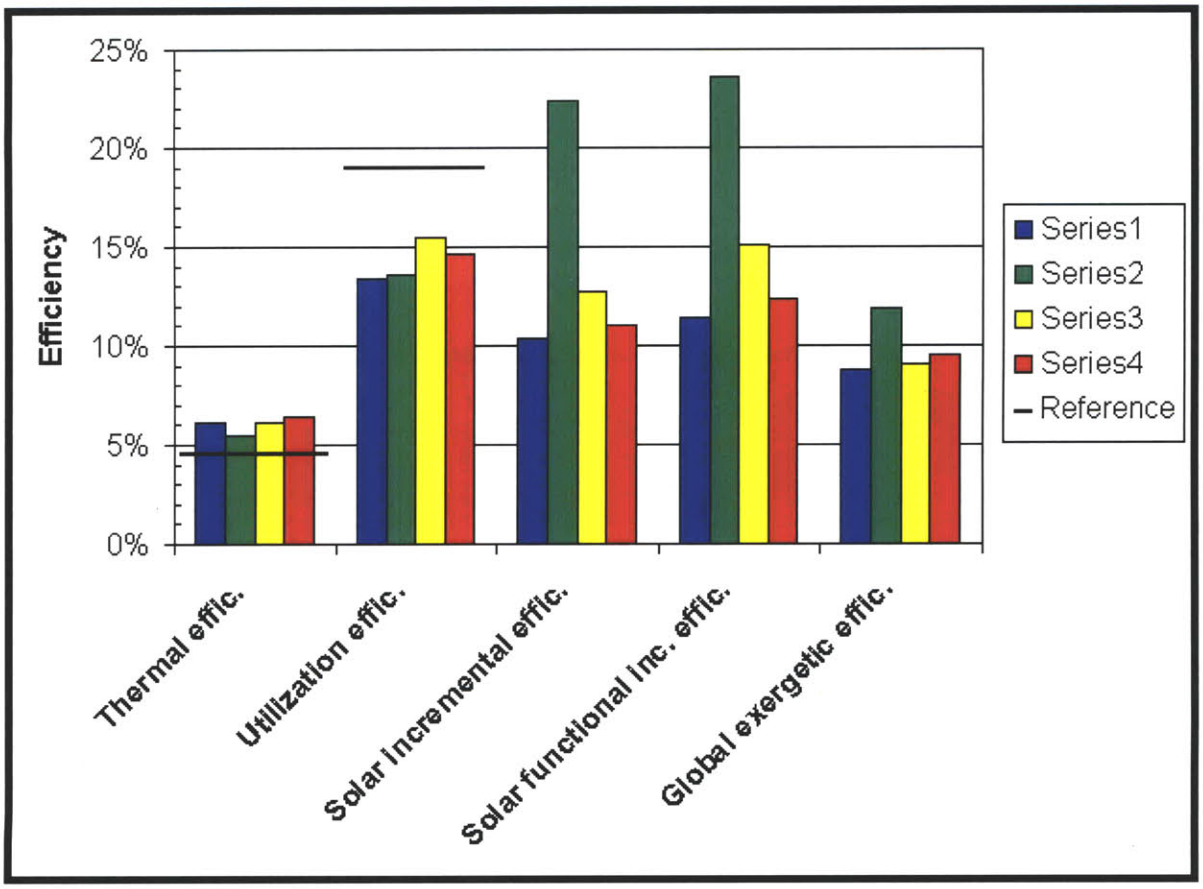


Figure 4.17 Comparison of Various Efficiencies for Candidate Hybrid Solar-Geothermal Systems: Series 1 = Recirculation; Series 2 = Cascade Reheat; Series 3 = Preheat; Series 4 = Preheat-Recirculation.

Table 4.2 shows the average solar collector area required for each hybrid system. Notice the sharp difference between the Preheat system on the low end and the Cascade Reheat system on the high end. This is due to the fact that the Cascade Reheat system needs to heat up a larger mass flow rate of brine and needs to heat the brine from a lower temperature (the unit outlet brine temperature) than the other systems. The Preheat system requires the least amount of solar area because the solar energy is used to raise the temperature of brine that is already hot.

The cost of the solar system as shown in the table is based on a unit cost of US$250/m^2. Of course, the cost is proportional to the area required. Thus, the Cascade Reheat system is the most expensive and the Preheat system is the least expensive. The other two hybrid systems are roughly in the middle between these two extremes.

The table also includes the thermal and utilization efficiencies of each hybrid system relative to the non-hybrid reference system. Notice that each hybrid system has a lower utilization efficiency than the reference case, indicative of a more wasteful use of incoming exergy in the hybrid configurations. The thermal efficiencies, however, are all higher for the hybrid cases than the pure geothermal plant.

It is seen that the solar utilization efficiency of the Cascade Reheat system is much higher than the other hybrid systems. However, the extra solar collector area needed in this case renders the system the most expensive of the four candidate systems.

The global exergetic efficiency was calculated using eq. (4.13) for all cases except the Cascade Reheat case, which used eq. (4.14). This was because the brine coming into the hybrid unit in the Cascade Reheat case was from the exit of the non-hybrid unit. Thus, it could not be compared similarly to the brine coming out of the ground in the other models.

4.4.3 Hybrid Cycle Operating Results for Viable Candidate Systems

In this section the power output is applied for each viable hybrid system over a full typical year to determine the total energy (electricity) that can be generated on an annual basis. Three systems will be studied: the Preheat, the Preheat-Recirculation and the Cascade Reheat (Reheat) systems. The Recirculation system was not considered here; it was ruled out because the Preheat-Recirculation system was seen as the more promising concept of the two systems for the old plant owing to its greater operational flexibility.

Historic solar and temperature data for Fallon, NV, were extracted from the National Solar Radiation Database. **Figure 4.18** shows a typical June 21 day at Fallon: the insolation in W/m^2 and the air temperature in ^{o}F are plotted for each hour of the day.

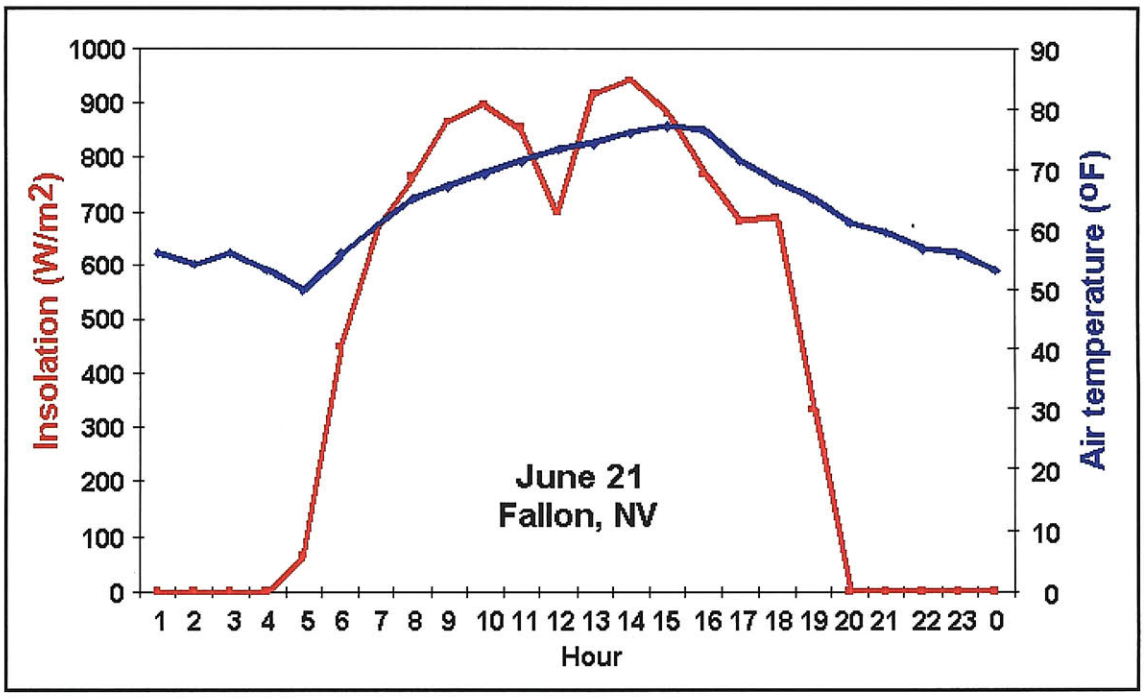


Figure 4.18 Hourly insolation and air temperature for a typical June 21 day at Fallon, NV. Ambient temperature ranges from 50°F to 78°F (10°C to 25.6°C).

Typical months are pieced together in the NSRD to form a typical year. For each hour of the year, the net power was calculated as generated under two scenarios:

Geothermal power only: The geothermal binary plant operated alone with no assist from the sun.
Hybrid power: The geothermal and solar inputs were combined according to the hybrid configuration under study for whatever amount of incident sunlight was available.

The final step involved summing the entire year's production of energy for each system. The results are presented in a form that shows the output from geothermal only, from the hybrid system, and the incremental annual energy generation that results from the use of the solar collector system.

Figure 4.19 presents an example of a monthly power output for the geothermal-only case as a function of the air temperature. The output is for a single unit, both levels, for the current plant configuration (i.e., no preheaters).

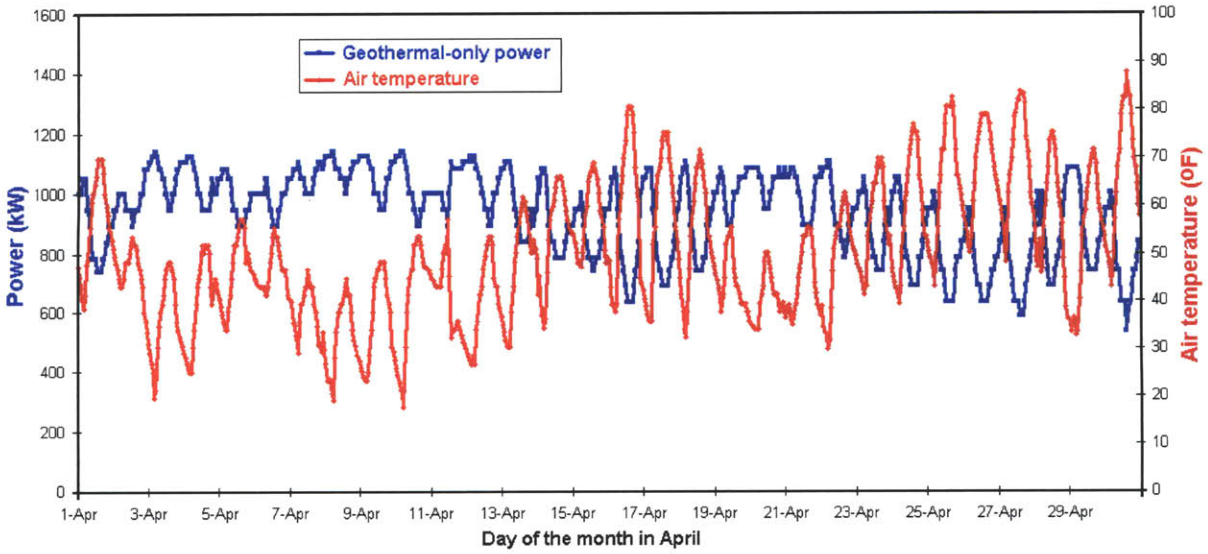


Figure 4.19 Current plant power output as a function of air temperature for a typical month of April in Fallon. NV. Ambient temperature ranges from 18°F to 89°F (-7.8°C to 31.7°C).

Preheat Hybrid System Operating Results:

The power output for the Preheat hybrid system is shown in **Figure 4.20**, along with the insolation and air temperature for each day of the typical April month. **Figure 4.21** shows a 3-D plot of power over the full range of the parametric study.

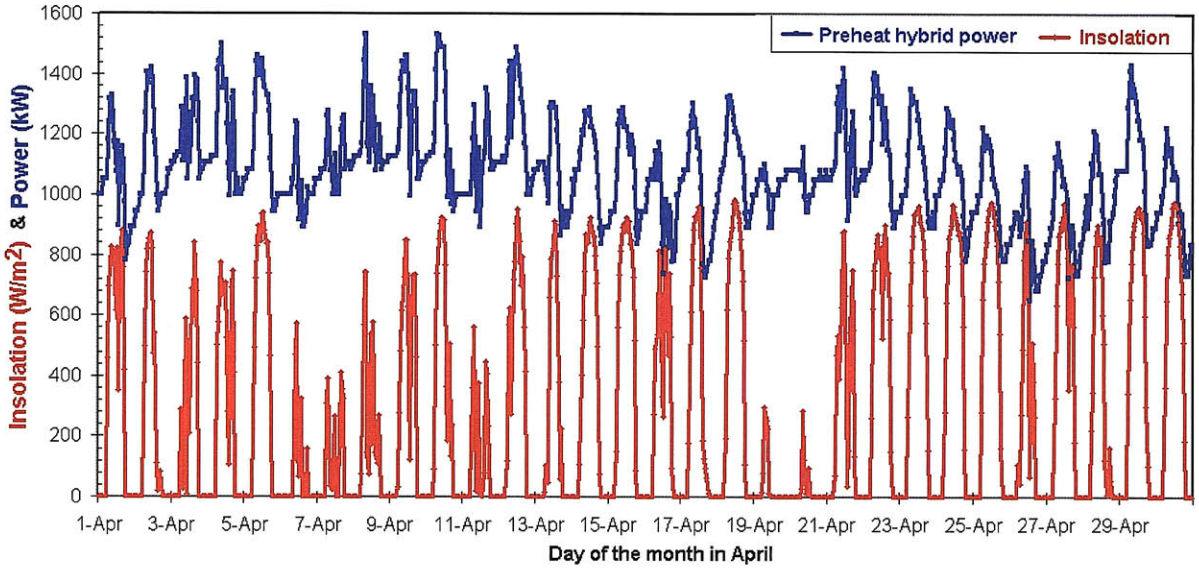


Figure 4.20 Power output for Preheat system over the typical April month.

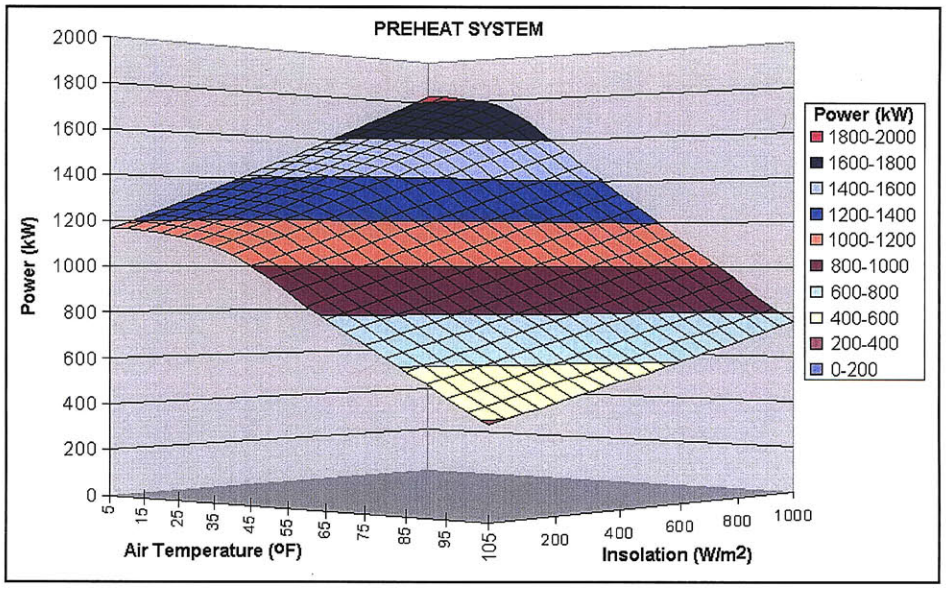


Figure 4.21 Power output for the Preheat system over the range of air temperature and solar insolation values studied.

The monthly electricity generated from the Preheat hybrid system is given in **Figure 4.22**. The solar contribution to the total generation varies from about 8% in winter to about 20% in the summer. The total annual generation comes to 8,980.6 MWh and the solar portion amounts to 1,202.5 MWh. In the month of July, the worst for generation of any binary plant, the Preheat system generates 25% more electricity than the base geothermal plant. In July, the Preheat system can produce about 67% of its peak (winter) generation. The base geothermal plant can produce only 60% of its peak generation in July.

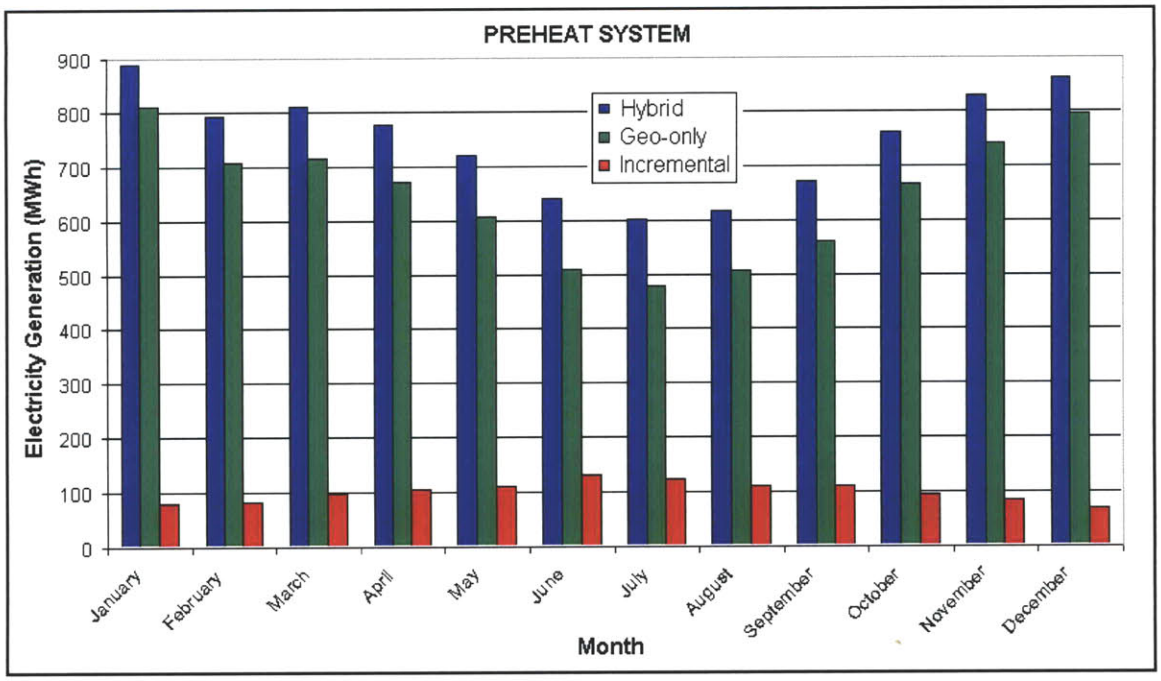


Figure 4.22 Monthly electricity generation for a typical year for the Preheat system.

Preheat-Recirculation Hybrid System Operating Results:

The results for the Preheat-Recirculation hybrid system are presented in **Figures 4.23 and 4.24.** It can be seen that the solar contribution to the electricity generation varies from about 11% in the winter to about 26% in the summer. The total annual generation comes to 9,347.4 MWh and the solar portion amounts to 1,574.6 MWh. In the month of July, the Preheat-Recirculation system generates 33% more electricity than the base geothermal plant. In July, the Preheat-Recirculation system can produce about 71% of its peak (winter) generation. The base geothermal plant can produce only 60% of its peak generation in July.

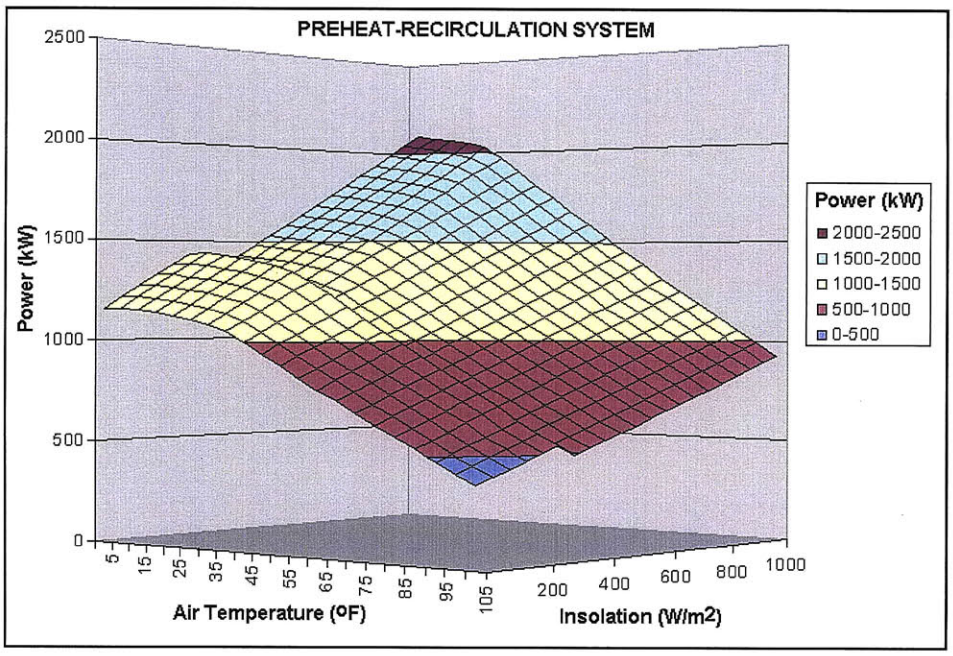


Figure 4.23 Power output for the Preheat-Recirculation system over a typical year as a function of air temperature and solar insolation. For lower values of insolation, this system acts as a preheat-only system and borrows data from Figure 4.21.

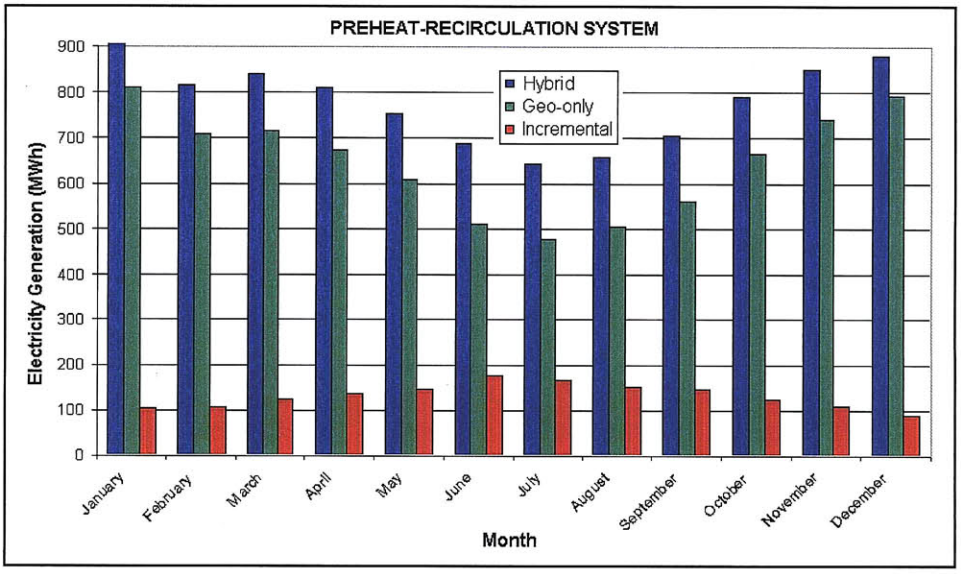


Figure 4.24 Monthly electricity generation for a typical year for the Preheat-Recirculation system.

Cascade Reheat Hybrid System Operating Results:

The results for the Cascade Reheat hybrid system are presented in **Figures 4.25 and 4.26**. It can be seen that the solar contribution to the electricity generation varies from about 11% in the winter to about 26% in the summer. The total annual generation comes to 10,326.8 MWh and the

solar portion amounts to 2,548.6 MWh. In July, the Cascade Reheat system generates 52% more electricity than the base geothermal plant, and can produce about 81% of its peak (winter) generation. The base geothermal plant can produce only 60% of its peak generation in July.

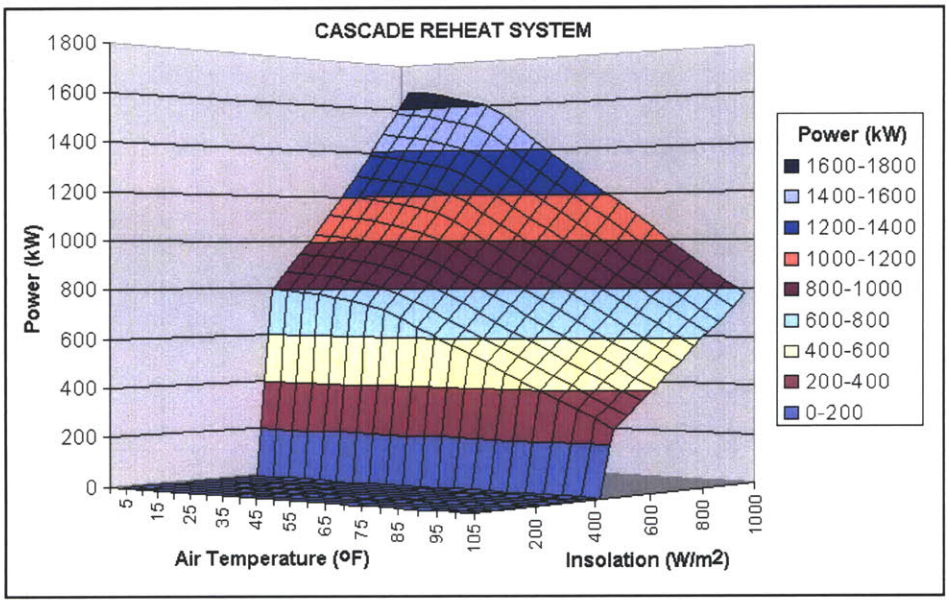


Figure 4.25 Power output for the Cascade Reheat system over a typical year as a function of air temperature and solar insolation.

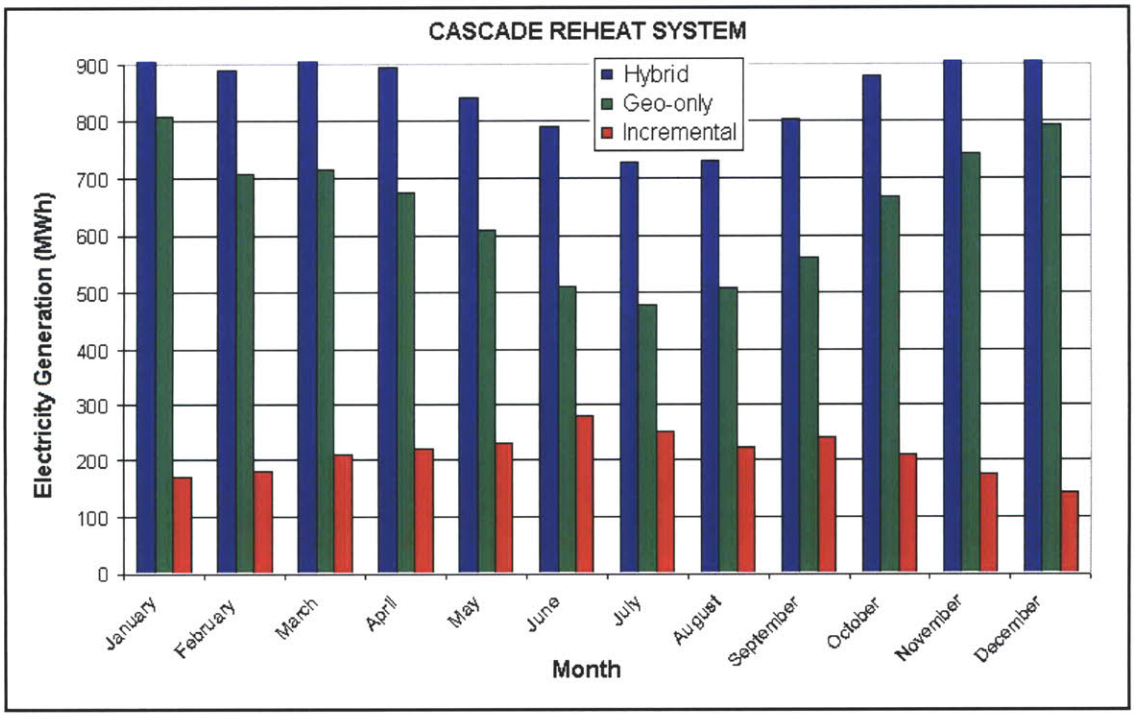


Figure 4.26 Monthly electricity generation for a typical year for the Cascade Reheat system.

Comparison of Power Outputs with Ambient Conditions:

The following three graphs, **Figures 4.27, 4.28 and 4.29**, show how the power of the three viable hybrid systems varies depending on the ambient conditions. **Tables 4.3, 4.4, and 4.5** give important cycle properties and system parameters. Three snapshots of air temperature and solar insolation were taken to represent a hot, sunny day, a moderate day, and a cold, cloudy day. The full height of the bars in the figures shows the gross power, with the parasitic power represented by the red top portions and the net power by the blue bottom portions.

To keep the results in perspective, it must be remembered that the Preheat and the Preheat-Recirculation systems operate with a single unit (both levels); the Cascade Reheat system operates with two units when the sun is available and reverts to one unit when the sun is not available. **Table 4.5** shows that on the cold, cloudy day, the Cascade Reheat system cannot drive the second unit since there is insufficient solar input to raise the brine temperature to a high enough value for it to be used in the second unit.

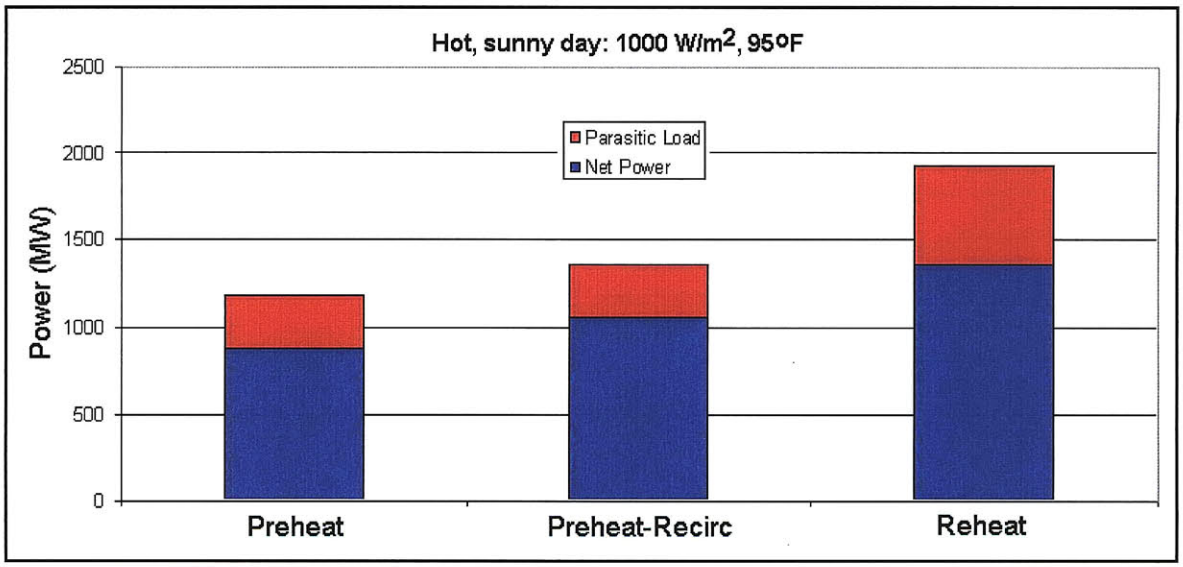


Figure 4.27 Comparison of three hybrid systems on a hot, sunny day. Amb. temp. is 95°F (35°C)

Table 4.3 Results for a hot, sunny day with 1000 W/m^2 and 95°F (35°C).

Parameter	Preheat	Preheat-Recirc.	Reheat
Brine Temperature (°C)	160	160	160
Level 1			
WF mass flow (kg/s)	17.24	18.18	17.26
Vaporizer Inlet temp (°C)	59.5	60.9	59.6
Vaporizer inlet press (bar)	13.93	14.74	13.94
Turbine inlet temp (°C)	135.7	138.8	135.8
Turbine inlet press (bar)	13.9	14.71	13.91
Turbine outlet temp (°C)	95.6	97.6	95.7
Turbine outlet press (bar)	2.7	2.79	2.7
Level 2			
WF mass flow (kg/s)	18.19	21.73	18.2
Vaporizer inlet temp (°C)	57.1	60.6	56.9
Vaporizer inlet press (bar)	5.85	7.06	5.85
Turbine inlet temp (°C)	93.5	101.9	93.5
Turbine inlet press (bar)	5.83	7.04	5.83
Turbine outlet temp (°C)	76.1	81.9	76.1
Turbine outlet press (bar)	2.53	2.79	2.53
Unit Power			
Gross Power (kW)	1,176.4	1,363.8	1,922.9
Parasitic Load (kW)	297.1	311.5	560.9
Net Power (kW)	879.3	1,052.3	1,362.0

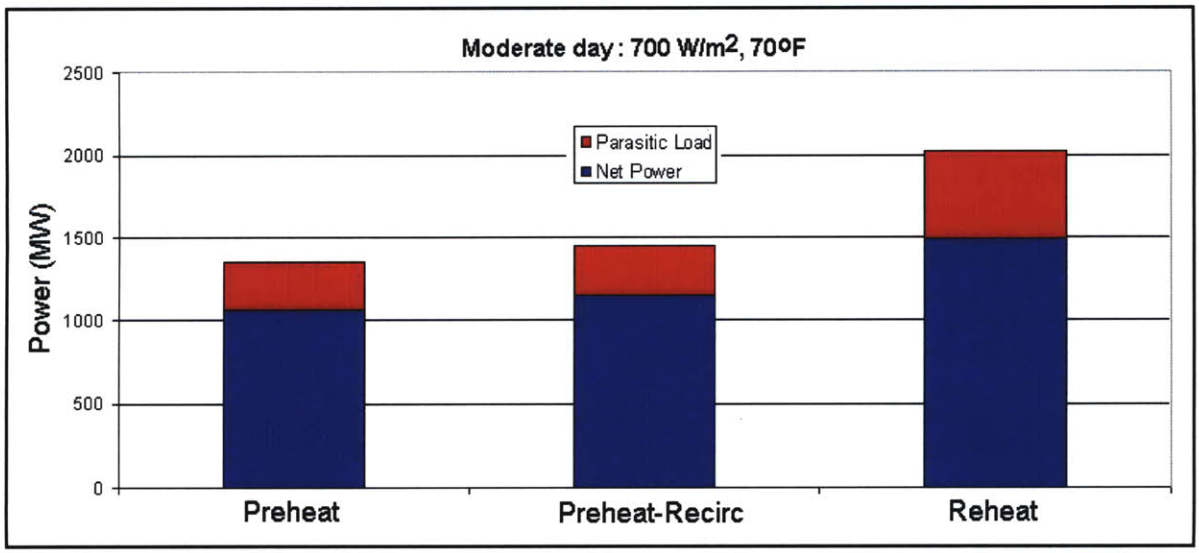


Figure 4.28 Comparison of three hybrid systems on a moderate day. Amb. temp. is 70°F (21.1°C)

Table 4.4 Results for moderate day with 700 W/m^2 and 70°F.

Parameter	Preheat	Preheat-Recirc.	Reheat
Brine Temperature (°C)	151.7	147.3	133
Level 1			
WF mass flow (kg/s)	15.46	15.34	11.97
Vaporizer Inlet temp (°C)	43.6	43.4	37.5
Vaporizer inlet press (bar)	12.4	12.3	9.44
Turbine inlet temp (°C)	129.6	129.1	115.7
Turbine inlet press (bar)	12.37	12.27	9.41
Turbine outlet temp (°C)	84.4	84.1	74.8
Turbine outlet press (bar)	1.72	1.72	1.45
Level 2			
WF mass flow (kg/s)	16.51	18.97	14.23
Vaporizer inlet temp (°C)	41.1	44.2	38
Vaporizer inlet press (bar)	5.28	6.12	4.51
Turbine inlet temp (°C)	89.1	95.4	82.4
Turbine inlet press (bar)	5.26	6.09	4.49
Turbine outlet temp (°C)	65.6	70.3	60.7
Turbine outlet press (bar)	1.61	1.76	1.47
Unit Power			
Gross Power (kW)	1,351.4	1,448.7	1,021.9
Parasitic Load (kW)	289.6	294.7	536.8
Net Power (kW)	1,061.8	1,154.0	1,488.2

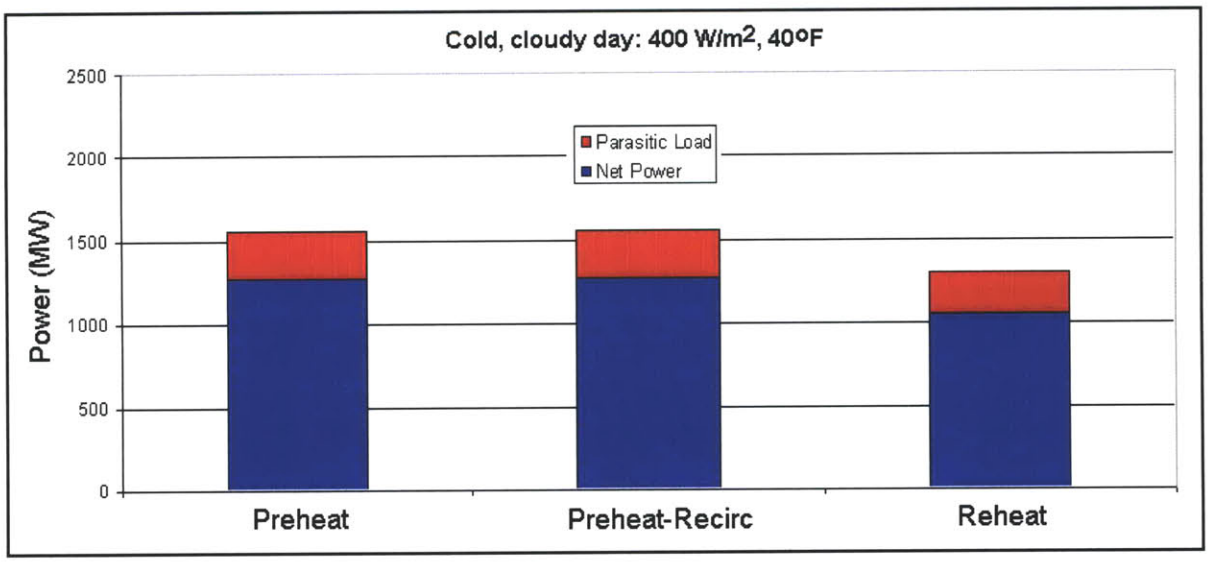


Figure 4.29 Comparison of three hybrid systems on a cold, cloudy day. Amb. temp is 40°F (4.4°C)

Table 4.5 Results for cold, cloudy day with 400 W/m^2 and 40°F.

Parameter	**Preheat**	**Preheat-Recirc.**	**Reheat**
Brine Temperature (°C)	143.4	134.5	n/a
Level 1			
WF mass flow (kg/s)	13.81	12.8	n/a
Vaporizer Inlet temp (°C)	24.7	23	n/a
Vaporizer inlet press (bar)	11	10.14	n/a
Turbine inlet temp (°C)	123.3	119.2	n/a
Turbine inlet press (bar)	10.97	10.11	n/a
Turbine outlet temp (°C)	71.1	68.3	n/a
Turbine outlet press (bar)	0.97	0.92	n/a
Level 2			
WF mass flow (kg/s)	14.91	16.39	n/a
Vaporizer inlet temp (°C)	22.6	24.7	n/a
Vaporizer inlet press (bar)	4.74	5.24	n/a
Turbine inlet temp (°C)	84.5	88.7	n/a
Turbine inlet press (bar)	4.72	5.22	n/a
Turbine outlet temp (°C)	53.4	56.6	n/a
Turbine outlet press (bar)	0.9	0.96	n/a
Unit Power			
Gross Power (kW)	1,557.5	1,553.4	1,301.0
Parasitic Load (kW)	282.8	280.2	251.7
Net Power (kW)	1,274.7	1,273.2	1,049.4

4.5 Economic analysis

The two graphs below, **Figures 4.30 and 4.31**, give the levelized cost of the incremental electricity (LCOE) gained through the use of the solar energy in the hybrid system. It is the sum of the capital cost for the new solar system (but includes only the solar collector cost represented as a certain cost in US$/m^2 of collector area) plus a nominal $0.02/kWh for operating and maintenance costs.

For this economic assessment, all costs are assumed to incur during the first year, followed by 25 years of revenue. The model assumes that all costs are incurred in the first year, as the money required would have to be raised and committed within the first year even though some disbursements may be made over the 18-24 month construction period. Affects of some creative financing are not considered but could be included to fine tune the economic model.

The cost of hybridization of the existing geothermal plant is dominated by the solar collection system. Therefore the costs presented in this report are based on the solar collector system. The additional costs for interfacing the geothermal and solar collector systems, such as piping, heat exchangers, valves, control systems and pumps are not included since these costs are small relative to the substantial uncertainty regarding the solar collector costs. In addition, no taxes or tax credits were factored into these calculations. Given these assumptions, the actual cost of electricity may differ from what is shown here. The results presented should be used for relative economics for comparing alternative systems. These results can be enhanced by applying them within a more sophisticated economic model.

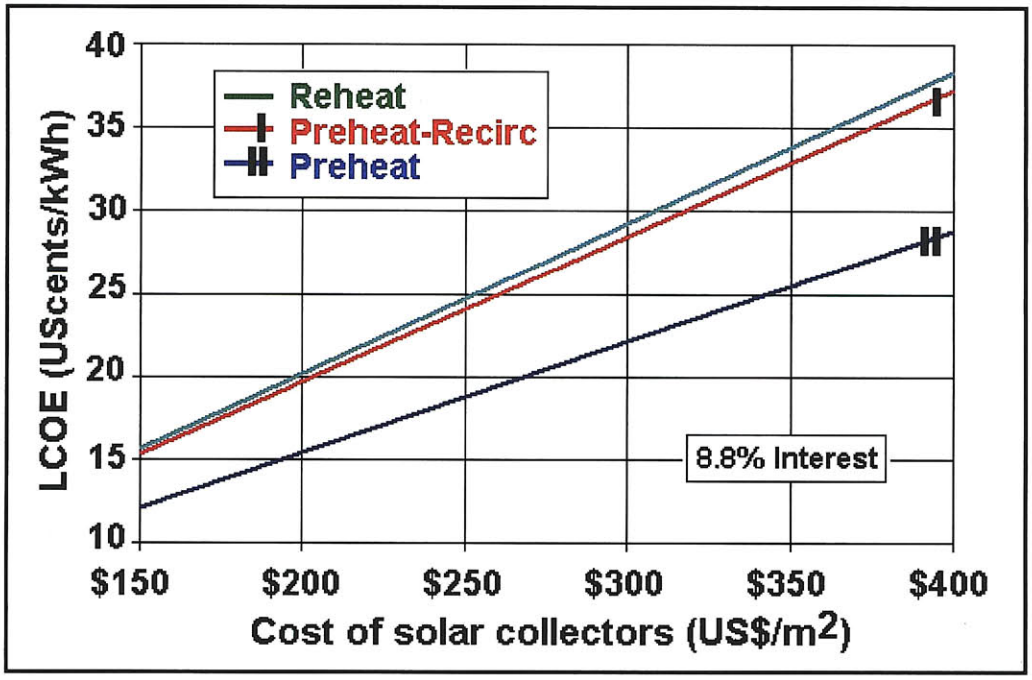


Figure 4.30 Cost of electricity versus solar collector cost.

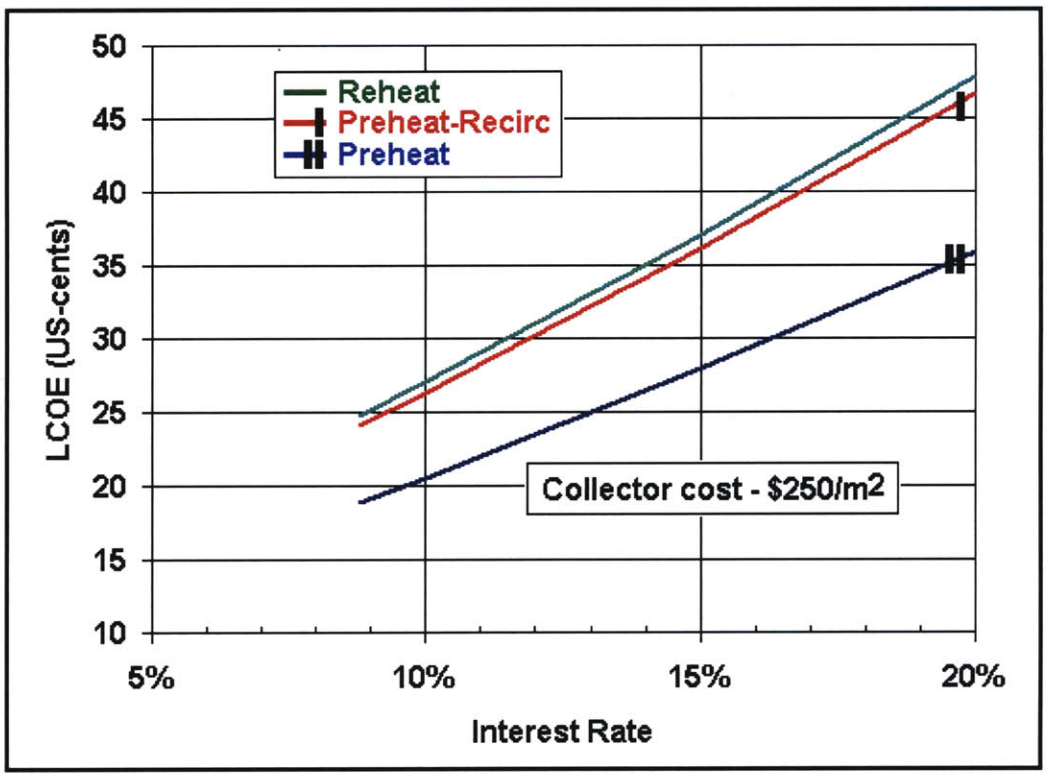


Figure 4.31 Cost of electricity versus interest rate.

The results show that the incremental unit cost of electricity ranges from about \$0.19-0.25/kWh for a collector cost of \$250/m^2 and an interest rate of 8.8%. Of the three viable hybrid systems, the Preheat system produces the lowest incremental LCOE. This cost may be compared to the current cost to generate a kWh from the existing plant to determine if adding the solar field to form a geothermal-solar hybrid plant makes economic sense.

Tables 4.6 and **4.7** show the net present value (NPV) and pay-back period (PBP) calculated for different assumed prices of electricity and solar collector costs. At the prices of \$0.095-0.115/kWh, the NPV are all negative for both assumed costs for the solar collector, which suggests that under these assumptions, this investment is not economically viable, given the previously stated provisos.

Table 4.6 Summary of economic results for a collector cost of $250/m^2$.

Hybrid Type	Interest Rate %	Price of Electricity US$/MWh	NPV US$1,000	PBP yr
Preheat	8.8%	95	($1,024)	22.3
Preheat	8.8%	105	($914)	19.7
Preheat	8.8%	115	($804)	17.6
Preheat	10.0%	95	($1,087)	22.3
Preheat	10.0%	105	($988)	19.7
Preheat	10.0%	115	($889)	17.6
Preheat-Recirc.	8.8%	95	($2,109)	29.4
Preheat-Recirc.	8.8%	105	($1,965)	26.0
Preheat-Recirc.	8.8%	115	($1,821)	23.2
Preheat-Recirc.	10.0%	95	($2,184)	29.4
Preheat-Recirc.	10.0%	105	($2,054)	26.0
Preheat-Recirc.	10.0%	115	($1,924)	23.2
Cascade Reheat	8.8%	95	($3,562)	30.3
Cascade Reheat	8.8%	105	($3,328)	26.7
Cascade Reheat	8.8%	115	($3,094)	23.9
Cascade Reheat	10.0%	95	($3,681)	30.3
Cascade Reheat	10.0%	105	($3,470)	26.7
Cascade Reheat	10.0%	115	($3,260)	23.9

Table 4.7 Summary of economic results for a collector cost of $150/m^2$.

Hybrid Type	Interest Rate %	Price of Electricity US$/MWh	NPV US$1,000	PBP yr
Preheat	8.8%	95	($284)	13.4
Preheat	8.8%	105	($173)	11.8
Preheat	8.8%	115	($63)	10.6
Preheat	10.0%	95	($355)	13.4
Preheat	10.0%	105	($256)	11.8
Preheat	10.0%	115	($156)	10.6
Preheat-Recirc.	8.8%	95	($833)	17.7
Preheat-Recirc.	8.8%	105	($688)	15.6
Preheat-Recirc.	8.8%	115	($544)	13.9
Preheat-Recirc.	10.0%	95	($921)	17.7
Preheat-Recirc.	10.0%	105	($791)	15.6
Preheat-Recirc.	10.0%	115	($661)	13.9
Cascade Reheat	8.8%	95	($1,435)	18.2
Cascade Reheat	8.8%	105	($1,202)	16.0
Cascade Reheat	8.8%	115	($968)	14.3
Cascade Reheat	10.0%	95	($1,577)	18.2
Cascade Reheat	10.0%	105	($1,367)	16.0
Cascade Reheat	10.0%	115	($1,157)	14.3

Table 4.8 shows the price of electricity that would be required for a 15% internal rate of return (IRR) based solely on the cost of the solar thermal collector. At $250/m^2$, the price varies from $280-370/MWh. At $150/m^2$, the prices vary from $175-230/MWh.

Table 4.8 Price of electricity required for 15% IRR.

Hybrid Type	Cost of collector US$/m^2	Price of Electricity US$/MWh
Preheat	250	280
Preheat-Recirc.	250	360
Cascade Reheat	250	370
Preheat	150	175
Preheat-Recirc.	150	225
Cascade Reheat	150	230

Chapter 5

5. Scenario Plant B

5.1 Objective

The objective of scenario plant B was to develop innovative solar-geothermal hybrid energy conversion systems for new low enthalpy geothermal-solar hybrid power plants. The goal was to find hybrid solutions that take advantage of the potential synergies of solar thermal and geothermal power cycles.

5.1.1 Approach

Preliminary conceptual designs of hybrid systems were developed in which solar heating was optimally integrated with a supercritical cycle using R134a as a working fluid. This geothermal only R134a supercritical cycle was selected based on unpublished research at M.I.T. on the utilization of low-enthalpy geothermal fluids [54] (see chapter 2 for details of this study). Based on economic and thermodynamic performance metrics, the most promising hybrid designs were modeled in Aspen Plus. Accurate pure component property data and equation of state models were utilized for mass and energy balance calculations. Based on the results of several sensitivity analyses, the most promising design was selected for detailed dynamic study.

A dynamic study using Aspen Dynamics was performed for the most promising hybrid design. This included a control system to manage flows in the model and to manage solar energy storage. Dynamic analysis was performed for a typical January and typical July day for configurations with different amounts of solar energy storage.

An economic assessment of the plant was performed using Aspen ICARUS, industry data, and rules of thumb from the National Renewable Energy Laboratory (NREL) to determine plant costs. Coupled with dynamic model performance, this assessment determined the levelized cost of electricity (LCOE) for various configurations to determine the lowest cost solar field size and storage amount.

5.2 Plant Design

5.2.1 Solar-Geothermal Hybrid Conceptual Designs

At the outset of this project, several conceptual designs were created for hybrid solar-geothermal power plants. A subset of these was for binary geothermal power plants. **Figures 5.1** and **5.2** below are illustrations of these designs in which the solar resource is added to different stages of the binary cycle.

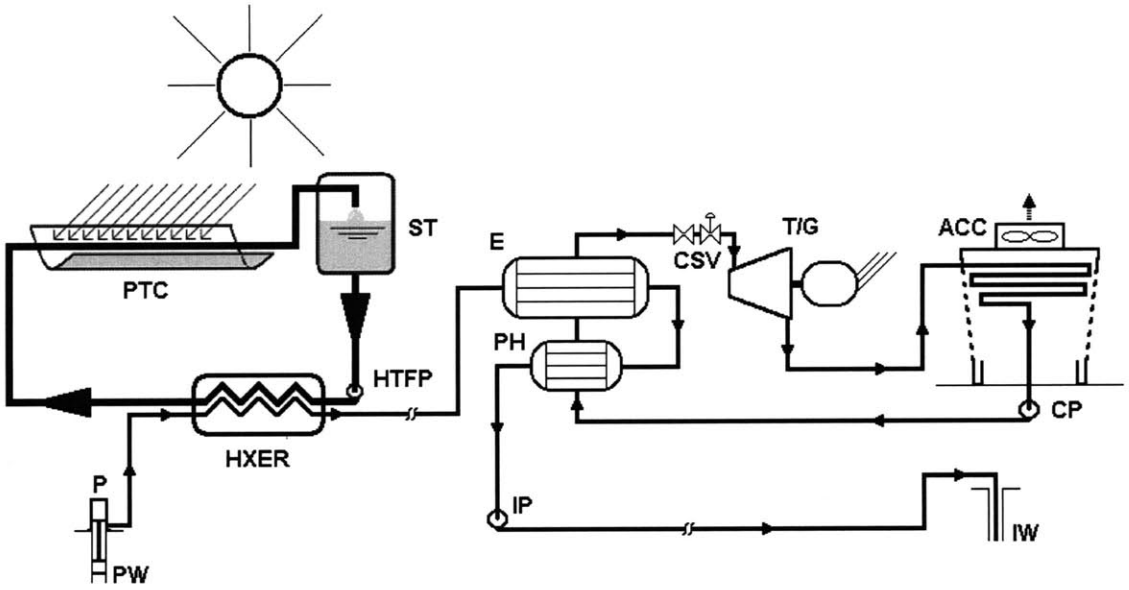


Figure 5.1 Hybrid solar-binary geothermal plant with brine temperature boost ("preheat").
(Source: Ron DiPippo, personal communication)

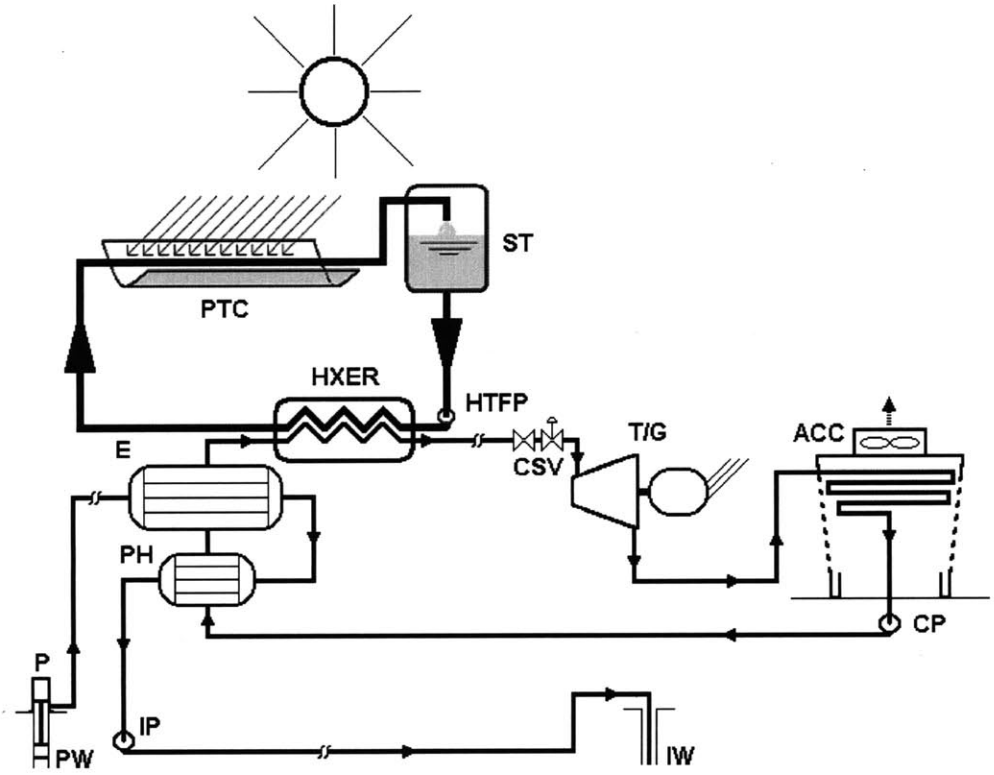


Figure 5.2 Hybrid solar-binary geothermal plant with working fluid superheat ("superheat").
(Source: Ron DiPippo, personal communication)

In the first design ("preheat" model), the solar heat transfer fluid (HTF) is used to heat the brine, and then the heated brine passes its energy to the binary fluid. This design emulates having available a hotter geothermal resource. This is perhaps the easiest way to boost performance of a lower-medium resource geothermal fluid. In this simple design, the brine remains a liquid.

The second design ("superheat" model) uses the solar HTF to directly heat the R134a after it has been heated by the brine. Thus, the solar energy directly boosts the temperature of the working fluid before it passes into the turbine.

Another possible arrangement would use the solar energy to preheat the R134a before it enters the brine preheater. However such a design was immediately eliminated from further consideration since the solar heat transfer fluid (HTF) is much hotter than the geothermal brine. This design would thus eliminate the need for the geothermal input.

In any hybrid system where solar energy is applied directly to the binary cycle working fluid, a critical constraint on the system is the upper temperature limit of R134a. At temperatures above 200°C, R134a begins to decompose. Thus, care must be taken to prevent the R134a temperature from reaching or approaching this limit. To be safe, the R134a should never to go above a bulk temperature of 180°C. This ensures that at any time when there is a temperature gradient in the working fluid, the highest temperatures do not exceed 200°C. This temperature ceiling ultimately limits the amount of solar energy that can be transferred to R134a at a given flow rate. Thus, in order to capture and use as much solar energy as possible in these conceptual designs, the R134a flow rate will become quite large. In order to avoid this consequence, a third conceptual model was developed wherein pressurized, solar-preheated brine is flashed to generate steam for steam-turbine cycle that is integrated with a binary cycle - "solar-flash-binary" model; see **Figure 5.3**. The exhaust steam from a back-pressure turbine is coupled to the R134a cycle through a heat recovery heat exchanger, LTH. This configuration offers the ability to capture and put to use more heat from the solar collector array, thus potentially producing more power at a higher efficiency.

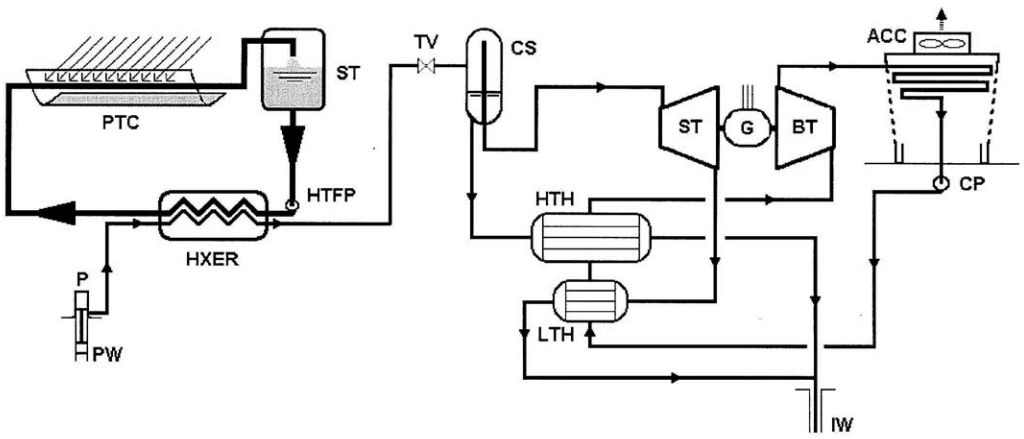


Figure 5.3 Hybrid solar-flash-binary geothermal plant.

(Source: Ron DiPippo, personal communication)

Of these conceptual designs, two were chosen for more detailed steady-state analysis. These are the superheat hybrid and the flash-binary hybrid. The preheat hybrid was omitted because it is similar to but less effective than the flash-binary design.

Superheat Hybrid: Conceptual Design Detail

An obvious means of integrating solar energy into a supercritical binary plant is to use a solar collector to raise the temperature of the R134a before it enters the turbines, thereby increasing the working fluid exergy and making it possible to increase the power from the turbine. **Figure 5.4** shows the process flow diagram of this cycle. The brine heats up the working fluid (WF) via the low-temperature heat exchanger (LTH), and the WF picks up extra heat in the high-temperature solar heat exchanger (HTH). This approach matches the two heat sources to the appropriate temperature levels of the WF: lower temperature brine to the cooler WF and the high temperature solar HTF to the hotter WF.

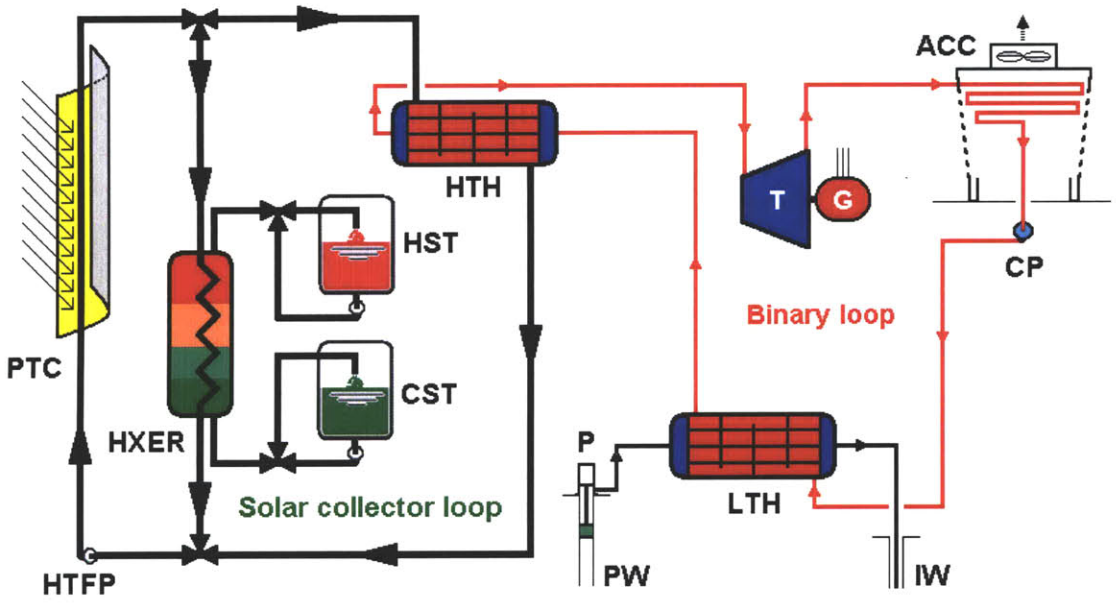


Figure 5.4 Solar-geothermal superheat hybrid concept.

(Source: Ron DiPippo, personal communication)

In this concept, the R134a quickly approaches its upper temperature limit unless its flow rate is dramatically increased. This improves the gross power coming from the turbine (T), but it greatly increases the parasitic loads from the WF circulating pump (CP) and the air cooled condenser (ACC) fans. Thus, there is an optimal flow rate for R134a that corresponds to an optimum net power produced. In addition, one can adjust the supercritical pressure to find the best value, but this does not affect the overall power significantly. These results and detailed flowsheet designs are discussed in **Section 5.3.2**.

This concept requires active monitoring of the solar resource available in order to pump the optimum flow of WF through the loop: too little and the R134a can overheat; too much and the parasitic loads will decrease performance considerably. Thus, this concept is not trivial to manage and will require a sophisticated control system.

Flash-Binary Hybrid: Conceptual Design

The flash-binary hybrid plant is shown below in **Figure 5.5**. It is intended for low-to-moderate geothermal resources with pumped wells. The solar energy is used to raise the temperature of the pressurized geofluid to a sufficiently high value (roughly saturation) to allow flashing at an appropriate pressure. This generates steam and hot brine for use in a steam turbine and for heating the WF in the binary loop, respectively. The flash pressure is an adjustable, optimizable quantity, constrained by the temperature limit on the WF described above.

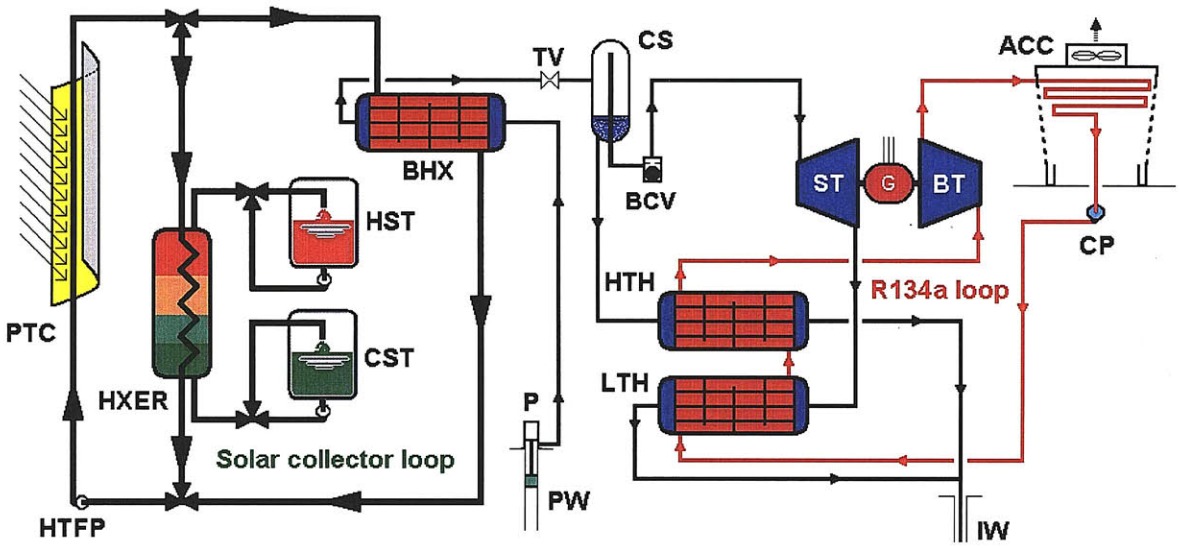


Figure 5.5 Solar-geothermal flash-binary concept.

(Source: Ron DiPippo, personal communication)

The flashed geosteam drives one of the two turbines connected to a common generator, in much the same way as in so-called "combined cycle" geothermal binary plants. After leaving the back-pressure steam turbine (ST), the exhaust steam is condensed against the R134a in the low-temperature heat exchanger (LTH) of the bottoming binary supercritical cycle. The hot separated brine from the separator is used to impart the final heating to the working fluid in the high-temperature heat exchanger (HTH). The solar HTF is used exclusively in the liquid-to-liquid BHX allowing for a good match between the heating and cooling curves, thereby reducing irreversibilities and increasing thermodynamic efficiency of exergy transfer. The ST exhaust pressure is another adjustable, optimizable parameter, as is the R134a binary turbine (BT) inlet pressure.

When the solar energy is no longer available (i.e., when the sun sets or is obscured by clouds, and the thermal storage is depleted), the system will continue to operate as a pure geothermal plant, albeit at a lower power generation rate. The brine would be directed from the well pumps to the two R134a heaters via a bypass line, and the steam turbine would be disconnected from the generator by means of a clutch. Bypass lines, bypass valves and the clutch are omitted from

Figure 5.5 for the sake of clarity. Two separate turbine-generator sets could be employed if the economics are favorable, eliminating the clutch arrangement.

Figure 5.6 shows the process diagram in temperature-entropy coordinates. The diagram is to scale for water substance and the R134a saturation curve is overlain roughly to scale; the s-axis has been stretched for the R134a to render it approximately within the range of water entropy values. The isobars pertain to water; the critical pressure for R134a is approximated 40.6 bar and its critical temperature is 101.2°C. The heat needed to raise the R134a from state D to A is supplied in two steps: from D-E in the low-temperature heater (LTH) with heat coming from the condensing steam (6-7), and from E-A in the high-temperature heater (HTH) with heat coming from the separated, hot brine (5-8). States 7 and 8 are close in temperature but state 8 is at a higher pressure. Since the condensing exhaust steam provides the lower-temperature heat, it could be at a somewhat lower temperature than shown in the figure. Furthermore, since R134a exhibits a normal condensation line, the binary turbine inlet state A could be raised to a higher temperature than shown and prevent any moisture in the last stages of the turbine. This temperature is limited by the approach to the flash temperature, T_3.

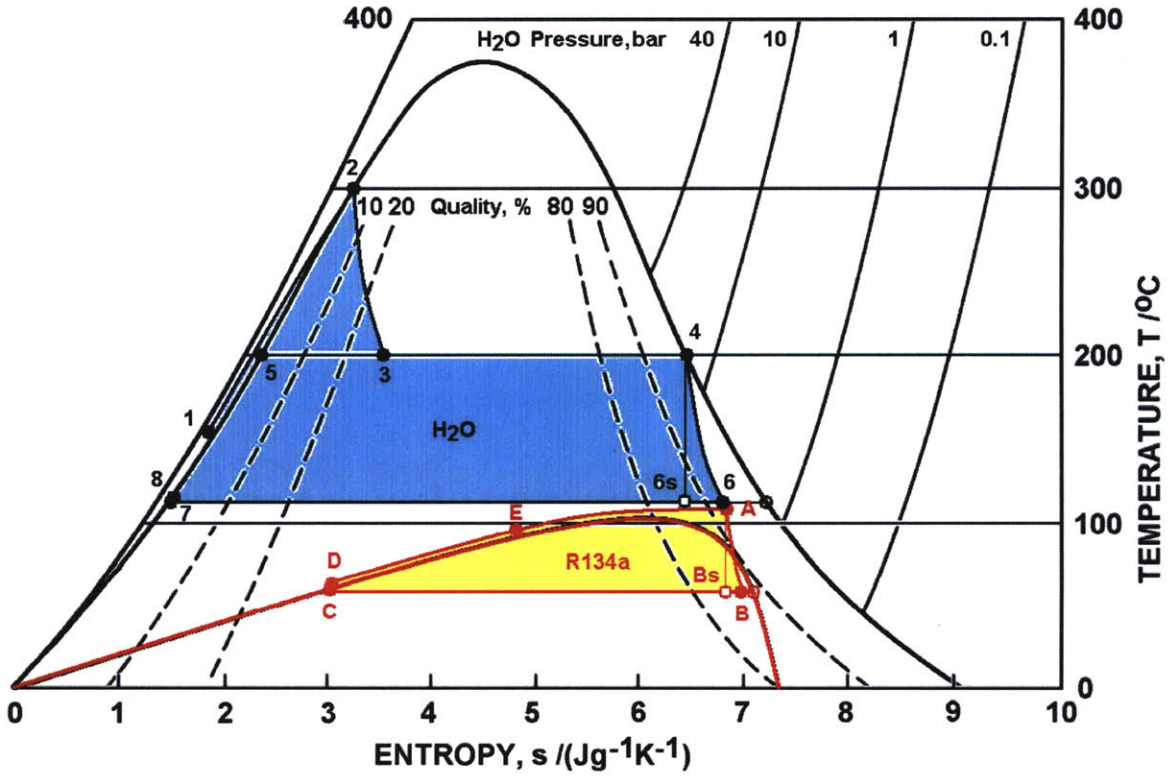


Figure 5.6 Process diagram in temperature-entropy coordinates.

(Source: Ron DiPippo, personal communication)

Night Operation: When the solar energy is not available, the only heat source is the brine at temperature T_1. It will likely be necessary to lower the R134a mass flow rate in order to raise its temperature sufficient to avoid approaching the critical point. Dropping the R134a pressure to subcritical at night would also solve this problem but might cause trouble with the turbine inlet

conditions. With Aspen Plus one can investigate different ways to operate the plant at the night situation. In Aspen Dynamics, one can look at ways to operate the plant with and without sun to avoid temperature extremes and hitting the critical point.

5.2.2 Turbine Selection Criteria

These two hybrid solar-geothermal designs offer challenges to selecting turbines. In the binary cycle, a binary turbine will see a large range of flow from day to night. In the steam cycle for the flash-binary model, the steam turbine will have to accommodate varying amounts of flow as well as turn on/off daily. Given these operating conditions, the following criteria were selected:

- Binary turbine will be a radial inflow turbine with a 30 in. (0.762 m) diameter rotor operating at 5000 rpm.
- Binary turbine turn-down ratio must not exceed 2:1 to avoid penalties in isentropic efficiency.
- A single binary turbine will be used for the binary cycle to avoid start-up/shut-downs during solar cycle.
- Steam turbine will be a multi-stage axial-flow turbine with inter-stage moisture removal.

Suppliers of these turbines are identified in **Table 5.1** with direct contact info where applicable:

Table 5.1 Turbine Suppliers.

Component	Supplier	Contact
Organic Turbine	Ormat	6225 Neil Road Reno, Nevada 89511-1136 Tel.: (775) 356-9029; Fax: (775) 356-9039 e mail: info@ormat.com Josh Nordquist 775-336-0164
Organic Turbine	UTC Power/Pratt & Whitney	Pratt & Whitney 400 Main Street East Hartford, CT 06108 860-565-4321 Power Systems Customer Support 866-769-3725 (866-POWER-ALL)
Organic Turbine	Barber-Nichols	Barber-Nichols Inc. 6325 West 55th Avenue Arvada, CO 80002 Phone: (303) 421-8111; Fax: (303) 420-4679 info@barber-nichols.com http://www.barber-nichols.com
Steam Turbine	Siemens	<10 MW turbines John Lindstrom Phone: +1 (281) 856-4471 Email: john.lindstrom@siemens.com
Steam Turbine	Fuji	http://www.fesys.co.jp/eng/products_services/power_generation/overview/index.html#organization
Steam Turbine	Mitsubishi	100 Colonial Center Parkway Lake Mary, FL 32746 Phone: (407) 688-6100 Fax: (407) 688-6481

Steam Turbine	GE	4200 Wildwood Parkway Atlanta, GA 30339 gepower.com GE Energy Helpdesk TollFree: 877-435-7375
Steam Turbine	Toshiba	https://www3.toshiba.co.jp/ power/english/contact/thermal/mailform1/topmail_e.htm
Steam Turbine	Alstom	Joseph Vasile Vice President Sales ALSTOM Power Inc. 2000 Day Hill Road, PO Box: 500 Phone: +1-860-285-3790; Fax: +1-860-285-3840

5.2.3 Metrics for Comparing Hybrid Options

The selection of one hybrid design over another involves an assessment of the thermodynamic performance advantage coupled with the cost of the new equipment to accomplish the design. Because of the complex coupling between equipment cost and performance, the design approach was to first maximize thermodynamic performance in steady state, take the best-case system to the dynamic environment, and then evaluate performance vs. cost for different levels of solar heat storage. This section details the metrics for comparing hybrid options in Scenario Plant B, which are slightly different than the metrics for Scenario Plant A.

A practical aspect in the selection process concerns the need to keep the geothermal plant in continuous operation, both under hybrid and stand-alone conditions. It is impractical to shut in the wells serving the plant or to shut down a unit when the sun is not shining.

The best subject on which there is no general agreement. One can, however, base his calculations on the exergy of the heat transfer fluid (HTF) that circulates through the solar collectors, using the value of exergy that the HTF possesses when it leaves the solar field and enters the solar-geo heat exchanger, shown as HTH in **Figure 5.4** and as BHX in **Figure 5.5**. This can be found from the following equation:

$$\dot{E}_{HTF} = \dot{m}_{HTF}[h_{1,HTF} - h_{0,HTF} - T_0(s_{1,HTF} - s_{0,HTF})] . \quad (5.1)$$

Alternatively, one may also consider the change in the exergy of the HTF as it passes through the heat exchanger. This is the rate of exergy that the HTF releases when in thermal contact with either the R134a (superheat hybrid, **Figure 5.4**) or the brine (flash-binary hybrid, **Figure 5.5**), and is given by:

$$\Delta\dot{E}_{HTF} = \dot{m}_{HTF}[h_{1,HTF} - h_{2,HTF} - T_0(s_{1,HTF} - s_{2,HTF})] . \quad (5.2)$$

Since exergy is not a conserved quantity, a fraction of the exergy released by the HTF will be destroyed by various irreversibilities within the heat exchanger, and thus not be transferred to the R134a or the brine.

There are several possible definitions of utilization efficiency that may be appropriate for our task. The first one is for a basic geothermal power plant, namely:

$$\eta_{u,GEO} = \frac{\dot{W}_{net}}{\dot{E}_{GEO}}. \tag{5.3}$$

For a hybrid plant in which there are two distinct forms of exergy input, the most meaningful definition is:

$$\eta_{u,HYB} = \frac{\dot{W}_{net,HYB}}{\dot{E}_{GEO}+\dot{E}_{HTF}}. \quad \text{[Utilization efficiency 1]} \tag{5.4}$$

However, since not all solar exergy absorbed by the HTF in the collectors is actually transferred to the power system (some will be stored in the thermal storage system and other exergy will be recirculated within the solar loop), an alternative utilization efficiency that better measures the performance of the plant is based on the amount of exergy transferred from the solar HTF to the power system. This utilization efficiency measures the ratio of net work from the hybrid plant to the exergy of the incoming brine plus the change in exergy of the HTF, namely:

$$\eta_{u,HYB} = \frac{\dot{W}_{net,HYB}}{\dot{E}_{GEO}+\Delta\dot{E}_{HTF}}. \quad \text{[Utilization efficiency 2]} \tag{5.5}$$

In addition, one may also calculate a utilization efficiency measured by the ratio of net work from the hybrid plant to the change in exergy of the brine plus the change in exergy of the HTF, namely: criterion for comparing the thermodynamic performance is the utilization efficiency, η_u. This factor is based on the Second Law of thermodynamics and can be used for any power generating system, whether it operates cyclically or as a sequence of processes. It uses the exergy (or available work) as the basis for the performance assessment.

In its general form as applied to a simple system operating under steady, open conditions, the rate of exergy of a fluid at a given state 1 is given by:

$$\dot{E}_1 = \dot{m}[h_1 - h_0 - T_0(s_1 - s_0)], \tag{5.6}$$

where $\dot{m}$ is the mass flow rate, h is the enthalpy, s is the entropy, and T_0 is the dead-state temperature. This equation may be applied directly in this form to calculate the exergy of the brine as it enters the power plant, relative to the ambient conditions.

One may also consider the change in exergy of the brine between inlet and outlet conditions. In this way, the re-injection temperature of the brine can be accounted.

$$\dot{E}_{GEO} = \dot{m}_{GEO}[h_{IN,GEO} - h_{OUT,GEO} - T_0(s_{IN,GEO} - s_{OUT,GEO})]. \tag{5.7}$$

The calculation of the exergy of the solar energy input is not as direct. Ideally one would like to calculate the exergy associated with the electromagnetic solar radiation, but this is a controversial

$$\eta_{u,HYB} = \frac{\dot{W}_{net,HYB}}{\Delta\dot{E}_{GEO}+\Delta\dot{E}_{HTF}}. \quad \text{[Utilization efficiency 3]} \tag{5.8}$$

The last of the efficiency equations is the familiar thermal efficiency, employed for various power cycles:

$$\eta_{th,HYB} = \frac{\dot{W}_{net,HYB}}{\dot{Q}_{WF}}. \tag{5.9}$$

In this case the denominator is the heat delivered to the cycle working fluid. Whereas eq. (5.9) can be applied to the superheat system in a straightforward manner, if it is used with the flash-binary system it will yield erroneous results because the flash-portion of the plant renders the system non-cyclical. Thus, only thermal efficiency values for the superheat hybrid system are shown.

These measures are combined with the cost of the additional systems to arrive at a thermodynamic-economic optimum design. Thus, the capital expense (CapEx) needed to generate a kilowatt of rated or installed power can be used as a criterion that combines thermodynamic performance and economics. Also, the levelized cost to generate a kilowatt-hour of electricity is an equally, if not more important economic factor.

5.3 Solar Field Design

5.3.1 Preliminary Solar Model

In Phase 1 of this study, a variety of solar collector types - parabolic trough, power towers, dish, linear Fresnel, and flat plate collectors - were evaluated based on four main criteria: cost per unit area, efficiency, durability, and operation and maintenance. Parabolic trough collectors were found to have the best performance per unit cost in today's market. A fixed solar field efficiency of 70% and cost of $\$250/m^2$ were agreed to be used for the cost analysis. These existing estimates were based on studies conducted several years ago by Sargent & Lundy LLC and SunLab.

For Phase 2 of this study, these assumptions were re-examined. NREL's Solar Advisor Mode (SAM) was used to model thermodynamic heat losses and parasitic loads in the solar field. Costs were collected from the most recent NREL model as well as available quotes from suppliers. And finally, optical efficiencies of the solar mirrors were modeled. These new assumptions, detailed below, were used to get a more accurate representation of the solar power cycle.

Storage tanks were modeled in steady state and dynamic operation to simulate their function as the solar resource varied. A detailed description of the storage dispatch strategy is described in **Section 5.3.3**. In the thermodynamic models, the storage system was modeled as a two-tank direct thermal liquid storage. This was chosen because it was simpler to model, but still accurately represented the thermodynamics of storage. In real applications, and in the economic model, a two-tank indirect solar-salt system was priced to give accurate representation of all costs. The next section will highlight all assumptions in the thermodynamic analysis, and **Section 5.6.5** will explain all economic assumptions in detail.

5.3.2 Modeling Assumptions for Steady-State and Dynamic Analysis

This project departs from the earlier study with the addition of solar energy to the cycle. For Plant B, the following solar constraints were chosen for the initial analysis:

- The solar energy will come from parabolic trough collectors.
- Solar heat transfer fluid (HTF) is Therminol-VP1.
- Maximum solar HTF temperature: 390°C.
- Solar loop pressure: 2 bar.
- Solar loop flow rate: 105 kg/s.
- Solar thermal heat storage method: Two-tank liquid storage using Therminol-VP1.

These constraints are based on research from Phase 1 of this project and proven technology in the solar thermal field. The exact size of the solar field is a design variable, which will be examined further in the steady-state analysis and is dependent on how much solar energy is requested by the brine and how much storage is needed (e.g., heating up the brine to a hotter temperature and having more energy left over to store requires a certain level of solar heat input, which in turn determines the required solar field area).

For the dynamic analysis, solar field thermodynamic parasitic penalties were included based on rules-of-thumb given in NREL's SAM. These were only accounted for when the solar cycle was active. These are:

- Power for collector drives and electronics = 2.66E-07 MWe/m^2, where m^2 represents the solar field area;
- Power for HTF recirculation pump = 1.052E-05 MWe/m^2;
- Power for thermal energy storage (TES) pumps = 0.02 MWe/MWt,g, where the denominator is the design gross turbine output.

In addition, thermodynamic penalties were included based on empirical formulas from NREL equipment tests and their Solar Advisor Model. These equations are all based on the Schott PTR70 heat collection element (HCE). They are:

- Optical efficiency of mirrors and HCE total = 75.4%;
- HCE heat loss along length (W/m) = 0.141*T_abs + 6.48E-09*T_abs^4, where T_abs is the average absolute temperature of the HTF in the HCE;
- Piping heat loss (W/m^2) = 10*(0.001693*ΔT – 1.683E-5*ΔT^2 + 6.78E-8*ΔT^3), where ΔT is T_abs – T_ambient;
- TES heat loss (kWt):
    - 150 kWt for 3 hours of storage
    - 200 kWt for 6 hours of storage.

5.3.3 Power Cycle and Storage Approach

There is a distinction between the steady state and dynamic solar model. In the steady state model, the solar HTF is either delivering "full power" to the brine, or delivering no power. Thus, there is no reason to model solar storage in a static sense. Also, it becomes difficult to measure parasitic power and thermal loses since they depend on the change of temperatures in the solar field and the

area of the solar field (which is also a function of storage time). However in the dynamic model, these parameters can be readily measured, and they depend on how you operate the solar field.

A logic strategy is required for the interaction between the storage tanks and the solar heat delivered to the brine. This is also known as the storage dispatch strategy. For this study, the following logic was derived from previous studies.

- If no sun is available and no heat is stored in the solar tank, bypass the solar field.
- If sun is available, but not enough to meet the desired heat input to the brine, direct all HTF to the brine.
- If sun is available and is more than enough to meet the desired heat input to the brine, store excess heat in the storage tanks.
- If the sun is not available or below the desired level and storage is available, dispatch additional heat from the tanks sufficient to meet the desired heat input to the brine.

The logical controllers that do this are explained further in **Section 5.5**.

5.3.4 Solar Collector Industry Review

For a complete list of solar parabolic troughs under construction or operational, NREL's website provides a good source: http://www.nrel.gov/csp/solarpaces/parabolic_trough.cfm. In this study, several solar suppliers were identified to obtain more current cost information, as shown in **Table 5.2**. Because of the sensitive and subjective nature of price quotes, these values are only provided for reference and were not used in the economic analysis. In fact, not all suppliers were willing to provide a quote given the large number of inquiries they had received recently and how few sales were made as a consequence.

Large solar companies sell the full package of solar components and offer engineering services to design the plant. Usually these companies have some proprietary technology (i.e., the collector structure) and work with suppliers for other subcomponents, i.e., the mirrors, controls, heat collection elements (or receivers), heat transfer fluid, pumps, and thermal energy storage system. The main suppliers for these other subcomponents are well known, such as Flabeg for mirrors (German) and Schott (German) and Solel (Israeli) for heat collection elements, however there are other smaller suppliers with interesting technology that should not be ignored. Competition among these suppliers will push their components to more efficient designs.

Table 5.2 Large Solar Thermal Companies.

Supplier	Abengoa Solar	Flagsol	Skyfuel	Solargenix
Parabolic Trough Components Sold	Whole storage system (collector structure and mirror are proprietary)	Whole storage system (collector structure is proprietary)	Whole storage system (collector structure and controls are proprietary)	Whole storage system (collector structure is proprietary)
Product(s)	Solucar TR, PT1, RMT	SKAL-ET, Heliotrough	Skytrough	SGX-1, DS-1

Status	All collectors in production.	SKAL-ET in production, Heliotrough available early 2010	Collectors are tested. First installation due in 2010	Commercially Available
Projects	Salana (280MW planned), Solnova 1-5 (50MW each), Helioenergy 1-2 (50MW each), 2 hybrid plants in North Africa, several smaller projects in U.S.	Andasol 1-3 (50MW each), El Kuraymat, SKAL-ET Demo Loop	Possible addition to Nevada Solar One	Nevada Solar One (64MW), Arizona Saguaro Project (1MW)
Quoted Price	n/a	n/a	Collector - $214.92/m^2 Other items - $152/m^2 Molten Salt TES - $95/kWht Includes delivery and installation	$4000-$6000 per kW all in cost
Contact	+34 913300669 www.abengoasolar.com abengoasolar@abengoa.com	FLAGSOL GmbH Agrippinawerft 22, 50678 Köln Tel.: +49 (221) 9259700 info@flagsol.com ---------- Head of Sales: Helmut Kern, E-Mail: Helmut.Kern@flagsol.de Tel: +49 (221) 925 970 73 Fax: +49 (221) 925 970 99 www.flagsol.com ---------- Subsidiary of Solar Millennium AG	Skyfuel.com Project Sales VP – Bill Felsher: william.felsher@skyfuel.com	Chicago, IL 3622 South Morgan St. S. Chicago, IL 60609 (773)-847-8333 solargenixchicago.com ---------- Alice Epstein Sales & Marketing aepstein@solargenix.com 773.847.8333 x201
Other Component Info	Proprietary, Rioglass solar mirrors	Schott and Solel receivers, Flabeg glass mirror	Schott PTR-80 Receiver, Reflectech mirror film	Schott PTR-70 and Solel UVAC receivers, Flabeg glass mirror
Storage Option (Y/N)	Optional. Planned for Salana in Arizona	Yes, Molten Salt	Yes, Molten Salt	No
Innovation	Proprietary collector assembly	SKAL-ET Collectors	Re-designed tracking system, Lightweight frame	collector design

5.4 Steady State Analysis

5.4.1 Flowsheet Modeling Assumptions

The design basis for the steady-state design case is defined by these parameters:

- Geothermal fluid mass flow: 100 kg/s
- Geothermal fluid temperature: 150°C
- Geothermal fluid pressure entering the plant: 20 bar
- Dead-state temperature (ambient air temperature): 20°C
- Turbine isentropic efficiency:
- 85% for fully-vapor expansions

- <85% when liquid is present (calculated from the Baumann rule)
- Turbine exit vapor quality ≥ 90%
- Mechanical & generator combined efficiency: 98%
- Pump efficiency: 80%
- Condenser subcooling: 2°C
- Main heat exchanger LMTD: > 5°C (HX2 for flash-binary cycle)
- Recuperator pinch temperature difference: 5°C
- Heat exchanger pressure drop: 0.2 bar
- Target R134a maximum temperature: 180°C
- Brine reinjection temperature: >70°C.

The incoming geothermal fluid state was fixed for all models. This allowed the model to focus on varying the solar heat added to the brine. Part of the design basis for Plant B examined the temperature of the brine returning to the reservoir. If it is below 70°C, the brine would be at risk of depositing minerals in the pipes or the formation. Thus, care was taken to ensure return brine temperature would exceed 70°C.

In addition, several rules defining optimum design parameters were taken from the supercritical geothermal binary cycle model in Tester et. al. [54] They are:

- The air flow rate through the air-cooled condenser (ACC) was set such that the air temperature rise is exactly half of the temperature difference between the working fluid condensing temperature and the ambient air temperature.
- The specific parasitic power to run the fans in the air-cooled condenser is very nearly constant and equal to 0.25 kW per kg/s of air flow through the ACC.
- The optimum configuration for the binary turbine expansion path was selected using the overall plant utilization efficiency as the criterion. The path was examined to determine that the working fluid did not pass through the critical point, did not cross the saturated liquid line, and that the exit vapor quality was at least 90% to minimize erosion on blades/vanes

5.4.2 Flash-Binary Hybrid Model

Steady State Design

The Aspen Plus model flowsheets were built using the supercritical R134a binary model from Project 1 as a base geothermal template. For the flash-binary hybrid model, a flash vessel, steam cycle, and solar cycle were added, as illustrated in **Figure 5.7**. In addition, a bypass line was added to model the plant operating when sun and solar storage are unavailable.

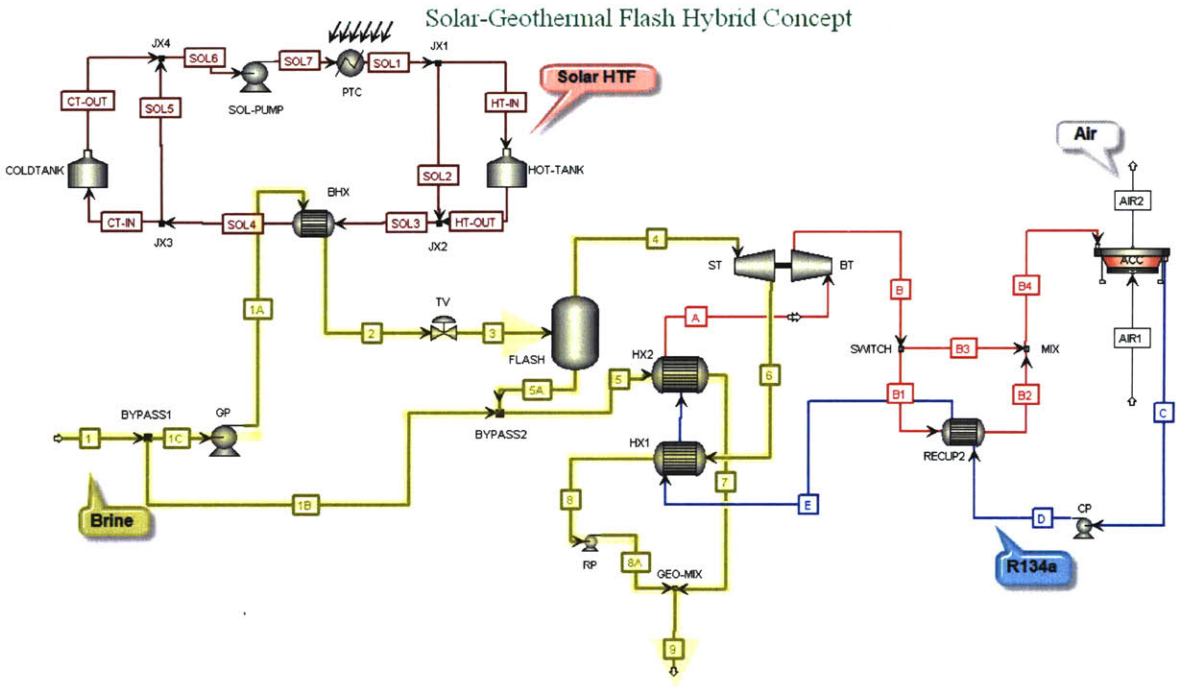


Figure 5.7 Solar-geothermal flash-binary hybrid Aspen flowsheet.

There are two main design variables in this model: the heat added to the brine by the solar HTF from state 1A to 2, and the flashing pressure determined by the throttle valve between states 2 and 3. The first variable is critical because it determines how much solar area is needed (if storage is needed, the area can be increased), and the second variable determines the balance of power between the steam cycle and the binary cycle. Both these design variables were looked at in detail to determine a best-case scenario for the flash-binary hybrid design and their results will be examined shortly.

There are a few other parameters that must be determined each time the model runs. These are the turbine pressure drops (both steam and organic), the WF supercritical pressure, and the WF mass flow rate. The strategy employed to determine these parameters was similar to that of Project 1 as explained in the previous section. That is, certain assumptions were made by design, and then Aspen Plus was programmed to optimize the remained parameters to find the combination where utilization efficiency was highest. By doing this, the analysis could focus on the two main design variables in the model.

Looking at the brine in state 1, it is pumped to a higher pressure before entering the brine-solar heat exchanger to prevent the brine from boiling. This is possible because the solar HTF was heated to 390°C to model a typical parabolic trough collector. This pressure was varied based on the designed temperature increase of the brine.

From there, the brine is throttled down to obtain a vapor/liquid mix in stream 3. The higher the flash pressure, the hotter stream 3 will be, and consequently more heat can be exchanged with the binary WF. Because the WF is R134a with an absolute upper temperature of 200°C, and a design

target temperature of 180°C, the flash pressure cannot be too high or the R134a will be in danger of chemical decomposition.

From the flash tank, a liquid brine stream and a steam stream emerge. The liquid brine transfers heat to the binary WF (HX2) and then is reinjected. The steam powers the back-pressure turbine and then is condensed in a heat exchanger (HX1) against the binary WF before mixing with the liquid brine. The steam condensing pressure in HX1 sets the lower pressure limit for the steam leaving the turbine.

The binary cycle operates similar to the supercritical geothermal binary plant in Project 1, however now there are two heat exchangers interfacing with the WF. This allows the mass flow rate of the WF to vary greatly based on how much heat is available from the sun. Looking forward to the dynamic model, this range of flow rates will impact the power output of the binary turbine through its performance curve, but in the steady state model, the turbines are modeled as the ideal size for the flow rate through them.

In the solar cycle, there are two storage tanks, a pump, and a heating element. The storage tanks are not useful in the steady state model, but they are represented here for conceptual understanding. The heating element provides the heat necessary to raise the temperature of the brine as needed and in this model is an abstraction of the concentrating solar collectors. This is sufficient for steady state analysis, but will be refined in dynamic analysis.

Steady State Results

For the first set of sensitivity studies conducted on the flash-binary hybrid model, the brine was heated to 275°C by the solar heat transfer fluid, and then the brine was flashed to a range of pressures between 15 and 22 bar. **Figure 5.8** shows the results of this study. As the flash pressure increased, so did the net power and the temperature of the R134a. Because R134a cannot go far above 180°C or it will decompose, the flash pressure must be limited to no higher than 15 bar.

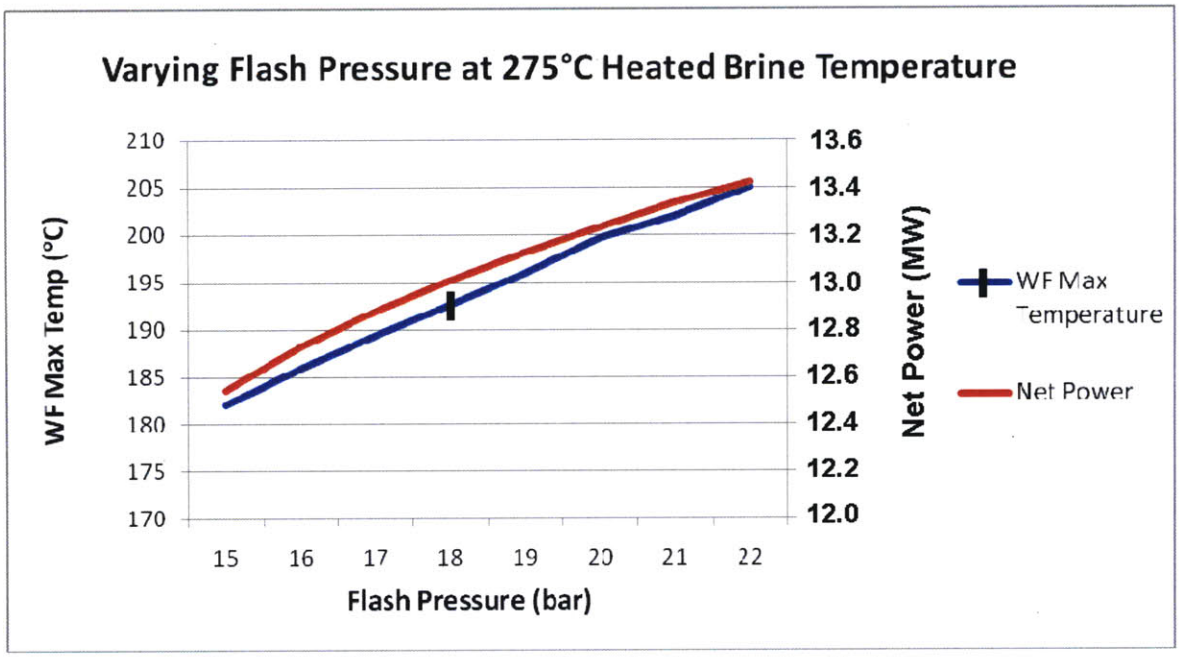


Figure 5.8 Flash-binary hybrid results from varying flash pressure.

Table 5.3 shows the complete results of other parameters for each of the runs in the sensitivity study. Utilization efficiencies 1, 2 and 3 represent equations 6, 7 and 8, respectively. In each of these cases, the steam turbine back pressure was 3 bar, as low as possible, and the binary turbine exit pressure was between 8-9 bar. Notice that all efficiencies, as well as the binary turbine gross power, increase as the flash pressure increases. However, the steam turbine goes through a maximum around 16 bar. This is because the higher flash pressures produce less steam, and the lower flash pressures produce steam at lower temperatures.

Table 5.3 Flash-binary hybrid results from varying flash pressure.

Flash Press	Steam Turb Power	Binary Turb Power	Parasitic Power	Net Power	Util. Eff. 1	Util. Eff. 2	Util. Eff. 3	Reinj Temp	WF Max Temp	WF SC Press	WF Mass Flow Rate	Air Mass Flow Rate
(bar)	(MW)	(MW)	(MW)	**(MW)**	(%)	(%)	(%)	(°C)	**(°C)**	(bar)	(kg/s)	(kg/s)
15	4.34	11.70	3.50	**12.55**	12.6	31.1	36.4	129	**182**	49.1	285	6820
16	4.36	11.96	3.60	**12.73**	12.7	31.6	36.8	127	**186**	48.0	285	7319
17	4.36	12.09	3.58	**12.88**	12.9	32.0	37.3	128	**189**	48.9	283	7161
18	4.35	12.20	3.55	**13.00**	13.0	32.3	37.6	127	**193**	48.4	284	7087
19	4.33	12.49	3.69	**13.12**	13.1	32.6	37.7	126	**196**	46.9	285	7804
20	4.29	12.77	3.83	**13.23**	13.2	32.8	38.4	129	**200**	50.8	275	8071
21	4.25	12.69	3.61	**13.33**	13.3	33.1	38.8	129	**202**	51.6	276	7070
22	4.20	13.05	3.82	**13.42**	13.4	33.3	38.7	127	**205**	49.2	279	8149

For the second sensitivity study, the flashing pressure was held constant at 15 bar while the heated brine temperature was varied (from 150°C, or no solar energy, to 325°C). Illustrated in **Figure 5.9**

is the varying net power and mass flow rate of the binary working fluid as the brine temperature changes. In these runs, the net power increased as a consequence of greater mass flow moving through the binary turbine. However, as stated in **Section 5.2.2**, the design turn down ratio for the binary turbine is 2:1. Therefore, the brine should not be heated above 275°C, which results in a WF mass flow rate of near 300 kg/s, or two times the non-solar mass flow rate.

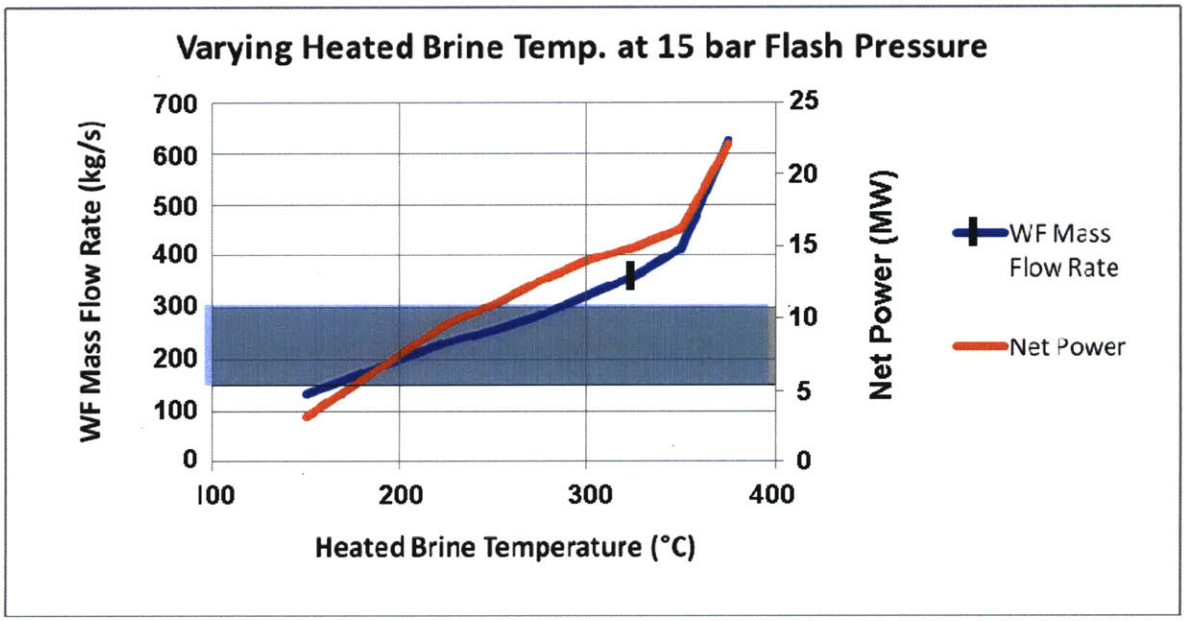


Figure 5.9 Flash-binary hybrid results from varying heated brine temperature.

Table 5.4 shows more parameters from the second sensitivity study. In these cases, the ST back pressure was set as low as possible while still condensing in HX1, and the binary turbine exit pressure was between 8-9 bar. The utilization efficiency 2 and 3, which take into account the change in exergy of the solar HTF, appear to be greatest as the brine temperature approaches its lowest value. In all runs, the reinjection temperature is kept above 70°C to prevent cooling the brine too much.

The apparent discontinuity in the results for utilization efficiency 1 between a brine temperature of 150°C and 225°C is caused by a shift in the definition from eq. (5.5) for the un-solar-heated brine (150°C) to eq. (5.6) for the solar-heated brine (225°C). Since the 150°C case is not a hybrid system, the definitions of utilization efficiencies 2 and 3 are not applicable.

Table 5.4 Flash-binary hybrid results from varying heated brine temperature.

Heated Brine Temp	**Steam Turb Power**	**Binary Turb Power**	**Parasitic Power**	**Net Power**	**Util. Eff. 1**	**Util. Eff. 2**	**Util. Eff. 3**	**Reinj Temp**	**WF Max Temp**	**WF SC Pressure**	**ST Back Press**	**WF Mass Flow Rate**	**Air Mass Flow Rate**
(°C)	(MW)	(MW)	(MW)	**(MW)**	(%)	(%)	(%)	(°C)	(°C)	(bar)	(bar)	**(kg/s)**	(kg/s)
150	0	4.68	1.44	**3.25**	34.0	n/a	n/a	82	149	44.7	n/a	**134**	3702
225	2.25	10.90	3.63	**9.53**	9.5	34.5	38.4	99	179	81.9	1	**233**	6914
250	3.14	10.92	3.15	**10.91**	10.9	32.2	36.8	115	190	45.1	3	**256**	7434
275	4.34	11.70	3.50	**12.55**	12.6	31.1	36.4	129	182	49.1	3	**285**	6820
300	5.43	12.88	4.32	**13.99**	14.0	29.8	35.0	138	178	44.9	4	**323**	8510
325	5.47	14.82	5.38	**14.91**	14.9	27.6	33.0	152	176	50.4	5	**360**	9447
350	5.42	17.93	7.22	**16.13**	16.1	26.0	31.1	163	175	55.2	7	**414**	12531
375	3.03	28.86	9.75	**22.14**	22.2	26.3	31.5	187	186	69.9	12	**626**	15105

5.4.3 Superheat Hybrid Model

Steady State Design

The superheat steady state model is shown in **Figure 5.10**. In this design, the solar HTF directly heats up the binary WF after the brine. Given that the R134a cannot exceed 180°C, the mass flow rate must be increased to accommodate the addition heat from the sun.

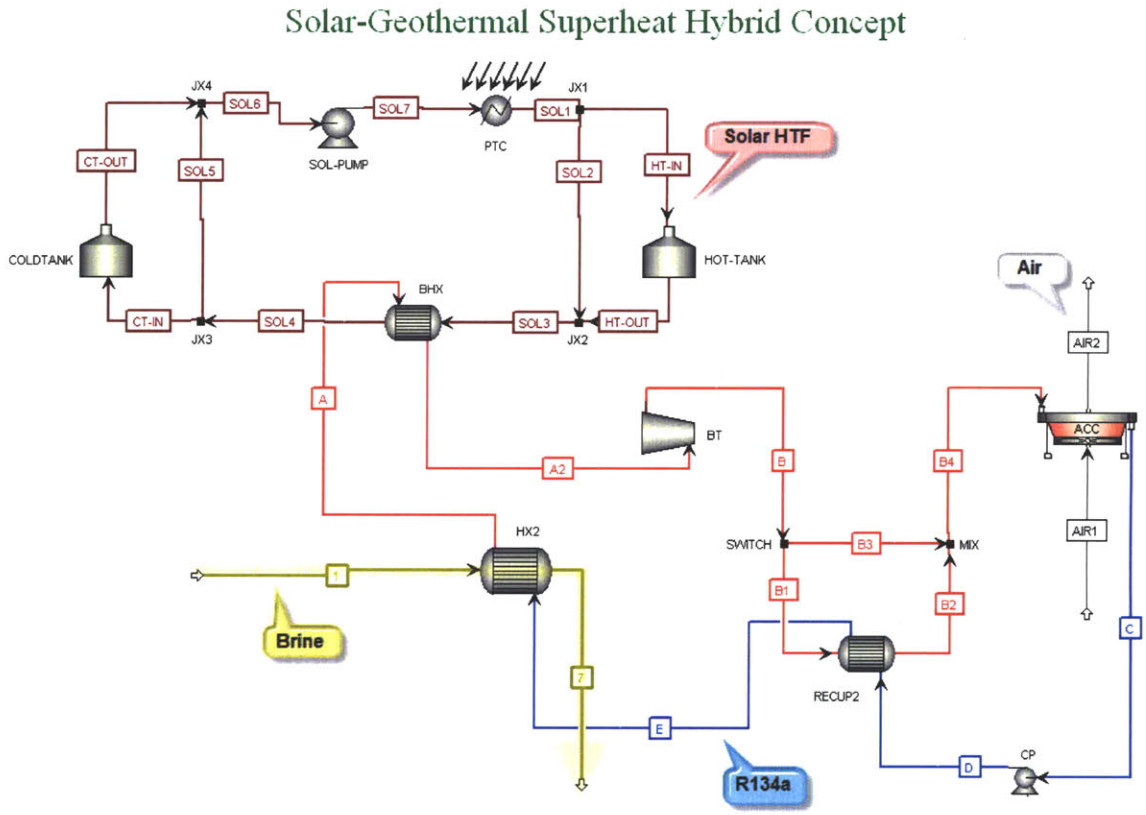


Figure 5.10 Solar-geothermal Superheat hybrid Aspen Flowsheet.

In this study, the amount of solar heat was varied, while maintaining a maximum WF temperature of 180°C. As with the flash-binary model, several internal parameters were calculated by the Aspen Plus optimization engine. These were the WF supercritical pressure, the WF mass flow rate, and the binary turbine exit pressure. All other parameters were set the same as in the flash-binary model.

Steady State Results

A sensitivity study was conducted on the superheat hybrid model to examine how it would behave with varying amounts of solar heat added. As expected and shown in **Figure 5.11**, the relationship is linear. As the solar heat increases, so does the net power, albeit at a higher parasitic load cost. Similarly, the mass flow rate of the WF increases linearly. Given the turn down ratio design of the binary turbine, this model is limited in total power it can generate with one turbine.

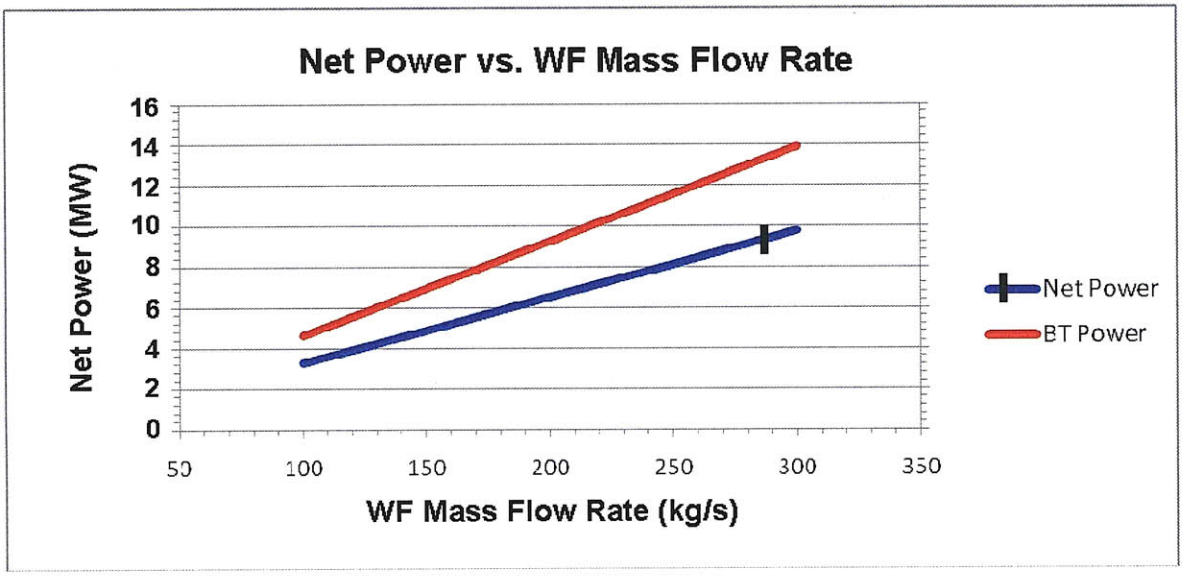


Figure 5.11 Superheat hybrid results from varying mass flow rate.

The table of results (**Table 5.5**) shows that the utilization efficiencies 2 and 3 go through a maximum at lower levels of solar heat. This is similar to the results for the flash-binary model, however, the optimum supercritical pressure is about 10 bar higher.

Table 5.5 Superheat hybrid results from varying mass flow rate.

BT Power	Parasitic Power	Net Power	Therm. Eff.	Util. Eff. 1	Util. Eff. 2	Util. Eff. 3	Reinj. Temp.	WF Super-critical Pressure	WF Mass Flow Rate	Air Mass Flow Rate
(MW)	(MW)	(MW)	(%)	(%)	(%)	(%)	(°C)	(bar)	(kg/s)	(kg/s)
4.60	**1.36**	3.25	14.4	3.2	26.0	42.0	109	67.8	**100**	2931
6.92	**2.05**	4.87	14.4	4.9	34.8	44.0	89	67.9	**150**	4450
9.22	**2.73**	6.49	14.4	6.5	36.8	41.7	77	68.1	**200**	5903
11.52	**3.41**	8.11	14.4	8.1	33.8	37.0	78	68.3	**250**	7340
13.84	**4.11**	9.74	14.4	9.7	32.1	34.6	78	68.2	**300**	8891

5.4.4 Flash-Binary Hybrid vs. Superheat Hybrid

After each steady state model was studied independently, the results were compared to determine which model better utilized the sun's energy. In **Figure 5.12**, both models were examined to see how much net power was produced for a given amount of solar heat added to the brine. The dotted line indicates the thermal energy (heat) of the incoming brine relative to a 20°C ambient temperature. As the amount of solar heat captured is increased, both models initially show the same rise in net power, and at even higher amounts of sun, the superheat model slightly outperforms the flash-binary model. However, this is a snapshot of maximum power at steady state and does not incorporate the dynamic performance of the turbines under a widening range of states.

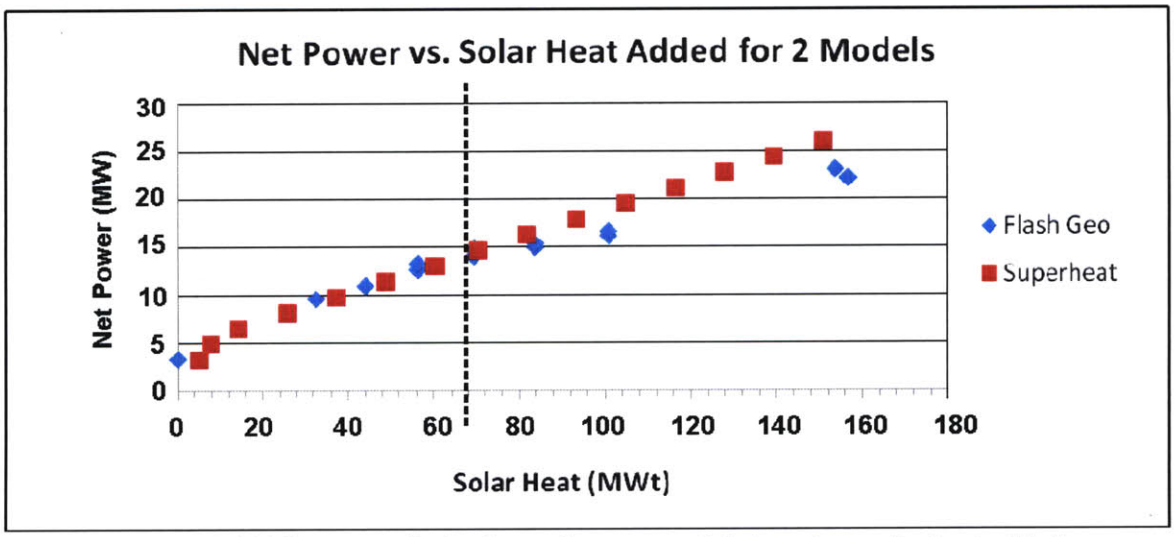


Figure 5.12 Net power for both steady state models based on solar heat added.

Consequently, the maximum WF mass flow rate was also examined per amount of solar heat added. **Figure 5.13** shows that at higher levels of sun, the flash model does a better job at keeping flow rates lower. Given the turbine constraint of a 2:1 turn down ratio, this means the flash model will realize higher net power. In the flash model run with a mass flow rate of approximately 285 kg/s, the net power was 12.55 MW, while in the superheat model with a mass flow rate of 300 kg/s, the net power was 9.74 MW. That is about a 30% difference in performance.

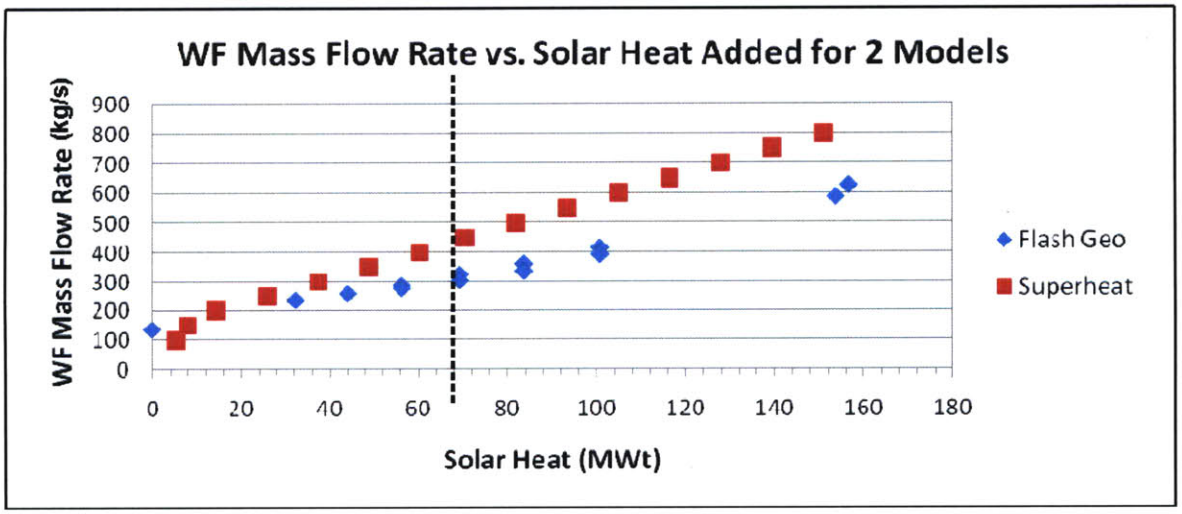


Figure 5.13 WF mass flow rates for both steady state models based on solar heat added.

Given the higher potential of the flash-binary model, it will be examined further in a dynamic setting. This dynamic model will determine how many kilowatt hours can be produced in a typical day that, when combined with economic costs, will determine the cost effectiveness of the model.

5.5 Dynamic Analysis

In this study, Aspen Dynamics v7.1 was used to perform all dynamic analyses. The flash hybrid model was imported into Aspen Dynamics to create a basis upon which refinements were made to accurately represent the model in a dynamic setting. These refinements are:

- Specifying which parameters will remain fixed and which are allowed to vary, e.g., specifying the pump inlet pressure to remain constant while freeing the pump power to vary;
- Specifying dynamic parameters for some objects, e.g., the heat loss in the solar collectors or the performance curve of the binary turbine;
- Programming controllers to run the plant automatically; and
- Inserting forcing functions in the model to simulate the position of the sun and air temperature.

This section will describe how the model was built, what assumptions were made, and show results of two typical days with different amounts of solar storage.

5.5.1 Dynamic Modeling Assumptions

The dynamic model was constrained by the following parameters. These parameters are based on the flash hybrid steady state model.

- Geothermal fluid mass flow: 100 kg/s
- Geothermal fluid temperature: 150°C
- Geothermal fluid starting pressure: 20 bar
- Steam Turbine isentropic efficiency:
- 85% for fully-vapor expansions
- <85% when liquid is present (calculated from the Baumann rule)
- Binary Turbine isentropic efficiency: **Figure 5.14**
- Turbine exit vapor quality ≥ 90%
- Mechanical & generator combined efficiency: 98%
- Pump efficiency: 80%
- Condenser subcooling: 2°C
- Heat exchanger area: Set by detailed heat exchanger model in steady state (**Appendix D**).
- Heat exchanger pressure drop: Calculated
- Power Law Curve for U value used for HX1 and HX2: U=U_ref*(Flow/Flow_ref)^0.6
- Liquid level in flash tank: 3m (total height: 5m, diameter: 1.8m)
- Flashing pressure: 15 bar
- Steam Turbine back pressure: 3 bar
- Design target heated brine temperature: 275°C

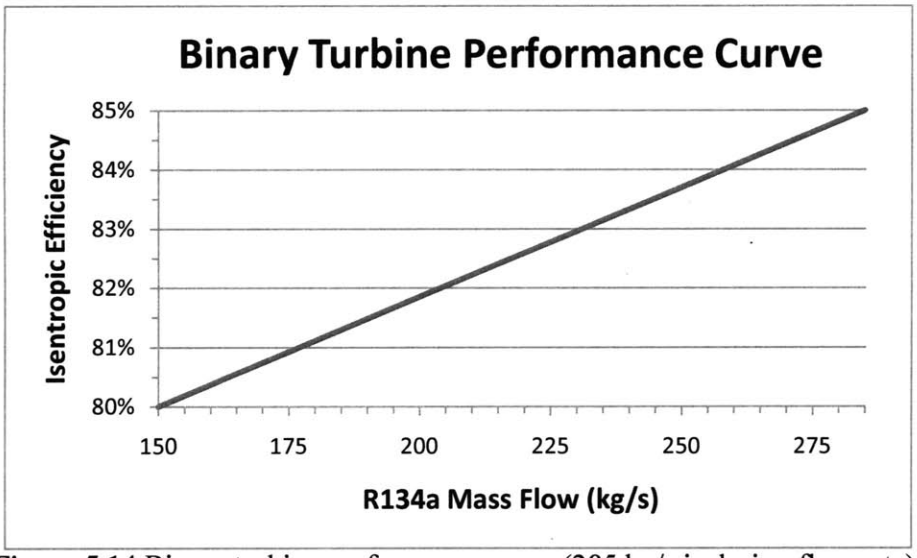


Figure 5.14 Binary turbine performance curve (285 kg/s is design flow rate).

(Source: Ken Nichols, personal communication)

5.5.2 Operation Strategy

The dynamic model flowsheet was derived from the flash hybrid steady state model. After initial testing and setup, the flowsheet was modified to improve both system performance and accuracy. This updated flowsheet can be seen in **Figure 5.15**. All the dotted blue lines represent control signals. The following were major changes made from the steady state to the dynamic version of the flash hybrid model:

1. Controllers were added to regulate key variables. Their target values are based on results from the steady state model.
2. A pressure relief vessel was added to the solar loop so the incompressible HTF could thermally expand as needed.
3. The solar heating element was replaced with a heated pipe model so that the model could represent a temperature gradient in the solar collectors.
4. Heat loss and parasitic functions were added to the solar cycle, as explained in **Section 5.3.2**. This was done in the flowsheet constraints, which can be viewed in **Appendix B**.

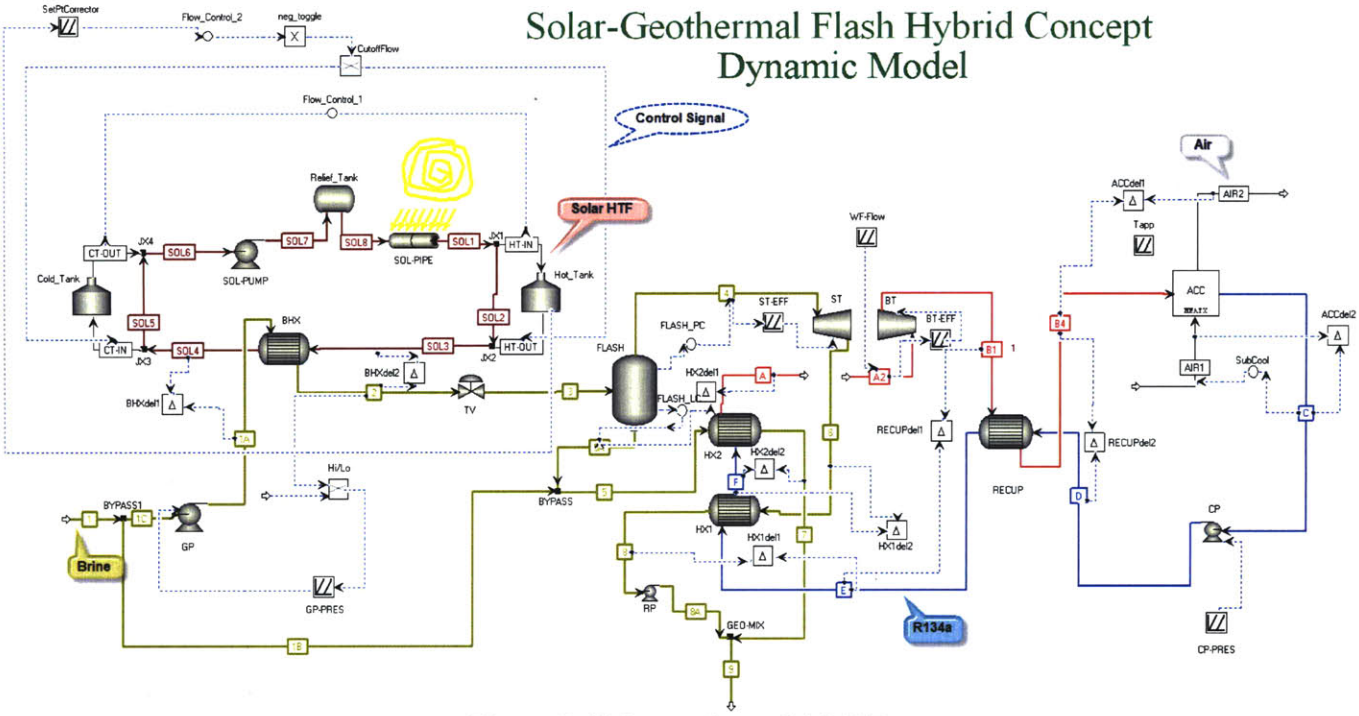


Figure 5.15 Dynamic model P&ID.

The dynamic model operation strategy is governed by five types of variables: fixed variables, free variables, simulated variables (forcing functions), design variables, and controlled variables. Fixed variables are set based on flowsheet assumptions, such as heat exchanger size. These variables perfectly constrain the model so it can solve for all the free variables that change in time.

The other three types of variables are actually special kinds of fixed variables. Simulated variables, or forcing functions, are based on data points and are fed to the model over time. These are the solar insolation and ambient air temperature. Design variables are variables set up for each run. These variables are the solar field area and size of solar energy storage. In fact, these two design variables are coupled by the fact that the design target heated brine temperature is fixed. This means that a larger solar field area will correlate with a larger energy storage system. Controlled variables are fixed variables that are governed by controllers in the model. These are the brine ground pump pressure, WF mass flow rate, flash tank liquid level, steam mass flow rate, binary turbine outlet pressure, condenser fan speed, WF supercritical pressure, and the amount of heat going to solar storage.

5.5.3 Control Strategy

Three types of controllers were used in the dynamic model – a PID controller, a Split Range controller, and an On/Off controller. The PID controller works by measuring a signal (input), or process variable, and modifying another signal (output) such as to drive the input signal to a specified setpoint. For example, a PID controller is used to regulate the liquid level in the flash tank. The measured signal is the liquid level and the output signal is the flow rate of liquid from the tank. If the liquid level goes too low, the controller will slow down the output flow rate, and if

the liquid gets too high, the controller will speed up the output flow rate. A PID controller stands for proportional integral derivative controller because it can use all three methods together to reduce the error between the process variable and the setpoint. All PID controllers in this model were tuned manually to provide a reasonable measure of control.

The Split Range controller does not utilize a setpoint. Instead it only uses an input and an output signal. When the input signal reaches a certain range of values, the output signal will vary between a different set of values in a linear fashion. For example, when the brine becomes heated by the solar HTF, the GP-Pump will be commanded to increase its power so as to keep the brine sufficiently pressurized not to boil. The relationships are linear and tuned by each controller.

The last controller employed was a simple On/Off controller, which works as a toggle switch to send or block a signal. This controller was used to stop the storage tank from running dry when the solar storage was running low.

Table 5.6 lists all the important controllers. For each controller, the type indicates how it operates and the input and output shows how it works. All of the controllers work as feedback loops, with exception for the "WF-Flow", "CP-Pres", and "Tapp", which work as feed-forward controllers. For these feed-forward controllers, the "ideal" output value was preselected for a given input value upstream. This "ideal" value was based on parameters measured in steady state studies.

Table 5.6 Dynamic Controllers.

Controller	Type	Function	Measured	Manipulated
Flow_Control_1	PID	Control Solar Storage Inlet	Heat flow of hot Solar HTF	Mass flow rate of liquid to hot storage
Flow_Control_2	PID	Control Solar Storage Exit	Heat flow of Solar HTF downstream of storage tank	Mass flow rate of liquid out of hot storage
GP-Pres	Split-Range	Pressurize brine so that heat from Solar HX does not boil it	Temperature of heated brine	GP output pressure (brine pressure)
Flash_LC	PID	Keep flash tank liquid level constant	Flash tank liquid level	Flow of liquid brine from bottom of tank
Flash_PC	PID	Keep flash tank pressure constant	Flash tank pressure	Flow of steam from top of tank
WF-Flow	Split-Range	Control WF Flow Rate Based on amount of Solar heat available	Temperature of heated brine	WF Mass Flow Rate
CP-Pres	Split-Range	Control WF Supercritical pressure based on amount of solar heat available	Temperature of heated brine	WF supercritical pressure
SubCool	PID	Maintain ACC subcooling of 2°C	Temperature of condensed WF	Air mass flow rate through condenser
Tapp	Split-Range	Control BT Output pressure based on amount of solar heat available	Temperature of heated brine	Help determine temperature along R134a Saturation curve at which BT Output pressure is based

5.5.4 Analysis Strategy

The dynamic model was analyzed by picking the forcing functions and running several cases with different design variables. The forcing functions used were:

- Source: Nevada Power Clark Station (NPCS), Las Vegas, NV
- Dates used: January 15th, 2009 and July 15th, 2009 (typical January and July day)
- Input values: DNI and Ambient Air Temperature every 3 minutes

Two sets of cases were run; one for the January day and one for the July day. In each case, the same day was run twice in succession, and data was captured from 12 PM on day 1 to 12 PM on day 2 (12th hour to 36th hour). The purpose of repeating the same day twice was to ensure that the starting point for the 24 period of interest is consistent with the end state. Thus any arbitrary state values at time = 0 are eliminated. The follow three cases were run for each set:

- No solar heat added to brine. Brine is bypassed to binary cycle.
- A solar field of 130,000 m^2 is used to heat the brine with solar storage as necessary.
- A solar field of 170,000 m^2 is used to heat the brine with solar storage as necessary.

5.5.5 Results

This section shows key results from the dynamic model runs. Additional results are located in **Appendix D**.

Figure 5.16 shows the solar insolation of the typical January and July day from 12 PM to 12 PM. The January day appears to be very clear with no disturbances in insolation, whereas the July day experiences periods of cloud cover. In these two days, the sun comes out at about the same time because of early morning clouds on the July day, and reaches a nearly equivalent maximum insolation of 950 W/m^2. However, the July sun lasts for about two hours longer.

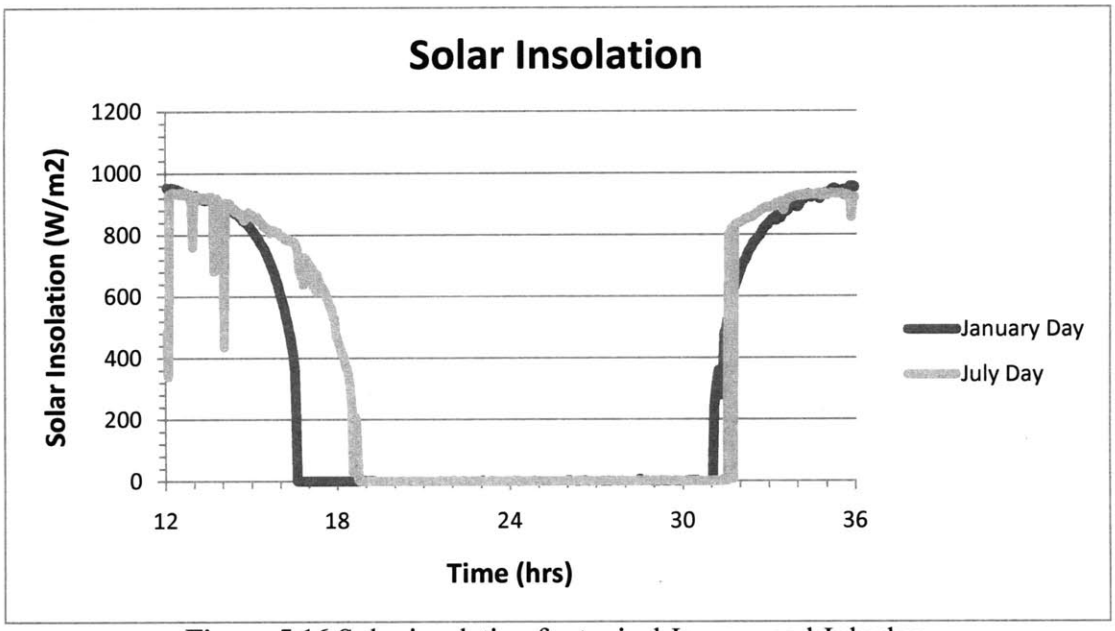


Figure 5.16 Solar insolation for typical January and July day.

Figure 5.17 shows the ambient air temperature of the typical January and July day. Each day follows a similar trend in air temperature change over the day, but the July air temperature consistently remains more than 20°C hotter than the January air temperature. The average January day temperature is 13°C and the average July day temperature is 40°C.

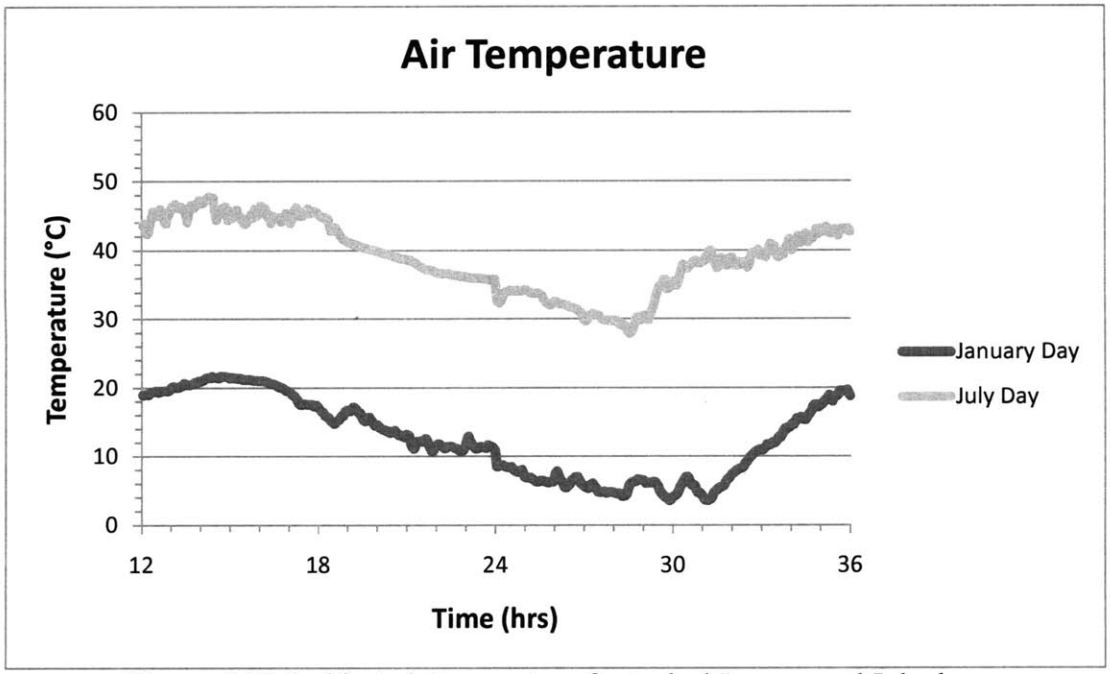


Figure 5.17 Ambient air temperature for typical January and July day.

Figures 5.18 and **5.19** show the net power produced by different model runs. There are several key points to note about these results.

- The January day produced more power than the July day due to the lower ambient air temperature, which required less condenser parasitics.
- The July day experienced longer storage times because of its longer sun period. For the 130K solar field, the January day had 3 hrs of storage while the July day had 3.5 hrs of storage. For the 170K solar field, the January day had 5 hrs of storage while the July day had 6.5 hrs of storage.
- The solar fields increased net power by 4-5x. When the sun is shining, the net power is limited by the 275C setpoint temperature for the brine. The variations in the net power are dictated mostly by the variations in the ambient temperature.
- Local spikes and dips at the beginning and end of the additional solar power is due to the controller logic in the model. During these transition periods, there are competing controllers and some times their oscilations cause overall plant power to oscilate.
- The dip at the 17th hour for **Figure 5.18** reflects the moment when solar insolation is too low to produce the desired level of heat for the brine and solar storage energy becomes engaged. The dip reflects the solar storage not releasing heat fast enough.
- For the July day, there are some times when the sun moves behind a cloud and solar storage is inadequate to cover these lapses. Hence, there are some blips in the net power and brine temperature (**Figure D.2**) during those times.

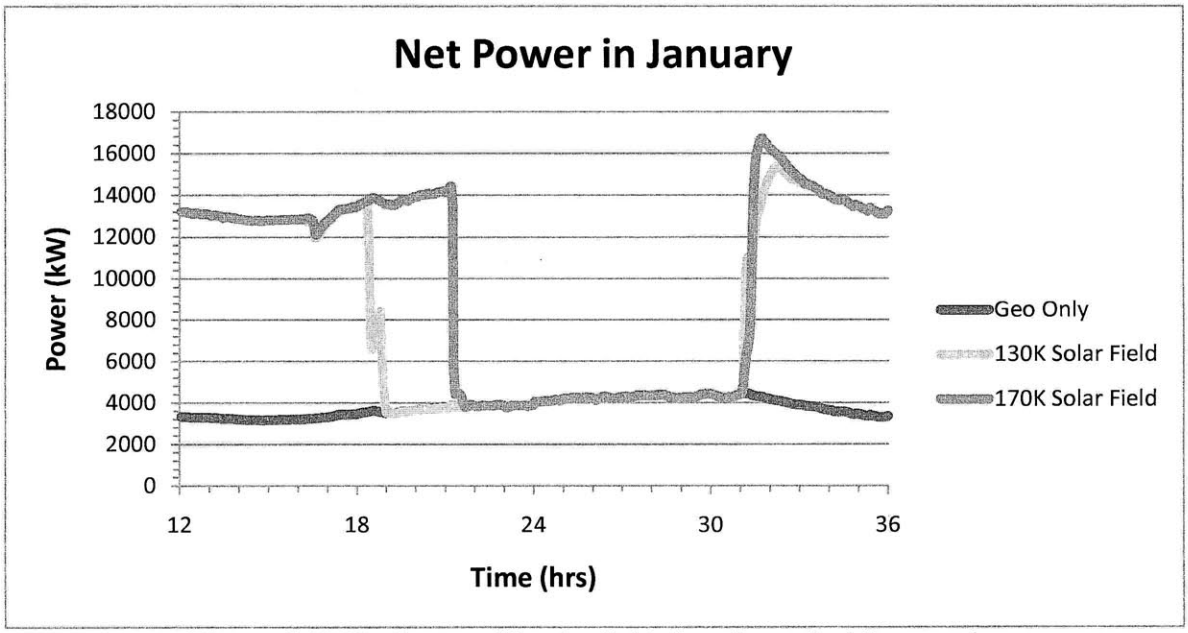


Figure 5.18 Net Power of 3 solar field sizes for typical January day.

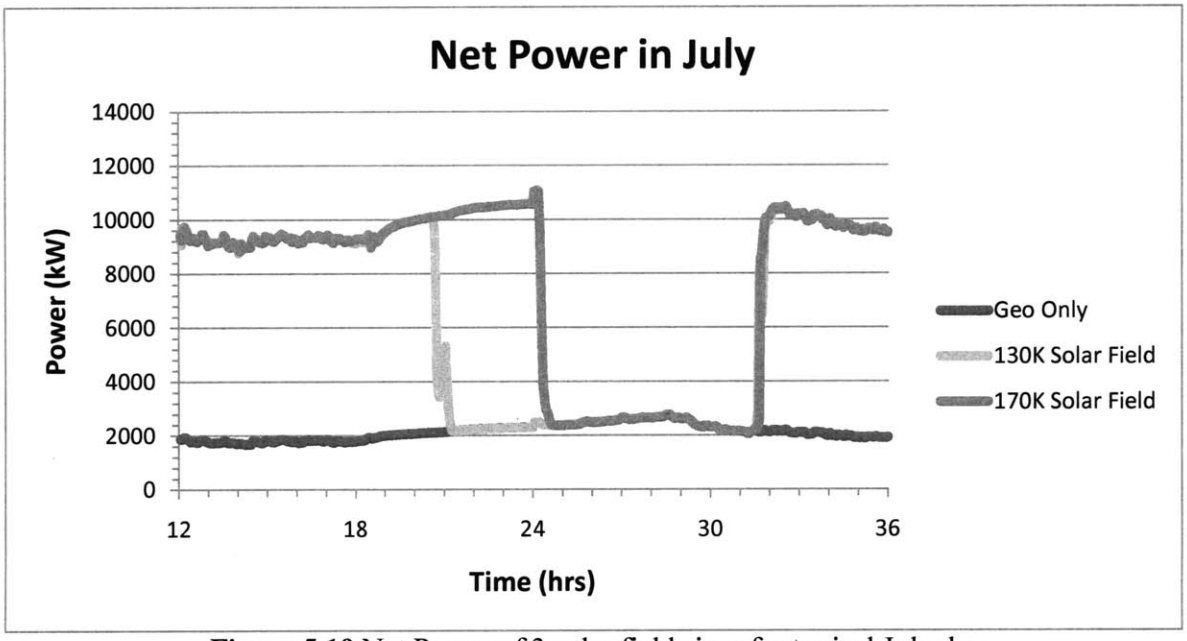


Figure 5.19 Net Power of 3 solar field sizes for typical July day.

A more detailed view of power can be seen in **Figure 5.20** for the case of a January day with a 130,000 m^2 solar field area. Notice that the net power is mostly influenced by the binary turbine, while the steam turbine is generally either at full capacity or off depending on solar heat availability. The parasitic load includes all pumps and fans in the model. Note how quickly the power drops when the solar storage tanks are exhausted.

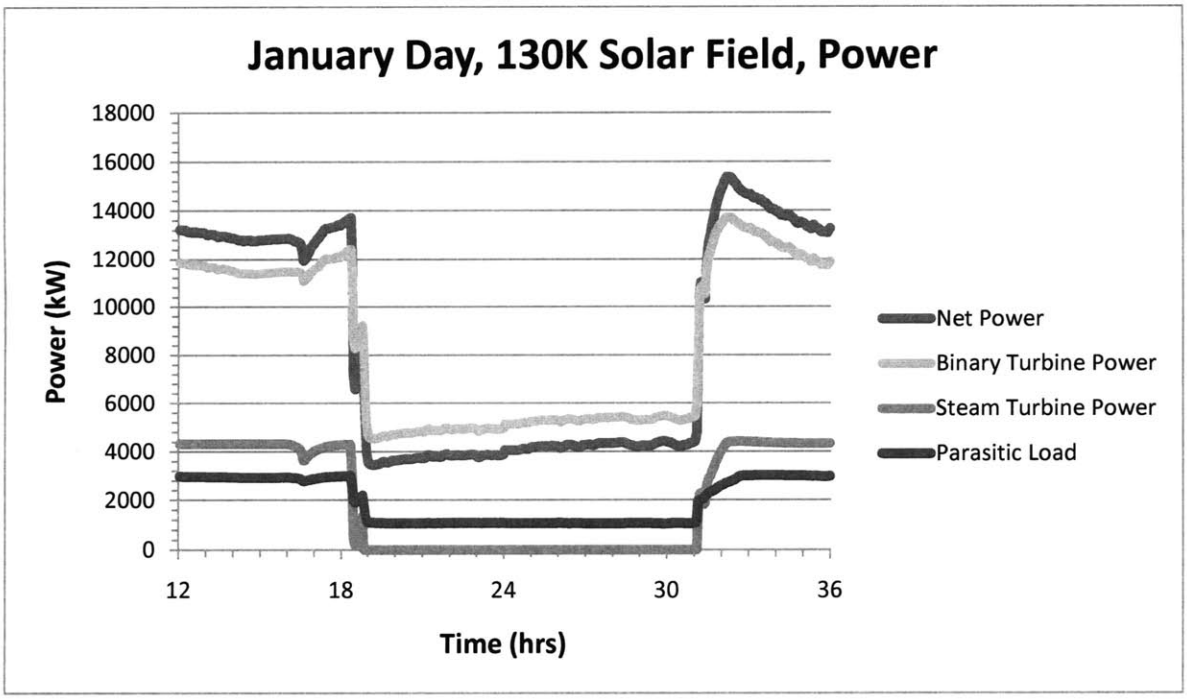


Figure 5.20 Detailed look at power produced in the 130,000 m^2 solar field case for January.

For each case, the total net power was summed up from the 12th to 36th hour to produce the results in **Figure 5.21**. The additional solar heat more than doubles the power produced in January and more than triples the power produced in July. In the next section, these scenarios are analyzed from an economic perspective to see how much their electricity costs per kilowatt-hour.

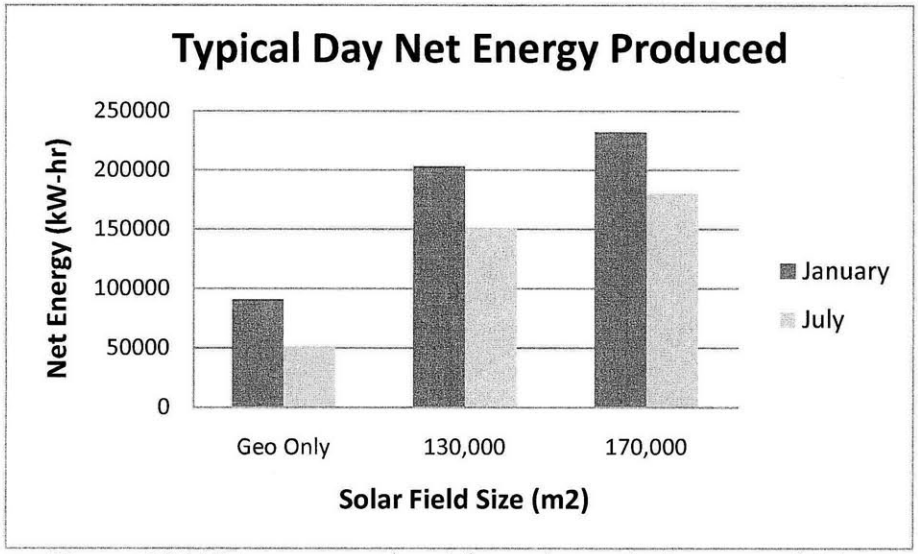


Figure 5.21 Net energy produced for each case in January and July.

5.6 Economic Assessment

5.6.1 Cost Modeling Assumptions

Cost modeling was conducted for all major plant components. Total installed cost was estimated using the best tools available. Where applicable, the plant costs were conducted in Aspen ICARUS and Aspen HTFS This was possible for all geothermal components, except the turbines and generator. For these components, supplier quotes coupled with experiential rules-of-thumb were used to predict total installed costs. For the solar components, costing numbers from NREL were assumed, with the price of solar collectors varied to measure its effect on levelized cost of electricity. Given this level of estimation, it is assumed that costs are within ± 40% of actual costs.

5.6.2 Geothermal System Cost Analysis

The major components of the geothermal system are the well costs, heat exchangers, air cooled condenser, flash tank, pumps, turbines, and generator. With the exception of the wells, turbines and the generator, plant components were sized and priced using the Aspen ICARUS and Aspen HTFS software. The turbines and generator were costed using rules-of-thumb based on supplier quotes and experiential knowledge.

The following assumptions were used for the turbines and generator:

- Binary Turbine: $215/kW direct cost. 30% increase for total installed cost.

- Steam Turbine: $500/kW direct cost. 30% increase for total installed cost.
- Generator: $60/kW direct cost. 30% increase for total installed cost.

The geothermal well costs were estimated for the input brine temperature of 150°C at a flow rate of 100 kg/s. The following assumptions are based on typical drilling depths for hydrothermal resources and parameters from Figure 6.1 and Table 9.4 in Tester et. al, 2006 [53]. The geothermal cost per depth is based on the Wellcost Lite model, which was developed by B. J. Livesay and Sandia National Laboratories over the past 20 years.

- Location: Las Vegas, Nevada
- Well Depth: 2km
- Geofluid flow rate per producer: 20 kg/s
- Number of production wells per injection well: 2
- Total number of wells: 7
- Cost per well: $2.5 Million

Table 5.7 shows a summary of all geothermal plant costs, including key design variables used to size components. The heat exchangers and condenser where further modeled in Aspen's detailed heat exchanger program to optimize the area and number of shells in order to minimize price for performance. These detailed heat exchanger results are shown in **Appendix D**.

Table 5.7 Geothermal equipment costs and key costing design variables.

Geothermal Equipment Costs			Key Design Variables	
Component Name	**Direct Cost Estimate**	**Total Installed Cost Estimate (Materials and Manpower)**	**Duty (kW)**	**A (m^2)**
ACC	$ 2,400,000	$ 2,780,000	51700	6530 (bare tube) 153300 (total)
HX1	$ 170,000	$ 285,000	37400	800
HX2	$ 1,170,000	$ 1,725,000	25000	7600
RECUP2	$ 1,220,000	$ 2,020,000	22900	6200
BHX	$ 124,000	$ 320,000	57100	270
			Output (kW)	
BT	$ 2,520,000	$ 3,270,000	11700	
ST	$ 2,170,000	$ 2,820,000	4300	
Generator	$ 960,000	$ 1,250,000	16000	
			Height (m)	**Diameter (m)**
Flash Tank	$ 36,000	$ 154,000	5	1.8
CP	$ 356,000	$ 550,000		
GP	$ 212,000	$ 350,000		
RP	$ 25,000	$ 64,000		
Well Costs	$ 17,500,000	$ 17,500,000		
Total Geothermal Equipment	**$ 28,863,000**	**$ 33,088,000**		

5.6.3 Solar Costs

The solar costs are based on numbers from NREL's solar advisor model, which in turn come from the U.S. Department of Energy's (DOE) *Office of Solar Energy Technologies Multi-Year Program Plan, 2007-2011* (https://www.nrel.gov/analysis/sam/cost_data.html). One must be careful not to assume these numbers are fixed, as NREL states: "actual costs will vary depending on the market, technology and geographic location of a project. Because of price volatility in solar markets, the cost data in the sample files is likely to be out of date."

The solar costs were broken down into three categories: solar collector costs (including the collectors, drives, and electronics), heat transfer fluid costs (including fluid and pumps), and solar storage costs (including tanks, heat exchangers and pumps for the system). The solar collector cost was $\$300/m^2$. The HTF cost was fixed at \$150/kWe (where you multiple by the design gross power), and the storage costs were fixed at \$40/kWhr-thermal. **Table 5.8** and **5.9** highlight the solar costs for each case where solar heat is added. Note the relative cost of the solar field and storage to the rest of the plant. Also note that solar storage costs 10-20% as much as the solar field.

Table 5.8 Solar field costs (not including storage).

Solar Field Size (m^2)	Solar Field Cost (USD)
130,000	\$41,406,000
170,000	\$53,406,000

Table 5.9 Solar storage costs.

Case	Storage Time (hrs)	Total Storage Cost
Jan,130K Field	3	\$4,860,000
Jan,170K Field	5	\$12,800,000
July,130K Field	3.5	\$5,970,000
July,170K Field	6.5	\$15,530,000

5.6.4 Levelized Costs of Electricity

Using the daily net energy produced in **Figure 5.21** and the total plant costs (geothermal, solar field, and solar storage), levelized cost of electricity was calculated. **Figure 5.22** shows the range in LCOE over the course of a year to be expected for different size solar fields.

For these calculations, the following assumptions were used:

- Plant Life: 25 years (all costs incurred in first year)
- Interest Rate: 8.8%
- O&M additional cost: 2 cents/kW-hr
- No additional taxes or credits

As can be seen below, the cost of electricity increases during July, and the greater the size of solar area (with storage), the greater the costs. However, when well costs are included, the costs of

electricity for geothermal only are very close to the hybrids. This suggests that there is potential for this hybrid system to be economic depending on the specific economic factors for a particular site.

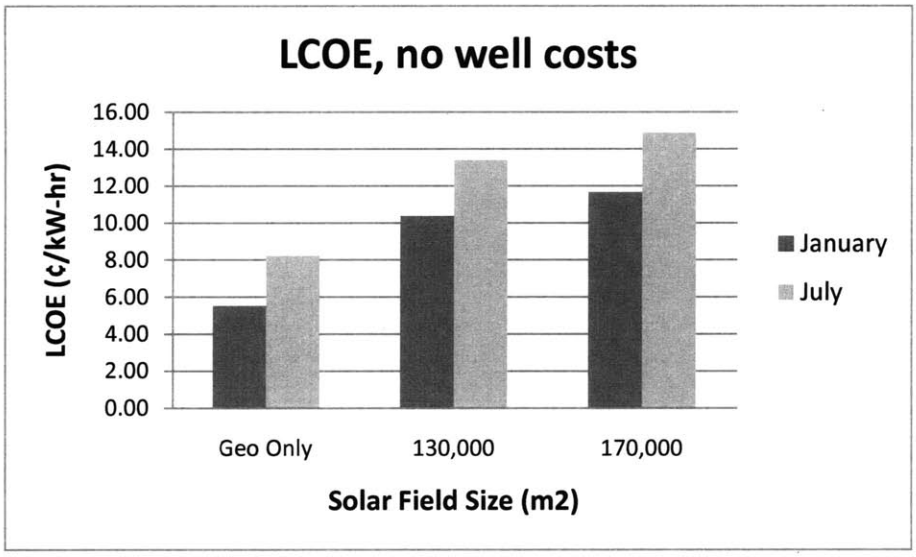


Figure 5.22 Levelized cost of electricity vs. solar field size evaluated without well costs.

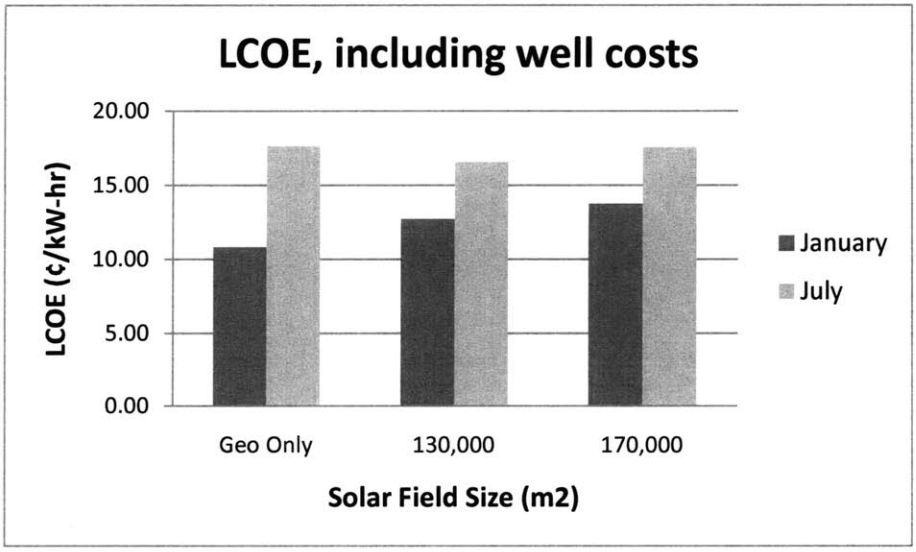


Figure 5.23 Levelized cost of electricity vs. solar field size evaluated with well costs.

Chapter 6

6. Discussion on Hybrid Models

6.1 Purpose

This chapter is designed to give the reader key insights on the models developed in this thesis. While chapters 4 and 5 discussed rigorous analysis done with computer models that produced meaningful results, this chapter discusses the assumptions behind those results and proposes extensions to them. This should allow a reader to balance results in this thesis with assumptions he or she would make when modeling these systems.

6.2 Review of Models

The models to be discussed in this chapter are as follows (see Appendices for full model descriptions and details):

Chapter 4: Hybrid energy conversion systems for existing low enthalpy geothermal power plants augmented with solar energy. These models were analyzed in pseudo-steady state across a range of solar insolation and ambient temperature conditions. A typical year performance profile for each model was generated and the levelized cost of electricity was determined with simple economic assumptions. The models considered in Chapter 4 are:

- Superheat
- Preheat
- Recirculation
- Preheat-Recirculation
- Cascade Reheat

Chapter 5: Energy conversion systems for new low enthalpy geothermal-solar hybrid power plants. These models were optimized in steady state, and one was selected for dynamic modeling. Dynamic analysis was performed for a typical January and typical July day for configurations with different amounts of solar energy storage. Costing analysis was performed on expensive equipment and the levelized cost of electricity was determined. The models considered in Chapter 5 are:

- Steady State Flash Hybrid
- Steady State Superheat Hybrid
- Dynamic Flash Hybrid

6.3 Key Parameters

When building Aspen models, each parameter must be specified with care in order to ensure meaningful results. These parameters, in effect, "reveal" the assumptions made for each model. For example, each turbine or pump has a specified efficiency value, which reveals the assumed efficiency for that equipment. However, note that these values can be superseded by design specifications or other flowsheet functions that modify initial input values. For the aspiring reader who wants to duplicate or improve these models, note that everything necessary is specified in Appendix B.

In all models, knowing equipment performance under varying conditions was critical to designing the models, especially for the turbines. Because solar energy is transient, the heat load on the equipment varied considerably. Therefore, a key set of assumptions for these models was in deciding what range of conditions was acceptable to the equipment without oversizing them.

Of the chapter 5 models, the most important parameter specified was the amount of solar heat added to the brine. This was chosen by looking at sensitivity studies on the amount of heat, in conjunction with the flashing pressure after the heat was added. While these two parameters were varied, the organic rankine cycle was continuously being optimized by the flow sheet design specifications. Without these specifications, there would be too many variables to process, and it would be difficult to make design decisions.

Another important decision universal to each model is choosing the working fluid for the rankine cycle. Care must be taken to ensure the working fluid can perform well under both day and night conditions, that is, with and without solar heat. This means that conventional working fluids for low temperatures may not work well at higher temperatures, thereby requiring working fluid flow rates to increase dramatically.

The dynamic mode is the most complex model, therefore many more parameters were specified. The most important parameters that govern the performance of the system are the controller parameters. These controllers guide the plant towards operating as efficiently as possible. Note that these parameters were guided by steady state studies, with the goal to the keep the plant operating between the most efficient steady state parameters with sun and without sun. This explains why split-range controllers were used – that is, parameters were linearly interpolated between the most efficient states.

6.4 Predictions for Solar-Geothermal Hybrids

As a 1st order exercise, all the final models from both scenarios were examined to see what connection, if any, the amount of solar thermal heat had with performance. For each model, the unit capacity cost in \$/kW and the solar heat contribution as a fraction of all thermal heat (solar and geothermal) were calculated. The solar field was assumed to be \$300/m^2, and no geothermal well costs were included in the unit capacity cost. The results are below in **Figure 6.1**.

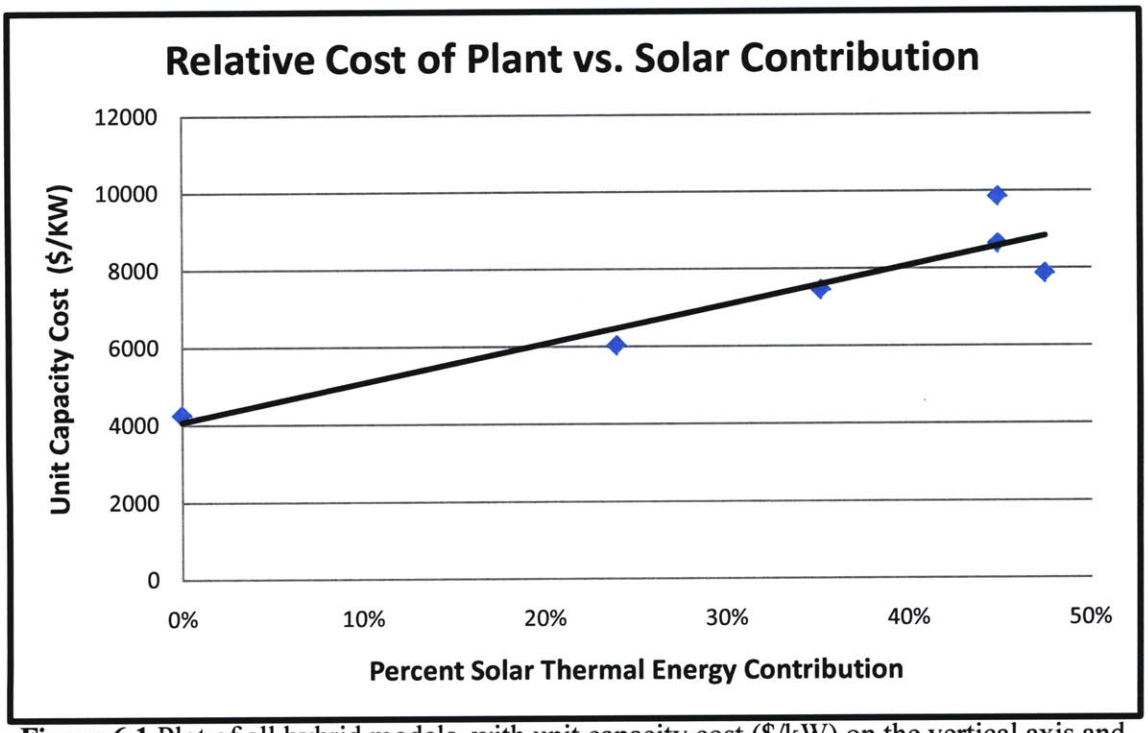


Figure 6.1 Plot of all hybrid models, with unit capacity cost ($/kW) on the vertical axis and percent solar contribution on the horizontal axis. Solar field cost is $300/m^2. Costs do not include geothermal well costs.

The six data points come from the scenario A models in section 4.5 and scenario B models in section 5.6. These models were chosen because they were the only ones evaluated for economic costs. Surprisingly, there is a near linear relationship between solar contribution to the power plant and unit capacity cost, which indicates that adding solar collectors increases the overall cost of the system. On the right side of the chart there is some scattering because they each have different amounts of solar storage. Since the overall cost analysis is about ±40%, the vertical axis intercept may be shifted to reflect true capacity costs for a pure geothermal power plant.

The linear fit is simply represented by:

$$\text{Unit Capacity Cost [\$/kW]} = 10000*S + 4000\ , \qquad (6.1)$$

where S is the fraction of all incoming thermal energy to the hybrid plant. This correlation suggests that every percentage point of solar energy adds $100/kW to the plant performance. Equation (6.1) is an empirical representation of studied model results and may not apply to different types of solar-geothermal hybrids. Conservatively, this correlation can be used as a general guide to estimate the costs for adding solar thermal collectors to low-enthalpy geothermal binary systems before well costs are included, where the total collector cost is $300/m^2.

Chapter 7

7. Conclusions

7.1 Model Results

The main conclusion of scenario A (chapter 4) is that among the hybrid configurations evaluated, the brine Preheat system is the most economic. The Preheat configuration was the only one that used the higher exergy solar heat source exclusively to heat the brine beyond its delivered temperature. The modest increase in temperature from its current value, 270°F (132.2°C), to its original design value, 320°F (160°C), does not require a large solar field, thereby keeping the capital cost of the new equipment to a manageable level. The Preheat system requires roughly 2 acres (8094 m^2) of collector area to support one unit of the existing plant analyzed. Since about 40 acres (161874 m^2) of flat, open area is available to the immediate west of the plant, there is ample land available to host the collector field.

The lesson here is that for the best hybrid configuration, the low-exergy, low-cost heat source (the geothermal energy) should be used to the maximum extent possible within its temperature limits, while the high-exergy, high-cost heat source (the solar energy) should be used only to extend the temperature above the temperature of the low-cost heat source.

The Preheat system requires roughly 2 acres (8094 m^2) of collector area to support one unit of the existing plant. Since about 40 acres (161874 m^2) of flat, open area is available to the immediate west of the plant, there is ample land available to host the collector field. **Figure 10.1** is an aerial view of the site.

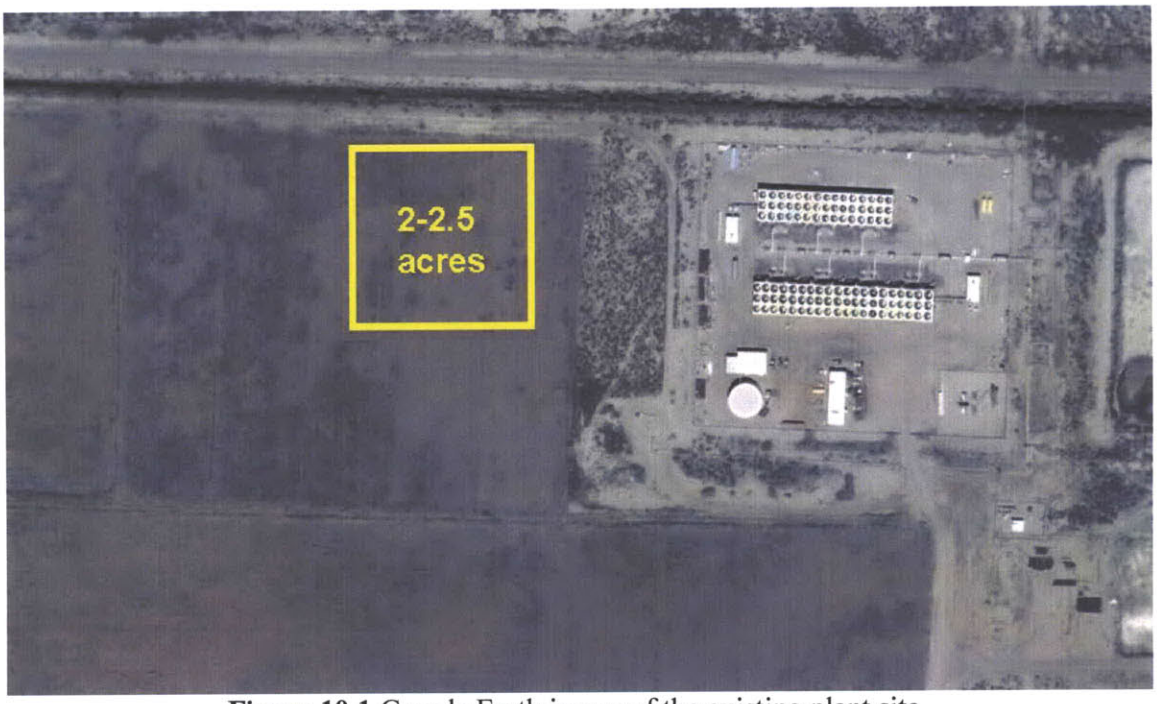


Figure 10.1 Google Earth image of the existing plant site.

In scenario B, two hybrid geothermal-solar models with different amounts of storage were compared to a similar model with only a geothermal contribution. The cases with solar heat added had a higher levelized cost of electricity than the case without solar heat if well costs were not included. However, when well costs are considered, each cases had a similar levelized cost of electricity. This suggests that there is potential for a hybrid system to be economic. Of course, if interesting government incentives and pricing plans are presented, the hybrid system can look even more attractive. For example, this hybrid system may be more reasonable if solar power is subsidized at about 6 cents/kW-hr. relative to geothermal power.

Looking back at the superheat hybrid model from chapter 5, one might consider if that configuration would be more economical that the flash hybrid model. Based on peak performance, it appears that for the same solar field size, the same amount of power would be produced (**Figure 5.12**). However, the efficiency consequence from having the larger WF flow range would substantially reduce the binary turbine efficiency and lead to less power produced over the course of a day. Taking out equipment not needed in the superheat hybrid saves about $3M from the total cost, but you would need to buy a larger binary turbine and air cooled condensor because the superheat design has a large Q duty for the air cooled condensor. Therefore most of the steam turbine savings will be offset by greater binary turbine and air cooled condensor costs. Given that the price reduction for a superheat hybrid is small, if any, and the performance reduction is unknown, it is probable that the LCOE of that model would come out the same or higher than the LCOE results from the flash hybrid.

The final models from both scenarios were examined to see what connection, if any, the amount of solar thermal heat had with performance. Surprisingly, there is a near linear relationship between

solar contribution to the power plant and unit capacity cost, which indicates that adding solar collectors increases the overall cost of the system. The linear fit is simply represented by:

$$\text{Unit Capacity Cost [\$/kW]} = 10000*S + 4000\,, \qquad (6.1)$$

where S is the fraction of all incoming thermal energy to the hybrid plant. This correlation suggests that every percentage point of solar energy adds \$100/kW to the plant performance. Equation (6.1) is an empirical representation of studied model results and may not apply to different types of solar-geothermal hybrids. Conservatively, this correlation can be used as a general guide to estimate the costs for adding solar thermal collectors to low-enthalpy geothermal binary systems before well costs are included, where the total collector cost is \$300/m^2.

7.2 Recommendations for further study

The hybrid systems chosen in this thesis are not the only possible systems that can combine geothermal energy (low specific exergy resource) and solar energy (high specific exergy resource).

For example, there are additional conceptual designs that place much more emphasis on the solar side of the hybrid system. In effect, these new ideas center on creating a new solar plant, akin to the Nevada Solar One plant, but having geothermal energy supply a portion of the feedwater heating to raise the utilization efficiency of the overall plant. This would be in keeping with the conclusion mentioned earlier. While such designs may lead to higher hybrid utilization efficiencies, they would be far more expensive and involve the design and construction of a major solar thermal power plant that would incorporate geothermal energy as an assist to the solar plant.

References

[1] Abengoa Solar, S. A. (2008). *Concentrating Solar Power.* http://www.abengoasolar.com/sites/solar/en/technologies/concentrated_solar_power/what_is_it/index.html.

[2] Alvarenga, Y., Handal, S., & Recinos, M. (2008). Solar steam booster in the ahuachapan geothermal field. *GRC, 32.*

[3] Aringhoff, R., Brakmann, G., Geyer, M., & Teske, S. (2005). *Concentrated Solar Thermal Power - NOW!*

[4] Augustine, C., Tester, J. W., Anderson, B. J., Petty, S., & Livesay, B. (2006). A Comparison of Geothermal with Oil and Gas Well Drilling Costs. *Thirty-First Workshop on Geothermal Reservoir Engineering.*

[5] Batton, W. D., & Barber, R. E. (1982). Rankine engine solar power generation, part II - the power generation module. *ASME, 81-WA/Sol-23.*

[6] Bialobrzeski, R. W. (2007). Optimization of a SEGS Solar Field for Cost Effective Power Output. (Master of Science, Georgia Institute of Technology).

[7] Burkholder, F., & Kutscher, C. (2008). *Heat-Loss Testing of Solel's UVAC3 Parabolic Trough Receiver* (NREL report no. TP-550-42394.

[8] Burkholder, F., & Kutscher, C. (2009). *Heat Loss Testing of Schott's 2008 PTR70 Parabolic Trough Receiver* (NREL Report No. TP-550-45633.

[9] Cravalho, E. G., Smith Jr., J. L., Brisson II, J. G., & McKinley, G. H. (2004). *Thermal-fluids engineering: An integrated approach to thermodynamics, fluid mechanics, and heat transfer.* Massachusetts Institute of Technology, Cambridge, MA 02139: Oxford University Press.

[10] DiPippo, R. (2008). *Geothermal power plants: Principles, applications, case studies and environmental impact* (2nd ed.). 30 Corporate Drive, Suite 400, Burlington, MA 01803: Butterworth-Heinemann.

[11] DiPippo, R., Khalifa, H. E., Correia, R.J., & Kestin, J. (1978). *Fossil Superheating in Geothermal Steam Power Plants.* Brown University, Providence, RI (DOE Report no. EY-76-S-02-4051.A001). Retrieved from http://www.osti.gov/bridge/servlets/purl/6548336-jbI5LE/6548336.pdf.

[12] Flagsol Gmbh. (2009). *Flagsol: About Us.* http://www.flagsol-gmbh.com/flagsol/cms/front_content.php.

[13] Gilman, P., Blair, N., Mehos, M., Christensen, C., Janzou, S., & Cameron, C. (2008). *Solar Advisor Model User Guide for Version 2.0* NREL Report No. TP-670-43704.

[14] Gonzalez-Aguillar, R. O., Geyer, M., Burgaleta, J. I., Zarza, E., Schiel, W., Pitz, R., et al. (2007). *Concentrating Solar Power: From Research to Implementation*Luxembourg: Office for Official Publications of the European Communities.

[15] Haberle, A., Zahler, C., Lerchenmuller, H., Mertins, M., Wittwer, C., Trieb, F., et al. The solarmundo line focussing fresnel collector. optical and thermal performance and cost calculations.

[16] Hassani, V., & Price, H. (2001). Modular Trough Power Plants. *ASME Forum 2000 "Solar Energy: The Power to Choose".*

[17] Herrmann, U., & Kearney, D. W. (2002). Survey of thermal energy storage for parabolic trough power plants. *Journal of Solar Energy Engineering, 124*(2), 145-152.

[18] Herrmann, U., Kelly, B., & Price, H. (2003). Two-tank molten salt storage for parabolic trough solar power plants. *Energy,* (29).
[19] Kearney, D. W. (2007). Parabolic Trough Collector Overview. Paper presented at the *Parabolic Trough Workshop,* Retrieved from http://www.nrel.gov/csp/troughnet/pdfs/2007/kearney_collector_technology.pdf.
[20] Kelly, B. (2006). *Nexant Parabolic Trough Solar Power Plant Systems Analysis - Task 1: Preferred Plant Size* NREL report no. SR-550-40162.
[21] Kelly, B. (2006). *Nexant Parabolic Trough Solar Power Plant Systems Analysis- Task 3: Multiple Plants at a Common Location* NREL report no. SR-550-40164.
[22] Kelly, B., Barth, D., Brosseau, D., Konig, S., & Fabrizi, F. (2007). Nitrate and Nitrite/Nitrate Salt - Heat Transport Fluids. *Parabolic Trough Technology Workshop.*
[23] Kelly, B., & Kearney, D. W. (2006). *Parabolic Trough Solar System Piping Model* No. NREL report no. TP-550-40165 Retrieved from http://www.nrel.gov/csp/troughnet/pdfs/40165.pdf.
[24] Kelly, B., & Kearney, D. W. (2006). *Thermal Storage Commercial Plant Design Study for a 2-Tank Indirect Molten Salt System* NREL report no. SR-550-40166.
[25] Khalifa, H.E. and B.W. Rhodes, *Analysis of Power Cycles for Geothermal Wellhead Conversion Systems,* EPRI AP-4070, Electric Power Research Institute, Palo Alto, CA, 1985.
[26] Luyben, W. L., Tyreus, B. D., & Luyben, M. L. (1999). *Plantwide process control.* 11 West 19th Street, New York, NY 10011: MgGraw-Hill.
[27] Manfrida, G. (1985). The choice of an optimal working point for solar collectors. *Solar Energy, 34*(6), 513.
[28] McMahan, A. C. (2006). Design & Optimization of Organic Rankine Cycle Solar-Thermal Powerplants. (Master of Science, University of Wisconsin-Madison).
[29] Mills, A. F. (1999). *Heat transfer* (2nd ed.). Upper Saddle River, NJ 07458: Prentice Hall.
[30] Mills, D. R., & Morgan, R. G. *Solar Thermal Electricity as the Primary Replacement for Coal and Oil in U.S. Generation and Transportation.*, 2008, from http://www.wired.com/images_blogs/wiredscience/files/MillsMorganUSGridSupplyCorrected.pdf.
[31] Mills, D. R., & Morgan, R. G. *Solar Thermal Power as the Plausible Basis of Grid Supply.*, 2008, from http://www.ausra.com/pdfs/T_1_1_David_Mills_2049.pdf.
[32] National Renewable Energy Laboratory. *National Solar Radiation Data Base.* http://rredc.nrel.gov/solar/old_data/nsrdb/1991-2005/tmy3/.
[33] National Renewable Energy Laboratory. (2007). *Assessment of Potential Impact of Concentrating Solar Power for Electricity Generation* DOE Report no. GO-102007-2400.
[34] National Renewable Energy Laboratory. (2008). *U.S. Parabolic Trough Power Plant Data.*http://www.nrel.gov/csp/troughnet/power_plant_data.html.
[35] National Renewable Energy Laboratory. (2009). *Concentrating Solar Power Projects.* http://www.nrel.gov/csp/solarpaces/.
[36] National Renewable Energy Laboratory. (2009). *Dynamic Maps, GIS Data, & Analysis Tools.* http://www.nrel.gov/gis/maps.html.
[37] National Renewable Energy Laboratory. (2009). *Nevada Power Clark Station, Las Vegas, NV.* http://www.nrel.gov/midc/npcs/.
[38] National Renewable Energy Laboratory. (2009). *Solar advisor model reference manual for CSP trough systems.* Unpublished manuscript.

[39] National Renewable Energy Laboratory, Sandia National Laboratory & U.S. Department of Energy. (2009). *Solar Advisor Model (SAM).* https://www.nrel.gov/analysis/sam/.

[40] Pacheco, J. E. (1999). Overview of Recent Results of the Solar Two Test and Evaluations Program. *Renewable and Advanced Energy Systems for the 21st Century,* Lahaina, Maui, Hawaii. (RAES99-7731).

[41] Perry, R.H., Green, D.W. (2008). *Perry's Chemical Engineers' Handbook* (8th ed.). Two Penn Plaza, New York, NY 10121: McGraw-Hill.

[42] Philibert, C. (2004). *International Technology Collaboration and Climate Change Mitigation. Case Study 1: Concentrating Solar Power Technologies.*

[43] Pilkington Solar International GmbH. (2000). *Survey of Thermal Storage for Parabolic Trough Power Plants* No. NREL report no. SR-550-27925.

[44] Price, H., Forristall, R., Wendelin, T., Lewandowski, A., Moss, T., Gummo, C. (2006). *Field Survey of Parabolic Trough Receiver Thermal Performance* NREL report no. CP-550-39459.

[45] *Process Heat Collectors - State of the Art within Task 33/IV*(2008). AEE INTEC.

[46] Richter, C., Teske, S., & Short, R. (2009). *Concentrating Solar Power Global Outlook 2009* Greenpeace International; SolarPACES; Estela.

[47] Sargent & Lundy LLC Consulting Group. (2003). *Assessment of Parabolic Trough and Power Tower Solar Technology Cost and Performance Forecasts* NREL Report No. TP-550-34440.

[48] Schott AG. (2009). *The key component, Schott PTR® 70 Receiver.* http://www.schottsolar.com/global/products/concentrated-solar-power/schott-ptr-70-receiver/ .

[49] Singh, N., Kaushik, S. C., & Misra, R. D. (2000). Exergetic analysis of a solar thermal power system. *Renewable Energy, 19*, 135.

[50] SkyFuel, I. (2009). *SkyTrough - Product information.* http://www.skyfuel.com/downloads/brochure/SkyTroughBrochure.pdf .

[51] Solargenix Energy. http://www.solargenix.com/ .

[52] Solel. (2009). *UVAC 2008.* http://www.solel.com/products/pgeneration/uv/ .

[53] Tester, J. W., Anderson, B. J., Batchelor, A. S., Blackwell, D. D., DiPippo, R., Drake, E. M., et al. (2006). *The Future of Geothermal Energy* .

[54] Tester, J. W., DiPippo, R., Field, R., Augustine, C., Frey, K., & Thorsteinsson, H. (2008). *Utilization of low-enthalpy geothermal fluids to produce electric power.* Unpublished manuscript.

[55] Tester, J. W., Drake, E. M., Driscoll, M. J., Golay, M. W., & Peters, W. A. (2005). *Sustainable energy: Choosing among options.* 5 Cambridge Center, Cambridge, MA 02142: MIT Press.

[56] U.S. Department of Energy. (2006). *U.S. Geothermal Resource Map.* http://www1.eere.energy.gov/geothermal/geomap.html.

[57] Winter, C., Sizmann, R. L., & Vant-Hull, S. (1991). *Solar power plants: Fundamentals, technology, systems, economics* Springer-Verlag.

[58] You, Y., & Hu, E. J. (2002). A medium-temperature solar thermal power system and its efficiency optimisation. *Applied Thermal Engineering, 22*, 357.

Appendix A: Summary of Models

Scenario A Models

Model Name	Purpose
Original Design	Model of geothermal plant based on original design document. This model includes a preheater and vaporizer in the organic cycle.
Current Operating Plant	Model of geothermal plant based on current operating conditions. There is no preheater and some design conditions are modified based on operating data (e.g. choked flow equation).
Base Geo for Hybrid Plant	Similar to current operating plant model, except with new brine specs to reflect low-temperature and low-flow artesian well in ther future. This model is used as the geothermal cycle in all hybrid models for scenario A.
Superheat	Solar heat is used to provide superheat to the working fluid between the vaporizer and the turbine.
Preheat	Solar heat is used to boost temperature of incoming brine (from 270°F to 320°F)
Recirculation	Solar heat is used to boost a fraction of brine leaving the unit and the brine is recirculated back into the unit.
Preheat-Recirculation	This model is a combination of the preheat and recirculation hybrids.
Cascade Reheat	Solar heat exiting one unit is reheated and then sent to another unit. This reduces the amount of total amount of brine needed to run two units.
Preheat (Parametric)	This model is used to provide parametric values for the preheat hybrid. Air temperature and solar insolation are varied in a controlled way to determine power produced.
Reheat (Parametric)	This model is used to provide parametric values for the recirculation hybrid. Air temperature and solar insolation are varied in a controlled way to determine power produced.
Preheat-Recirculation (Parametric)	This model is used to provide parametric values for the preheat-recirculation hybrid. Air temperature and solar insolation are varied in a controlled way to determine power produced.

Scenario B Models

Model Name	Purpose
SS Flash Hybrid	Model of solar-geothermal hybrid with brine preheat and flash. The geothermal cycle is supercritical R134a. This model is in steady state.
SS Superheat Hybrid	Model of solar-geothermal hybrid with working fluid superheat. The geothermal cycle is supercritical R134a. This model is in steady state.
Dynamic Flash Hybrid	This is a dynamic model of the flash hybrid. It includes controllers and forcing functions.

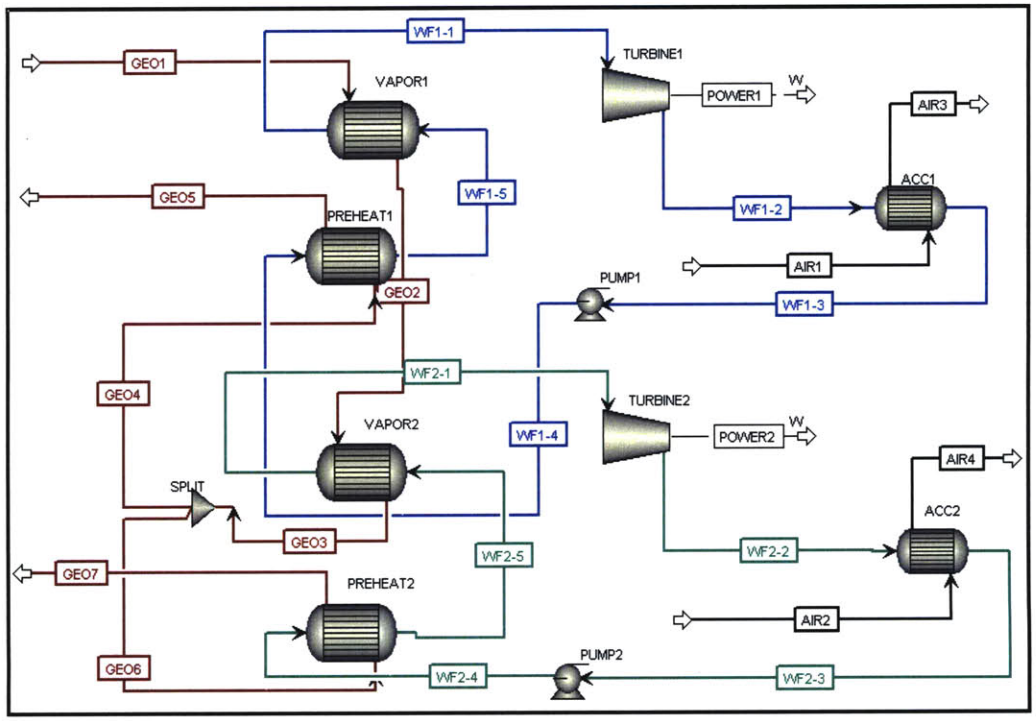
Appendix B: Scenario A Models

ORIGINAL PLANT DESIGN

FILE NAME
"Phase 1 –Original Design.bkp"

SOFTWARE
Aspen Plus 2006.5

FLOWSHEET

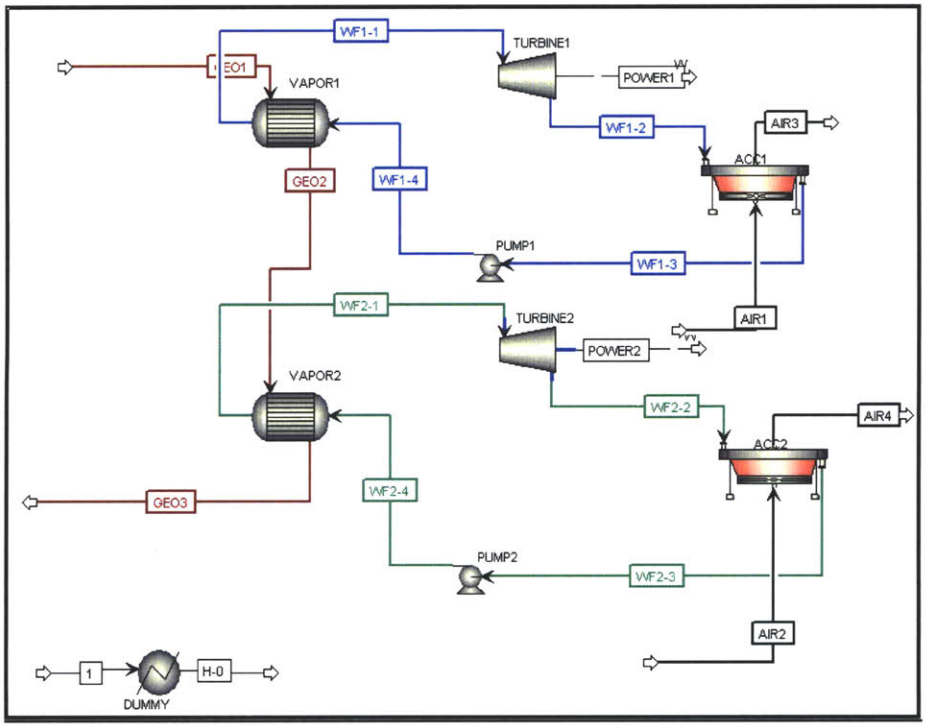


CURRENT OPERATING PLANT

FILE NAME
"Phase 1 –Current Operating Plant.bkp"

SOFTWARE
Aspen Plus 2006.5

FLOWSHEET

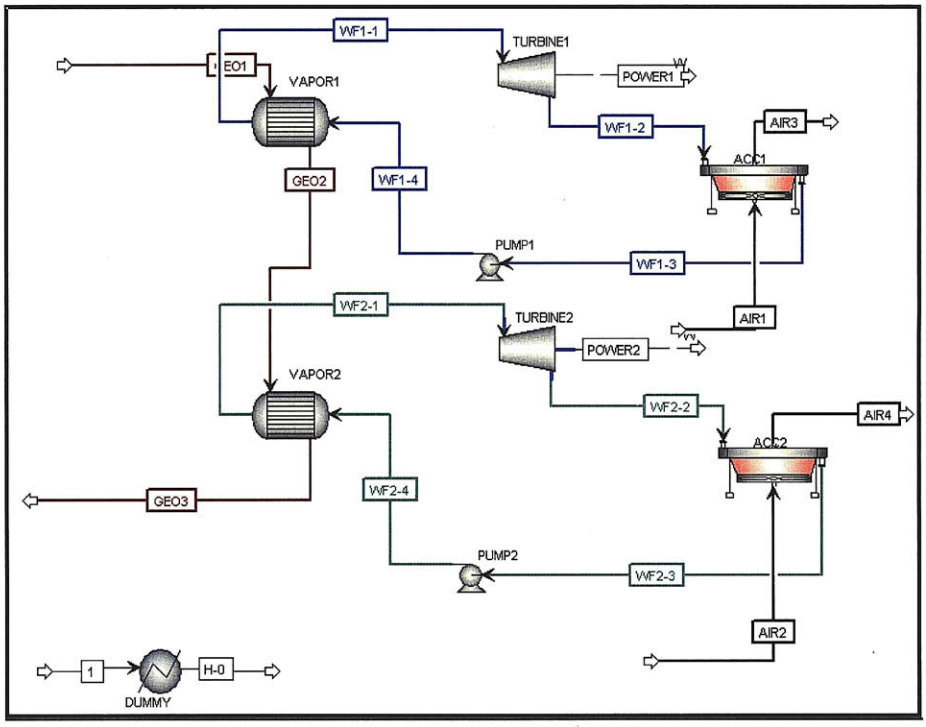


REFERENCE BASE MODEL FOR HYBRIDS

FILE NAME
"Phase 1 –Base Geo for Hybrid Plant.bkp"

SOFTWARE
Aspen Plus 2006.5

FLOWSHEET



SETUP>SPECIFICATIONS
Ambient Pressure = 12.75 psi

SETUP>SIMULATION OPTIONS
Error tolerance: 1E-08

COMPONENTS >SPECIFICATIONS
WATER
2-MET-01 (2-METHYL-BUTANE)
AIR

PROPERTIES>PROPERTY METHODS
BWRS: R134A streams
STEAMNBS: Water streams

PROPERTIES>PROP-SETS
H: Enthalpy, mixture (kJ/kmol)
S: Entropy, mixture (kJ/kmol-K)
TBUB: Buble point TEMP, mixture
VOLFLOW: Vol. flow rate (cum/sec)

STREAMS [INPUT][1 2]
[1]All Temperature is in °F, Pressure in psia, Mass flow rate in lb/hr unless otherwise written.
[2]Input values may be superseded with flowsheet options values.
STREAM 1: TEMP=50. PRES=162. <psig> STDVOL-FLOW=750. <gal/min> MASS-FRAC WATER 1.
STREAM AIR1: TEMP=70. PRES=0. <barg> VOLUME-FLOW=1052400. <cuft/min> MASS-FRAC AIR 1.
STREAM AIR2: TEMP=28. PRES=0. <barg> VOLUME-FLOW=1323000. <cuft/min> MASS-FRAC AIR 1.

STREAM GEO1: TEMP=270. PRES=162. <psig> STDVOL-FLOW=750. <gal/min> MASS-FRAC WATER 1.
STREAM WF1-3: TEMP=84.6 VFRAC=0. MASS-FLOW=151991. MASS-FRAC 2-MET-01 1.
STREAM WF1-4: TEMP=205.4 PRES=216.6 MASS-FLOW=120000. MASS-FRAC 2-MET-01 1.
STREAM WF2-3: TEMP=79.5 VFRAC=0. MASS-FLOW=173599. MASS-FRAC 2-MET-01 1.
STREAM WF2-4: TEMP=203.9 PRES=102.4 MASS-FLOW=100000. MASS-FRAC 2-MET-01 1.

BLOCKS[1,2]

[1]All Temperature is in °F, Pressure in psia, Mass flow rate in lb/hr, UA in BTU/hr-R unless otherwise written.
[2]Input values may be superseded with flowsheet options values.
DUMMY HEATER: Property Method=STEAMNBS. TEMP=0. PRES=162. <psig>
ACC1 HEATX: Property Methods=BWRS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: Hot out DEGSUB=0.1. PDROP=-1.35 (Hot)/0 (Cold).
ACC2 HEATX: Property Methods=BWRS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: Hot out DEGSUB=0.1. PDROP=-1.1 (Hot)/0 (Cold).
VAPOR1 HEATX: Property Methods=STEAMNBS (Hot)/BWRS (Cold). Calc Method=Shortcut. Type=Simulation. Constant UA=815234.8. MIN-TAPP=2. PDROP=-7.5 (Hot)/-0.4 (Cold).
VAPOR2 HEATX: Property Methods=STEAMNBS (Hot)/BWRS (Cold). Calc Method=Shortcut. Type=Simulation. Constant UA=898965.456. MIN-TAPP=2. PDROP=-7.5 (Hot)/-0.3 (Cold).
PUMP1 PUMP: Property Method=BWRS. PRES=177.1.
PUMP2 PUMP: Property Method=BWRS. PRES=85. EFF=0.65.
TURBINE1 COMPR: Property Method=BWRS. MODEL-TYPE=TURBINE. PRES=19.4. SEFF=0.795. MEFF=0.931.
TURBINE2 COMPR: Property Method=BWRS. MODEL-TYPE=TURBINE. PRES=25. <psi> SEFF=0.742. MEFF=0.931.

FLOWSHEET OPTIONS>DESIGN SPEC[1]

[1]Fortran language is paraphrased here. Actual syntax may be different.
ACC1-BP2:
Define: *VFRAC*=STREAM-VAR STREAM=WF1-3 SUBSTREAM=MIXEDVARIABLE=VFRAC. *TBUB*=STREAM-PROP STREAM=WF1-3 PROPERTY=TBUB. *TEMP*=STREAM-VAR STREAM=WF1-3 SUBSTREAM=MIXED VARIABLE=TEMP. *P*=STREAM-VAR STREAM=WF1-2 SUBSTREAM=MIXED VARIABLE=PRES. *UA*=BLOCK-VAR BLOCK=ACC1 VARIABLE=UA SENTENCE=RESULTS.
Spec: SPEC "SPEC" TO "1" TOL-SPEC ".001".
Vary: VARY BLOCK-VAR BLOCK=TURBINE1 VARIABLE=PRES SENTENCE=PARAM LIMITS "5" "40" STEP-SIZE=0.01.
Fortran: uaspec = 650000
pmin =13.0
if (P.gt. pmin*1.01 .or. ua .gt. uaspec*1.01) then
spec = uaspec/ua
else
spec = p/pmin
end if
ACC2-BP2:
Define: *VFRAC*=STREAM-VAR STREAM=WF2-3 SUBSTREAM=MIXEDVARIABLE=VFRAC. *TBUB*=STREAM-PROP STREAM=WF2-3 PROPERTY=TBUB. *TEMP*=STREAM-VAR STREAM=WF2-3 SUBSTREAM=MIXED VARIABLE=TEMP. *P*=STREAM-VAR STREAM=WF2-2 SUBSTREAM=MIXED VARIABLE=PRES. *UA*=BLOCK-VAR BLOCK=ACC2 VARIABLE=UA SENTENCE=RESULTS.
Spec: SPEC "SPEC" TO "1" TOL-SPEC ".001".
Vary: VARY BLOCK-VAR BLOCK=TURBINE2 VARIABLE=PRES SENTENCE=PARAM LIMITS "5" "40" STEP-SIZE=0.01.
Fortran: uaspec = 700000
pmin =13.0
if (P.gt. pmin*1.01 .or. ua .gt. uaspec*1.01) then
spec = uaspec/ua
else
spec = p/pmin
end if
CHOKE1:
Define: *PRES*=STREAM-VAR STREAM=WF1-1 SUBSTREAM=MIXED VARIABLE=PRES. *TEMP*=STREAM-VAR STREAM=WF1-1 SUBSTREAM=MIXED VARIABLE=TEMP. *FLOW*=STREAM-VAR STREAM=WF1-1 SUBSTREAM=MIXED VARIABLE=MASS-FLOW.
Spec: SPEC "FLOW*DSQRT(TEMP+459.67)" TO "(18412*PRES)" TOL-SPEC "10".
Vary: VARY BLOCK-VAR BLOCK=PUMP1 VARIABLE=PRES SENTENCE=PARAM LIMITS "100" "300" MAX-STEP-SIZ=0.05.
CHOKE2:
Define: *PRES*=STREAM-VAR STREAM=WF2-1 SUBSTREAM=MIXED VARIABLE=PRES. *TEMP*=STREAM-VAR STREAM=WF2-1 SUBSTREAM=MIXED VARIABLE=TEMP. *FLOW*=STREAM-VAR STREAM=WF2-1 SUBSTREAM=MIXED VARIABLE=MASS-FLOW.
Spec: SPEC "FLOW*DSQRT(TEMP+459.67)" TO "(43874*PRES)" TOL-SPEC "10".
Vary: VARY BLOCK-VAR BLOCK=PUMP2 VARIABLE=PRES SENTENCE=PARAM LIMITS "40" "150" MAX-STEP-SIZ=0.1.
FLOW1B:
Define: *TEMP*=STREAM-VAR STREAM=WF1-1 SUBSTREAM=MIXED VARIABLE=TEMP. *VFRAC*=STREAM-VAR STREAM=WF1-1 SUBSTREAM=MIXED VARIABLE=VFRAC. *TBUB*=STREAM-PROP STREAM=WF1-1 PROPERTY=TBUB.
Spec: SPEC "(VFRAC-1)+(TEMP-TBUB-5)" TO "0" TOL-SPEC ".01".
Vary: VARY STREAM-VAR STREAM=WF1-4 SUBSTREAM=MIXED VARIABLE=MASS-FLOW LIMITS "25000" "200000" MAX-STEP-SIZ=0.05.
FLOW2B:
Define: *TEMP*=STREAM-VAR STREAM=WF2-1 SUBSTREAM=MIXED VARIABLE=TEMP. *VFRAC*=STREAM-VAR STREAM=WF2-1 SUBSTREAM=MIXED VARIABLE=VFRAC. *TBUB*=STREAM-PROP STREAM=WF2-1 PROPERTY=TBUB.
Spec: SPEC "(VFRAC-1)+(TEMP-TBUB-5)" TO "0" TOL-SPEC ".02".

Vary: VARY STREAM-VAR STREAM=WF2-4 SUBSTREAM=MIXED VARIABLE=MASS-FLOW LIMITS "50000" "225000" MAX-STEP-SIZ=0.05.

FLOWSHEET OPTIONS>CALCULATOR[1]

[1]Fortran language is paraphrased here. Actual syntax may be different.

FANS:

Define: *FAN1*=PARAMETER 5 PHYS-QTY=POWER UOM="kW" INIT-VAL=104. *FAN2*=PARAMETER 6 PHYS-QTY=POWER UOM="kW" INIT-VAL=130. *ACC1UA*=BLOCK-VAR BLOCK=ACC1 VARIABLE=UA SENTENCE=RESULTS. *ACC2UA*=BLOCK-VAR BLOCK=ACC2 VARIABLE=UA SENTENCE=RESULTS.

Fortran: <Comment>This will calculate the correct fan parasitic for low air temp. In normal operation, fans are shut down. In this simulation, we will model the decrease in fan power linearly. Note that 650000 is the initial UA value for ACC1</C>
FAN1 = 104 * (ACC1UA / 650000)
<Comment> Note that 700000 is the initial UA value for ACC2</C>
FAN2 = 130 * (ACC2UA / 700000)

Sequence: READ-VARS *ACC1UA ACC2UA*. WRITE-VARS *FAN1 FAN2*.

GEO-EFF:

Define: *AIRTP*=STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=TEMP. *EGEO*=PARAMETER 2 PHYS-QTY=POWER UOM="kW". *POWER1*=BLOCK-VAR BLOCK=TURBINE1 VARIABLE=NET-WORK SENTENCE=RESULTS. *POWER2*=BLOCK-VAR BLOCK=TURBINE2 VARIABLE=NET-WORK SENTENCE=RESULTS. *GEOEF*=PARAMETER 3 PHYS-QTY=DIMENSIONLES UOM= "Unitless". *H*=STREAM-PROP STREAM=GEO1 PROPERTY=H. *HO*=STREAM-PROP STREAM=H-0 PROPERTY=H. *S*=STREAM-PROP STREAM=GEO1 PROPERTY=S. *SO*=STREAM-PROP STREAM=H-0 PROPERTY=S. *MOLEFL*=STREAM-VAR STREAM=GEO1 SUBSTREAM=MIXED VARIABLE=MOLE-FLOW. *PUMP1P*=BLOCK-VAR BLOCK=PUMP1 VARIABLE=NET-WORK SENTENCE=RESULTS. *PUMP2P*=BLOCK-VAR BLOCK=PUMP2 VARIABLE=NET-WORK SENTENCE=RESULTS. *QVAP1*=BLOCK-VAR BLOCK=VAPOR1 VARIABLE=CALC-DUTY SENTENCE=RESULTS. *QVAP2*=BLOCK-VAR BLOCK=VAPOR2 VARIABLE=CALC-DUTY SENTENCE=RESULTS. *THEFF*=PARAMETER 4 PHYS-QTY=DIMENSIONLES UOM="Unitless". *FAN1*=PARAMETER 5 PHYS-QTY=POWER UOM="kW" INIT-VAL=104. *FAN2*=PARAMETER 6 PHYS-QTY=POWER UOM="kW" INIT-VAL=130.

Fortran: <Comment>This is the exergy of the Geo-fluid where ambient air temp = 94. Note that if you change the amb. air temp, you must also change the definition of H-O and S-O Property-Sets </C>
egeo = molefl*(h-ho-(273.15+(airtp-32)*5/9)*(s-so))
<Comment> This is the utilization eff. of basic geothermal plant. The numerator is net-work, so pumps and acc fans are subtracted </C>
geoeff = (-1*(power1+power2)-pump1p-pump2p-fan1-fan2)/egeo
<Comment> This is the thermal efficiency </C>
theff=(-1*(power1+power2)-pump1p-pump2p-fan1-fan2)/(qvap1+qvap2)

Sequence: READ-VARS *AIRTP POWER1 POWER2 H HO S SO MOLEFL PUMP1P PUMP2P QVAP1 QVAP2 FAN1 FAN2.* WRITE-VARS *EGEO GEOEFF THEFF.*

FLOWSHEET OPTIONS>TRANSFER

AIR1TO2: SET STREAM-VAR STREAM=AIR2 SUBSTREAM=MIXED VARIABLE=TEMP EQUAL-TO STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=TEMP.

DUMMY-PR: SET BLOCK-VAR BLOCK=DUMMY VARIABLE=PRES SENTENCE=PARAM EQUAL-TO STREAM-VAR STREAM=GEO1 SUBSTREAM=MIXED VARIABLE=PRES.

DUMMY-TP: BLOCK-VAR BLOCK=DUMMY VARIABLE=TEMP SENTENCE=PARAM EQUAL-TO STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=TEMP.

MODEL ANALYSIS TOOLS>SENSITIVITY

VARYAIR1:

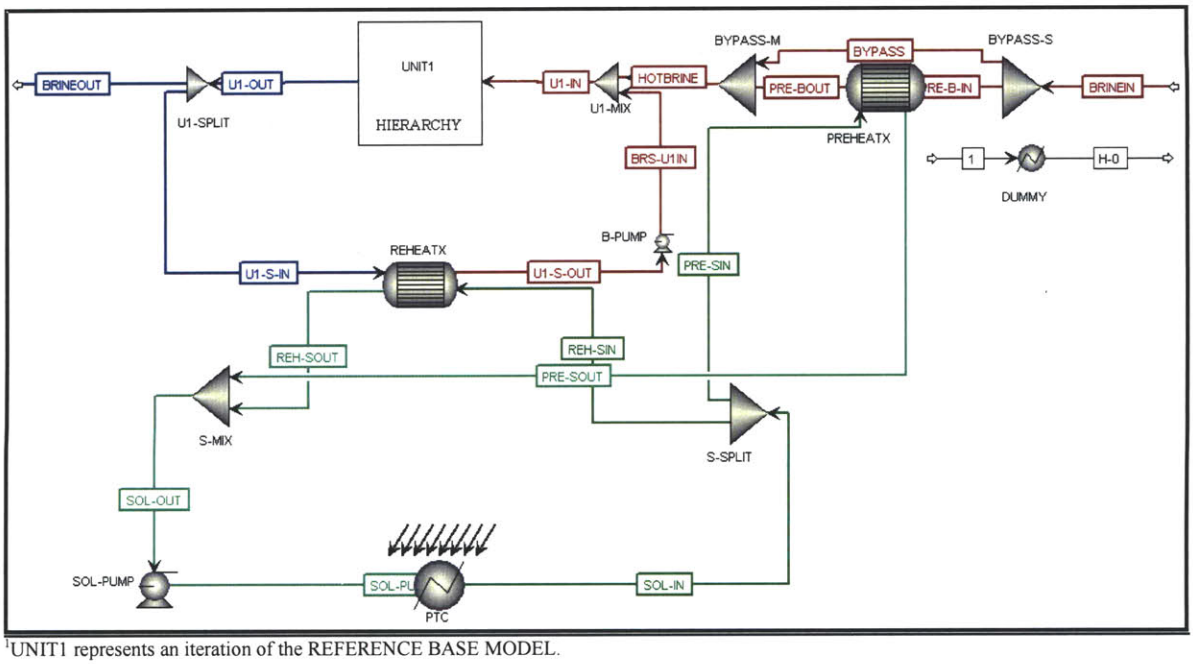
Vary: (1) VARY STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=TEMP RANGE LOWER="93" UPPER="94" INCR="1".

PREHEAT, RECIRCULATION, AND PREHEAT-RECIRCULATION MODELS

FILE NAMES
"Phase 1 – Preheat.bkp"
"Phase 1 – Recirculation.bkp"
"Phase 1 – Preheat-Recirculation.bkp"

SOFTWARE
Aspen Plus 2006.5

FLOWSHEET[1]

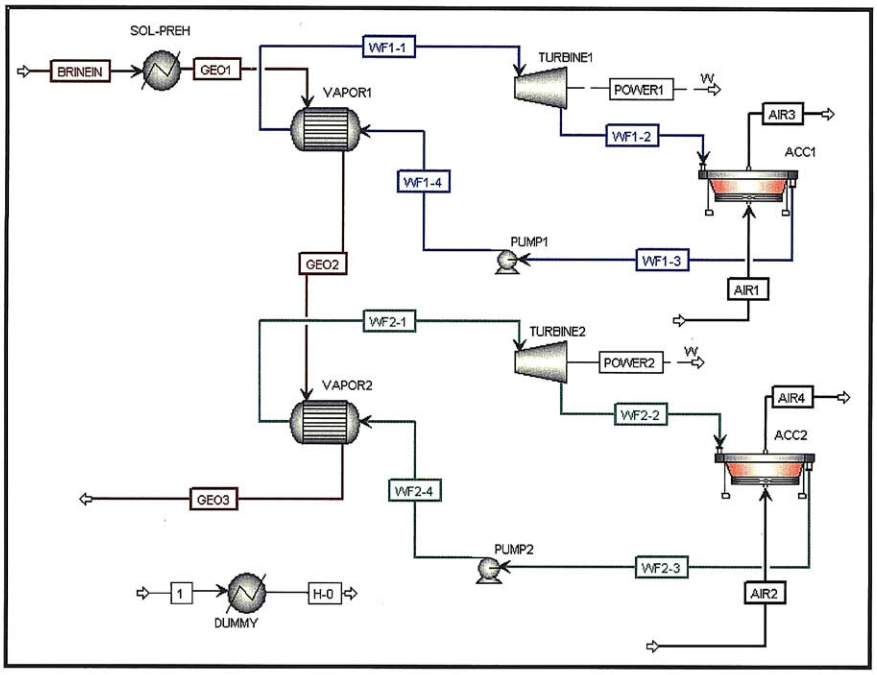


[1]UNIT1 represents an iteration of the REFERENCE BASE MODEL.

FILE NAMES

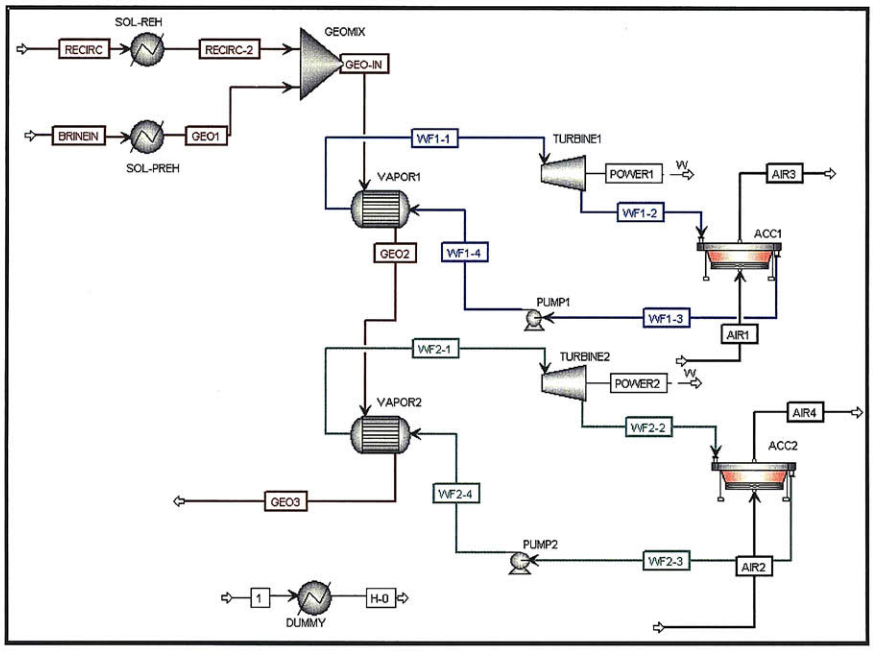
"Phase 1 – Preheat (Parametric).bkp"

FLOWSHEET[1]



FILE NAMES
"Phase 1 – Preheat-Recirculation (Parametric).bkp"

FLOWSHEET[1]

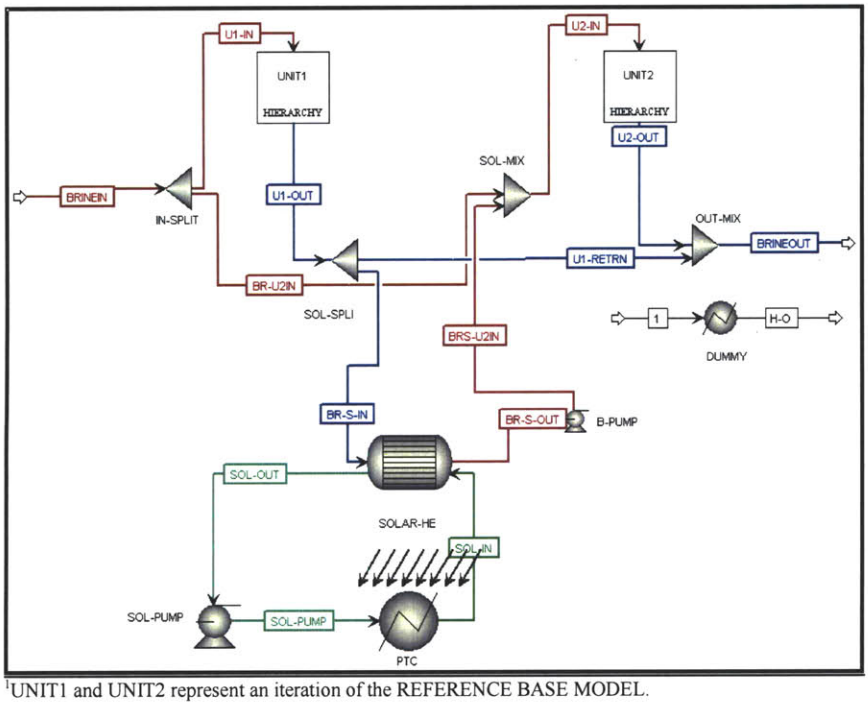


CASCADE REHEAT MODELS

FILE NAME
"Phase 1 – Cascade Reheat.bkp"

SOFTWARE
Aspen Plus 2006.5

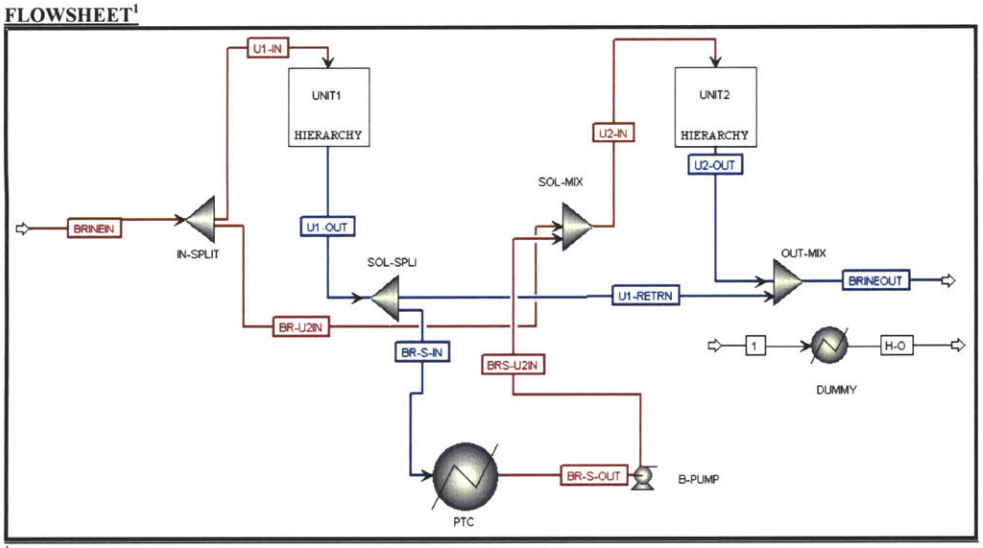
FLOWSHEET[1]



[1]UNIT1 and UNIT2 represent an iteration of the REFERENCE BASE MODEL.

FILE NAME
"Phase 1 – Reheat (Parametric).bkp"

FLOWSHEET[1]



[1]UNIT1 and UNIT2 represent an iteration of the REFERENCE BASE MODEL.

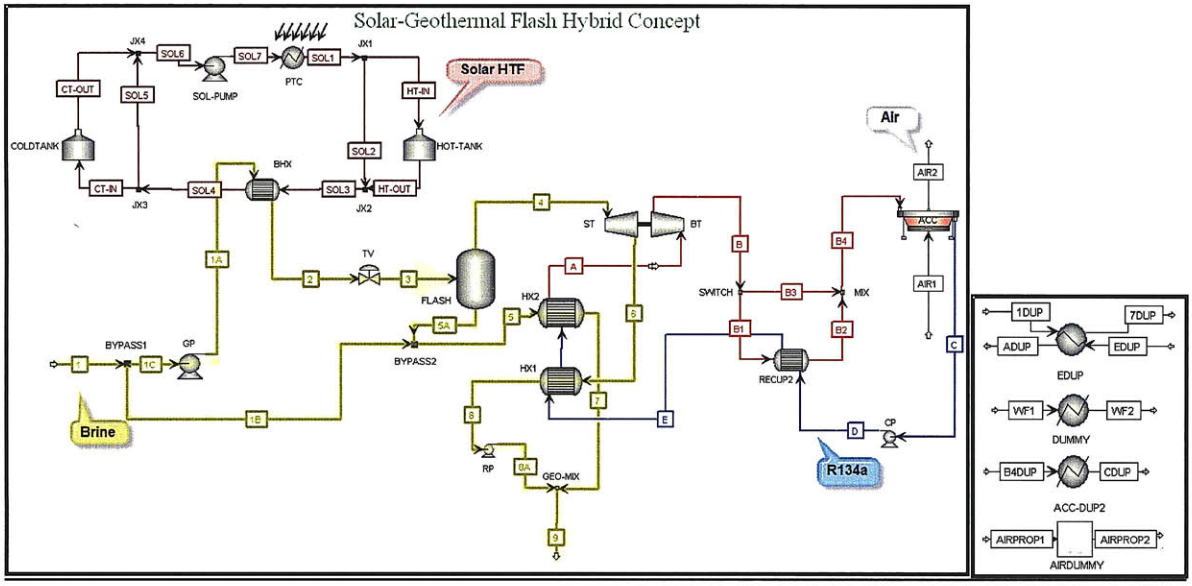
Appendix C: Scenario B Models

FLASH-GEOTHERMAL STEADY-STATE MODEL

FILE NAME
"Phase 2 – SS Flash Hybrid.bkp"

SOFTWARE
Aspen Plus V7.1

FLOWSHEET



SETUP>SPECIFICATIONS
Ambient Pressure = 1.01325 bar

SETUP>SIMULATION OPTIONS
Error tolerance: 1E-06

COMPONENTS >SPECIFICATIONS
WATER
R134A (1,1,1,2-TETRAFLUOROETHANE)
AIR
Therminol-VP-1

PROPERTIES>PROPERTY METHODS
BWRS: R134A streams
NRTL: Therminol-VP-1 streams
STEAMNBS: Water streams
IDEAL: Air streams

PROPERTIES>PROP-SETS
CPAIR: heat capacity, pure component (kJ/kg-K) at 20°C and 1 bar
CPMX: heat capacity, mixture (kJ/kg-K)
H0: Enthalpy, mixture (kJ/kmol) at 20°C
H1: Enthalpy, mixture (kJ/kmol)
PC: Critical Pressure, mixture
PDEW: Dew point pressure, mixture
SO: Entropy, mixture (kJ/kmol-K) at 20°C
S1: Entropy, mixture (kJ/kmol-K)
TDEW: Dew point temperature, mixture

STREAMS [INPUT][1,2]

[1]All Temperature is in °C, Pressure in bar, Mass flow rate in kg/sec unless otherwise written.
[2]Input values may be superseded with flowsheet options values.

STREAM 1: TEMP=150. PRES=20. MASS-FLOW=100. MASS-FRAC WATER=1.
STREAM 5: TEMP=200. PRES=10. MASS-FLOW=100. MASS-FRAC WATER=1.
STREAM 5DUP: TEMP=44. PRES=44. MASS-FLOW=50. MASS-FRAC WATER=1.
STREAM A2: TEMP=175.6. PRES=80. MASS-FLOW R134A=200.
STREAM AIR1: TEMP=20. PRES=1. MASS-FLOW=9301.69148. MASS-FRAC AIR=1.
STREAM AIRPROP1: TEMP=20. PRES=1. MASS-FLOW=5000. MASS-FRAC AIR=1.
STREAM B4: TEMP=40. VFRAC=1. MASS-FLOW=285. MASS-FRAC R134A=1.
STREAM B4DUP: TEMP=99. PRES=99. MASS-FLOW=99. MASS-FRAC WATER=1.
STREAM C: TEMP=30. VFRAC=0. MASS-FLOW=285. MASS-FRAC R134A=1.
STREAM D: TEMP=40. PRES=80. MASS-FLOW=285. MASS-FRAC R134A=1.
STREAM E: TEMP=40. PRES=80. MASS-FLOW=285. MASS-FRAC R134A=1.
STREAM FDUP: TEMP=44. PRES=44. MASS-FLOW=1. MASS-FLOW WATER=1.
STREAM SOL1: TEMP=698. <F> PRES=150. <psia> MASS-FLOW=300. MASS-FRAC THERMVP1=1.
STREAM WF1: TEMP=30. PRES=1. MASS-FLOW=1. MASS-FRAC R134A=1.

BLOCKS[1,2]

[1]All Temperature is in °C, Pressure in bar, Mass flow rate in kg/sec unless otherwise written.
[2]Input values may be superseded with flowsheet options values.

AIRDUMMY MIXER
BYPASS2 MIXER: Property Method=STEAMNBS.
GEO-MIX MIXER: Valid Phases=Liquid-Only. Property Method=STEAMNBS.
JX2 MIXER: Property Method=NRTL. TEMP Estimate=400.
JX4 MIXER: Property Method=NRTL. TEMP Estimate=270.
MIX MIXER: Property Method=BWRS.
BYPASS1 FSPLIT: Property Method=STEAMNBS. FRAC 1B=0.
JX1 FSPLIT: Property Method=NRTL. FRAC SOL2=1.
JX3 FSPLIT: Property Method=NRTL. FRAC SOL5=1.
SWITCH FSPLIT: Property Method=BWRS. FRAC B1=0.
ACC-DUP2 HEATER: Property Method=BWRS. PRES=0. DEGSUB=2.
DUMMY HEATER: Property Method=BWRS. PRES=0. ΔTEMP=2.
PTC HEATER: Valid Phases=Liquid-Only. Property Method=NRTL. TEMP=390. PRES=-6. <psi>
COLDTANK FLASH2: Property Method=NRTL. TEMP=150. PRES=1.
FLASH FLASH2: Property Method=STEAMNBS. PRES=0. DUTY=0.
HOT-TANK FLASH2: Property Method=NRTL. TEMP=150. PRES=1.
ACC HEATX: Property Methods=BWRS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: Hot out DEGSUB=2. PDROP=-0.2 (Hot)/-0.2 (Cold).
BHX HEATX: Property Methods=NRTL (Hot)/STEAMNBS (Cold). Calc Method=Shortcut. SPEC: Cold out TEMP=275. PDROP=-0.2 (Hot)/-0.2 (Cold).
HX1 HEATX: Property Methods=STEAMNBS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: Hot out VFRAC=0. PDROP=-0.2 (Hot)/-0.2 (Cold).
HX2 HEATX: Property Methods=STEAMNBS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: (Hot out – Cold in)=3. PDROP=-0.2 (Hot)/-0.2 (Cold).
RECUP2 HEATX: Property Methods=BWRS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: (Hot out – Cold in)=5. PDROP=-0.2 (Hot)/-0.2 (Cold).
EDUP MHEATX: Property Methods=STEAMNBS (Hot)/BWRS (Cold). Valid Phases=Liquid-Only (Hot)/Liquid-Only (Cold). TEMP=190 (Cold).
CP PUMP: Property Method=BWRS. PRES=80. EFF=0.8.
GP PUMP: Property Method=BWRS. PRES=60.4 EFF=0.8.
RP PUMP: Property Method=STEAMNBS. PRES=15. EFF=0.8.
SOL-PUMP PUMP: Property Method=NRTL. DELP=9. <psi> EFF=0.8.
BT COMPR: Property Method=BWRS. MODEL-TYPE=TURBINE. PRES=10. SEFF=0.85. MEFF=0.98.
ST COMPR: Property Method=STEAMNBS. MODEL-TYPE=TURBINE. PRES=3. SEFF=0.85. MEFF=0.98.
TV VALVE: Property Method=STEAMNBS. PRES=16.

FLOWSHEET OPTIONS>DESIGN SPEC[1]

[1]Fortran language is paraphrased here. Actual syntax may be different.

BAUMANN1:
Define: *VFRAC4*=STREAM-VAR STREAM=4 SUBSTREAM=MIXED VARIABLE=VFRAC. *VFRAC6*=STREAM-VAR STREAM=6 SUBSTREAM=MIXED VARIABLE=VFRAC. *EFF*=BLOCK-VAR BLOCK=ST VARIABLE=SEFF SENTENCE=RESULTS.
Spec: SPEC "EFF" TO "TARGET" TOL-SPEC ".00001".
Vary: VARY BLOCK-VAR BLOCK=ST VARIABLE=SEFF SENTENCE=PARAM LIMITS ".1" "1" MAX-STEP-SIZ=0.1.
Fortran: <Comment>This Design Spec looks at the *Steam Turbine (ST)* and the streams going into and out of ST. The Baumann Rule penalizes the ST efficiency (linearly) if some liquid is present in the input or output streams. SOMETIMES THE SUPERCRITICAL FLUID </C>
TARGET = 0.85 * (VFRAC4 + VFRAC6)/2

BAUMANN2:
Define: *VFRCA2*=STREAM-VAR STREAM=A2 SUBSTREAM=MIXED VARIABLE=VFRAC. *VFRCB*=STREAM-VAR STREAM=B SUBSTREAM=MIXED VARIABLE=VFRAC. *EFF*=BLOCK-VAR BLOCK=BT VARIABLE=SEFF SENTENCE=RESULTS.
Spec: SPEC "EFF" TO "TARGET" TOL-SPEC ".00001".

Vary: VARY BLOCK-VAR BLOCK=BT VARIABLE=SEFF SENTENCE=PARAM LIMITS ".1" "1" MAX-STEP-SIZ=0.1.
Fortran: <Comment>This Design Spec looks at the *Binary Turbine (BT)* and the streams going into and out of BT (A2 & B). The Baumann Rule penalizes the BT efficiency (linearly) if some liquid is present in the input or output streams. SOMETIMES THE SUPERCRITICAL FLUID (A) IS LABELED AS LIQUID. FOR THESE CASES, WE OVERRIDE IT WITH THE CSV (A2). </C>
IF (VFRCA2 .GT. .1) THEN
TARGET = 0.85 * (VFRCA2 + VFRCB)/2
ELSE
TARGET = 0.85 * (1.0 + VFRCB)/2
END IF
GPUMP:
Define: *SATPR*=STREAM-PROP STREAM=2 PROPERTY=PDEW. *PR1A* =STREAM-VAR STREAM=1A SUBSTREAM=MIXED VARIABLE=PRES.
Spec: SPEC "PR1A" TO "SATPR + 1" TOL-SPEC ".01".
Vary: VARY BLOCK-VAR BLOCK=GP VARIABLE=PRES SENTENCE=PARAM LIMITS "20" "300".
LMTD: (disabled)
Define: *DTLM*=BLOCK-VAR BLOCK=HX2 VARIABLE=DTLM SENTENCE=RESULTS. *TSPEC*=PARAMETER 5 PHYS-QTY=DELTA-T UOM="C" INIT-VAL=10. *GEOFLO*=STREAM-VAR STREAM=5 SUBSTREAM=MIXED VARIABLE=MASS-FLOW.
Spec: SPEC "DTLM" TO "TSPEC" TOL-SPEC ".0001".
Vary: VARY STREAM-VAR STREAM=E SUBSTREAM=MIXED VARIABLE=MASS-FLOW LIMITS "100" "800" MAX-STEP-SIZ=0.05.
ST-PRES: (disabled)
Define: *VFRAC8*=STREAM-VAR STREAM=8 SUBSTREAM=MIXED VARIABLE=VFRAC.
Spec: SPEC "VFRAC8" TO "0" TOL-SPEC ".01".
Vary: VARY BLOCK-VAR BLOCK=ST VARIABLE=PRES SENTENCE=PARAM LIMITS "2" "8".
THROTTLE: (disabled)
Define: *TEMP3*=STREAM-VAR STREAM=3 SUBSTREAM=MIXED VARIABLE=TEMP.
SPEC: SPEC "TEMP3" TO "200" TOL-SPEC ".01"
VARY: VARY BLOCK-VAR BLOCK=TV VARIABLE=P-OUT SENTENCE=PARAM LIMITS "10" "18" MAX-STEP-SIZ=0.1.

FLOWSHEET OPTIONS>CALCULATOR[1]
[1]Fortran language is paraphrased here. Actual syntax may be different.
AIRFLOW:
Define: *MASAIR*=STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=MASS-FLOW. *QACC*=BLOCK-VAR BLOCK=ACC-DUP2 VARIABLE=QCALC SENTENCE=PARAM. *TAPP*=PARAMETER 19 PHYS-QTY=DELTA-T UOM="C" INIT-VAL=15.0395836. *CPAIR*=STREAM-PROP STREAM=AIRPROP1 PROPERTY=CPMX.
Fortran: MASAIR = -(QACC*1e6/3600) / (CPAIR * TAPP /2)
Sequence: READ-VARS *QACC TAPP CPAIR*. WRITE-VARS *MASAIR*.
LPCALC:
Define: *PDEW*=STREAM-PROP STREAM=WF1 PROPERTY=PDEW. *PTURB*=BLOCK-VAR BLOCK=BT VARIABLE=PRES SENTENCE=PARAM.
Fortran: PTURB = PDEW
Sequence: READ-VARS *PDEW*. WRITE-VARS *PTURB*. EXECUTE AFTER CALCULATOR SPECS.
PRSPEC:
Define: *PC*=STREAM-PROP STREAM=C PROPERTY=PC. *PRES5*=BLOCK-VAR BLOCK=CP VARIABLE=PRES SENTENCE=PARAM. *PR*=PARAMETER 15 INIT-VAL=1.21152988.
Fortran: PRES5 = PR * PC
Sequence: READ-VARS *PC PR*. WRITE-VARS *PRES5*.
RECUP:
Define: *TEMP3*=STREAM-VAR STREAM=B SUBSTREAM=MIXED VARIABLE=TEMP. *TEMP5*=STREAM-VAR STREAM=D SUBSTREAM=MIXED VARIABLE=TEMP. *SWITCH*=BLOCK-VAR BLOCK=SWITCH SENTENCE=FRAC VARIABLE=FRAC ID1=B1.
Fortran:
IF (TEMP3 .GT. TEMP5+10) THEN
SWITCH = 1D0
ELSE
SWITCH = 0D0
END IF
Sequence: READ-VARS *TEMP3 TEMP5*. WRITE-VARS *SWITCH*.
SPECS:
Define: *TDEADI*=PARAMETER 7 PHYS-QTY=TEMPERATURE UOM="C" INIT-VAL=20. *TAIR1*=STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=TEMP. *TDEW*=STREAM-VAR STREAM=WF1 SUBSTREAM=MIXED VARIABLE=TEMP. *TAPP*=PARAMETER 19 PHYS-QTY=DELTA-T UOM="C" INIT-VAL=15.0395836.
Fortran: TAIR1 = TDEADI
TDEW = TAIR1 + TAPP
Sequence: READ-VARS *TDEADI TAPP*. WRITE-VARS *TAIR1 TDEW*.
U-EFF:
Define: *SGEO0*=STREAM-PROP STREAM=1 PROPERTY=S0. *HGEO0*=STREAM-PROP STREAM=1 PROPERTY=H0. *SGEO1*=STREAM-PROP STREAM=1 PROPERTY=S1. *HGEO1*=STREAM-PROP STREAM=1 PROPERTY=H1. *SGEO9*=STREAM-PROP STREAM=9 PROPERTY=S1. *HGEO9*=STREAM-PROP STREAM=9 PROPERTY=H1. *SSOL0*=STREAM-PROP STREAM=SOL3 PROPERTY=S0. *HSOL0*=STREAM-PROP STREAM=SOL3 PROPERTY=H0. *SSOL1*=STREAM-PROP STREAM=SOL3 PROPERTY=S1. *HSOL1*=STREAM-PROP STREAM=SOL3 PROPERTY=H1. *SSOL2*=STREAM-PROP STREAM=SOL4 PROPERTY=S1. *HSOL2*=STREAM-PROP STREAM=SOL4 PROPERTY=H1. *MGEO*=STREAM-VAR STREAM=1 SUBSTREAM=MIXED

VARIABLE=MOLE-FLOW. *MSOL*=STREAM-VAR STREAM=SOL1 SUBSTREAM=MIXED VARIABLE=MOLE-FLOW. *MASAIR*=STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=MASS-FLOW. *UEFF*=PARAMETER 1 PHYS-QTY=DIMENSIONLES UOM="Unitless" INIT-VAL=1. *UEFF2*=PARAMETER 10 PHYS-QTY=DIMENSIONLES UOM= "Unitless" INIT-VAL=1. *UEFF3*=PARAMETER 11 PHYS-QTY=DIMENSIONLES UOM= "Unitless" INIT-VAL=1. *THEFF*=PARAMETER 2 PHYS-QTY=DIMENSIONLES UOM="Unitless" INIT-VAL=1. *TDEAD*=PARAMETER 7 PHYS-QTY=TEMPERATURE UOM="C" INIT-VAL=20. *NETPOW*=PARAMETER 3 PHYS-QTY=POWER UOM="kW" INIT-VAL=10000. *ST*=BLOCK-VAR BLOCK=ST VARIABLE=NET-WORK SENTENCE=RESULTS. *BT*=BLOCK-VAR BLOCK=BT VARIABLE=NET-WORK SENTENCE=RESULTS. *CP*=BLOCK-VAR BLOCK=CP VARIABLE=NET-WORK SENTENCE=RESULTS. *RP*=BLOCK-VAR BLOCK=RP VARIABLE=NET-WORK SENTENCE=RESULTS. *ACC*=PARAMETER 21 PHYS-QTY=POWER UOM="MW". *EGEO*=PARAMETER 33 PHYS-QTY=POWER UOM="kW". *DEGEO*=PARAMETER 36 PHYS-QTY=POWER UOM="kW". *ESOL*=PARAMETER 34 PHYS-QTY=POWER UOM="kW". *DESOL*=PARAMETER 35 PHYS-QTY=POWER UOM="kW". *QHX1*=BLOCK-VAR BLOCK=HX1 VARIABLE=CALC-DUTY SENTENCE=RESULTS. *QHX2*=BLOCK-VAR BLOCK=HX2 VARIABLE=CALC-DUTY SENTENCE=RESULTS. *TOTPOW*=PARAMETER 98. *GP*=BLOCK-VAR BLOCK=GP VARIABLE=NET-WORK SENTENCE=RESULTS. *SGEO2*=STREAM-PROP STREAM=2 PROPERTY=S1. *HGEO2*=STREAM-PROP STREAM=2 PROPERTY=H1. *UEFF4*=PARAMETER 12. *DEGEO2*=PARAMETER 37. *UEFF5*=PARAMETER 13.

Fortran: <Comment> Base case for air cooler power require was 521 kw for 2200 kg/s of Air flow. This was for R134a for a standalone ACOL+ run. When I put it into A+ and using Aspen Plus properties and with air flow rule of thumb to cut the Tapp in half, the power load from ACOL+ increased by 5% With the adjustment. This simplifies to .55/2200 or 1/4000 </C>
ACC = MASAIR * (.55/2200)
EGEO = (MGEO*(HGEO1-HGEO0-(273.15+TDEAD)*(SGEO1-SGEO0)))
ESOL = (MSOL*(HSOL1-HSOL0-(273.15+TDEAD)*(SSOL1-SSOL0)))
DESOL = (MSOL*(HSOL1-HSOL2-(273.15+TDEAD)*(SSOL1-SSOL2)))
DEGEO = (MGEO*(HGEO1-HGEO9-(273.15+TDEAD)*(SGEO1-SGEO9)))
<Comment>The following code works when Solar Loop is used</C>
NETPOW = ((-1*ST)+(-1*BT)-CP-ACC-RP-GP) * 1000
UEFF = (NETPOW) / (EGEO + ESOL)
UEFF2 = (NETPOW) / (EGEO + DESOL)
UEFF3 = (NETPOW) / (DEGEO + DESOL)
<Comment>The following code works when the Solar Loop is bypassed</C>
NETPOW = ((-1*BT)-CP-ACC) * 1000
UEFF = (NETPOW) / (EGEO)
UEFF3 = (NETPOW) / (DEGEO)
THEFF = (NETPOW) / ((QHX1 + QHX2)*1000*1000/3600)
<Comment>The next few calculations are for effective utilization eff. As if the geothermal brine came out of the ground at the Temperature it is heated to by the sun (taking values of Geothermal stream 2). This is useful to compare the Effectiveness of the power-generating plant design to the super-critical design in Project 1. </C>
DEGEO2 = (MGEO*(HGEO2-HGEO9-(273.15+TDEAD)*(SGEO2-SGEO9)))
UEFF4 = (NETPOW) / (DEGEO2)
EGEO2 = (MGEO*(HGEO2-HGEO0-(273.15+TDEAD)*(SGEO2-SGEO0)))
UEFF5 = (NETPOW) / (EGEO2)

Sequence: READ-VARS *SGEO0 HGEO0 SGEO1 HGEO1 SSOL0 HSOL0 HSOL1 SSOL1 MGEO MSOL MASAIR TDEAD BT CP RP QHX1 QHX2 TOTPOW GP HSOL2 SSOL2 SGEO9 HGEO9 SGEO2 HGEO2 ST*. WRITE-VARS *UEFF THEFF NETPOW ACC EGEO ESOL UEFF2 UEFF3 DESOL DEGEO UEFF4 DEGEO2 UEFF5*.

FLOWSHEET OPTIONS>TRANSFER

A-A2MF: SET STREAM-VAR STREAM=A2 SUBSTREAM=MIXED VARIABLE=MASS-FLOW EQUAL-TO STREAM-VAR STREAM=A SUBSTREAM=MIXED VARIABLE=MASS-FLOW.
A-A2PR: SET STREAM-VAR STREAM=A2 SUBSTREAM=MIXED VARIABLE=PRES EQUAL-TO STREAM-VAR STREAM=A SUBSTREAM=MIXED VARIABLE=PRES.
A-A2TP: SET STREAM-VAR STREAM=A2 SUBSTREAM=MIXED VARIABLE=TEMP EQUAL-TO STREAM-VAR STREAM=A SUBSTREAM=MIXED VARIABLE=TEMP.
RP-OUTPR: SET BLOCK-VAR BLOCK=RP VARIABLE=PRES SENTENCE=PARAM EQUAL-TO STREAM-VAR STREAM=7 SUBSTREAM=MIXED VARIABLE=PRES.
T-2: SET STREAM FDUP EQUAL-TO STREAM F.
T-3: SET STREAM 5DUP EQUAL-TO STREAM 5.
T-3D: SET STREAM B4DUP EQUAL-TO STREAM B4.
T-4: SET BLOCK-VAR BLOCK=EDUP VARIABLE=TEMP SENTENCE=COLD-SIDE & ID1=FDUP EQUAL-TO STREAM-VAR STREAM=A. SUBSTREAM=MIXED VARIABLE=TEMP.

MODEL ANALYSIS TOOLS>SENSITIVITY

SENSTUDY:
Vary: (1) VARY BLOCK-VAR BLOCK=TV VARIABLE=P-OUT SENTENCE=PARAM RANGE LIST=15. (2) VARY BLOCK-VAR BLOCK=ST VARIABLE=PRES SENTENCE=PARAM RANGE LIST=3. (3) VARY BLOCK-VAR BLOCK=BHX VARIABLE=T-COLD SENTENCE=PARAM RANGE LIST=275.

MODEL ANALYSIS TOOLS>OPTIMIZATION

OPT-UEFF:
Define: UEFF PARAMETER 1 INIT-VAL=1.
Objective: MAXIMIZE "UEFF".

Vary: (1) PARAMETER 15 INIT-VAL=2. LIMITS "1.1" "3.5" STEP-SIZE=0.05 MAX-STEP-SIZ=0.1. (2) BLOCK-VAR BLOCK=HX2 VARIABLE=DELT-HOT SENTENCE=PARAM LIMITS "3" "40" STEP-SIZE=0.03 MAX-STEP-SIZ=0.05. (3) VARY PARAMETER 19 PHYS-QTY=DELTA-T UOM=C INIT-VAL=18. LIMITS "8" "40" STEP-SIZE=0.05 MAX-STEP-SIZ=0.1.

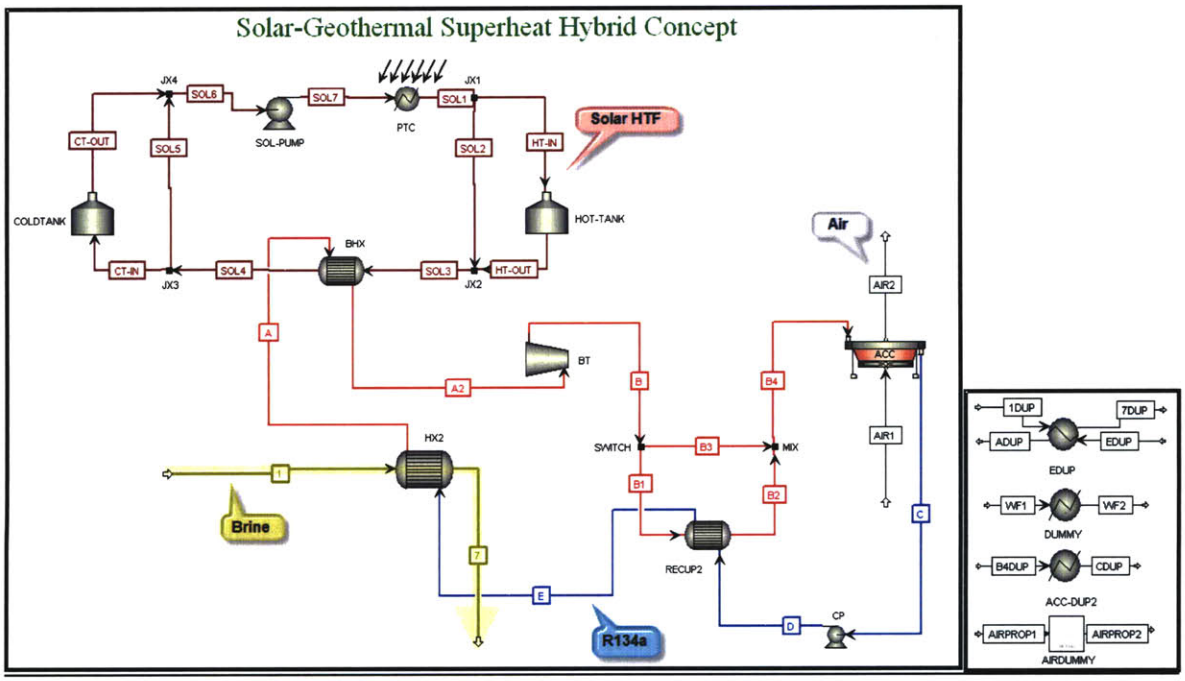
CONVERGENCE>CONV OPTIONS
METHODS>BROYDEN: X tolerance=5E-05.

SUPERHEAT STEADY-STATE MODEL

FILE NAME
"Phase 2 – SS Superheat Hybrid.bkp"

SOFTWARE
Aspen Plus V7.1

FLOWSHEET



SETUP>SPECIFICATIONS
Ambient Pressure = 1.01325 bar

SETUP>SIMULATION OPTIONS
Error tolerance: 1E-06

COMPONENTS >SPECIFICATIONS
WATER
R134A (1,1,1,2-TETRAFLUOROETHANE)
AIR
Therminol-VP-1

PROPERTIES>PROPERTY METHODS
BWRS: R134A streams
NRTL: Therminol-VP-1 streams
STEAMNBS: Water streams
IDEAL: Air streams

PROPERTIES>PROP-SETS
CPAIR: heat capacity, pure component (kJ/kg-K) at 20°C and 1 bar
CPMX: heat capacity, mixture (kJ/kg-K)
H0: Enthalpy, mixture (kJ/kmol) at 20°C
H1: Enthalpy, mixture (kJ/kmol)
PC: Critical Pressure, mixture
PDEW: Dew point pressure, mixture
SO: Entropy, mixture (kJ/kmol-K) at 20°C
S1: Entropy, mixture (kJ/kmol-K)
TDEW: Dew point temperature, mixture

STREAMS [INPUT][1,2]

[1]All Temperature is in °C, Pressure in bar, Mass flow rate in kg/sec unless otherwise written.
[2]Input values may be superseded with flowsheet options values.
STREAM 1: TEMP=150. PRES=20. MASS-FLOW=100. MASS-FRAC WATER=1.
STREAM 5DUP: TEMP=44. PRES=44. MASS-FLOW=50. MASS-FRAC WATER=1.
STREAM A2: TEMP=175.6. PRES=80. MASS-FLOW R134A=200.
STREAM AIR1: TEMP=20. PRES=1. MASS-FLOW=9301.69148. MASS-FRAC AIR=1.
STREAM AIRPROP1: TEMP=20. PRES=1. MASS-FLOW=5000. MASS-FRAC AIR=1.
STREAM B4: TEMP=40. VFRAC=1. MASS-FLOW=200. MASS-FRAC R134A=1.
STREAM B4DUP: TEMP=99. PRES=99. MASS-FLOW=99. MASS-FRAC WATER=1.
STREAM C: TEMP=30. VFRAC=0. MASS-FLOW=200. MASS-FRAC R134A=1.
STREAM D: TEMP=40. PRES=80. MASS-FLOW=200. MASS-FRAC R134A=1.
STREAM E: TEMP=40. PRES=80. MASS-FLOW=200. MASS-FRAC R134A=1.
STREAM FDUP: TEMP=44. PRES=44. MASS-FLOW=1. MASS-FLOW WATER=1.
STREAM SOL1: TEMP=698. <F> PRES=150. <psia> MASS-FLOW=300. MASS-FRAC THERMVP1=1.
STREAM WF1: TEMP=30. PRES=1. MASS-FLOW=1. MASS-FRAC R134A=1.

BLOCKS[1 2]
[1]All Temperature is in °C, Pressure in bar, Mass flow rate in kg/sec unless otherwise written.
[2]Input values may be superseded with flowsheet options values.
AIRDUMMY MIXER
JX2 MIXER: Property Method=NRTL. TEMP Estimate=400.
JX4 MIXER: Property Method=NRTL. TEMP Estimate=270.
MIX MIXER: Property Method=BWRS.
BYPASS1 FSPLIT: Property Method=STEAMNBS. FRAC 1B=0.
JX1 FSPLIT: Property Method=NRTL. FRAC SOL2=1.
JX3 FSPLIT: Property Method=NRTL. FRAC SOL5=1.
SWITCH FSPLIT: Property Method=BWRS. FRAC B1=0.
ACC-DUP2 HEATER: Property Method=BWRS. PRES=0. DEGSUB=2.
DUMMY HEATER: Property Method=BWRS. PRES=0. ΔTEMP=0.
PTC HEATER: Valid Phases=Liquid-Only. Property Method=NRTL. TEMP=390. PRES=-6. <psi>
COLDTANK FLASH2: Property Method=NRTL. TEMP=150. PRES=1.
HOT-TANK FLASH2: Property Method=NRTL. TEMP=150. PRES=1.
ACC HEATX: Property Methods=BWRS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: Hot out DEGSUB=2. PDROP=-0.2 (Hot)/-0.2 (Cold).
BHX HEATX: Property Methods=NRTL (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: Cold out TEMP=180. PDROP=-0.2 (Hot)/-0.2 (Cold).
HX2 HEATX: Property Methods=STEAMNBS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: (Hot out – Cold in)=5. PDROP=-0.2 (Hot)/-0.2 (Cold).
RECUP2 HEATX: Property Methods=BWRS (Hot)/BWRS (Cold). Calc Method=Shortcut. SPEC: (Hot out – Cold in)=5. PDROP=-0.2 (Hot)/-0.2 (Cold).
EDUP MHEATX: Property Methods=STEAMNBS (Hot)/BWRS (Cold). Valid Phases=Liquid-Only (Hot)/Liquid-Only (Cold). TEMP=190 (Cold).
CP PUMP: Property Method=BWRS. PRES=80. EFF=0.8.
SOL-PUMP PUMP: Property Method=NRTL. DELP=9. <psi> EFF=0.8.
BT COMPR: Property Method=BWRS. MODEL-TYPE=TURBINE. PRES=10. SEFF=0.85. MEFF=0.98.

FLOWSHEET OPTIONS>DESIGN SPEC[1]
[1]Fortran language is paraphrased here. Actual syntax may be different.
BAUMANN2:
Define: *VFRCA2*=STREAM-VAR STREAM=A2 SUBSTREAM=MIXED VARIABLE=VFRAC. *VFRCB*=STREAM-VAR STREAM=B SUBSTREAM=MIXED VARIABLE=VFRAC. *EFF*=BLOCK-VAR BLOCK=BT VARIABLE=SEFF SENTENCE=RESULTS.
Spec: SPEC "EFF" TO "TARGET" TOL-SPEC ".00001".
Vary: VARY BLOCK-VAR BLOCK=BT VARIABLE=SEFF SENTENCE=PARAM LIMITS ".1" "1" MAX-STEP-SIZ=0.1.
Fortran: <Comment>This Design Spec looks at the *Binary Turbine (BT)* and the streams going into and out of BT (A2 & B). The Baumann Rule penalizes the BT efficiency (linearly) if some liquid is present in the input or output streams. SOMETIMES THE SUPERCRITICAL FLUID (A) IS LABELED AS LIQUID. FOR THESE CASES, WE OVERRIDE IT WITH THE CSV (A2). </C>
IF (VFRCA2 .GT. .1) THEN
TARGET = 0.85 * (VFRCA2 + VFRCB)/2
ELSE
TARGET = 0.85 * (1.0 + VFRCB)/2
END IF
LMTD: (disabled)
Define: *DTLM*=BLOCK-VAR BLOCK=HX2 VARIABLE=DTLM SENTENCE=RESULTS. *TSPEC*=PARAMETER 5 PHYS-QTY=DELTA-T UOM="C" INIT-VAL=10. *GEOFLO*=STREAM-VAR STREAM=5 SUBSTREAM=MIXED VARIABLE=MASS-FLOW.
Spec: SPEC "DTLM" TO "TSPEC" TOL-SPEC ".0001".
Vary: VARY STREAM-VAR STREAM=E SUBSTREAM=MIXED VARIABLE=MASS-FLOW LIMITS "100" "800" MAX-STEP-SIZ=0.05.

FLOWSHEET OPTIONS>CALCULATOR[1]
[1]Fortran language is paraphrased here. Actual syntax may be different.
AIRFLOW:

Define: *MASAIR*=STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=MASS-FLOW. *QACC*=BLOCK-VAR BLOCK=ACC-DUP2 VARIABLE=QCALC SENTENCE=PARAM. *TAPP*=PARAMETER 19 PHYS-QTY=DELTA-T UOM="C" INIT-VAL=15.0395836. *CPAIR*=STREAM-PROP STREAM=AIRPROP1 PROPERTY=CPMX.
Fortran: MASAIR = -(QACC*1e6/3600) / (CPAIR * TAPP /2)
Sequence: READ-VARS *QACC TAPP CPAIR*. WRITE-VARS *MASAIR*.
LPCALC:
Define: *PDEW*=STREAM-PROP STREAM=WF1 PROPERTY=PDEW. *PTURB*=BLOCK-VAR BLOCK=BT VARIABLE=PRES SENTENCE=PARAM.
Fortran: PTURB = PDEW
Sequence: READ-VARS *PDEW*. WRITE-VARS *PTURB*. EXECUTE AFTER CALCULATOR SPECS.
PRSPEC:
Define: *PC*=STREAM-PROP STREAM=C PROPERTY=PC. *PRES5*=BLOCK-VAR BLOCK=CP VARIABLE=PRES SENTENCE=PARAM. *PR*=PARAMETER 15 INIT-VAL=1.21152988.
Fortran: PRES5 = PR * PC
Sequence: READ-VARS *PC PR*. WRITE-VARS *PRES5*.
RECUP:
Define: *TEMP3*=STREAM-VAR STREAM=B SUBSTREAM=MIXED VARIABLE=TEMP. *TEMP5*=STREAM-VAR STREAM=D SUBSTREAM=MIXED VARIABLE=TEMP. *SWITCH*=BLOCK-VAR BLOCK=SWITCH SENTENCE=FRAC VARIABLE=FRAC ID1=B1.
Fortran:
IF (TEMP3 .GT. TEMP5+10) THEN
SWITCH = 1D0
ELSE
SWITCH = 0D0
END IF
Sequence: READ-VARS *TEMP3 TEMP5*. WRITE-VARS *SWITCH*.
SPECS:
Define: *TDEADI*=PARAMETER 7 PHYS-QTY=TEMPERATURE UOM="C" INIT-VAL=20. *TAIR1*=STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=TEMP. *TDEW*=STREAM-VAR STREAM=WF1 SUBSTREAM=MIXED VARIABLE=TEMP. *TAPP*=PARAMETER 19 PHYS-QTY=DELTA-T UOM="C" INIT-VAL=15.0395836.
Fortran: TAIR1 = TDEADI
TDEW = TAIR1 + TAPP
Sequence: READ-VARS *TDEADI TAPP*. WRITE-VARS *TAIR1 TDEW*.
U-EFF:
Define: *SGEO0*=STREAM-PROP STREAM=1 PROPERTY=S0. *HGEO0*=STREAM-PROP STREAM=1 PROPERTY=H0. *SGEO1*=STREAM-PROP STREAM=1 PROPERTY=S1. *HGEO1*=STREAM-PROP STREAM=1 PROPERTY=H1. *SGEO7*=STREAM-PROP STREAM=7 PROPERTY=S1. *HGEO7*=STREAM-PROP STREAM=7 PROPERTY=H1. *SSOL0*=STREAM-PROP STREAM=SOL3 PROPERTY=S0. *HSOL0*=STREAM-PROP STREAM=SOL3 PROPERTY=H0. *SSOL1*=STREAM-PROP STREAM=SOL3 PROPERTY=S1. *HSOL1*=STREAM-PROP STREAM=SOL3 PROPERTY=H1. *SSOL2*=STREAM-PROP STREAM=SOL4 PROPERTY=S1. *HSOL2*=STREAM-PROP STREAM=SOL4 PROPERTY=H1. *MGEO*=STREAM-VAR STREAM=1 SUBSTREAM=MIXED VARIABLE=MOLE-FLOW. *MSOL*=STREAM-VAR STREAM=SOL1 SUBSTREAM=MIXED VARIABLE=MOLE-FLOW. *MASAIR*=STREAM-VAR STREAM=AIR1 SUBSTREAM=MIXED VARIABLE=MASS-FLOW. *UEFF*=PARAMETER 1 PHYS-QTY=DIMENSIONLES UOM="Unitless" INIT-VAL=1. *UEFF2*=PARAMETER 10 PHYS-QTY=DIMENSIONLES UOM= "Unitless" INIT-VAL=1. *UEFF3*=PARAMETER 11 PHYS-QTY=DIMENSIONLES UOM= "Unitless" INIT-VAL=1. *THEFF*=PARAMETER 2 PHYS-QTY=DIMENSIONLES UOM="Unitless" INIT-VAL=1. *TDEAD*=PARAMETER 7 PHYS-QTY=TEMPERATURE UOM="C" INIT-VAL=20. *NETPOW*=PARAMETER 3 PHYS-QTY=POWER UOM="kW" INIT-VAL=10000. *BT*=BLOCK-VAR BLOCK=BT VARIABLE=NET-WORK SENTENCE=RESULTS. *CP*=BLOCK-VAR BLOCK=CP VARIABLE=NET-WORK SENTENCE=RESULTS. *ACC*=PARAMETER 21 PHYS-QTY=POWER UOM="MW". *EGEO*=PARAMETER 33 PHYS-QTY=POWER UOM="kW". *DEGEO*=PARAMETER 36 PHYS-QTY=POWER UOM="kW". *ESOL*=PARAMETER 34 PHYS-QTY=POWER UOM="kW". *DESOL*=PARAMETER 35 PHYS-QTY=POWER UOM="kW". *QHX2*=BLOCK-VAR BLOCK=HX2 VARIABLE=CALC-DUTY SENTENCE=RESULTS. *TOTPOW*=PARAMETER 98. *QBHX*=BLOCK-VAR BLOCK=BHX VARIABLE=CALC-DUTY SENTENCE=RESULTS.
Fortran: <Comment> Base case for air cooler power require was 521 kw for 2200 kg/s of Air flow. This was for R134a for a standalone ACOL+ run. When I put it into A+ and using Aspen Plus properties and with air flow rule of thumb to cut the Tapp in half, the power load from ACOL+ increased by 5% With the adjustment. This simplifies to .55/2200 or 1/4000 </C>
ACC = MASAIR * (.55/2200)
NETPOW = ((-1*BT)-CP-ACC) * 1000
EGEO = (MGEO*(HGEO1-HGEO0-(273.15+TDEAD)*(SGEO1-SGEO0)))
ESOL = (MSOL*(HSOL1-HSOL0-(273.15+TDEAD)*(SSOL1-SSOL0)))
DESOL = (MSOL*(HSOL1-HSOL2-(273.15+TDEAD)*(SSOL1-SSOL2)))
DEGEO = (MGEO*(HGEO1-HGEO7-(273.15+TDEAD)*(SGEO1-SGEO7)))
UEFF = (NETPOW) / (EGEO + ESOL)
UEFF2 = (NETPOW) / (EGEO + DESOL)
UEFF3 = (NETPOW) / (DEGEO + DESOL)
THEFF = (NETPOW) / ((QHX2 + QBHX)*1000*1000/3600)
Sequence: READ-VARS *SGEO0 HGEO0 SGEO1 HGEO1 SSOL0 HSOL0 HSOL1 SSOL1 MGEO MSOL MASAIR TDEAD BT CP QHX2 TOTPOW HSOL2 SSOL2 SGEO7 HGEO7 QBHX*. WRITE-VARS *UEFF THEFF NETPOW ACC EGEO ESOL UEFF2 UEFF3 DESOL DEGEO*.

FLOWSHEET OPTIONS>TRANSFER
T-2: SET STREAM FDUP EQUAL-TO STREAM F
T-3: SET STREAM 5DUP EQUAL-TO STREAM 5
T-3D: SET STREAM B4DUP EQUAL-TO STREAM B4

T-4: SET BLOCK-VAR BLOCK=EDUP VARIABLE=TEMP SENTENCE=COLD-SIDE & ID1=FDUP EQUAL-TO STREAM-VAR STREAM=A SUBSTREAM=MIXED VARIABLE=TEMP

MODEL ANALYSIS TOOLS>SENSITIVITY
SENSTUDY:
Vary: (1) VARY STREAM-VAR STREAM=E SUBSTREAM=MIXED VARIABLE=MASS-FLOW RANGE LOWER="50" UPPER="400" INCR="50".

MODEL ANALYSIS TOOLS>OPTIMIZATION
OPT-UEFF:
Define: UEFF PARAMETER 1 INIT-VAL=1.
Objective: MAXIMIZE "UEFF".
Vary: (1) PARAMETER 15 INIT-VAL=2. LIMITS "1.1" "3.5" STEP-SIZE=0.05 MAX-STEP-SIZ=0.1. (2) BLOCK-VAR BLOCK=HX2 VARIABLE=DELT-HOT SENTENCE=PARAM LIMITS "3" "40" STEP-SIZE=0.03 MAX-STEP-SIZ=0.05. (3) VARY PARAMETER 19 PHYS-QTY=DELTA-T UOM=C INIT-VAL=18. LIMITS "8" "40" STEP-SIZE=0.05 MAX-STEP-SIZ=0.1.

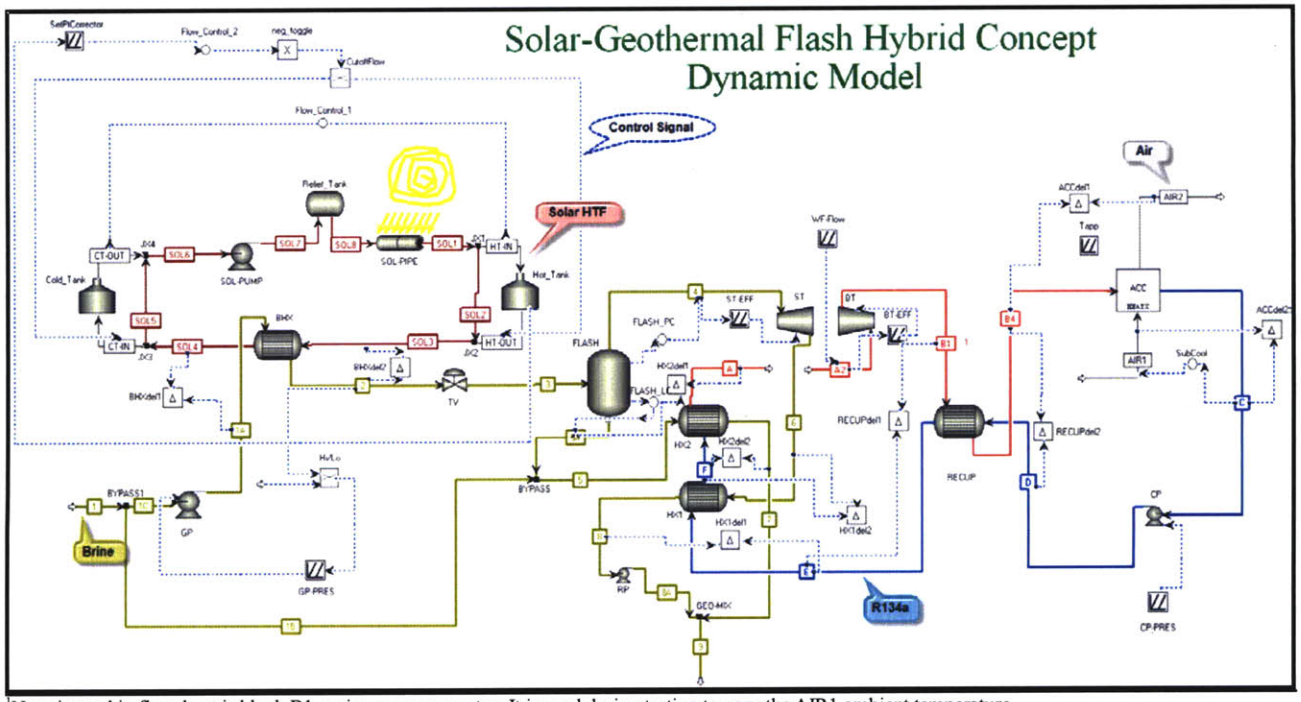
CONVERGENCE>CONV OPTIONS
METHODS>BROYDEN: X tolerance=5E-05.

FLASH-GEO DYNAMIC MODEL

FILE NAMES
"Phase 2 – Dynamic Flash Hybrid.dynf"
"Phase 2 – Dynamic Flash Hybrid Input File.appdf"

SOFTWARE
Aspen Plus Dynamics V7.1

FLOWSHEET[1]



[1]Not pictured in flowsheet is block B1, a sine wave generator. It is used during testing to vary the AIR1 ambient temperature

CHANGES TO COMPONENT VARIABLES (NON-DEFAULT)
FIXED:

Variable	Value	Spec	Description	Units
B1.PhaseAngle	210	Fixed	Phase angle	deg
RECUP.A_exch	1785.62	Fixed	Exchanger area	m2
BLOCKS("HX2").A_exch	2929.65	Fixed	Exchanger area	m2
BLOCKS("HX1").A_exch	1906.97	Fixed	Exchanger area	m2
BLOCKS("BHX").A_exch	472.68	Fixed	Exchanger area	m2
BLOCKS("ACC").A_exch	6224.1	Fixed	Exchanger area	m2
BLOCKS("HX2").uexp	0.6	Fixed	HTC power-law exponent	
BLOCKS("HX1").uexp	0.6	Fixed	HTC power-law exponent	
Tapp.Output1Min	12.56	Fixed	Minimum value of Output 1	
Tapp.Output1Max	15.0396	Fixed	Maximum value of Output 1	
Tapp.Output1InMin	150	Fixed	Value of input above which Output 1 starts to change	
Tapp.Output1InMax	275	Fixed	Value of input above which Output 1 stops changing	
SubCool.SPo	23	Fixed	Steady state set point	C
SubCool.OPss	1000	Fixed	Steady state manual output	kg/s
SubCool.OPmin	0	Fixed	Output range minimum	kg/s
SubCool.OPmax	20000	Fixed	Output range maximum	kg/s

SubCool.Gain	10	Fixed	Gain	
SetPtCorrector.Output1Min	0	Fixed	Minimum value of Output 1	
SetPtCorrector.Output1Max	205	Fixed	Maximum value of Output 1	
SetPtCorrector.Output1InMin	10	Fixed	Value of input above which Output 1 starts to change	m
SetPtCorrector.Output1InMax	10.1	Fixed	Value of input above which Output 1 stops changing	m
Flow_Control_2.SPo	0	Fixed	Steady state set point	
Flow_Control_2.PVmin	1.00E-03	Fixed	PV range minimum	
Flow_Control_2.PVmax	300	Fixed	PV range maximum	
Flow_Control_2.OPss	3.6	Fixed	Steady state manual output	
Flow_Control_2.OPmin	3.6	Fixed	Output range minimum	
Flow_Control_2.OPmax	720000	Fixed	Output range maximum	
Flow_Control_2.Gain	5	Fixed	Gain	
Flow_Control_1.SPRemote	205	Fixed	Remote setpoint	
Flow_Control_1.SPo	205	Fixed	Steady state set point	
Flow_Control_1.PVmin	1.00E-03	Fixed	PV range minimum	
Flow_Control_1.PVmax	300	Fixed	PV range maximum	
Flow_Control_1.OPss	1.00E-03	Fixed	Steady state manual output	kg/s
Flow_Control_1.OPmin	1.00E-03	Fixed	Output range minimum	kg/s
Flow_Control_1.OPmax	100	Fixed	Output range maximum	kg/s
Flow_Control_1.Gain	10	Fixed	Gain	
CutoffFlow.Input2	3.6	Fixed	Input signal 2	
Blocks("WF-Flow").Output2Min	133.59	Fixed	Minimum value of Output 2	kg/s
Blocks("WF-Flow").Output2Max	285.013	Fixed	Maximum value of Output 2	kg/s
Blocks("WF-Flow").Output2InMin	150	Fixed	Value of input above which Output 2 starts to change	
Blocks("WF-Flow").Output2InMax	275	Fixed	Value of input above which Output 2 stops changing	
Blocks("WF-Flow").Output1Min	8.18474	Fixed	Minimum value of Output 1	
Blocks("WF-Flow").Output1Max	9.05207	Fixed	Maximum value of Output 1	
Blocks("WF-Flow").Output1InMin	150	Fixed	Value of input above which Output 1 starts to change	
Blocks("WF-Flow").Output1InMax	275	Fixed	Value of input above which Output 1 stops changing	
Blocks("ST-EFF").Output1Min	0.6	Fixed	Minimum value of Output 1	
Blocks("ST-EFF").Output1Max	0.75	Fixed	Maximum value of Output 1	
Blocks("ST-EFF").Output1InMax	14	Fixed	Value of input above which Output 1 stops changing	kg/s
Blocks("Hi/Lo").Input2	150	Fixed	Input signal 2	C
Blocks("GP-PRES").Output1Max	550.732	Fixed	Maximum value of Output 1	kW
Blocks("GP-PRES").Output1InMin	150	Fixed	Value of input above which Output 1 starts to change	C
Blocks("GP-PRES").Output1InMax	275	Fixed	Value of input above which Output 1 stops changing	C
BLOCKS("FLASH_PC").SPRemote	14.4443	Fixed	Remote setpoint	bar
BLOCKS("FLASH_PC").SPo	15	Fixed	Steady state set point	bar
BLOCKS("FLASH_PC").PVmin	0	Fixed	PV range minimum	bar
BLOCKS("FLASH_PC").PVmax	31.073	Fixed	PV range maximum	bar
BLOCKS("FLASH_PC").OPss	3674.51	Fixed	Steady state manual output	kmol/hr
BLOCKS("FLASH_PC").OPmin	1.00E-03	Fixed	Output range minimum	kmol/hr
BLOCKS("FLASH_PC").OPmax	9000	Fixed	Output range maximum	kmol/hr
BLOCKS("FLASH_PC").Gain	20	Fixed	Gain	
BLOCKS("FLASH_LC").SPRemote	11.75	Fixed	Remote setpoint	m
BLOCKS("FLASH_LC").SPo	3	Fixed	Steady state set point	m

BLOCKS("FLASH_LC").PVmin	0	Fixed	PV range minimum	m
BLOCKS("FLASH_LC").PVmax	23.5	Fixed	PV range maximum	m
BLOCKS("FLASH_LC").OPmin	0	Fixed	Output range minimum	kg/hr
BLOCKS("FLASH_LC").OPmax	500000	Fixed	Output range maximum	kg/hr
BLOCKS("FLASH_LC").Gain	10	Fixed	Gain	
Blocks("CP-PRES").Output1Min	550	Fixed	Minimum value of Output 1	kW
Blocks("CP-PRES").Output1Max	1214.49	Fixed	Maximum value of Output 1	kW
Blocks("CP-PRES").Output1InMin	150	Fixed	Value of input above which Output 1 starts to change	
Blocks("CP-PRES").Output1InMax	275	Fixed	Value of input above which Output 1 stops changing	
Blocks("BT-EFF").Output1Min	0.8	Fixed	Minimum value of Output 1	
Blocks("BT-EFF").Output1Max	0.85	Fixed	Maximum value of Output 1	
Blocks("BT-EFF").Output1InMin	150	Fixed	Value of input above which Output 1 starts to change	kg/s
Blocks("BT-EFF").Output1InMax	285	Fixed	Value of input above which Output 1 stops changing	kg/s
B1.Datum	20	Fixed	Datum point for deviation	
B1.Amplitude	10	Fixed	Amplitude of deviation	
BLOCKS("HX2").EquipCp	0.5	Fixed	Specific heat capacity of exchanger material	kJ/kg/K
BLOCKS("HX1").EquipCp	0.5	Fixed	Specific heat capacity of exchanger material	kJ/kg/K
BLOCKS("BHX").EquipCp	0.5	Fixed	Specific heat capacity of exchanger material	kJ/kg/K
Hot_Tank.Qr	-150	Fixed	Specified heating duty	kW
Hot_Tank.QCoolr	0	Fixed	Specified cooling duty	kW
Cold_Tank.Qr	-150	Fixed	Specified heating duty	kW
Cold_Tank.QCoolr	0	Fixed	Specified cooling duty	GJ/hr
BLOCKS("FLASH").Qr	0	Fixed	Specified heating duty	GJ/hr
BLOCKS("HX2").Fm_ref	285	Fixed	Reference mass flow	kg/s
BLOCKS("HX1").Fm_ref	285	Fixed	Reference mass flow	kg/s
STREAMS("1").FmR	100	Fixed	Specified total mass flow	kg/s
SOL8.FmR	105	Fixed	Specified total mass flow	kg/s
SubCool.Cascade	1	Fixed	0=Use local SP, 1=Use remote SP	
Flow_Control_2.Cascade	1	Fixed	0=Use local SP, 1=Use remote SP	
Flow_Control_1.Cascade	0	Fixed	0=Use local SP, 1=Use remote SP	
Flow_Control_1.AutoMan	0	Fixed	0=Auto, 1=Manual, 2=Relay	
BYPASS1.sf("1C")	1	Fixed	Material stream split fraction	
BLOCKS("ST").Meff	0.98	Fixed	Mechanical efficiency	
BLOCKS("HX2").ft	1	Fixed	LMTD correction factor	
BLOCKS("HX1").ft	1	Fixed	LMTD correction factor	
BLOCKS("FLASH_PC").Cascade	0	Fixed	0=Use local SP, 1=Use remote SP	
BLOCKS("FLASH").LF_V	0	Fixed	Mole fraction of entrained liquid in vapor stream	
BLOCKS("BT").Meff	0.98	Fixed	Mechanical efficiency	
BLOCKS("BHX").ft	1	Fixed	LMTD correction factor	
BLOCKS("ACC").ft	1	Fixed	LMTD correction factor	
Solarflux	0	Fixed	Solar Flux from 0 - 1 kW/m2	W/m2
BLOCKS("HX2").U_ref	850	Fixed	Reference overall heat transfer coefficient	W/m2/K
BLOCKS("HX1").U_ref	850	Fixed	Reference overall heat transfer coefficient	W/m2/K
BLOCKS("BHX").U	850	Fixed	Overall heat transfer coefficient	W/m2/K
BLOCKS("ACC").U	850	Fixed	Overall heat transfer coefficient	W/m2/K

SolarArea	170000	Fixed	Solar Field area in sqm	m
Relief_Tank.L_Tank	10	Fixed	Reactor height	m
Relief_Tank.D_Tank	1.5	Fixed	Reactor diameter	m
Hot_Tank.L_Tank	55	Fixed	Reactor height	m
Hot_Tank.D_Tank	12	Fixed	Reactor diameter	m
Cold_Tank.L_Tank	45	Fixed	Reactor height	m
Cold_Tank.D_Tank	12	Fixed	Reactor diameter	m
Blocks("SOL-PIPE").L	10	Fixed	Pipe specified length	m
Blocks("SOL-PIPE").Diam	0.15	Fixed	Pipe inlet inner diameter	m
BLOCKS("FLASH").L_Tank	5	Fixed	Reactor height	m
BLOCKS("FLASH").D_Tank	1.8	Fixed	Reactor diameter	m
BLOCKS("HX2").MassOutHot	25000	Fixed	Hot side outlet mass	kg
BLOCKS("HX2").MassOutCold	25000	Fixed	Hot side outlet mass	kg
BLOCKS("HX2").MassInHot	25000	Fixed	Hot side inlet mass	kg
BLOCKS("HX2").MassInCold	25000	Fixed	Cold side inlet mass	kg
BLOCKS("HX1").MassOutHot	25000	Fixed	Hot side outlet mass	kg
BLOCKS("HX1").MassOutCold	25000	Fixed	Hot side outlet mass	kg
BLOCKS("HX1").MassInHot	25000	Fixed	Hot side inlet mass	kg
BLOCKS("HX1").MassInCold	25000	Fixed	Cold side inlet mass	kg
BLOCKS("BHX").MassOutHot	10000	Fixed	Hot side outlet mass	kg
BLOCKS("BHX").MassOutCold	1000	Fixed	Hot side outlet mass	kg
BLOCKS("BHX").MassInHot	10000	Fixed	Hot side inlet mass	kg
BLOCKS("BHX").MassInCold	10000	Fixed	Cold side inlet mass	kg
STREAMS("AIR1").ZmR("AIR")	1	Fixed	Specified mass fraction	kg/kg
STREAMS("1").ZmR("WATER")	1	Fixed	Specified mass fraction	kg/kg
GP.Eff	0.8	Fixed	Pump efficiency	
BLOCKS("SOL-PUMP").Meff	1	Fixed	Driver efficiency	
BLOCKS("SOL-PUMP").Eff	0.8	Fixed	Pump efficiency	
BLOCKS("RP").Meff	1	Fixed	Driver efficiency	
BLOCKS("RP").Eff	0.8	Fixed	Pump efficiency	
BLOCKS("CP").Meff	1	Fixed	Driver efficiency	
BLOCKS("CP").Eff	0.8	Fixed	Pump efficiency	
Relief_Tank.Pdrop	0	Fixed	Pressure drop	bar
Hot_Tank.Pdrop	0	Fixed	Pressure drop	bar
Cold_Tank.Pdrop	0	Fixed	Pressure drop	bar
BLOCKS("GEO-MIX").PdropR	0	Fixed	Specified pressure drop	bar
BLOCKS("FLASH").Pdrop	0	Fixed	Pressure drop	bar
Blocks("SOL-PIPE").pint(1).P_drop_in	6.57E-08	Fixed	Pressure drop at pipe inlet	bar
BLOCKS("TV").C0max	5554.18	Fixed	Maximum flow coefficient	m1.5 kg0.5/hr/bar0.5
BLOCKS("HX2").HotSide.C0	2.01E-09	Fixed	Pressure drop coefficient	bar hr2/kg/m3
BLOCKS("HX2").ColdSide.C0	1.38E-10	Fixed	Pressure drop coefficient	bar hr2/kg/m3
BLOCKS("HX1").HotSide.C0	8.15E-11	Fixed	Pressure drop coefficient	bar hr2/kg/m3
BLOCKS("HX1").ColdSide.C0	2.67E-10	Fixed	Pressure drop coefficient	bar hr2/kg/m3
BLOCKS("BHX").HotSide.C0	1.19E-11	Fixed	Pressure drop coefficient	bar hr2/kg/m3
BLOCKS("BHX").ColdSide.C0	1.42E-09	Fixed	Pressure drop coefficient	bar hr2/kg/m3
BLOCKS("ACC").HotSide.C0	7.08E-12	Fixed	Pressure drop coefficient	bar hr2/kg/m3
BLOCKS("ACC").ColdSide.C0	2.12E-16	Fixed	Pressure drop coefficient	bar hr2/kg/m3
STREAMS("AIR1").P	1	Fixed	Pressure	bar
STREAMS("1").P	20	Fixed	Pressure	bar
BLOCKS("ST").Pout	3	Fixed	Discharge pressure	bar
STREAMS("AIR1").T	15	Fixed	Temperature	C
STREAMS("1").T	150	Fixed	Temperature	C
Blocks("SOL-PIPE").pint(1).T_amb.Value(0)	20	Fixed	Variable value at a node point	C
SubCool.IntegralTime	20	Fixed	Integral time	min

SubCool.DerivTime	0	Fixed	Derivative time	min
Flow_Control_2.IntegralTime	20	Fixed	Integral time	min
Flow_Control_2.DerivTime	0	Fixed	Derivative time	min
Flow_Control_1.IntegralTime	60	Fixed	Integral time	min
Flow_Control_1.DerivTime	0	Fixed	Derivative time	min
BLOCKS("FLASH_PC").IntegralTime	12	Fixed	Integral time	min
BLOCKS("FLASH_PC").DerivTime	0	Fixed	Derivative time	min
BLOCKS("FLASH_LC").IntegralTime	60	Fixed	Integral time	min
BLOCKS("FLASH_LC").DerivTime	0	Fixed	Derivative time	min
B1.Period	24	Fixed	Period	hr
BLOCKS("HX2").VolOutHot	1	Fixed	Hot side outlet Volume	m3
BLOCKS("HX2").VolOutCold	1	Fixed	Cold side outlet Volume	m3
BLOCKS("HX2").VolInHot	1	Fixed	Hot side inlet Volume	m3
BLOCKS("HX2").VolInCold	1	Fixed	Cold side outlet Volume	m3
BLOCKS("HX1").VolOutHot	1	Fixed	Hot side outlet Volume	m3
BLOCKS("HX1").VolOutCold	1	Fixed	Cold side outlet Volume	m3
BLOCKS("HX1").VolInHot	1	Fixed	Hot side inlet Volume	m3
BLOCKS("HX1").VolInCold	1	Fixed	Cold side outlet Volume	m3
BLOCKS("BHX").VolOutHot	1	Fixed	Hot side outlet Volume	m3
BLOCKS("BHX").VolOutCold	1	Fixed	Cold side outlet Volume	m3
BLOCKS("BHX").VolInHot	1	Fixed	Hot side inlet Volume	m3
BLOCKS("BHX").VolInCold	1	Fixed	Cold side outlet Volume	m3

INITIALIZED:

Variable	Value	Spec	Description	Units
BLOCKS("FLASH").T	170.533	Initial	Temperature	C
Cold_Tank.T	160.143	Initial	Temperature	C
Hot_Tank.T	399.997	Initial	Temperature	C
Relief_Tank.T	163.062	Initial	Temperature	C
Cold_Tank.level	35	Initial	Liquid level	m
Hot_Tank.level	9.99833	Initial	Liquid level	m
Relief_Tank.level	4.75795	Initial	Liquid level	m
BLOCKS("BHX").ColdInVol(1).Mc("WATER")	50.9672	Initial	Component molar holdup	kmol
BLOCKS("BHX").ColdOutVol(1).Mc("WATER")	49.8664	Initial	Component molar holdup	kmol
BLOCKS("BHX").HotInVol(1).Mc("THERMVP1")	5.83031	Initial	Component molar holdup	kmol
BLOCKS("BHX").HotOutVol(1).Mc("THERMVP1")	6.06736	Initial	Component molar holdup	kmol
BLOCKS("FLASH").Mc("WATER")	362.442	Initial	Component molar holdup	kmol
BLOCKS("HX1").ColdInVol(1).Mc("R134A")	11.0282	Initial	Component molar holdup	kmol
BLOCKS("HX1").ColdOutVol(1).Mc("R134A")	11.0276	Initial	Component molar holdup	kmol
BLOCKS("HX1").HotInVol(1).Mc("WATER")	0.078926	Initial	Component molar holdup	kmol
BLOCKS("HX1").HotOutVol(1).Mc("WATER")	55.3785	Initial	Component molar holdup	kmol
BLOCKS("HX2").ColdInVol(1).Mc("R134A")	11.0276	Initial	Component molar holdup	kmol
BLOCKS("HX2").ColdOutVol(1).Mc("R134A")	2.53734	Initial	Component molar holdup	kmol
BLOCKS("HX2").HotInVol(1).Mc("WATER")	49.789	Initial	Component molar holdup	kmol
BLOCKS("HX2").HotOutVol(1).Mc("WATER")	53.8986	Initial	Component molar holdup	kmol
BLOCKS("BHX").ColdInVol(1).E	-13.3427	Initial	Internal energy	GJ
BLOCKS("BHX").ColdOutVol(1).E	-13.6252	Initial	Internal energy	GJ
BLOCKS("BHX").HotInVol(1).E	0.937113	Initial	Internal energy	GJ
BLOCKS("BHX").HotOutVol(1).E	0.643925	Initial	Internal energy	GJ
BLOCKS("GEO-MIX").Qcum	7.92E-19	Initial	Cumulative heating duty	GJ
BLOCKS("HX1").ColdInVol(1).E	-9.35865	Initial	Internal energy	GJ
BLOCKS("HX1").ColdOutVol(1).E	-9.35809	Initial	Internal energy	GJ
BLOCKS("HX1").HotInVol(1).E	1.48374	Initial	Internal energy	GJ
BLOCKS("HX1").HotOutVol(1).E	-15.5648	Initial	Internal energy	GJ
BLOCKS("HX2").ColdInVol(1).E	-9.35809	Initial	Internal energy	GJ

BLOCKS("HX2").ColdOutVol(1).E	-0.325824	Initial	Internal energy	GJ
BLOCKS("HX2").HotInVol(1).E	-11.5556	Initial	Internal energy	GJ
BLOCKS("HX2").HotOutVol(1).E	-14.1622	Initial	Internal energy	GJ
Cold_Tank.QCum	0	Initial	Cumulative heating duty	GJ
Hot_Tank.QCoolCum	0	Initial	Cumulative cooling duty	GJ
Hot_Tank.QCum	0	Initial	Cumulative heating duty	GJ
BLOCKS("FLASH_LC").OPMan	359995	Initial	Manual/initial output	kg/hr
BLOCKS("FLASH_LC").SP	3	Initial	Set point	m
BLOCKS("FLASH_PC").OPMan	-89.999	Initial	Manual/initial output	kmol/hr
BLOCKS("FLASH_PC").SP	15	Initial	Set point	bar
Flow_Control_1.OPMan	-0.99899	Initial	Manual/initial output	kg/s
Flow_Control_1.SP	205	Initial	Set point	
Flow_Control_2.OPMan	-7196.36	Initial	Manual/initial output	
Flow_Control_2.SP	-4.51E-262	Initial	Set point	
SubCool.OPMan	2993.73	Initial	Manual/initial output	kg/s
SubCool.SP	24.8584	Initial	Set point	C

UNFIXED & UNINITIALIZED:

Variable	Value	Spec	Description	Units
ACCdel1.Input1	27.6169	Free	Input signal 1	C
ACCdel1.Input2	24.7009	Free	Input signal 2	C
ACCdel2.Input1	24.2781	Free	Input signal 1	C
ACCdel2.Input2	15	Free	Input signal 2	C
BHXdel1.Input1	163.05	Free	Input signal 1	C
BHXdel1.Input2	150.113	Free	Input signal 2	C
BHXdel2.Input1	162.225	Free	Input signal 1	C
BHXdel2.Input2	170.303	Free	Input signal 2	C
Blocks("BT-EFF").Input_	158.463	Free	Input signal	kg/s
Blocks("CP-PRES").Input_	170.533	Free	Input signal	
Blocks("GP-PRES").Input_	170.303	Free	Input signal	C
Blocks("Hi/Lo").Input1	170.303	Free	Input signal 1	C
Blocks("ST-EFF").Input_	5.00E-06	Free	Input signal	kg/s
Blocks("WF-Flow").Input_	170.533	Free	Input signal	
CutoffFlow.Input1	-3.6	Free	Input signal 1	
Flow_Control_1.PV	-0.61116	Free	Process variable	
Flow_Control_2.PV	-0.609751	Free	Process variable	
Flow_Control_2.SPRemote	0	Free	Remote setpoint	
HX1del1.Input1	22.6708	Free	Input signal 1	C
HX1del1.Input2	64.079	Free	Input signal 2	C
HX1del2.Input1	133.555	Free	Input signal 1	C
HX1del2.Input2	55.4317	Free	Input signal 2	C
HX2del1.Input1	170.533	Free	Input signal 1	C
HX2del1.Input2	154.57	Free	Input signal 2	C
HX2del2.Input1	81.7889	Free	Input signal 1	C
HX2del2.Input2	55.4317	Free	Input signal 2	C
neg_toggle.Input1	3.6	Free	Input signal 1	
neg_toggle.Input2	-1	Free	Input signal 2	
RECUPdel1.Input1	83.1968	Free	Input signal 1	C
RECUPdel1.Input2	64.079	Free	Input signal 2	C
RECUPdel2.Input1	27.6169	Free	Input signal 1	C
RECUPdel2.Input2	26.8579	Free	Input signal 2	C
SetPtCorrector.Input_	9.99833	Free	Input signal	m
SetPtCorrector.Output2	0	Free	Output signal 2	
SubCool.PV	24.2781	Free	Process variable	C

SubCool.SPRemote	25.7409	Free	Remote setpoint	C
Tapp.Input_	170.533	Free	Input signal	
BLOCKS("ACC").Q	104.688	Free	Duty	GJ/hr
Cold_Tank.Qenv	0	Free	Rate of heat transfer to environment	GJ/hr
Hot_Tank.Qenv	0	Free	Rate of heat transfer to environment	kW
Hot_Tank.QReact	0	Free	Heat of reaction correction	GJ/hr
Streams("5A").FmR	359995	Free	Specified total mass flow	kg/hr
STREAMS("A2").FmcR("R134A")	158.463	Free	Specified component mass flow	kg/s
STREAMS("AIR1").FmR	2993.73	Free	Specified total mass flow	kg/s
Streams("CT-IN").FmR	1.00E-03	Free	Specified total mass flow	kg/s
Streams("CT-OUT").Fm	1.00E-03	Free	Total mass flow	kg/s
Streams("CT-OUT").FmR	1.00E-03	Free	Specified total mass flow	kg/s
Streams("HT-IN").Fm	1.00E-03	Free	Total mass flow	kg/s
Streams("HT-IN").FmR	1.00E-03	Free	Specified total mass flow	kg/s
Streams("HT-OUT").FmR	1.00E-03	Free	Specified total mass flow	kg/s
STREAMS("4").FR	1.00E-03	Free	Specified total molar flow	kmol/hr
BLOCKS("ACC").HotSide.vf	4.75E-21	Free	Outlet molar vapor fraction	
BLOCKS("BT").Ieff	0.803135	Free	Isentropic efficiency	
BLOCKS("ST").Ieff	0.829572	Free	Isentropic efficiency	
JX1.sf("HT-IN")	9.52E-06	Free	Material stream split fraction	
JX3.sf("CT-IN")	9.52E-06	Free	Material stream split fraction	
Blocks("SOL-PIPE").Q_flux_l	-17.248	Free	Heat flux	kW/m
BLOCKS("FLASH").E	-99.5833	Free	Internal energy	GJ
Cold_Tank.E	-681.902	Free	Internal energy	GJ
Hot_Tank.E	291.144	Free	Internal energy	GJ
Relief_Tank.E	-1.40093	Free	Internal energy	GJ
Cold_Tank.Mc("THERMVP1")	23393.4	Free	Component molar holdup	kmol
Hot_Tank.Mc("THERMVP1")	4482.59	Free	Component molar holdup	kmol
Relief_Tank.Mc("THERMVP1")	49.6735	Free	Component molar holdup	kmol
STREAMS("1").vfR	-0.745335	Free	Specified molar vapor fraction	
BLOCKS("TV").Pos	71.3902	Free	Specified valve position	%
BLOCKS("CP").EpowerR	659.151	Free	Specified electrical power	kW
BLOCKS("RP").EpowerR	0	Free	Specified electrical power	kW
BLOCKS("SOL-PUMP").Epower	2.3658	Free	Electrical power	kW
BLOCKS("SOL-PUMP").EpowerR	2.3658	Free	Specified electrical power	kW
GP.EpowerR	89.454	Free	Specified electrical power	kW
GP.PDiff	6.56895	Free	Pressure difference	bar
Hot_Tank.Pdropr	0	Free	Specified pressure drop	bar
BLOCKS("BT").Pout	7.24116	Free	Discharge pressure	bar
BLOCKS("CP").Pout	47.5346	Free	Discharge pressure	bar
BLOCKS("RP").Pout	7.7273	Free	Discharge pressure	bar
Blocks("SOL-PIPE").P	1.82021	Free	Outlet pressure	bar
BLOCKS("SOL-PUMP").Pout	2	Free	Discharge pressure	bar
BLOCKS("TV").P_o	8.01715	Free	Outlet pressure	bar
Cold_Tank.P	1.81845	Free	Pressure	bar
GP.Pout	26.5689	Free	Discharge pressure	bar
Hot_Tank.P	1.82021	Free	Pressure	bar
Relief_Tank.P	2	Free	Pressure	bar
STREAMS("A2").P	47.414	Free	Pressure	bar
Streams("B1").P	7.24116	Free	Pressure	bar
BLOCKS("ACC").HotSide.T	24.2781	Free	Outlet temperature	C
BLOCKS("ACC").T_out_hot	24.2781	Free	Hot side outlet port temperature	C
STREAMS("A2").T	154.57	Free	Temperature	C
STREAMS("C").T	24.2781	Free	Temperature	C

FLOWSHEET CODE

```
CONSTRAINTS
  // Flowsheet variables and equations...

//**************FLOW SHEET*****************************************************************************

//Continuity between streams 'A' and 'A2'
Streams("A").T = Streams("A2").T;
Streams("A").P = Streams("A2").P;

//RP pump output pressure set
RPpressure as pressure (description:"Pressure of str 7. Used to set Pout for RP");
RPpressure = streams("7").P;
blocks("RP").Pout = RPpressure;

//Bauman Equation for Steam Turbine
STEfficiency as RealVariable (description:"Bauman Equation Efficiency for Steam Turbine");
STEfficiency = 0.85* (blocks("ST").vf_in + blocks("ST").vf_out) / 2;
blocks("ST").Ieff = STEfficiency;

//**************SOLAR LOOP*****************************************************************************

//Solar Loop Enthalpies and controler PV. This logic determines how solar loop stores and releases fluid from storage.
SOL2_Qdot as enthflow (description:"heat flow of SOL2");
SOL3_Qdot as enthflow (description:"heat flow of SOL3");
SOL4_Qdot as enthflow (description:"heat flow of SOL4");
SOL2_Qdot = streams("SOL2").h * streams("SOL2").F;
SOL3_Qdot = streams("SOL3").h * streams("SOL3").F;
SOL4_Qdot = streams("SOL4").h * streams("SOL4").F;
Flow_Control_1.PV = SOL2_Qdot - SOL4_Qdot;
Flow_Control_2.PV = SOL3_Qdot - SOL4_Qdot;

//If hot_tank runs low on fluid, cut the output (actually, there is a controller that will decrease the output
//out of the hot tank as it approaches the lower level point so that the simulation can integrate properly.
if blocks("Hot_Tank").level < 10 then
blocks("neg_toggle").Input2 = -1;
else
blocks("neg_toggle").Input2 = 1;
endif

//Solar Field Variables
Solarflux as heat_flux (description:"Solar Flux from 0 - 1 kW/m2");
SolarArea as length (description:"Solar Field area in sqm"); //Although this is a length variable, you should enter area. This is b/c the solar pipe
model take kW/m
DeltaT as Temperature (description:"Difference between average Solar Field Temp and Ambient Temp");
DeltaT = blocks("SOL-PIPE").pint(1).T(5) - streams("AIR1").T;

//Heat Loss in pipes and HCE
PIPEloss as heat_flux (description:"Pipe Heat Loss in kW/m2");
PIPEloss = (10*((6.78*10^(-8))*(DeltaT)^3 - (1.683*10^(-5))*(DeltaT)^2 + (0.001693)*(DeltaT)))/1000;
HCEloss as heat_flux_lin (description:"HCE Heat Loss in W/m");
HCEloss = 0.141*(blocks("SOL-PIPE").pint(1).T(5)) + (6.48*10^(-9))*(blocks("SOL-PIPE").pint(1).T(5))^4;  //NREL Schott 2008 model year
PTR70 correlation

//Optical Losses in Solar Collectors
OptEfficiency as RealVariable (description:"Optical Efficiency of Solar Collectors due to several factors");
OptEfficiency = 0.754;

//TES losses programmed into Hot_Tank and Cold_Tank objects

//Total Losses on Solar Field
blocks("SOL-PIPE").Q_flux_l = (Solarflux*1000)*(SolarArea/10)*(OptEfficiency) - (HCEloss+(PIPEloss*1000*SolarArea/10));

//**************POWER CALCULATIONS*****************************************************************************

//Power Variables
ACCFanLoad as power (description:"ACC Fan Load Calculated by Mass of air through ACC and rule of thumb equation");
ACCFanLoad = streams("AIR1").Fm / 3600 * (.55/2200) * 1000 ; //Convert stream from kg/hr to kg/s, use rule of thumb equation to compare mass
of airflow to fan load, convert from MW to kW
BTPower as power (description:"Gross power from binary turbine");
```

```
BTPower = (-1*blocks("BT").Bpower); // convert to positive value
STPower as power (description:"Gross power from steam turbine");
STPower = (-1*blocks("ST").Bpower); // convert to positive value

//Parasitic Loads from Solar field as calculated in NREL's Solar Advisor Model
SolElecParasitic as power (description:"Electrical losses from electric or hydraulic tracking controllers and alarm monitoring devices in kW");
SolPumpParasitic as power (description:"Electrical losses from cold HTF pumping in the solar field in kW");
TESParasitic as power (description:"Electrical losses from pumps in the Thermal Energy Storage system in kW");

if blocks("SOL3").T > 150 then
SolElecParasitic = (2.66*10^(-7))*SolarArea;
SolPumpParasitic = (1.052*10^(-5))*SolarArea;
TESParasitic = 0.02 * 16.046; //16.046MW is the design turbine gross output
else
SolElecParasitic = 0;
SolPumpParasitic = 0;
TESParasitic = 0;
endif

//Net Power
NetPower as power (description:"Calculated net power, based on turbine power - (circulating power and ACC load parasitics)");
NetPower = (STPower + BTPower - blocks("CP").Bpower - blocks("GP").Bpower - blocks("RP").Bpower - ACCFanLoad - SolElecParasitic -
SolPumpParasitic - TESParasitic); // in kW

//**************CONTROLLER LOGIC*********************************************************************************

//R134a Saturation Temperature (C) from Pressure (bar) (curve fit range from 5-20bar) and vice versa
//Use this information for determining ACC subcooling
R134aPressure as pressure (description:"Take the pressure of stream B4");
R134aSatPres as pressure (description:"Saturation pressure of R134aTemperatureDew");
R134aTemperatureDew as temperature (description:"Dew Temperature measures as Tdead+Tapp");
R134aSatTemp as temperature (description:"Saturation Temperature of R134a Stream B4");
R134aPressure = streams("B4").P;
R134aSatTemp = 0.0054*(R134aPressure^3) - 0.3114*(R134aPressure^2) + 8.3877*(R134aPressure) - 18.718;
blocks("SubCool").SPRemote = R134aSatTemp - 2; //2 degrees subcooling

//R134a Saturation Pressure (bar) from Temperature (C) (curve fit range from 5-20bar)
//Use this information for determining BT Output pressure
R134aTemperatureDew = blocks("Tapp").Output1 + streams("AIR1").T;
R134aSatPres = 0.0029*(R134aTemperatureDew^2) + 0.0468*(R134aTemperatureDew) + 3.664;
streams("B1").P = R134aSatPres;

//Enthalpy method doesn't work well with the dynamic model (too shaky around flash tank). Instead use a proportional "Effective Temperature"
//to model how much heat is delivered from the flash tank and use to relate to controllers.
FlashEffectiveTemperature as Temperature (description:"Attempt to compare energy coming from flash tank to energy going in. Temperature is a
smooth variable to follow and run controllers against");
FlashEffectiveTemperature = streams("5").T + (streams("4").Fm*4.018194384/3600);
blocks("WF-Flow").Input_ = FlashEffectiveTemperature;
blocks("Tapp").Input_ = FlashEffectiveTemperature;
blocks("CP-PRES").Input_ = FlashEffectiveTemperature;

END
```

FLOWSHEET SCRIPTS (Forcing Functions)

Test Day:

```
//*************************************************************************************************************
Task Test_Day runs at 6
parallel
//sramp(streams("AIR1").T , 40, 3); // ambient air temperature
sramp(Solarflux, 1, 3); // solar insolation
endparallel;
wait 8;
parallel
//sramp(streams("AIR1").T, 8, 3); // ambient air temperature
sramp(Solarflux, 0, 3); // solar insolation
endparallel;
wait 10;
parallel
//sramp(streams("AIR1").T , 40, 3); // ambient air temperature
sramp(Solarflux, 1, 3); // solar insolation
endparallel;
wait 8;
```

```
parallel
//sramp(streams("AIR1").T, 8, 3); // ambient air temperature
sramp(Solarflux, 0, 3); // solar insolation
endparallel;
End

Jan_15_2009: //*****************************************************************************************************
Task Jan_15_2009 runs at 4      //run this data set several times if your model goes beyond 28 hours
parallel sramp(streams("AIR1").T ,      8.53    , .05); sramp(Solarflux,        0.00011529      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.51    , .05); sramp(Solarflux,        0.00048828      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.68    , .05); sramp(Solarflux,        0.00082736      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.66    , .05); sramp(Solarflux,        0.00098672      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.71    , .05); sramp(Solarflux,        0.00037299      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.61    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.5     , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.4     , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.36    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.37    , .05); sramp(Solarflux,        0.00026109      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.48    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.22    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.98    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.81    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.65    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.7     , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.84    , .05); sramp(Solarflux,        0.0010817 , .05); endparallel;
parallel sramp(streams("AIR1").T ,      8.02    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.46    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.986   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.864   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.81    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.859   , .05); sramp(Solarflux,        0.00021701      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.854   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.692   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.536   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.427   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.309   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.314   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.337   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.468   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.466   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.376   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.295   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.261   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.143   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.293   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.293   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.221   , .05); sramp(Solarflux,        0.00001356      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.582   , .05); sramp(Solarflux,        0.0024481 , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.51    , .05); sramp(Solarflux,        0.0028448 , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.83    , .05); sramp(Solarflux,        0.0011427 , .05); endparallel;
parallel sramp(streams("AIR1").T ,      7.19    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.621   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.34    , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.028   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.502   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.397   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.57    , .05); sramp(Solarflux,        0.00049843      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.869   , .05); sramp(Solarflux,        0.0012579 , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.06    , .05); sramp(Solarflux,        0.00068492      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.541   , .05); sramp(Solarflux,        0.0023803 , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.701   , .05); sramp(Solarflux,        0.00039332      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.884   , .05); sramp(Solarflux,        0.00037976      , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.936   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.953   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.549   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      6.011   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.856   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.651   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.526   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.425   , .05); sramp(Solarflux,        0       , .05); endparallel;
parallel sramp(streams("AIR1").T ,      5.313   , .05); sramp(Solarflux,        0       , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  5.784  , .05); sramp(Solarflux,  0.00026108  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.927  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.05  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.699  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.373  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.037  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.754  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.736  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.74  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.84  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.868  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.595  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.657  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.778  , .05); sramp(Solarflux,  0.0012207  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.801  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.716  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.671  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.594  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.539  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.548  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.553  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.376  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.188  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.235  , .05); sramp(Solarflux,  0.00068492  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.246  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.58  , .05); sramp(Solarflux,  0.0015733  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.258  , .05); sramp(Solarflux,  0.0059371  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.834  , .05); sramp(Solarflux,  0.0040688  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.099  , .05); sramp(Solarflux,  0.0021633  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.215  , .05); sramp(Solarflux,  0.0023498  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.259  , .05); sramp(Solarflux,  0.00038993  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.347  , .05); sramp(Solarflux,  0.0011698  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.621  , .05); sramp(Solarflux,  0.00064423  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.642  , .05); sramp(Solarflux,  0.00004747  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.574  , .05); sramp(Solarflux,  0.00046792  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.505  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.507  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.371  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.048  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.105  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.103  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.086  , .05); sramp(Solarflux,  0.0008409  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.162  , .05); sramp(Solarflux,  0.00055269  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.292  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.272  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.004  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.638  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.159  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.645  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.425  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.214  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.046  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.918  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.667  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.532  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.705  , .05); sramp(Solarflux,  0.00068155  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.988  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.186  , .05); sramp(Solarflux,  0.00044081  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.332  , .05); sramp(Solarflux,  0.00076633  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.528  , .05); sramp(Solarflux,  0.00097317  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.997  , .05); sramp(Solarflux,  0.0016378  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.486  , .05); sramp(Solarflux,  0.0020006  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.863  , .05); sramp(Solarflux,  0.0014716  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.222  , .05); sramp(Solarflux,  0.0023397  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.613  , .05); sramp(Solarflux,  0.0019464  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.889  , .05); sramp(Solarflux,  0.0013089  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  7.04  , .05); sramp(Solarflux,  0.0023465  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.669  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.04  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.791  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.817  , .05); sramp(Solarflux,  0  , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  5.672  , .05); sramp(Solarflux,  0         , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.138  , .05); sramp(Solarflux,  0         , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.671  , .05); sramp(Solarflux,  0         , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.638  , .05); sramp(Solarflux,  0         , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.454  , .05); sramp(Solarflux,  0         , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.238  , .05); sramp(Solarflux,  0         , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.85   , .05); sramp(Solarflux,  0.010092  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.618  , .05); sramp(Solarflux,  0.24402   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.557  , .05); sramp(Solarflux,  0.2974    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.573  , .05); sramp(Solarflux,  0.3279    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.686  , .05); sramp(Solarflux,  0.35871   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  3.735  , .05); sramp(Solarflux,  0.31071   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.184  , .05); sramp(Solarflux,  0.27703   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.67   , .05); sramp(Solarflux,  0.34439   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  4.944  , .05); sramp(Solarflux,  0.481     , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.074  , .05); sramp(Solarflux,  0.49093   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.297  , .05); sramp(Solarflux,  0.52096   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.385  , .05); sramp(Solarflux,  0.51769   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.46   , .05); sramp(Solarflux,  0.56583   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.624  , .05); sramp(Solarflux,  0.57295   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  5.837  , .05); sramp(Solarflux,  0.60513   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.234  , .05); sramp(Solarflux,  0.6251    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.583  , .05); sramp(Solarflux,  0.64064   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  6.769  , .05); sramp(Solarflux,  0.65452   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  7.03   , .05); sramp(Solarflux,  0.66897   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  7.32   , .05); sramp(Solarflux,  0.68244   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  7.52   , .05); sramp(Solarflux,  0.6959    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  7.68   , .05); sramp(Solarflux,  0.70629   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  7.88   , .05); sramp(Solarflux,  0.71483   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  8.01   , .05); sramp(Solarflux,  0.72331   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  8.17   , .05); sramp(Solarflux,  0.73693   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  8.2    , .05); sramp(Solarflux,  0.74691   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  8.3    , .05); sramp(Solarflux,  0.75483   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  8.49   , .05); sramp(Solarflux,  0.76149   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  8.99   , .05); sramp(Solarflux,  0.77223   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  9.16   , .05); sramp(Solarflux,  0.77528   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  9.37   , .05); sramp(Solarflux,  0.78304   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  9.79   , .05); sramp(Solarflux,  0.78931   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10     , .05); sramp(Solarflux,  0.79307   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.19  , .05); sramp(Solarflux,  0.79898   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.45  , .05); sramp(Solarflux,  0.80889   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.51  , .05); sramp(Solarflux,  0.8164    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.75  , .05); sramp(Solarflux,  0.82473   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.85  , .05); sramp(Solarflux,  0.82805   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.97  , .05); sramp(Solarflux,  0.83137   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11     , .05); sramp(Solarflux,  0.83854   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.88  , .05); sramp(Solarflux,  0.8428    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.06  , .05); sramp(Solarflux,  0.84865   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.38  , .05); sramp(Solarflux,  0.8518    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.73  , .05); sramp(Solarflux,  0.85576   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.63  , .05); sramp(Solarflux,  0.84602   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.64  , .05); sramp(Solarflux,  0.86319   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.82  , .05); sramp(Solarflux,  0.86365   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.04  , .05); sramp(Solarflux,  0.85311   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.9   , .05); sramp(Solarflux,  0.86432   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.12  , .05); sramp(Solarflux,  0.86928   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.41  , .05); sramp(Solarflux,  0.87235   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.79  , .05); sramp(Solarflux,  0.87467   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.73  , .05); sramp(Solarflux,  0.88029   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.07  , .05); sramp(Solarflux,  0.88906   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.58  , .05); sramp(Solarflux,  0.89496   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.02  , .05); sramp(Solarflux,  0.89768   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.87  , .05); sramp(Solarflux,  0.89386   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.14  , .05); sramp(Solarflux,  0.89336   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.14  , .05); sramp(Solarflux,  0.88881   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.48  , .05); sramp(Solarflux,  0.88862   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.69  , .05); sramp(Solarflux,  0.897     , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.57  , .05); sramp(Solarflux,  0.90822   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.13  , .05); sramp(Solarflux,  0.91299   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.49  , .05); sramp(Solarflux,  0.91426   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.37  , .05); sramp(Solarflux,  0.92236   , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  15.51  , .05); sramp(Solarflux,  0.92681  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.65  , .05); sramp(Solarflux,  0.93185  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.35  , .05); sramp(Solarflux,  0.92733  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.3   , .05); sramp(Solarflux,  0.91961  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.27  , .05); sramp(Solarflux,  0.92239  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.66  , .05); sramp(Solarflux,  0.92845  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.98  , .05); sramp(Solarflux,  0.92362  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.28  , .05); sramp(Solarflux,  0.92939  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.68  , .05); sramp(Solarflux,  0.92515  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.15  , .05); sramp(Solarflux,  0.91577  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.43  , .05); sramp(Solarflux,  0.92845  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.48  , .05); sramp(Solarflux,  0.92868  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.08  , .05); sramp(Solarflux,  0.92917  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.14  , .05); sramp(Solarflux,  0.93476  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.44  , .05); sramp(Solarflux,  0.93379  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.52  , .05); sramp(Solarflux,  0.9378   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.78  , .05); sramp(Solarflux,  0.93823  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.16  , .05); sramp(Solarflux,  0.94275  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.16  , .05); sramp(Solarflux,  0.94871  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.44  , .05); sramp(Solarflux,  0.94951  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.87  , .05); sramp(Solarflux,  0.94779  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.58  , .05); sramp(Solarflux,  0.9426   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.93  , .05); sramp(Solarflux,  0.93909  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.39  , .05); sramp(Solarflux,  0.93905  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.65  , .05); sramp(Solarflux,  0.94121  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.72  , .05); sramp(Solarflux,  0.9445   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.97  , .05); sramp(Solarflux,  0.94558  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.43  , .05); sramp(Solarflux,  0.94303  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.55  , .05); sramp(Solarflux,  0.94492  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.39  , .05); sramp(Solarflux,  0.94878  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.51  , .05); sramp(Solarflux,  0.95666  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.57  , .05); sramp(Solarflux,  0.95406  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.67  , .05); sramp(Solarflux,  0.95605  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.23  , .05); sramp(Solarflux,  0.95203  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.7   , .05); sramp(Solarflux,  0.95367  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.92  , .05); sramp(Solarflux,  0.95254  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.93  , .05); sramp(Solarflux,  0.94943  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19     , .05); sramp(Solarflux,  0.95073  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.95  , .05); sramp(Solarflux,  0.95189  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.98  , .05); sramp(Solarflux,  0.95125  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.24  , .05); sramp(Solarflux,  0.95004  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.37  , .05); sramp(Solarflux,  0.94769  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.36  , .05); sramp(Solarflux,  0.94801  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.38  , .05); sramp(Solarflux,  0.9447   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.55  , .05); sramp(Solarflux,  0.94396  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.32  , .05); sramp(Solarflux,  0.94213  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.31  , .05); sramp(Solarflux,  0.93932  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.54  , .05); sramp(Solarflux,  0.93775  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.46  , .05); sramp(Solarflux,  0.93886  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.53  , .05); sramp(Solarflux,  0.93538  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.56  , .05); sramp(Solarflux,  0.93137  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.51  , .05); sramp(Solarflux,  0.93056  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.54  , .05); sramp(Solarflux,  0.92776  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.6   , .05); sramp(Solarflux,  0.9268   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.87  , .05); sramp(Solarflux,  0.92772  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.09  , .05); sramp(Solarflux,  0.9294   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.12  , .05); sramp(Solarflux,  0.92481  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.02  , .05); sramp(Solarflux,  0.92007  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.95  , .05); sramp(Solarflux,  0.92037  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.04  , .05); sramp(Solarflux,  0.92005  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.04  , .05); sramp(Solarflux,  0.91641  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.26  , .05); sramp(Solarflux,  0.91281  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.25  , .05); sramp(Solarflux,  0.91279  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.63  , .05); sramp(Solarflux,  0.91633  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.53  , .05); sramp(Solarflux,  0.91427  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.47  , .05); sramp(Solarflux,  0.91277  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.41  , .05); sramp(Solarflux,  0.90821  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.38  , .05); sramp(Solarflux,  0.90522  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.5   , .05); sramp(Solarflux,  0.91266  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.53  , .05); sramp(Solarflux,  0.91027  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.72  , .05); sramp(Solarflux,  0.90512  , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  20.63  , .05); sramp(Solarflux,  0.89857  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.88  , .05); sramp(Solarflux,  0.89777  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.9  , .05); sramp(Solarflux,  0.89913  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.87  , .05); sramp(Solarflux,  0.89582  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.98  , .05); sramp(Solarflux,  0.89662  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.11  , .05); sramp(Solarflux,  0.89738  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.21  , .05); sramp(Solarflux,  0.89002  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.4  , .05); sramp(Solarflux,  0.89049  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.39  , .05); sramp(Solarflux,  0.88757  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.45  , .05); sramp(Solarflux,  0.8833  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.39  , .05); sramp(Solarflux,  0.88354  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.61  , .05); sramp(Solarflux,  0.87063  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.48  , .05); sramp(Solarflux,  0.86799  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.47  , .05); sramp(Solarflux,  0.87065  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.48  , .05); sramp(Solarflux,  0.86683  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.41  , .05); sramp(Solarflux,  0.85856  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.39  , .05); sramp(Solarflux,  0.85416  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.65  , .05); sramp(Solarflux,  0.84628  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.59  , .05); sramp(Solarflux,  0.84349  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.61  , .05); sramp(Solarflux,  0.84315  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.58  , .05); sramp(Solarflux,  0.83614  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.58  , .05); sramp(Solarflux,  0.83091  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.46  , .05); sramp(Solarflux,  0.82676  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.36  , .05); sramp(Solarflux,  0.8184  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.46  , .05); sramp(Solarflux,  0.81194  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.43  , .05); sramp(Solarflux,  0.80481  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.37  , .05); sramp(Solarflux,  0.79967  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.38  , .05); sramp(Solarflux,  0.79192  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.31  , .05); sramp(Solarflux,  0.78423  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.3  , .05); sramp(Solarflux,  0.77353  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.37  , .05); sramp(Solarflux,  0.77003  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.26  , .05); sramp(Solarflux,  0.76177  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.15  , .05); sramp(Solarflux,  0.7529  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.16  , .05); sramp(Solarflux,  0.74244  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.16  , .05); sramp(Solarflux,  0.7348  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.19  , .05); sramp(Solarflux,  0.72041  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.16  , .05); sramp(Solarflux,  0.7113  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.15  , .05); sramp(Solarflux,  0.69775  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.11  , .05); sramp(Solarflux,  0.6851  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.04  , .05); sramp(Solarflux,  0.67355  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.04  , .05); sramp(Solarflux,  0.65477  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21.05  , .05); sramp(Solarflux,  0.64393  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  21  , .05); sramp(Solarflux,  0.62442  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.98  , .05); sramp(Solarflux,  0.60637  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.96  , .05); sramp(Solarflux,  0.59  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.95  , .05); sramp(Solarflux,  0.57319  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.91  , .05); sramp(Solarflux,  0.55875  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.87  , .05); sramp(Solarflux,  0.53413  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.82  , .05); sramp(Solarflux,  0.51501  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.73  , .05); sramp(Solarflux,  0.48702  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.62  , .05); sramp(Solarflux,  0.46929  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.58  , .05); sramp(Solarflux,  0.44019  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.53  , .05); sramp(Solarflux,  0.41122  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.5  , .05); sramp(Solarflux,  0.363  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.41  , .05); sramp(Solarflux,  0.21923  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.28  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.13  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.02  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  20.01  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.92  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.74  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.57  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.54  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.39  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.27  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  19.02  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.89  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.69  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.42  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  18.1  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.66  , .05); sramp(Solarflux,  0  , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  17.5  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.48  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.6  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.58  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.55  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.52  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.55  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.51  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.47  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.24  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.25  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.41  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.24  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.91  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.62  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.44  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.24  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.94  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.77  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.61  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.4  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.13  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.97  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.7  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.78  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.09  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.23  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.44  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.79  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.89  , .05); sramp(Solarflux,  0.00018307  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.27  , .05); sramp(Solarflux,  0.0016646  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.58  , .05); sramp(Solarflux,  0.0024648  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.69  , .05); sramp(Solarflux,  0.0017629  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.65  , .05); sramp(Solarflux,  0.00019664  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.6  , .05); sramp(Solarflux,  0.0013087  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.83  , .05); sramp(Solarflux,  0.0025632  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.13  , .05); sramp(Solarflux,  0.002136  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  17.02  , .05); sramp(Solarflux,  0.0013257  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.9  , .05); sramp(Solarflux,  0.0010477  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.51  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.26  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  16.22  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.61  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.22  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.05  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.29  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.64  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.66  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.34  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  15.01  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.66  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.45  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.6  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.46  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14.12  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  14  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.89  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.83  , .05); sramp(Solarflux,  0.00026787  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.82  , .05); sramp(Solarflux,  0.00038654  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.65  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.64  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.38  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.65  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.53  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.74  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.47  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.29  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.08  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.07  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.93  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.81  , .05); sramp(Solarflux,  0  , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  12.73  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.14  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  13.05  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.95  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.82  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.31  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.07  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.53  , .05); sramp(Solarflux,  0.00011868  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.18  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.11  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.11  , .05); sramp(Solarflux,  0.00021362  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.26  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.05  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.34  , .05); sramp(Solarflux,  0.00088841  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.53  , .05); sramp(Solarflux,  0.00037978  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.34  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.72  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.15  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.71  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.75  , .05); sramp(Solarflux,  0.00071208  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.17  , .05); sramp(Solarflux,  0.00036621  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.41  , .05); sramp(Solarflux,  0.00068834  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.74  , .05); sramp(Solarflux,  0.00083754  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.75  , .05); sramp(Solarflux,  0.00065105  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.6  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.42  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.24  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.12  , .05); sramp(Solarflux,  0.00028483  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.25  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.38  , .05); sramp(Solarflux,  0.0011597 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.38  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.44  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.35  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.24  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.14  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.09  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.96  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.78  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.74  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.8  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.07  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.55  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.51  , .05); sramp(Solarflux,  0.0021532 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.88  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  12.2  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.9  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.55  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.27  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.08  , .05); sramp(Solarflux,  0.00087482  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.1  , .05); sramp(Solarflux,  0.0015292 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.24  , .05); sramp(Solarflux,  0.00032891  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.37  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.29  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.26  , .05); sramp(Solarflux,  0.00008816  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.22  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.15  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.62  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.44  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.45  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.3  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  11.12  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  10.85  , .05); sramp(Solarflux,  0  , .05); endparallel;
End
```

July_15_2009: //**********

```
Task July_15_2009 runs at 4 //run this data set several times if your model goes beyond 28 hours
parallel sramp(streams("AIR1").T ,  32.63  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.46  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.31  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.53  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.18  , .05); sramp(Solarflux,  0.001433  , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  33.7  , .05); sramp(Solarflux,  0.0015075 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.97  , .05); sramp(Solarflux,  0.00077241  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.08  , .05); sramp(Solarflux,  0.00040314  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.09  , .05); sramp(Solarflux,  0.00034217  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.18  , .05); sramp(Solarflux,  0.00042686  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.17  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.17  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.15  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.18  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.15  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.11  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.09  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.14  , .05); sramp(Solarflux,  0.00065723  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.23  , .05); sramp(Solarflux,  0.00029813  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.33  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.28  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.15  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.03  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.91  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.79  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.74  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.76  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.84  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.84  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.77  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.67  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.38  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.82  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.42  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.41  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.15  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.96  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.06  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.26  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.58  , .05); sramp(Solarflux,  0.00084696  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.58  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.49  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.34  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.17  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.17  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.17  , .05); sramp(Solarflux,  0.0010875 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.11  , .05); sramp(Solarflux,  0.001843  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.01  , .05); sramp(Solarflux,  0.0014094 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.9  , .05); sramp(Solarflux,  0.00043704  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.78  , .05); sramp(Solarflux,  0.00058611  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.7  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.62  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.56  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.49  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.44  , .05); sramp(Solarflux,  0.00092151  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.26  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.04  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.97  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.71  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.94  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.6  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.66  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.08  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.36  , .05); sramp(Solarflux,  0.0010333 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.73  , .05); sramp(Solarflux,  0.00044382  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.8  , .05); sramp(Solarflux,  0.0021073 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.7  , .05); sramp(Solarflux,  0.0022123 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.66  , .05); sramp(Solarflux,  0.0011451 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.59  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.45  , .05); sramp(Solarflux,  0.0019921 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.23  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.04  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.84  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.78  , .05); sramp(Solarflux,  0.00057934  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.79  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.84  , .05); sramp(Solarflux,  0.0010706 , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  29.75  , .05); sramp(Solarflux,  0.00038623  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.65  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.69  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.84  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.86  , .05); sramp(Solarflux,  0.00026766  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.72  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.51  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.32  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.13  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.24  , .05); sramp(Solarflux,  0.0001152 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  28.94  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  28.37  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  28.22  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  27.87  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  28.13  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  28.13  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  28.93  , .05); sramp(Solarflux,  0.0025718 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.9  , .05); sramp(Solarflux,  0.0028937 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.29  , .05); sramp(Solarflux,  0.00043031  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.87  , .05); sramp(Solarflux,  0.00045743  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.72  , .05); sramp(Solarflux,  0.00050147  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.9  , .05); sramp(Solarflux,  0.00095212  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.43  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.49  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.29  , .05); sramp(Solarflux,  0.0012028 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.23  , .05); sramp(Solarflux,  0.0017247 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  29.79  , .05); sramp(Solarflux,  0.001396  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  30.58  , .05); sramp(Solarflux,  0.0027988 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.21  , .05); sramp(Solarflux,  0.0024261 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  31.72  , .05); sramp(Solarflux,  0.0019619 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  32.73  , .05); sramp(Solarflux,  0.002399  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  33.56  , .05); sramp(Solarflux,  0.0019246 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.26  , .05); sramp(Solarflux,  0.0019111 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.71  , .05); sramp(Solarflux,  0.00060655  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.04  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.33  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.7  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.76  , .05); sramp(Solarflux,  0.00089794  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.51  , .05); sramp(Solarflux,  0.00077593  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.52  , .05); sramp(Solarflux,  0.0016704 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.59  , .05); sramp(Solarflux,  0.00087758  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.28  , .05); sramp(Solarflux,  0.00052857  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.45  , .05); sramp(Solarflux,  0.0000305 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.49  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.26  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  34.92  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  35.44  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  36.55  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.21  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.89  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.51  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.43  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.36  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.53  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.69  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.05  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.17  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.39  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.43  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.16  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.09  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.16  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.34  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.41  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.62  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.06  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.16  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.79  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.96  , .05); sramp(Solarflux,  0  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.27  , .05); sramp(Solarflux,  0  , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  38.3   , .05); sramp(Solarflux,  0         , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.19  , .05); sramp(Solarflux,  0.0072639 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.41  , .05); sramp(Solarflux,  0.0045332 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.55  , .05); sramp(Solarflux,  0.0013654 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.03  , .05); sramp(Solarflux,  0.80126   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.86  , .05); sramp(Solarflux,  0.75119   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.71  , .05); sramp(Solarflux,  0.82152   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.71  , .05); sramp(Solarflux,  0.0065388 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.63  , .05); sramp(Solarflux,  0.83016   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.77  , .05); sramp(Solarflux,  0.83261   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.15  , .05); sramp(Solarflux,  0.83613   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.6   , .05); sramp(Solarflux,  0.83658   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.97  , .05); sramp(Solarflux,  0.84061   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.44  , .05); sramp(Solarflux,  0.84193   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.82  , .05); sramp(Solarflux,  0.84532   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.71  , .05); sramp(Solarflux,  0.84915   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.75  , .05); sramp(Solarflux,  0.84961   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.04  , .05); sramp(Solarflux,  0.85049   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.25  , .05); sramp(Solarflux,  0.85604   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.88  , .05); sramp(Solarflux,  0.85474   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.18  , .05); sramp(Solarflux,  0.85844   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.28  , .05); sramp(Solarflux,  0.85809   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.75  , .05); sramp(Solarflux,  0.86304   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.49  , .05); sramp(Solarflux,  0.86845   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.03  , .05); sramp(Solarflux,  0.87069   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.85  , .05); sramp(Solarflux,  0.87433   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.63  , .05); sramp(Solarflux,  0.8769    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.65  , .05); sramp(Solarflux,  0.88186   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.73  , .05); sramp(Solarflux,  0.88399   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.69  , .05); sramp(Solarflux,  0.8855    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.02  , .05); sramp(Solarflux,  0.8885    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.19  , .05); sramp(Solarflux,  0.88808   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.25  , .05); sramp(Solarflux,  0.88738   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.33  , .05); sramp(Solarflux,  0.89045   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.35  , .05); sramp(Solarflux,  0.88927   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.97  , .05); sramp(Solarflux,  0.89201   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.41  , .05); sramp(Solarflux,  0.8965    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.47  , .05); sramp(Solarflux,  0.90193   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.89  , .05); sramp(Solarflux,  0.90034   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.8   , .05); sramp(Solarflux,  0.90245   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.45  , .05); sramp(Solarflux,  0.90818   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.52  , .05); sramp(Solarflux,  0.90502   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.3   , .05); sramp(Solarflux,  0.8806    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.09  , .05); sramp(Solarflux,  0.90304   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.96  , .05); sramp(Solarflux,  0.90987   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.41  , .05); sramp(Solarflux,  0.91719   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.61  , .05); sramp(Solarflux,  0.91784   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.31  , .05); sramp(Solarflux,  0.91922   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.2   , .05); sramp(Solarflux,  0.92064   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40     , .05); sramp(Solarflux,  0.92238   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.83  , .05); sramp(Solarflux,  0.92553   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.66  , .05); sramp(Solarflux,  0.92698   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.64  , .05); sramp(Solarflux,  0.92499   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.95  , .05); sramp(Solarflux,  0.92783   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.68  , .05); sramp(Solarflux,  0.92877   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.58  , .05); sramp(Solarflux,  0.92975   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.38  , .05); sramp(Solarflux,  0.92631   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.06  , .05); sramp(Solarflux,  0.92828   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.03  , .05); sramp(Solarflux,  0.92727   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.09  , .05); sramp(Solarflux,  0.92988   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.66  , .05); sramp(Solarflux,  0.93011   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.8   , .05); sramp(Solarflux,  0.93118   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.34  , .05); sramp(Solarflux,  0.93106   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.01  , .05); sramp(Solarflux,  0.92966   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.12  , .05); sramp(Solarflux,  0.92946   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.83  , .05); sramp(Solarflux,  0.93306   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.82  , .05); sramp(Solarflux,  0.93343   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.67  , .05); sramp(Solarflux,  0.93344   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.14  , .05); sramp(Solarflux,  0.92733   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.98  , .05); sramp(Solarflux,  0.93037   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.92  , .05); sramp(Solarflux,  0.9312    , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  42.28  , .05); sramp(Solarflux,  0.93248  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.08  , .05); sramp(Solarflux,  0.92616  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.08  , .05); sramp(Solarflux,  0.93185  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.83  , .05); sramp(Solarflux,  0.93227  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.59  , .05); sramp(Solarflux,  0.93253  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.38  , .05); sramp(Solarflux,  0.93284  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.14  , .05); sramp(Solarflux,  0.93186  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.49  , .05); sramp(Solarflux,  0.93301  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.37  , .05); sramp(Solarflux,  0.93282  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.41  , .05); sramp(Solarflux,  0.93312  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.68  , .05); sramp(Solarflux,  0.93295  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.97  , .05); sramp(Solarflux,  0.93259  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.64  , .05); sramp(Solarflux,  0.92891  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.06  , .05); sramp(Solarflux,  0.92575  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.78  , .05); sramp(Solarflux,  0.92378  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.98  , .05); sramp(Solarflux,  0.92887  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.12  , .05); sramp(Solarflux,  0.92275  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.87  , .05); sramp(Solarflux,  0.9197   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.1   , .05); sramp(Solarflux,  0.85546  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.86  , .05); sramp(Solarflux,  0.9109   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.08  , .05); sramp(Solarflux,  0.92145  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.65  , .05); sramp(Solarflux,  0.54003  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.59  , .05); sramp(Solarflux,  0.48322  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.86  , .05); sramp(Solarflux,  0.33894  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43     , .05); sramp(Solarflux,  0.9308   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.34  , .05); sramp(Solarflux,  0.93363  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.53  , .05); sramp(Solarflux,  0.9371   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.64  , .05); sramp(Solarflux,  0.93329  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.8   , .05); sramp(Solarflux,  0.93317  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.62  , .05); sramp(Solarflux,  0.93449  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.37  , .05); sramp(Solarflux,  0.93608  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.94  , .05); sramp(Solarflux,  0.92978  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.25  , .05); sramp(Solarflux,  0.93517  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.74  , .05); sramp(Solarflux,  0.93668  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.97  , .05); sramp(Solarflux,  0.9299   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.92  , .05); sramp(Solarflux,  0.93844  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.7   , .05); sramp(Solarflux,  0.93448  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.89  , .05); sramp(Solarflux,  0.92501  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.88  , .05); sramp(Solarflux,  0.91927  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.04  , .05); sramp(Solarflux,  0.92796  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.44  , .05); sramp(Solarflux,  0.76074  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.83  , .05); sramp(Solarflux,  0.92572  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.34  , .05); sramp(Solarflux,  0.91783  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.23  , .05); sramp(Solarflux,  0.92071  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.64  , .05); sramp(Solarflux,  0.92405  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.37  , .05); sramp(Solarflux,  0.9141   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.14  , .05); sramp(Solarflux,  0.91887  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.12  , .05); sramp(Solarflux,  0.92181  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.32  , .05); sramp(Solarflux,  0.92141  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.16  , .05); sramp(Solarflux,  0.91602  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.95  , .05); sramp(Solarflux,  0.916    , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.21  , .05); sramp(Solarflux,  0.92286  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.06  , .05); sramp(Solarflux,  0.92316  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.85  , .05); sramp(Solarflux,  0.92505  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.17  , .05); sramp(Solarflux,  0.91765  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.5   , .05); sramp(Solarflux,  0.68174  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.55  , .05); sramp(Solarflux,  0.88523  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.17  , .05); sramp(Solarflux,  0.91829  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.53  , .05); sramp(Solarflux,  0.85217  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.81  , .05); sramp(Solarflux,  0.90271  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.2   , .05); sramp(Solarflux,  0.9018   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.22  , .05); sramp(Solarflux,  0.89966  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.71  , .05); sramp(Solarflux,  0.43571  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.84  , .05); sramp(Solarflux,  0.90335  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.14  , .05); sramp(Solarflux,  0.89616  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.37  , .05); sramp(Solarflux,  0.90328  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.59  , .05); sramp(Solarflux,  0.90076  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.78  , .05); sramp(Solarflux,  0.89077  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.54  , .05); sramp(Solarflux,  0.88795  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.58  , .05); sramp(Solarflux,  0.88366  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  47.59  , .05); sramp(Solarflux,  0.87574  , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  45.22  , .05); sramp(Solarflux,  0.87848  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.33  , .05); sramp(Solarflux,  0.87299  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.22  , .05); sramp(Solarflux,  0.86763  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.15  , .05); sramp(Solarflux,  0.86844  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.66  , .05); sramp(Solarflux,  0.86232  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.04  , .05); sramp(Solarflux,  0.84821  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.22  , .05); sramp(Solarflux,  0.86009  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.35  , .05); sramp(Solarflux,  0.86058  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.99  , .05); sramp(Solarflux,  0.87344  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.33  , .05); sramp(Solarflux,  0.86912  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.71  , .05); sramp(Solarflux,  0.86426  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.8   , .05); sramp(Solarflux,  0.86346  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.64  , .05); sramp(Solarflux,  0.85467  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.58  , .05); sramp(Solarflux,  0.85197  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.82  , .05); sramp(Solarflux,  0.85736  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.8   , .05); sramp(Solarflux,  0.84659  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.92  , .05); sramp(Solarflux,  0.84386  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.95  , .05); sramp(Solarflux,  0.83982  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.44  , .05); sramp(Solarflux,  0.82978  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.3   , .05); sramp(Solarflux,  0.82889  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.74  , .05); sramp(Solarflux,  0.82238  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.93  , .05); sramp(Solarflux,  0.82119  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.21  , .05); sramp(Solarflux,  0.81568  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.58  , .05); sramp(Solarflux,  0.80579  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.1   , .05); sramp(Solarflux,  0.81464  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.73  , .05); sramp(Solarflux,  0.81595  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.69  , .05); sramp(Solarflux,  0.81466  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.94  , .05); sramp(Solarflux,  0.80784  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.21  , .05); sramp(Solarflux,  0.81103  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.02  , .05); sramp(Solarflux,  0.80557  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.84  , .05); sramp(Solarflux,  0.79725  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.43  , .05); sramp(Solarflux,  0.79783  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.24  , .05); sramp(Solarflux,  0.79395  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.51  , .05); sramp(Solarflux,  0.78652  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.08  , .05); sramp(Solarflux,  0.7837   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.27  , .05); sramp(Solarflux,  0.78682  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.75  , .05); sramp(Solarflux,  0.78236  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.42  , .05); sramp(Solarflux,  0.7797   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.92  , .05); sramp(Solarflux,  0.77855  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.01  , .05); sramp(Solarflux,  0.77936  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.7   , .05); sramp(Solarflux,  0.7748   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.56  , .05); sramp(Solarflux,  0.76045  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.6   , .05); sramp(Solarflux,  0.7553   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.72  , .05); sramp(Solarflux,  0.68595  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.5   , .05); sramp(Solarflux,  0.66614  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.07  , .05); sramp(Solarflux,  0.72277  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.27  , .05); sramp(Solarflux,  0.63815  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.91  , .05); sramp(Solarflux,  0.72743  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.3   , .05); sramp(Solarflux,  0.72224  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.65  , .05); sramp(Solarflux,  0.70653  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.98  , .05); sramp(Solarflux,  0.70498  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.9   , .05); sramp(Solarflux,  0.65445  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.42  , .05); sramp(Solarflux,  0.69497  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.35  , .05); sramp(Solarflux,  0.62853  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.83  , .05); sramp(Solarflux,  0.67585  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  46.21  , .05); sramp(Solarflux,  0.61495  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.65  , .05); sramp(Solarflux,  0.66968  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.92  , .05); sramp(Solarflux,  0.65599  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.45  , .05); sramp(Solarflux,  0.63622  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.95  , .05); sramp(Solarflux,  0.63406  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.4   , .05); sramp(Solarflux,  0.62708  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.52  , .05); sramp(Solarflux,  0.61609  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.75  , .05); sramp(Solarflux,  0.60865  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.99  , .05); sramp(Solarflux,  0.59173  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.32  , .05); sramp(Solarflux,  0.58035  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.34  , .05); sramp(Solarflux,  0.56481  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.49  , .05); sramp(Solarflux,  0.56415  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.61  , .05); sramp(Solarflux,  0.5326   , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.58  , .05); sramp(Solarflux,  0.52842  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.32  , .05); sramp(Solarflux,  0.47053  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  45.31  , .05); sramp(Solarflux,  0.46703  , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,  44.94  , .05); sramp(Solarflux,  0.44562  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.87  , .05); sramp(Solarflux,  0.43552  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.73  , .05); sramp(Solarflux,  0.41488  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.7   , .05); sramp(Solarflux,  0.41068  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.78  , .05); sramp(Solarflux,  0.39287  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.67  , .05); sramp(Solarflux,  0.36816  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  44.37  , .05); sramp(Solarflux,  0.36689  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.25  , .05); sramp(Solarflux,  0.33285  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.74  , .05); sramp(Solarflux,  0.30884  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.77  , .05); sramp(Solarflux,  0.24365  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.13  , .05); sramp(Solarflux,  0.02912  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  43.3   , .05); sramp(Solarflux,  0.21069  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.97  , .05); sramp(Solarflux,  0.20209  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.53  , .05); sramp(Solarflux,  0.13229  , .05); endparallel;
parallel sramp(streams("AIR1").T ,  42.27  , .05); sramp(Solarflux,  0.0053879 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.88  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.61  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.42  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.3   , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.25  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.1   , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  41.03  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.92  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.9   , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.78  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.7   , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.64  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.56  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.47  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.39  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.35  , .05); sramp(Solarflux,  0.00002032 , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.29  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.24  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.16  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.1   , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40.06  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  40     , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.98  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.93  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.84  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.74  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.71  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.62  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.56  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.53  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.46  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.45  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.37  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.29  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.21  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.11  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39.03  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  39     , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.97  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.86  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.8   , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.76  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.78  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.76  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.68  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.58  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.56  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.5   , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.37  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.25  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  38.12  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.98  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.83  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.68  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.57  , .05); sramp(Solarflux,  0          , .05); endparallel;
parallel sramp(streams("AIR1").T ,  37.45  , .05); sramp(Solarflux,  0          , .05); endparallel;
```

```
parallel sramp(streams("AIR1").T ,      37.34     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      37.25     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      37.2      , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      37.17     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      37.13     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      37.06     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.99     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.9      , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.81     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.73     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.69     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.68     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.64     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.63     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.66     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.68     , .05); sramp(Solarflux,      0.00017274    , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.65     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.59     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.51     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.43     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.38     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.36     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.33     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.3      , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.26     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.23     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.2      , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.18     , .05); sramp(Solarflux,      0.00000677    , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.13     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.09     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.04     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36.02     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.98     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.96     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.98     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.96     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.96     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      36        , .05); sramp(Solarflux,      0.00031843    , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.98     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.94     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.87     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.84     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.81     , .05); sramp(Solarflux,      0.00032521    , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.82     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.76     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.73     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.77     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.79     , .05); sramp(Solarflux,      0             , .05); endparallel;
parallel sramp(streams("AIR1").T ,      35.82     , .05); sramp(Solarflux,      0.00026423    , .05); endparallel;
End
```

Appendix D: Parametric Performance of Scenario A Hybrids

I. PREHEAT MODEL[1]

Table D.1: Preheat model lookup chart for parametric study.

AMBIENT TEMPERATURE (°C) \ SOLAR INSOLATION (W/m2)	0	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900	950	1000
5	1161	1191	1222	1253	1285	1317	1349	1382	1415	1448	1481	1515	1549	1583	1618	1653	1689	1724	1760	1797	1833
10	1161	1191	1222	1253	1285	1317	1349	1382	1415	1448	1481	1515	1549	1583	1618	1653	1689	1724	1760	1797	1833
15	1151	1181	1212	1243	1274	1306	1338	1371	1403	1436	1470	1503	1537	1571	1606	1640	1676	1711	1747	1783	1819
20	1139	1169	1200	1231	1262	1293	1325	1357	1389	1422	1455	1488	1522	1556	1590	1625	1659	1695	1730	1766	1802
25	1125	1155	1185	1215	1246	1277	1308	1340	1372	1405	1437	1470	1503	1537	1571	1605	1639	1674	1709	1744	1780
30	1106	1136	1165	1195	1226	1256	1287	1319	1350	1382	1414	1447	1479	1512	1546	1579	1611	1642	1674	1705	1736
35	1081	1111	1140	1170	1199	1229	1260	1291	1321	1353	1382	1412	1442	1472	1502	1532	1563	1593	1624	1654	1685
40	1049	1077	1106	1134	1162	1190	1218	1246	1275	1302	1330	1359	1387	1415	1443	1472	1501	1529	1558	1587	1616
45	1000	1026	1053	1079	1106	1132	1159	1186	1214	1241	1268	1296	1323	1351	1378	1406	1434	1462	1490	1518	1546
50	946	971	997	1023	1049	1075	1101	1127	1153	1180	1206	1233	1260	1287	1314	1341	1368	1395	1422	1449	1477
55	892	917	942	967	992	1017	1043	1068	1094	1119	1145	1171	1197	1224	1250	1276	1302	1329	1356	1382	1408
60	836	863	887	912	936	961	985	1010	1035	1060	1085	1110	1135	1161	1186	1212	1238	1263	1289	1309	1341
65	783	810	833	857	881	904	928	952	976	1001	1025	1050	1074	1099	1124	1149	1173	1199	1223	1249	1274
70	735	757	780	803	826	849	872	895	919	942	966	990	1014	1038	1062	1086	1110	1135	1159	1183	1208
75	683	705	727	749	771	794	816	839	862	885	907	931	954	977	1000	1024	1047	1071	1095	1119	1142
80	631	654	675	696	718	740	761	783	805	827	850	872	895	917	940	963	986	1009	1032	1055	1078
85	583	603	624	644	665	686	707	728	750	771	793	814	836	858	880	902	924	947	969	992	1014
90	533	553	573	593	613	633	654	674	695	716	737	757	779	800	821	843	864	886	907	929	951
95	478	497	516	535	555	574	593	613	633	653	673	693	714	734	754	775	796	817	837	858	879
100	430	448	466	485	503	522	541	560	579	599	618	637	657	677	697	717	737	757	777	797	818
105	382	400	417	435	453	471	489	508	526	545	564	582	601	620	640	659	678	698	717	737	757

[1]Table represents new power produced by model in kW

II. PREHEAT-RECIRCULATION MODEL[1]

Table D.2: Preheat-Recirculation model lookup chart for parametric study.

AMBIENT TEMPERATURE (°C)	SOLAR INSOLATION (W/m2)																				
	0	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900	950	1000
5	1161	1214	1269	1324	1379	1435	1324	1375	1426	1479	1532	1586	1641	1696	1752	1809	1867	1925	1984	2043	2103
10	1161	1214	1269	1324	1379	1435	1314	1365	1417	1469	1522	1576	1630	1685	1741	1797	1854	1912	1971	2030	2090
15	1151	1204	1258	1313	1368	1423	1304	1354	1405	1457	1510	1563	1617	1672	1727	1783	1840	1897	1955	2014	2073
20	1139	1191	1246	1300	1355	1409	1290	1340	1391	1442	1495	1548	1601	1655	1710	1766	1822	1879	1937	1995	2053
25	1125	1176	1230	1284	1338	1392	1274	1323	1374	1425	1476	1528	1581	1635	1689	1744	1800	1856	1913	1970	2028
30	1106	1157	1210	1263	1317	1370	1253	1302	1352	1402	1453	1504	1556	1609	1662	1717	1771	1821	1869	1918	1966
35	1081	1132	1184	1236	1288	1340	1226	1274	1322	1371	1421	1468	1516	1563	1609	1656	1703	1750	1797	1844	1892
40	1049	1098	1146	1195	1243	1292	1184	1229	1273	1317	1362	1407	1452	1497	1543	1588	1634	1679	1725	1771	1818
45	1000	1045	1092	1138	1184	1230	1129	1171	1214	1257	1301	1344	1388	1432	1476	1521	1565	1610	1654	1699	1744
50	946	990	1035	1080	1125	1170	1072	1114	1155	1197	1239	1282	1325	1368	1410	1454	1497	1541	1584	1628	1672
55	892	935	979	1022	1066	1109	1016	1057	1097	1138	1179	1220	1262	1304	1345	1388	1430	1472	1514	1557	1600
60	836	880	923	965	1008	1050	961	1000	1040	1079	1119	1159	1200	1240	1281	1322	1363	1404	1446	1487	1528
65	783	826	867	909	950	992	906	944	982	1021	1060	1099	1138	1177	1217	1257	1297	1337	1377	1418	1458
70	735	774	814	853	893	933	852	889	926	963	1001	1039	1077	1115	1154	1193	1232	1271	1310	1349	1388
75	683	721	760	798	837	876	798	834	870	907	943	980	1017	1054	1091	1129	1157	1195	1232	1270	1308
80	631	669	706	744	781	819	745	780	815	850	886	921	957	994	1030	1066	1103	1140	1177	1214	1251
85	583	618	654	690	727	763	693	726	760	794	829	864	898	934	969	1004	1040	1076	1112	1148	1184
90	533	567	602	637	672	707	641	674	706	739	773	807	840	874	909	943	978	1013	1048	1083	1118
95	478	511	544	578	612	645	590	621	653	685	717	750	783	816	849	883	916	950	984	1018	1052
100	430	461	494	526	559	591	539	570	601	632	663	694	726	758	790	823	855	888	921	954	987
105	382	413	444	475	506	538	490	519	549	579	609	639	670	701	733	764	796	827	859	891	924

[1]Table represents new power produced by model in kW

III. REHEAT MODEL[1]

Table D.3: Reheat model lookup chart for parametric study.

AMBIENT TEMPERATURE (°C) \ SOLAR INSOLATION (W/m2)	0	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900	950	1000
5	0	0	0	0	0	0	0	0	0	0	796	871	949	1030	1115	1202	1292	1386	1481	1579	1680
10	0	0	0	0	0	0	0	0	0	0	790	864	942	1023	1107	1194	1284	1376	1472	1569	1670
15	0	0	0	0	0	0	0	0	0	0	783	857	934	1014	1098	1184	1273	1366	1460	1557	1657
20	0	0	0	0	0	0	0	0	0	0	774	847	924	1004	1087	1172	1261	1352	1446	1543	1642
25	0	0	0	0	0	0	0	0	0	0	763	836	912	991	1073	1158	1246	1336	1429	1525	1623
30	0	0	0	0	0	0	0	0	0	0	750	822	897	975	1056	1140	1226	1316	1407	1502	1598
35	0	0	0	0	0	0	0	0	0	0	733	803	877	954	1034	1116	1201	1289	1379	1467	1555
40	0	0	0	0	0	0	0	0	0	0	709	778	850	925	1003	1083	1166	1248	1330	1412	1495
45	0	0	0	0	0	0	0	0	0	0	682	749	818	891	965	1040	1117	1195	1274	1354	1435
50	0	0	0	0	0	0	0	0	0	0	644	711	780	850	922	996	1071	1147	1224	1302	1381
55	0	0	0	0	0	0	0	0	0	0	609	674	741	810	880	952	1025	1099	1174	1250	1327
60	0	0	0	0	0	0	0	0	0	0	574	638	703	769	838	908	978	1050	1124	1198	1272
65	0	0	0	0	0	0	0	0	0	0	539	601	664	729	796	863	932	1003	1074	1146	1218
70	0	0	0	0	0	0	0	0	0	0	504	564	626	689	753	819	886	955	1024	1094	1165
75	0	0	0	0	0	0	0	0	0	0	469	528	587	649	711	775	841	907	974	1042	1111
80	0	0	0	0	0	0	0	0	0	0	435	491	549	608	669	731	795	859	911	991	1058
85	0	0	0	0	0	0	0	0	0	0	400	454	511	568	627	688	749	812	861	940	989
90	0	0	0	0	0	0	0	0	0	0	365	418	472	528	586	644	704	765	812	889	936
95	0	0	0	0	0	0	0	0	0	0	330	382	434	489	544	601	659	718	764	823	884
100	0	0	0	0	0	0	0	0	0	0	296	345	396	449	503	558	614	671	716	773	832
105	0	0	0	0	0	0	0	0	0	0	261	309	359	409	462	515	569	625	667	724	780

[1]Table represents new power produced by model in kW

Appendix E: Detailed Results from Scenario B

I. Dynamic Model Charts

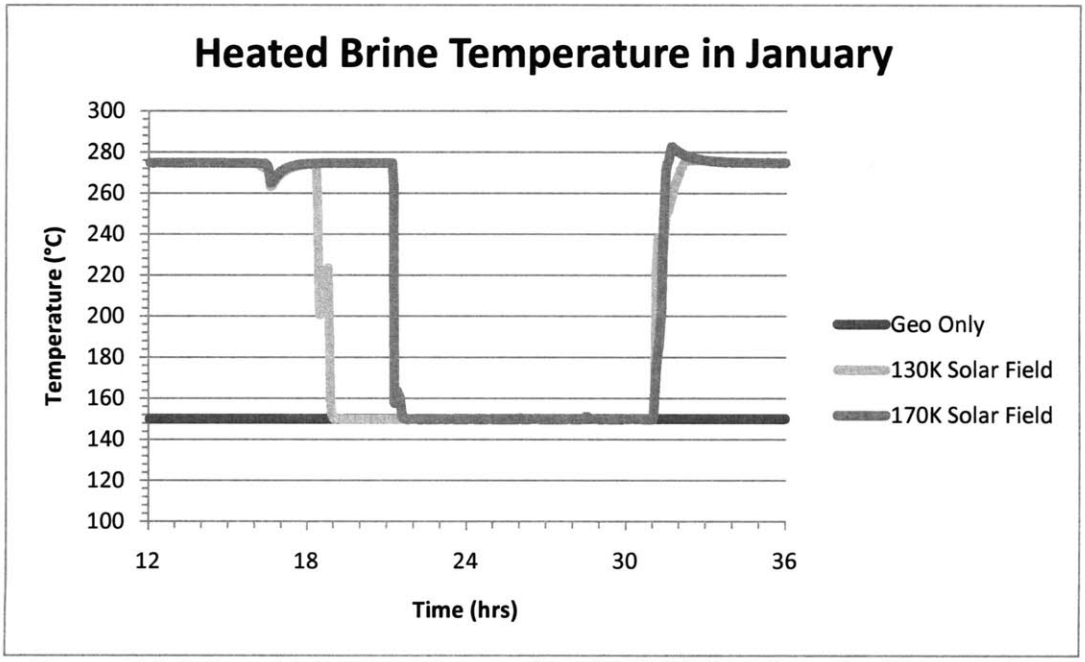


Figure E.0.1 Heated brine temperature of 3 solar field sizes for typical January day.

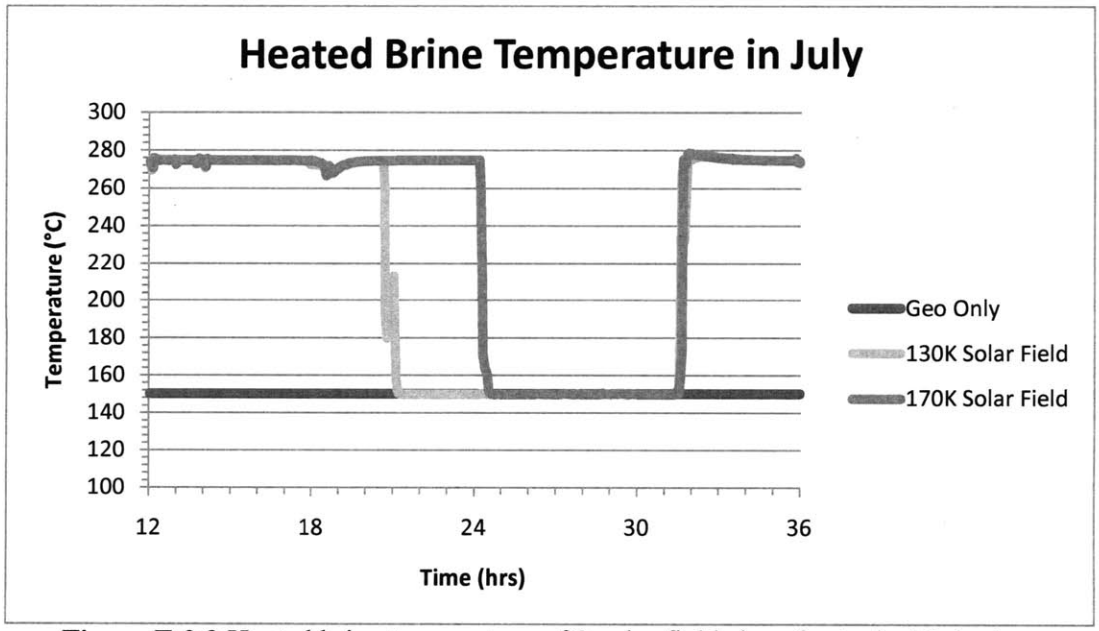


Figure E.0.2 Heated brine temperature of 3 solar field sizes for typical July day.

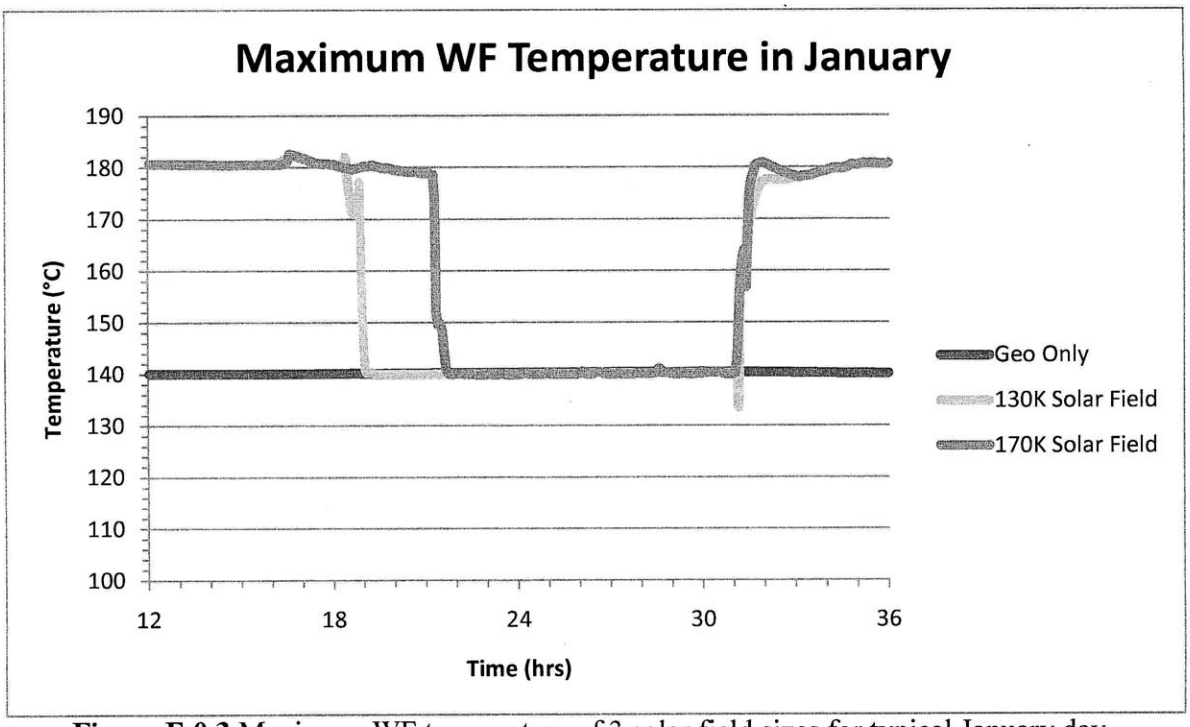


Figure E.0.3 Maximum WF temperature of 3 solar field sizes for typical January day.

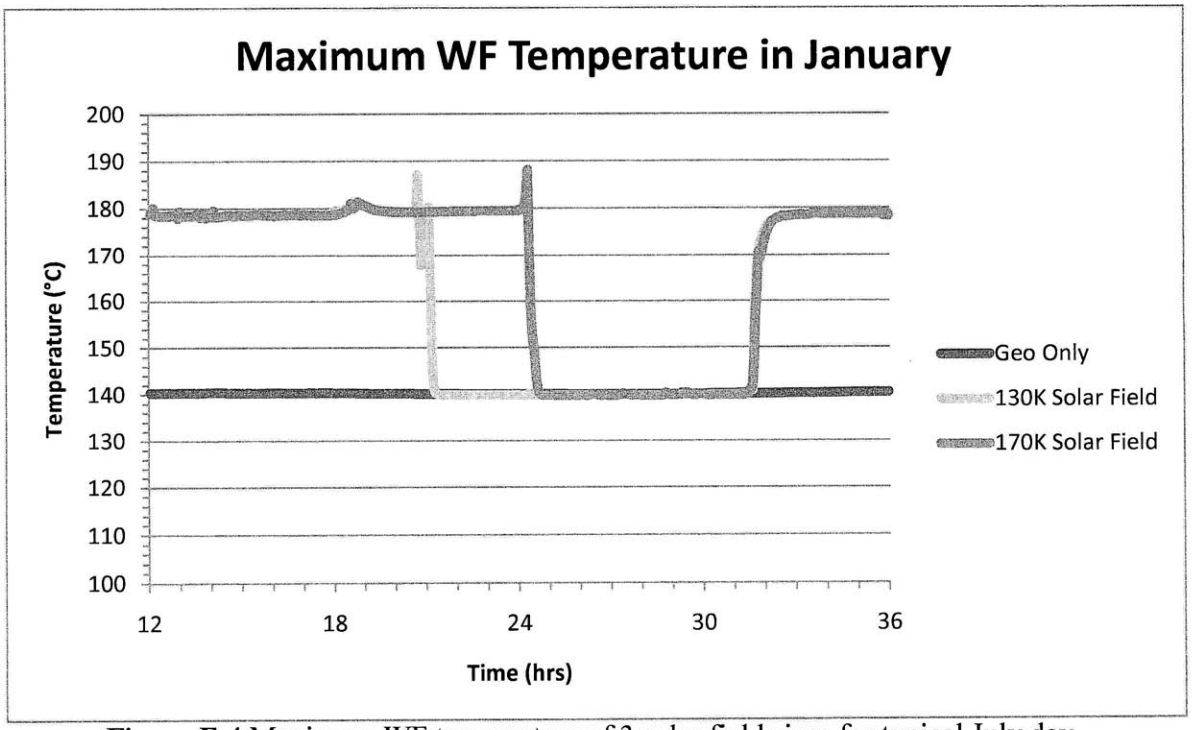


Figure E.4 Maximum WF temperature of 3 solar field sizes for typical July day.

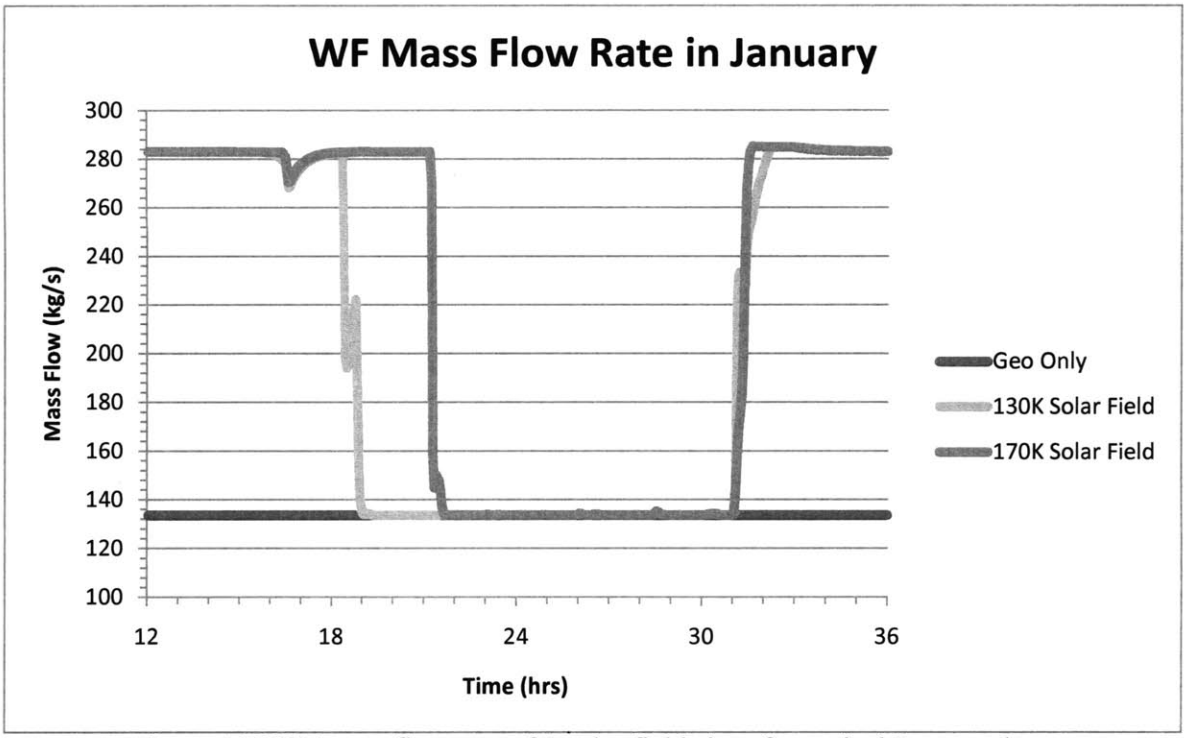


Figure E.5 WF mass flow rate of 3 solar field sizes for typical January day.

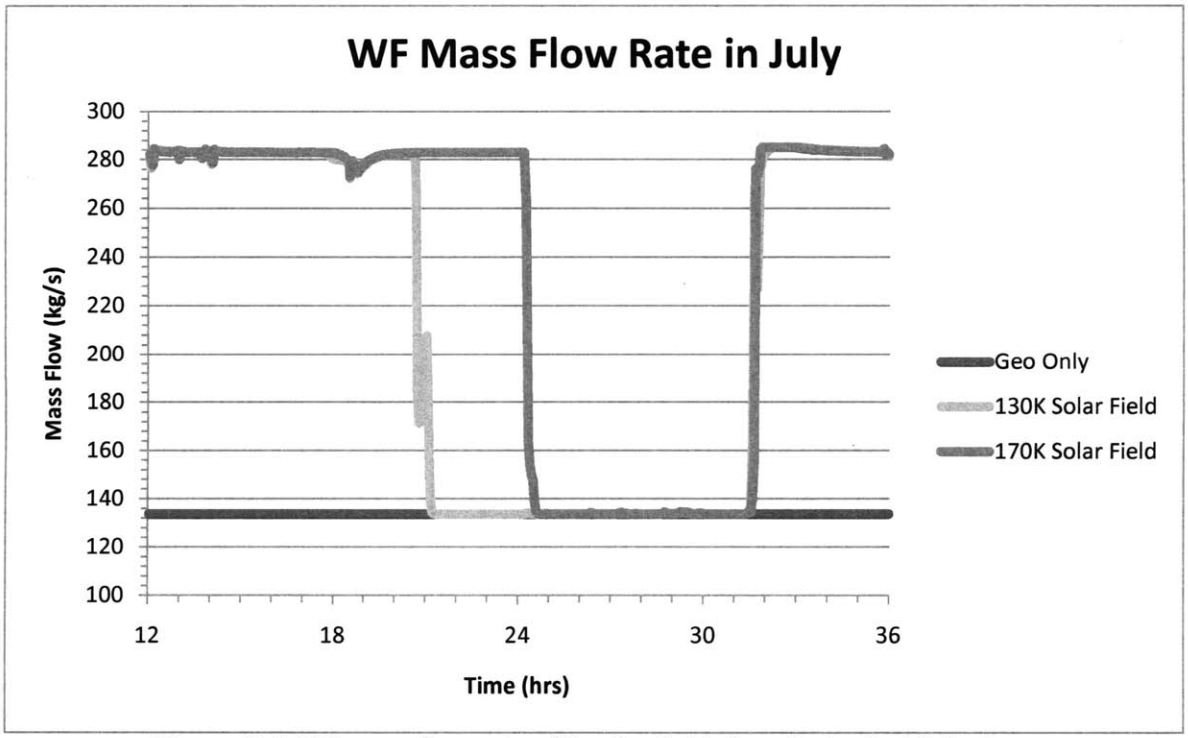


Figure E.6 WF mass flow rate of 3 solar field sizes for typical July day.

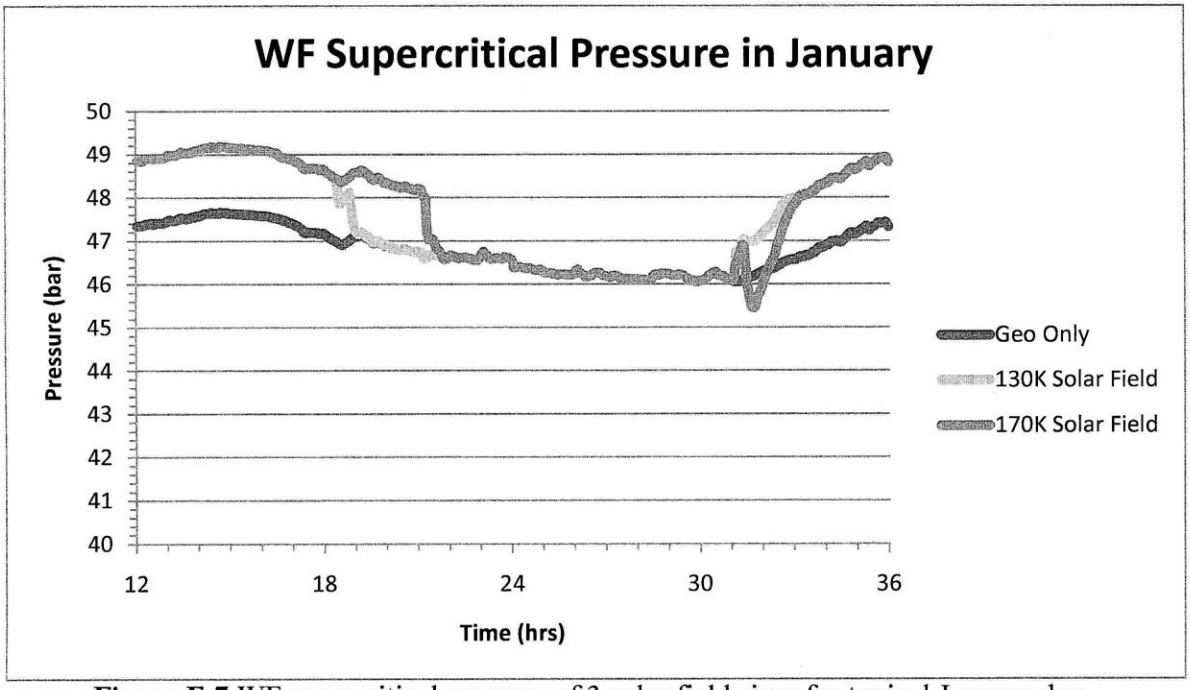


Figure E.7 WF supercritical pressure of 3 solar field sizes for typical January day.

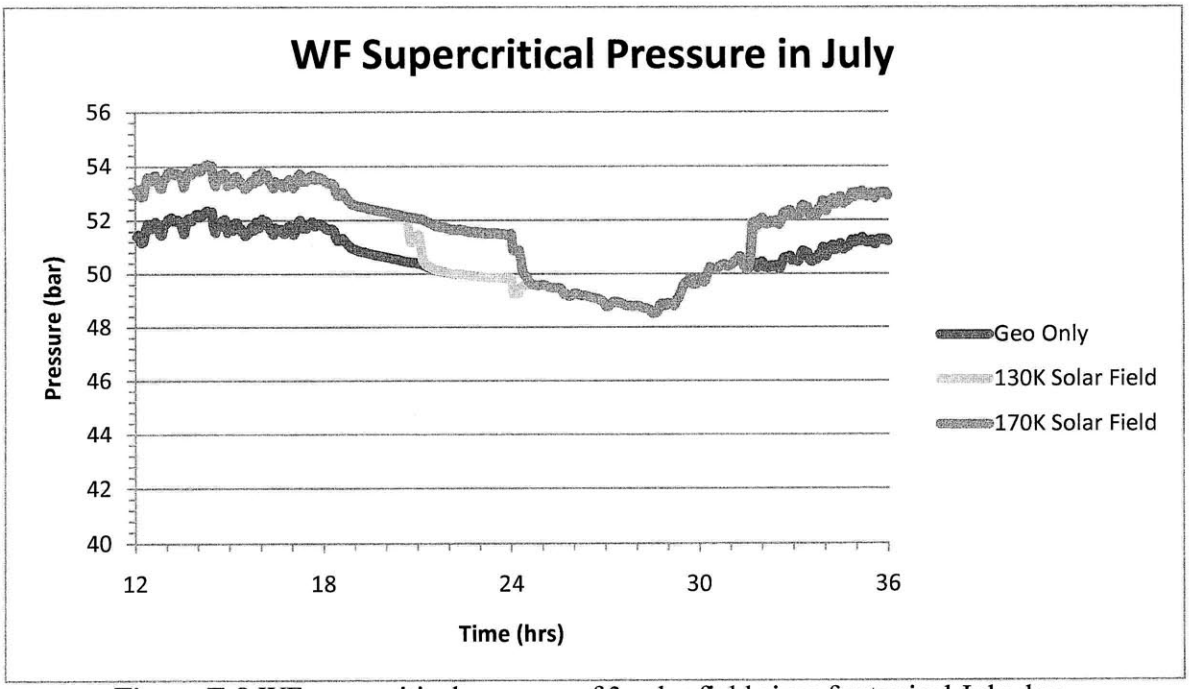


Figure E.8 WF supercritical pressure of 3 solar field sizes for typical July day.

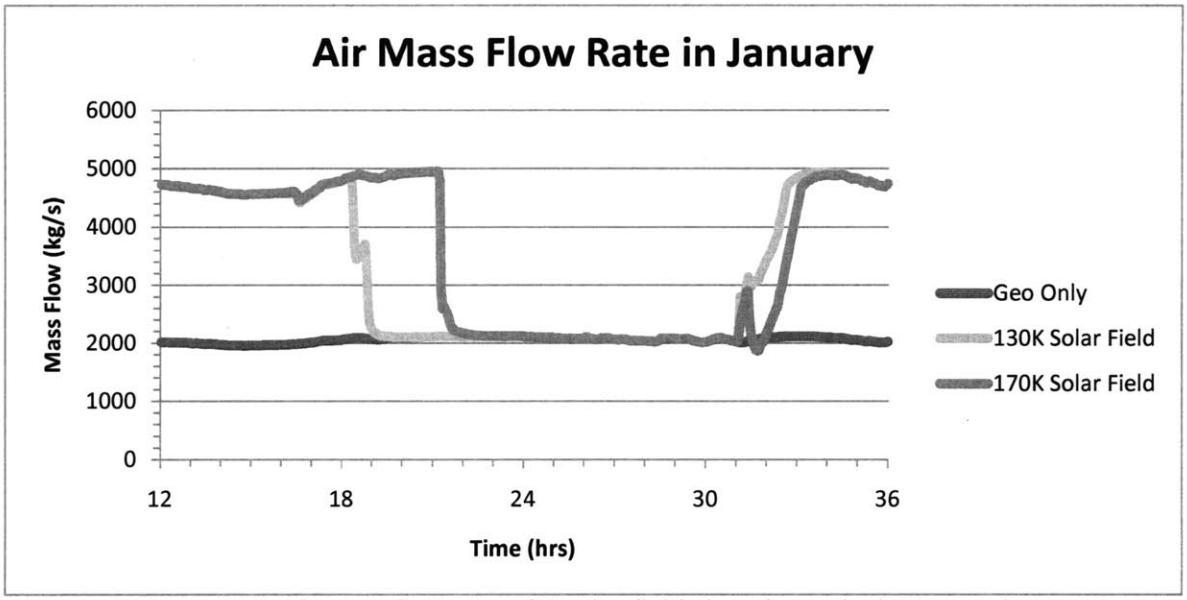


Figure E.9 Air mass flow rate of 3 solar field sizes for typical January day.

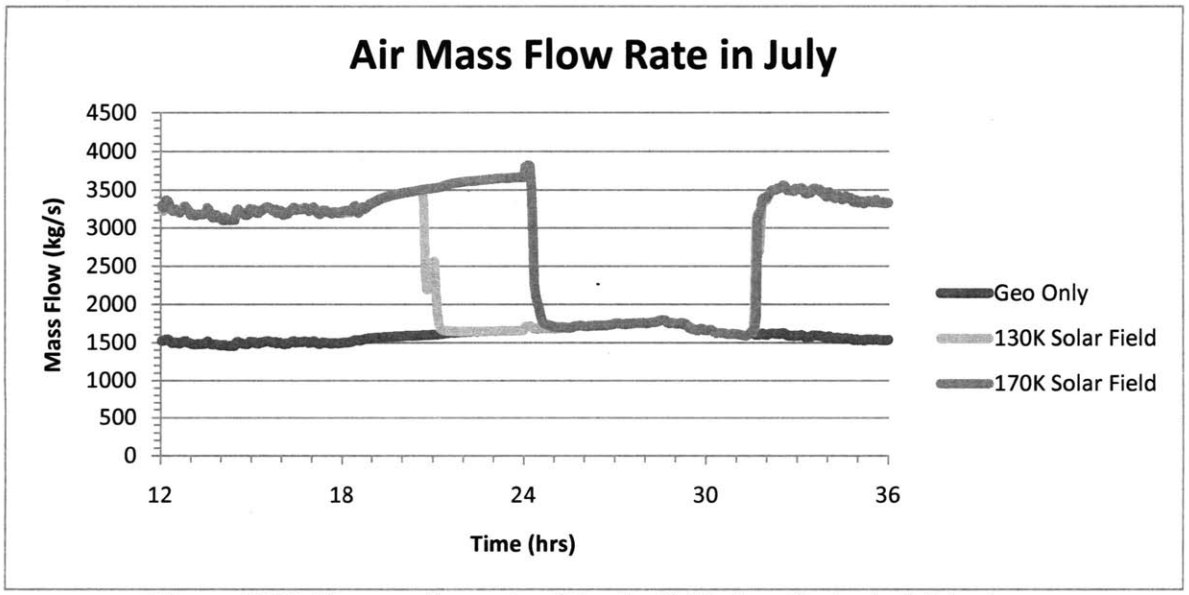


Figure E.10 Air mass flow rate of 3 solar field sizes for typical July day.

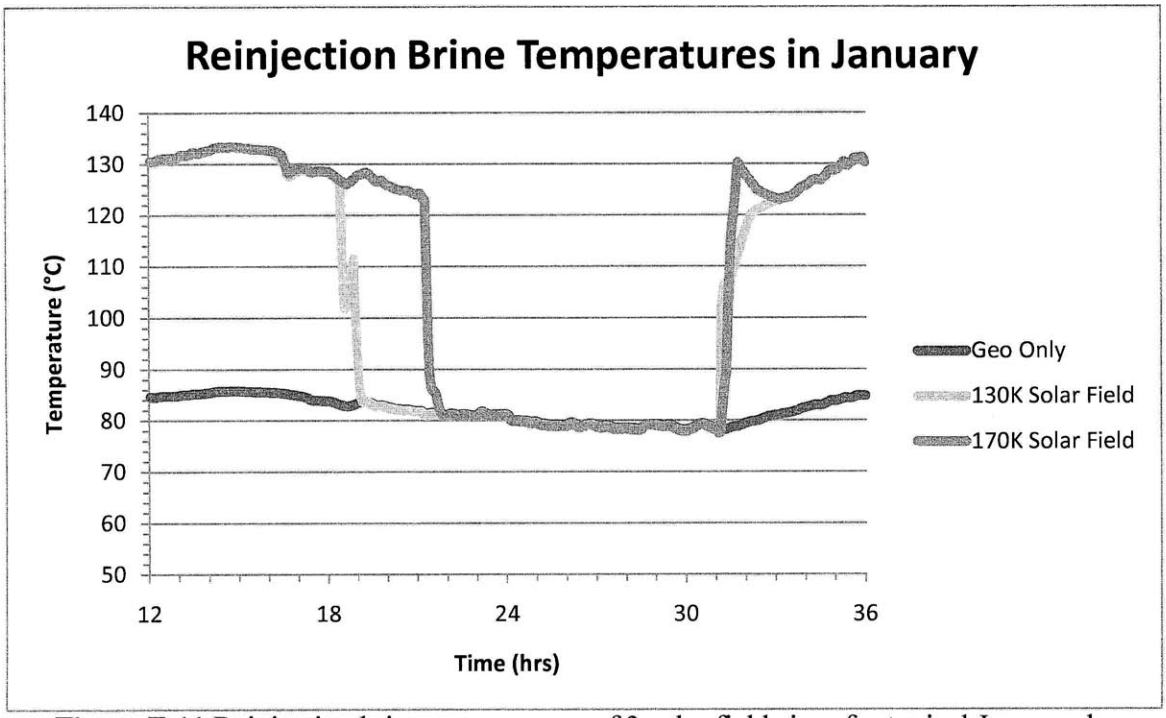


Figure E.11 Reinjection brine temperatures of 3 solar field sizes for typical January day.

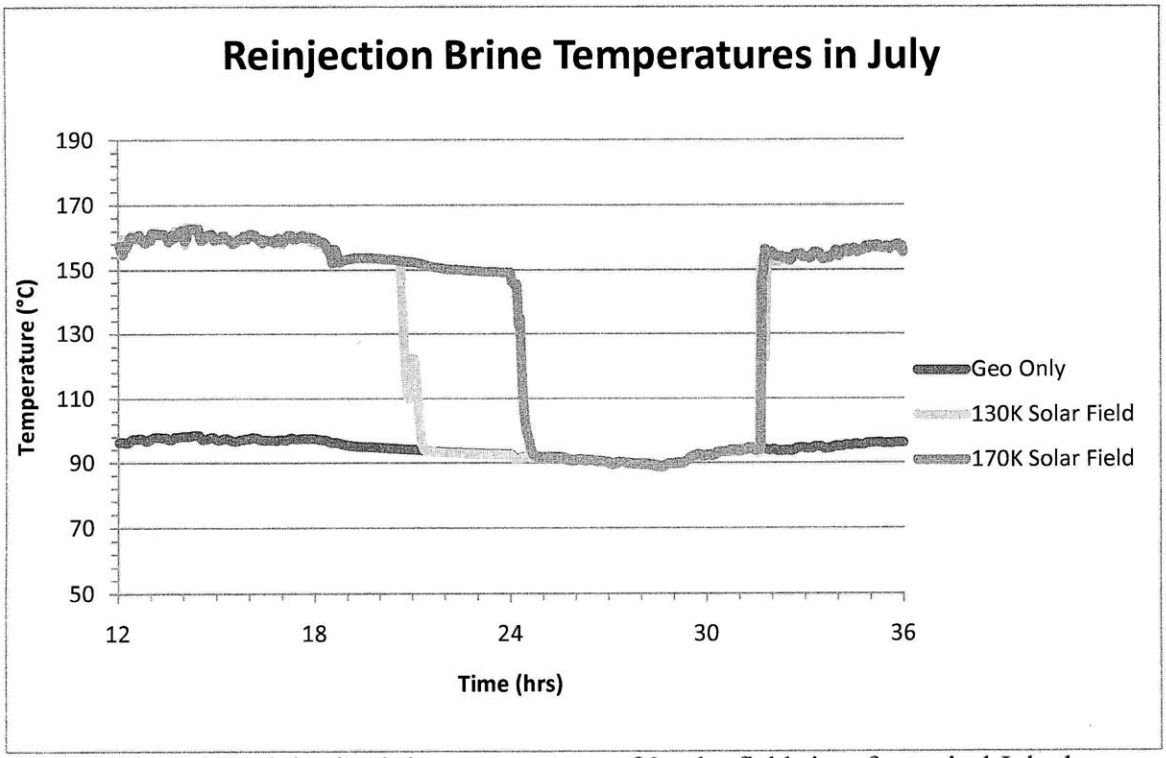


Figure E.12 Reinjection brine temperatures of 3 solar field sizes for typical July day.

II. Detailed Heat Exchanger Designs

BHX

Heat Exchanger Specification Sheet

Size	533.4 / 657.6 mm		Type	BEM	Hor	Connected in	4	parallel	1	series	
Surf/unit(eff.)	272.6 m2		Shells/unit	4		Surf/shell (eff.)		68.2		m2	
PERFORMANCE OF ONE UNIT											
Fluid allocation				**Shell Side**				**Tube Side**			
Fluid name											
Fluid quantity, Total			kg/s	300.0001				100			
Vapor (In/Out)			kg/s	0	0			0	0		
Liquid			kg/s	300.0001	300.0001			100	100		
Noncondensable			kg/s	0				0			
Temperature (In/Out)			C	390	317.85			150.69	275		
Dew / Bubble point			C								
Density (Vap / Liq)			kg/m3	/ 703.48	/ 792.9			/ 919.55	/ 759.39		
Viscosity			mPa*s	/ 0.149	/ 0.206			/ 0.1829	/ 0.0956		
Molecular wt, Vap											
Molecular wt, NC											
Specific heat			kJ/(kg*K)	/ 2.829	/ 2.479			/ 4.296	/ 5.188		
Thermal conductivity			W/(m*K)	/ 0.0779	/ 0.093			/ 0.6849	/ 0.5888		
Latent heat			kJ/kg								
Pressure			bar	10.04637	9.84737			60.43125	60.41517		
Velocity			m/s	1.95				0.52			
Pressure drop, allow./calc.			bar	0.2	0.199			0.2	0.01608		
Fouling resist. (min)			m2*K/W	0				0	0	Ao based	
Heat exchanged	57134.4	kW					MTD corrected		138.38	C	
Transfer rate, Service	1514.3		Dirty	1570.5		Clean	1570.5			W/(m2*K)	
CONSTRUCTION OF ONE SHELL								**Sketch**			
			Shell Side			**Tube Side**					
Design/Vac/Test pressu		bar	1.7210 / /			6.8791 / /					
Design temperature		C	410			310					
Number passes per shell			1			1					
Corrosion allowance		mm	3.18			3.18					
Connections	In	mm	1 304.8 / -			1 203.2 / -					
Size/rating	Out		1 304.8 / -			1 152.4 / -					
Nominal	Intermediate		/ -			/ -					
Tube No. 327	OD 19.05	Tks- Avg 1.65	mm	Lengtl 3657.6 mm		Pitch 23.81	mm				
Tube type	Plain	Material	Carbon Steel	Tube pattern	30						
Shell	Carbon Steel	ID 539.75	OD 558.8	mm	Shell cover	-					
Channel or bonnet	Carbon Steel				Channel cover	-					
Tubesheet-stationary	Carbon Steel				Tubesheet-floating	-					
Floating head cover	-				Impingement protection	None					
Baffle-crossing	Carbon Steel	Type	Single segme	Cut(%d)	40.45 H	Spacing: c/c	590.55	mm			
Baffle-long	-	Seal type				Inlet	855.66	mm			
Supports-tube		U-bend		Type							
Bypass seal		Tube-tubesheet joint		Exp.							
Expansion joint	-	Type									
RhoV2-Inlet nozzle	1502	Bundle entrance	858	Bundle exit	784	kg/(m*s2					
Gaskets - Shell side	Flat Metal Jacket Fibe	Tube Side	Flat Metal Jacket Fibe								
Floating head	-										
Code requirements	ASME Code Sec VIII Div 1			TEMA class	R - refinery service						
Weight/Shell	2765.4	Filled with water	3672.6	Bundle	1301.3	kg					
Remarks											

Figure E.13 Brine-solar heat exchanger specification sheet.

Air-Cooled Heat Exchanger Specification Sheet

Size & Type	9.744	/ 147.7437	m	Type	Forced		Number of Bays		30		
Surf/Unit-Finned Tube		153309.1	m2	Bare area/bundle	108.8	m2	Area ratio		23.50		
Heat exchanged		51732.5	kW			MTD, Eff	10.14	C			
Transfer Rate-Finned	34.3	Bare, Service		781.6	Clean		806.2	W/(m2*K)			
PERFORMANCE DATA - TUBE SIDE											
Fluid Circulated							In/Out				
Total Fluid Entering	kg/s		285.008	Density, Liq	kg/m3		1155.9	/	1184.73		
		In/Out		Density, Vap	kg/m3		41.31	/	42.46		
Temperature	C	40.16 /	32.12	Specific Heat, Liq	kJ/(kg*K)		1.477	/	1.438		
Liquid	kg/s	0 /	285.008	Specific Heat, Vap	kJ/(kg*K)		1.009	/	1.018		
Vapor	kg/s	285.008 /	0	Therm. Cond, Liq	W/(m*K)		0.0833	/	0.0874		
Noncondensable	lb/h	/		Therm. Cond, Vap	W/(m*K)		0.0156	/	0.0149		
Steam	lb/h	/		Freeze Point	C						
Water	lb/h	/		Bubble / Dew point	C		34.24	/	34.24		
Molecular wt, Vap		102.03 /	102.03	Latent heat	BTU/lb						
Molecular wt, NC				Inlet pressure (abs)	bar		8.85207				
Viscosity, Liq	mPa*s	0.1457 /	0.1595	Pres Drop, Allow/Calc			0.2	/	0.18175		
Viscosity, Vap		0.0125 /	0.0122	Fouling resistance	m2*K/W						
PERFORMANCE DATA - AIR SIDE											
Air Quantity, Total		6855.078	kg/s	Altitude					m		
Air Quantity/Fan		96.127	m3/s	Temperature In			20		C		
Static Pressure		214	Pa	Temperature Out			27.52		C		
Face Velocity	4.63	m/s	Bundle velocity	5.5	kg/s/m2	Design Ambient	0		C		
DESIGN-MATERIALS-CONSTRUCTION											
Design pressure	13.78951	bar	Test Pressure			Design temperature			148.89	C	
TUBE BUNDLE			**Header**			**Tube**					
Size	m	9.744	Type	Box		Material			Carbon Steel		
Number/bay	2		Material	Carbon Steel		Specifications					
Tube Rows	4		Passes	3		OD	25.4	Min Thk.	1.65	mm	
Arrangement			Plug Mat.			No./Bur	152	Lng	9.144	m	
Bundles	2	par	Gasket Mat.			Pitch	60 /	51.96	30	deg	
Bays	30	par	Corr. Allow.		mm	**Fin**					
Bundle frame			Inlet Nozzle	1	146.33 mm	Type			G-finned		
MISCELLANEOUS			Outlet nozzle	1	53.98 mm	Material			Aluminum 1060		
Struct. Mount.			Special Nozzles			OD	57.15	Tks	0.28	mm	
Surf.Prep			Rating			No.	433 #/m	DesTemp		C	
Louvers			TI		PI	Code					
Vibration Switches			Chem Cleaning			Stamp		Specs			
MECHANICAL EQUIPMENT											
Fan,Mfr., Model			Driver, Type			Speed Reducer, Type					
No./Bay	2	RPM		Mfr.		Mfr.&Model					
Dia.	3.3528 m	Blade(s)		No./Bay		No./Bay					
Pitch		Angle		RPM		Rating					
Blade(s)		Hub		Enclosure		Ratio					
hp/Fan	35.684 kW	MinAmb		V/Phase/Hz		Support					
Control Action on Air Failure-						Louvers					
Degree Control of Outlet Process Temperature											
Recirculation						Steam Coil					
Plot Area	m2	Drawing No.		Wt.Bundle	4570.9	Wt. Unit	274252.5 kg				
Notes:											

Figure E.14 Air cooled condenser heat exchanger specification sheet.

Heat Exchanger Specification Sheet

Size	990.6 / 486.4	mm	Type	BEM	Hor	Connected in	2	parallel	1	series		
Surf/unit(eff.)	794.7	m2	Shells/unit	2		Surf/shell (eff.)		397.3		m2		
PERFORMANCE OF ONE UNIT												
Fluid allocation				**Shell Side**				**Tube Side**				
Fluid name												
Fluid quantity, Total		kg/s		18.7632				285.008				
Vapor (In/Out)		kg/s		17.2894		0		0		0		
Liquid		kg/s		1.4739		18.7632		285.008		285.008		
Noncondensable		kg/s		0				0				
Temperature (In/Out)		C		133.56		133.55		83.89		124.4		
Dew / Bubble point		C				133.56						
Density (Vap / Liq)		kg/m3		1.65 / 931.84		1.55 / 931.8		/ 955.37		/ 281.76		
Viscosity		mPa*s		0.0134 / 0.2067		.013 / 0.207		/ 0.0936		/ 0.026		
Molecular wt, Vap				18.02		18.02						
Molecular wt, NC												
Specific heat		kJ/(kg*K)		2.195 / 4.275		2.195 / 4.275		/ 2.02		/ 2.145		
Thermal conductivity		W/(m*K)		0.0292 / 0.6828		.029 / 0.683		/ 0.0667		/ 0.0321		
Latent heat		kJ/kg		2163.6		2163.6						
Pressure		bar		3		2.81737		48.93965		48.74222		
Velocity		m/s		35.63				4.12				
Pressure drop, allow./calc.		bar		0.2		0.18263		0.2		0.19743		
Fouling resist. (min)		m2*K/W		0				0	0	Ao based		
Heat exchanged	37408.3	kW						MTD corrected		24.42	C	
Transfer rate, Service	1927.8		Dirty	2120.6		Clean	2120.6			W/(m2*K)		
CONSTRUCTION OF ONE SHELL								**Sketch**				
			Shell Side			**Tube Side**						
Design/Vac/Test pressu		bar	3.44738 / /			4.4686 / /						
Design temperature		C	171.11			171.11						
Number passes per shell			1			2						
Corrosion allowance		mm	3.18			3.18						
Connections	In	mm	1	558.8 / -		1	406.4 / -					
Size/rating	Out		1	88.9 / -		1	406.4 / -					
Nominal	Intermediate		/ -			/ -						
Tube No.	1262	OD	19.05	Tks- Avg	1.65	mm	Lengtl 5486.4	mm	Pitch	23.81	mm	
Tube type	Plain			Material		Carbon Steel		Tube pattern	30			
Shell	Carbon Steel	ID	990.6	OD	1012.82	mm	Shell cover	-				
Channel or bonnet		Carbon Steel					Channel cover	-				
Tubesheet-stationary		Carbon Steel					Tubesheet-floating	-				
Floating head cover		-					Impingement protection		None			
Baffle-crossing	Carbon Steel		Type	Single segme		Cut(%d)	39.75 V	Spacing: c/c	647.7	mm		
Baffle-long	-			Seal type				Inlet	687.39	mm		
Supports-tube				U-bend			Type					
Bypass seal				Tube-tubesheet joint			Exp.					
Expansion joint	-			Type								
RhoV2-Inlet nozzle	984		Bundle entrance		1806		Bundle exit	55	kg/(m*s2			
Gaskets - Shell side	Flat Metal Jacket Fibe		Tube Side		Flat Metal Jacket Fibe							
Floating head	-											
Code requirements	ASME Code Sec VIII Div 1					TEMA class		R - refinery service				
Weight/Shell	11549.6		Filled with water		16319.7			Bundle	6534.9	kg		
Remarks												

Figure E.15 HX1 heat exchanger specification sheet.

Heat Exchanger Specification Sheet

Size	1549.4 / 6096	mm	Type	BEM	Hor	Connected in	6	parallel	1	series		
Surf/unit(eff.)	7567	m2	Shells/unit	6		Surf/shell (eff.)	1261.2			m2		
PERFORMANCE OF ONE UNIT												
Fluid allocation			**Shell Side**				**Tube Side**					
Fluid name												
Fluid quantity, Total		kg/s	285.008				81.2368					
Vapor (In/Out)		kg/s	0	0			0	0				
Liquid		kg/s	285.008	285.008			81.2368	81.2368				
Noncondensable		kg/s	0				0					
Temperature (In/Out)		C	124.58	181.91			198.33	127.58				
Dew / Bubble point		C										
Density (Vap / Liq)		kg/m3	/ 278.18	/ 167.8			/ 866.69	/ 937.57				
Viscosity		mPa*s	/ 0.026	/ 0.029			/ 0.1355	/ 0.2175				
Molecular wt, Vap												
Molecular wt, NC												
Specific heat		kJ/(kg*K)	/ 2.114	/ 1.337			/ 4.481	/ 4.259				
Thermal conductivity		W/(m*K)	/ 0.0321	/ 0.039			/ 0.6638	/ 0.6835				
Latent heat		kJ/kg										
Pressure		bar	48.73965	48.60244			15	14.98511				
Velocity		m/s	2.2				0.02					
Pressure drop, allow./calc.		bar	0.2	0.13721			0.2	0.01489				
Fouling resist. (min)		m2*K/W	0				0	0	Ao based			
Heat exchanged	25016	kW				MTD corrected		10.03	C			
Transfer rate, Service	329.7		Dirty	331.1		Clean	331.1		W/(m2*K)			
CONSTRUCTION OF ONE SHELL									**Sketch**			
			Shell Side			**Tube Side**						
Design/Vac/Test pressu		bar	3.77913 /	/		6.54741 /	/					
Design temperature		C	237.78			237.78						
Number passes per shell			1			1						
Corrosion allowance		mm	3.18			3.18						
Connections	In	mm	1 304.8 /	-		1 152.4 /	-					
Size/rating	Out		1 304.8 /	-		1 88.9 /	-					
Nominal	Intermediate		/	-		/	-					
Tube No.	3578	OD	19.05	Tks- Avg	1.65	mm	Lengtl	6096	mm	Pitch	23.81	mm
Tube type	Plain		Material		Carbon Steel		Tube pattern	30				
Shell	Carbon Steel	ID	1549.4	OD	1631.95	mm	Shell cover	-				
Channel or bonnet		Carbon Steel					Channel cover	-				
Tubesheet-stationary		Carbon Steel					Tubesheet-floating	-				
Floating head cover		-					Impingement protection		None			
Baffle-crossing	Carbon Steel	Type	Single segme	Cut(%d)	20.05 H	Spacing: c/c	463.55	mm				
Baffle-long	-	Seal type				Inlet	858.84	mm				
Supports-tube		U-bend		Type								
Bypass seal		Tube-tubesheet joint		Exp.								
Expansion joint	-	Type										
RhoV2-Inlet nozzle	1523	Bundle entrance	350	Bundle exit	672	kg/(m*s2						
Gaskets - Shell side	Flat Metal Jacket Fibe	Tube Side	Flat Metal Jacket Fibe									
Floating head	-											
Code requirements	ASME Code Sec VIII Div 1			TEMA class	R - refinery service							
Weight/Shell	34140	Filled with water	44503.6	Bundle	19183	kg						
Remarks												

Figure E.16 HX2 heat exchanger specification sheet.

Heat Exchanger Specification Sheet

Size	889 / 1486.4	mm	Type	BEM	Hor	Connected in	6	parallel	3	series			
Surf/unit(eff.)	6228.3	m2	Shells/unit	18		Surf/shell (eff.)	346			m2			
PERFORMANCE OF ONE UNIT													
Fluid allocation				**Shell Side**				**Tube Side**					
Fluid name													
Fluid quantity, Total			kg/s	285.008				285.008					
Vapor (In/Out)			kg/s	0		0		285.008		285.008			
Liquid			kg/s	285.008		285.008		0		0			
Noncondensable			kg/s	0				0					
Temperature (In/Out)			C	34.24		84.51		117.9		39.23			
Dew / Bubble point			C										
Density (Vap / Liq)			kg/m3	/ 1198.3		/ 950.6		30.5 /		32.38 / 32.7			
Viscosity			mPa*s	/ 0.1671		/ 0.093		0.0158 /		0.0125 / 0.0239			
Molecular wt, Vap								102.03		102.03			
Molecular wt, NC													
Specific heat			kJ/(kg*K)	/ 1.398		/ 2.043		1.05 /		1.015 / 1.031			
Thermal conductivity			W/(m*K)	/ 0.0898		/ 0.066		0.0228 /		0.0155 / 0.0286			
Latent heat			kJ/kg					20.9		3000			
Pressure			bar	49.13965		48.97493		9.05207		8.85504			
Velocity			m/s	0.57				7.39					
Pressure drop, allow./calc.			bar	0.2		0.16473		0.2		0.19703			
Fouling resist. (min)			m2*K/W	0				0	0	Ao based			
Heat exchanged	22892.7	kW						MTD corrected		13.86	C		
Transfer rate, Service	265.1		Dirty	274.1		Clean	274.1			W/(m2*K)			
CONSTRUCTION OF ONE SHELL								**Sketch**					
			Shell Side			**Tube Side**							
Design/Vac/Test pressu		bar	5.46861 /	/		0.3421 /	/						
Design temperature		C	154.44			154.44							
Number passes per shell			1			1							
Corrosion allowance		mm	3.18			3.18							
Connections	In	mm	1	304.8 / -		1	508 / -						
Size/rating	Out		1	304.8 / -		1	508 / -						
Nominal	Intermediate		1	304.8 / -		1	508 / -						
Tube No.	1082	OD	19.05	Tks- Avg	1.65	mm	Lengtl	5486.4	mm	Pitch	23.81	mm	
Tube type	Plain			Material		Carbon Steel		Tube pattern	30				
Shell	Carbon Steel	ID	889	OD	939.8	mm	Shell cover	-					
Channel or bonnet		Carbon Steel					Channel cover	-					
Tubesheet-stationary		Carbon Steel					Tubesheet-floating	-					
Floating head cover		-					Impingement protection		None				
Baffle-crossing	Carbon Steel	Type	Single segme	Cut(%d)	34.92 H	Spacing: c/c	508	mm					
Baffle-long	-		Seal type				Inlet	893.76	mm				
Supports-tube			U-bend			Type							
Bypass seal			Tube-tubesheet joint			Exp.							
Expansion joint	-		Type										
RhoV2-Inlet nozzle	354	Bundle entrance	169			Bundle exit	181	kg/(m*s2					
Gaskets - Shell side	Flat Metal Jacket Fibe	Tube Side	Flat Metal Jacket Fibe										
Floating head	-												
Code requirements	ASME Code Sec VIII Div 1					TEMA class	R - refinery service						
Weight/Shell	10789.3	Filled with water	14833.1			Bundle	5118.6	kg					
Remarks													

Figure E.17 Recuperator heat exchanger specification sheet.